

RCS P-series macro instructions for ABB robot controllers



This page is intentionally left blank.

Contents

1	Introduction	6
1.1	Before you start.	6
1.2	The RCS macros.	6
1.2.1	Installing the macros.	7
1.2.1.1	Using RobotStudio	7
1.2.1.2	Using a USB memory stick.	7
1.2.2	List of macros	8
1.2.2.1	RCS core probing licence	8
1.2.2.2	RCS advanced probing licence	8
1.2.2.3	RCS spindle probing licence	8
2	RCS core probing licence	9
2.1	Test utilities	9
2.1.1	Hello	9
2.1.2	Check data communication.	10
2.2	Manage frames	11
2.2.1	Set work object	11
2.2.2	Set tool frame	11
2.3	Touch parameters management	12
2.3.1	Set the back off distance.	12
2.3.2	Set the move speed	12
2.3.3	Set the double touch behaviour	13
2.3.4	Set the search distance	13
2.3.5	Reset all parameters.	13
2.3.6	Reset the back off distance.	14
2.3.7	Reset the double touch behaviour	14
2.3.8	Reset the move speed	14
2.3.9	Reset the search distance	14
2.4	Probe management.	15
2.4.1	Select mobile probe	15
2.4.2	Select tool setter (fixed probe)	15
2.4.3	Probe power on.	15
2.4.4	Probe power off.	16
2.5	Touch moves	17
2.5.1	Touch in work object direction.	17
2.5.2	Touch in tool frame direction.	18
2.5.3	Touch a target point within the active work object.	19
2.5.4	Touch circle internal and external	20
2.5.5	Touch sphere	22

2.5.6	Record the stem direction	24
2.5.7	Set the stem direction	26
2.5.8	Touch sphere with stem avoidance	28
2.6	Probe calibration	30
2.6.1	Initialise mobile probe calibration	31
2.6.2	Run mobile probe calibration	31
2.6.3	Initialise tool setter calibration	32
2.6.4	Run tool setter calibration using a calibration sphere	33
2.6.5	Run tool setter calibration using a calibrated probe	33
2.7	Simple alignment	34
2.7.1	Initialise simple alignment	34
2.7.2	321 alignment	34
2.7.2.1	Work object	35
2.7.2.2	Tool	36
2.7.3	Plane-plane-plane alignment	36
2.7.3.1	Work object	37
2.7.3.2	Tool	38
2.7.4	Plane-cylinder-cylinder alignment	38
2.7.4.1	Work object	39
2.7.4.2	Tool	40
2.7.5	Plane-plane-point alignment	40
2.7.5.1	Work object	41
2.7.5.2	Tool	42
2.7.6	Sphere-sphere-sphere alignment	42
2.7.6.1	Work object	42
2.7.6.2	Tool	43
2.8	Spherical tool calibration	44
2.8.1	Initialise TCP calibration	44
2.8.2	Run TCP calibration on plane	44
2.8.3	Run TCP calibration on sphere	45
2.9	Residual error statistics	46
3	RCS advanced probing licence	47
3.1	Complex alignment	47
3.1.1	Initialise the datum builder	47
3.1.2	Run the datum builder to a frame	48
3.1.2.1	Work object	48
3.1.2.2	Tool	48
3.1.3	Run the datum builder to a Pose	49
3.2	Best fit feature evaluation	50
3.2.1	Initialise best fit	50
3.2.2	Best fit a plane	51

3.2.3	Best fit a sphere	52
3.2.4	Best fit a cylinder	53
3.2.5	Best fit a 2D circle	54
3.2.6	Best fit a cone	55
3.3	Angle and distance between features	56
3.3.1	Angle between features	56
3.3.2	Distance between features	57
4	RCS spindle calibration licence	58
4.1	Spindle artefact and cutting tool calibration	58
4.1.1	Initialise spindle artefact calibration	60
4.1.2	Run spindle artefact calibration	60
4.1.3	Align spindle artefact or cutting tool	62
4.1.4	Initialise spindle cutting tool measurement	62
4.1.5	Set disc stylus reference height	63
4.1.6	Set nominal spindle cutting tool TCP	64
4.1.7	Set spindle cutting tool TCP	65
4.1.8	Set spindle cutting tool diameter	66
5	Glossary of terms	67
5.1	Approach position	67
5.2	Best fit	67
5.2.1	Least-squares method	67
5.3	Centroid	67
5.4	Co-ordinate system, Frame	67
5.4.1	Tool frame, tool centre point (TCP)	67
5.4.2	Work object	67
5.4.3	Measurement frame	67
5.4.4	Null frame	68
5.5	Datum	68
5.6	Feature	68
5.7	Nominal	68
5.8	Normal	68
5.9	Orthogonal	68
5.10	Parameter	68
5.11	Residual error	68
5.11.1	Absolute maximum residual error	68
5.12	Right-hand rule	69
5.13	Surface	69
5.14	Touch point	69
5.15	Vector	69

1 Introduction

RCS P-series offers an in-process probing solution for robotic automation cells. To communicate with robots, RCS P-series features the RCS probing server application (Server).

To give commands to the Server and your robot, we have developed macro instructions. The macros enable you to manage the Renishaw probes as well as collect and analyse data.

This document provides descriptions of the macros developed for ABB robot controllers.

1.1 Before you start

Ensure you have read and understood the RCS P-series user guide (Renishaw part no. H-6852-8001).

To use this document, you must have working knowledge of robot operation and programming. You will need to know how to:

- Structure robot programs.
- Write robot programs.
- Call and combine macros within a robot program.

There is a “Glossary of terms” on page 67 to define common terms used throughout this document.

1.2 The RCS macros

For each macro, we have provided the structure and description of the arguments it uses. When showing syntax, arguments have a shorthand name. Tables below each macro syntax provide further description of the arguments.

Some macro argument names are appended with “ID”. These are locations within the Server which store data under a number. For some macros, you must provide the ID numbers that you would like the Server to store data under. For other macros, you must reference the IDs where you have previously saved data for the Server to use in calculations.

Surface IDs store position data that a probe collects. Best fit IDs store data that the Server calculates using surface ID data. DatumConfigID is the alignment ID that you set during the Probing Alignment session in RCS Software Suite (see “3.1 Complex alignment” on page 47).

NOTE: Argument names given in syntax tables are for descriptive purposes only. Some arguments are for ABB data types, such as work objects, which must be user-defined by name. These do not need to have the same name as given in syntax tables.

1.2.1 Installing the macros

To install the RCS macros on your robot controller, you will need the RswProc.sys file from the ABB macros folder.

Download the ABB macros folder from www.renishaw.com/softwarelicensing.

Install the macros onto your ABB controller using RobotStudio or a USB memory stick.

1.2.1.1 Using RobotStudio

To install the macros with RobotStudio, follow these steps:

1. Save the RswProc.sys file to a known location on your computer.
2. Open RobotStudio and connect to your robot.
3. In RobotStudio, click “Request Write Access”.
4. On your robot teach pendant, grant the write access.
5. In RobotStudio, right click on your project in the “Controller” window.
6. Click “Load Module”.
7. Go to the folder location of the RswProc.sys file and open it.
8. Edit the “User parameters” section of the file as required for your equipment.
9. In the “Communication” section of the file:
 - a. Change the IP address to the same as the device you are running the Server application on.
 - b. Change the number after the IP address to the port number you set on your device to connect with your robot.

1.2.1.2 Using a USB memory stick

To install the macros with a USB memory stick, follow these steps:

1. Open the RswProc.sys file in a text editor on your computer.
2. Edit the “User parameters” section of the file as required for your equipment.
3. In the “Communication” section of the file:
 - a. Change the IP address to the same as the device you are running the Server application on.
 - b. Change the number after the IP address to the port number you set on your device to connect with your robot.
4. Copy the RswProc.sys file to a USB stick.
5. Plug the USB stick into your robot teach pendant.
6. On your teach pendant, open the ABB menu.
7. Go to Program Editor > Modules > File > Load Module.
8. Go to the folder location of the RswProc.sys file and open it.

1.2.2 List of macros

To use the Renishaw macros, you must have an active RCS probing licence. There are three types of probing licence. Each licence will activate a different collection of macros to use in your robot programs.

To activate your licence, refer to your activation email.

1.2.2.1 RCS core probing licence

1. RswAlignmentInit, page 34
2. RswCheckComm, page 10
3. RswHello, page 9
4. RswPartAlgn321, page 34
5. RswPartAlgnPCC, page 38
6. RswPartAlgnPPP, page 37
7. RswPartAlgnPPPN, page 41
8. RswPartAlgnSSS, page 42
9. RswPowerProbeOff, page 16
10. RswPowerProbeOn, page 15
11. RswProbeCallInit, page 31
12. RswProbeCalRun, page 31
13. RswResetAll, page 13
14. RswResetBackDistance, page 14
15. RswResetDualTouch, page 14
16. RswResetMoveSpeed, page 14
17. RswResetSearchDistance, page 14
18. RswSelectProbeFix, page 15
19. RswSelectProbeMob, page 15
20. RswSetBackDistance, page 12
21. RswSetDualTouch, page 13
22. RswSetFramePart, page 11
23. RswSetFrameTool, page 11
24. RswSetMoveSpeed, page 12
25. RswSetSearchDistance, page 13
26. RswStatRes, page 46
27. RswStemRecord, page 24
28. RswStemSet, page 26
29. RswTCPCallInit, page 32
30. RswTcpcalPlane, page 45
31. RswTcpcalSphere, page 45
32. RswToolAlgn321, page 36
33. RswToolAlgnPCC, page 40
34. RswToolAlgnPPP, page 38
35. RswToolAlgnPPPN, page 42
36. RswToolAlgnSSS, page 43
37. RswToolSetterCallInit, page 32
38. RswToolSetterCalProbeRun, page 33
39. RswToolSetterCalRun, page 33
40. RswTouchCircleExt, page 20
41. RswTouchCircleInt, page 20
42. RswTouchDirPart, page 17
43. RswTouchDirTool, page 18
44. RswTouchPosition, page 19
45. RswTouchSphere, page 22
46. RswTouchSphereStemAvoid, page 28

1.2.2.2 RCS advanced probing licence

1. RswAnlsAngle, page 56
2. RswAnlsDist, page 57
3. RswBestFitCircle2D, page 54
4. RswBestFitCone, page 55
5. RswBestFitCylinder, page 53
6. RswBestFitPlane, page 51
7. RswBestFitSphere, page 52
8. RswDatabagInit, page 50
9. RswDtmBldrInit, page 48
10. RswDtmBldrRunPose, page 49
11. RswPartDtmBldrRun, page 48
12. RswToolDtmBldrRun, page 48

1.2.2.3 RCS spindle probing licence

1. RswSpndlAlign, page 62
2. RswSpndlCallIni, page 60
3. RswSpndlCalRun, page 60
4. RswSpndlInit, page 62
5. RswSpndlNom, page 64
6. RswSpndlRef, page 63
7. RswSpndlSize, page 66
8. RswSpndlTcpc, page 65

2 RCS core probing licence

To use the macros in this section, you will need an active RCS core probing licence.

2.1 Test utilities

Use the test utility macros to check the configuration of your hardware and software. These will confirm whether your robot can connect and communicate with the Server.

2.1.1 Hello

This macro tells the robot to send an integer to the Server. It tells the Server to return an integer to the robot. This will check whether the robot and Server are connected.

The robot will display the result in the FlexPendant. If the connection is unsuccessful, check the communication parameters in the RswProc.sys file on your robot controller match the parameters in the “Server” tab of the RCS probing server application.

Syntax

```
RswHello
```

Example:

```
RswHello; //Robot exchanges an integer with Server.
```

Results

```
Sending request to RPU... //This result confirms the robot and Server are  
Received from RPU: "Hello Robot !" connected.
```

```
Sending request to RPU... //This result means there is a connection error.  
Event Message 41581: Socket Error Troubleshooting required.
```

2.1.2 Check data communication

This macro tells the robot to send values for all data types to the Server. It tells the Server to send the inverse of the data values it receives back to the robot. This will check whether the robot and Server understand each other.

The robot will display the result in the FlexPendant. The Server application will display the inverse results in the “Output” tab. If communication is unsuccessful, check the version number at the start of the RswProc.sys file matches the number in the “About” tab of the Server application. You will need to update the versions if they do not match. Also, check the robot controller you are using is selected in the “Robot” tab of the Server application.

Syntax

```
RswCheckComm
```

Example:

```
RswCheckComm;
```

Results

```
Communication test -> Success
```

```
//This result confirms that the robot and Server  
understand each other. Any other result will require  
troubleshooting.
```

2.2 Manage frames

2.2.1 Set work object

This macro tells the robot to activate the given work object and send the frame data to the Server.

This is useful when you need to set a new active work object without calling an initialisation macro.

Syntax

`RswSetFramePart WorkObject`

Argument	Definition
WorkObject	wobjdata. The name of the work object to activate.

Example:

`RswSetFramePart MyPart;` //Robot sets “MyPart” as the active work object and sends frame data to Server.

2.2.2 Set tool frame

This macro tells the robot to activate the given tool frame and send the frame data to the Server.

This is useful when you need to set a new active tool frame without calling an initialisation macro.

Syntax

`RswSetFrameTool ToolFrame`

Argument	Definition
ToolFrame	tooldata. The name of the tool frame to activate.

Example:

`RswSetFrameTool MyTool;` //Robot sets “MyTool” as the active tool frame and sends frame data to Server.

2.3 Touch parameters management

The RCS touch parameters are contained within the RswProc.sys file (see “1.2.1 Installing the macros” on page 7). The robot uses these parameters when executing a touch move (see “2.5 Touch moves” on page 17).

The parameter values within the RswProc.sys file on your controller are the “default” values.

You can change some parameters from their default values in-program with macros. You can also use macros to reset parameters that you have changed back to their default values.

If you change a touch parameter using a macro, it will remain at that value for all future robot programs until you change it again or reset it. This will not change the default value set in the RswProc.sys file on your controller.

2.3.1 Set the back off distance

This macro sets the back off distance. This is the distance by which the probe will retract after touching a surface. The distance is measured from the surface that the probe touches to the surface of the probe tip.

Syntax

```
RswSetBackDistance BackoffDistance
```

Argument	Definition
BackoffDistance	num. The new back off distance in millimetres (mm).

Example:

```
RswSetBackDistance 10; //Sets the back off distance to 10 mm.
```

2.3.2 Set the move speed

This macro sets the speed the robot uses to move between touch point approach positions (see 5.1 on page 67). This will apply to touch move macros which take more than one touch point (see “2.5 Touch moves” on page 17). The robot will also use this speed to back off from all touch points.

Syntax

```
RswSetMoveSpeed MoveSpeed
```

Argument	Definition
MoveSpeed	speeddata. The new move speed.

Example:

```
RswSetMoveSpeed v100; //Sets the move speed to v100.
```

2.3.3 Set the double touch behaviour

This macro enables or disables double touch behaviour.

With double touch behaviour, a robot will approach surfaces quickly to take a first touch point. This identifies the approximate location of the surface. The robot will then back off, and approach the surface again at a slower speed to capture the data. This is useful when your touch speed is set low, as it will speed up the time to capture data.

The double touch and touch speeds are set in the RswProc.sys file. Changing these values could negatively impact the performance of your robot.

Syntax

```
RswSetDualTouch OnOff
```

Argument	Definition
OnOff	bool. FALSE turns off, TRUE turns on the double touch behaviour.

Example:

```
RswSetDualTouch TRUE; //Turns on double touch behaviour.
```

2.3.4 Set the search distance

This macro sets the probe search distance. This is the maximum distance that the probe will travel when searching for a surface.

Syntax

```
RswSetSearchDistance SearchDistance
```

Argument	Definition
SearchDistance	num. The new search distance in millimetres (mm).

