

RCS P-series macro instructions for Fanuc 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
1.2.2	List of macros	7
1.2.2.1	RCS core probing licence.....	7
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
2.1.3	Get version.....	10
2.2	Manage frames	11
2.2.1	Set user frame	11
2.2.2	Set tool frame.....	11
2.3	Touch parameters management	12
2.3.1	Load parameters values	12
2.3.2	Set the back off distance.....	12
2.3.3	Set the move speed	13
2.3.4	Set the double touch behaviour	13
2.3.5	Set the search distance	14
2.3.6	Reset all parameters.....	14
2.3.7	Reset the back off distance.....	14
2.3.8	Reset the double touch behaviour	15
2.3.9	Reset the move speed	15
2.3.10	Reset the search distance	15
2.4	Probe management.....	16
2.4.1	Select mobile probe	16
2.4.2	Select tool setter (fixed probe)	16
2.4.3	Probe power on.....	17
2.4.4	Probe power off.....	17
2.5	Touch moves	18
2.5.1	Touch in user frame direction.....	19
2.5.2	Touch in tool frame direction	20
2.5.3	Touch a target point within the active user frame.....	21
2.5.4	Touch circle internal and external	22

2.5.5	Touch sphere	24
2.5.6	Record the stem direction	26
2.5.7	Set the stem direction	28
2.5.8	Touch sphere with stem avoidance	30
2.6	Probe calibration	32
2.6.1	Initialise mobile probe calibration	33
2.6.2	Run mobile probe calibration	33
2.6.3	Initialise tool setter calibration	34
2.6.4	Run tool setter calibration using a calibration sphere	35
2.6.5	Run tool setter calibration using a calibrated probe	35
2.7	Simple alignment	36
2.7.1	Initialise simple alignment	36
2.7.2	321 alignment	37
2.7.3	Plane-plane-plane alignment	38
2.7.4	Plane-cylinder-cylinder alignment	40
2.7.5	Plane-plane-point alignment	42
2.7.6	Sphere-sphere-sphere alignment	44
2.8	Spherical tool calibration	45
2.8.1	Initialise TCP calibration	45
2.8.2	Run TCP calibration on plane	46
2.8.3	Run TCP calibration on sphere	46
2.9	Residual error statistics	47
3	RCS advanced probing licence	48
3.1	Complex alignment	48
3.1.1	Initialise the datum builder	48
3.1.2	Run the datum builder to a frame	49
3.1.3	Run the datum builder to a position register	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	User frame	67
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	69
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 Fanuc 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 48).

1.2.1 Installing the macros

To install the RCS macros on your robot controller, you will need the Fanuc macros folder. If your RCS P-series includes an RPU (controller for robot probing), the folder will be pre-installed on it.

Alternatively, you can download the Fanuc macros folder from www.renishaw.com/softwarelicensing.

1. Open the folder for your Fanuc controller.
2. Copy the contents of the MD folder to the MD drive on your robot controller.
3. The FR folder contains the RCS touch and communication parameter files (RswPrmTouch.xml and RswPrmCom.xml). Open these files and edit the parameters as required for your equipment.
4. Copy the contents of the FR folder to the FR drive on your robot controller.

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. RswAlgn321, page 37 | 16. RswRstBckDst, page 14 | 31. RswTchDirPrt, page 19 |
| 2. RswAlgnInit, page 36 | 17. RswRstDulTch, page 15 | 32. RswTchDirTol, page 20 |
| 3. RswAlgnPCC, page 40 | 18. RswRstMovSpd, page 15 | 33. RswTchPos, page 21 |
| 4. RswAlgnPPP, page 38 | 19. RswRstSchDst, page 15 | 34. RswTchSph, page 24 |
| 5. RswAlgnPPPnt, page 42 | 20. RswSetBckDst, page 12 | 35. RswTchSphStm, page 30 |
| 6. RswAlgnSSS, page 44 | 21. RswSetDulTch, page 13 | 36. RswTCPInit, page 45 |
| 7. RswCheckComm, page 10 | 22. RswSetMovSpd, page 13 | 37. RswTCPPlane, page 46 |
| 8. RswHello, page 9 | 23. RswSetSchDst, page 14 | 38. RswTCPSph, page 46 |
| 9. RswPCInit, page 33 | 24. RswSfPrt, page 11 | 39. RswTsCallnit, page 34 |
| 10. RswPCRRun, page 33 | 25. RswSfTol, page 11 | 40. RswTsCalPRun, page 35 |
| 11. RswPrbPwrOff, page 17 | 26. RswStatRes, page 47 | 41. RswTsCalRun, page 35 |
| 12. RswPrbPwrOn, page 17 | 27. RswStemRec, page 30 | 42. RswVarLoad, page 12 |
| 13. RswPrbSlctFx, page 16 | 28. RswStemSet, page 28 | 43. RswVersion, page 10 |
| 14. RswPrbSlctMb, page 16 | 29. RswTchCirExt, page 22 | |
| 15. RswRstAll, page 14 | 30. RswTchCirIn, page 22 | |

