ABB Robotics

Application manual
SmarTac

Power and productivity
for a better world™

Application Manual

3HAC024845-001

Revision B

SmarTac

Global Application Platform

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

Except as may be expressly stated anywhere in this manual, nothing herein shall be construed as any kind of guarantee or warranty by ABB for losses, damages to persons or property, fitness for a specific purpose or the like.
In no event shall ABB be liable for incidental or consequential damages arising from use of this manual and products described herein.

Additional copies of this manual may be obtained from ABB.
The original language for this publication is English. Any other languages that are supplied have been translated from English.

ABB AB
Robotics Products
SE-721 68 Västerås
Sweden

1 Product Documentation

Hardware manuals

All hardware, robots and controllers, will be delivered with a **Product manual**:

- Safety information
- Installation and commissioning (descriptions of mechanical installation, electrical connections)
- Maintenance (descriptions of all required preventive maintenance procedures including intervals)
- Repair (descriptions of all recommended repair procedures including spare parts)
- Additional procedures, if any (calibration, decommissioning)
- Reference information (article numbers for documentation referred to in Product manual, procedures, lists of tools, safety standards)
- Part list
- Foldouts or exploded views
- Circuit diagrams

Technical reference manuals

The following manuals describe the robot software in general and contain relevant reference information:

- **RAPID Overview**: An overview of the RAPID programming language.
- **RAPID Instructions, Functions and Data types**: Description and syntax for all RAPID instructions, functions and data types.
- **System parameters**: Description of system parameters and configuration workflows.

Application manuals

Specific applications (e.g. 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 (e.g. cables, I/O boards, RAPID instructions, system parameters, CD with PC software)
- How to use the application
- Examples of how to use the application

Operating manuals

This group of manuals is aimed at those having first hand operational contact with the robot, i.e. production cell operators, programmers and trouble shooters. The group of manuals includes:

- **Getting started - IRC5**
- **IRC5 with FlexPendant**
- **RobotStudio**
- **Trouble shooting - IRC5** for the controller and robot

2 Introduction

2.1 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 WSD supports powerful programming tools explained in section *5 User´s Guide*.

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.

2.2 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 38VDC. 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 fixturing costs. It can also help account for part variability that can not otherwise be controlled.

2.3 Requirements Overview

2.3.1 System Prerequisites

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

- BaseWare requirements: 5.06
- Controller requirements: IRC5

SmarTac Package

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

2.3.2 User's Qualifications

Robot programmer

Any competent robot programmer (S4 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.

2.4 Precautions!

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

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

All personnel working with the robot system must be very familiar with the safety regulations outlined in the chapter on Safety in the robot controller User's Guide. Incorrect operation can damage the robot or injure someone

3 Installation

3.1 General

Component list

The following items are supplied with the SmarTac option:

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

Note: The SmarTac™ baseware option may be purchased as stand-alone, without any hardware. Please consult the documentation for your sensor, for the correct hardware setup and diagnostics.

Tools required

The following tools are required to install the SmarTac option:

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

3.2 Software set-up

3.2.1 Compatibility

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

- BaseWare requirements: 5.06 or higher.
- Controller requirements: IRC5

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 stand-alone, 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.

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

3.2.2 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™:

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...
```

```
    -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"
```

Load I/O signals for default case

Note that SmarTac™ 9 does not install any I/O signals in the EIO configuration database. It provides only a mechanism to connect to existing signals in the system. If the robotic system is not a turn-key system, I/O signals will need to be installed in the system.

Below is an example of an I/O configuration file that could be used to load I/O signals for the default case:

```
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"
```

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. Please consult your ESAB retailer on how to sense with the gas nozzle.