Example:

```
RswSetSearchDistance 15; //Sets the search distance to 15 mm.
```

2.3.5 Reset all parameters

This macro resets all the touch parameters to their default values.

Syntax

```
RswResetAll
```

Example:

```
RswResetAll; //Resets all touch parameters to their default values.
```

2.3.6 Reset the back off distance

This macro resets the back off distance to its default value.

Syntax

```
RswResetBackDistance
```

Example:

```
RswResetBackDistance; //Resets the back off distance to its default value.
```

2.3.7 Reset the double touch behaviour

This macro resets the double touch behaviour to its default value.

Syntax

```
RswResetDualTouch
```

Example:

```
RswResetDualTouch; //Resets the double touch behaviour to its default value.
```

2.3.8 Reset the move speed

This macro resets the move speed to its default value.

Syntax

```
RswResetMoveSpeed
```

Example:

```
RswResetMoveSpeed; //Resets the move speed to its default value.
```

2.3.9 Reset the search distance

This macro resets the search distance to its default value.

Syntax

```
RswResetSearchDistance
```

Example:

```
RswResetSearchDistance; //Resets the search distance to its default value.
```

2.4 Probe management

2.4.1 Select mobile probe

The probe that you mount on your robot is the “mobile” probe.

This macro informs the robot that the mobile probe is in use. This means the robot will use the mobile probe parameters defined in the RswProc.sys file (see “1.2.1 Installing the macros” on page 7). For example, the robot will check the mobile probe input to detect probe triggers. Both the robot and Server will also use the mobile probe tip diameter in calculations.

Syntax

```
RswSelectProbeMob
```

Example:

```
RswSelectProbeMob; //Selects the mobile probe.
```

2.4.2 Select tool setter (fixed probe)

The tool setter probe that you mount in your robot working volume is the “fixed” probe.

This macro informs the robot that the fixed probe is in use. This means the robot will use the fixed probe parameters defined in the RswProc.sys file (see “1.2.1 Installing the macros” on page 7). For example, the robot will check the fixed probe input to detect probe triggers. Both the robot and Server will also use the fixed probe tip diameter in calculations.

Syntax

```
RswSelectProbeFix
```

Example:

```
RswSelectProbeFix; //Selects the tool setter probe.
```

2.4.3 Probe power on

This macro turns on the wireless probe. It will have no effect on a wired probe.

You must select the wireless probe before calling this macro.

Syntax

```
RswPowerProbeOn
```

Example

```
RswPowerProbeOn; //Turns on the selected wireless probe.
```

2.4.4 Probe power off

This macro turns off the selected wireless probe. It will have no effect on a wired probe.

Syntax

```
RswPowerProbeOff
```

Example:

```
RswPowerProbeOff; //Turns off the selected wireless probe.
```


2.5 Touch moves

Touch move macros collect position data from surfaces using a probe. When a probe touches a surface, it will send a trigger signal — this position is a “touch point”. The Server will store the position data from the touch point in a surface ID, which you provide. You can reference surface IDs in other macros to conduct:

- Probe calibration (see 2.6 on page 30)
- Simple alignments (see 2.7 on page 34)
- Best fit feature evaluation (see 3.2 on page 50)
- Spindle artefact and cutting tool calibration (see 4.1 on page 58)

2.5.1 Touch in work object direction

This macro tells the robot to move from its current position until it detects a probe trigger. The robot will travel in the direction that you define within the active work object. If the robot does not detect a probe trigger, it will stop moving once it has travelled the search distance (see 2.3.4 on page 13).

After the probe triggers, the robot will back off from the surface and send the position data to the Server.

Syntax

`RswTouchDirPart SurfaceID, TouchDirectionX, TouchDirectionY, TouchDirectionZ`

Argument	Definition
SurfaceID	num. The surface ID where the Server should store the position data.
TouchDirectionX	num. The X component of the touch direction, within the active work object. The value will not affect the distance the robot travels.
TouchDirectionY	num. The Y component of the touch direction, within the active work object. The value will not affect the distance the robot travels.
TouchDirectionZ	num. The Z component of the touch direction, within the active work object. The value will not affect the distance the robot travels.

Example:

```
MoveL p01, v100, fine, tool; //Robot moves to approach position p01.

RswTouchDirPart 1,1,0,0; //Robot moves from p01 in the +X direction of the active work object
                        until the probe triggers. Server stores position data in surface ID 1.

MoveL p02, v100, fine, tool; //Robot moves to approach position p02.

RswTouchDirPart 2,0,-1,0; //Robot moves from p02 in the -Y direction of the active work object
                        until the probe triggers. Server stores position data in surface ID 2.

MoveL p04, v100, fine, tool; //Robot moves to approach position p04.

RswTouchDirPart 2,0,-1,-1; //Robot moves from p04 in the -Y-Z direction (45°) of the active
                        work object until the probe triggers. Server stores position data in
                        surface ID 2.
```

2.5.2 Touch in tool frame direction

This macro tells the robot to move from its current position until it detects a probe trigger. The robot will travel in the direction that you define within the active tool frame. If the robot does not detect a probe trigger, it will stop moving once it has travelled the search distance (see 2.3.4 on page 13).

After the probe triggers, the robot will back off from the surface and send the position data to the Server.

Syntax

```
RswTouchDirTool SurfaceID, TouchDirectionX, TouchDirectionY, TouchDirectionZ
```

Argument	Definition
SurfaceID	num. The surface ID where the Server should store the position data.
TouchDirectionX	num. The X component of the touch direction, within the active tool frame. The value will not affect the distance the probe travels.
TouchDirectionY	num. The Y component of the touch direction, within the active tool frame. The value will not affect the distance the robot travels.
TouchDirectionZ	num. The Z component of the touch direction, within the active tool frame. The value will not affect the distance the robot travels.

Example:

```
MoveL p01, v100, fine, tool; //Robot moves to approach position p01.

RswTouchDirTool 3,1,0,0; //Robot moves from p01 in the +X direction of the
                           active tool frame until the probe triggers. Server
                           stores position data in surface ID 3.

MoveL p02, v100, fine, tool; //Robot moves to approach position p02.

RswTouchDirTool 4,0,-1,0; //Robot moves from p02 in the -Y direction of the
                           active tool frame until the probe triggers. Server
                           stores position data in surface ID 4.

MoveL p04, v100, fine, tool; //Robot moves to approach position p04.

RswTouchDirTool 4,0,-1,1; //Robot moves from p04 in the -Y+Z direction (45°)
                           of the active tool frame until the probe triggers.
                           Server stores position data in surface ID 4.
```

2.5.3 Touch a target point within the active work object

This macro tells the robot to move to an approach position (see 5.1 on page 67) and then probe a point on a surface. The point is at a nominal position that you define within the active work object.

You must define the touch direction and approach distance. The robot will calculate its approach position using these values. The robot will travel in the touch direction within the active work object until it detects a probe trigger. If the robot does not detect a probe trigger, it will stop moving once it has travelled the search distance (see 2.3.4 on page 13).

After the probe triggers, the robot will back off from the surface and send the position data to the Server.

Syntax

RswTouchPosition SurfaceID, NominalPositionX, NominalPositionY, NominalPositionZ, TouchDirectionX, TouchDirectionY, TouchDirectionZ, ApproachDistance

Argument	Definition
SurfaceID	num. The surface ID where the Server should store the position data.
NominalPositionX	num. The X co-ordinate of the nominal surface point, within the active work object.
NominalPositionY	num. The Y co-ordinate of the nominal surface point, within the active work object.
NominalPositionZ	num. The Z co-ordinate of the nominal surface point, within the active work object.
TouchDirectionX	num. The X component of the touch direction, within the active work object. The value will not affect the distance the robot travels.
TouchDirectionY	num. The Y component of the touch direction, within the active work object. The value will not affect the distance the robot travels.
TouchDirectionZ	num. The Z component of the touch direction, within the active work object. The value will not affect the distance the robot travels.
ApproachDistance	num. The approach distance in millimetres (mm). Measured from the surface of the probe tip to the target point.

Example:

```
MoveL p01, v100, fine, tool; //Robot moves to position p01.

RswTouchPosition 5,50,0,0,1,0,0,10; //Robot will touch a point at (50, 0, 0): Robot starts
//at an approach position 10 mm from the touch point.
//It moves in the +X direction of the active work object
//until the probe triggers. Server stores position data
//in surface ID 5.

MoveL p02, v100, fine, tool; //Robot moves to position p02.

RswTouchPosition 6,0,0,-50,0,-1,0,25; //Robot will touch a point at (0, 0, -50): Robot starts
//at an approach position 25 mm from the touch point.
//It moves in the -Y direction of the active work object
//until the probe triggers. Server stores position data
//in surface ID 6.
```

2.5.4 Touch circle internal and external

These macros tell the robot to measure an internal or external circle by taking four equally-spaced touch points (P1, P2, P3, P4).

The robot will start from its current position. It will move in the given touch direction, within the active work object, until it detects a probe trigger (P1). The robot will then back off from the surface. If the robot does not detect a probe trigger, it will stop moving once it has travelled the search distance (see 2.3.4 on page 13).

The robot will calculate the moves to the approach positions for P2, P3 and P4. After P1, the robot will move to P2–P4 by rotating about the normal direction you provide. The normal is the direction that the circle plane is facing within the active work object. The rotation will follow the right-hand rule, where your thumb is pointing in the normal direction (see “5.12 Right-hand rule” on page 69). To change the movement direction between approach positions, invert your normal direction. Approach positions for P2–P4 will be at the approach distance you define from the surface.

After taking all four touch points, the robot will back off from the surface. The robot will send the position data to the Server after each touch.

This macro is useful for measuring cylinders and circles. The circle can also be evaluated for position and size (see “Best fit a 2D circle” on page 54).

Syntax

```
RswTouchCircleInt SurfaceID, TouchDirectionX, TouchDirectionY, TouchDirectionZ,  
NormalDirectionX, NormalDirectionY, NormalDirectionZ, CircleDiameter,  
ApproachDistance
```

```
RswTouchCircleExt SurfaceID, TouchDirectionX, TouchDirectionY, TouchDirectionZ,  
NormalDirectionX, NormalDirectionY, NormalDirectionZ, CircleDiameter,  
ApproachDistance
```

Argument	Definition
SurfaceID	num. The surface ID where the Server should store the position data (P1, P2, P3, P4).
TouchDirectionX	num. The X component of the touch direction, within the active work object. The value will not affect the distance the robot travels.
TouchDirectionY	num. The Y component of the touch direction, within the active work object. The value will not affect the distance the robot travels.
TouchDirectionZ	num. The Z component of the touch direction, within the active work object. The value will not affect the distance the robot travels.
NormalDirectionX	num. The X component of the circle plane normal direction, within the active work object.
NormalDirectionY	num. The Y component of the circle plane normal direction, within the active work object.
NormalDirectionZ	num. The Z component of the circle plane normal direction, within the active work object.
CircleDiameter	num. The nominal diameter of the circle in millimetres (mm).
ApproachDistance	num. The approach distance in millimetres. Measured from the surface of the probe tip to the surface on which P2, P3 and P4 lie.

Example:

MoveL p05, v100, fine, tool;

//Robot moves to approach position p05.

RswTouchCircleInt 7,1,0,0,0,0,-1,75,10;

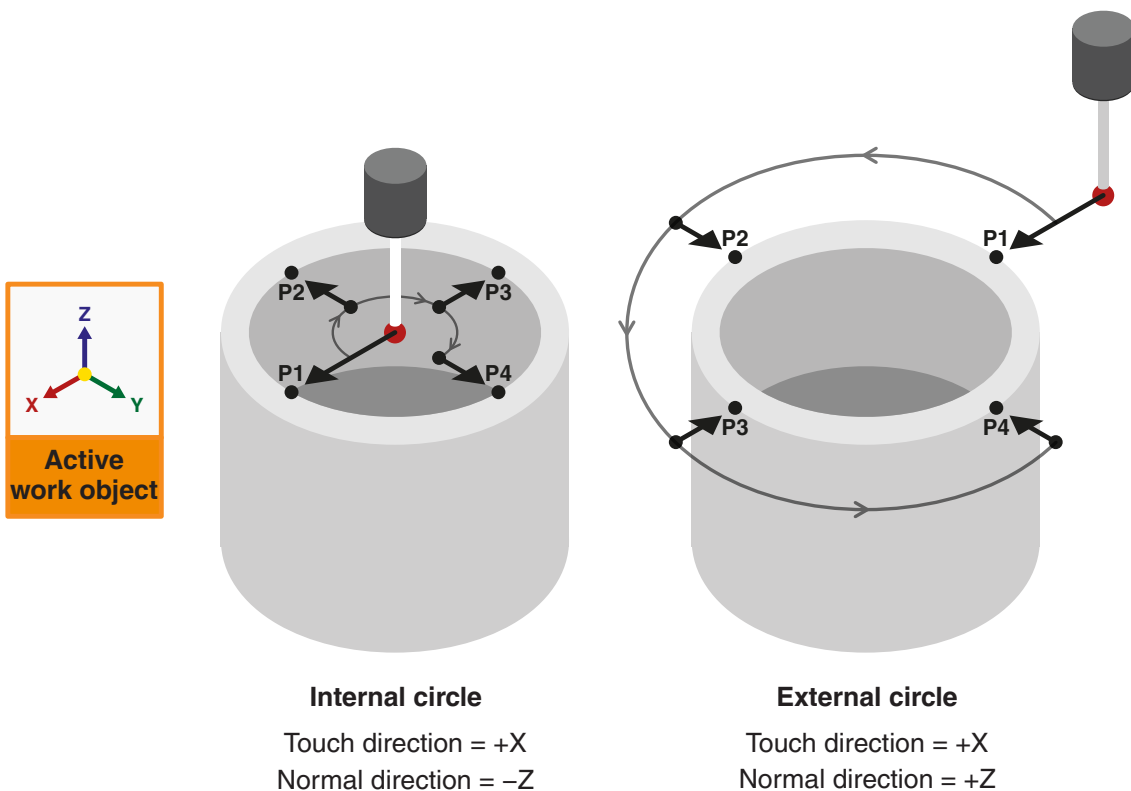
//Robot will measure an internal circle: Robot moves from p05 in the +X direction of the active work object until the probe triggers. The circle has a normal direction of -Z and diameter of 75 mm. Subsequent touch points will have an approach distance of 10 mm. Server stores position data in surface ID 7.

MoveL p06, v100, fine, tool;

//Robot moves to approach position p06.

RswTouchCircleExt 8,1,0,0,0,0,1,85,10;

//Robot will measure an external circle: Robot moves from p06 in the +X direction of the active work object until the probe triggers. The circle has a normal direction of +Z and diameter of 85 mm. Subsequent touch points will have an approach distance of 10 mm. Server stores position data in surface ID 8.



All directions are given within the active work object.

2.5.5 Touch sphere

This macro tells the robot to measure a sphere by taking five touch points (P1, P2, P3, P4, P5).

Before calling this macro:

- Set a nominal frame for your selected probe where the +Z direction points out of the probe tip. For a mobile probe, this should be a tool frame. For a tool setter, this should be a work object.
- Move your robot so that the probe points towards the centre of the sphere.

The robot will start from its current position. If you have selected the mobile probe, the robot will move in the +Z direction of the active tool frame until it detects a probe trigger (P1). If you have selected the tool setter, the robot will move in the –Z direction of the active work object until it detects a probe trigger (P1). The robot will then back off from the surface. If the robot does not detect a probe trigger, it will stop moving once it has travelled the search distance (see 2.3.4 on page 13).

The robot will consider P1 as the sphere “pole”. For the remaining touches, the robot will use the sphere pole and diameter to find the equator of the sphere. It will then take touch points P2, P3, P4 and P5 around the equator. The robot will calculate the moves to the approach positions for P2–P5. Approach positions for P2–P5 will be at the approach distance you define from the surface.

After taking all five touch points, the robot will return to the pole at the approach distance away from P1. The robot will send the position data to the Server after each touch.