1.2.2.2 RCS advanced probing licence

- | | | |
|--------------------------|-----------------------|-------------------------|
| 1. RswAnIsAngle, page 56 | 5. RswBFCyl, page 53 | 9. RswDbInit, page 48 |
| 2. RswAnIsDist, page 57 | 6. RswBFInit, page 50 | 10. RswDbRun, page 49 |
| 3. RswBFCir, page 54 | 7. RswBFPIIn, page 51 | 11. RswDbRunPR, page 49 |
| 4. RswBFCone, page 55 | 8. RswBFSphr, page 52 | |

1.2.2.3 RCS spindle probing licence

- | | | |
|--------------------------|--------------------------|--------------------------|
| 1. RswSpnAlign, page 62 | 4. RswSpnCalRun, page 60 | 7. RswSpndlSize, page 66 |
| 2. RswSpnCallni, page 60 | 5. RswSpndlInit, page 62 | 8. RswSpndlTCP, page 65 |
| 3. RswSpnCalRef, page 63 | 6. RswSpndlNom, page 64 | |

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 teach pendant “User” window. If the connection is unsuccessful, check the communication parameters in the RswPrmCom.xml file on your robot controller match the parameters in the “Server” tab of the RCS probing server application.

Syntax

RSWHELLO

Example

```
: CALL RSWHELLO; //Robot exchanges an integer with Server.
```

Results

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

```
Sending request to RPU... //This result means there is a connection  
Error Connecting error. Troubleshooting required.  
Connecting timed out
```

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 teach pendant “User” window. The Server application will display the inverse results in the “Output” tab. If communication is unsuccessful, check the macros and Server version numbers match (see 2.1.3). Also, check the robot controller you are using is selected in the “Robot” tab of the Server application.

Syntax

```
RSWCHECKCOMM
```

Example

```
: CALL RSWCHECKCOMM;
```

Results

```
Communication test -> Success
```

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

2.1.3 Get version

This macro tells the robot to send the macros version number to the Server. It tells the Server to send the RCS probing server version number to the robot.

The robot will display both version numbers in the teach pendant “User” window. If the macros and server version numbers do not match, you must update them to the same version number.

Syntax

```
RSWVERSION
```

Example

```
: CALL RSWVERSION;
```

```
//Robot sends macros version to Server.
```

```
//Server sends RCS probing server version  
to robot.
```

Results

```
RCS robot version: 1.0.18  
RCS server version: 1.0.18
```

```
//This result confirms the Server and macro  
versions match
```

```
RCS robot version: 1.0.17  
RCS server version: 1.0.18
```

```
//This result means the Server and macro  
versions do not match.
```

```
RCS robot and server versions do not match
```

NOTE: The version numbers in these examples may differ from the version numbers that you have.

2.2 Manage frames

2.2.1 Set user frame

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

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

Syntax

RSWSFPRT (UserFrame)

Argument	Definition
UserFrame	Integer. The index to set as the active user frame.

Example

```
: CALL RSWSFPRT (10) ;
```

//Robot sets UserFrame[10] as the active user frame 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

RSWSFTOL (ToolFrame)

Argument	Definition
ToolFrame	Integer. The index to set as the active tool frame.

Example:

```
: CALL RSWSFTOL (11) ;
```

//Robot sets ToolFrame[11] as the active tool frame and sends frame data to Server.

2.3 Touch parameters management

The RCS touch parameters are contained within the RswPrmTouch.xml 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 18). The parameter values are loaded onto your robot controller using the RswVarLoad macro. The latest values loaded onto your controller become 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 or the value in the RswPrmTouch.xml file.

2.3.1 Load parameters values

This macro loads the latest parameter values from RswPrmTouch.xml onto the robot.

If a parameter has been changed using a macro, the value will not be reset by calling RswVarLoad. You can reset the parameter to the latest value loaded by RswVarLoad using a reset macro.