```
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"
```


3.2.3 Loading Software

Software is loaded by purchasing the SmarTac™ baseware option.

3.3 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 section *3.2.2 System parameters* for system parameter specifications. Verify that diSE_DET1 is on at this time. If it is not, consult Section *6 Troubleshooting Guide*.
3.	Turn on doSE_REF1. Verify that diSE_DET1 is off at this time. If it is not, consult Section *6 Troubleshooting Guide*.
4.	Using a voltmeter, check that there is about 38VDC at the gas cup when referenced to the fixture (ground). If less than 25VDC is measured, consult Section *6 Troubleshooting Guide*.
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, consult Section *6 Troubleshooting Guide*.
6.	Write a simple test routine using the Search_1D instruction (Consult Section *5 User´s Guide* for programming tips). If the Search_1D instruction is not present in the system, refer back to section *3.2.3 Loading Software* for instructions on loading SmarTac software.
7.	Execute the test routine. The robot should stop when the part is detected. If not, consult Section *6 Troubleshooting Guide*.

4 Application Guide

4.1 Searching Conditions

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, e.g. 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.

4.2 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.

4.3 SmarTac Circuit Characteristics

4.3.1 Interaction with the Welding Equipment and Weldment

SmartTac 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

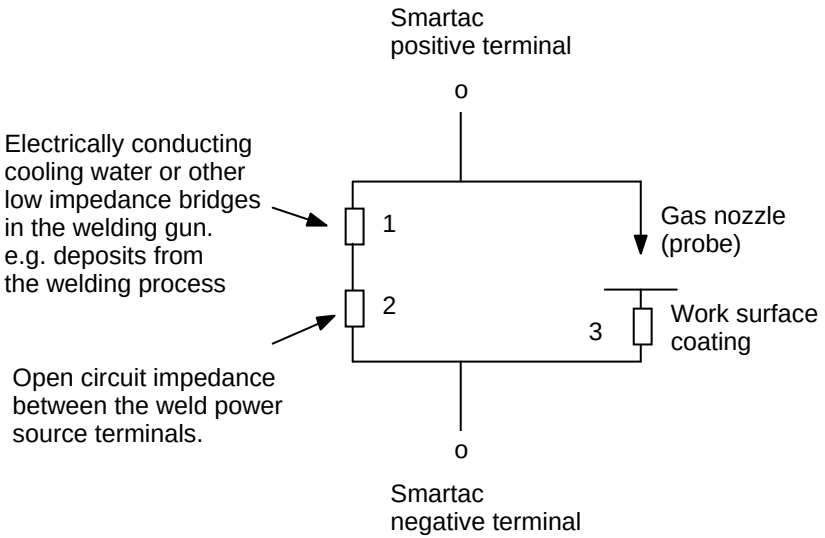


Figure 1 Electric Diagram

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 leas secondary contact

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

Water-cooled gun

When SmarTac is used with a water-cooled gun, the quality of the coolant becomes very important. Impure water, e.g. 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 *Figure 1, Electric Diagram* above, 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.

4.3.2 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.

4.3.3 Sensitivity

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 diagram below.

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

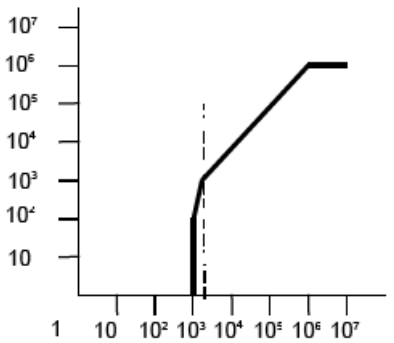


Figure 2 Sensitivity diagram

4.3.4 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.

4.3.5 Signals and Connections

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.

4.3.6 I/O Time line

Illustration

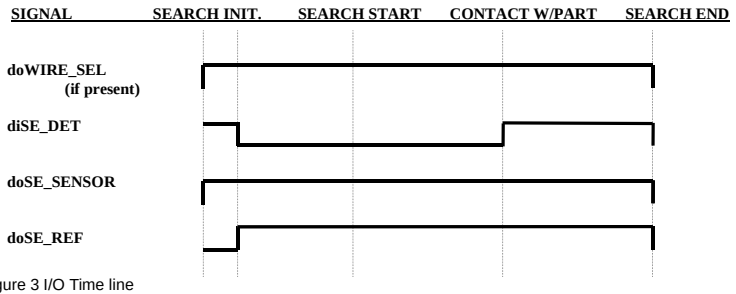


Figure 3 I/O Time line

5 User´s Guide

5.1 Safety

Instructions

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

The SmarTac printed circuit board is typically supplied with 230VAC from the robot main power transformer. This supply power is potentially dangerous. Consequently, the control cabinet door(s) 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 38VDC, low current source. This sensing current is harmless.

The programmer must take all safety precautions presented in the Robot Controller's User's Guide while operating the robotic system.

5.2 Introduction

5.2.1 General

SmartTac system

The SmarTac system available from ABB Welding Systems Division, is a very versatile tool for accurately locating weld seams.

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 don't, 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 part fit-up and/or fixturing is poor, you'll 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. This guide will start with a discussion of these important elements. As the programming examples in the guide become more complex, this topic will be further elaborated on.

3. **What are the RAPID instructions and how are they used to manipulate my weld routines?**

In this guide we will look at several search techniques and the instructions required. Detailed examples will be provided for each of these. Plus the appendices, SmarTac Instructions and SmarTac Functions, provide detailed explanations of the instructions and functions included with the SmarTac software.

5.3 What is a frame

5.3.1 General

Chapter overview

The ABB controller utilizes a number of frames to describe the relationship of the robot tool to the world. The following is a very brief synopsis of what is described in the *Operating manual - RobotStudio*.The next page or so may be difficult to understand at first. Don't worry about it. The exercises later on will help to solidify your understanding of these concepts.

5.3.2 Frames

What are frames?

Frames are coordinate systems.

In the robot controller, frames are described by location and orientation. The location of a frame is defined by three dimensions, x, y, and z. These three dimensions dictate where the origin of the frame is located. The orientation of a frame is described by four quaternions.

When are frames used?

Frames are used in tool definitions, tooldata, robot targets, robtargets, work object definitions, wobjdata, and program displacement definitions, pose.

The tool frame definition describes how the Tool Center Point (TCP) is related to the end of the robot arm, which is related to the world frame through the base of the robot.

A robtarget is related to a program displacement frame, which is related to the object frame of a work object, which is related to the user frame of the work object, which is related to the world frame. Take a look at this figure from the *Operating manual - IRC5 with FlexPendant*:

Illustration

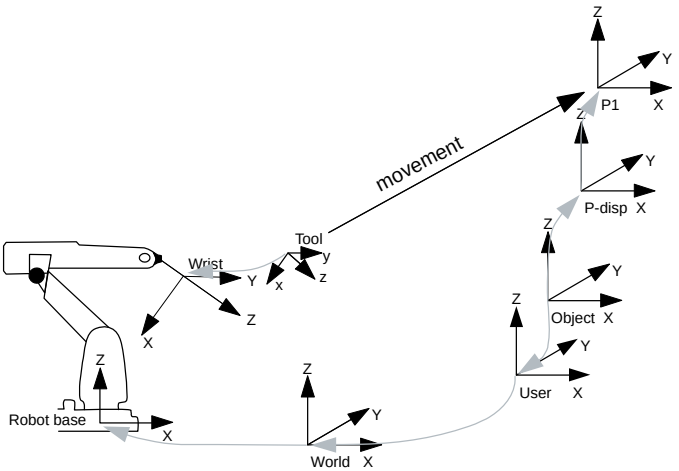


Figure 4 What is a frame

P1 — In the figure, P1 is a robtarget frame.

Pdisp — Pdisp is the current program displacement frame.

Object — Object is the object frame of the current work object.

User — User is the user frame of the current work object.

Note!
Work objects have two frames: object and user.

World — World is the frame that all the other frames are related to in one way or another.

Robot base — The Robot base is defined as the baseframe of the robot.

What happens if a frame is changed?

In the figure above, if any of the frames are changed, the relationship of the TCP to the robtarget changes.

Example

If the user frame of the work object is changed by 10mm in 'X', in relation to the World frame, then the robtarget will move 10mm in 'X', in relation to the World frame. If the program displacement is changed by 10mm in 'X', in relation to the object frame, then the robtarget will move 10mm in 'X', in relation to the object frame.

5.3.3 Exercise 1, Program Displacement

About the exercise

This exercise demonstrates how a program displacement works.

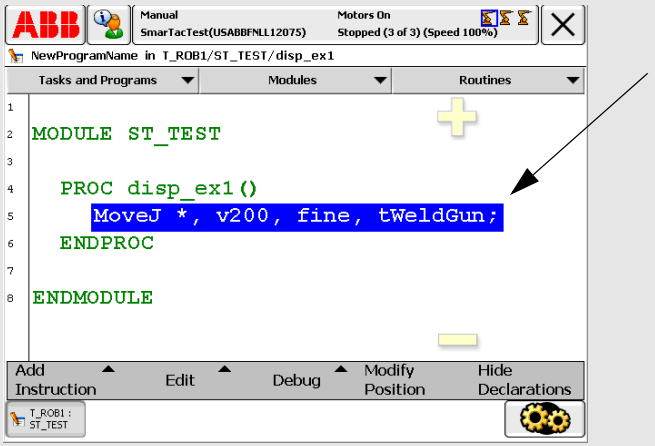