This macro is useful for measuring spheres. The sphere can also be evaluated for position and size (see “3.2.3 Best fit a sphere” on page 52).

Syntax

RswTouchSphere SurfaceID, SphereDiameter, ApproachDistance

Argument	Definition
SurfaceID	num. The surface ID where the Server should store the sphere position data (P1, P2, P3, P4, P5).
SphereDiameter	num. The nominal diameter of the sphere in millimetres (mm).
ApproachDistance	num. The approach distance in millimetres (mm). Measured from the surface of the probe tip to the sphere surface on which P2, P3, P4 and P5 lie.

Example:

```

RswSelectProbeMob; //Selects mobile probe (see page 15).

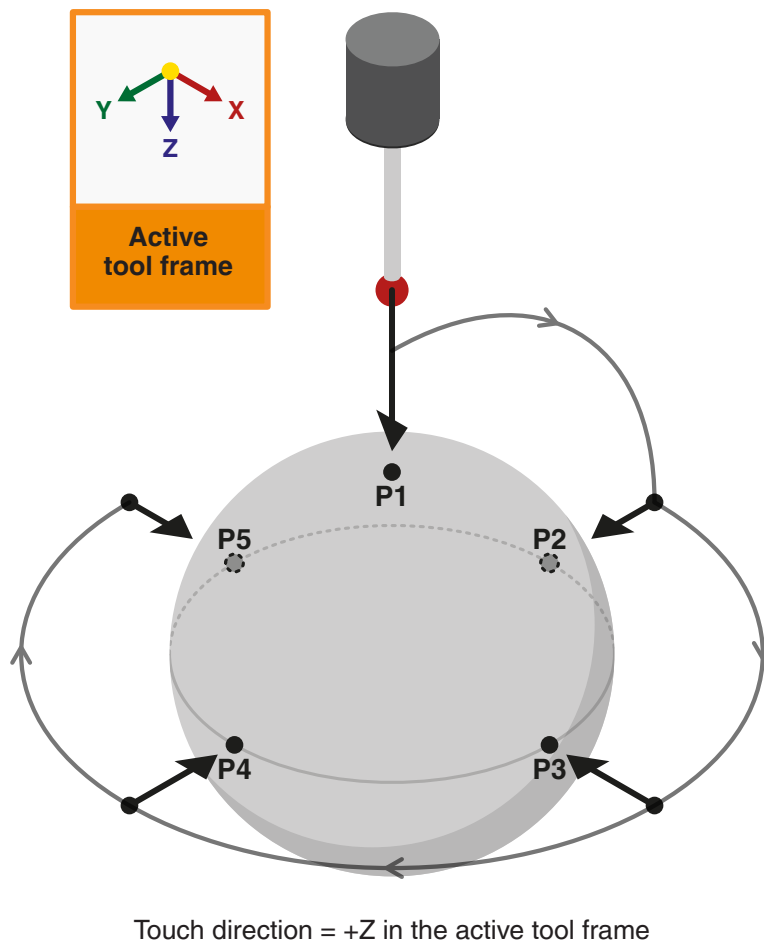
RswPowerProbeOn; //Turns wireless probe on (see page 15).

MoveL p06, v100, fine, tool; //Robot moves to approach position p06.

RswTouchSphere 9, 40, 25.5; //Robot measures a sphere with a diameter of
                               40 mm. It moves from p01 in the +Z direction
                               of the active tool frame until the probe triggers.
                               Subsequent touch points will have an approach
                               distance of 25.5 mm.

                               //Server stores position data in surface ID 9.

```



2.5.6 Record the stem direction

A stem is the supporting structure of a sphere. The stem direction is given from the stem base to the sphere centre. The `RswTouchSphereStemAvoid` macro uses the stem direction to avoid hitting the stem when measuring a sphere (see “2.5.8 Touch sphere with stem avoidance” on page 28).

This macro records directions in the active frames to set the stem direction.

Before calling this macro:

- Set a nominal frame for your selected probe where the +Z direction points out of the probe tip. For a mobile probe, this should be a tool frame. For a tool setter, this should be a work object.
- Align your selected probe with the sphere stem. The probe tip should point through the sphere centre and towards the stem base.

When called, this macro records the –Z direction of the:

- Active tool frame if you have selected the mobile probe (see “2.4.1 Select mobile probe” on page 15).
- Active work object if you have selected the tool setter (see “2.4.2 Select tool setter (fixed probe)” on page 15).

After recording the –Z direction, the macro sets the equivalent direction expressed in the other active frame as the stem direction.

This macro is primarily used during probe and tool setter calibration (see 2.6 on page 30).

Syntax

`RswStemRecord`

Example:

RswProbeCalInit 16, MyPart, MyTool;

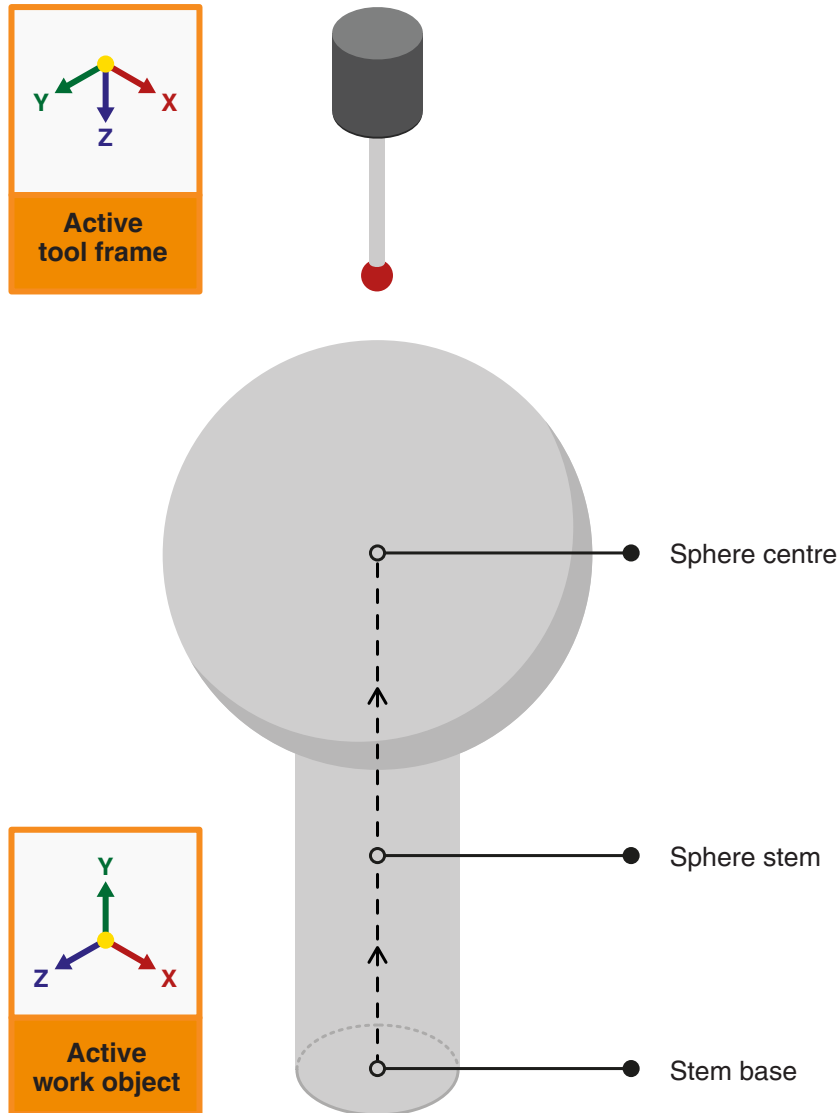
//Server initialises probe calibration. Robot activates
"MyPart" and "MyTool" (see page 31).

RswSelectProbeMob;

//Selects mobile probe (see page 15).

RswStemRecord;

//Records the -Z direction of "MyTool" and sets
the equivalent direction in "MyPart" as the stem
direction.



Selected probe = Mobile
Stem direction = +Y in the active work object

2.5.7 Set the stem direction

A stem is the supporting structure of a sphere. The stem direction is given from the stem base to the sphere centre. The `RswTouchSphereStemAvoid` macro uses the stem direction to avoid hitting the stem when measuring a sphere (see “2.5.8 Touch sphere with stem avoidance” on page 28).

This macro sets the stem direction with a given vector (see 5.15 on page 69).

The vector will define the stem direction within the:

- Active work object if you have selected the mobile probe (see “2.4.1 Select mobile probe” on page 15).
- Active tool frame if you have selected the tool setter (see “2.4.2 Select tool setter (fixed probe)” on page 15).

This macro is primarily used during probe and tool setter calibration (see “2.6 Probe calibration” on page 30).

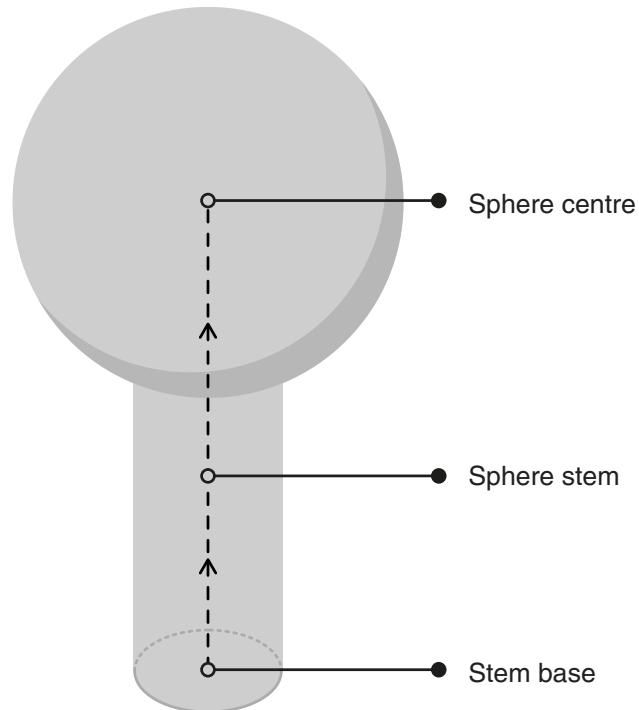
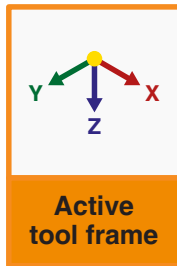
Syntax

```
RswStemSet StemDirectionX, StemDirectionY, StemDirectionZ
```

Argument	Definition
StemDirectionX	num. The X component of the stem direction.
StemDirectionY	num. The Y component of the stem direction.
StemDirectionZ	num. The Z component of the stem direction.

Example:

```
RswProbeCalInit 16, MyPart, MyTool; //Server initialises probe calibration. Robot activates  
"MyPart" and "MyTool" (see page 31).  
  
RswSelectProbeFix; //Selects tool setter (see page 15).  
  
RswStemSet 0,0,-1; //Sets the -Z direction of "MyTool" as the stem  
direction.
```



Selected probe = Tool setter
Stem direction = -Z in the active tool frame

2.5.8 Touch sphere with stem avoidance

This macro tells the robot to measure a sphere by taking five touch points (P1, P2, P3, P4, P5). It will also tell the robot to avoid the stem of the sphere whilst taking the touch points.

Before calling this macro:

- Set a nominal frame for your selected probe where the +Z direction points out of the probe tip. For a mobile probe, this should be a tool frame. For a tool setter, this should be a work object.
- Move your robot so that the probe points towards the centre of the sphere.
- Define the sphere stem direction using either RswStemRecord (see 2.5.6 on page 24) or RswStemSet (see 2.5.7 on page 26).

The robot will start from its current position. If you have selected the mobile probe, the robot will move in the +Z direction of the active tool frame until it detects a probe trigger (P1). If you have selected the tool setter, the robot will move in the –Z direction of the active work object until it detects a probe trigger (P1). The robot will then back off from the surface. If the robot does not detect a probe trigger, it will stop moving once it has travelled the search distance (see 2.3.4 on page 13).

The robot will consider P1 as the sphere “pole”. For the remaining touches, the robot will use the sphere pole and diameter to find the equator of the sphere. It will then take touch points P2, P3, P4 and P5 around the equator. The robot will calculate the moves to the approach positions for P2–P5, avoiding the sphere stem. The stem will lie in the gap between P2 and P5. Approach positions will be at the approach distance you define from the surface.

After taking all five touch points, the robot will return to the pole at the approach distance away from P1. The robot will send the position data to the Server after each touch.

This macro is useful for calibrating probes. The sphere can also be evaluated for position and size (see “3.2.3 Best fit a sphere” on page 52).

Syntax

RswTouchSphereStemAvoid SurfaceID, SphereDiameter, ApproachDistance

Argument	Definition
SurfaceID	num. The surface ID where the Server should store the sphere position data (P1, P2, P3, P4, P5).
SphereDiameter	num. The nominal diameter of the sphere in millimetres (mm).
ApproachDistance	num. The approach distance in millimetres (mm). Measured from the surface of the probe tip to the sphere surface on which P2, P3, P4 and P5 lie.

Example:

```

RswSelectProbeMob; //Selects mobile probe (see page 15).

RswPowerProbeOn; //Turns wireless probe on (see page 15).

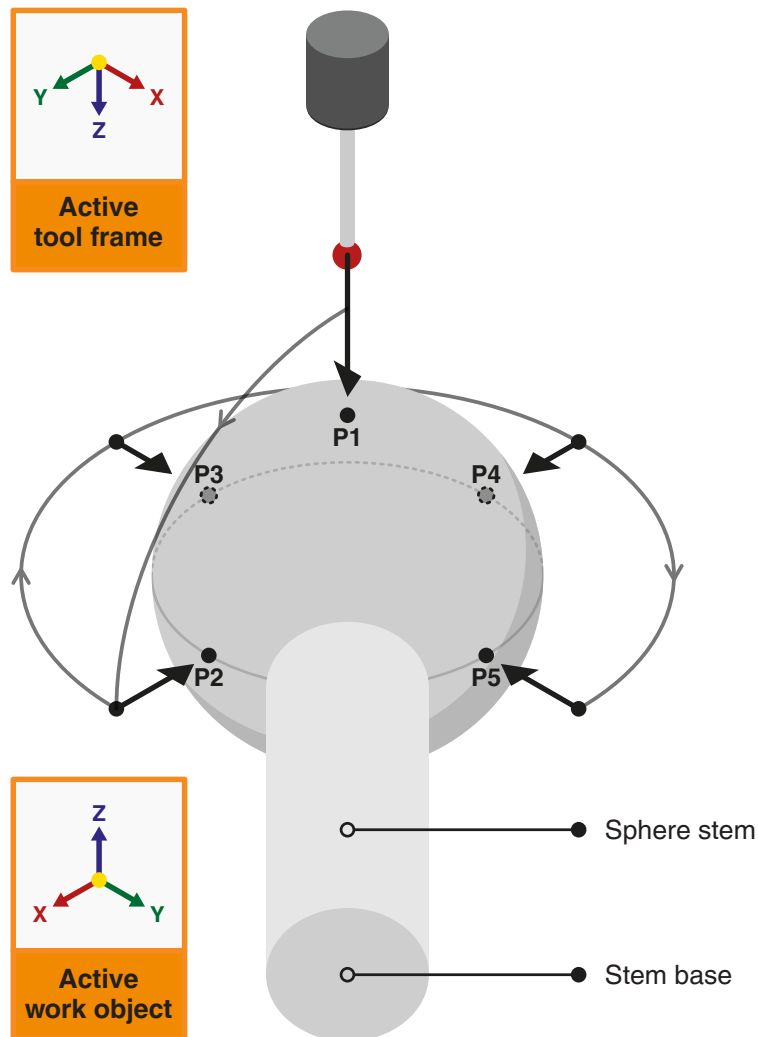
RswStemSet -1,-1,0; //Sets the -X-Y direction of the active work object
                    //as the stem direction.

MoveL p06, v100, fine, tool; //Robot moves to approach position p06.

RswTouchSphereStemAvoid 10,40,25.5; //Robot measures a sphere with a 40 mm diameter.
                                       //It moves from p06 in the +Z direction of the active
                                       //tool frame until the probe triggers. Subsequent
                                       //touch points will have an approach distance of
                                       //25.5 mm and avoid the sphere stem.

                                       //Server stores position data in surface ID 10.