Syntax

RSWVARLOAD

Example:

```
: CALL RSWVARLOAD; //Robot loads the touch parameters from RswPrmTouch.xml.
```

2.3.2 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

RSWSETBCKDST (BackoffDistance)

Argument	Definition
BackoffDistance	Number. The new back off distance in millimetres.

Example:

```
: CALL RSWSETBCKDST (10); //Sets the back off distance to 10 mm.
```

2.3.3 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 18). The robot will also use this speed to back off from all touch points.

Syntax

RSWSETMOVSPD (MoveSpeed)

Argument	Definition
MoveSpeed	Number. The new move speed in millimetres per second.

Example:

```
: CALL RSWSETMOVSPD (10) ; //Sets the move speed to 10 mm/s
```

2.3.4 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 RswPrmTouch.xml file. Changing these values could negatively impact the performance of your robot.

Syntax

RSWSETDULTCH (OnOff)

Argument	Definition
OnOff	Integer. 0 turns off double touch behaviour, 1 turns on double touch behaviour.

Example:

```
: CALL RSWSETDULTCH (1) ; //Turns on double touch behaviour.
```

2.3.5 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

RSWSETSCHDST(SearchDistance)

Argument	Definition
SearchDistance	Number. The new search distance in millimetres.

Example:

```
: CALL RSWSETSCHDST(15); //Sets the search distance to 15 mm
```

2.3.6 Reset all parameters

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

These are the latest values that were loaded onto the robot using the RswVarLoad macro (see “2.3.1 Load parameters values” on page 12).

Syntax

RSWRSTALL

Example:

```
: CALL RSWRSTALL; //Resets all touch parameters to the latest default values.
```

2.3.7 Reset the back off distance

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

This is the latest value that was loaded onto the robot using the RswVarLoad macro (see “2.3.1 Load parameters values” on page 12).

Syntax

RSWRSTBCKDST

Example:

```
: CALL RSWRSTBCKDST; //Resets the back off distance to the latest default value.
```

2.3.8 Reset the double touch behaviour

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

This is the latest value that was loaded onto the robot using the RswVarLoad macro (see “2.3.1 Load parameters values” on page 12).

Syntax

```
RSWRSTDULTCH
```

Example:

```
: CALL RSWRSTDULTCH; //Resets double touch behaviour to the latest default value.
```

2.3.9 Reset the move speed

This macro resets the move speed to its default value.

This is the latest value that was loaded onto the robot using the RswVarLoad macro (see “2.3.1 Load parameters values” on page 12).

Syntax

```
RSWRSTMOVSPD
```

Example:

```
: CALL RSWRSTMOVSPD; //Resets the move speed to the latest default value.
```

2.3.10 Reset the search distance

This macro resets the search distance to its default value.

This is the latest value that was loaded onto the robot using the RswVarLoad macro (see “2.3.1 Load parameters values” on page 12).

Syntax

```
RSWRSTSCHDST
```

Example:

```
: CALL RSWRSTSCHDST; //Resets the search distance to the latest 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 RswPrmTouch.xml file (see “1.2.1 Installing the macros” on page 7). For example, the robot will check the mobile probe input index to detect probe triggers. Both the robot and Server will also use the mobile probe tip diameter in calculations.

Syntax

```
RSWPRBSLCTMB
```

Example:

```
: CALL RSWPRBSLCTMB; //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 RswPrmTouch.xml file (see “1.2.1 Installing the macros” on page 7). For example, the robot will check the fixed probe input index to detect probe triggers. Both the robot and Server will also use the fixed probe tip diameter in calculations.

Syntax

```
RSWPRBSLCTFX
```

Example:

```
: CALL RSWPRBSLCTFX; //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

RSWPRBPWRON

Example:

```
: CALL RSWPRBPWRON; //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

RSWPRBPWROFF

Example:

```
: CALL RSWPRBPWROFF; //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 32)
- Simple alignments (see 2.7 on page 36)
- 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 user 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 user frame. If the robot does not detect a probe trigger, it will stop moving once it has travelled the search distance (see 2.3.5 on page 14).

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

Syntax

RSWTCHDIRPRT(SurfaceID, TouchDirectionX, TouchDirectionY, TouchDirectionZ)

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

Example:

```

: L P[1] 100mm/sec FINE; //Robot moves to approach position P[1].

: CALL RSWTCHDIRPRT(1,1,0,0); //Robot moves from P[1] in the +X direction
of the active user frame until the probe
triggers. Server stores position data in
surface ID 1.

