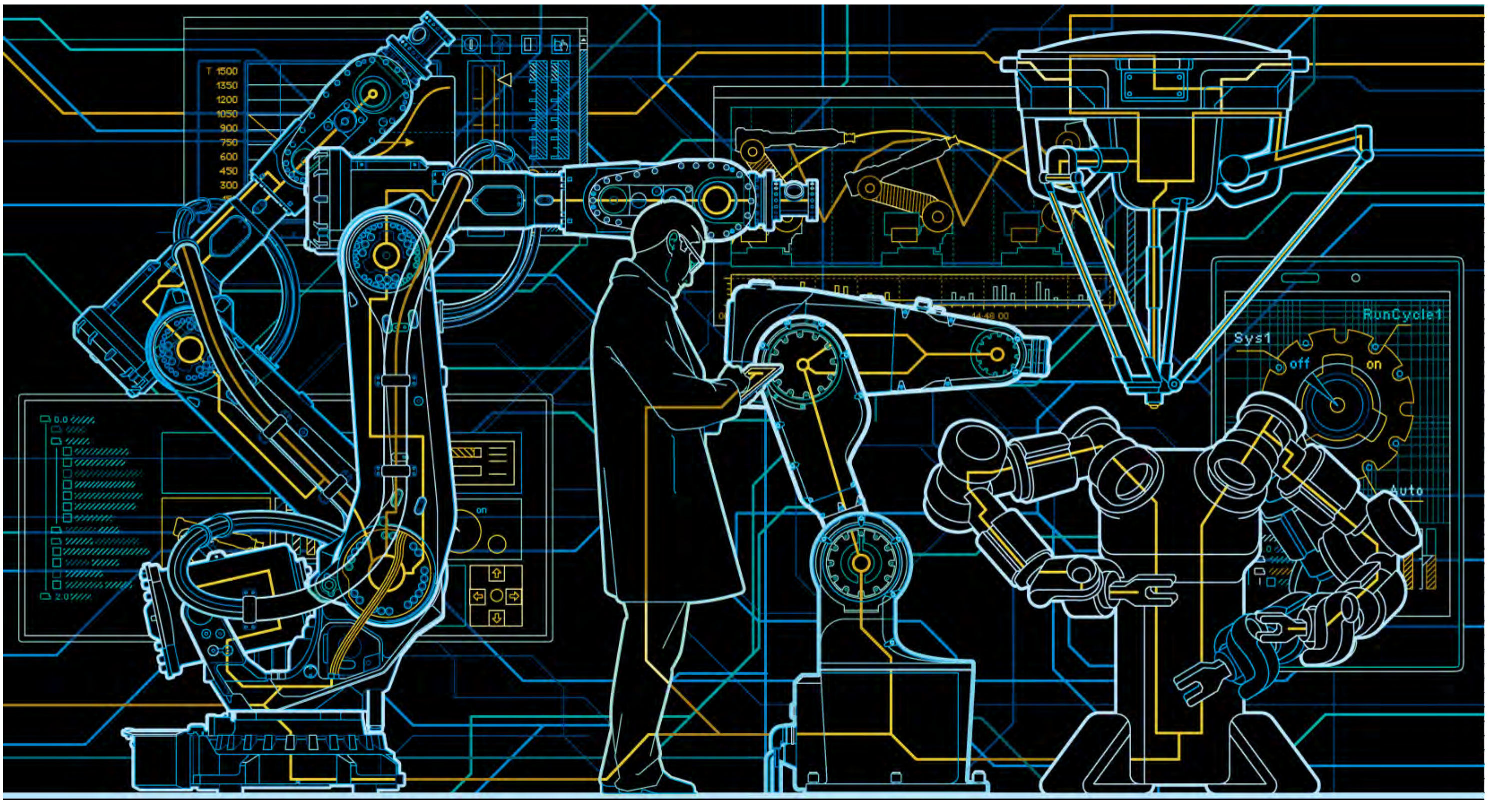



# Application manual
# SmarTac

Trace back information:
Workspace R16-1 version a11
Checked in 2016-03-29
Skribenta version 4.6.284

# Application manual

# SmarTac

Document ID: 3HAC024845-001

Revision: A

ABB AB
Robotics Products
Se-721 68 Västerås
Sweden

# Table of contents

# Overview of this manual

## About this manual

This manual explains the basics of when and how to use the option SmarTac:

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

## 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.

## Who should read this 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.

## Reference documents

| References | Document ID |
|---|---|
| *Technical reference manual - RAPID overview* | *3HAC050947-001* |
| *Technical reference manual - RAPID Instructions, Functions and Data types* | *3HAC050917-001* |
| *Operating manual - Getting started, IRC5 and RobotStudio* | *3HAC027097-001* |
| *Operating manual - IRC5 with FlexPendant* | *3HAC050941-001* |
| *Technical reference manual - System parameters* | *3HAC050948-001* |
| *Operating manual - RobotStudio* | *3HAC032104-001* |
| *Application manual - Production Manager* | *3HAC052855-001* |

## Revisions

| Revision | Description |
|---|---|
| - | Released with RobotWare 6.0. |
| A | Minor corrections. |

# Product documentation, IRC5

## Categories for user documentation from ABB Robotics

The user documentation from ABB Robotics is divided into a number of categories. This listing is based on the type of information in the documents, regardless of whether the products are standard or optional.

All documents listed can be ordered from ABB on a DVD. The documents listed are valid for IRC5 robot systems.

## Product manuals

Manipulators, controllers, DressPack/SpotPack, and most other hardware is delivered with a **Product manual** that generally contains:

- Safety information.
- Installation and commissioning (descriptions of mechanical installation or electrical connections).
- Maintenance (descriptions of all required preventive maintenance procedures including intervals and expected life time of parts).
- Repair (descriptions of all recommended repair procedures including spare parts).
- Calibration.
- Decommissioning.
- Reference information (safety standards, unit conversions, screw joints, lists of tools).
- Spare parts list with exploded views (or references to separate spare parts lists).
- Circuit diagrams (or references to circuit diagrams).

## Technical reference manuals

The technical reference manuals describe reference information for robotics products.

- *Technical reference manual - Lubrication in gearboxes*: Description of types and volumes of lubrication for the manipulator gearboxes.
- *Technical reference manual - RAPID overview*: An overview of the RAPID programming language.
- *Technical reference manual - RAPID Instructions, Functions and Data types*: Description and syntax for all RAPID instructions, functions, and data types.
- *Technical reference manual - RAPID kernel*: A formal description of the RAPID programming language.
- *Technical reference manual - System parameters*: Description of system parameters and configuration workflows.

*Continues on next page*

### Application manuals

Specific applications (for example software or hardware options) are described in **Application manuals**. An application manual can describe one or several applications.

An application manual generally contains information about:

- The purpose of the application (what it does and when it is useful).
- What is included (for example cables, I/O boards, RAPID instructions, system parameters, DVD with PC software).
- How to install included or required hardware.
- How to use the application.
- Examples of how to use the application.

### Operating manuals

The operating manuals describe hands-on handling of the products. The manuals are aimed at those having first-hand operational contact with the product, that is production cell operators, programmers, and trouble shooters.

The group of manuals includes (among others):

- *Operating manual - Emergency safety information*
- *Operating manual - General safety information*
- *Operating manual - Getting started, IRC5 and RobotStudio*
- *Operating manual - Introduction to RAPID*
- *Operating manual - IRC5 with FlexPendant*
- *Operating manual - RobotStudio*
- *Operating manual - Trouble shooting IRC5*, for the controller and manipulator.

# Safety

## Safety of personnel

When working inside the robot controller it is necessary to be aware of voltage-related risks.

A danger of high voltage is associated with the following parts:

- Devices inside the controller, for example I/O devices, can be supplied with power from an external source.
- The mains supply/mains switch.
- The power unit.
- The power supply unit for the computer system (230 VAC).
- The rectifier unit (400-480 VAC and 700 VDC). Capacitors!
- The drive unit (700 VDC).
- The service outlets (115/230 VAC).
- The power supply unit for tools, or special power supply units for the machining process.
- The external voltage connected to the controller remains live even when the robot is disconnected from the mains.
- Additional connections.

Therefore, it is important that all safety regulations are followed when doing mechanical and electrical installation work.

## Safety regulations

Before beginning mechanical and/or electrical installations, ensure you are familiar with the safety regulations described in *Operating manual - General safety information*[1].

1 This manual contains all safety instructions from the product manuals for the manipulators and the controllers.

# 1 Introduction

## 1.1 Product overview

### General

SmarTac™ is a tactile sensor used to find the location of inconsistent weld joints and offset the programmed points in a weld program.

### Main component

The main component is an electronic sensor board, which detects contact with the part feature to be located. The SmarTac™ board is supplied as an add-on unit and installed in the robot cabinet.

### RAPID system module

A RAPID system module, `SmarTac.sys`, provided by ABB supports powerful programming tools explained in *User's guide on page 33*.

### Searching with SmarTac

SmarTac searching can be added to programs while programming a part, or it can be added to a pre-existing weld routine.

## 1.2 Operation overview

**General**

With SmarTac™ a part feature may be "searched" using part of the torch. Typically the welding wire or the gas cup is used 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 torch feature (gas cup or wire) is energized with about 38 VDC. When the torch feature makes contact with the part (at ground potential) an input is set in the robot controller. When the input is detected, robot location is stored and motion stops.

**Search instructions**

The search instructions included in the SmarTac™ 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.

**Why use SmarTac?**

Using SmarTac™ effectively can dramatically reduce fixture costs. It can also help account for part variability that can not otherwise be controlled.

## 1.3 Prerequisites

### 1.3.1 System prerequisites

**Introduction**

This SmarTac version is intended for use in arc welding systems incorporating IRB 1400, 2400, etc. robots.

- RobotWare requirements: 5.06
- Controller requirements: IRC5

**SmarTac package**

The SmarTac package includes software that is loaded into all arc welding motion tasks, when the option is purchased.

Process configuration parameters are used to connect real I/O signals and to modify the default settings.

**Compatibility**

Programs with SmarTac searches written with versions of SmarTac prior to revision 6.0 are not compatible with SmarTac 6.0 and higher.

### 1.3.2 User's qualifications

**Robot programmer**

Any competent robot programmer (RAPID language) may be self-taught to program and use basic SmarTac searches. Some complex searching techniques are best reserved for those programmers that have attended an advanced programming class offered by ABB, unless the programmer has a solid mathematical background.

**Robot system operator**

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

# 2 Installation

## 2.1 Safety instructions

**WARNING**

The power supply must always be switched off whenever work is carried out in the control cabinet.

**CAUTION**

Circuit boards - printed circuit boards and components - must never be handled without electrostatic discharge (ESD) protection in order not to damage them. Use the wrist strap located on the inside of the controller door.

**WARNING**

All personnel working with the robot system must be very familiar with the safety regulations, see *Safety on page 33*. Incorrect operation can damage the robot or injure someone.

## 2.2 Hardware installation

**Component list**

The following items are supplied with the SmarTac option:

- SmarTac printed circuit board with mounting hardware and safety cover (optional).
- SmarTac software for RobotWare Arc
- Relevant electrical schematics (optional).

**Note**

The SmarTac option can be purchased as stand-alone, without any hardware.

**Required tools**

The following tools are required to install the SmarTac option:

- Terminal block screwdriver
- Multi-meter
- Wire cutters
- Wire strippers

**Installation**

See the documentation for your sensor for the correct hardware setup and diagnostics.

## 2.3 Software installation

### 2.3.1 About SmarTac software

**SmarTac package**

The SmarTac package includes one system module that is installed in each motion task that requires SmarTac functionality. The module, `SmarTac.sys`, is a standalone, read-only, no-step-in, module.

Consequently, it is compatible with any RAPID program, assuming the I/O configuration is non-conflicting, and no previous version of SmarTac is loaded.

**Compatibility**

Programs with SmarTac searches that are written with versions of SmarTac prior to revision 6.0 are not compatible with SmarTac 6.0 and higher.

### 2.3.2 System parameters

**I/O mapping**

Version 9.0 introduces a new fully configurable I/O mapping feature not available in previous SmarTac versions. SmarTac I/O connections are now configured in the process configuration database (PROC). Actual I/O assignments to real I/O boards are not made by the SmarTac installation. These definitions must be added to the EIO configuration database by the user or system designer.

The files `procSmarTacSet_X.cfg`, `procSmarTacSig_X.cfg`, and `procSmarTacSpd.cfg` are loaded by the SmarTac installation into the appropriate motion tasks.

**procSmarTacSig_X.cfg file**

The `procSmarTacSig_X.cfg` files load default I/O connections for up to 4 motion tasks, where the 'X' represents task numbers 1-4.

**procSmarTacSpd.cfg file**

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

**procSmarTacSet_X.cfg file**

The `procSmarTacSet_X.cfg` files load default references to SmarTac speed and signal configuration groupings included in the `procSmarTacSig_X.cfg`, and `procSmarTacSpd.cfg` files for up to 4 motion tasks, where the 'X' represents task numbers 1-4.

**Override defaults**

The user may override the defaults by replacing the entries with new entries. Below is the default file loaded by SmarTac:

```
PROC:CFG_1.0:5.0:
# Smartac procSmarTacSet_1.cfg file
# created 2005/09/22
#
SMARTAC_SETTINGS:
  # Structure created by SmarTac, defaults filled by SmarTac.
  # Cell Layer may overwrite with replace.
  -name "T_ROB1" -uses_signals "smtsig1" -uses_speeds "smtspeedstd"
PROC:CFG_1.0:5.0::
# Smartac procSmarTacSig_1.cfg file
# created 2005/09/22
SMARTAC_SIGNALS:
  # Structure created by SmarTac, defaults filled by SmarTac.
  # Cell Layer may overwrite with replace.
  -name "smtsig1" -detect_input "diSE_DET1" \ -reference_set_output
       "doSE_REF1" \ -wire_select_output "" \ -sensor_on_output
       "doSE_SENSOR1"
  # Use these when configuring for wire searching option...
```

*Continues on next page*

```
  -name "smtsigwire1" -detect_input "diSE_DET1" \
       -reference_set_output "doSE_REF1" \ -wire_select_output
       "doWIRE_SEL1" \ -sensor_on_output "doSE_SENSOR1"
PROC:CFG_1.0:5.0::
# Smartac procSmarTacSpd.cfg file
# created 2005/09/22

SMARTAC_SPEEDS:
  # Structure created by SmarTac, defaults filled by SmarTac.
  # Cell Layer may overwrite with replace.
  -name "smtspeedstd" -main_search_speed 20
  groove_search_speed 15
```

## Change settings

To change settings, the user must use the **Add or Replace** feature to override the existing fields with new settings. For example, a user could activate the wire search capability by using the pre-defined predefined wire search I/O set-up:

```
PROC:CFG_1.0:5.0:
# Example override Smartac proc.cfg file
# created 2005/09/22

#
SMARTAC_SETTINGS:
-name "T_ROB1" -uses_signals "smtsigwire1" -uses_speeds
     "smtspeedstd"
```

## Change default I/O names

To change the default I/O names, the user should supply a new I/O settings by creating new assignments:

```
PROC:CFG_1.0:5.0:
# Example override Smartac proc.cfg file
# for new signal names.
# created 2005/09/24

#
SMARTAC_SETTINGS:
  # Structure created by SmarTac, defaults filled by
  # SmarTac. Cell Layer may overwrite with replace.
  -name "T_ROB1" -uses_signals "mysmtsig" -uses_speeds "smtspeedstd"
#
SMARTAC_SIGNALS:
  -name "mysmtsig" -detect_input "diMyDetect" \
       -reference_set_output "doMyRef" \ -wire_select_output
       "doMyWireSel" \ -sensor_on_output "doMySensorOn"
```

*Continues on next page*

### Load I/O signals for default case

**Note**

SmarTac does not install any I/O signals in the EIO configuration. It provides only a mechanism to connect to existing signals in the system. If the robotic system is not a turnkey system, I/O signals will need to be installed in the system.

The following is an example of an I/O configuration file that can be used to load I/O signals as default.

```
EIO:CFG_1.0:5.0:
#
EIO_SIGNAL:

  -Name "diSE_DET1" -SignalType "DI" -Unit "Board_A"\ -SignalLabel
       "SmarTac Detect" -UnitMap 0
  -Name "doSE_REF1" -SignalType "DO" -Unit "Board_A"\ -SignalLabel
       "SmarTac Ref" -UnitMap 0
  -Name "doSE_SENSOR1" -SignalType "DO" -Unit "Board_A"\
       -SignalLabel "SmarTac Sensor" -UnitMap 1
  -Name "doWIRE_SEL1" -SignalType "DO" -Unit "Board_A"\ -SignalLabel
       "SmarTac Wire" -UnitMap 2
```

### Use SmarTac with Fronius TouchSense

Below is an example of a configuration that could be used to set up a SmarTac system with Fronius TouchSense. In this case only the software package for SmarTac is used, no SmarTac hardware should be used, instead the touch sensing capability of the Fronius welder is used.

```
EIO:CFG_1.0:5:0::
#
EIO_SIGNAL:
  -Name "doFr1SensorRef" -SignalType "DO" -Unit "ioFroniusSim1"
       -UnitMap "4"
  -Name "diFr1PartDetect" -SignalType "DI" -Unit "ioFroniusSim1"
       -UnitMap "4"
#
EIO_CROSS:
  -Res "diFr1PartDetect" -Act1 "doFr1SensorRef"
  -Act1_invert -Oper1 "OR" -Act2 "diFr1ArcStable"
PROC:CFG_1.0:5.0:
# Example using Fronius TouchSense for wire or gas nozzle sensing
#
SMARTAC_SIGNALS:
  -name "smtsig1" -detect_input "diFr1PartDetect"\
       -reference_set_output "doFr1SensorRef"\ -wire_select_output
       ""\ -sensor_on_output "doFr1TouchSense"
```

*Continues on next page*

### Use SmarTac with ESAB AristoMig touch sense

Below is an example of a configuration that could be used to set up a SmarTac system with ESAB touch sense. In this case only the software package for SmarTac is used, no SmarTac hardware should be used, instead the touch sensing capability of the AristoMig welder is used.