```



Selected probe = Mobile

Touch direction = +Z in the active tool frame

Stem direction = -X -Y in the active work object

2.6 Probe calibration

These macros enable calibration of mobile probes and fixed tool setter probes.

Calibrating a probe will determine the accurate position of its stylus tip centre and diameter. To take reliable measurements with a probe, you must calibrate it first.

You will need to use touch move macros (see 2.5 on page 17) to collect data for calibration.

For mobile probe calibration, take touch points on a fixed calibration sphere.

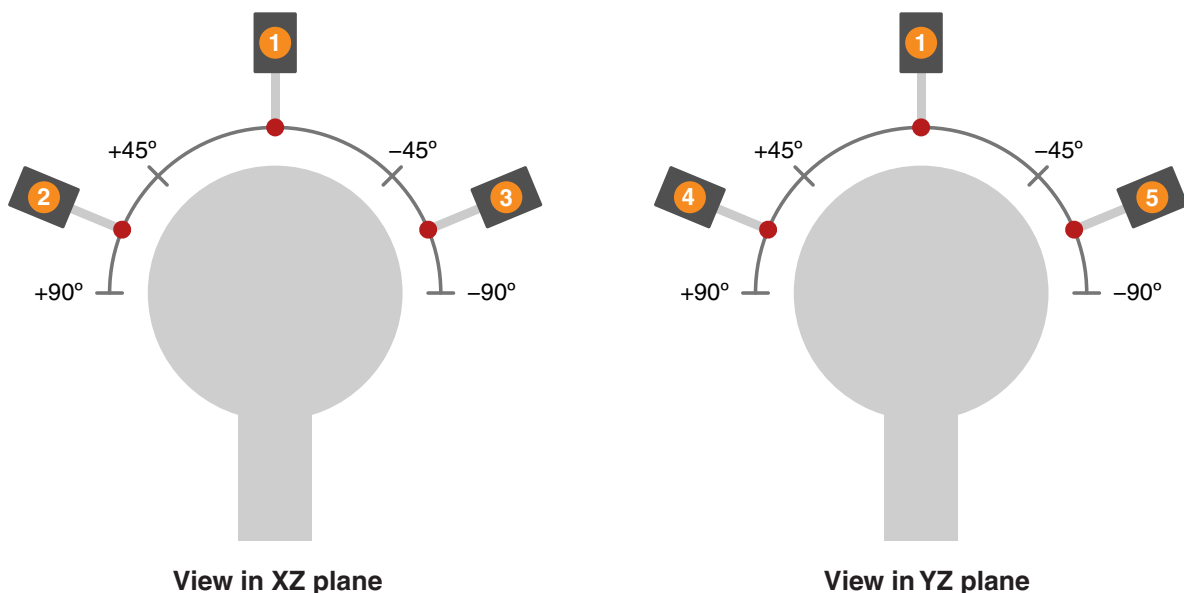
There are two possible ways to gather data for tool setter probe calibration. These are:

- Mount a calibration sphere on your robot arm and take touch points with the tool setter probe.
- Calibrate a mobile probe first. Use the mobile probe to take touch points with the tool setter probe. Ensure the tool setter probe triggers when taking touch points. The mobile probe should not trigger.

When calibrating mobile probes or tool setters, if you are using a calibration sphere, call `RswTouchSphereStemAvoid` (see 2.5.8 on page 28) at least five times to collect enough data. This will equate to at least 25 total touch points. Before each macro call, record a different start position in your robot program. Each start position should be between $\pm 45^\circ$ – 90° about the X and Y axes around the calibration sphere.

When calibrating a tool setter with a mobile probe, you only need to take five total touch points. Use touch move macros, such as “2.5.1 Touch in work object direction” on page 17, to take five separate touch points around the mobile probe with the tool setter probe.

After collecting position data, call the appropriate macro for your chosen calibration method.



Example of five different start positions to collect data for a mobile probe calibration using the `RswTouchSphereStemAvoid` macro.

2.6.1 Initialise mobile probe calibration

This macro tells the Server to discard existing data from the given surface ID, ready to receive new data. It also tells the robot to activate the given work object and tool frame for the probe calibration. The robot sends all active frame data to the Server.

This macro also resets the current diameter of the mobile probe tip to the nominal value in the RswProc.sys file “on your robot controller” (see “1.2.1 Installing the macros” on page 7).

Call this macro in your robot program before probing a calibration sphere with a mobile probe.

Syntax

RswProbeCalInit SurfaceID, WorkObject, ToolFrame

Argument	Definition
SurfaceID	num. The surface ID to discard existing data from.
WorkObject	wobjdata. The name of the work object to activate for the probe calibration.
ToolFrame	tooldata. The name of the tool frame to activate for the probe calibration.

Example:

```
RswProbeCalInit 15,MyPart,MyProbe;           //Server discards existing data from surface ID 15.

                                           //Robot activates “MyPart” and “MyProbe” and
                                           sends all frame data to Server.
```

2.6.2 Run mobile probe calibration

This macro tells the Server to calculate a mobile probe tool centre point (TCP) and tip diameter.

Call this macro in your robot program after probing a calibration sphere with a mobile probe.

The Server will use the data stored in the surface ID you provide for the calculation. The robot will write the result to the tool frame you provide. The robot will set the current diameter of the mobile probe tip to the calculated value. Touch move macros will use this updated value when collecting position data.

Syntax

RswProbeCalRun SurfaceID, NewToolFrame

Argument	Definition
SurfaceID	num. The surface ID which contains position data for the calibration sphere.
NewToolFrame	tooldata. The name of the tool frame where the robot should write the calculated frame data.

Example:

```
RswProbeCalRun 15,MyNewProbe;           //Server calculates a mobile probe TCP using data
                                           stored in surface ID 15.

                                           //Robot writes the calibrated probe TCP to
                                           “MyNewProbe”. Robot also updates the current
                                           probe tip diameter.
```

2.6.3 Initialise tool setter calibration

This macro tells the Server to discard existing data from the given surface ID, ready to receive new data. It also tells the robot to activate the given work object and tool frame for the tool setter calibration. The robot sends all active frame data to the Server.

This macro also resets the current diameter of the fixed probe tip to the nominal value in the RswProc.sys file “on your robot controller” (see “1.2.1 Installing the macros” on page 7).

Call this macro in your robot program before probing either:

- A calibration sphere with a tool setter.
- A calibrated mobile probe with a tool setter.

Syntax

```
RswTool setterCalInit SurfaceID, WorkObject, ToolFrame
```

Argument	Definition
SurfaceID	num. The surface ID to discard existing data from.
WorkObject	wobjdata. The name of the work object to activate for the tool setter calibration.
ToolFrame	tooldata. The name of the tool frame to activate for the tool setter calibration.

Example:

```
RswToolSetterCalInit           //Server discards existing data from surface ID 16.  
16, MyToolSetter, MyTool;  
  
                               //Robot activates “MyToolSetter” and “MyTool” and sends all frame  
                               data to Server.
```


2.6.4 Run tool setter calibration using a calibration sphere

This macro tells the Server to calculate a tool setter probe work object and tip diameter.

Call this macro in your robot program after probing a calibration sphere with a tool setter.

The Server will use the data stored in the surface ID you provide for the calculation. The robot will write the result to the work object you provide. The robot will set the current diameter of the tool setter probe tip to the calculated value. Touch move macros will use this updated value when collecting position data.

Syntax

```
RswToolSetterCalRun SurfaceID, WobjType, NewWorkObject
```

Argument	Definition
SurfaceID	num. The surface ID which contains position data for the calibration sphere.
WobjType	alias. 0 or UF for a user frame, 1 or OF for an object frame.
NewWorkObject	wobjdata. The name of the work object where the robot should write the calculated frame data.

Example:

```
RswToolSetterCalRun 16, //Server calculates a tool setter user frame using data stored in surface
UF, MyNewToolSetter; ID 16.
```

```
//Robot writes result to the user frame of "MyNewToolSetter". Robot
also updates the current tool setter tip diameter.
```

2.6.5 Run tool setter calibration using a calibrated probe

This macro tells the Server to calculate a tool setter probe work object and tip diameter.

Call this macro in your robot program after probing a calibrated mobile probe with a tool setter.

The Server will use the data stored in the surface ID you provide for the calculation. The robot will write the result to the work object you provide. The robot will set the current diameter of the tool setter probe tip to the calculated value. Touch move macros will use this updated value when collecting position data.

Syntax

```
RswToolsetterCalProbeRun SurfaceID, WobjType, NewWorkObject
```

Argument	Definition
SurfaceID	num. The surface ID which contains position data for the calibrated mobile probe.
WobjType	alias. 0 or UF for a user frame, 1 or OF for an object frame.
NewWorkObject	wobjdata. The name of the work object where the robot should write the calculated frame data.

Example:

```
RswToolsetterCalProbeRun //Server calculates a tool setter user frame using data stored in surface ID
16, UF, MyNewToolSetter; 16.
```

```
//Robot writes result to the user frame of "MyNewToolSetter". Robot also
updates the current tool setter tip diameter.
```

2.7 Simple alignment

Simple alignment macros use feature position data to calculate work objects or tool frames. Simple alignments require three or fewer features. The features can be either a plane, sphere or cylinder.

You will need to use touch move macros (see “2.5 Touch moves” on page 17) to probe surfaces and assign the position data to surface IDs. Reference these surface IDs in the alignment macros.

The Server will best fit the referenced features when calculating a frame. The Server will also adjust its calculations based on the calibrated diameter of the selected probe tip (see “2.6 Probe calibration” on page 30).

2.7.1 Initialise simple alignment

This macro tells the Server to discard existing data from the given surface IDs, ready to receive new data. It also tells the robot to activate the given work object and tool frame for the alignment. The robot sends all active frame data to the Server.

Call this macro in your robot program before probing surfaces for an alignment.

Syntax

```
RswAlignmentInit SurfaceID1, SurfaceID2, SurfaceID3, WorkObject, ToolFrame
```

Argument	Definition
SurfaceID1	num. The first surface ID to discard existing data from.
SurfaceID2	num. The second surface ID to discard existing data from.
SurfaceID3	num. The third surface ID to discard existing data from.
WorkObject	wobjdata. The name of the work object to activate for the alignment.
ToolFrame	tooldate. The name of the tool frame to activate for the alignment.

Example:

```
RswAlignmentInit //Server discards existing data from surface IDs 1, 2 and 3.  
1, 2, 3, MyPart, MyTool;  
//Robot activates “MyPart” and “MyTool” and sends all frame  
data to Server.
```

2.7.2 321 alignment

This macro tells the Server to calculate a frame using a 321 alignment.

A 321 alignment requires three planes which are nominally orthogonal to each other (see 5.7 and 5.9 on page 68).

The first plane used in the macro requires three measured points (P1, P2, P3). The points must be distributed around the plane, not in a single line. This will define the +Z direction and Z origin.

The second plane used in the macro requires two measured points (P4, P5). This will define the +Y direction and X origin. The Server will use the direction from P4 to P5 to calculate the +Y direction.

The third plane used in the macro requires one measured point (P6). This will define the Y origin.

This macro can be used to calculate a work object or a tool frame. The robot will write the result to the frame you provide.

2.7.2.1 Work object

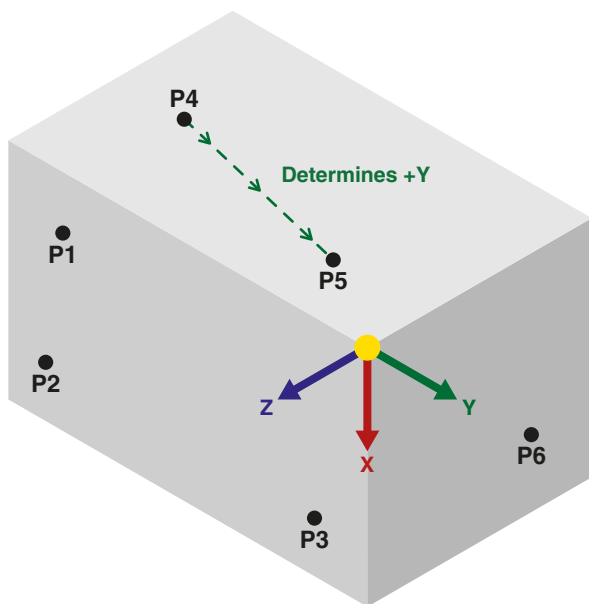
Syntax

RswPartAlgn321 SurfaceID1, SurfaceID2, SurfaceID3, WobjType, NewWorkObject

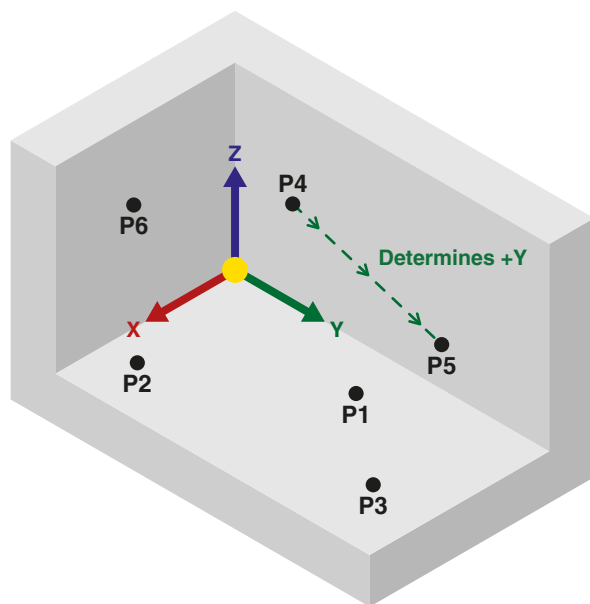
Argument	Definition
SurfaceID1	num. The surface ID which contains position data for the first plane (P1, P2, P3).
SurfaceID2	num. The surface ID which contains position data for the second plane (P4, P5).
SurfaceID3	num. The surface ID which contains position data for the third plane (P6).
WobjType	alias. 0 or UF for a user frame, 1 or OF for an object frame.
NewWorkObject	wobjdata. The name of the work object where the robot should write the calculated frame data.

Example:

```
RswPartAlgn321 10,11,12,UF,MyNewPart; //Server calculates a user frame with a 321
alignment.
//Surface ID 10 contains three points.
//Surface ID 11 contains two points
//Surface ID 12 contains one point.
//Robot writes result to the user frame of
“MyNewPart”.
```



321 alignment outside of a box



321 alignment inside of a box

2.7.2.2 Tool

Syntax

```
RswToolAlgn321 SurfaceID1, SurfaceID2, SurfaceID3, NewToolFrame
```

Argument	Definition
SurfaceID1	num. The surface ID which contains position data for the first plane (P1, P2, P3).
SurfaceID2	num. The surface ID which contains position data for the second plane (P4, P5).
SurfaceID3	num. The surface ID which contains position data for the third plane (P6).
NewToolFrame	tooldata. The name of the tool frame where the robot should write the calculated frame data.

Example:

```
RswToolAlgn321 10,11,12,MyNewTool;           //Server calculates a tool frame with a 321
                                                alignment.

                                                //Surface ID 10 contains three points.
                                                //Surface ID 11 contains two points
                                                //Surface ID 12 contains one point.

                                                //Robot writes frame result to "MyNewTool".
```

2.7.3 Plane-plane-plane alignment

This macro tells the Server to calculate a frame using a plane-plane-plane alignment.