: L P[2] 100mm/sec FINE; //Robot moves to approach position P[2].

: CALL RSWTCHDIRPRT(2,0,(-1),0); //Robot moves from P[2] in the -Y direction
of the active user frame until the probe
triggers. Server stores position data in
surface ID 2.

: L P[3] 100mm/sec FINE; //Robot moves to approach position P[3].

: CALL RSWTCHDIRPRT(2,0,(-1),(-1)); //Robot moves from P[3] in the -Y-Z
direction (45°) of the active user frame 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 2.3.5 on page 14).

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

Syntax

RSWTCHDIRTOL(SurfaceID, TouchDirectionX, TouchDirectionY, TouchDirectionZ)

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

Example:

```
: L P[4] 100mm/sec FINE; //Robot moves to approach position P[4].

: CALL RSWTCHDIRTOL(3,1,0,0); //Robot moves from P[4] in the +X direction
of the active tool frame until the probe
triggers. Server stores position data in
surface ID 3.

: L P[5] 100mm/sec FINE; //Robot moves to approach position P[5].

: CALL RSWTCHDIRTOL(4,0,(-1),0); //Robot moves from P[5] in the -Y direction
of the active tool frame until the probe
triggers. Server stores position data in
surface ID 4.

: L P[6] 100mm/sec FINE; //Robot moves to approach position P[6]

: CALL RSWTCHDIRTOL(4,0,(-1),1); //Robot moves from P[6] 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 user frame

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 user frame.

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 user frame 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 2.3.5 on page 14).

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

Syntax

RSWTCHPOS(SurfaceID, NominalPositionX, NominalPositionY, NominalPositionZ, TouchDirectionX, TouchDirectionY, TouchDirectionZ, ApproachDistance)

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

Example:

```

: L P[1] 100mm/sec FINE; //Robot moves to position P[1].

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

: L P[2] 100mm/sec FINE; //Robot moves to position P[2].

: CALL RSWTCHPOS(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 user frame 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 user frame, 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 2.3.5 on page 14).

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 user frame. 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

```
RSWTCHCIRIN(SurfaceID, TouchDirectionX, TouchDirectionY, TouchDirectionZ,  
NormalDirectionX, NormalDirectionY, NormalDirectionZ, CircleDiameter,  
ApproachDistance)
```

```
RSWTCHCIREXT(SurfaceID, TouchDirectionX, TouchDirectionY, TouchDirectionZ,  
NormalDirectionX, NormalDirectionY, NormalDirectionZ, CircleDiameter,  
ApproachDistance)
```

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

Example:

```
: L P[5] 100mm/sec FINE;
```

```
//Robot moves to approach position P[5].
```

```
: CALL RSWTCHCIRIN(7,1,0,0,0,0,(-1),75,10);
```

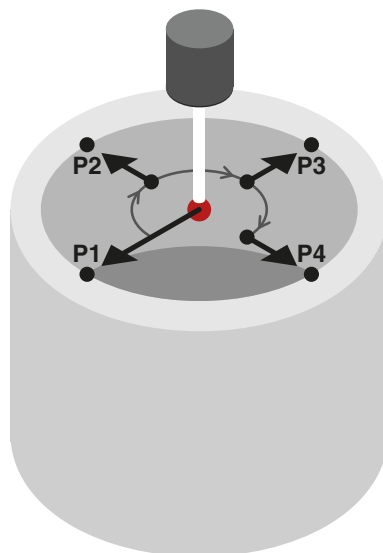
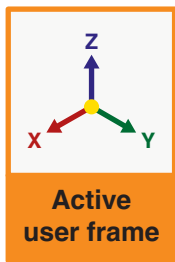
```
//Robot will measure an internal circle:  
Robot moves from P[5] in the +X direction  
of the active user frame until the probe  
triggers. The circle has a normal direction  
of -Z and diameter of 75 mm. Subsequent  
touch points will have an approach distance  
of 10 mm. Server stores position data in  
surface ID 7.
```

```
: L P[6] 100mm/sec FINE;
```

```
//Robot moves to approach position P[6].
```