**Note**

Wire sensing is the default method for ESAB AristoMig. Contact your ESAB retailer on how to use the gas nozzle as sensor.

```
EIO:CFG_1.0:5:0::
#
EIO_SIGNAL:
-Name "doTouchSenseActive" -SignalType "DO" -Unit "B_AW_PROC_40"
     -UnitMap "15"
-Name "diTouchSenseContact" -SignalType "DI" -Unit "B_AW_PROC_40"
     -UnitMap "50"
PROC:CFG_1.0:5.0:

# Example using ESAB AristoMig touch sense for wire or gas sensing
     nozzle
#
SMARTAC_SIGNALS:
  -name "smtsig1" -detect_input "diTouchSenseContact"\
       -reference_set_output ""\ -wire_select_output ""\
       -sensor_on_output "doTouchSenseActive"
```

## 2.3.3 Loading software

### Loading software

The software is loaded automatically when the option SmarTac is purchased for RobotWare.

## 2.4 Start-up test

### Instruction

| | Action |
|---|---|
| 1 | Turn on the control cabinet power switch. |
| 2 | In the I/O window, turn on the output `doSE_SENSOR1` and make sure `doWIRE_SEL1` and `doSE_REF1` are off. If these I/O assignments do not exist, see *System parameters on page 18*. Verify that `diSE_DET1` is on at this time. If it is not, see *Trouble shooting on page 75*. |
| 3 | Turn on `doSE_REF1`. Verify that `diSE_DET1` is off at this time. If it is not, see *Trouble shooting on page 75*. |
| 4 | Using a voltmeter, verify that there is about 38 VDC at the gas cup when referenced to the fixture (ground). If less than 25 VDC is measured, see *Trouble shooting on page 75*. |
| 5 | Ground the gas cup to the weld fixture using a length of wire, steel tool, or similarly conductive object. Verify that `diSE_DET1` is on at this time. If it is not, see *Trouble shooting on page 75*. |
| 6 | Write a simple test routine using the `Search_1D` instruction (see *User's guide on page 33*).<br>If the `Search_1D` instruction is not available in the system, see *Loading software on page 22*. |
| 7 | Execute the test routine. The robot should stop when the part is detected. If not, see *Trouble shooting on page 75*. |

This page is intentionally left blank

# 3 Application guide

## 3.1 Searching conditions

### Introduction

SmarTac is intended for use in the following conditions:

- In applications where surfaces are free from rust, mill scale, paint, or other electrically-insulating layer or coating.
- If the gas nozzle is used for a search probe, it must be cleaned at regular intervals.
- If a water-cooled torch is used, the quality of the cooling water is very important. Impure water, for example containing salt solution, will act as a load that will reduce the sensitivity and/or reduce the sensing voltage below SmarTac working range. Distilled water or a non-conductive coolant such as ethylene glycol is recommended as gun coolant solution. Tap water is unacceptable.
- A positive lead break box (secondary contact) is required to isolate the power source when SmarTac sensing is taking place when using a water-cooled torch or when searching with the wire.

## 3.2 Programming limitations

### Searching with welding wire

In systems where searching with the welding wire is needed, a wire trimmer is necessary to ensure a known wire stick-out. A wire trimmer is a hardware device that requires extra I/O. This option may be purchased though ABB.

The use of searches ranges from very simplistic to very complex. In some instances, very complex searching techniques must be used to adequately determine weld seam locations. In such instances, assistance from an experienced ABB technician may be required.

## 3.3 SmarTac board characteristics

### 3.3.1 Interaction with the welding equipment and weldment

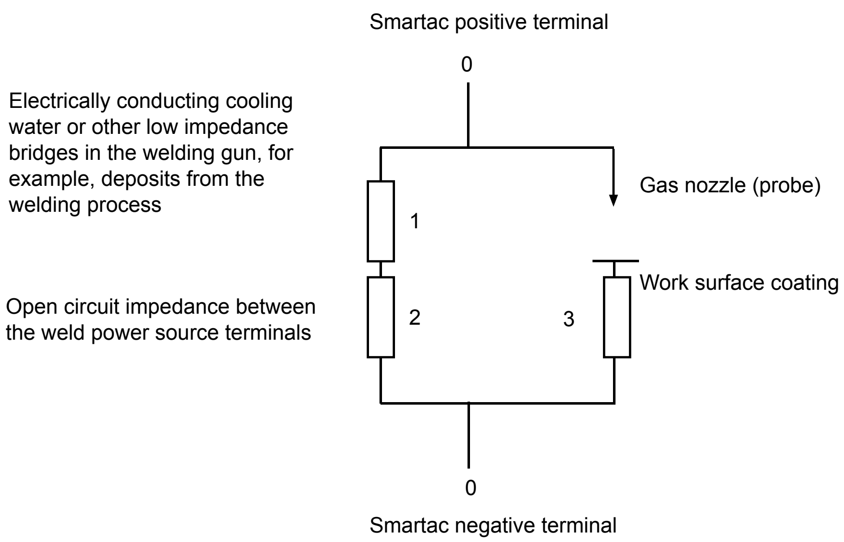
#### SmarTac board

The SmarTac board (alone) is capable of generating a stop signal from high-resistance surfaces with up to 1 Mom contact resistance. In real applications, however, the SmarTac sensing circuitry is normally loaded from the surrounding welding equipment. The electric equivalent diagram below explains the situation.

#### Sensitivity

Maximum sensitivity will only be obtained when using a separate, highly insulated probe (not supplied).

#### Electric equivalent diagram




xx1400001476

#### Convection-cooled welding gun

SmarTac used with a convection-cooled (air cooled) welding gun will loose some sensitivity due to losses through the weld equipment. In the circuit above, this loss is described by resistor #2.

#### Positive lead secondary contact

In systems with a positive lead secondary contact, the loss through the weld equipment is eliminated.

Continues on next page

### Water-cooled gun

When SmarTac is used with a water-cooled gun, the quality of the coolant becomes very important. Impure water, for example containing salt solution, acts as a conductor to ground potential, effectively reducing the sensitivity or even reduce the sensing voltage below SmarTac working range. In *Electric equivalent diagram on page 27*, this loss is represented by resistor #1. Non-conductive liquid such as ethylene glycol or distilled water is recommended as gun coolant. Tap water is unacceptable.

### 3.3.2 Detection reference

#### SmarTac sensing circuitry

The SmarTac sensing circuitry is self-optimizing. The detection level is adapted to the load from the welding gun. This function reduces the effect of weld equipment impedance. It involves the use of a memory feature that latches SmarTac's reference at the time of searching. The memory feature is controlled by an output from the robot.

## 3.3.3 Sensitivity

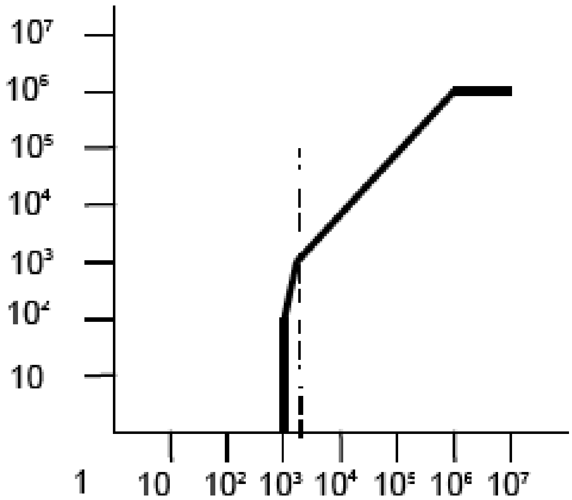
**Introduction**

When/if SmarTac is affected by the welding equipment during searching, and the trigger reference is set before searching, the sensitivity will be as shown in the following diagram.

**Diagram**

The diagram shows SmarTac's working range, which is the area to the right of the vertical dashed line in the diagram. This line is equivalent to a load of 1.5 k ohm.

| | |
|---|---|
| On the Y axis | Sensitivity = Object contact resistance in ohms |
| On the X axis | Equipment load on the measurement circuit in ohms |




xx1400001477

## 3.3.4 Sensing voltage

### Maximum sensing voltage

The maximum sensing voltage is 40 VDC, but the voltage is progressively reduced as the load from the welding equipment increases. The higher the sensing voltage, the more accurate the search will be.

### Typical voltage

A typical voltage seen on a real system is about 37 VDC.

## 3.3.5 Signals and connections

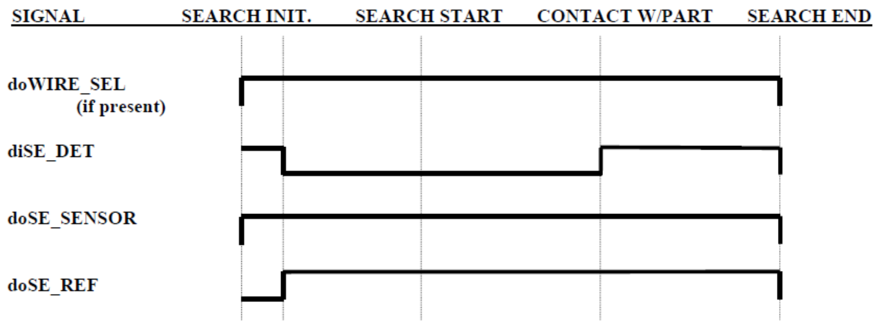
### About

SmarTac is operated by signals from the robot. Signals and connections are described below.

### SmarTac I/O

There are 4 I/O signals used by SmarTac in each applicable motion task.

| | |
|---|---|
| diSE_DETX | Input used for surface detection and sensor validity. |
| doSE_SENSORX | Output used to activate the SmarTac board. |
| doSE_REFX | Output used to set the sensing reference voltage. |
| doWIRE_SELX | Output used to switch the search detection signal between channel 11 and 12. Channel 11 is connected to the gas cup and channel 12 to the wire. The signal will be defined as a simulated signal if optional wire searching hardware has not been installed. |




xx1400001478

*Figure 3.1: I/O time line*

# 4 User's guide

## 4.1 Safety

### Safety instructions

**WARNING**

Failure to follow safety guidelines presented throughout this manual can result in property damage, serious injury, or death!

The SmarTac board is typically supplied with 230 VAC from the robot main power transformer. This supply power is potentially dangerous. Consequently, the control cabinet door should always be closed when the control cabinet is turned on. Only qualified technicians should ever attempt trouble shooting.

The SmarTac sensing voltage applied at the torch when searching is supplied by a 38 VDC, low current source. This sensing current is harmless.

**CAUTION**

All users must read the safety instructions in the product manual for the controller and the robot before operating the robotic system. See *Safety on page 10*.

## 4.2 Introduction

> **Note**
>
> All exercises assume that the SmarTac software and hardware are installed and working properly.

### System module

The system module, `SmarTac.sys`, included in the package, contains useful search instructions that simplify the programming. The module also includes mathematical functions that are useful in advanced searching techniques. All of these are discussed in this section.

### Questions

Before tactile searching can be used effectively, you need to be able to answer these questions:

1. How do my parts deviate?

   Knowing where the parts move, and where they do not, is critical for determining what features to search. Searching takes time. Unnecessary searches increase cycle time and programming complexity. In this manual, the simplest cases will be handled first. In many cases these techniques will be enough. In some situations where the part fit-up and/or fixture is poor, you will need to understand all the tricks described in the manual.
2. What is a frame?

   A good understanding of work objects and displacement frames is the key to successful programming with SmarTac searching. See *Operating manual - IRC5 with FlexPendant* and *Operating manual - RobotStudio*.
3. What are the RAPID instructions and how are they used in my weld routines?

   In this guide we will look at several search techniques, with detailed examples. The SmarTac instructions and functions are described in *RAPID reference on page 81*.

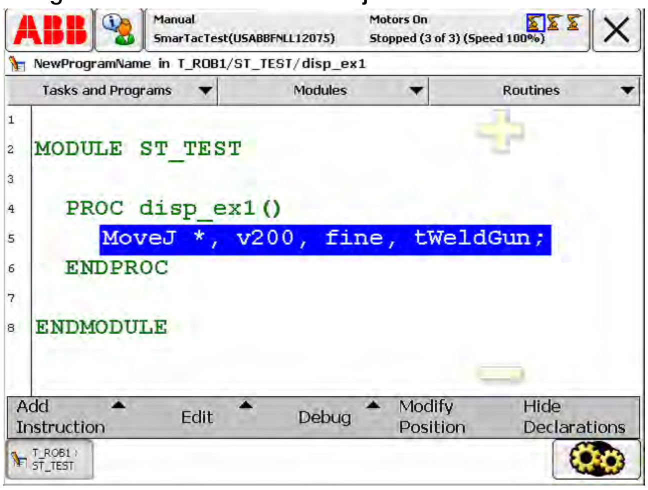
## 4.3 Exercise 1: program displacement

### About the exercise

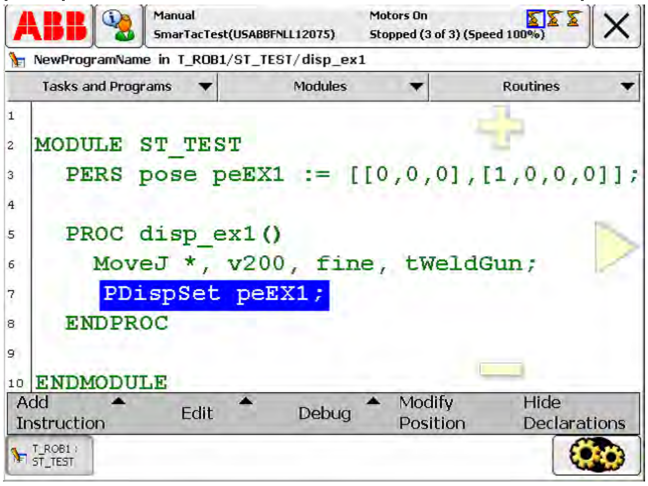
This exercise demonstrates how a program displacement works.

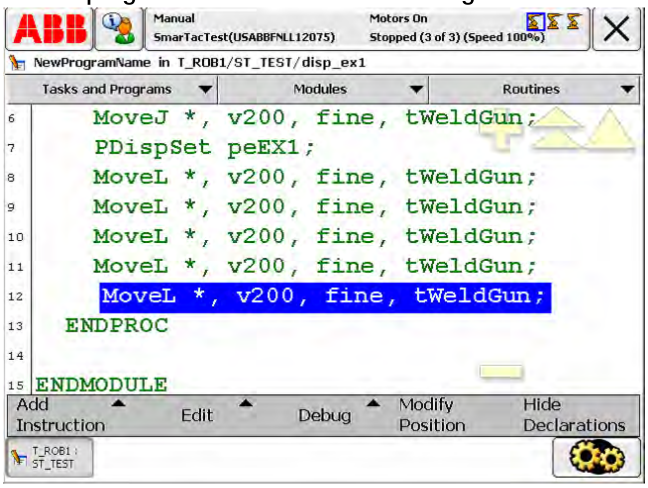
> **Note**
>
> The exercises later in this guide will not be as detailed as this one. Please take the time to understand this exercise before attempting others.