A plane-plane-plane alignment requires three planes which are nominally orthogonal to each other.

All surfaces used in this macro must have at least three measured points to form three planes. The points must be distributed around each plane, not in a single line.

The first plane (Plane 1) used in the macro will define the +Z direction and Z origin.

The second plane (Plane 2) used in the macro will define the +X direction and X origin.

The third plane (Plane 3) used in the macro will define the Y origin.

This macro can be used to calculate a work object or a tool frame. The robot will write the result to the frame you provide.

2.7.3.1 Work object

Syntax

RswPartAlgnPPP SurfaceID1, SurfaceID2, SurfaceID3, WobjType, NewWorkObject

Argument	Definition
SurfaceID1	num. The surface ID which contains position data for the first plane (Plane 1).
SurfaceID2	num. The surface ID which contains position data for the second plane (Plane 2).
SurfaceID3	num. The surface ID which contains position data for the third plane (Plane 3).
WobjType	alias. 0 or UF for a user frame, 1 or OF for an object frame.
NewWorkObject	wobjdata. The name of the work object where the robot should write the calculated frame data.

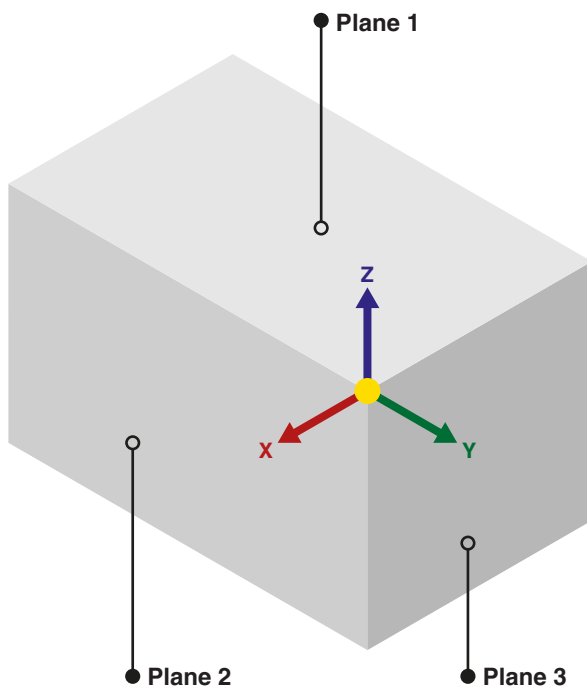
Example:

```
RswPartAlgnPPP 1, 2, 3, OF, MyNewPart;
```

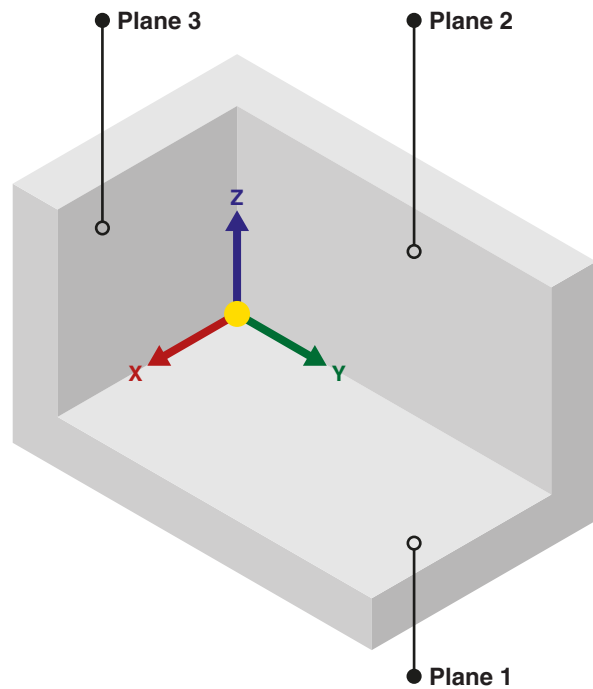
//Server calculates an object frame with a plane-plane alignment.

//Surface IDs 1, 2 and 3 all contain at least three points.

//Robot writes result to the object frame of "MyNewPart".



Plane-plane-plane alignment
outside of a box



Plane-plane-plane alignment
inside of a box

2.7.3.2 Tool

Syntax

```
RswToolAlignPPP SurfaceID1, SurfaceID2, SurfaceID3, NewToolFrame
```

Argument	Definition
SurfaceID1	num. The surface ID which contains position data for the first plane (Plane 1).
SurfaceID2	num. The surface ID which contains position data for the second plane (Plane 2).
SurfaceID3	num. The surface ID which contains position data for the third plane (Plane 3).
NewToolFrame	tooldata. The name of the tool frame where the robot should write the calculated frame data.

Example:

```
RswToolAlignPPP 1, 2, 3, MyNewTool;           //Server calculates a tool frame with a plane-plane-  
                                               plane alignment.  
  
                                               //Surface IDs 1, 2 and 3 all contain at least three  
                                               points.  
  
                                               //Robot writes frame result to "MyNewTool".
```

2.7.4 Plane-cylinder-cylinder alignment

This macro tells the Server to calculate a frame using a plane-cylinder-cylinder alignment.

A plane-cylinder-cylinder alignment requires two cylinders which are normal to a plane (see 5.8 on page 68).

The first surface used in the macro (a plane) requires at least three measured points (P1, P2, P3). The points must be distributed around the plane, not in a single line. This will define the +Z direction and Z origin.

The second and third surfaces are cylinders (C2, C3). You can measure these using a touch circle macro (see 2.5.4 on page 20).

The centreline between the two faces of a cylinder is its axis. The axis of C2 will define the X and Y origin.

A fitted line from the centre of C2 to the centre of C3 will define the +Y direction. The Server will calculate the +X direction as a result of the Y and Z axes.

This macro can be used to calculate a work object or a tool frame. The robot will write the result to the frame you provide.

2.7.4.1 Work object

Syntax

RswPartAlignPCC SurfaceID1, SurfaceID2, SurfaceID3, WobjType, NewWorkObject

Argument	Definition
SurfaceID1	num. The surface ID which contains position data for a plane (P1, P2, P3).
SurfaceID2	num. The surface ID which contains position data for the first cylinder (C2).
SurfaceID3	num. The surface ID which contains position data for the second cylinder (C3).
WobjType	alias. 0 or UF for a user frame, 1 or OF for an object frame.
NewWorkObject	wobjdata. The name of the work object where the robot should write the calculated frame data.

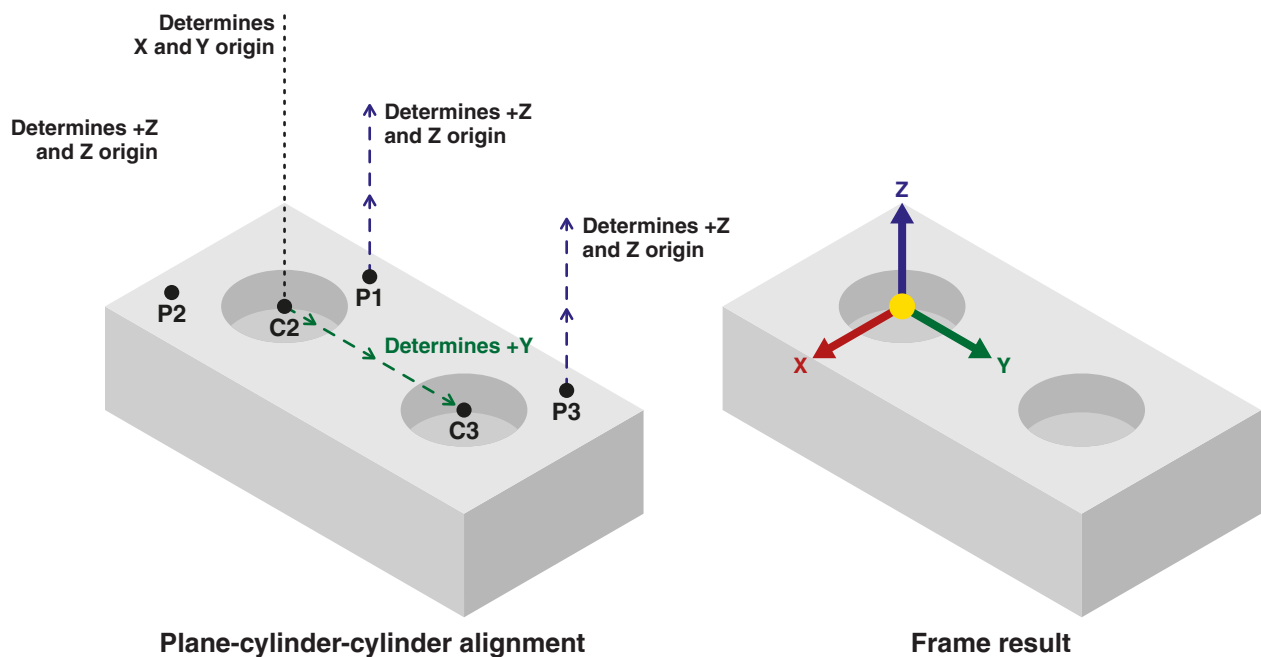
Example:

```
RswPartAlignPCC 10,11,12,UF,MyNewPart;
```

//Server calculates an user frame with a plane-cylinder-cylinder alignment.

//Surface ID 10 contains data of at least three points on a plane. Surface ID 11 contains data for the first cylinder (C2). Surface ID 12 contains data for the second cylinder (C3).

//Robot writes frame result to the user frame of "MyNewPart".



2.7.4.2 Tool

Syntax

```
RswToolAlignPCC SurfaceID1, SurfaceID2, SurfaceID3, NewToolFrame
```

Argument	Definition
SurfaceID1	num. The surface ID which contains position data for a plane (P1, P2, P3).
SurfaceID2	num. The surface ID which contains position data for the first cylinder (C2).
SurfaceID3	num. The surface ID which contains position data for the second cylinder (C3).
NewToolFrame	tooldata. The name of the tool frame where the robot should write the calculated frame data.

Example:

```
RswToolAlignPCC 10,11,12,MyNewTool; //Server calculates a tool frame with a plane-  
cylinder-cylinder alignment.  
  
//Surface ID 10 contains data of at least three  
points on a plane. Surface ID 11 contains data for  
the first cylinder (C2). Surface ID 12 contains data  
for the second cylinder (C3).  
  
//Robot writes frame result to "MyNewTool".
```

2.7.5 Plane-plane-point alignment

This macro tells the Server to calculate a frame using a plane-plane-point alignment.

This alignment will create the frame at the intersection of two planes (Plane 1, Plane 2). It can be useful for welding applications.

The X axis will sit along the intersecting line between the two planes. The Server will use the rotation between the plane normal directions (see 5.8 on page 68) to calculate the +X direction. The plane normal directions will point out of the plane surfaces, not into the feature. The rotation from the Plane 2 normal (N2) to the Plane 1 normal (N1) will define the +X direction. Using the right-hand rule (see 5.12 on page 69), align your thumb with the X axis so that your curled fingers point in the opposite direction of N2 towards N1. Your thumb will be pointing in the +X direction. To reverse the +X direction, swap the order of surface IDs for Plane 1 and Plane 2 in the macro.

A single point (P3) will define the X origin.

The Z axis will be the bisector of the two planes. The intersection of the planes will also define the Z and Y origin.

The Server will calculate the +Y direction as a result of the +X and +Z directions.

This macro can be used to calculate a work object or a tool frame. The robot will write the result to the frame you provide.

2.7.5.1 Work object

Syntax

RswPartAlignPPPn SurfaceID1, SurfaceID2, SurfaceID3, WobjType, NewWorkObject

Argument	Definition
SurfaceID1	num. The surface ID which contains position data for Plane 1.
SurfaceID2	num. The surface ID which contains position data for Plane 2.
SurfaceID3	num. The surface ID which contains position data for a point (P3).
WobjType	alias. 0 or UF for a user frame, 1 or OF for an object frame.
NewWorkObject	wobjdata. The name of the work object where the robot should write the calculated frame data.

Example:

```
RswPartAlignPPPn
```

```
11, 22, 33, UF, MyNewPart;
```

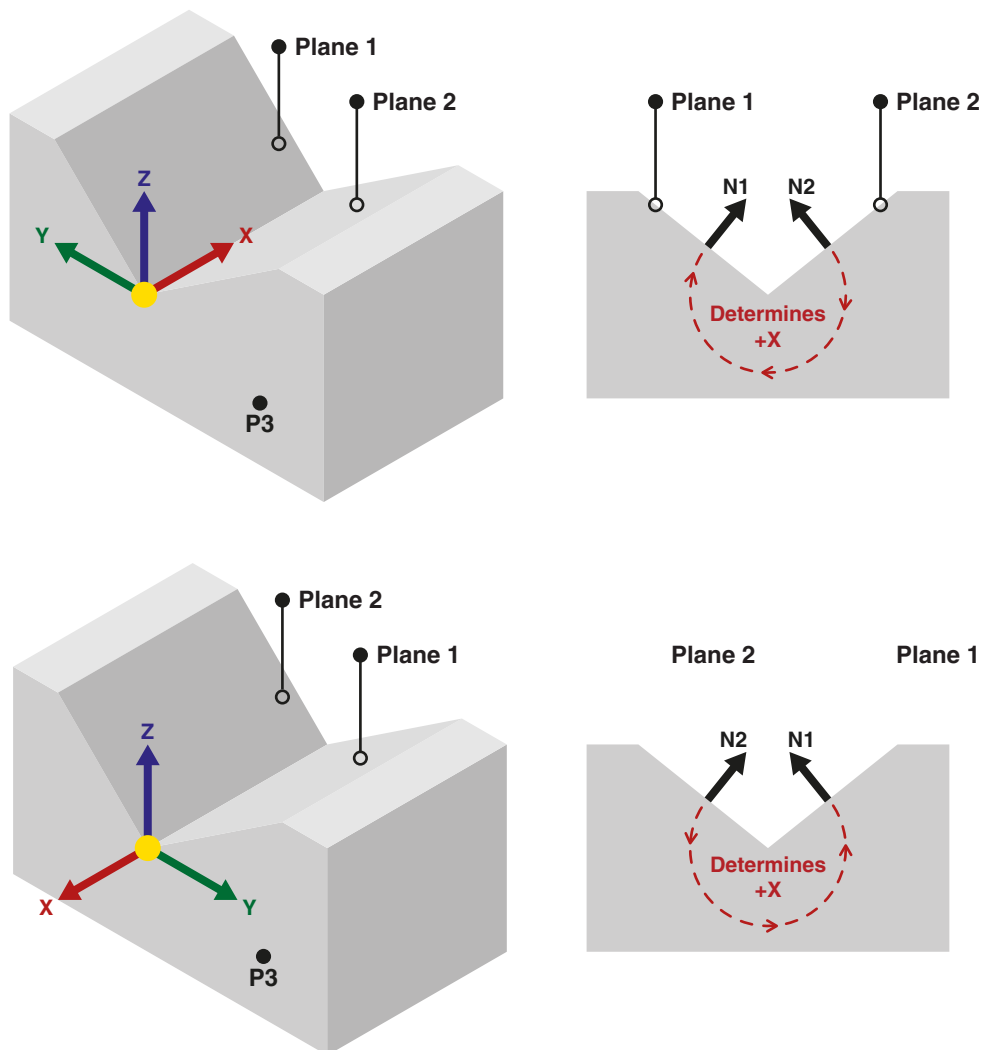
```
//Server calculates a user frame with a plane-plane-  
point alignment.
```

```
//Surface ID 11 contains Plane 1 data.
```

```
//Surface ID 22 contains Plane 2 data.
```

```
//Surface ID 33 contains P3 data.
```

```
//Robot writes result to the user frame of "MyNewPart".
```



Two examples of a plane-plane-point alignment

The +X direction changes based on the order that the planes are referenced in the macro

2.7.5.2 Tool

Syntax

```
RswToolAlgnPPPn SurfaceID1, SurfaceID2, SurfaceID3, NewToolFrame
```

Argument	Definition
SurfaceID1	num. The surface ID which contains position data for Plane 1.
SurfaceID2	num. The surface ID which contains position data for Plane 2.
SurfaceID3	num. The surface ID which contains position data for a point (P3).
NewToolFrame	tooldata. The name of the tool frame where the robot should write the calculated frame data.