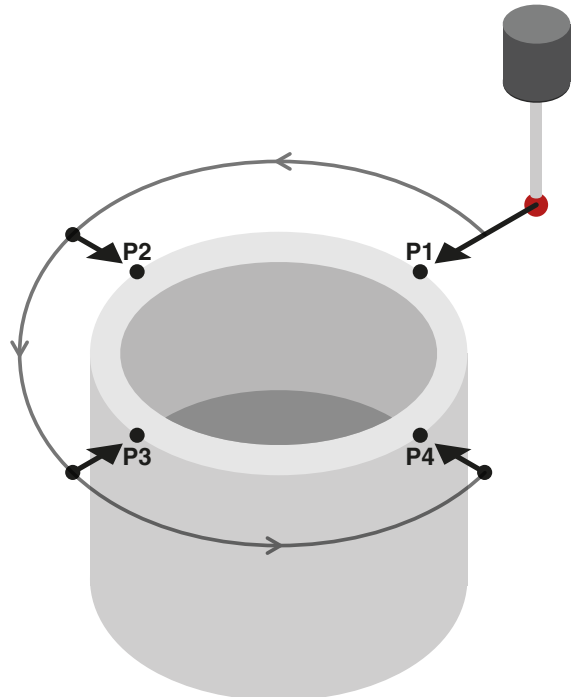
```
: CALL RSWTCHCIREXT(8,1,0,0,0,0,1,85,10);
```

```
//Robot will measure an external circle:  
Robot moves from P[6] in the +X direction  
of the active user frame 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.
```



Internal circle

Touch direction = +X
Normal direction = -Z



External circle

Touch direction = +X
Normal direction = +Z

All directions are given within the active user frame

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 user frame.
- 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 user frame 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 2.3.5 on page 14).

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

RSWTCHSPH(SurfaceID, SphereDiameter, ApproachDistance)

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

Example:

: CALL RSWPRBSLCTMB;

//Selects mobile probe (see page 16).

: CALL RSWPRBPWRON;

//Turns wireless probe on (see page 17).

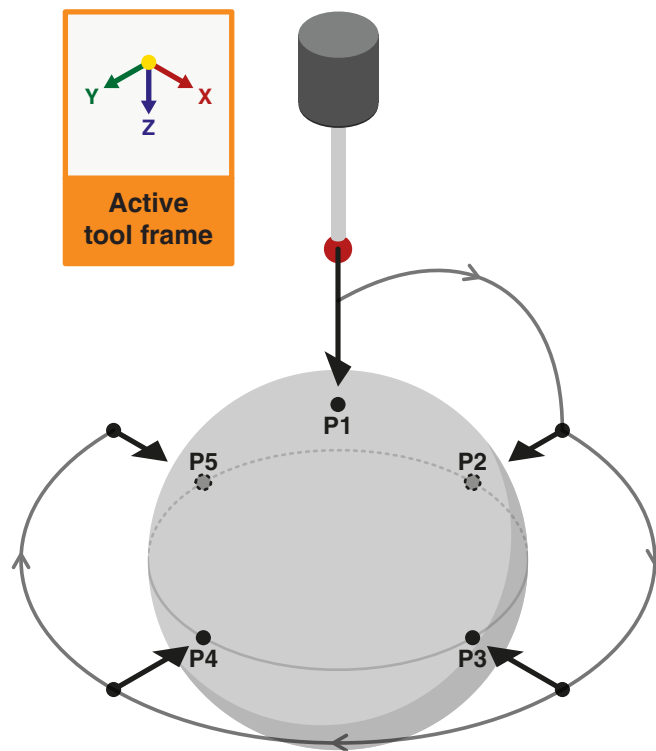
: L P[6] 100mm/sec FINE;

//Robot moves to approach position P[6]

: CALL RSWTCHSPH(9,40,25.5);

//Robot measures a sphere with a diameter of 40 mm. It moves from P[6] 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.



Touch direction = +Z in the active tool frame

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 RswTchSphStm 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 30).

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 user frame.
- 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 16).
- Active user frame if you have selected the tool setter (see “2.4.2 Select tool setter (fixed probe)” on page 16).

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 32).