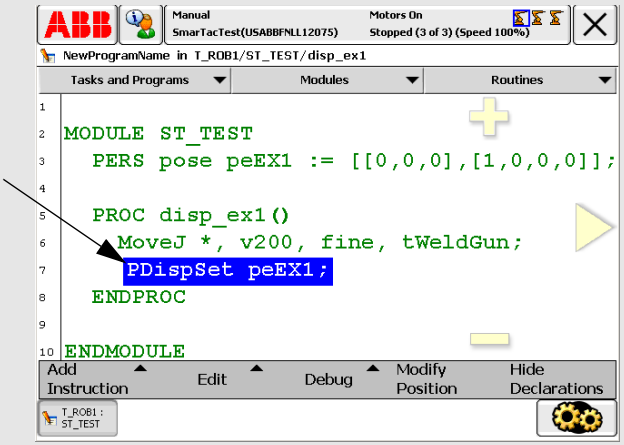
Note!
The exercises later in this guide will not be as detailed as this one.
Please take the time to understand this exercise before attempting others.

If the SmarTac module has not been loaded, instructions are found in section *3, Installation* of this manual.

All exercises assume that the SmarTac software and hardware is installed and working properly.

Instruction

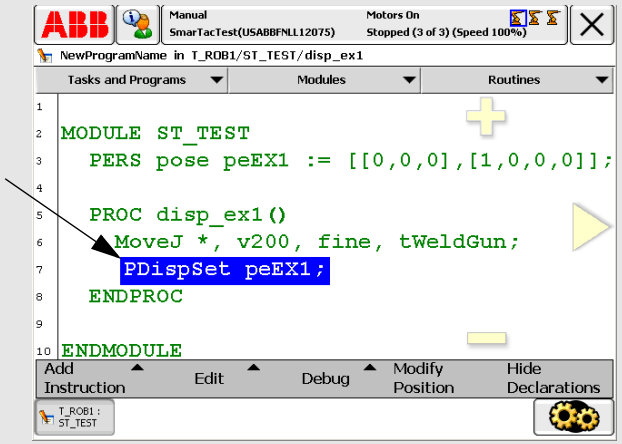
	Action
1.	Create a new program module to work in. Call it “ST_TEST”. Consult the *Operating manual - IRC5 with FlexPendant* if you need help.
2.	Create a new routine in that module and call it “disp_ex1”.
3.	If not already done, define the tool using the five point method or BullsEye. Call the tool “tWeldGun”. Consult the *Operating manual - IRC5 with FlexPendant*, or *Application manual - BullsEye* if you need help. To make programming easier you may want to add in these commands into one of your “Most Common” pick lists: PDispAdd PDispOff PDispSet Search_1D Search_Groove 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. (Perfection is not required here.)

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 you drew. 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: 
7.	Now insert the instruction, PDispSet. 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. The PDispSet instruction requires one argument: a Displacement Frame. When prompted for this data, select new… and create a new pose data type called “peEX1”: 

8. Now insert the instruction, PDispSet. 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.

The PDispSet instruction requires one argument: a Displacement Frame.

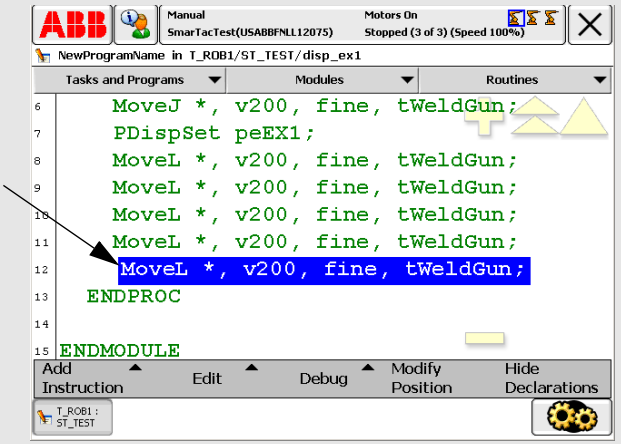
When prompted for this data, select new… and create a new pose data type called “peEX1”:



9. Now jog the robot down to the rectangle so that the tip is just above one of the rectangle corners.

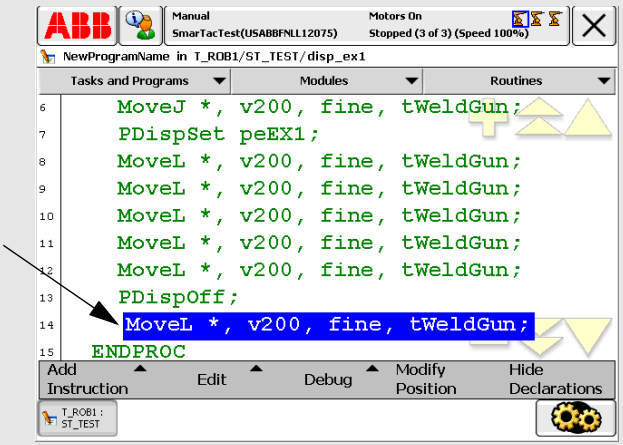
Create a MoveL at this position using tWeldGun and no work object selected.

10. 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:



11. Now insert the instruction: PDispOff.

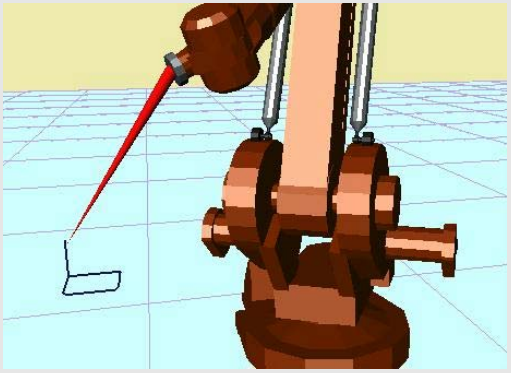
12. Jog the robot away from the rectangle a few inches and insert a final MoveL:



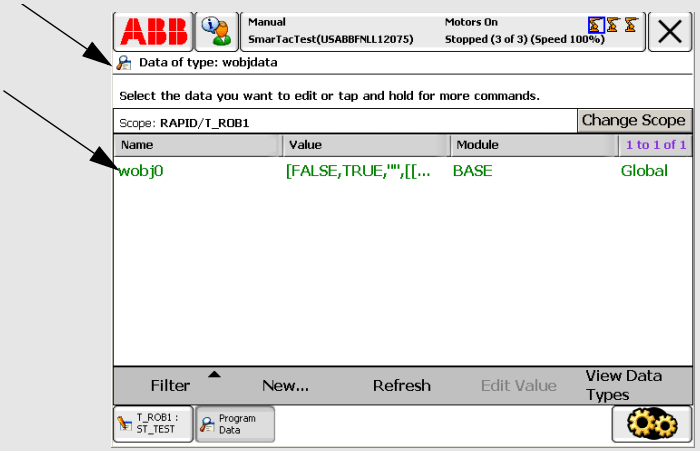
13. Execute this routine from the beginning in manual mode to be sure the program works correctly. The path should look something like this:

This routine executes a simple movement path. Each of the robtargets in the MoveL instructions is related to the World frame through the chain of frames discussed earlier.

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.



14. Take a look at the values in wobj0 by selecting Program Data from the ABB menu. Then choose wobjdata off the list. The list of work objects defined in the system is shown:



15. Push wobj0 and hold. The following will be visible:

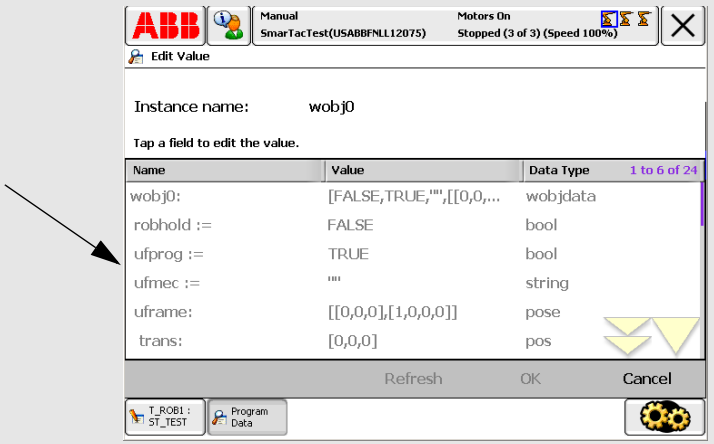
Cursor down and look at the data that is present.

The only other frame that can change the robtarget positions in our “rectangle” routine, is the Displacement Frame.

Note!

That a work object has two frames, the user frame, uframe, and the object frame, oframe.

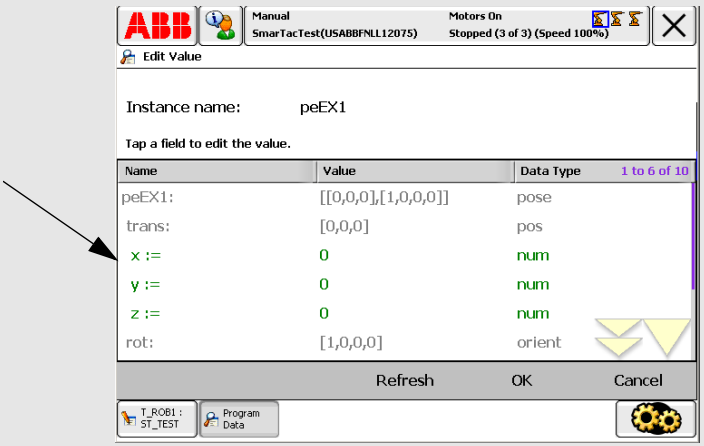
Also note that the values are all zero for the locations, and ones and zeros for the orientations. That's why the work object has no affect on our program. It is the same as the World frame.



16. Hit “Cancel” to get out of the work object window. Then select View Data Types.

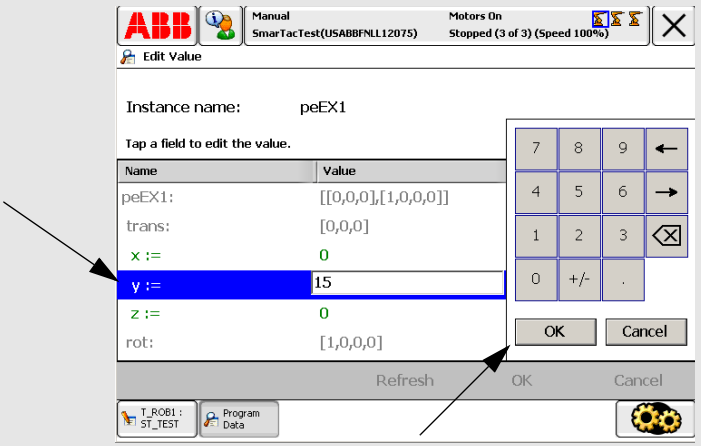
17. Select pose data types.

18. View the data in peEX1 by selecting it and then choose Edit Value from the menu. 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.



19. Move the cursor to the Y value and change the number to 15:

20. Hit OK and run the routine, disp_ex1, again.



What happens?

The moves are shifted 15mm in the positive Y direction. That's 15mm 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 15mm 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 manipulation will be discussed later).

Try making other changes in X, Y, and Z of peEX1. Remember, positive Z will move the rectangle up. Don't use a negative Z, or you'll crash the tool. Use the technique described in step #17.

Important!

Note!

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. This will be discussed later.

Advanced:

Look at the robtarget data using the same technique described in step #17 for looking at pose data. Notice that the robtarget data does not change when the rectangle is moved using the displacement frame.

Do you understand why?

5.4 Using SmarTac to modify a Displacement Frame

5.4.1 General

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 Exercise 1 demonstrated, using program displacements is an easy way of shifting programmed robtargets. The most basic search is a one-dimensional search. A 1-D search finds an offset in one direction.

5.4.2 Search_1D

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.

Structure

The Search_1D instruction has the following structure:

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

Arguments

Search_1D has five required arguments:

	Argument	Example Name	Data Type
1	Result	peOffset	pose
2	StartPoint	p1	robtarget
3	SearchPoint	p2	robtarget
4	Speed	v200	speeddata
5	Tool	tWeldGun	tooldata

Execution

When executed, the robot makes an 'L' move to the StartPoint, 'p1'. The TCP velocity described in Speed determines the speed of this first move. The SmarTac board is activated and motion starts towards the SearchPoint, 'p2'.

The robot will continue past the search point for a total search distance described by twice the distance between StartPoint and SearchPoint, or until the part feature is sensed. Once the part feature is sensed, motion stops, and the displacement data, Result, is stored.

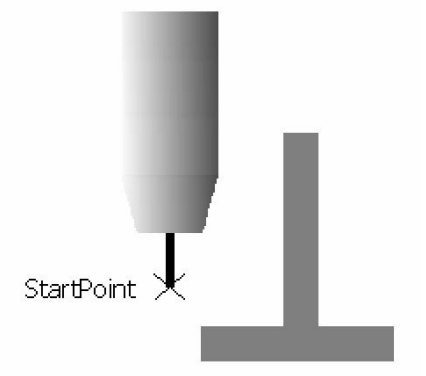


Figure 5 Start point

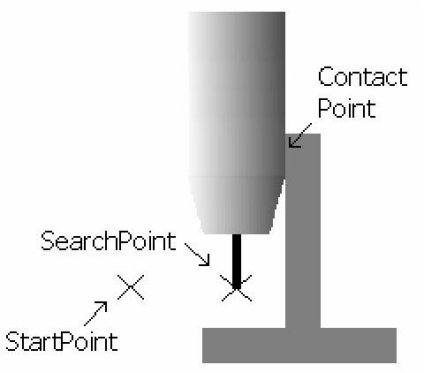


Figure 6 Search point

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.

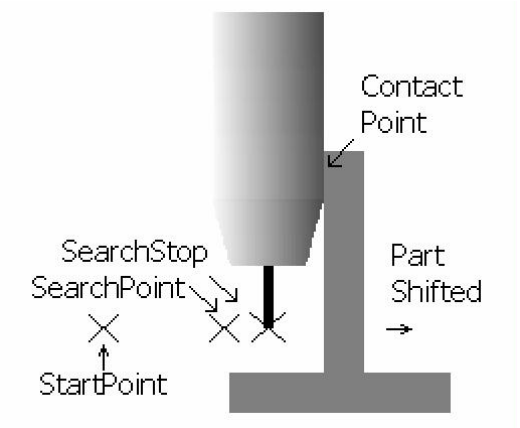


Figure 7 Part shifted

The following exercise demonstrates the instruction, Search_1D.

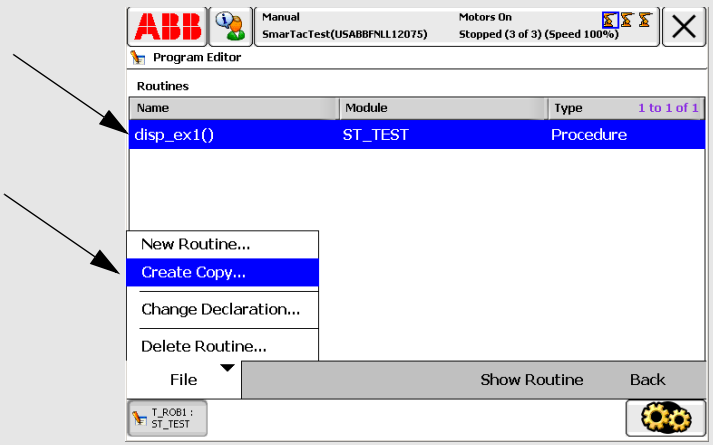
5.4.3 Exercise 2, One Dimensional Search

About the exercise

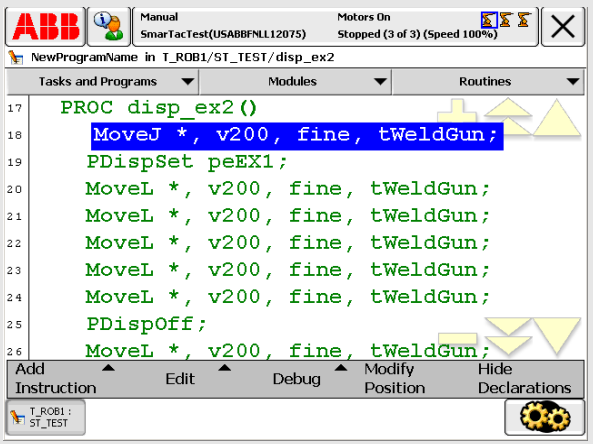
For this exercise the movement routine used in Exercise 1 is used.

Note!
If you did not complete Exercise 1, you will need to write a similar routine

Instruction

	Action
1.	View the routines in the module “ST_TEST”.
2.	Highlight “disp_ex1” and open the File menu at the bottom of the teach pendant screen. Select Create Copy… 

	Action