### Instruction

| | Action |
|---|---|
| 1 | Create a new program module. Name it `ST_TEST`. |
| 2 | Create a new routine in that module and name it `disp_ex1`. |
| 3 | If not already done, define the tool using the five point method or BullsEye. Name the tool `tWeldGun`.<br>To make programming easier you can add in these instructions into one of your *Most Common* pick lists:<br>`PDispAdd`<br>`PDispOff`<br>`PDispSet`<br>`Search_1D`<br>`Search_Groove`<br>`Search_Part` |
| 4 | Tape a piece of paper to a table, or similar surface, within the robot's reach. On the paper draw a rectangle. |
| 5 | View the modules, select the new module `ST_TEST`, and select the new routine, `disp_ex1`. |
| 6 | Jog the robot so that the torch is pointing at the rectangle on the paper. The tip of the torch should be a few inches above the rectangle. Create a `MoveJ` at this point using `tWeldGun` and no work object selected.<br><br><br>xx1400001480 |

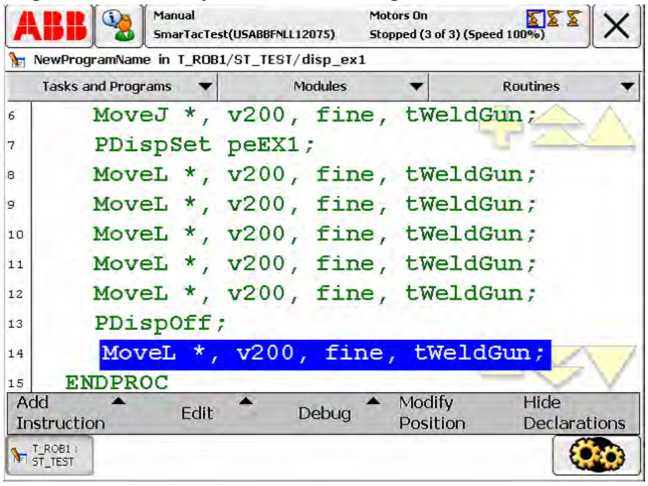
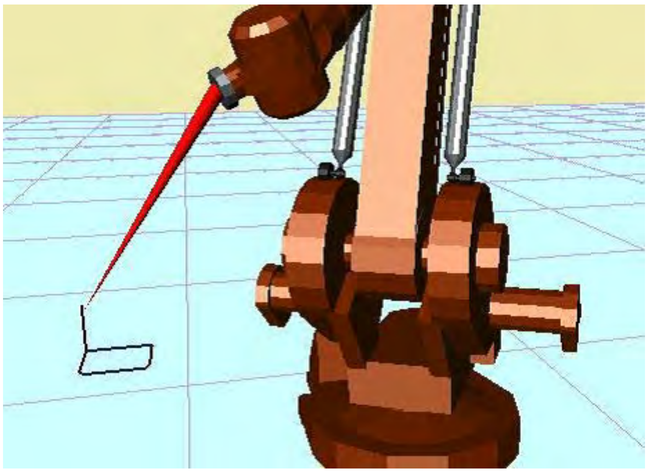
Continues on next page

<table>
<tr><th></th><th>Action</th></tr>
<tr><td>7</td><td>Insert the instruction <code>PDispSet</code>.<br>
This is a RAPID command that will be found on one of the standard instruction pick-lists. Here, you will see that we used a custom most common pick-list.<br>
The <code>PDispSet</code> instruction requires one argument: a displacement frame. When prompted for this data, select new and create a new pose data type called <code>peEX1</code>.<br>


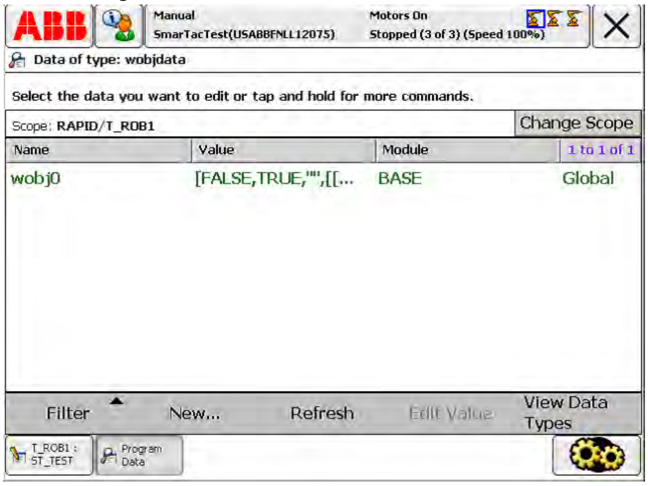
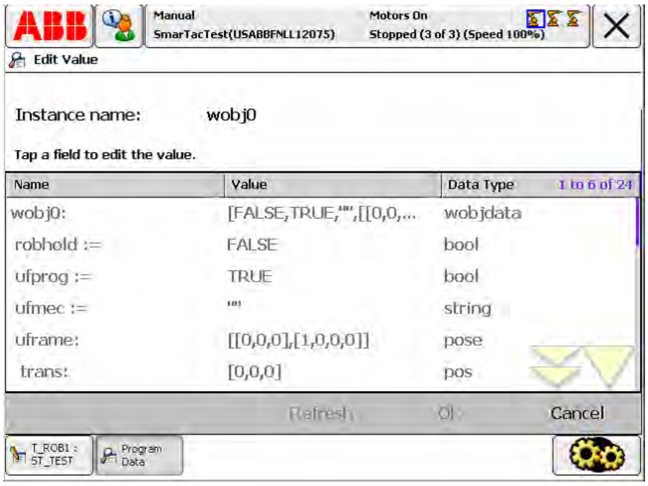
xx1400001481</td></tr>
<tr><td>8</td><td>Jog the robot down to the rectangle so that the tip is just above one of the rectangle corners.<br>
Create a <code>MoveL</code> at this position using <code>tWeldGun</code> and no work object selected.</td></tr>
<tr><td>9</td><td>Do the same for the rest of the corners, returning to the first corner, so that you have a short program that traces out the rectangle.<br>


xx1400001482</td></tr>
<tr><td>10</td><td>Insert the instruction <code>PDispOff</code>.</td></tr>
</table>

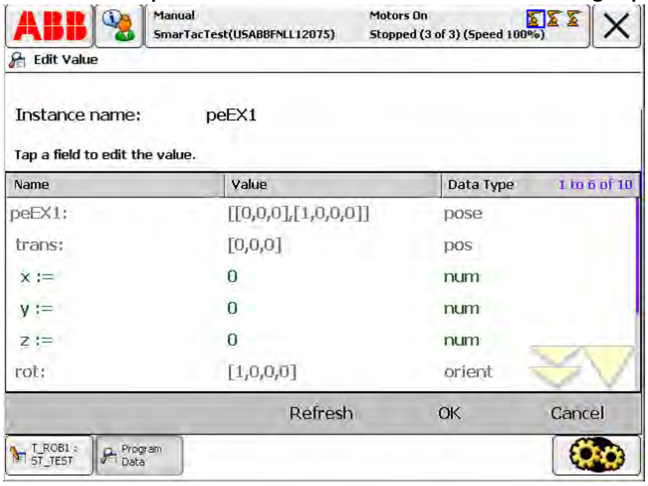
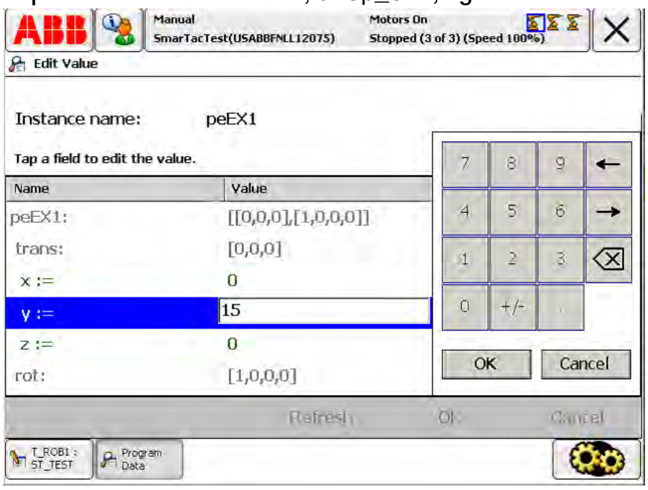
*Continues on next page*

| | Action |
|---|---|
| 11 | Jog the robot away from the rectangle a few inches and insert a final `MoveL`.<br><br><br>xx1400001483 |
| 12 | Execute this routine from the beginning in manual mode to be sure the program works correctly.<br>This routine executes a simple movement path. Each of the robtargets in the `MoveL` instructions is related to the world frame. In this case, however, the work object has not been specified, so `wobj0` is used by default. This work object is the same as the world frame.<br><br>xx1400001484 |

*Continues on next page*

| | Action |
|---|---|
| 13 | Open the Program Data window and take a look at the values in `wobj0`. Then tap to select `wobjdata`.<br><br><br>xx1400001485 |
| 14 | Tap and hold `wobj0`. The following will be visible:<br>Cursor down and look at the data that is present. The only other frame that can change the robtarget positions in our rectangle routine, is the displacement frame.<br>**Note**<br>A work object has two frames, the user frame, uframe, and the object frame, oframe.<br>Also note that the values are all zero for the locations, and ones and zeros for the orientations. That is why the work object has no affect on our program. It is the same as the World frame.<br><br><br>xx1400001486 |
| 15 | Tap **Cancel** to close the work object window. Then select **View Data Types**. |
| 16 | Select pose data types. |

*Continues on next page*

| | Action |
|---|---|
| 17 | View the data in `peEX1` by selecting it and then select **Edit Value**.<br>A screen will appear showing the values of this data instance. X, Y, and Z will all be zero. This displacement frame did not alter the rectangle program at all.<br><br>xx1400001487 |
| 18 | Move the cursor to the Y value and change the number to 15. |
| 19 | Tap OK and run the routine, `disp_ex1`, again.<br><br>xx1400001488 |

## What happens?

The movements are shifted 15 mm in the positive Y direction. That is 15 mm in Y relative to the work object, object frame. And, as discussed earlier, the object frame and user frame in `wobj0` are the same as the world frame. So the rectangle moved 15 mm relative to the world, as well.

This is what program displacement frames do. A change in the displacement frame changes the location of the robtargets. Displacement frames can be turned on and off using `PDispSet` and `PDispOff`. Similarly, changes in the work object will move the robtargets as well (work object modifications are shown in *Exercise 5: object frame manipulation on page 62*).

*Continues on next page*

Try making other changes in X, Y, and Z of `peEX1`. Remember, positive Z will move the rectangle up. Do not use a negative Z, as this will crash the tool.

**CAUTION**

Do not make changes to the four quaternions, q1-q4. If quaternions are changed manually, errors could occur. Quaternions must be normalized, so it is not possible to choose numbers randomly.

## Advanced

Look at the robtarget data using the same technique for looking at pose data. Note that the robtarget data does not change when the rectangle is moved using the displacement frame.

## 4.4 Exercise 2: using SmarTac to modify a displacement frame

### 4.4.1 Introduction

#### About one-dimensional search

SmarTac programming tools provide a simple way to search a part feature and apply the search results to a program displacement frame. As seen in *Exercise 1: program displacement on page 35*, using program displacements is an easy way of shifting programmed robtargets. The most basic search is a one-dimensional search. A one-dimensional search finds an offset in one direction.

#### What is Search_1D?

`Search_1D` is an instruction that is included in the SmarTac system module. Of the three search instructions included in the module, this is the most common. It is useful in a variety of situations and will be present in most of the exercises and examples from this point on.

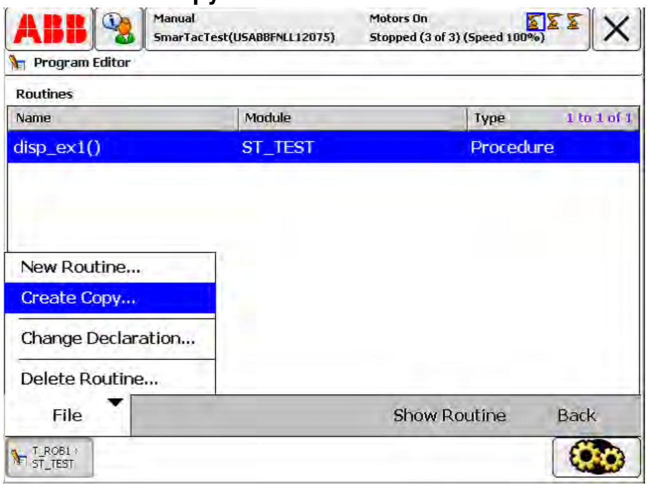
The instruction `Search_1D` is described in *Search_1D - One-dimensional search on page 81*.

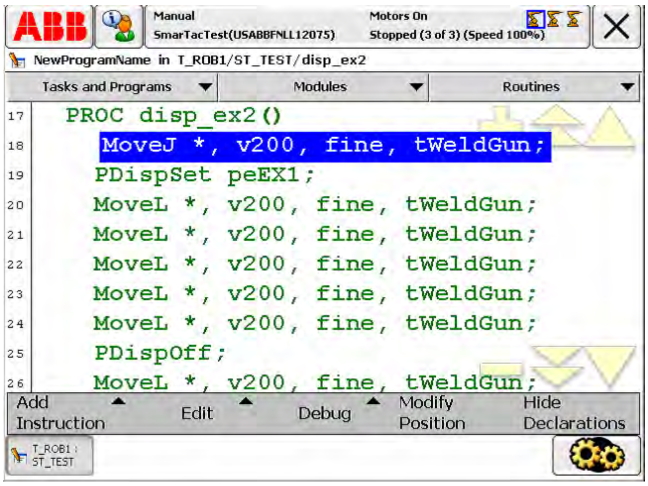
## 4.4.2 Exercise 2: one-dimensional search

### About the exercise

For this exercise the movement routine used in Exercise 1 is used, see *Exercise 1: program displacement on page 35*.

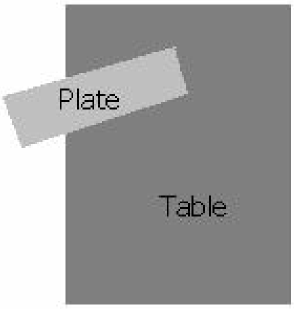
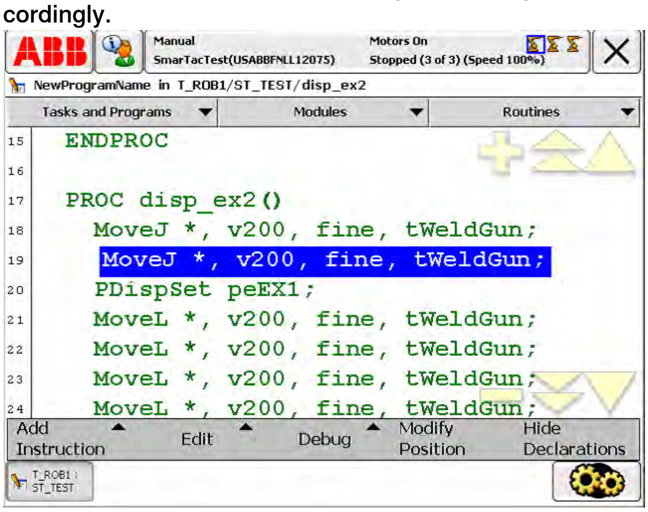
### Instruction

<table>
<tr><th></th><th>Action</th></tr>
<tr><td>1</td><td>View the routines in the module `ST_TEST`.</td></tr>
<tr><td>2</td><td>Select `disp_ex1` and open the File menu.<br>Select Create Copy.<br>

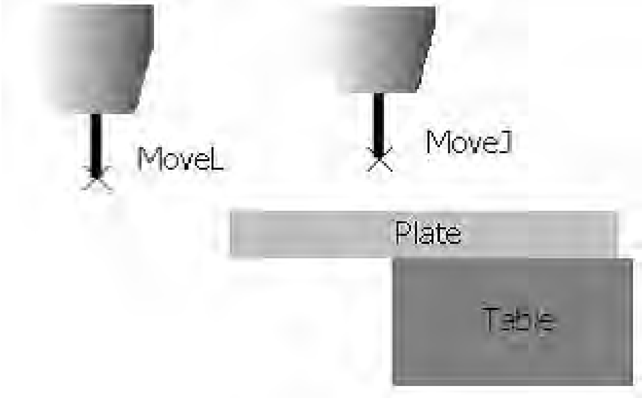
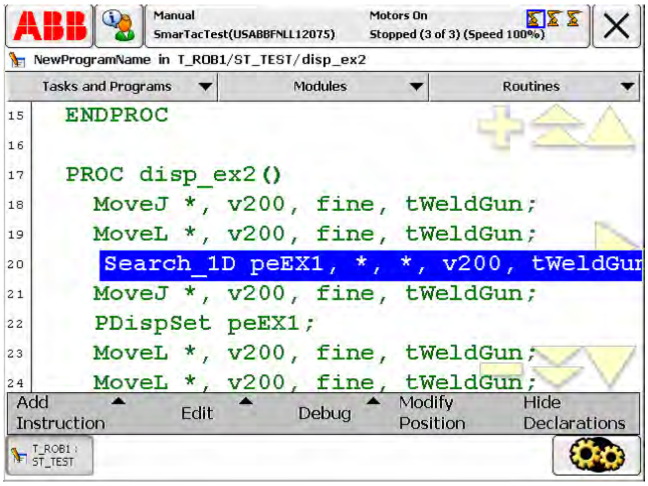
xx1400001492</td></tr>
<tr><td>3</td><td>Change the default routine name, `disp_ex1Copy`, to `disp_ex2`, then look at the instructions in the new routine.<br>

xx1400001493</td></tr>
<tr><td>4</td><td>Remove the paper with the rectangle drawn on it.</td></tr>
</table>

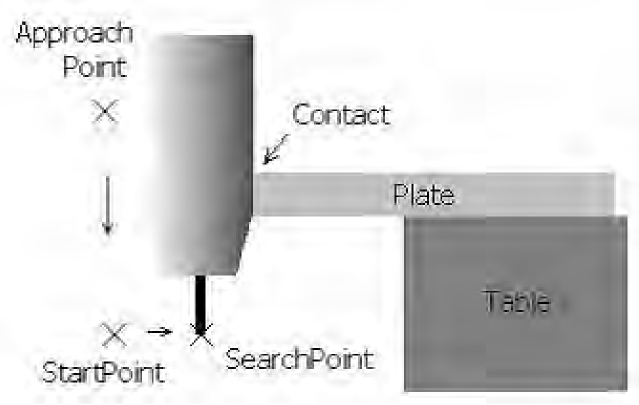
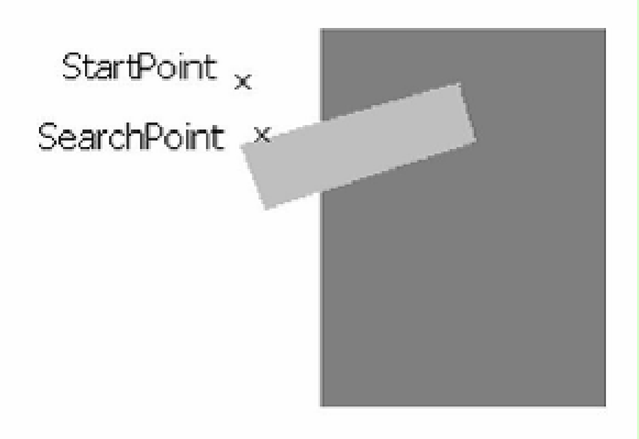
*Continues on next page*

| | Action |
|---|---|
| 5 | Place a metal plate on the table so that a portion of the plate sticks out over the edge of the table.<br><br><br>xx1400001494 |
| 6 | Edit the values in `peEX1` so that X, Y, and Z equal zero. |
| 7 | Move the program pointer to the new routine and toggle the Program Window to test mode. |
| 8 | Step through the rectangle program and modify each of the points so that they correspond to each of the plate's corners. |
| 9 | Continue to step though the routine until the program pointer loops back to the beginning. |
| 10 | Toggle the Program Window to instruction mode. If you have a Most Common pick-list with SmarTac instructions, select it. |
| 11 | Move the cursor to the first line, `MoveJ`. Using the **Copy** and **Paste** buttons, copy the `MoveJ` and paste the copy below.<br>We will add more movements and a search instruction between the two `MoveJ` instructions. The search will collect information about the location of the plate, and store the information in `peEX1`. Our plate-tracing movements will then be shifted accordingly.<br><br><br>xx1400001495 |