Syntax

RSWSTEMREC

Example:

```
: CALL RSWPCINIT(16,3,2);
```

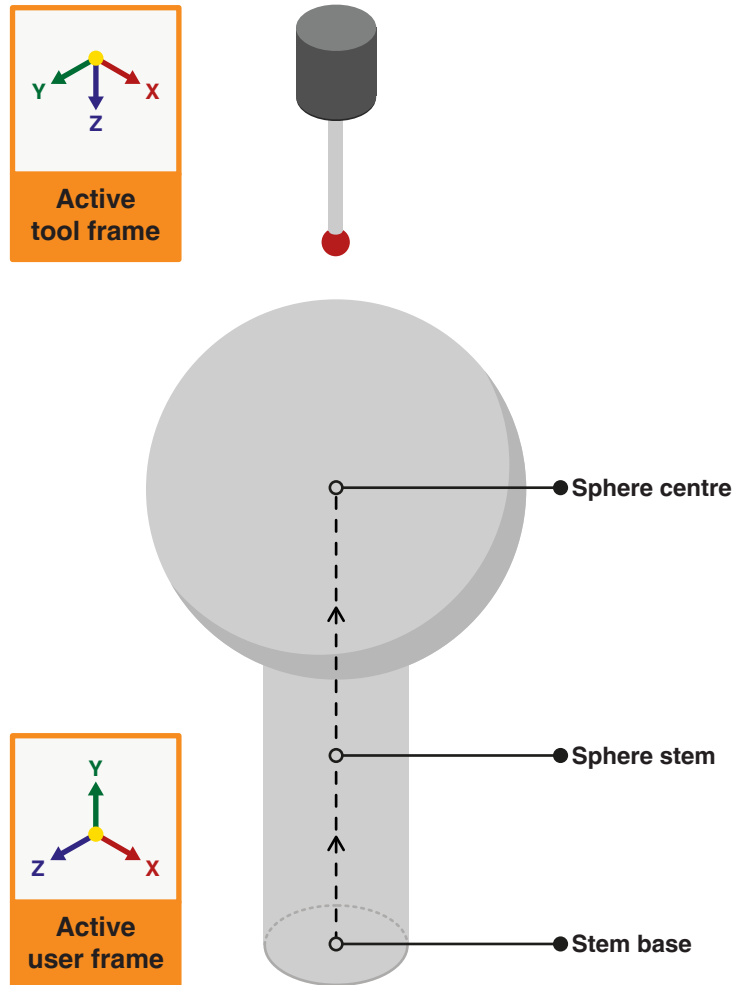
//Server initialises probe calibration (see page 33). Robot activates UserFrame[3] and ToolFrame[2]

```
: CALL RSWPRBSLCTMB;
```

//Selects mobile probe (see page 16).

```
: CALL RSWSTEMREC;
```

//Records the -Z direction of ToolFrame[2]. and sets the equivalent direction in UserFrame[3] as the stem direction.



Selected probe = Mobile
Stem direction = +Y in the active user frame

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 RswTchSphStm 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 30).

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 user frame if you have selected the mobile probe (see “2.4.1 Select mobile probe” on page 16).
- Active tool frame if you have selected the tool setter (see “2.4.2 Select tool setter (fixed probe)” on page 16).

This macro is primarily used during probe and tool setter calibration (see 2.6 on page 32).

Syntax

RSWSTEMSET(StemDirectionX, StemDirectionY, StemDirectionZ)

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

Example:

```
: CALL RSWPCINIT(16,3,2);
```

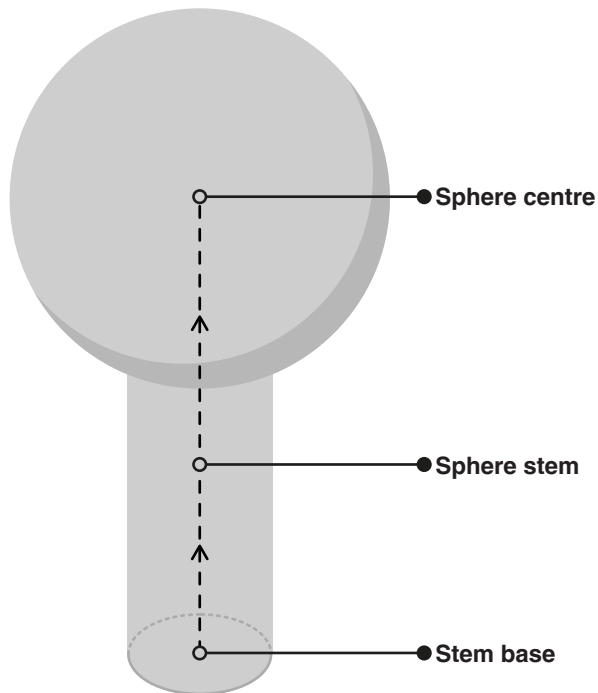
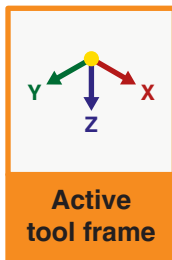
//Server initialises probe calibration (see page 33).

```
: CALL RSWPRBSLCTFX;
```

//Selects tool setter (see page 16).

```
: CALL RSWSTEMSET(0,0,(-1));
```

//Sets the -Z direction of ToolFrame[2] 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 user frame.
- Move your robot so that the probe points towards the centre of the sphere.
- Define the sphere stem direction using either RswStemRec (see 2.5.6 on page 26) or RwsStemSet (see 2.5.7 on page 28).