3.	The controller will offer the default routine name, “disp_ex1Copy”. Change this to “disp_ex2”, then look at the instructions in the new routine:
4.	Remove the paper with the rectangle drawn on it.
5.	Place a metal plate on the table so that a portion of the plate is overhanging:
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.



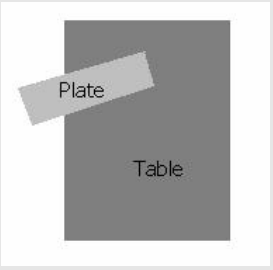
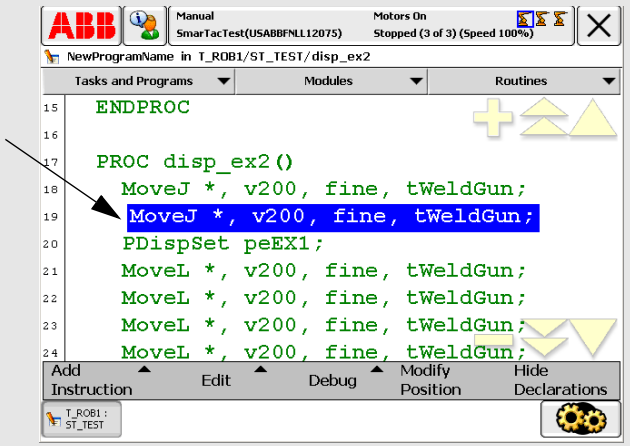
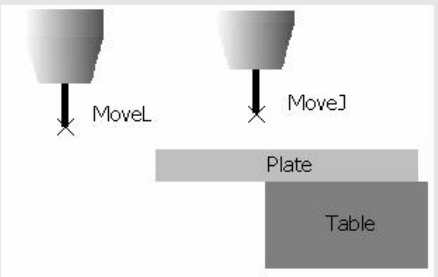
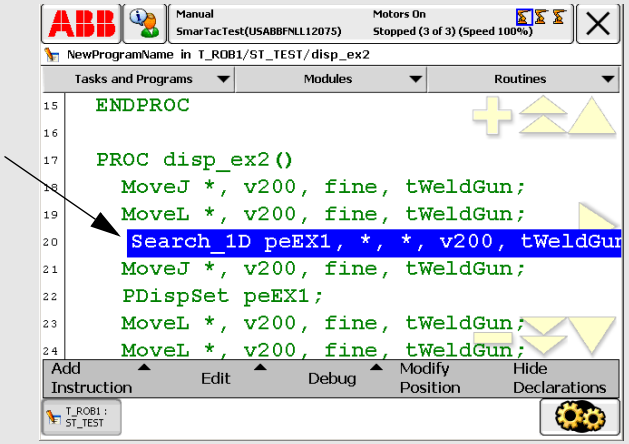
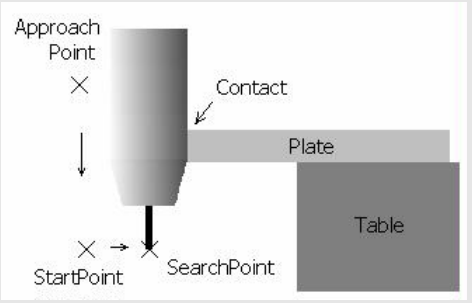
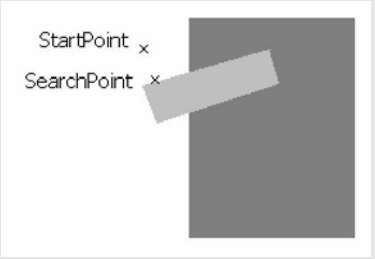


Figure 1 Exercise 2.a “Top View”

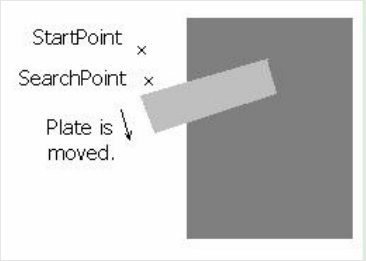
	Action
10.	Toggle the Program Window to instruction mode. If a “Most Common” pick-list with SmarTac instructions was created, select it.
11.	Move the cursor to the first line, MoveJ. Using the Copy and Paste keys at the bottom of the screen, copy the MoveJ and paste it right below it: Between the two MoveJ instructions we will be adding more moves and a search instruction. 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. 
12.	Jog the robot torch so that it is above and past the edge of the plate. Insert a MoveL instruction for this location between the two MoveJ instructions.  Figure 8 Exercise 2. b

	Action
13.	Insert the instruction, Search_1D, after the MoveL. (If no “Most Common” pick-list was created, use ProcCall to find the instruction. When prompted for data, use this data: `Search_1D peEX1,*,*,v200,tWeldGun;` The routine should look like this: 
14.	Jog the robot to the SearchPoint and “ModPos” the second robtarget in the Search_1D instruction. The gas cup should make contact with the plate without deflecting the torch.

	Action
15.	Jog the robot to the StartPoint and “ModPos” the first robtarget in the Search_1D instruction.  Figure 9 Exercise 2.c  Figure 10 Exercise 2.d “Top View”
16.	Toggle the Program Window to the test mode and move the Program Pointer to the Search_1D instruction.
17.	Enable the robot and hit the Fwd key. The robot should move to the StartPoint, then search towards the SearchPoint, until the plate is detected. (See *Figure 9, Exercise 2.c*)

	Action

18.	Toggle the Program Window to instruction mode and using the Copy and Paste keys, copy the MoveL ahead of the Search_1D and paste it after the Search_1D. Your final routine, disp_ex2, should look like this:
	
	`PROC disp_ex2()`
	`  MoveJ *,v200,fine,tWeldGun;`
	`  MoveL *,v200,fine,tWeldGun;`
	`  Search_1D peEX1,*,*,v200,tWeldGun;`
	`  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`

19.	Run the routine from the beginning. The torch should search the plate and then trace out the plate.
20.	Move the plate about 10mm away from the SearchPoint and try running the routine (See *Figure 11, Exercise 2.e*). If the plate was moved in the direction of the search, without any rotation, the torch should still trace out the plate correctly.  Figure 11 Exercise 2.e

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?
2. What errors occur when the plate is moved too far away? Experiment with the error recovery options to see what they do. Consult Section *7.1 Instructions* in this manual to see detailed explanations of the error handing capabilities.

5.4.4 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 Step key. When the robot stops motion with the torch touching the part, move the cursor to the SearchPoint and “mod-pos” the robtarget. This ensures that a search on a “perfect” part will return a displacement that is very close to zero.

5.5 Using SmarTac for Multi-Dimensional Searching

5.5.1 General

About multi-dimensional search

As seen in Example 2, 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.

Limitation of 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.

Example 3 demonstrates this for a two-dimensional search.

5.5.2 Exercise 3, Two Dimensional Search

About the exercise

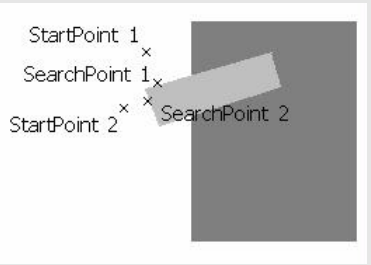
In this exercise the Search_1D instruction will be used twice to determine a two-dimensional shift in a plate on a table.

Instruction

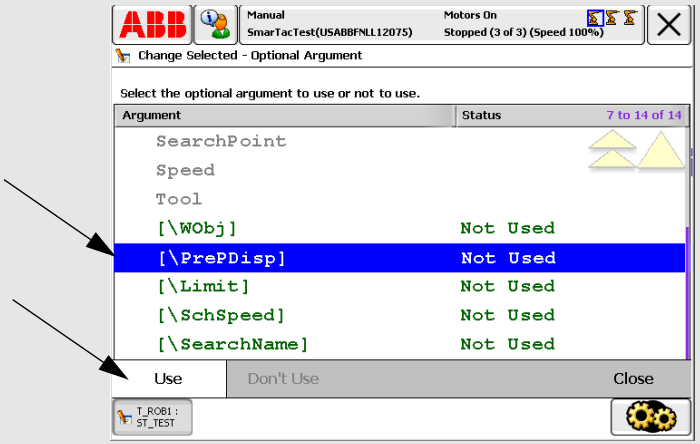
	Action
1.	View the routines in the module, ST_TEST.
2.	Highlight disp_ex2 and duplicate it. Name the new routine disp_ex3.
3.	Toggle the Program Window to test mode.
4.	Move the Program Pointer to the new procedure, disp_ex3.
5.	Move the Program Pointer to the PDispOff instruction near the end of the routine.
6.	Enable the robot and step forward once to execute this 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.	Enable the robot and step forward 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, enable the robot, and step forward until the search is complete, and the robot stops at the MoveL immediately following the Search_1D instruction.
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 MoveL created in step #9.
12.	Copy the MoveL created in step #9 and insert it after the last Search_1D.

	Action

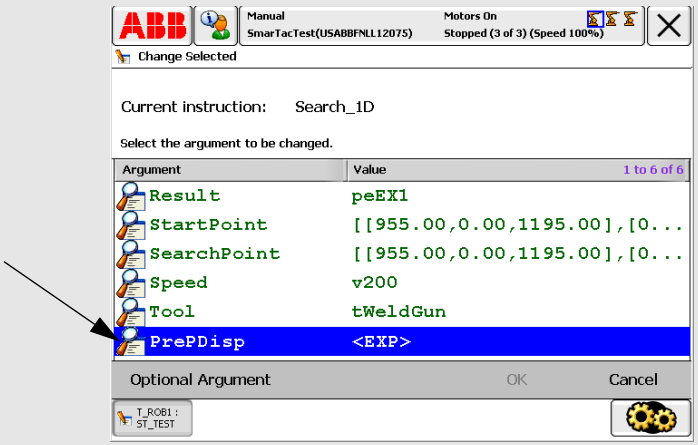
13.	The routine should now look like this:
	
	`PROC disp_ex3()`
	`  MoveJ *,v200,fine,tWeldGun;`
	`  MoveL *,v200,fine,tWeldGun;`
	`  Search_1D peEX1,*,*,v200, tWeldGun;`
	`  MoveL *,v200,fine,tWeldGun;`
	`  MoveL *,v200,fine,tWeldGun; Z New MoveL from step #9`
	`  Search_1D peEX1,*,*,v200,tWeldGun; Z New Search_1D`
	`  MoveL *,v200,fine,tWeldGun; Z Copy of MoveL from step #9`
	`  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`

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":  Figure 12 Exercise 3.a
15.	Highlight the second Search_1D instruction. And hit the Enter key.
16.	Press the OptArg key to look at the optional arguments.

	Action
17.	Move the cursor down to the optional argument named, [\PrePDisp], and Use it.



18.	Select OK.
19.	Select the new argument to open the new window:



	Action

20.	From the list of available pose data select peEX1. Press OK. The routine should look like this:
	
	`PROC disp_ex3()`
	`  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;`
	`  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`

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 10mm in any direction and re-execute the routine. The torch should trace out the plate.

Questions:

1. Why is it necessary that the PrePDisp be 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?
   Hint: Look at the Search_1D section of section *7.1 Instructions*.
3. What would happen if the [\NotOff] argument was added to the second search and the next section of the routine had a welding instruction?
   Hint: Look in section *7.1 Instructions*.
4. Version 7.0 only: Why must there always be at least one Move instruction between two searches?
   Hint: 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
```