Example:

```
RswToolAlgnPPPn 11,22,33,MyNewTool; //Server calculates a tool frame with a plane-plane-  
point alignment.  
  
//Surface ID 11 contains Plane 1 data.  
//Surface ID 22 contains Plane 2 data.  
//Surface ID 33 contains P3 data.  
  
//Robot writes frame result to "MyNewTool".
```

2.7.6 Sphere-sphere-sphere alignment

This macro tells the Server to calculate a frame using a sphere-sphere-sphere alignment.

This alignment will create the frame using the centre points of three spheres (S1, S2 and S3).

S1 will define the X, Y and Z origin. A fitted line (L1) from S1 to S2 will define the +Y direction.

A fitted line (L2) from S3 and perpendicular to L1 will define the +X direction. The Server will calculate the +Z direction as a result of the X and Y axes.

This macro can be used to calculate a work object or a tool frame. The robot will write the frame result to the frame you provide.

2.7.6.1 Work object

Syntax

```
RswPartAlgnSSS SurfaceID1, SurfaceID2, SurfaceID3, WobjType, NewWorkObject
```

Argument	Definition
SurfaceID1	num. The surface ID which contains position data for the first sphere (S1).
SurfaceID2	num. The surface ID which contains position data for the second sphere (S2).
SurfaceID3	num. The surface ID which contains position data for the third sphere (S3).
WobjType	alias. 0 or UF for a user frame, 1 or OF for an object frame.
NewWorkObject	wobjdata. The name of the work object where the robot should write the calculated frame data.

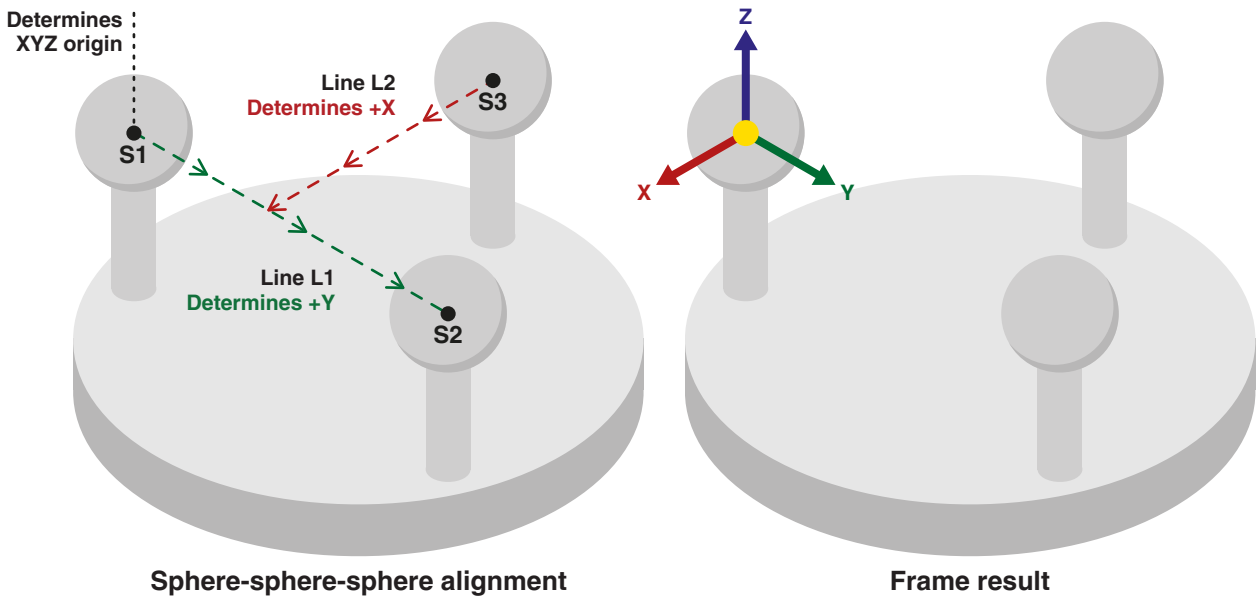
Example:

```
RswPartAlgnSSS 10,11,12,OF,MyNewPart;
```

```
//Server calculates an object frame with a sphere-  
sphere-sphere alignment.
```

```
//Surface ID 10 contains S1 data.  
//Surface ID 11 contains S2 data.  
//Surface ID 12 contains S3 data.
```

```
//Robot writes result to the object frame of  
"MyNewPart".
```



2.7.6.2 Tool

Syntax

```
RswToolAlgnSSS SurfaceID1, SurfaceID2, SurfaceID3, NewToolFrame
```

Argument	Definition
SurfaceID1	num. The surface ID which contains position data for the first sphere (S1).
SurfaceID2	num. The surface ID which contains position data for the second sphere (S2).
SurfaceID3	num. The surface ID which contains position data for the third sphere (S3).
NewToolFrame	tooldata. The name of the tool frame where the robot should write the calculated frame data.

Example:

```
RswToolAlgnSSS 10,11,12,MyNewTool;
```

```
//Server calculates a tool frame with a sphere-  
sphere-sphere alignment.
```

```
//Surface ID 10 contains S1 data.  
//Surface ID 11 contains S2 data.  
//Surface ID 12 contains S3 data.
```

```
//Robot writes frame result to "MyNewTool".
```

2.8 Spherical tool calibration

These macros enable TCP calibration of a tool with a spherical end.

2.8.1 Initialise TCP calibration

This macro tells the Server to discard existing data from the given surface ID, ready to receive new data. It also tells the robot to activate the given work object and tool frame for the TCP calibration. The robot sends all active frame data to the Server.

Syntax

```
RswTCPCalInit SurfaceID, WorkObject, ToolFrame.
```

Argument	Definition
SurfaceID	Integer. The surface ID to discard existing data from.
WorkObject	wobjdata. The name of the work object to activate for the TCP calibration.
ToolFrame	tooldata. The name of the tool frame to activate for the TCP calibration.

Example:

```
RswTCPCalInit //Server discards existing data from surface ID 15.
15,MySphereTool
//Robot activates "MyPart" and "MySphereTool" and sends all
frame data to Server.
```

2.8.2 Run TCP calibration on plane

This macro tells the Server to calculate the TCP of a spherical tool using points measured on the plane of a tool setter disc stylus. You must define the normal direction of the disc stylus plane. This is the direction that the plane is facing within the active work object. The robot will write the calculated TCP to the given tool frame.

Syntax

```
RswTpcCalPlane SurfaceID, PlaneNormalX, PlaneNormalY, PlaneNormalZ, NewToolFrame
```

Argument	Definition
SurfaceID	num. The surface ID which contains position data of the spherical tool measured on the disc stylus.
PlaneNormalX	num. The X component of the disc stylus plane normal direction.
PlaneNormalY	num. The Y component of the disc stylus plane normal direction.
PlaneNormalZ	num. The Z component of the disc stylus plane normal direction.
NewToolFrame	tooldata. The name of the tool frame where the robot should write the calculated frame data.

Example:

```
RswTpcCalPlane //Server calculates a tool TCP using data stored in surface ID 15.
15,0,0,1,MySphereTool; The plane normal direction is +Z in the active work object.
//Robot writes frame result to "MySphereTool".
```

2.8.3 Run TCP calibration on sphere

This macro tells the Server to calculate the TCP of a spherical tool using points measured on a spherical tool setter stylus. You must provide the diameter of the spherical tool end. The robot will write the calculated TCP to the given tool frame.

Syntax

RswTcpCalSphere SurfaceID, ToolDiameter, NewToolFrame

Argument	Definition
SurfaceID	num. The surface ID which contains position data of the spherical tool measured on the spherical stylus.
ToolDiameter	num. The diameter of the spherical tool end in millimetres (mm)
NewToolFrame	tooldata. The name of the tool frame where the robot should write the calculated frame data.

Example:

```
RswTcpCalSphere           //Server calculates the TCP of a tool using data stored in surface
15,25,MySphereTool;      ID 15. The tool has a spherical end with diameter 25 mm.

                           //Robot writes frame result to "MySphereTool".
```

2.9 Residual error statistics

This macro tells the Server to provide statistical data for the residual errors of its last calculation (see “5.11 Residual error” on page 68). You can call this macro after any of the following calculations:

- Simple alignments (see 2.7 on page 34)
- Complex alignments (see 3.1 on page 47)
- Probe calibration (see 2.6 on page 30)
- Spindle artefact and cutting tool calibration (see 4.1 on page 58)
- Best fit feature evaluation (see 3.2 on page 50)

The robot will write the results to the variables you provide.

Syntax

`RswStatRes Average, Min, Max, AbsMax, StdDev, RMS, NbVal, MinPoint, MaxPoint`

Argument	Definition
Average	num. The name of the variable to write the average residual error value.
Min	num. The name of the variable to write the minimum residual error value.
Max	num. The name of the variable to write the maximum residual error value.
AbsMax	num. The name of the variable to write the absolute maximum residual error value (see 5.11.1 on page 68).
StdDev	num. The name of the variable to write the standard deviation of residual errors.
RMS	num. The name of the variable to write the root mean square (RMS) of the residual error values.
NbVal	num. The name of the variable to write the total number of touch points used in the calculation.
MinPoint	num. The name of the variable to write the touch point which had the minimum residual error value.
MaxPoint	num. The name of the variable to write the touch point which had the maximum residual error value.

Example:

```
RswProbeCalRun 15,MyTool; //Server calculates a probe calibration using data stored in surface ID 15.
```

```
RswStatRes Average,Min,Max,AbsMax,StdDev,RMS,NbVal,MinPoint,Maxpoint; //Server provides the statistical data of the residual errors of the last calculation, which was a probe calibration.
```

```
//Robot stores the data in the given numeric type variables.
```

3 RCS advanced probing licence

To use the macros in this section, you will need an active RCS advanced probing licence. You will also need an active RCS core probing licence to collect the data required for the advanced macro calculations.

3.1 Complex alignment

Complex alignments use feature position data to calculate a co-ordinate system. Complex alignments will have one or more of the following characteristics:

- An alignment requiring four or more features.
- An alignment using any features which are not a plane, sphere or cylinder.
- An alignment using constructed features which are based on real features. For example, two measured planes extended to construct a line where they would intersect in space.
- An alignment which has a datum offset from real features (see “5.5 Datum” on page 68).
- An alignment based on a freeform shape.

We recommend that you configure complex alignments in RCS Software Suite. Use the Probing Alignment session to generate datum configuration files and template programs. You can set surfaces for your robot to probe using either CAD models or nominal values. The generated robot programs will contain all instructions required to conduct the alignment.

3.1.1 Initialise the datum builder

This macro sends the given datum configuration ID to the Server. The Server will check the corresponding datum configuration file. It will discard existing data from all surface IDs the file references. This macro also tells the robot to activate the given work object and tool frame for the alignment. The robot sends all active frame data to the Server.

Syntax

```
RswDtmBlDrInit DatumConfigID, WorkObject, ToolFrame
```

Argument	Definition
DatumConfigID	num. The ID of the datum configuration file the Server should check for surface IDs to discard data from. In RCS Software Suite, this is the alignment ID you set when defining the Probing Alignment session.
WorkObject	wobjdata. The name of the work object to activate for the alignment.
ToolFrame	tooldata. The name of the tool frame to activate for the alignment.

Example:

```
RswDtmBlDrInit 15,MyPart,MyTool; //Server discards existing data from surface IDs
//referenced in DatumConfig15.

//Robot activates “MyPart” and “My Tool” and sends
//all frame data to Server.
```

3.1.2 Run the datum builder to a frame

This macro tells the Server to calculate a co-ordinate system. Generated robot programs will call this macro after collecting position data with touch macros. The Server will use data stored in the surface IDs that the datum configuration file references to calculate the result. The robot will write the result to the given frame.

3.1.2.1 Work object

Syntax

```
RswPartDtmBldrRun DatumConfigID, WobjType, NewWorkObject
```

Argument	Definition
DatumConfigID	num. The ID of the datum configuration file which references the surface IDs to use in the calculation. In RCS Software Suite, this is the alignment ID you set when defining the Probing Alignment session.
WobjType	alias. 0 or UF for a user frame, 1 or OF for an object frame.
NewWorkObject	wobjdata. The name of the work object where the robot should write the calculated result.

Example:

```
RswPartDtmBldrRun 15,UF,MyNewPart;           //Server calculates a co-ordinate system using  
                                              surfaces referenced in the file DatumConfig15.  
  
                                              //Robot writes frame result to "MyNewPart".
```

3.1.2.2 Tool

Syntax

```
RswToolDtmBldrRun DatumConfigID, NewToolFrame
```

Argument	Definition
DatumConfigID	num. The ID of the datum configuration file which references the surface IDs to use in the calculation. In RCS Software Suite, this is the alignment ID you set when defining the Probing Alignment session.
NewToolFrame	tooldata. The name of the tool frame where the robot should write the calculated result.

Example:

```
RswToolDtmBldrRun 16,MyNewTool;           //Server calculates a co-ordinate system using  
                                              surfaces referenced in the file DatumConfig16.  
  
                                              //Robot writes frame result to "MyNewTool".
```


3.1.3 Run the datum builder to a Pose

This macro tells the Server to calculate a co-ordinate system. You can define the co-ordinate system within the measurement frame, or the null frame (see “5.4.3 Measurement frame” on page 67 and “5.4.4 Null frame” on page 68). The Server will use data stored in the surface IDs that the datum configuration file references to calculate the result.

Before calling this macro, you must declare a pose type variable for the robot to write the result to.

If you generate your robot program using RCS Software Suite, you will need to edit it to include this macro.

In the generated program, find the “Run the Datum Builder” comment. The macro call below this comment will either be RswPartDtmBldrRun or RswToolDtmBldrRun. Replace the macro with RswDtmBldrRunPose. Edit the argument values within the macro if required.

Syntax

RswDtmBldrRunPose DatumConfigID, PoseOrigin, Pose

Argument	Definition
DatumConfigID	num. The ID of the datum configuration file which references the surface IDs to use in the calculation. In RCS Software Suite, this is the alignment ID you set when defining the Probing Alignment session.
PoseOrigin	num. 0 to set the coordinate system within the null frame (World/flange), 1 to set it within the measurement frame (work object/tool frame)
Pose	pose. The name of the variable where the robot should write the calculated result.

Example:

```
RswDtmBldrRunPose 17,0,NewPose;
```

//Server calculates a co-ordinate system using surfaces referenced in the file DatumConfig17.

//Robot writes frame result to “NewPose”, set within the null frame.

3.2 Best fit feature evaluation

Use these macros to evaluate feature size and position with a best fit algorithm.

Before performing a best fit, use touch move macros (see 2.5 on page 17) to assign feature position data to surface IDs. You will need to reference the surface IDs in the best fit macros.

The Server will store best fit calculation results in the best fit IDs you provide. You can use these best fit IDs to calculate the distance or angle between features (see 3.3 on page 56)

3.2.1 Initialise best fit

This macro tells the Server to discard existing data from the given surface ID, ready to receive new data.

Call this macro in your robot program before collecting feature position data.

Syntax

```
RswDatabagInit SurfaceID
```

Argument	Definition
SurfaceID	num. The surface ID to discard existing data from.

Example:

```
RswDatabagInit 15; //Server discards existing data from surface ID 15.
```

3.2.2 Best fit a plane

This macro tells the Server to best fit a plane using the least-squares method.

The Server will use data stored in the surface ID you provide to calculate the best fit.