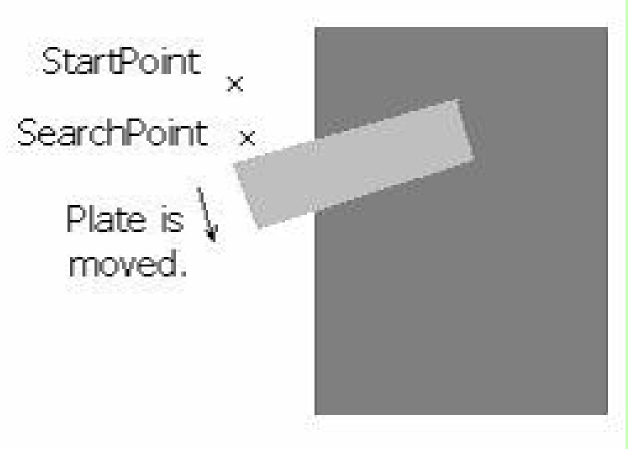
*Continues on next page*

<table>
<tr><th></th><th>Action</th></tr>
<tr><td>12</td><td>Jog the robot torch so that it is above and past the edge of the plate. Insert a <code>MoveL</code> instruction for this location between the two <code>MoveJ</code> instructions.<br>
<br>xx1400001496</td></tr>
<tr><td>13</td><td>Insert the instruction, <code>Search_1D</code>, after the <code>MoveL</code>. Use the following data:<br><code>Search_1D peEX1,*,*,v200,tWeldGun;</code><br>The routine should look like this:<br>
<br>xx1400001497</td></tr>
<tr><td>14</td><td>Jog the robot to the <code>SearchPoint</code> and modify the position of the second robtarget in the <code>Search_1D</code> instruction. The gas cup should make contact with the plate without deflecting the torch.</td></tr>
</table>

*Continues on next page*

| | Action |
|---|---|
| 15 | Jog the robot to the `StartPoint` and modify the position of the first `robtarget` in the `Search_1D` instruction.<br><br><br>xx1400001498<br><br><br>xx1400001499 |
| 16 | Toggle the Program Window to the test mode and move the program pointer to the `Search_1D` instruction. |
| 17 | Press the enabling device and press the forward button. The robot should move to the `StartPoint`, then search towards the `SearchPoint`, until the plate is detected. |

*Continues on next page*

| | Action |
|---|---|
| 18 | Toggle the Program Window to instruction mode and using the **Copy** and **Paste** buttons, copy the `MoveL` before the `Search_1D` and paste it after the `Search_1D`. Your final routine, `disp_ex2`, should look like this:<br>`PROC disp_ex2()`<br>`MoveJ *,v200,fine,tWeldGun;`<br>`MoveL *,v200,fine,tWeldGun;`<br>`Search_1D peEX1,*,*,v200,tWeldGun;`<br>`MoveL *,v200,fine,tWeldGun;`<br>`MoveJ *,v200,fine,tWeldGun;`<br>`PDispSet peEX1;`<br>`MoveL *,v200,fine,tWeldGun;`<br>`MoveL *,v200,fine,tWeldGun;`<br>`MoveL *,v200,fine,tWeldGun;`<br>`MoveL *,v200,fine,tWeldGun;`<br>`MoveL *,v200,fine,tWeldGun;`<br>`PDispOff;`<br>`MoveL *,v200,fine,tWeldGun;`<br>`ENDPROC` |
| 19 | Run the routine from the beginning. The torch should search the plate and then trace out the plate. |
| 20 | Move the plate about 10 mm away from the `SearchPoint` and try running the routine (see *Exercise 2, displacement on page 46*).<br>If the plate was moved in the direction of the search, without any rotation, the torch should still trace out the plate correctly.<br><br><br>xx1400001500<br>*Figure 4.1: Exercise 2, displacement* |

## Questions

1. Look at the data in `peEX1`. How does it change after searching different locations?
2. What happens when the plate is moved in other directions?

## Advanced

1. What happens when the search is programmed so that the search direction is not perpendicular to the plate's edge?

*Continues on next page*

2 What errors occur when the plate is moved too far away? Experiment with the error recovery options to see what they do. See *Instructions on page 81* for details on error handing.

## 4.4.3 Programming tips

**Tips**

1. Remember that the direction of the search dictates the direction that the resulting program displacement can shift a program.
2. You should almost always search perpendicular to the part feature surface. The search accuracy will suffer if the search direction is at an angle to the feature surface.
3. For a newly programmed search try executing the search using the forward button. When the robot stops motion with the torch touching the part, move the cursor to the `SearchPoint` and modify the position of the robtarget. This ensures that a search on a perfect part will return a displacement that is very close to zero.

## 4.5 Exercise 3: using SmarTac for multi-dimensional searching

### 4.5.1 Introduction

#### About multi-dimensional search

As seen in exercise 2 (*Exercise 2: one-dimensional search on page 42*), a one-dimensional search will determine where a weld seam is if it is constrained to move in only one direction. In some cases this is adequate. More likely, though, a two or three-dimensional search is required. A two-dimensional search would provide information about where a plate is located on a table, for example. A three-dimensional search would also determine how high the table surface was.

#### Limitations for single and multi-dimensional searching

In example 2 you may have noticed that if the plate was rotated when moved, the displacement frame would not compensate for the rotation. This is the limitation of single and multi-dimensional searching. These search techniques are relatively easy to master and, despite the limitation, provide accurate search information about the weld seam when used correctly.

To search a part in more than one direction, a combination of one-dimensional searches is used and the result of each search is added together. Exercise 3 demonstrates this for a two-dimensional search.

## 4.5.2 Exercise 3: two-dimensional search

### About the exercise

In this exercise the `Search_1D` instruction will be used twice to determine a two-dimensional shift in a plate on a table.

The instruction `Search_1D` is described in *Search_1D - One-dimensional search on page 81*.

### Instruction

| | Action |
|---|---|
| 1 | View the routines in the module `ST_TEST`. |
| 2 | Select `disp_ex2` and duplicate it. Name the new routine `disp_ex3`. |
| 3 | Toggle the Program Window to test mode. |
| 4 | Move the program pointer to the new procedure `disp_ex3`. |
| 5 | Move the program pointer to the instruction `PDispOff` near the end of the routine. |
| 6 | Press the enabling device and press the step forward button once to execute the instruction. |
| 7 | Make sure the robot can move to the first `MoveL` that traces out the plate, then move the program pointer to this `MoveL`. |
| 8 | Press the enabling device and press the step forward button once. Align the plate to the torch tip. Step though the rest of the points to get the plate back to where it was when we first wrote the routine. |
| 9 | Move the cursor to the top of the routine. Press the enabling device and step forward until the search is complete and the robot stops at the `MoveL` after the instruction `Search_1D`. |
| 10 | Add another `MoveL` here. Its location should be off the end of the plate. You are going to add another search to this routine that will search the end of the plate. This move will provide safe passage. |
| 11 | Copy the last `Search_1D` and insert it after the new `MoveL`. |
| 12 | Copy the new `MoveL` and insert it after the last `Search_1D`. |

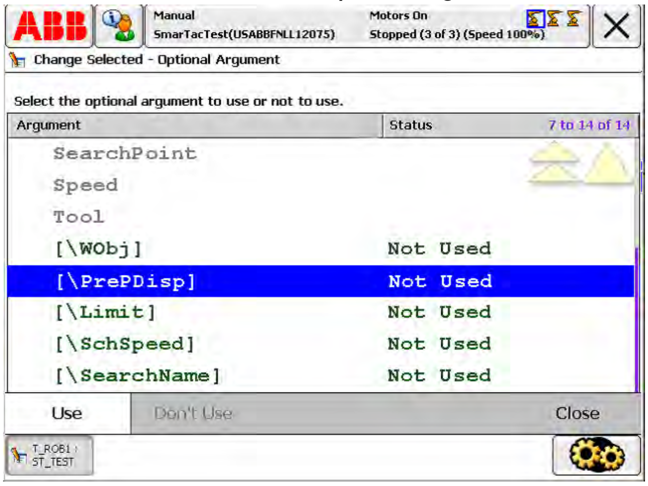
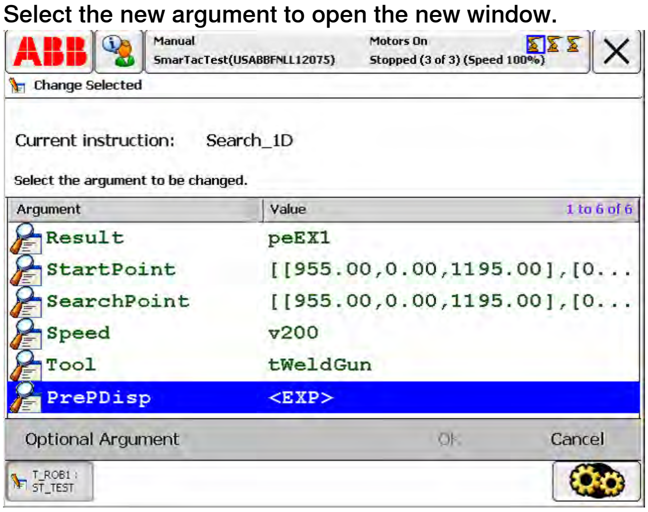
*Continues on next page*

| | Action |
|---|---|
| 13 | The routine should now look like this:<br>`PROC disp_ex3()`<br>`  MoveJ *,v200,fine,tWeldGun;`<br>`  MoveL *,v200,fine,tWeldGun;`<br>`  Search_1D peEX1,*,*,v200, tWeldGun;`<br>`  MoveL *,v200,fine,tWeldGun;`<br>`  MoveL *,v200,fine,tWeldGun; ! New MoveL`<br>`  Search_1D peEX1,*,*,v200,tWeldGun; ! New Search_1D`<br>`  MoveL *,v200,fine,tWeldGun; ! Copy of MoveL`<br>`  MoveJ *,v200,fine,tWeldGun;`<br>`  PDispSet peEX1;`<br>`  MoveL *,v200,fine,tWeldGun;`<br>`  MoveL *,v200,fine,tWeldGun;`<br>`  MoveL *,v200,fine,tWeldGun;`<br>`  MoveL *,v200,fine,tWeldGun;`<br>`  MoveL *,v200,fine,tWeldGun;`<br>`  PDispOff;`<br>`  MoveL *,v200,fine,tWeldGun;`<br>`ENDPROC` |
| 14 | Modify the robtargets in the new `Search_1D` to search the end of the plate. The new search will be referred to as `search 2`.<br><br><br>xx1400001501 |
| 15 | Highlight the second `Search_1D` instruction and tap Enter. |
| 16 | Tap **OptArg** to look at the optional arguments. |

*Continues on next page*

## 4.5.2 Exercise 3: two-dimensional search

*Continued*

| | Action |
|---|---|
| 17 | Move the cursor down to the optional argument named `PrePDisp` and tap **Use**.<br><br><br>xx1400001502 |
| 18 | Tap **OK**. |
| 19 | Select the new argument to open the new window.<br><br><br>xx1400001503 |

*Continues on next page*

| | Action |
|---|---|
| 20 | From the list of available pose data select `peEX1`. Tap **OK**. The routine should look like this:<br>`PROC disp_ex3()`<br>`  MoveJ *, v200, fine, tWeldGun;`<br>`  MoveL *, v200, fine, tWeldGun;`<br>`  Search_1D peEX1, *, *, v200, tWeldGun;`<br>`  MoveL *, v200, fine, tWeldGun;`<br>`  MoveL *, v200, fine, tWeldGun;`<br>`  Search_1D peEX1, *, *, v200, tWeldGun\PrePDisp:=peEX1;`<br>`  MoveL *, v200, fine, tWeldGun;`<br>`  MoveJ *, v200, fine, tWeldGun;`<br>`  PDispSet peEX1;`<br>`  MoveL *, v200, fine, tWeldGun;`<br>`  MoveL *, v200, fine, tWeldGun;`<br>`  MoveL *, v200, fine, tWeldGun;`<br>`  MoveL *, v200, fine, tWeldGun;`<br>`  MoveL *, v200, fine, tWeldGun;`<br>`  PDispOff;`<br>`  MoveL *, v200, fine, tWeldGun;`<br>`ENDPROC` |
| 21 | Jog the torch so that it is above the plate and execute the routine from the beginning. The torch should trace out the plate. |
| 22 | Move the plate about 10 mm in any direction and re-execute the routine. The torch should trace out the plate. |

## Questions

1. Why is it necessary that the `PrePDisp` is set to `peEX1` in this example? What happens when a different displacement frame (other than `peEX1`) is used in the first `Search_1D`?
2. What happens when this optional argument in the second `Search_1D` is not present?
3. Two and three-dimensional searches should almost always use search directions that are perpendicular to one another. Why?

## Advanced

1. What happens when the plate is rotated slightly? Why?
2. Add the optional argument `NotOff` to the first search instruction and execute the program. What does this do?

> **Tip**
>
> See *Search_1D - One-dimensional search on page 81*.

*Continues on next page*

3 What would happen if the argument `NotOff` was added to the second search and the next section of the routine had a welding instruction?

> **Tip**
>
> See *Search_1D - One-dimensional search on page 81*.

4 Version 7.0 only: Why must there always be at least one Move instruction between two searches?

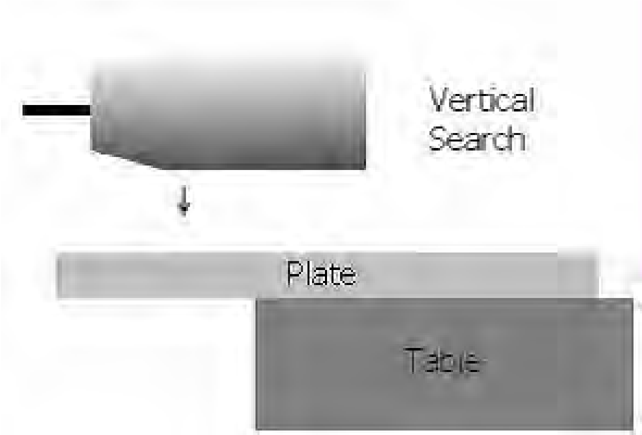
> **Tip**
>
> What happens when SmarTac is activated while the torch is touching the part?

5 If there is time try to write a three-dimensional search. Do not corrupt `disp_ex3` as it will be used later. The 3-D search should look something like this:

```
PROC disp_ex3_3D()
  MoveJ *,v200,fine,tWeldGun;
  MoveL *,v200,fine,tWeldGun;
  Search_1D peEX1,*,*,v200,tWeldGun;
  MoveL *,v200,fine,tWeldGun;
  MoveL *,v200,fine,tWeldGun;
  Search_1D peEX1,*,*,v200,tWeldGun\PrePDisp:=peEX1;
  MoveL *,v200,fine,tWeldGun;
  MoveL *,v200,fine,tWeldGun;
  MoveL *,v200,fine,tWeldGun;
  Search_1D peEX1,*,*,v200,tWeldGun\PrePDisp:=peEX1;
  MoveL *,v200,fine,tWeldGun;
  MoveJ *,v200,fine,tWeldGun;
  PDispSet peEX1;
  MoveL *,v200,fine,tWeldGun;
  MoveL *,v200,fine,tWeldGun;
  MoveL *,v200,fine,tWeldGun;
  MoveL *,v200,fine,tWeldGun;
  MoveL *,v200,fine,tWeldGun;
  PDispOff;
  MoveL *,v200,fine,tWeldGun;
ENDPROC
```

*Continues on next page*

Your routine may have more or less moves to re-orient the torch when searching in the vertical direction:




xx1400001504

6 In Example 1, what happens if you search the same edge twice using `PrePDisp` to add the second search result to the first?

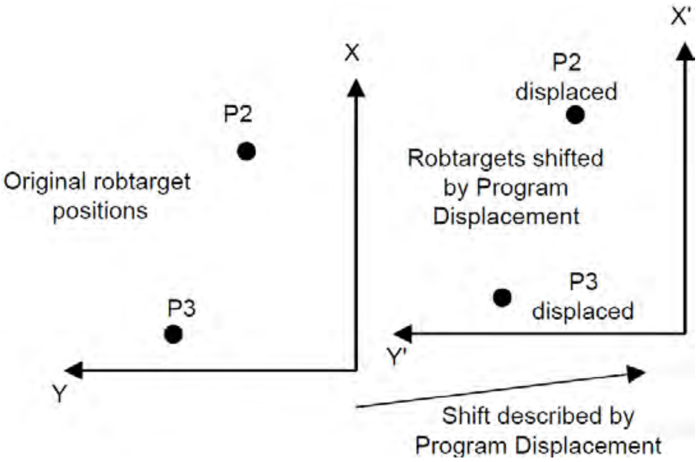
# 4.6 Exercise 4: using SmarTac to determine simple rotational changes

## 4.6.1 Introduction

### Translation and rotation

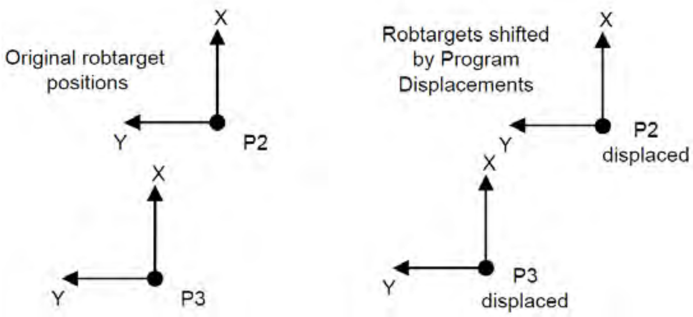
Up to this point basic one dimensional searches have been used to accurately locate part features that have moved only in translation, not rotation.