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

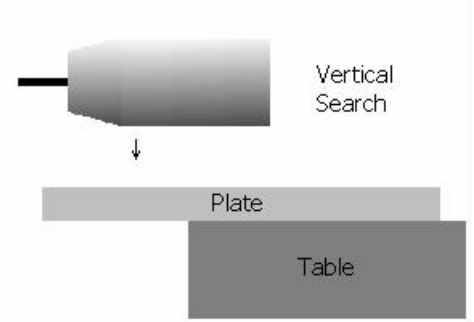


Figure 13 Exercise 3.b

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

5.6 Using SmarTac to Determine Simple Rotational Changes

5.6.1 General

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:

Illustration

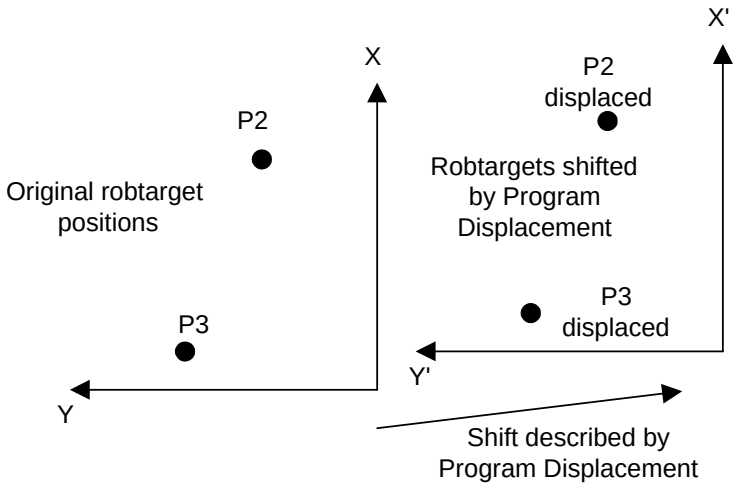
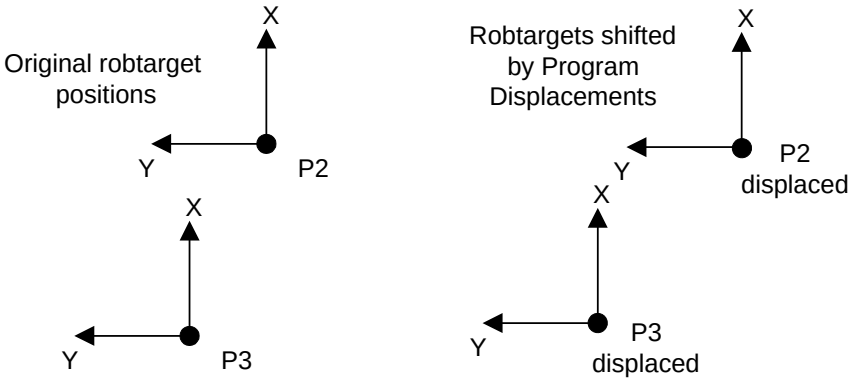


Figure 14 Using SmarTac to determine simple rotational changes

Using this same basic concept, it is possible to search a weld seam that moves both in translation and rotation. Imagine that the robtargets in the above sketch, P2 and P3, 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 rotationally as well.

Illustration



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. Example 4 illustrates this technique.

5.6.2 Exercise 4, Part feature with simple rotation

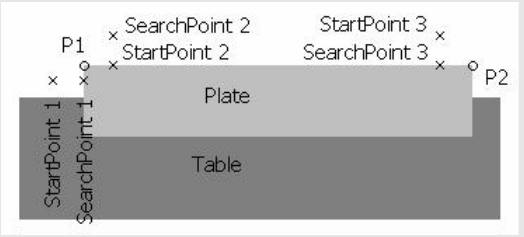
About the exercise

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

Instruction

	Action

1.	Create a new routine called, disp_ex4, that looks like this:
	`PROC disp_ex4()`
	`MoveJ *,v200,fine,tWeldGun;`
	`MoveL *,v200,fine,tWeldGun;`
	`Search_1D peEX1,*,*,v200,tWeldGun; ZSearch 1`
	`MoveL *,v200,fine,tWeldGun;`
	`MoveL *,v200,fine,tWeldGun;`
	`Search_1D peEX2,*,*,v200,tWeldGun\PrePDisp:=peEX1;ZSearch 2`
	`MoveL *,v200,fine,tWeldGun;`
	`MoveL *,v200,fine,tWeldGun;`
	`Search_1D peEX3,*,*,v200,tWeldGun\PrePDisp:=peEX1;ZSearch 3`
	`MoveL *,v200,fine,tWeldGun;`
	`MoveL *,v200,fine,tWeldGun;`
	`PDispSet peEX2;`
	`MoveL P1,v200,fine,tWeldGun; ZCorner 1`
	`PDispSet peEX3;`
	`MoveL P2,v20,fine,tWeldGun; ZCorner 2`
	`PDispOff;`
	`MoveL *,v200,fine,tWeldGun;`
	`ENDPROC`

	Action
	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. The searches should look similar to those used in Example 2 & 3 (See *Figure 9, Exercise 2.c*), but positioned around the plate like this:  Figure 15 Exercise 4.a
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. 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 moves” to clear the plate.
4.	Step through the routine to check the positions. If P1 and P2 are out of position, you can jog them into position and mod-pos them with the program displacement turned on.
5.	Execute the program from the beginning. And watch the robot trace the edge of the plate.
6.	Move the plate in various directions, including rotationally, and execute the routine each time. Does the robot torch follow edge each time? 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 Example 3?
2. Look at the values in each of the program displacements, peEX1, peEX2, and peEX3. What 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?

5.7 Using SmarTac with Work Object Manipulation

5.7.1 General

Weld paths

Sometimes using multiple displacement frames can not provide an easy way of determining a weld seam's location. In Example 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, i.e. a stitch weld, this would not work. There is no displacement information about the intermediate weld points.