Before calling this macro, you must declare variables for the robot to write the results to. Declare two position type variables – one for the fitted plane position and one for the plane normal. The Server will store the best fit result in the best fit ID you provide for future reference.

Syntax

`RswBestFitPlane SurfaceID, BestFitID, PlaneVar, NormalVar`

Argument	Definition
SurfaceID	num. The surface ID which contains position data for a plane.
BestFitID	num. The best fit ID where the Server should store the fitted plane data for future reference.
PlaneVar	pos. The name of the variable where the robot should write the fitted plane position.
NormalVar	pos. The name of the variable where the robot should write the fitted plane normal.

Example:

```
RswBestFitPlane 16,           //Server calculates a best fit for a plane using data stored in surface ID 16.
102,PlanePos,PlaneDir;      //Robot writes the fitted plane position to position type variable "PlanePos"
                              //and the fitted plane normal to position type variable "PlaneDir".

                              //Server stores the best fit result in best fit ID 102.
```

3.2.3 Best fit a sphere

This macro tells the Server to best fit a bore or boss sphere using the least-squares method.

The Server will use data stored in the surface ID you provide to calculate the best fit.

Before calling this macro, you must declare variables for the robot to write the results to. Declare a position type variable for the fitted sphere centre. Declare a numeric type variable for the fitted sphere diameter. The Server will store the best fit result in the best fit ID you provide for future reference.

Syntax

```
RswBestFitSphere SurfaceID, BestFitID, BoreBoss, CenterVar, DiameterVar
```

Argument	Definition
SurfaceID	num. The surface ID which contains position data for a sphere.
BestFitID	num. The best fit ID where the Server should store the fitted sphere data for future reference.
BoreBoss	alias. -1 or "Bore" for a bore sphere, 1 or "Boss" for a boss sphere.
CentreVar	pos. The name of the variable where the robot should write the fitted sphere centre.
DiameterVar	num. The name of the variable where the robot should write the fitted sphere diameter in millimetres (mm).

Example:

```
RswBestFitSphere 15,  
101, Boss, SphPos, SphDiam;
```

```
//Server calculates a best fit for a boss sphere  
using data stored in surface ID 15.
```

```
//Robot writes the fitted sphere centre to position  
type variable "SphPos", and the fitted sphere  
diameter to numeric type variable "SphDiam".
```

```
//Server stores the best fit result in best fit ID 101.
```

3.2.4 Best fit a cylinder

This macro tells the Server to best fit a bore or boss cylinder using the least-squares method.

The Server will use data stored in the surface ID you provide to calculate the best fit.

Before calling this macro, you must declare variables for the robot to write the results to. Declare two position type variables – one for the fitted cylinder centre and one for the cylinder direction. Declare a numeric type variable for the fitted cylinder diameter. The Server will store the best fit result in the best fit ID you provide for future reference.

Syntax

```
RswBestFitCylinder SurfaceID, BestFitID, BoreBoss, CentreVar, DirectionVar, DiameterVar
```

Argument	Definition
SurfaceID	num. The surface ID which contains position data for a cylinder.
BestFitID	num. The best fit ID where the Server should store the fitted cylinder data for future reference.
BoreBoss	alias. -1 or "Bore" for a bore cylinder, 1 or "Boss" for a boss cylinder.
CentreVar	pos. The name of the variable where the robot should write the fitted cylinder centre.
DirectionVar	pos. The name of the variable where the robot should write the fitted cylinder direction.
DiameterVar	num. The name of the variable where the robot should write the fitted cylinder diameter in millimetres (mm).

Example:

```
RswBestFitCylinder //Server calculates a best fit for a boss cylinder
17,104,Boss,CylPos,CylDir,CylDiam; using data stored in surface ID 17.

//Robot writes the fitted cylinder centre to position
//Robot writes the fitted cylinder axis
//Robot writes the fitted cylinder diameter to numeric
//Server stores the best fit result to best fit ID 104.
```

3.2.5 Best fit a 2D circle

This macro tells the Server to best fit a bore or boss 2D circle using the least-squares method.

The Server will use data stored in the surface ID you provide to calculate the best fit. The normal is the direction that the 2D circle is facing (NormalDirectionX/Y/Z). You must define the normal direction within the active frame that the circle lies in, which can be a work object or tool frame.

Before calling this macro, you must declare variables for the robot to write the results to. Declare a position type variable for the fitted circle centre. Declare a numeric type variable for the fitted circle diameter. The Server will store the best fit result in the best fit ID you provide for future reference.

Syntax

```
RswBestFitCircle2D SurfaceID, BestFitID, BoreBoss, NormalDirection, CentreVar,  
DiameterVar
```

Argument	Definition
SurfaceID	num. The surface ID which contains position data for a 2D circle.
BestFitID	num. The best fit ID where the Server should store the fitted circle data for future reference.
BoreBoss	alias. -1 or "Bore" for a bore circle, 1 or "Boss" for a boss circle.
NormalDirection	pos. The co-ordinates of the circle plane normal direction.
CentreVar	pos. The name of the variable where the robot should write the fitted circle centre.
DiameterVar	num. The name of the variable where the robot should write the fitted circle diameter in millimetres (mm).

Example:

```
RswBestFitCircle2D  
15,105,Bore,[0,0,1],CirPos,CirDiam;
```

```
//Server calculates a best fit for a bore 2D circle  
using data stored in surface ID 15.
```

```
//The circle plane normal is in the +Z direction of  
the active frame that it lies in.
```

```
//Robot writes the fitted circle centre to position  
type variable "CirPos".
```

```
//Robot writes the fitted circle diameter to numeric  
type variable "CirDiam".
```

```
//Server stores the best fit result in best fit ID 105.
```

3.2.6 Best fit a cone

The macro tells the Server to best fit a bore or boss cone using the least-squares method.

The Server will use data stored in the surface ID you provide to calculate the best fit. The Server will project the centroid (see 5.3 on page 67) of the measured points onto the cone axis. The Server will calculate the cone centre point and diameter at this position.

Before calling this macro, you must declare variables for the robot to write the results to. Declare two position type variables – one for the fitted cone centre and one for the cone direction. Declare two numeric type variables – one for the fitted cone diameter and one for the half angle. The Server will store the best fit result in the best fit ID you provide for future reference.

Syntax

RswBestFitCone SurfaceID, BestFitID, BoreBoss, CentreVar, DirectionVar, DiameterVar, AngleVar

Argument	Definition
SurfaceID	num. The surface ID which contains position data for a cone.
BestFitID	num. The best fit ID where the Server should store the fitted cone data for future reference.
BoreBoss	alias. -1 or “Bore” for a bore cone, 1 or “Boss” for a boss cone.
CentreVar	pos. The name of the variable where the robot should write the fitted cone centre.
DirectionVar	pos. The name of the variable where the robot should write the fitted cone direction.
DiameterVar	num. The name of the variable where the robot should write the fitted cone diameter in millimetres (mm).
AngleVar	num. The name of the variable where the robot should write the fitted cone half angle in degrees (°).

Example:

```
RswBestFitCone 19,106,Boss,  
ConePos,ConeDir,ConeDiam,ConeAng;
```

//Server calculates a best fit for a boss cone using data stored in surface ID 19.

//Robot writes the fitted cone centre to position type variable “ConePos”, and the fitted cone axis direction to position type variable “ConeDir”.

//Robot writes the fitted cone diameter to numeric type variable “ConeDiam”, and the half angle to numeric type variable “ConeAng”.

//Server stores the best fit result in best fit ID 106.

3.3 Angle and distance between features

3.3.1 Angle between features

This macro tells the Server to calculate the angle between two best fitted features. Before calling this macro, you must declare a numeric type variable for the robot to write the result to.

Syntax

```
RswAnlsAngle BestFitID1, BestFitID2, AngleVar
```

Argument	Definition
BestFitID1	num. The best fit ID which contains calculated data for the first best fit feature.
BestFitID2	num. The best fit ID which contains calculated data for the second best fit feature.
AngleVar	num. The name of the variable where the robot should write the calculated angle between features in degrees (°).

Example:

```
RswBestFitCylinder //Server calculates a best fit for a cylinder. It stores
15,115,Boss,Cyl1Pos,Cyl1Dir,Cyl1Diam; the result in best fit ID 115.

RswBestFitCylinder //Server calculates a best fit for another cylinder. It
16,116,Boss,Cyl2Pos,Cyl2Dir,Cyl2Diam; stores the result in best fit ID 116.

RswAnlsAngle 115,116,Angle; //Server calculates the angle between the two best
//Robot writes the result to numeric type variable
//Server calculates the angle between the two best fitted cylinders using data stored in best fit IDs 115 and 116.
"Angle".
```


3.3.2 Distance between features

This macro tells the Server to calculate the distance between two best fitted features. Before calling this macro, you must declare a numeric type variable for the robot to write the result to.

Syntax

RswAnlsDist BestFitID1, BestFitID2, DistanceVar

Argument	Definition
BestFitID1	num. The best fit ID which contains calculated data for the first best fit feature.
BestFitID2	num. The best ID which contains calculated data for the second best fit feature.
DistanceVar	num. The name of the variable where the robot should write the calculated distance between features in millimetres (mm).

Example:

```
RswBestFitCylinder //Server calculates a best fit for a cylinder. It stores
15,115,Boss,Cyl1Pos,Cyl1Dir,Cyl1Diam; the result in best fit ID 115.
```

```
RswBestFitCylinder //Server calculates a best fit for another cylinder. It
16,116,Boss,Cyl2Pos,Cyl2Dir,Cyl2Diam; stores the result in best fit ID 116.
```

```
RswAnlsDist 115,116,Distance; //Server calculates the distance between the two
//Server calculates the distance between the two best fitted cylinders using data stored in best fit IDs 115 and 116.
```

```
//Robot writes the result to numeric type variable "Distance".
```

4 RCS spindle calibration licence

To use the macros in this section, you will need an active RCS spindle calibration licence. You will also need an active RCS core probing licence to collect the data required for the macro calculations.

4.1 Spindle artefact and cutting tool calibration

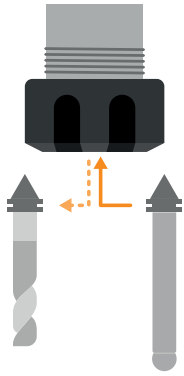
For robotic applications which use a cutting tool, accurate tool vectors and lengths must be set. This ensures tools will follow cutting paths and machine features correctly.

These macros enable spindle cutting tool calibration. This means that tool vectors and lengths can be automatically measured in-process.

To calibrate a spindle cutting tool with RCS P-series, you will need the ball-ended spindle artefact and tool setter disc stylus from the RCS spindle calibration kit.

1. Replace the spindle cutting tool on your robot with the ball-ended spindle artefact.
2. Manually set a nominal tool frame where the +Z direction points out of the artefact (i.e. from the spindle mounting point towards the centre of the artefact sphere).
3. With the spindle artefact, take at least three touch points around the top plane of the disc stylus. This will determine the disc stylus plane normal direction for “4.1.2 Run spindle artefact calibration” on page 60.
4. Take at least five touch points on the top plane of the disc stylus at different rotations between 45°–90° about the X and Y axes around the spindle artefact tip. This will determine the position of the spindle artefact TCP and disc stylus plane for “4.1.2 Run spindle artefact calibration” on page 60.
5. Take touch points at two different heights around the sides of the disc stylus with the artefact shaft. These touch points can be taken using two external circle touch commands (see 2.5.4 on page 20). This will determine the direction of the spindle artefact and cutting tool for “4.1.2 Run spindle artefact calibration” on page 60.
6. Take a single touch point on the top of the disc stylus plane. This is used in “4.1.5 Set disc stylus reference height” on page 63.
7. Replace the spindle artefact with the spindle cutting tool. Take another touch point on top of the disc stylus plane. This is used in “4.1.7 Set spindle cutting tool TCP” on page 65 to find the exact cutting tool length.
8. Take touch points around the sides of the disc stylus with the spindle cutting tool. These touch points can be taken using a single external circle touch command (see 2.5.4 on page 20). This is used in “4.1.8 Set spindle cutting tool diameter” on page 66.

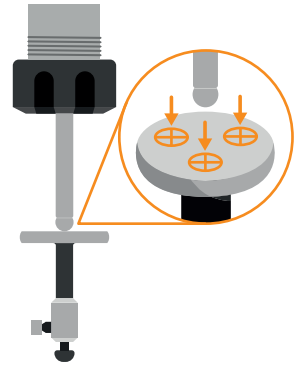
The Server stores the touch point data in the surface IDs you provide. You can reference these in the spindle calibration macros.



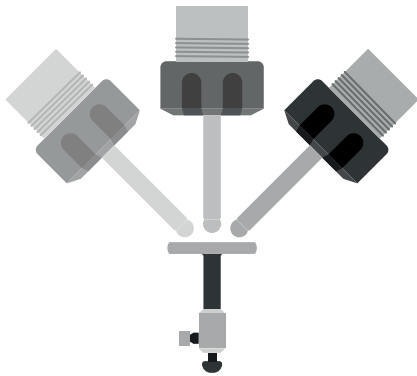
1 Swap tool with spindle artefact



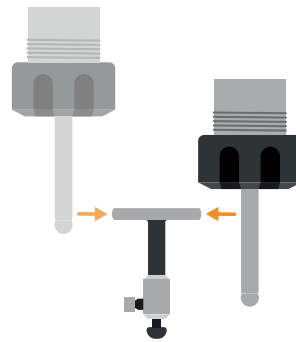
2 Manually set a nominal tool frame where +Z points out of the artefact



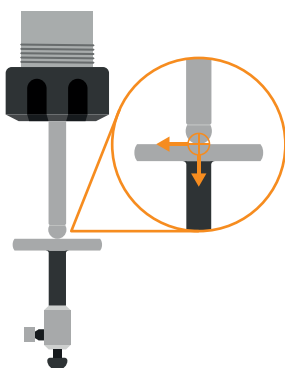
3 Take three touch points around top of disc stylus



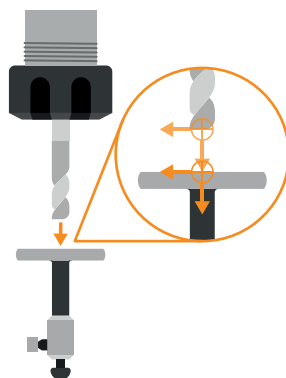
4 Use artefact on top of disc stylus in several orientations



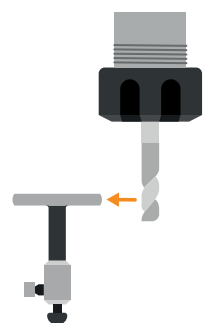
5 Use artefact on sides of disc stylus at different heights



6 Touch top of disc stylus to set reference height



7 Re-insert tool and find working tool length



8 Use tool on sides of disc to find diameter

4.1.1 Initialise spindle artefact calibration

This macro tells the Server to discard existing data from the given surface IDs, ready to receive new data. It also tells the robot to activate the given work object and tool frame for the spindle calibration. The robot sends all active frame data to the Server.

Call this macro in your robot program before probing the spindle artefact with the tool setter disc stylus.

Syntax

```
RswSpndlCalIni DiscID, ArtefactID, DirectionID, WorkObject, ToolFrame
```

Argument	Definition
DiscID	num. The first surface ID to discard existing data from.
ArtefactID	num. The second surface ID to discard existing data from.
DirectionID	num. The third surface ID to discard existing data from.
WorkObject	wobjdata. The name of the work object to activate for the spindle calibration.
ToolFrame	tooldata. The name of the tool frame to activate for the spindle calibration.

Example:

```
RswSpndlCalIni                               //Server discards existing data from surface IDs 21,  
21, 22, 23, WorkObject, ToolName;           22 and 23.  
  
                                             //Robot activates WorkObject and ToolName and  
                                             sends all frame data to Server.
```

4.1.2 Run spindle artefact calibration

This macro uses the data from the given surface IDs to populate a work object and two tool frames.