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 user frame 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 2.3.5 on page 14).

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

RSWTCHSPHSTM(SurfaceID, SphereDiameter, ApproachDistance)

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

Example:

```

: CALL RSWPRBSLCTMB;

: CALL RSWPRBPWRON;

: CALL RSWSTEMSET((-1), (-1), 0);

: L P[6] 100mm/sec FINE;

: CALL RSWTCHSPHSTM(10, 40, 25.5);

```

//Selects mobile probe (see page 16).

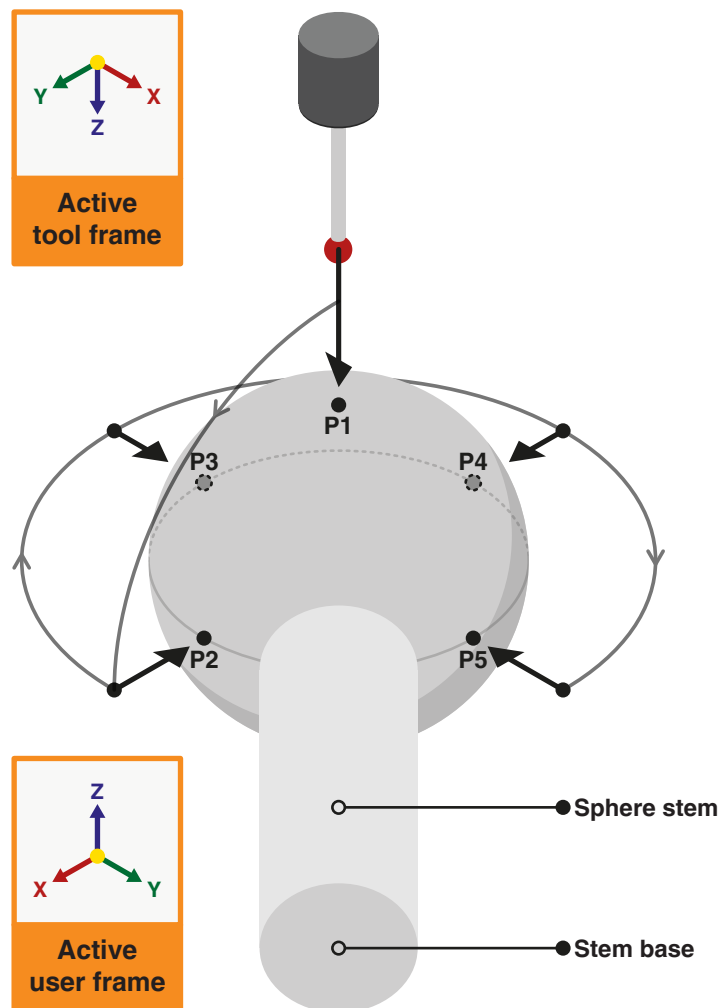
//Turns wireless probe on (see page 17).

//Sets the -X-Y direction of the active user frame as the stem direction (see page 28).

//Robot moves to approach position P[6].

//Robot measures a sphere with a 40 mm diameter. It moves from P[6] 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 user frame

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 18) to collect data for calibration.