Work object

In some instances 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.

Illustration

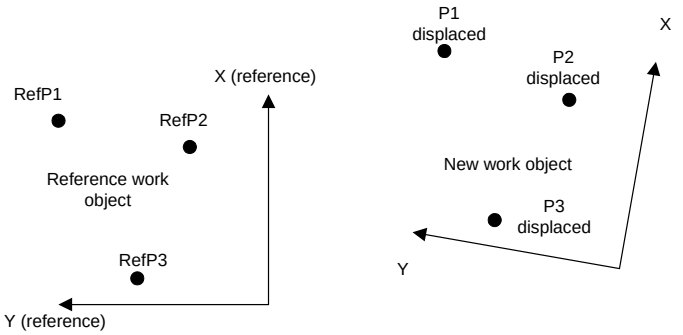


Figure 16 Using SmarTac with Work Object Manipulation a

Important!

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. See *Figure 16, Using SmarTac with Work Object Manipulation a*.

Illustration

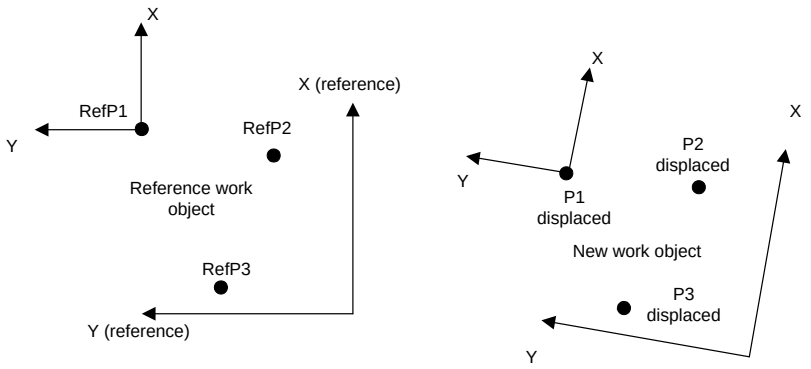


Figure 17 Using SmarTac with Work Object Manipulation b

In this image, 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.

5.7.2 SmarTac Functions

Mathematical functions

Two global mathematical functions are provided in the SmarTac module:

- PoseAdd
- OFrameChange

Detailed explanations of each of these functions can be found in section *7.2 Functions*.

Exercise 5 will illustrate the usage of these mathematical tools.

PoseAdd

PoseAdd is a simple function used to add the transportion 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 a Search_1D.

OFrameChange

The second function is OFrameChange. This function 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);
```

5.7.3 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:

Illustration

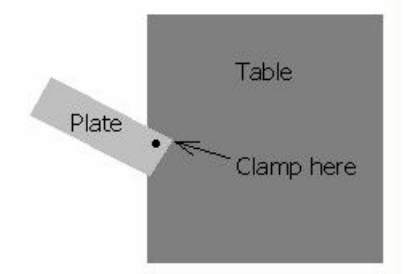


Figure 18 Exercise 5a

Reference sketch

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

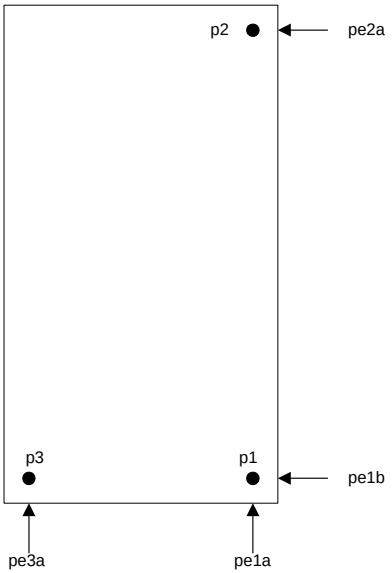
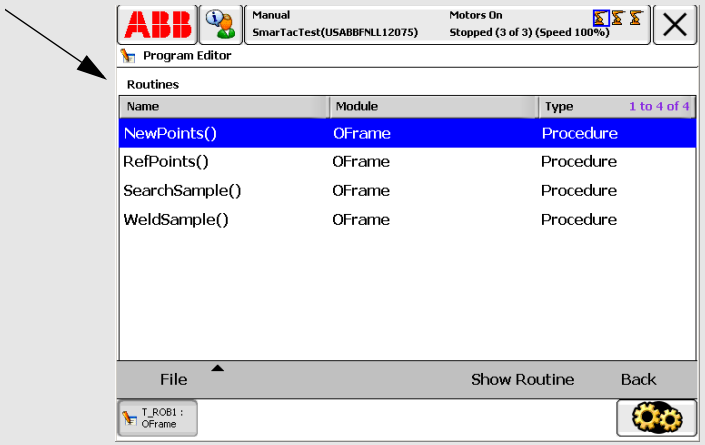
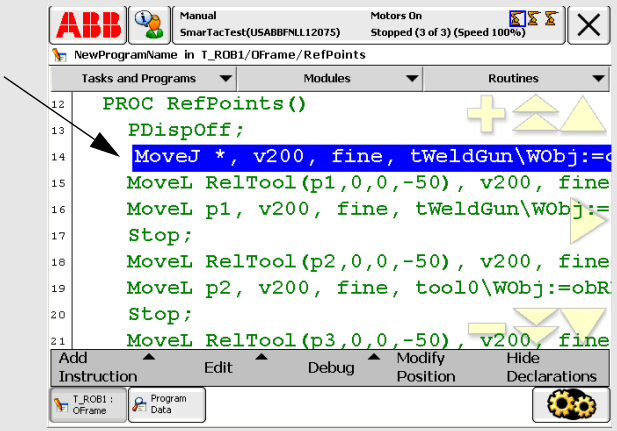
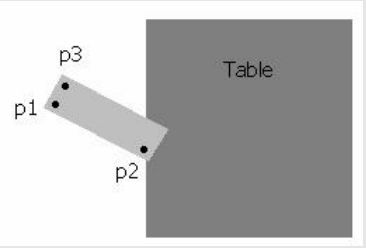


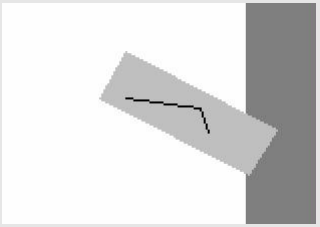
Figure 19 Exercise 5b

Instruction

	Action
1.	Load the program module, “OFrame”, from the provided disk. The disk has been provided to speed up this exercise. If you don't have the disk, you will have to write the routines from scratch using the teach pendant. A printout can be found in section *7.3 Oframe Module Reference* of this manual.

	Action
2.	View the routines in the module. You should see the following routines: 
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 *Figure 19, Exercise 5b* & *Figure 20, Exercise 5.c*.
4.	Move the program pointer to the procedure called, “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 Step key. (This ensures that no displacements are active.)
6.	Jog the robot so that the torch is situated about 150mm above the plate surface and pointing down at the plate.

	Action
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. 
8.	Jog the robot so that the torch TCP is just touching the p1 mark, and modify the second MoveL: `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`…
11.	Jog the robot so the torch is about 150mm 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 labelled: “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.  Figure 20 Exercise 5.c

	Action
15.	Jog the robot down so that the torch gas cup is in a position to search the edge of the plate. Refer to *Figure 19, Exercise 5b* and modify the points the first Search_1D. The search direction is indicated in *Figure 19, Exercise 5b* 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 prescribed by *Figure 19, Exercise 5b*
17.	Test run the SearchSample procedure.
18.	Move the Program Pointer to the routine called: “WeldSample”. WeldSample does not have any ArcL instructions so that ArcWare need not 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 (See *Figure 21, Exercise 5.d*). WeldSample has only two segments, you may add more if you desire.  Figure 21 Exercise 5.d
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.
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 10mm at the end.
29.	Run SearchSample.
30.	Run WeldSample. The path should follow correctly.

	Action
31.	Run NewPoints. The points should be pointed out correctly.

Questions:

1. What work object is used in RefPoints and SearchSample?
2. What work object(s) is (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:

- 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.
- Select obREF as the work object and WObj for the Coordinate System in the Jog 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.
- Go though Example 5 again with the new work object definition.
  How might this help when programming?

5.8 Search_Part

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.

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
```

5.8.1 Exercise 6, Using Search_Part

Instruction

	Action
1.	Create a new procedure in the module, ST_TEST. Call it 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?

5.9 Wire Searching

5.9.1 General

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.

5.9.2 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 *Figure 22, Exercise 7.a*)?

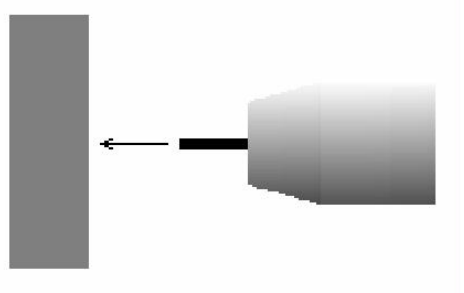
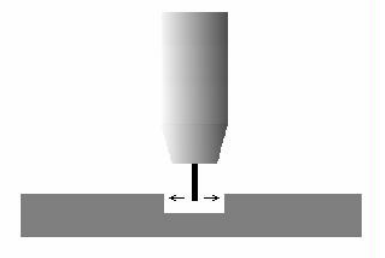


Figure 22 Exercise 7.a

5.10 Searching for a groove

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:



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.

What does Search_Groove require?

Search_Groove actually 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.