### Using SmarTac to determine simple rotational changes




xx1400001505

Using this same basic concept, it is possible to search a weld seam that moves both in translation and rotation. Imagine that the robtargets `P2` and `P3` in the illustration, describe the `ArcL\On` and `ArcL\Off` of a weld. If each robtarget is represented by a different program displacement then the weld seam can be moved rotational as well.




xx1400001506

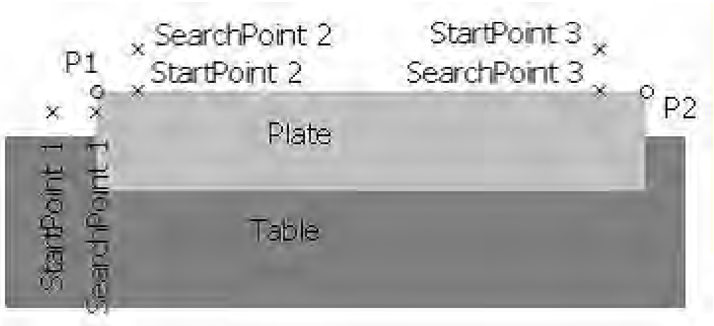
To do this for a real weld seam the robtargets `P2` and `P3` will have to be searched separately and the displacement data stored in two different `pose` data elements. See *Exercise 4: part feature with simple rotation on page 57*.

## 4.6.2 Exercise 4: part feature with simple rotation

### About the exercise

In this example a simple path with two points will be moved in translation as well as in rotation.

### Instructions

| | Action |
|---|---|
| 1 | Create a new routine called `disp_ex4` that looks like this:<br>`PROC disp_ex4()`<br>`  MoveJ *, v200, fine, tWeldGun;`<br>`  MoveL *, v200, fine, tWeldGun;`<br>`  ! Search 1`<br>`  Search_1D peEX1, *, *, v200, tWeldGun;`<br>`  MoveL *,v200,fine,tWeldGun;`<br>`  MoveL *,v200,fine,tWeldGun;`<br>`  ! Search 2`<br>`  Search_1D peEX2, *, *, v200, tWeldGun\PrePDisp:=peEX1;`<br>`  MoveL *, v200, fine, tWeldGun;`<br>`  MoveL *, v200, fine, tWeldGun;`<br>`  ! Search 3`<br>`  Search_1D peEX3, *, *, v200, tWeldGun\PrePDisp:=peEX1;`<br>`  MoveL *, v200, fine, tWeldGun;`<br>`  MoveL *, v200, fine, tWeldGun;`<br>`  PDispSet peEX2;`<br>`  ! Corner 1`<br>`  MoveL P1, v200, fine, tWeldGun;`<br>`  PDispSet peEX3;`<br>`  ! Corner 2`<br>`  MoveL P2, v20, fine, tWeldGun;`<br>`  PDispOff;`<br>`  MoveL *, v200, fine, tWeldGun;`<br>`ENDPROC`<br>Three displacement frames will be used, `peEX1`, `peEX2`, and `peEX3`. You will need to create these if they do not exist in the system. Be careful that the correct pre-displacement is called for each search instruction.<br>The searches should look similar to those used in exercise 2 & 3 (see on page 45), but positioned around the plate like this:<br><br>xx1400001507 |

Continues on next page

| | Action |
|---|---|
| 2 | Modify P1 and P2 to be at the corners of the plate, as shown above. You will have to create the named robtargets, P1 and P2, if they do not exist in the system.<br>It is not necessary that these two points be named for the test to work. They are named here as a teaching aid only. |
| 3 | Modify the air movements to clear the plate. |
| 4 | Step through the routine to test the positions. If P1 and P2 are out of position, you can jog them into position and modify the positions with the program displacement turned on. |
| 5 | Execute the program from the beginning. Watch the robot trace the edge of the plate. |
| 6 | Move the plate in various directions, including rotation, and execute the routine each time. Does the robot torch follow edge each time?<br>If it does not, there check the program again to be sure all the correct displacement frames are in the right places. |

## Questions

1. How is the usage of PrePDisp different from that in *Exercise 3: two-dimensional search on page 50*?
2. Look at the values in each of the program displacements peEX1, peEX2, and peEX3. What are the values for the rotation portions, q1-q4?

## Advanced

1. When the plate is rotated significantly, do you see any error in the positioning of P1 and P2? Why will large rotations of the plate cause some error in this example?
2. Why would it be difficult to shift an intermittent stitch weld in this way?

# 4.7 Exercise 5: using SmarTac with work object manipulation

## 4.7.1 Introduction

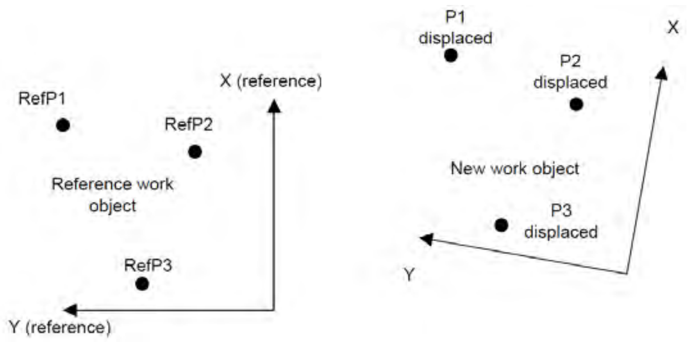
### Weld paths

Sometimes using multiple displacement frames can not provide an easy way of determining a weld seam's location. In exercise 4 we proved that a simple weld path could be moved in translation and rotation using two displacement frames; one for the start and one for the end of the weld path. If the weld were not continuous, that is a stitch weld, this would not work. There is no displacement information about the intermediate weld points.

### Work object

In some cases it is necessary to determine how the whole part has moved in translation and rotation. The best way to do this is to use a work object to describe where the part is in relation to the world frame. Based on search information, the object frame of a work object can be moved in translation and rotation. If the weld sequence in written in this work object, the points in the sequence will move with changes to the work object.

### Example 1



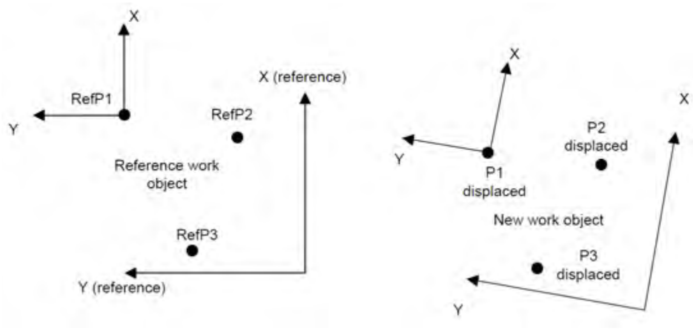

xx1400001508

**Note**

An important benefit to this technique is that searching and program displacements can still be used for features on the part after the part program has been rotated in the work object.

Continues on next page

### Example 2




xx1400001509

In this example the robtargets `P1`, `P2`, and `P3` all move with the work object. In addition, `P1` moves with a program displacement frame relative to that work object.

The SmarTac module contains two mathematical functions that can be used in conjunction with the `Search_1D` instruction to make this searching technique easier.

## 4.7.2 SmarTac functions

### Mathematical functions

Two global mathematical functions are provided in the SmarTac module.

- *PoseAdd - Adds the translation portions of pose data on page 99*
- *OFrameChange - Create a new shifted object frame on page 100*

Exercise 5 will illustrate the usage of these mathematical tools.

### PoseAdd

`PoseAdd` is a simple function used to add the transport of two or three displacement frames. The function returns pose data. In use it looks like this:

```
peSUM:=PoseAdd(peFIRST, peSECOND);
```

Using `PoseAdd` is similar to using the optional argument `PrePDisp` in `Search_1D`.

### OFrameChange

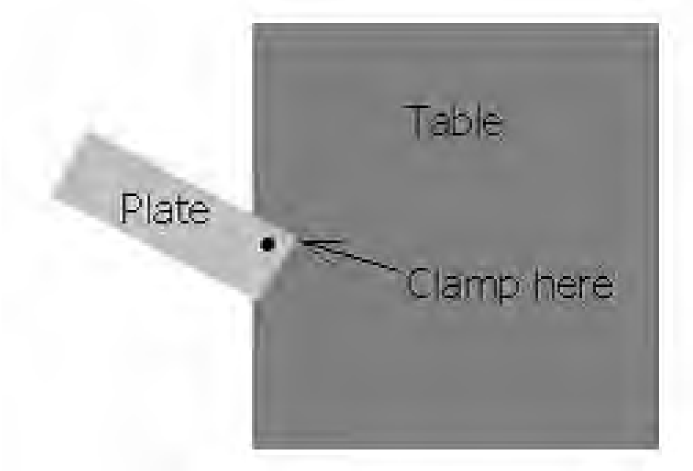
`OFrameChange` uses seven arguments; a reference work object, three reference points, and three displacement frames. The function returns `wobjdata`. In use it looks like this:

```
obNEW:=OFrameChange(obREF, p1,p2,p3,pe1,pe2,pe3);
```

## 4.7.3 Exercise 5: object frame manipulation

**About the exercise**

In this exercise a two dimensional example will be used, as in exercise 4. There will be four searches for this technique, so the plate will have to be clamped so that most of the plate is off the table.

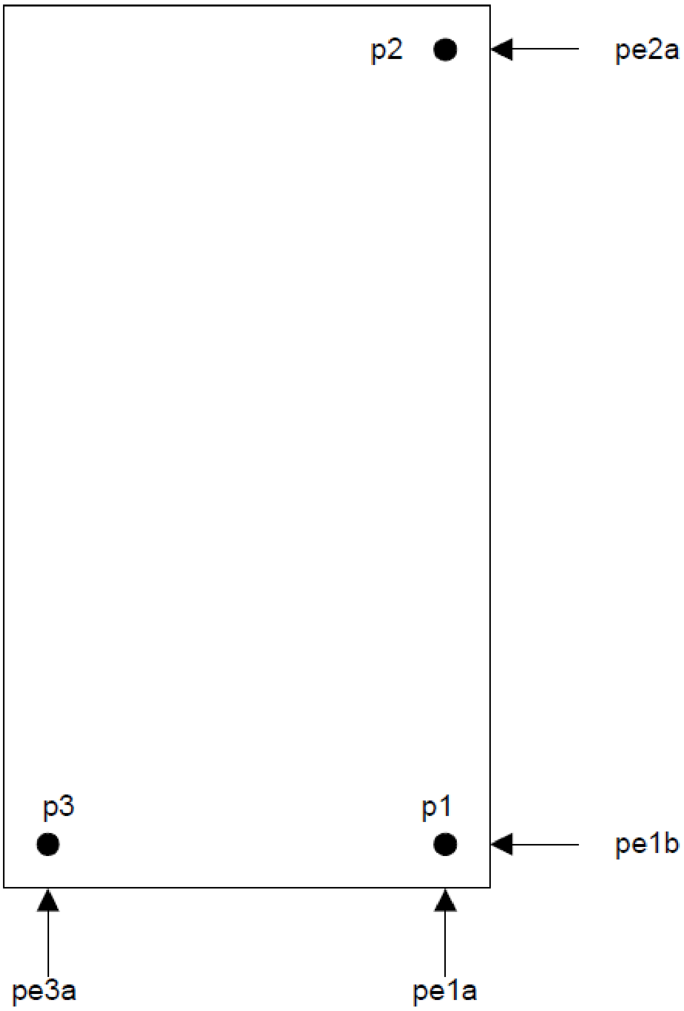



xx1400001510

*Continues on next page*

## Reference sketch

Refer to this sketch when laying out the points for this exercise.

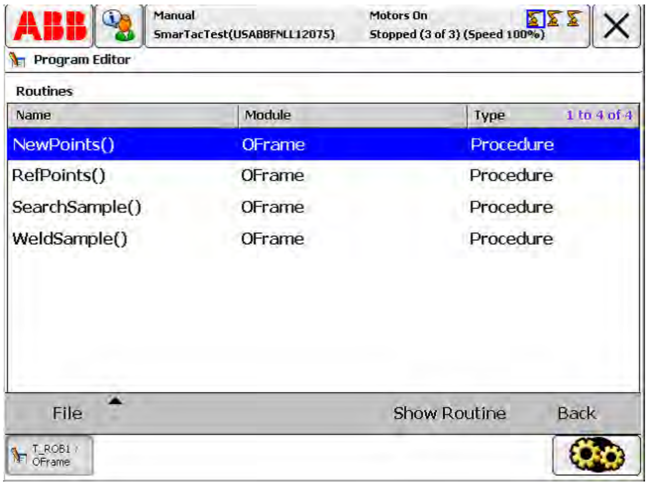
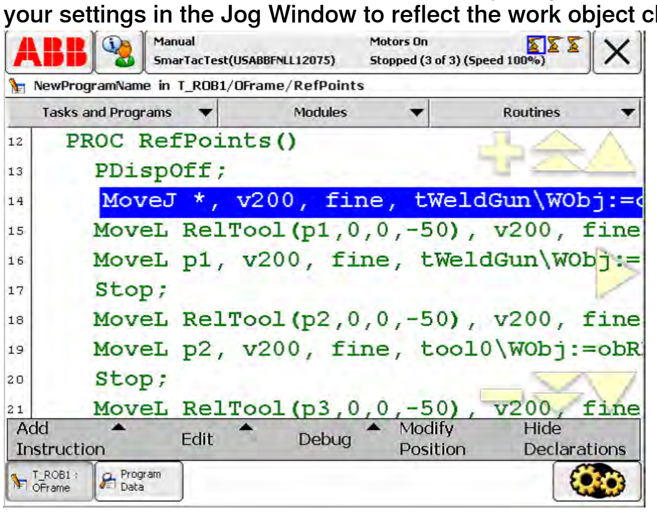



xx1400001511

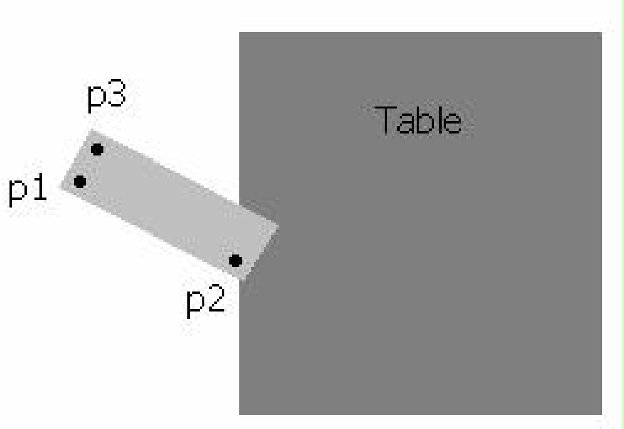
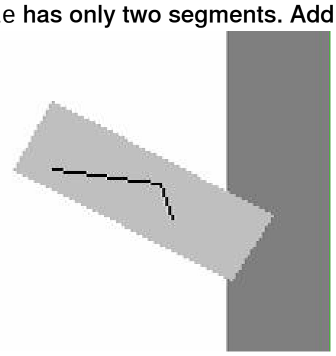
## Instruction

| | Action |
|---|---|
| 1 | Load the program module `OFrame`.<br>**Tip**<br>The module is included on the disk included in the delivery. It is also shown in *The module OFrame on page 102*. |

*Continues on next page*

| | Action |
|---|---|
| 2 | View the routines in the module. You should see the following routines:<br><br><br>xx1400001512 |
| 3 | Mark three points on the surface of the plate and label them `p1`, `p2`, and `p3`. The location of the points is not critical, but they should be near the corners as shown in *Reference sketch on page 63* or step 14. |
| 4 | Move the program pointer to the procedure `RefPoints`. `RefPoints` is a routine that, once updated, will point out the three reference points. |
| 5 | Execute the instruction `PDispOff` at the beginning of the procedure using the forward button. (This ensures that no displacements are active.) |
| 6 | Jog the robot so that the torch is situated about 150 mm above the plate surface and pointing down at the plate. |
| 7 | Move the cursor to the first `MoveJ` and modify the position. You may have to change your settings in the Jog Window to reflect the work object change.<br><br><br>xx1400001513 |
| 8 | Jog the robot so that the torch TCP is just touching the `p1` mark, and modify the second `MoveL`.<br>`MoveL p1, v200, fine, tWeldGun\WObj:= obREF;` |
| 9 | Jog the robot to the `p2` mark and modify the `MoveL`: `MoveL p2, v200…` |
| 10 | Jog the robot to the `p3` mark and modify the `MoveL: MoveL p3, v200…` |