Before calling this macro, complete steps 1–5 of the spindle calibration process (see 4.1 on page 58).

Using the collected position data, the Server will calculate:

- A work object for the tool setter disc stylus.
- A tool frame at the ball-end of the spindle artefact.
- A reference tool frame at the spindle mounting point. To do this, it will move the calculated spindle artefact tool frame in the –Z direction by the artefact length you provide.

The Server will send the frame results to the robot. The robot will:

- Write the tool setter result to the user frame of the work object you provide (NewWorkObject).
- Write the spindle artefact tool frame to the tool frame you provide (NewToolFrame).
- Write the reference tool frame at the spindle mounting point to the tool frame you provide (RefToolFrame).

Syntax

RswSpndlCalRun DiscID, ArtefactID, DirectionID, ArtefactLength, NewWorkObject, NewToolFrame, RefToolFrame

Argument	Definition
DiscID	num. The surface ID which contains the disc stylus plane position data.
ArtefactID	num. The surface ID which contains the artefact TCP position data.
DirectionID	num. The surface ID which contains the artefact direction position data.
ArtefactLength	Number. The length of your spindle artefact in millimetres (mm). Measure your artefact in place from the spindle mounting point to the centre of the artefact sphere.
NewWorkObject	wobjdata. The name of the work object where the robot should write the calculated tool setter disc stylus work object.
NewToolFrame	tooldata. The name of the tool frame where the robot should write the spindle artefact tool frame.
RefToolFrame	tooldata. The name of the tool frame where the robot should write the reference tool frame.

Example:

```
RswSpndlCalRun
21, 22, 23, 98.5, DiscWorkObject,
ArtefactToolFrame, RefToolFrame;
```

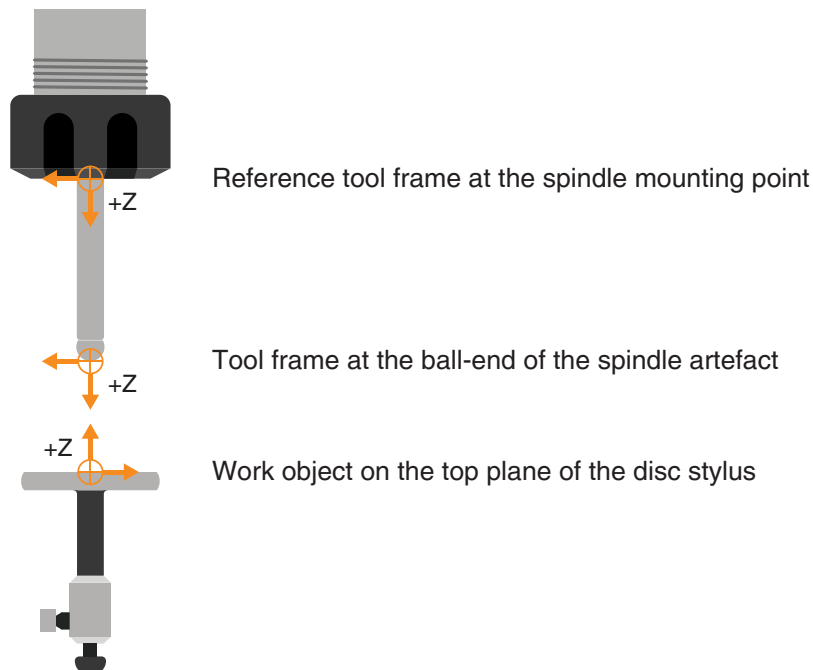
//Server uses data from surface IDs 21, 22 and 23 to calculate: a work object for the disc stylus and a tool frame for the artefact TCP.

//Server uses the artefact TCP and length of 98.5 mm to calculate a reference tool frame at the spindle mounting point.

//Robot writes the user frame result to "DiscWorkObject".

//Robot writes the tool frame result to "ArtefactToolFrame".

//Robot writes the reference tool frame result to "RefToolFrame".



4.1.3 Align spindle artefact or cutting tool

This macro tells the robot to rotate the active tool TCP about its current position. It will align the –Z direction of the TCP with the +Z direction of the active work object. The active work object should be the tool setter disc stylus which was calculated by running the spindle artefact calibration (see 4.1.2 on page 60).

Syntax

```
RswSpndlAlign
```

Example:

```
RswSpndlAlign; //Robot rotates the active tool TCP so that its –Z
                //direction aligns with the +Z direction of the active
                //work object.
```

4.1.4 Initialise spindle cutting tool measurement

This macro tells the Server to discard existing data from the given surface ID, ready to receive new data. It also tells the robot to activate the given work object and tool frame for the measurement. The robot sends all active frame data to the Server.

Call this macro in your robot program when preparing to take touch points for any of the following calculations:

- Disc stylus reference height (see 4.1.5 on page 63)
- Spindle cutting tool TCP (see 4.1.7 on page 65)
- Spindle cutting tool diameter (see 4.1.8 on page 66)

Syntax

```
RswSpndlInit SurfaceID, WorkObject, ToolFrame
```

Argument	Definition
SurfaceID	num. The surface ID to discard existing data from.
WorkObject	wobjdata. The name of the tool setter disc stylus work object.
ToolFrame	tooldata. The name of the tool frame to activate for the measurement. This may be the spindle artefact TCP or spindle cutting tool TCP.

Example:

```
RswSpndlInit //Server discards existing data from surface ID 31.
31, DiscWorkObject, ArtefactToolFrame;
//Robot activates “DiscWorkObject” and
//“ArtefactToolFrame” and sends all frame data to
//Server.
```

4.1.5 Set disc stylus reference height

This macro tells the robot to update the disc stylus work object to set its reference height.

Before calling this macro:

1. Calibrate your spindle artefact (see 4.1.2 on page 60).
2. Align the calibrated spindle artefact normal to the disc stylus plane (see 4.1.3 on page 62).
3. Use a touch command to take a single touch point in the +Z direction of the artefact on top of the disc stylus (SurfaceID). This is step 6 on page 58.

The Server will calculate the disc stylus height using the data stored in the surface ID you provide. The robot will use the result to update the user frame of the given work object, which can be the same work object you set in “4.1.2 Run spindle artefact calibration” on page 60.

This process will mean that the same robot orientation will be used for this macro and the setting spindle cutting tool TCP macros (see 4.1.6 on page 64 and 4.1.7 on page 65). This will minimise errors caused by inaccuracies in your robot.

Syntax

RswSpndlRef SurfaceID, WorkObject

Argument	Definition
SurfaceID	num. The surface ID which contains data of a single touch on the disc stylus plane using a calibrated spindle artefact.
WorkObject	wobjdata. The name of the work object for the tool setter disc stylus. The robot will update the user frame values to set the disc reference height.

Example:

```
RswSpndlRef 24,DiscWorkObject; //Server uses the data stored in surface ID 21 to calculate
                                a reference work object on top of the disc stylus.

                                //Robot writes the result to the user frame of
                                "DiscWorkObject".
```

4.1.6 Set nominal spindle cutting tool TCP

This macro tells the Server to calculate a nominal TCP. It will move the reference tool frame along the tool Z axis by the tool length you provide. The robot will write the result to the tool frame you provide.

Before calling this macro, you need the reference tool frame from “4.1.2 Run spindle artefact calibration” on page 60.

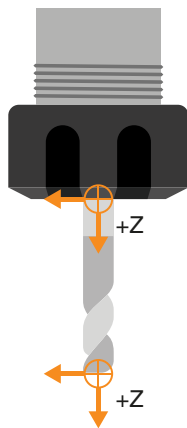
Syntax

```
RswSpndlNom ToolLength, RefToolFrame, NewToolFrame
```

Argument	Definition
ToolLength	num. The nominal length of your spindle cutting tool in millimetres (mm).
RefToolFrame	tooldata. The name of the tool frame which contains the reference tool frame data.
NewToolFrame	tooldata. The name of the tool frame where the robot should write the nominal spindle cutting tool TCP.

Example:

```
RswSpndlNom  
175, RefToolFrame, SpindleToolTCP;  
  
//Server calculates a nominal TCP. It moves the reference  
//tool frame “RefToolFrame” along the tool Z axis by 175 mm.  
  
//Robot writes the tool frame result to “SpindleToolTCP”.
```



Reference tool frame at the spindle mounting point

Nominal tool frame created from moving the reference tool frame by the spindle cutting tool length

4.1.7 Set spindle cutting tool TCP

This macro tells the Server to use the true length of the spindle cutting tool to calculate an accurate TCP.

Before calling this macro:

- Align your spindle cutting tool normal to the disc stylus plane (see 4.1.3 on page 62).
- Use a touch command to take a single touch point in the +Z direction of the tool on top of the disc stylus (SurfaceID). This will determine the true length of the spindle cutting tool (step 7 on page 58).

The Server will calculate the TCP of the spindle cutting tool using data stored in the surface ID you provide. The robot will write the result to the given tool frame, which can be the same tool frame you used for “4.1.6 Set nominal spindle cutting tool TCP” on page 64.

Syntax

RswSpndlTcp SurfaceID, NewToolFrame

Argument	Definition
SurfaceID	num. The surface ID which contains data from a single touch point of the spindle cutting tool on top of the disc stylus plane.
NewToolFrame	tooldata. The name of the tool frame where the robot should write the calibrated spindle cutting tool TCP.

Example:

```
RswSpndlTcp 66,SpindleToolTCP;           //Server calculates an accurate TCP using data
                                           stored in surface ID 66.

                                           //Robot writes the tool frame result to
                                           “SpindleToolTCP”.
```

4.1.8 Set spindle cutting tool diameter

This macro tells the Server to calculate the spindle cutting tool diameter.

Before calling this macro:

- Align your spindle cutting tool normal to the disc stylus plane (see 4.1.3 on page 62).
- Take touch points around the sides of the disc stylus (SurfaceID). You can use the external circle touch command (2.5.4 on page 20) to do this. This is step 8 on page 58.
- Declare a numeric type variable for the robot to write the result to.

The Server will calculate the diameter of the spindle cutting tool using data stored in the surface ID you provide.

Syntax

```
RswSpndlSize SurfaceID, DiameterVar
```

Argument	Definition
SurfaceID	num. The surface ID which contains the spindle cutting tool diameter data.
DiameterVar	num. The name of the variable where the robot should write the calculated diameter in millimetres (mm).

Example:

```
RswSpndlSize 31,ToolSize;
```

//Server calculates a diameter using data stored in surface ID 31.

//Robot writes the result to "ToolSize".

5 Glossary of terms

5.1 Approach position

A point from which a robot will start moving to touch a surface or feature with a probe.

5.2 Best fit

The “best” model to represent a set of data.

With RCS advanced probing macros, you can best fit a feature. This uses data collected from a real feature to create a model for the best approximation of its position and orientation in space (see “3.2 Best fit feature evaluation” on page 50).

5.2.1 Least-squares method

A technique to calculate the best fit model for a set of data. It minimises the sum of squared residual errors between the model and the data points.

RCS probing macros use the least-squares method for all best fit calculations.

5.3 Centroid

The centre of mass of an object.

5.4 Co-ordinate system, Frame

Three orthonormal vectors (X, Y, Z) which originate at a single point. From this origin, any point in space can be positioned using three numerical co-ordinates.

5.4.1 Tool frame, tool centre point (TCP)

A co-ordinate system defined on the tool attached to a robot arm. It is located relative to the robot flange.

Once set, a robot can drive the active tool in position and orientation within the active work object to perform its work.

5.4.2 Work object

A co-ordinate system that is defined on a part within a robot working volume. It is located relative to the robot base co-ordinate system. A work object consists of two co-ordinate systems: a user frame and an object frame. The user frame is the parent of the object frame.

Once set, all cutting and probing co-ordinates are referenced relative to the work object.

5.4.3 Measurement frame

The frame(s) used during a measurement. This will be the frame(s) used in the last initialisation macro before collecting data.

5.4.4 Null frame

The co-ordinate system which frames are located relative to. For a work object, this would be the World frame. For a tool frame, this would be the robot flange frame.

5.5 Datum

The origin of a co-ordinate system, or a known position and orientation in space. With RCS P-series, you can set a datum using any combination of geometric features (see “3.1 Complex alignment” on page 47). Datums can be used to locate work objects and tool frames.

5.6 Feature

A geometric object consisting of one or more surfaces.

5.7 Nominal

A stated quantity or dimension but not necessarily the exact value.

5.8 Normal

Intersecting a line or surface at a right angle (90°). A vector orthogonal to a surface.

5.9 Orthogonal

Two elements are orthogonal when they are at right angles (90°) to each other.

5.10 Parameter

An editable piece of information a robot uses to control its movement or behaviour.

5.11 Residual error

The error which remains after an optimisation takes place. An optimisation could be a correction, calibration, alignment, or best fit. In RCS probing, the error is the distance between a measured touch point and the best fitted model of an optimised feature.

When the Server calculates a feature, it will best fit the feature touch points to create a model. A residual error is how far a touch point used in the calculation is from the best fit model that the Server created. In a set of data, there will be values for the maximum, minimum, average and standard deviation of the residual errors.

Residual errors are called “calibration errors” in RCS Software Suite.

5.11.1 Absolute maximum residual error

The largest residual error value, regardless of its sign.

For example, in a set of data where the maximum residual error is 50 and the minimum residual error is -100, the absolute maximum residual error is 100.

5.12 Right-hand rule

This determines the direction of vectors in three-dimensional space. In this document, use the rule to find the direction of rotation about a given vector, and vice versa. To apply the rule, point your right-hand thumb in the direction of the given vector and curl your fingers. Your fingers will point in the direction of rotation. Or, if you curl your right-hand fingers in the direction of a known rotation, your thumb will point in the direction of the resulting vector.

For example, you may apply this rule to “2.5.4 Touch circle internal and external” on page 20. Point your thumb in the normal direction you define. Your curled fingers will point in the direction the probe will take to move to the remaining approach positions.

5.13 Surface

The outside part or uppermost layer of an object.

5.14 Touch point

Also called a “touch”. This is a position on a surface or feature that a probe touches, causing the probe to trigger. A touch point provides position data of a surface or feature. The Server uses this data to calculate positions and orientations within a given frame. Taking touch points on a surface or feature is called “probing”.

5.15 Vector


A quantity with both direction and size. In RCS probing macros, 3D vectors are defined with three numerical components (X, Y, Z).

www.renishaw.com/rcs-support



#renishaw

 +44 (0) 1453 524524

 industrialautomation@renishaw.com

© 2021–2025 Renishaw plc. All rights reserved. This document may not be copied or reproduced in whole or in part, or transferred to any other media or language by any means, without the prior written permission of Renishaw.

RENISHAW® and the probe symbol are registered trade marks of Renishaw plc. Renishaw product names, designations and the mark 'apply innovation' are trade marks of Renishaw plc or its subsidiaries. Other brand, product or company names are trade marks of their respective owners.

WHILE CONSIDERABLE EFFORT WAS MADE TO VERIFY THE ACCURACY OF THIS DOCUMENT AT PUBLICATION, ALL WARRANTIES, CONDITIONS, REPRESENTATIONS AND LIABILITY, HOWSOEVER ARISING, ARE EXCLUDED TO THE EXTENT PERMITTED BY LAW. RENISHAW RESERVES THE RIGHT TO MAKE CHANGES TO THIS DOCUMENT AND TO THE EQUIPMENT, AND/OR SOFTWARE AND THE SPECIFICATION DESCRIBED HEREIN WITHOUT OBLIGATION TO PROVIDE NOTICE OF SUCH CHANGES.

Renishaw plc. Registered in England and Wales. Company no: 1106260. Registered office: New Mills, Wotton-under-Edge, Glos, GL12 8JR, UK.

Part no.: H-6852-8029-01-A

Issued: 01.2025