Illustration

In use it looks like this:

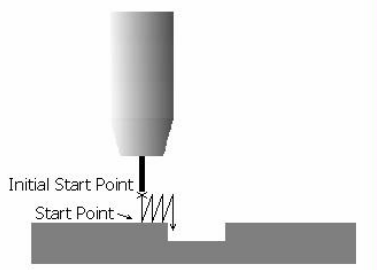


Figure 23 10.b

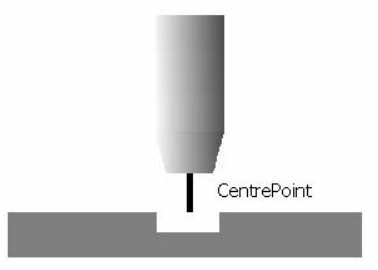


Figure 24 10.c

```
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 10mm 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 15mm above the StartPoint by default.

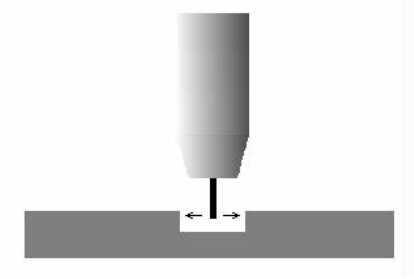


Figure 25 Searching for groove width and groove location, 10. d

Note!

More details of how to program the Search_Groove instruction are found in section *7.1 Instructions*.

5.10.1 Exercise 8, Searching for a groove weld

Instruction

	Action
1.	Build a seam something like the one shown in *Figure 25 Searching for groove width and groove location, 10. d*.
2.	Try programming a Search_Groove instruction using the Search_Groove section of section *7.1 Instructions* as a reference. Remember: Only the wire can be used on a groove search.

Questions:

Use section *7.1 Instructions* 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?

5.10.2 Conclusion

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 section *7.1 Instructions*. You will also find an instruction called PDispAdd which is used with the same effect as the function PoseAdd.

Work objects

Work objects were discussed briefly throughout this manual. Please consult the *Operating manual - IRC5 with FlexPendant* for any questions about how to use work objects to simplify programming. 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.

6 Troubleshooting Guide

6.1 Possible Problems

6.1.1 Symptom 1

The SmarTac board 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. If not check the following:

Possible causes:

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, consult the *Product manual - Robot controller IRC5*.
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 with items 1 and 2 above, replace the SmarTac board.

6.1.2 Symptom 2

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 *6.1.1 Symptom 1* and *6.1.3 Symptom 3*.
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.

6.1.3 Symptom 3

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.

6.1.4 Symptom 4

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 *6.1.5 Symptom 5*). If no voltage is present, check that the SmarTac board is activating properly (See *6.1.2 Symptom 2*). 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.

6.1.5 Symptom 5

Search results are inaccurate.

Possible causes:

1. Search result data is being used improperly. Verify that RAPID search instructions are being used correctly. Refer to section 5 *User´s Guide*.
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 *6.1.3 Symptom 3* 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.

7 Software Reference

7.1 Instructions

7.1.1 Search_1D, One-dimensional search

About Search_ID

Search_1D is an instruction used for tactilely searching a feature 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.

Example

```
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] Datatype: *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] Datatype: *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 Datatype: *pose*

The displacement frame that will be updated

[\SearchStop] Datatype: *robtarget*

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

StartPoint Datatype: *robtarget*

The start point of the search motion.

SearchPoint Datatype: *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 Datatype: *speeddata*

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

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

Tool Datatype: *tooldata*

The tool used during the search.

[\WObj] Datatype: *wobjdata*

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

[\PrePDisp] Datatype: *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] Datatype: *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. (millimetres)

[\SearchName] Datatype: *string*

If selected, the search will be assigned this identifying name. The name will accompany any error messages that are written to the User Error Log.

[\TLoad] Datatype: *loaddata*

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

If the \TLoad argument is set to load0, then the \TLoad argument is not considered and the loaddata in the current tooldata is used instead. For a complete description of the TLoad argument, see MoveL - Moves the robot linearly.

Program execution

When executed, the robot makes an 'L' move 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.

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.

Fault management

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 pre-offset included in the search instruction. A message will be logged in the User Error 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 end point of the search is shifted farther into the part feature. This may give a correct search in cases where the part feature is unusually far away from the search.

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

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;
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;
```

Important!

Important!
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 4mm. If the magnitude of the transportion of peOffset is greater than 4mm 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 User Error Log. See: Fault Management above.

```
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	SmarTac Instructions
Search_Part	SmarTac Instructions
datatype: pose	Rapid Reference datatypes
datatype: wobjdata	Rapid Reference datatypes
datatype: robtarget	Rapid Reference datatypes
MoveL	*Technical reference manual - RAPID Instructions, functions and data types - MoveL*
Definition of loaddata	*Technical reference manual - RAPID Instructions, functions and data types - loaddata*

7.1.2 Search_Groove, Find groove width and location

About Search_Groove

Search_Groove is an instruction used for tactilely searching a “groove” type feature with SmarTac. Searching occurs 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.

Example

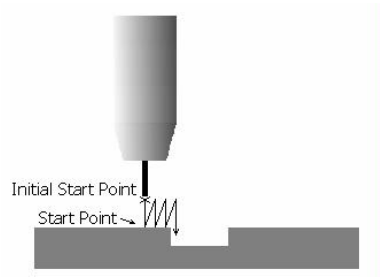


Figure 26 Initial start point

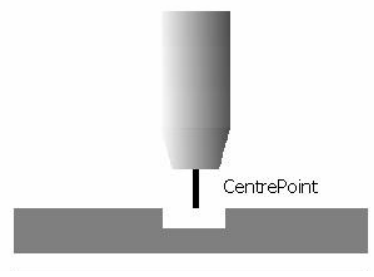


Figure 27 Centre point

```
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 10mm 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 15mm above the StartPoint by default.

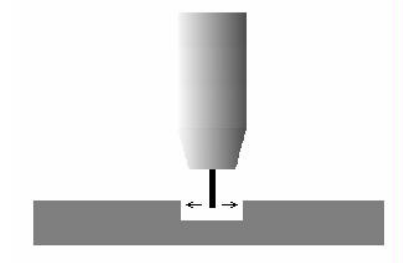


Figure 28 Searching for groove width and groove location

Arguments

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

[\NotOff] Datatype: *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 Datatype: *pose*

The displacement frame that will be updated

GrooveWidth Datatype: *num*

The calculated groove width determined by the search. (millimetres)

[\SearchStop] Datatype: *robtarget*

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

StartPoint Datatype: *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 *Figure 26 Initial start point*.

CentrePoint Datatype: *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 *Figure 27 Centre point*.

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

NomWidth Datatype: *num*

The expected groove width in millimetres. This number will effect the dimensions of the search sequence.

[\NomDepth] Datatype: *num*

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

[\InitSchL] Datatype: *num*

The length of the first search. If selected, this changes the Initial Start Point. The default is 15mm. See *Figure 26 Initial start point*.

Speed Datatype: *speeddata*

The speed data used when moving to the Initial Start Point. The velocity of the search motion is unaffected.

Tool Datatype: *tooldata*

The tool used during the search.

[\WObj] Datatype: *wobjdata*

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

[\PrePDisp] Datatype: *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] Datatype: *string*

If selected, the search will be assigned this identifying name. The name will accompany any error messages that are written to the User Error Log.

[\TLoad] Datatype: *loaddata*

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

If the \TLoad argument is set to load0, then the \TLoad argument is not considered and the loaddata in the current tooldata is used instead. For a complete description of the TLoad argument, see MoveL - Moves the robot linearly.

Program execution

When executed, the robot makes an 'L' move 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 *Figure 26 Initial start point*).

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. 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 *Figure 28 Searching for groove width and groove location*).

The start of both searches is beneath the CentrePoint (See *Figure 27 Centre point* and *Figure 28 Searching for groove width and groove location*). 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.

Fault management

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 pre-offset included in the search instruction. A message will be logged in the User Error 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 end point moved 50%
RETURN	Continues the program with default search result
RAISE	Sends error to calling routine.

When RETRY is selected the end point of the search is shifted farther into the part feature. This may give a correct search in cases where the part feature is unusually far away from the search.

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

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
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 pre-offset included in the search instruction. A message will be logged in the User Error Log.

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;
```

Other variations

Groove search with optional returned robtarget.
The robtarget p3 is updated with the actual groove centerline:

```
Search_Groove\Search-
Stop:=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 User Error Log. See *Fault management* on page 98.

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

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

```
Search_Groove peOff-
set,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 > ','
[ 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	SmarTac Instructions
Search_Part	SmarTac Instructions
datatype: pose	Rapid Reference datatypes
datatype: wobjdata	Rapid Reference datatypes
datatype: robtarget	Rapid Reference datatypes

7.1.3 Search_Part, Search for feature presence

About Search_Part

Search_Part is an instruction used for tactilely searching a feature 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.