*Continues on next page*

| | Action |
|---|---|
| 11 | Jog the robot so the torch is about 150 mm above the plate and modify the last `MoveJ`. |
| 12 | Move the program pointer to the beginning of the routine, and start execution. The robot should go from point to point with the torch TCP, stopping at each point so that the position can be checked. If any positions need to be changed, change them now. |
| 13 | Move the program pointer to the procedure `SearchSample`. |
| 14 | Jog the robot so that the torch TCP is above the plate and off the corner where `p1` is. Modify the first `MoveJ`.<br><br><br>xx1400001514 |
| 15 | Jog the robot down so that the torch gas cup is in a position to search the edge of the plate. See *Reference sketch on page 63*, and modify the points the first `Search_1D`. The search direction is indicated in the sketch for the displacement frame `pe1a` (the first search result). Remember that the search direction should be perpendicular to the edge of the part. |
| 16 | Modify all the rest of the moves and searches as shown in *Reference sketch on page 63*. |
| 17 | Test run the `SearchSample` procedure. |
| 18 | Move the program pointer to the routine called `WeldSample`. `WeldSample` does not have any `ArcL` instructions so RobotWare Arc does not need to be present to load this module. It has only `MoveL` instructions with slow speeds to simulate welding. |
| 19 | Draw a simulated weld on the surface of the plate using a straight edge and marker. `WeldSample` has only two segments. Add more if desired.<br><br>xx1400001515 |
| 20 | Modify the first point to be above the plate at least 100 mm. |
| 21 | Modify the second point to be the start of the simulated weld. |
| 22 | Modify the third and fourth points to be the middle and end of the weld. |

*Continues on next page*

| | Action |
|---|---|
| 23 | Modify the last point to be above the plate at least 100 mm. |
| 24 | Run `WeldSample` to be sure it follows the line correctly. |
| 25 | Run `SearchSample`. |
| 26 | Run `NewPoints`. The points `p1`, `p2`, and `p3` should be pointed out correctly. If not, there is a mistake somewhere. Check your program. |
| 27 | Run `WeldSample` again to be sure everything is ok. If the path is not followed, check the program again. |
| 28 | Leaving the plate clamped at the corner, move the plate about 10 mm at the end. |
| 29 | Run `SearchSample`. |
| 30 | Run `WeldSample`. The path should follow correctly. |
| 31 | Run `NewPoints`. The points should be pointed out correctly. |

## Questions

1. What work object is used in `RefPoints` and `SearchSample`?
2. What work objects are used in `NewPoints` and `WeldSample`?
3. Is `PDispSet` used in this exercise? Why, or why not? Look at the last several lines of `SearchSample`
4. `pe1` is the combination of what two searches?
5. `pe2` is the combination of what two searches?
6. `pe3` is the combination of what two searches?
7. For the best accuracy, there should be two searches for each reference point, located close to each reference point. In this exercise we use only four searches to approximate this. How far do you have to rotate the plate before you notice the inaccuracy?
8. Why does this occur?
9. Would this be a concern for most real-world fixtures?

## Advanced

1. Define the object frame of `obREF` so that the origin is at the corner of the plate where `p1` is. Align the object frame with the plate.
2. Select `obREF` as the work object and `WObj` for the coordinate system in the Jogging window. You should be able to jog along the edges of the skewed plate with straight deflections of the joystick. If not, the object frame was not defined properly.
3. Go though Example 5 again with the new work object definition. How might this help when programming?

# 4.8 Exercise 6: Search_Part

## 4.8.1 Introduction

### About Search_Part

Sometimes it is necessary to search a part feature to determine if it is there or not. Information like this can be used to determine what type of part is present, or if a part is loaded at all. The SmarTac instruction, `Search_Part` is provided for this use.

`Search_Part` is programmed very much like a `Search_1D` instruction, but it returns a Boolean instead of a program displacement. In use it looks like this:

```
Search_Part bPresent,p1,p2,v200,tWeldGun;
```

The robot moves on a path from `p1` through `p2`. If contact is made with the part feature, the Boolean, `bPresent`, is set to `TRUE`. If no contact is made, it is set to `FALSE`.

See *Search_Part - Search for feature presence on page 95*.

### Example

In this example a weld procedure is selected based on the presence of a particular part feature:

```
PROC Which_Part()
  MoveJ *,v200,z10, tWeldGun;
  MoveJ *,v200,fine, tWeldGun;
  Search_Part bPresent,p1,p2,v200,tWeldGun;
  IF bPresent THEN
    Big_Part;
  ELSE
    Small_Part;
  ENDIF
ENDPROC
```

## 4.8.2 Exercise 6: using Search_Part

### Instruction

| | Action |
|---|---|
| 1 | Create a new procedure in the module `ST_TEST`, named `disp_ex6`. |
| 2 | You need only one instruction in this procedure, `Search_Part`. You will have to create Boolean to use as your result. The robtargets need not be named. Search for the edge of the plate such that you can take the plate away later. |
| 3 | Run the routine to be sure it works OK. |

### Questions

1. With the plate in place, what is the value of the Boolean after searching?
2. With the plate removed, what is the value of the Boolean after searching?

### Advanced

1. What happens when you move the plate so that it is touching the gas cup at the start of the search?

# 4.9 Exercise 7: wire searching

## 4.9.1 Introduction

### About wire searching

Sometimes it is necessary to search with the welding wire, rather than the gas cup. In some systems with the necessary optional hardware installed, this is possible. The SmarTac instructions are designed to handle this. `Search_1D` and `Search_Part` each have an optional argument, `Wire`, that will switch the signal to the wire if selected. The instruction `Search_Groove` assumes that wire searching capabilities are present.

## 4.9.2 Exercise 7: wire searching

**Note**

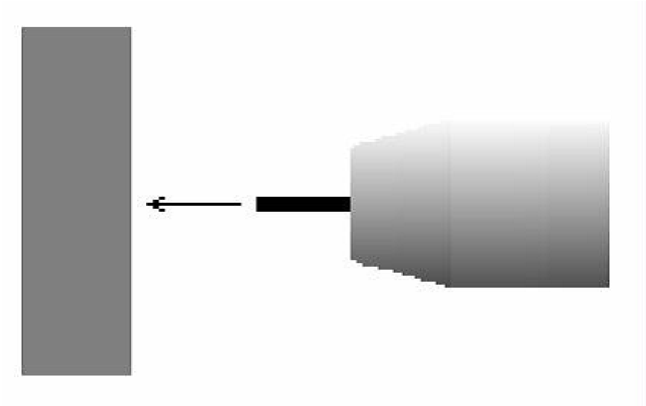
This exercise can only be done on systems that have wire searching capability.

### Instruction

| | Action |
|---|---|
| 1 | Add the optional argument `Wire` to the `Search_Part` instruction used in exercise 6. |
| 2 | Move the search points so that the wire will touch the part instead of the gas cup. |

### Questions

1. Did the system work correctly?
2. What problems could arise from searching with the side of the wire?
3. What problems could arise when searching with the tip of the wire in the direction of the wire? See the following figure.



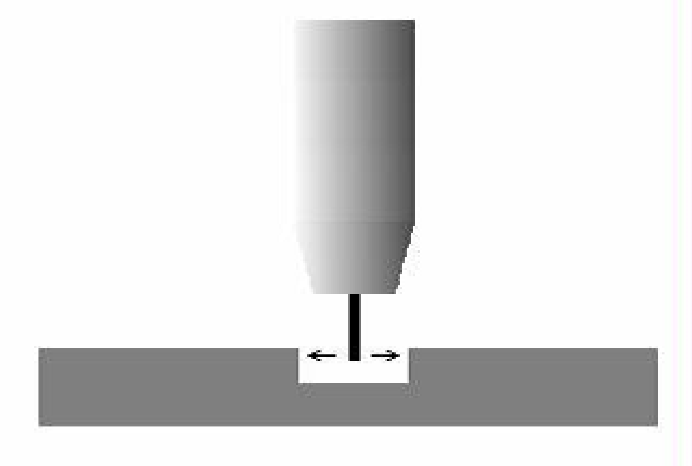
xx1400001516

# 4.10 Exercise 8: searching for a groove

## 4.10.1 Introduction

### Example of weld seam

In heavy welding applications it is common to have *groove* type weld seams. The simplest example is the square groove with a backing.



xx1400001517

### Searching

In heavy welding tolerances are usually large so searching is critical in determining the location and width of seams like the one above. The instruction `Search_Groove` has been provided to satisfy this need.

**Note**

See *Search_Groove - Find groove width and location on page 88*.

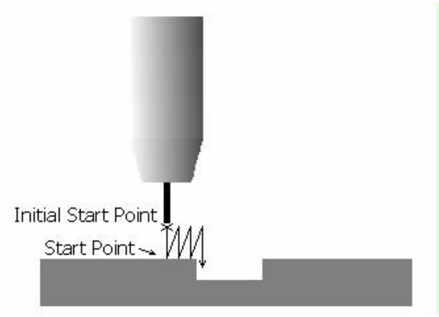
### Requirements for Search_Groove

`Search_Groove` performs a series of searches when executed. It requires two robtargets. One is programmed outside the groove, and the other in the center of the groove. It requires a displacement frame that will be returned as the seam offset. It requires a number that will be returned as the actual width of the seam. It also requires a nominal width, a speed, and a tool.

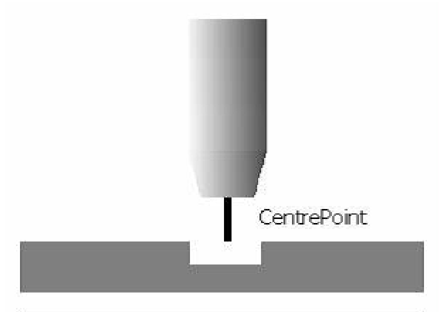
*Continues on next page*

### Illustration

In use it looks like this:




xx1400001518




xx1400001519

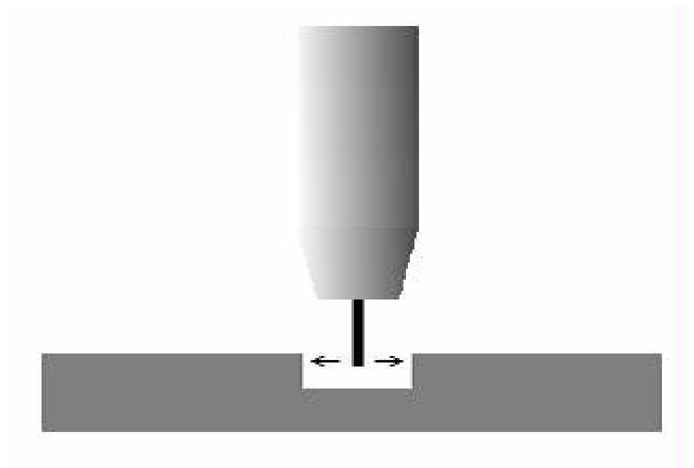
```
Search_Groove peOffset, nWidth, p1, p2, 10, v200, tWeldGun;
```

### Facts

The groove search is used to find the location and width of the groove to be welded.

- The groove has a 10 mm nominal width.
- The program displacement is stored in `peOffset`.
- The actual width of the groove is stored in `nWidth`.
- The `StartPoint` is `p1` and the `CentrePoint` is `p2`.
- The `Initial Start Point` is 15 mm above the `StartPoint` by default.

Searching for groove width and groove location



xx1400001520

## 4.10.2 Exercise 8: Searching for a groove weld

### Instruction

| | Action |
|---|---|
| 1 | Build a seam something like the one shown in *Searching for groove width and groove location on page 72*. |
| 2 | Try programming a `Search_Groove` instruction using the information in *Search_Groove - Find groove width and location on page 88*. Remember: Only the wire can be used on a groove search. |

### Questions

Use *Search_Groove - Find groove width and location on page 88* as a reference in answering these questions.

1 Where should the first robtarget, `StartPoint`, be programmed?

2 Where should the second robtarget, `CentrePoint`, be programmed?

3 What effect does changing the nominal groove width, `NomWidth`, have on the search pattern?

4 What effect does it have on the results (displacement & actual width)?

5 Which moves are effected by changes in the `speeddata`?

### Advanced

| | Action |
|---|---|
| 1 | Add the optional arguments, `InitSchL` and `NomDepth`. |
| 2 | Set the `InitSchL` equal to 15. |
| 3 | Set the `NomDepth` equal to 3. |

1 What happens when `InitSchL` is changed?

2 What happens when `NomDepth` is changed?

## 4.11 Conclusions

### About the overview

This overview provides most of the techniques required to use SmarTac searching on the majority of real-world weldments. A number of optional arguments for the search instructions have not been explained here. For more information about these, as well as more examples, see *Instructions on page 81*. There is also an instruction called `PDispAdd` which is used with the same effect as the function `PoseAdd`.

### Work objects

Work objects are described in *Operating manual - IRC5 with FlexPendant*. Especially for users with coordinated work objects on positioning equipment, a firm understanding of work object user and object frames is critical to writing good weld routines.

# 5 Trouble shooting

## 5.1 SmarTac board not "on"

### Description

The SmarTac board is not "on". Normally the SmarTac board is powered up when the robot cabinet is powered up. If the board is "on" a green LED labelled "D23 Search Sensor Valid" will be lit.

### Possible causes

If the LED is not lit:

1. Make sure the torch sensing surface is not touching the part or shorted to ground in any way.
2. 220VAC supplies power to the board on terminals 13 and 14. Check that terminal 14 has 220VAC when referenced to terminal 13 (neutral). If no power is present, see the product manual for the controller.
3. If power is supplied to the board, check that terminals 4 and 7 are at 0VDC. If these are not set low, set `doSE_REF` and `doSE_SENSOR` to zero. Check that terminals 4 and 7 are at 0VDC. If they are not at zero, check that the physical outputs, set `doSE_REF` and `doSE_SENSOR`, are at zero at the I/O board. If it is not, check the system parameters. If ok, check the SmarTac circuit per the wiring schematics in section *Electrical Reference* in the SmarTac hardware manual.
4. If nothing can be found wrong in the above list then replace the SmarTac board.

## 5.2 SmarTac board activation error

### Description

The SmarTac board does not activate correctly.

### Possible causes

1. The board is not supplied with power or the sensing surface of the torch is shorted to ground. See *SmarTac board not "on" on page 75* and *False-positive torch contact on page 77*.
2. Turn on `doSE_SENSOR` and turn off `doWIRE_SEL` and `doSE_REF`. When the part is not in contact with the torch sensing surface, the following green LEDs on the SmarTac board should indicate the following: *D22 Search Refoff D23 Search Sensor Validon* If not, check that terminals 4 and 7 have 24VDC present when referenced to ground. If not, check the SmarTac wiring per the schematics in section *Electrical Reference* in the SmarTac hardware manual. Also check the system parameters. If nothing can be found wrong with the wiring or the system parameters, replace the SmarTac board.
3. Turn on `doSE_REF`. When the part is not in contact with the torch sensing surface, the following green LEDs on the SmarTac board should indicate the following: *D22 Search Refon D23 Search Sensor Validon* If not, check that terminals 4 and 7 have 24VDC present when referenced to ground. If not, check the SmarTac wiring per the schematics in section *Electrical Reference* in the SmarTac hardware manual. Also check the system parameters. If nothing can be found wrong with the wiring or the system parameters, replace the SmarTac board.

## 5.3 False-positive torch contact

### Description

An error message appears on the screen stating that the torch has made contact with the part, before searching has begun, but the torch is clearly not touching the part. Or an activation error message appears on the screen.

### Possible causes

1. If the torch is a fluid-cooled gun, check that the coolant is non-conductive. If it is not, flush the system with de-ionized water and replace with new coolant. Never use tap water or automotive coolant.
2. If a secondary contactor or "positive lead break box" is present, check to see that it is working properly. The positive welding lead should be open when `doSE_SENSOR` is turned on. If the output to switch the contactor is working correctly, but the contactor is not working, repair or replace the secondary contactor or "positive lead break box". If the output to the contactor is not present, inspect the SmarTac wiring per the prints in section *Electrical Reference* in the SmarTac hardware manual.
3. Check the continuity of the torch between the contact tip and the gas cup. Resistance should be greater that 10Kohms. If not, check that the gas cup and gas diffuser are cleaned well. If there is still a short, replace the torch.
4. Check that the welding wire is not making contact with earth potential at any point.

## 5.4 No detection of the part

### Description

The robot never detects the part when searching. A crash results.

### Possible causes

1. Check that the sensing surface is free of dirt, soot, etc. that would otherwise prevent good electrical contact. Clean the cup at regular intervals. Grind any non-conductive coatings from the part that is to be searched.
2. Activate the SmarTac board by turning `doSE_SENSOR` on and `doWIRE_SEL` and `doSE_REF` off. Check that there is at least 25 VDC measured between the torch sensing surface and the part. Ideally there should be about 38 VDC present. A reading lower than 25 VDC indicates that there is a significant loss that will make search results inaccurate (see *Inaccurate results on page 79*). If no voltage is present, check that the SmarTac board is activating properly (see *SmarTac board activation error on page 76*). If the board is activating correctly, check the wiring from the torch to the SmarTac board. Also check that the ground leads on the part fixture are properly attached.

**Note**

The orange LED on the SmarTac board labelled *D24 Workpiece Det. Stop* should be lit when the board is activated followed by the torch making contact with the part or fixture.

## 5.5 Inaccurate results

### Description

Search results are inaccurate.

### Possible causes

1. Search result data is being used improperly. Verify that the RAPID search instructions are being used correctly. See *User's guide on page 33*.
2. The sensing voltage is too low. Activate the SmarTac board by turning `doSE_SENSOR` on, and `doWIRE_SEL` and `doSE_REF` off. Check that there is at least 25 VDC measured between the torch sensing surface and the part. Ideally there should be about 38 VDC present. A reading lower than 25 VDC indicates that there is a significant loss that will make search results inaccurate. Low voltage can result from contaminated coolant, defective torches, and grounded welding wire. See *False-positive torch contact on page 77* for more details.
3. The TCP is moving. Check to see that the torch is securely mounted to the robot and that no movement is detected when searching.