Example

```
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] Datatype: *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] Datatype: *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 Datatype: *bool*

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

StartPoint Datatype: *robtarget*

The start point of the search motion.

SearchPoint Datatype: *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 Datatype: *speeddata*

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

Tool Datatype: *tooldata*

The tool used during the search.

[\WObj] Datatype: *wobjdata*

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

[\TLoad] Datatype: *loaddata*

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

If the \TLoad argument is set to load0, then the \TLoad argument is not considered and the loaddata in the current tooldata is used instead. For a complete description of the TLoad argument, see MoveL - Moves the robot linearly.

Program execution

When executed, the robot makes an 'L' move 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.

Fault management

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.

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 User Error Log.

Example

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,tWeld-
Gun\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	SmarTac Instructions
datatype: bool	Rapid Reference datatypes
MoveL	*Technical reference manual - RAPID Instructions, functions and data types - MoveL*
Definition of loaddata	*Technical reference manual - RAPID Instructions, functions and data types - loaddata*

7.1.4 PDispAdd, Add program displacements

About PDispAdd

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

Example

```
PDispAdd pose2;
```

Pose2 is added to the current displacement frame.

Arguments

PDispAdd Result

Result Datatype: *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	SmarTac Instructions
PoseAdd	SmarTac Functions
datatype: pose	Rapid Reference datatypes

7.2 Functions

7.2.1 PoseAdd, Adds the translation portions of pose data

About PoseAdd

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

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

Example

```
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

Description	Data type
The displacement frame	pose

Arguments

PoseAdd (Pose1 Pose2 [\Pose3])

Pose1 Datatype: *pose*

Pose data to be added

Pose2 Datatype: *pose*

Pose data to be added

[\Pose3] Datatype: *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

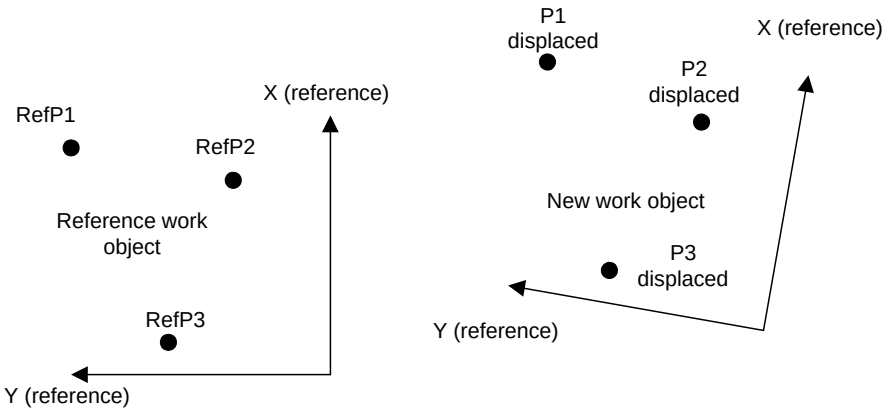
	Described in:
PDispAdd	SmarTac Instructions
datatype: pose	Rapid Reference datatypes

7.2.2 OFrameChange, Create a new shifted object frame

About OFrameChange

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.

Example




```
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

Description	Data type
The new work object.	wobjdata

Arguments

OFrameChange (WObj RefP1 RefP2 RefP3 DispP1 DispP2 DispP3)

WObj Datatype: *robtarget*

Reference work object

RefP1 Datatype: *robtarget*

Reference point number one. (Defined in WObj)

RefP2 Datatype: *robtarget*

Reference point number two. (Defined in WObj)

RefP3 Datatype: *robtarget*

Reference point number three. (Defined in WObj)

DispP1 Datatype: *pose*

Displacement frame affecting reference point RefP1

DispP2 Datatype: *pose*

Displacement frame affecting reference point RefP2

DispP3 Datatype: *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	SmarTac Instructions & Functions
PoseAdd	SmarTac Instructions & Functions
datatype: pose	Rapid Reference datatypes
datatype: wobjdata	Rapid Reference datatypes
datatype: robtarget	Rapid Reference datatypes

7.3 Oframe Module Reference

About OFrame

Exercise 5 uses a program module called “OFrame”.

The module is included on a floppy disk with the manual. 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.

Example Module

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

  PROC WeldSample()
    MoveJ *,v200,fine,tWeldGun\WObj:=obNEW;
    ! Simulated weld:
    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,tWeld-
Gun\WObj:=obREF;
    MoveL p1,v200,fine,tWeldGun\WObj:=obREF;
    Stop;
    MoveL RelTool(p2,0,0,-50),v200,fine,tWeld-
Gun\WObj:=obREF;
    MoveL p2,v200,fine,tWeldGun\WObj:=obREF;
    Stop;
    MoveL RelTool(p3,0,0,-50),v200,fine,tWeld-
Gun\WObj:=obREF;
    MoveL p3,v200,fine,tWeldGun\WObj:=obREF;
    Stop;
    MoveL RelTool(p3,0,0,-50),v200,fine,tWeld-
Gun\WObj:=obREF;
    MoveJ *,v200,fine,tWeldGun\WObj:=obREF;
  ENDPROC
ENDMODULE
```

8 Warranty

8.1 Different warranties

(a)	EQUIPMENT WARRANTY
(b)	SERVICES
(c)	SOFTWARE
(d)	CONDITIONS OF WARRANTY

8.1.1 (a) EQUIPMENT WARRANTY

ABB warrants that the equipment shall be free from defects in material and workmanship for the applicable Warranty Period as described below. The Warranty Period shall be either: one year from date of shipment; or if ABB installs the equipment or accepts a standard ABB Robot Installation Report from the installer, for a period of one year from completion of installation, but not to exceed eighteen months from date of shipment; but in any event the Warranty Period shall not exceed 4000 operating hours. Should any failure to conform with the applicable warranty appear during the specified period, ABB shall, if given prompt notice by Buyer, repair, replace, or modify the defective part or parts. New spare parts and refurbished parts shall be subject to the same warranty as original equipment.

Repairs or replacements pursuant to warranty shall not renew or extend the original equipment warranty period; provided, however, that any such repairs or replacements shall be warranted for the time remaining of the original warranty period or 30 days, whichever is longer. ABB's repair or replacement hereunder shall be exclusive of any removal or installation costs, freight or insurance. ABB shall not be responsible for providing working access to the defect.

8.1.2 (b) SERVICES

ABB warrants that the services of its personnel, if provided, will be performed in a workmanlike manner. Should a failure to comply with this warranty appear within six months from the date of completion of such services, ABB shall, if promptly notified in writing, at its option, either provide the services anew or pay Buyer the cost of procuring such services.

8.1.3 (c) SOFTWARE

ABB warrants for a period of one (1) year from the date of shipment from ABB that the software furnished under this order will perform in accordance with published or other written specifications prepared, approved, and issued by ABB, when used with specifically identified hardware. In any event, ABB makes no representation or warranty, express or implied, that the operation of the software will be uninterrupted or error free, or that the functions contained in the software will meet or satisfy the Buyer's intended use or requirements.

8.1.4 (d) CONDITIONS OF WARRANTY

This warranty shall not apply to any equipment or parts which (i) have been improperly installed, repaired or altered, (ii) have been subjected to misuse, negligence or accident, or (iii) have been used in a manner contrary to ABB operating and maintenance procedures.

THE ABOVE WARRANTIES AND REMEDIES ARE EXCLUSIVE AND IN LIEU OF ANY AND ALL OTHER REPRESENTATIONS, SPECIFICATIONS, WARRANTIES AND REMEDIES EITHER EXPRESS OR IMPLIED, HEREIN OR ELSEWHERE, OR WHICH MIGHT ARISE UNDER LAW OR EQUITY OR CUSTOM OF TRADE INCLUDING WITHOUT LIMITATION WARRANTIES OF MERCHANTABILITY AND OF FITNESS FOR A SPECIFIED OR INTENDED PURPOSE. THE REMEDY SPECIFIED REPRESENTS THE SOLE LIABILITY OF ABB AND THE SOLE REMEDY OF BUYER WITH RESPECT TO OR ARISING OUT OF THE EQUIPMENT OR SERVICES WHETHER BASED ON CONTRACT, TORT (INCLUDING NEGLIGENCE AND STRICT LIABILITY), OR OTHERWISE.

9 Contacts

Please direct all technical and part inquiries to:

ABB AB, Robotics Products, SE-72168 Västerås, Sweden.

Please have IRB serial numbers on hand.

10 Glossary

Word	Description
Baseframe	A frame representing the relationship of a robot's base to the world frame.
Displacement	A frame that represents the spatial relationship between a work object and a robtarget. Displacements are represented by pose data in RAPID. Search results are typically displacements
Electrode Extension	The location at the tip of the consumable welding wire where metal transfer occurs.
Frame	A coordinate system. Work objects, tool data, displacements, robtargets, etc. are all constructed from frames. Frames are represented by pose data in RAPID.
Gas Cup	The outer cylindrical portion of a MIG gun that directs the shielding gas over the weld puddle. Often used as a search-sensing surface. Also called a nozzle.
Nozzle	See Gas Cup
Pose data	A frame in RAPID. It is represented by Cartesian coordinates and quaternions. (x,y,z,q1,q2,q3,q4)
RAPID	A C-based programming language used for ABB robots.
Robtarget	A frame representing a position in space. Movement instructions like MoveL, MoveJ, ArcL, and Search_1D, all require robtargets.
SmarTac	A tactile sensor used to detect inconsistent weld joints and return correctional data to adjust welding position.
Stickout	The distance measure from the contact tip to the weld seam.
TCP	Tool Center Point. The frame representing the spatial relationship between the electrode extension and the wrist of the robot.
Tooldata	A frame representing the TCP in RAPID.
Work object	wobjdata. RAPID data type composed of two frames, the “user frame” and “object frame”. These two frames together represent the spatial relationship between the World and a program displacement frame.
World	All spatial frame relationships are ultimately related to the world frame.

10 Glossary

Contact us

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

3HAC024845-001 Rev B, en