# 6 RAPID reference

## 6.1 Instructions

### 6.1.1 Search_1D - One-dimensional search

**Usage**

`Search_1D` is an instruction used for tactile searching with SmarTac. The search path is described by two required robtargets. The search result is stored as `pose` data in the required argument `Result`. All SmarTac board activation and deactivation is automatically handled.

**Basic examples**

```
Search_1D peOffset, p1, p2, v200, tWeldGun;
```

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

**Arguments**

```
Search_1D [\NotOff] [\Wire] Result [\SearchStop] StartPoint
     SearchPoint Speed Tool [\WObj ] [\PrePDisp] [\Limit]
     [\SearchName] [\TLoad]
```

`[\NotOff]`

Data type: `switch`

If selected, the welding positive lead secondary contact (break box) remains open at the end of the search. Additionally, the SmarTac board remains activated after the search ends. If this switch is selected directly before a welding instruction, welding current will not reach the torch.

`[\Wire]`

Data type: `switch`

If selected the output `doWIRE_SEL` will be set high when the SmarTac activation occurs. The SmarTac sensor will be switched from the gas cup to the wire when selected.

`Result`

Data type: `pose`

The displacement frame that will be updated

`[\SearchStop]`

Data type: `robtarget`

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

`StartPoint`

Data type: `robtarget`

*Continues on next page*

The start point of 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.

`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).

`[\SearchName]`

Data type: `string`

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

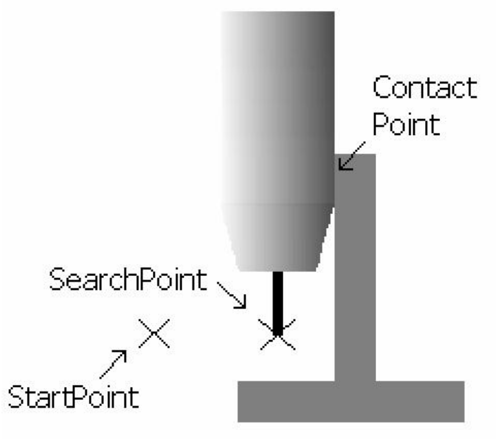
If the `\TLoad` argument is set to `load0`, then the `\TLoad` argument is not considered and the `loaddata` in the current `tooldata` is used instead. For a complete description of the `TLoad` argument, see `MoveL` in ***Technical reference manual - RAPID Instructions, Functions and Data types***.

*Continues on next page*

**Program execution**

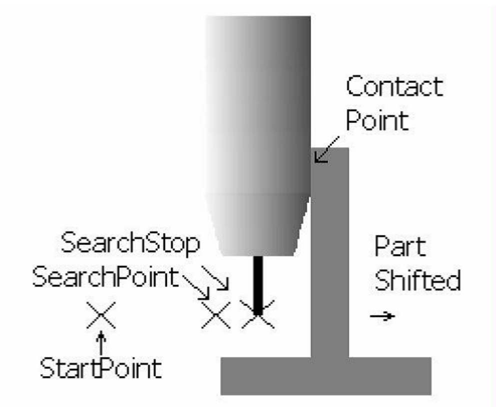
When executed, the robot makes a linear movement to the start point, `StartPoint`. The SmarTac board is activated and motion starts towards the search point, `SearchPoint`. The robot will continue past the search point for a total search distance described by twice the distance between `StartPoint` and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

Normally the gas cup is used for searching, however, on some systems the wire can be used for searching. When the switch, `Wire`, is selected, the digital output, `doWIRE_SEL`, is set high. This switches the SmarTac signal from the gas cup to the wire.




xx1400001490

The `StartPoint` and `SearchPoint` are programmed. The two points determine the direction of the search. The `SearchPoint` is programmed so the torch is touching the part feature. The `Result` is the difference between the programmed `SearchPoint`, and the actual `SearchStop` that is found when a different part is present.




xx1400001491

*Continues on next page*

## Limitations

If the switch, `NotOff`, is selected, the welding positive lead secondary contact (break box) remains open at the end of the search. If this switch is selected directly before a welding instruction, welding current will not reach the torch and an Arc Ignition Error will occur.

## Error handling

| Fault | Menu message |
|---|---|
| Fault 1 | Activation of the SmarTac failed |
| Fault 2 | Search failed |
| Fault 3 | `GasCup` or `Wire` touching part |

### Fault 1

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

**Activation of the SmarTac failed**

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

When RETRY is selected the start point of the search is shifted farther from the part feature. This may give a good search result in cases where the part feature is unusually close and the torch 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 preoffset 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 |
| RAISE | Sends error to calling routine. |

When RETRY is selected the start point of the search is shifted farther from the part feature. This may give a good search result in cases where the part feature is unusually close and the torch 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 preoffset included in the search instruction. A message will be logged in the event log.

*Continues on next page*

### Fault 3

If the torch makes contact with the part before the search begins, the following menu appears:

**GasCup or Wire touching part**

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

When RETRY is selected the start point of the search is shifted farther from the part feature. This may give a good search result in cases where the part feature is unusually close and the torch 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 preoffset 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 preset 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.

## More examples

### Single dimension search in any direction

```
MoveJ *, vmax, fine, tWeldGun;
Search_1D peOffset, p1, p2, v200, tWeldGun;
PDispSet peOffset;
ArcL\On,*, vmax, sm1, wd1, wv1, z1, tWeldGun;
ArcL\Off,*, vmax, sm1, wd1, wv1, z1, tWeldGun;
MoveJ *, vmax, z10, tWeldGun;
ArcL\On,*,vmax, sm1, wd1, wv1, z1, tWeldGun;
ArcL\Off,*, vmax, sm1, wd1, wv1, z1, tWeldGun;
PDispOff;
```

### Two dimension searching in any direction in a defined work object

```
MoveJ *, vmax,fine, tWeldGun\WObj:= wobj2;
Search_1D\NotOff, pose1, p1, p2, v200, tWeldGun\WObj:=wobj2;
Search_1D pose1, p3, p4, v200, tWeldGun\WObj:= wobj2\PrePDisp:=
     pose1;
PDispSet pose1;
ArcL\On,*, vmax, sm1, wd1, wv1, z1, tWeldGun\Wobj:= wobj2;
ArcL\Off,*, vmax, sm1, wd1, wv1, z1, tWeldGun\Wobj:= wobj2;
```

*Continues on next page*

```
MoveJ *, vmax, z10, tWeldGun\WObj:= wobj2;
ArcL\On,*,vmax, sm1, wd1, wv1, z1, tWeldGun\Wobj:= wobj2;
ArcL\Off,*, vmax, sm1, wd1, wv1, z1, tWeldGun\Wobj:= wobj2;
PDispOff;
```

> **Note**
>
> It is typically unproductive to have two searches in the same direction for the same feature. Multiple searches in the same direction using the `PrePDisp` option will be averaged. Searches for a single feature should almost always be 90 degrees from each other. This fact implies that usually there should never be more than three searches on any one feature.

### Other variations

One dimensional search with the wire active and the maximum limit set at 4 mm. If the magnitude of the transport of `peOffset` is greater than 4 mm an error is flagged.

```
Search_1D\Wire, peOffset, p1, p2, v200, tWeldGun\Limit:=4;
```

One dimensional search with the gas cup. The robtarget `p3` is updated with the actual search position.

```
Search_1D\SearchStop:=p3, pose1, p1, p2, v200, tWeldGun;
```

One dimensional search with the gas cup. If an error occurs while searching and the operator elects to continue with default results, the name, `First`, will appear along with the error description, in the event log. See *Error handling on page 84*.

```
Search_1D pose1, p1, p2, v200, tWeldGun\SearchName:="First";
```

## Syntax

```
Search_1D
  ['\ ' NotOff ',']
  ['\ ' Wire ',']
  [ Result ':=' ] < expression (INOUT) of pose > ','
  [ '\' SearchStop ':=' < expression (INOUT) of robtarget >',' ]
  [ StartPoint ':=' ] < expression (IN) of robtarget > ','
  [ SearchPoint ':=' ] < expression (IN) of robtarget > ','
  [ 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 > ]
  [ '\' SearchName ':=' < expression (IN) of string > ]
  [ '\' TLoad ':=' ] < persistent (PERS) of loaddata > ] ';'
```

## Related information

| | Described in |
|---|---|
| `Search_Groove` | *Search_Groove - Find groove width and location on page 88* |
| `Search_Part` | *Search_Part - Search for feature presence on page 95* |
| Data type `pose` | *Technical reference manual - RAPID Instructions, Functions and Data types* |

*Continues on next page*

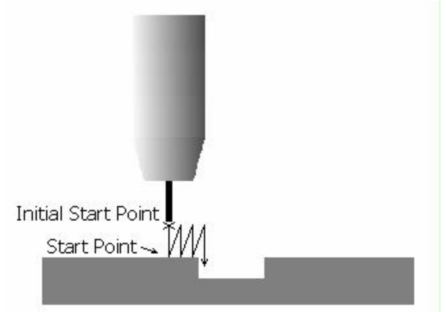
| | **Described in** |
|---|---|
| **Data type** `wobjdata` | *Technical reference manual - RAPID Instructions, Functions and Data types* |
| **Data type** `robtarget` | *Technical reference manual - RAPID Instructions, Functions and Data types* |
| `MoveL` | *Technical reference manual - RAPID Instructions, Functions and Data types* |
| **Definition of** `loaddata` | *Technical reference manual - RAPID Instructions, Functions and Data types* |

## 6.1.2 Search_Groove - Find groove width and location

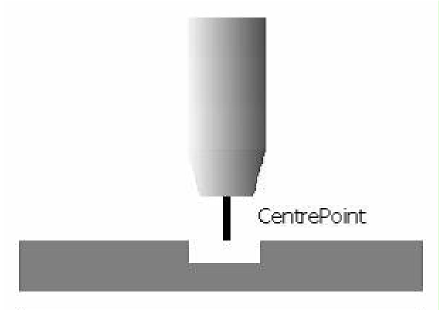
### Usage

`Search_Groove` is an instruction used for tactile searching of a "groove" with SmarTac. Searching is done with the wire. A series of searches are preformed to find the groove and determine its width. The `StartPoint` is programmed outside the groove at a point touching the part. The `CentrePoint` is programmed level with the `StartPoint`, but in the center of the groove. The search result is stored as pose data in the required argument Result. All SmarTac board activation and deactivation is automatically handled.

### Basic examples



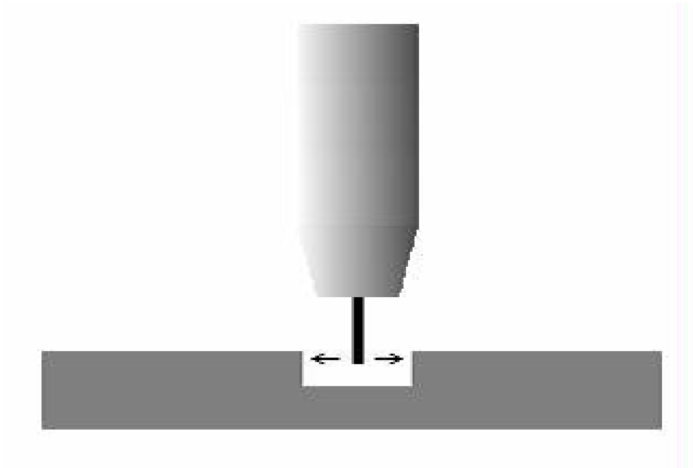

xx1400001521




xx1400001522

```
Search_Groove peOffset,nWidth,p1,p2,10,v200,tWeldGun;
```

The groove search is used to find the location and width of the groove to be welded. The groove has a 10 mm nominal width. The program displacement is stored in `peOffset`. The actual width of the groove is stored in `nWidth`. The `StartPoint` is `p1` and the `CentrePoint` is `p2`. The `Initial Start Point` is 15 mm above the `StartPoint` by default.



xx1400001520

*Continues on next page*

## Arguments

```
Search_Groove [\NotOff] Result GrooveWidth [\SearchStop] StartPoint
     CentrePoint NomWidth [\NomDepth] [\InitSchL] Speed Tool [\WObj
     ] [\PrePDisp] [\SearchName] [\TLoad]
```

`[\NotOff]`

Data type: `switch`

If selected, the welding positive lead secondary contact (break box) remains open at the end of the search. Additionally, the SmarTac board remains activated after the search ends. If this switch is selected directly before a welding instruction, welding current will not reach the torch.

`Result`

Data type: `pose`

The displacement frame that will be updated

`GrooveWidth`

Data type: `num`

The calculated groove width (in mm) determined by the search.

`[\SearchStop]`

Data type: `robtarget`

If selected, this robtarget will be updated as the point where the center of the groove is.

`StartPoint`

Data type: `robtarget`

The start point of the search sequence. This point should be programmed outside the groove, touching the part surface with the wire's tip. See the initial start point in *Basic examples on page 88*.

`CentrePoint`

Data type: `robtarget`

The point where the groove should be. This robtarget should be programmed so that the wire's tip is above the center of the groove, level with the adjacent part surface. See the center point in *Basic examples on page 88*.

`NomWidth`

Data type: `num`

The expected groove width (in mm). This number will effect the dimensions of the search sequence.

`[\NomDepth]`

Data type: `num`

The expected groove depth (in mm). If selected, this number will effect the dimensions of the search sequence. The default is 2.5 mm.

`[\InitSchL]`

Data type: `num`

*Continues on next page*

The length of the first search. If selected, this changes the `Initial Start Point`. The default is 15 mm. See the initial start point in *Basic examples on page 88*.

`speed`

Data type: `speeddata`

The speed data used when moving to the `Initial Start Point`. 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`.

`[\SearchName]`

Data type: `string`

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

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

## Program execution

When executed, the robot makes a linear movement to a point above the start point, the `Initial Start Point`. The height of the `Initial Start Point` above the `StartPoint` can be changed by the optional parameter, `InitSchL`. The SmarTac board is activated and motion starts towards the `StartPoint` (see initial start point in *Basic examples on page 88*).

The robot will continue past the `StartPoint` for a total search distance described by twice the distance between the `Initial Start Point` and the `StartPoint`.

*Continues on next page*

When the surface of the plate is found, more searches occur, each one closer to the edge of the groove.

When the groove is found, two searches are made inside the groove to determine the location and width (see *Searching for groove width and groove location on page 72*).

The start of both searches is beneath the `CentrePoint` (see the center point in *Basic examples on page 88*, and *Searching for groove width and groove location on page 72*). The optional parameter, `NomDepth`, will control how far into the groove the width and location searches will be. The displacement data is stored in Result.

This program displacement can later be used to shift programmed points using the RAPID instruction `PDispSet`. The width of the groove is stored in `GrooveWidth`.

## Limitations

To use `Search_Groove`, the system must have wire-searching capability.

If the switch, `NotOff`, is selected, the welding positive lead secondary contact (break box) remains open at the end of the search.

If this switch is selected directly before a welding instruction, welding current will not reach the torch and an Arc Ignition Error will occur.

## Error handling

| Fault | Menu message: |
|---|---|
| Fault 1 | Activation of the SmarTac failed |
| Fault 2 | Search failed |
| Fault 3 | GasCup or Wire touching part |
| Fault 4 | Groove not found |
| Fault 5 | Groove search failed |

### Fault 1

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

**Activation of the SmarTac failed**

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

When RETRY is selected the start point of the search is shifted farther from the part feature. This may give a good search result in cases where the part feature is unusually close and the torch 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 preoffset included in the search instruction. A message will be logged in the User Error Log.

*Continues on next page*

### 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 |
| RAISE | Sends error to calling routine. |

When RETRY is selected the start point of the search is shifted farther from the part feature. This may give a good search result in cases where the part feature is unusually close and the torch 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 preoffset included in the search instruction. A message will be logged in the User Error Log.

### Fault 3

If the torch makes contact with the part before the search begins, the following menu appears:

**GasCup or Wire touching part**

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

When RETRY is selected the start point of the search is shifted farther from the part feature. This may give a good search result in cases where the part feature is unusually close and the torch 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 preoffset included in the search instruction. A message will be logged in the User Error Log.

### Fault 4

If the groove walls are not found when searching for the groove width and location, the following message appears:

**Groove not found**

| | |
|---|---|
| RETURN | Continues the program with default search result. |
| RAISE | Sends error to calling routine. |

When RETURN is selected a default search result is used which will include any preoffset included in the search instruction. A message will be logged in the User Error Log.

*Continues on next page*

### Fault 5

If an error occurs when searching for the groove width and location, the following message appears:

**Groove search failed**

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

When RETRY is selected, the robot tries the search again.

When RETURN is selected a default search result is used which will include any preoffset included in the search instruction. A message will be logged in the User Error Log.

## More examples

The groove search is used to find the location and width of the groove to be welded. The program displacement is stored in `peOffset` and the width of the groove is stored in `nWidth`. The weave width in this example is set to `nWidth`.

```
MoveJ *, vmax,fine, tWeldGun;
Search_Groove peOffset,nWidth,p1,p2,10,v200,tWeldGun;
WvAdapt.weave_width:=nWidth;
PDispSet peOffset;
ArcL\On,*,vmax,sm1,wd1,wvAdapt,fine,tWeldGun;
ArcL\Off,*, vmax,sm1,wd1,wvAdapt,fine,tWeldGun;
PDispOff;
```

### Groove search with optional returned robtarget

The robtarget `p3` is updated with the actual groove centerline:

```
Search_Groove\SearchStop:= p3, pose1, nWidth, p1, p2, 10, v200,
     tWeldGun;
```

### Groove search that is "named"

If an error occurs while searching and the operator elects to continue with default results, the name, `First`, will appear along with the error description, in the event log. See *Error handling on page 91*.

```
Search_Groove peOffset, nWidth, p1, p2, 15, v200,
     tWeldGun\SearchName:= "First";
```

### Groove search that has a 30 mm first-search instead of the default 15 mm

```
Search_Groove peOffset, nWidth\InitSchL:= 30, p1, p2, 7, v200,
     tWeldGun;
```

## Syntax

```
Search_Groove
  ['\ ' NotOff ',']
  [ Result ':=' ] < expression (INOUT) of pose > ','
  [ GrooveWidth ':=' ] < expression (INOUT) of num >
  [ '\' SearchStop ':=' < expression (INOUT) of robtarget >','
  ]
  [ StartPoint ':=' ] < expression (IN) of robtarget > ','
  [ CentrePoint ':=' ] < expression (IN) of robtarget > ','
```

*Continues on next page*

```
[ NomWidth ':=' ] < expression (IN) of num >
[ '\' NomDepth ':=' < expression (IN) of num > ]
[ '\' InitSchL ':=' < expression (IN) of num > ] ','
[ Speed ':=' ] < expression (IN) of speeddata > ','
[ Tool ':=' ] < persistent (PERS) of tooldata >
[ '\' WObj ':=' < persistent (PERS) of wobjdata > ]
[ '\' PrePDisp ':=' < expression (IN) of pose > ]
[ '\' SearchName ':=' < expression (IN) of string > ]
[ '\' TLoad ':=' ] < persistent (PERS) of loaddata > ] ';'
```

## Related information

| | **Described in:** |
|---|---|
| `Search_1D` | *Search_1D - One-dimensional search on page 81* |
| `Search_Part` | *Search_Part - Search for feature presence on page 95* |
| **Data type** `pose` | *Technical reference manual - RAPID Instructions, Functions and Data types* |
| **Data type** `wobjdata` | *Technical reference manual - RAPID Instructions, Functions and Data types* |
| **Data type** `robtarget` | *Technical reference manual - RAPID Instructions, Functions and Data types* |

## 6.1.3 Search_Part - Search for feature presence

### Usage

`Search_Part` is an instruction used for tactile searching with SmarTac. The search path is described by two required robtargets. If a feature is detected, a required Boolean is set to `TRUE`, otherwise it is set to `FALSE`. In either case, program execution continues.

### Basis examples

```
Search_Part bPresent,p1,p2,v200,tWeldGun;
```

The robot moves on a path from `p1` through `p2`. If contact is made with the part feature, the Boolean `bPresent` is set to `TRUE`. If no contact is made, it is set to `FALSE`.

### Arguments

```
Search_Part [\NotOff] [\Wire] bDetect StartPoint SearchPoint Speed
    Tool [\WObj] [\TLoad]
```

`[\NotOff]`

Data type: `switch`

If selected, the welding positive lead secondary contact (break box) remains open at the end of the search. Additionally, the SmarTac board remains activated after the search ends. If this switch is selected directly before a welding instruction, welding current will not reach the torch.

`[\Wire]`

Data type: `switch`

If selected, the output, `doWIRE_SEL`, will be set high when the SmarTac activation occurs. The SmarTac sensor will be switched from the gas cup to the wire when selected.

`bDetect`

Data type: `bool`

The Boolean that will be updated. `TRUE`: if the part is sensed, `FALSE`: if the part is not sensed.

`StartPoint`

Data type: `robtarget`

The start point of 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.

`speed`

Data type: `speeddata`

*Continues on next page*

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.

`[\TLoad]`

Data type: `loaddata`

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

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

## Program execution

When executed, the robot makes a linear movement to the `StartPoint` with the velocity selected in `Speed`. The SmarTac board is activated and motion starts towards the `SearchPoint`.

The robot will continue past the search point for a total search distance described by twice the distance between `StartPoint` and `SearchPoint`. If a feature is detected, the required Boolean is set to `TRUE`, otherwise it is set to `FALSE`. In either case, program execution continues.

## Limitations

If the switch, `NotOff`, is selected, the welding positive lead secondary contact (break box) remains open at the end of the search.

If this switch is selected directly before a welding instruction, welding current will not reach the torch and an Arc Ignition Error will occur.

## Error handling

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

| | |
|---|---|
| RETRY | Tries to search again with start point moved 50% |
| DETECT | Continues the program with detection `TRUE` |
| REJECT | Continues the program with detection `FALSE` |
| RAISE | Sends error to calling routine. |

*Continues on next page*

If RETRY is selected the robot will move to the `StartPoint`, then to the approach point before searching.

When DETECT or REJECT are selected, a message is stored in the event log.

## Examples

In this example a procedure is selected based on the presence of a particular part feature:

```
PROC Which_Part()
  MoveJ *,v200,z10, tWeldGun;
  MoveJ *,v200,fine, tWeldGun;
  Search_Part bPresent,p1,p2,v200,tWeldGun;
  IF bPresent THEN
    Big_Part;
  ELSE
    Small_Part;
  ENDIF
ENDPROC
```

### Other variations

Searching with the wire:

```
Search_Part\Wire, bPresent, p1, p2, v200, tWeldGun;
```

Two searches in a work object:

```
Search_Part\NotOff, bPart1, p1, p2, v200, tWeldGun\WObj:= obPart;
Search_Part bPart2, p3, p4, v200, tWeldGun\WObj:= obPart;
```

## Syntax

```
Search_Part
  ['\ ' NotOff ',']
  ['\ ' Wire ',']
  [ bDetect':=' ] < expression (INOUT) of bool > ','
  [ StartPoint ':=' ] < expression (IN) of robtarget > ','
  [ SearchPoint ':=' ] < expression (IN) of robtarget > ','
  [ Speed ':=' ] < expression (IN) of speeddata > ','
  [ Tool ':=' ] < persistent (PERS) of tooldata > ','
  [ '\' WObj ':=' < persistent (PERS) of wobjdata > ]
  [ '\' TLoad':=' ] < persistent (PERS) of loaddata > ] ';'
```

## Related information

| | **Described in:** |
|---|---|
| `Search_1D` | *Search_1D - One-dimensional search on page 81* |
| Data type `bool` | *Technical reference manual - RAPID Instructions, Functions and Data types* |
| `MoveL` | *Technical reference manual - RAPID Instructions, Functions and Data types* |
| Definition of `loaddata` | *Technical reference manual - RAPID Instructions, Functions and Data types* |

## 6.1.4 PDispAdd - Add program displacements

### Usage

`PDispAdd` is an instruction used to add a program displacement frame to the current program displacement frame.

### Basic examples

```
PDispAdd pose2;
```

`Pose2` is added to the current displacement frame.

### Arguments

```
PDispAdd Result
```

`Result`

Data type: `pose`

The displacement frame added to the current program displacement frame.

### Program execution

When executed, `Result` is added to the current displacement frame, and the new program displacement frame is activated.

### Syntax

```
PDispAdd
  [ Result ':=' ] < expression (IN) of pose > ';'
```

### Related information

| | Described in: |
|---|---|
| `Search_1D` | *Search_1D - One-dimensional search on page 81* |
| `PoseAdd` | *PoseAdd - Adds the translation portions of pose data on page 99* |
| Data type `pose` | *Technical reference manual - RAPID Instructions, Functions and Data types* |

## 6.2 Functions

### 6.2.1 PoseAdd - Adds the translation portions of pose data

#### Usage

PoseAdd is a function that requires two or three pose data arguments and returns the sum of the translation portions in pose form.

The returned pose data will have the quaternions set to [1,0,0,0].

#### Basic examples

```
peSUM:=PoseAdd (peFIRST,peSECOND);
```

peSUM.trans is set equal to peFIRST.trans + peSECOND.trans. The rotational portion of the peSUM is set to [1,0,0,0] by default.

#### Return value

Data type: pose

The displacement frame.

#### Arguments

```
PoseAdd (Pose1 Pose2 [\Pose3])
```

Pose1

Data type: pose

Pose data to be added

Pose2

Data type: pose

Pose data to be added

[\Pose3]

Data type: pose

Pose data to be added

#### Syntax

```
PoseAdd '('
  [ Pose1 ':=' ] < expression (IN) of pose > ','
  [ Pose2 ':=' ] < expression (IN) of pose > ','
  [' \'Pose3 ':=' < expression (IN) of pose > ] ')'
```

#### Related information

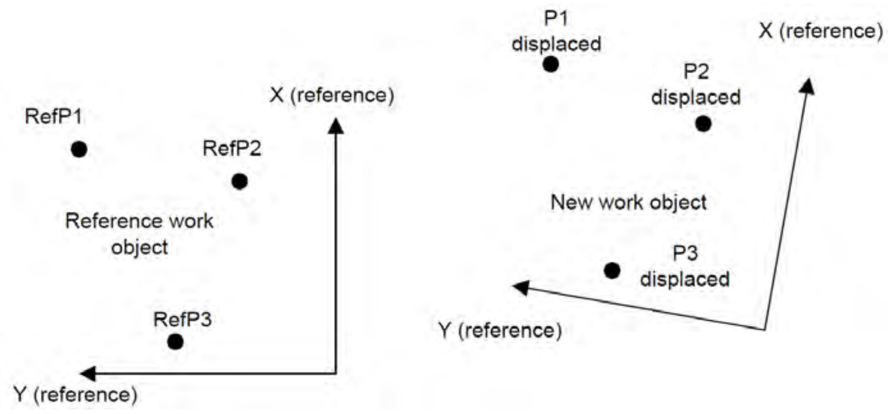
| | Described in: |
|---|---|
| PDispAdd | *PDispAdd - Add program displacements on page 98* |
| Data type pose | *Technical reference manual - RAPID Instructions, Functions and Data types* |

## 6.2.2 OFrameChange - Create a new shifted object frame

### Usage

`OFrameChange` is a function that returns a work object based on a required reference work object, three reference points, and three corresponding displacements, described within the reference work object.

### Basic examples




xx1400001523

```
obNEW:=OframeChange(obREF, p1, p2, p3, pe1, pe2, pe3);
```

The movement of the points `p1`, `p2`, and `p3` described by displacement frames `pe1`, `pe2`, and `pe3`, is superimposed over the object frame of the reference work object, `obREF`. The new work object, `obNEW`, has this new object frame and the original user frame information from `obREF`.

### Return value

Data type: `wobjdata`

The new work object.

### Arguments

```
OFrameChange ( WObj RefP1 RefP2 RefP3 DispP1 DispP2 DispP3)
```

`WObj`

Data type: `robtarget`

Reference work object.

`RefP1`

Data type: `robtarget`

Reference point number one. (Defined in `WObj`.)

`RefP2`

Data type: `robtarget`

Reference point number two. (Defined in `WObj`.)

*Continues on next page*

`RefP3`

Data type: `robtarget`

Reference point number three. (Defined in `WObj`.)

`DispP1`

Data type: `pose`

Displacement frame affecting reference point `RefP1`.

`DispP2`

Data type: `pose`

Displacement frame affecting reference point `RefP2`.

`DispP3`

Data type: `pose`

Displacement frame affecting reference point `RefP3`.

## Limitations

The reference points can be any three points in space, but they must be defined in the reference work object. Similarly, the displacements should be related to the reference work object.

The reference points do not have to be the same points as those used in defining the reference work object.

## Syntax

```
OFrameChange '('
  [ WObj ':=' ] < expression (IN) of wobjdata > ','
  [ RefP1 ':=' ] < expression (IN) of robtarget > ','
  [ RefP2 ':=' ] < expression (IN) of robtarget > ','
  [ RefP3 ':=' ] < expression (IN) of robtarget > ','
  [ DispP1 ':=' ] < expression (IN) of pose > ','
  [ DispP2 ':=' ] < expression (IN) of pose > ','
  [ DispP3 ':=' ] < expression (IN) of pose > ')'
```

## Related information

| | **Described in:** |
|---|---|
| `PDispAdd` | *PDispAdd - Add program displacements on page 98* |
| `PoseAdd` | *PoseAdd - Adds the translation portions of pose data on page 99* |
| Data type `pose` | *Technical reference manual - RAPID Instructions, Functions and Data types* |
| Data type `wobjdata` | *Technical reference manual - RAPID Instructions, Functions and Data types* |
| Data type `robtarget` | *Technical reference manual - RAPID Instructions, Functions and Data types* |

## 6.3 The module OFrame

### About OFrame

Exercise 5 uses a program module called `OFrame`.

The module is included on a disk with the delivery. Its purpose is to speed up the training process, whether it be an ABB training course or end-users training themselves. If the disk is not present, use this printout to assist in writing the code.

**Note**

Generic robtargets have been reduced to "*" to save space.

### Code for OFrame

```
! Example Module
  MODULE OFrame
    PERS wobjdata obREF:=[FALSE,TRUE,"",[[0,0,0], [1,0,0,0]],
         [[0,0,0], [1,0,0,0]]];
    PERS wobjdata obNEW:=[FALSE,TRUE,"",[[0,0,0], [1,0,0,0]],
         [[0,0,0], [1,0,0,0]]];
    PERS robtarget p1:=*;
    PERS robtarget p2:=*;
    PERS robtarget p3:=*;
    PERS pose pe1a:=[[0,0,0],[1,0,0,0]];
    PERS pose pe1b:=[[0,0,0],[1,0,0,0]];
    PERS pose pe2a:=[[0,0,0],[1,0,0,0]];
    PERS pose pe3a:=[[0,0,0],[1,0,0,0]];
    PERS pose pe1:=[[0,0,0],[1,0,0,0]];
    PERS pose pe2:=[[0,0,0],[1,0,0,0]];
    PERS pose pe3:=[[0,0,0],[1,0,0,0]];

    PROC NewPoints()
      PDispOff;
      MoveJ RelTool(p1,0,0,-100), v200, fine, tWeldGun\WObj:= obREF;
      MoveL RelTool(p1,0,0,-50), v200, fine, tWeldGun\WObj:= obNEW;
      MoveL p1, v200, fine, tWeldGun\WObj:= obNEW;
      Stop;
      MoveL RelTool(p2,0,0,-50), v200, fine, tWeldGun\WObj:= obNEW;
      MoveL p2, v200, fine, tWeldGun\WObj:= obNEW;
      Stop;
      MoveL RelTool(p3,0,0,-50), v200, fine, tWeldGun\WObj:=obNEW;
      MoveL p3, v200, fine, tWeldGun\WObj:= obNEW;
      Stop;
      MoveL RelTool(p3,0,0,-50), v200, fine, tWeldGun\WObj:= obNEW;
      MoveJ RelTool(p3,0,0,-100), v200, fine, tWeldGun\WObj:= obREF;
    ENDPROC

    PROC WeldSample()
      MoveJ *, v200, fine, tWeldGun\WObj:=obNEW;
      ! Simulated weld:
```

*Continues on next page*

```
    MoveL *, v200, fine, tWeldGun\WObj:=obNEW;
    MoveL *,v20, z1, tWeldGun\WObj:=obNEW;
    MoveL *,v20, fine, tWeldGun\WObj:=obNEW;
    MoveJ *,v200, fine, tWeldGun\WObj:=obNEW;
  ENDPROC

  PROC SearchSample()
    PDispOff;
    MoveJ *,v200, fine, tWeldGun\WObj:=obREF;
    Search_1D pe1a,*,*, v200, tWeldGun\WObj:=obREF;
    MoveL *,v200, fine, tWeldGun\WObj:=obREF;
    Search_1D pe1b,*,*, v200, tWeldGun\WObj:=obREF;
    MoveL *,v200, fine, tWeldGun\WObj:=obREF;
    Search_1D pe2a,*,*, v200, tWeldGun\WObj:=obREF;
    MoveL *,v200, z10, tWeldGun\WObj:=obREF;
    MoveL *,v200, z10, tWeldGun\WObj:=obREF;
    MoveL *,v200, fine, tWeldGun\WObj:=obREF;
    Search_1D pe3a,*,*, v200, tWeldGun\WObj:=obREF;
    MoveL *,v200, fine, tWeldGun\WObj:=obREF;
    pe1:=PoseAdd(pe1a,pe1b);
    pe2:=PoseAdd(pe1a,pe2a);
    pe3:=PoseAdd(pe1b,pe3a);
    obNEW:=OFrameChange(obREF, p1, p2, p3, pe1, pe2, pe3);
  ENDPROC

  PROC RefPoints()
    PDispOff;
    MoveJ *, v200, fine, tWeldGun\WObj:=obREF;
    MoveL RelTool(p1, 0, 0, -50), v200, fine,
         tWeldGun\WObj:=obREF;
    MoveL p1, v200, fine, tWeldGun\WObj:=obREF;
    Stop;
    MoveL RelTool(p2,0,0,-50), v200, fine, tWeldGun\WObj:=obREF;
    MoveL p2, v200, fine, tWeldGun\WObj:=obREF;
    Stop;
    MoveL RelTool(p3,0,0,-50), v200, fine, tWeldGun\WObj:=obREF;
    MoveL p3, v200, fine, tWeldGun\WObj:=obREF;
    Stop;
    MoveL RelTool(p3,0,0,-50), v200, fine, tWeldGun\WObj:=obREF;
    MoveJ *,v200, fine, tWeldGun\WObj:=obREF;
  ENDPROC
ENDMODULE
```

This page is intentionally left blank

# Index

# Contact us

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

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

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

www.abb.com/robotics

3HAC024845-001, Rev A, en