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

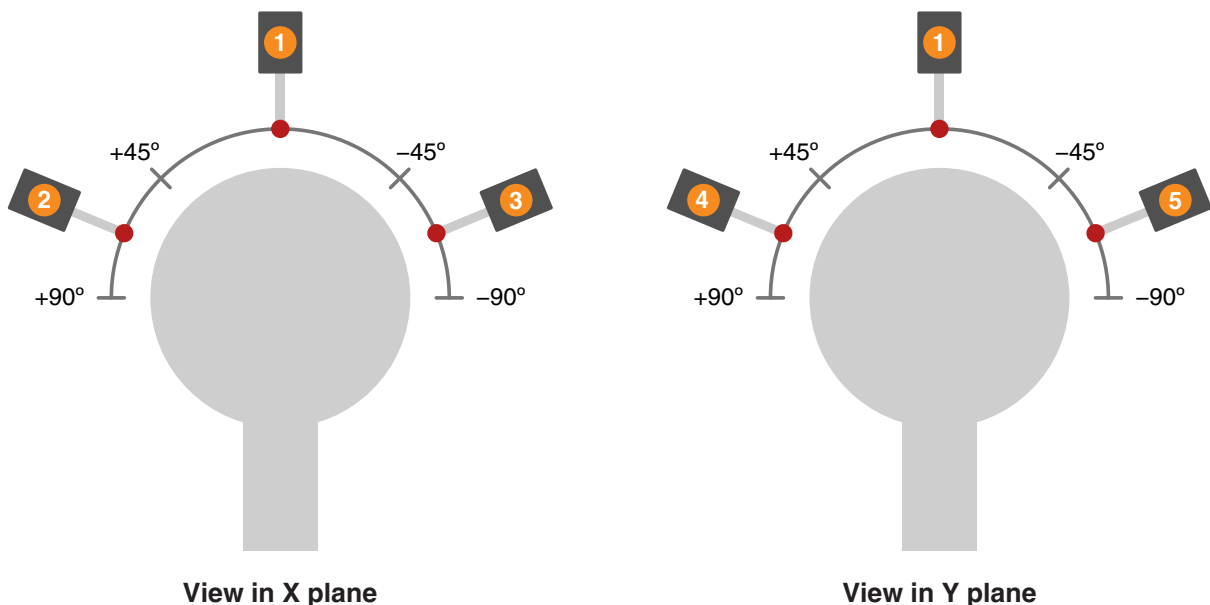
There are two possible ways to gather data for tool setter probe calibration. These are:

- 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 both mobile probes and tool setters, if you are using a calibration sphere, call RswTchSphStm (see 2.5.8 on page 30) 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 user frame direction” on page 19, 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 RswTchSphStm 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 user frame 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 latest nominal value loaded from the RswPrmTouch.xml file (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

RSWPCINIT(SurfaceID, UserFrame, ToolFrame)

Argument	Definition
SurfaceID	Integer. The surface ID to discard existing data from.
UserFrame	Integer. The user frame index to activate for the probe calibration.
ToolFrame	Integer. The tool frame index to activate for the probe calibration

Example:

```
: CALL RSWPCINIT(15,10,12);
```

//Server discards existing data from surface ID 15.

//Robot activates UserFrame[10] and ToolFrame[12] 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 index 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

RSWPCRUN(SurfaceID, NewToolFrame)

Argument	Definition
SurfaceID	Integer. The surface ID which contains position data for the calibration sphere.
NewToolFrame	Integer. The tool frame index where the robot should write the calculated frame data.

Example:

```
: CALL RSWPCRUN(15,13);
```

//Server calculates a mobile probe TCP using data stored in surface ID 15.

//Robot writes the calibrated probe TCP to ToolFrame[13]. 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 user frame 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 latest nominal value loaded from the RswPrmTouch.xml file (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

RSWTSCALINIT(SurfaceID, UserFrame, ToolFrame)

Argument	Definition
SurfaceID	Integer. The surface ID to discard existing data from.
UserFrame	Integer. The user frame index to activate for the tool setter calibration.
ToolFrame	Integer. The tool frame index to activate for the tool setter calibration

Example:

```
: CALL RSWTSCALINIT(16,10,12);
```

```
//Server discards existing data from  
surface ID 16.
```

```
//Robot activates UserFrame[10] and  
ToolFrame[12] 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 user frame 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 user frame index 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

RSWTSCALRUN (SurfaceID, NewUserFrame)

Argument	Definition
SurfaceID	Integer. The surface ID which contains position data for the calibration sphere.
NewUserFrame	Integer. The user frame index where the robot should write the calculated frame data.

Example:

```
: CALL RSWTSCALRUN (16, 11);
```

//Server calculates a tool setter user frame using data stored in surface ID 16.

//Robot writes the calibrated tool setter user frame to UserFrame[11]. 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 user frame 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 user frame index 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

RSWTSCALPRUN (SurfaceID, NewUserFrame)

Argument	Definition
SurfaceID	Integer. The surface ID which contains position data for the calibrated mobile probe.
NewUserFrame	Integer. The user frame index where the robot should write the calculated frame data.

Example:

```
: CALL RSWTSCALPRUN (16, 11);
```

//Server calculates a tool setter user frame using data stored in surface ID 16.

//Robot writes the calibrated tool setter user frame to UserFrame[11]. Robot also updates the current tool setter tip diameter.

2.7 Simple alignment

Simple alignment macros use feature position data to calculate user frames 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 on page 18) 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 32).

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 user frame 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

```
RSWALGNINIT(SurfaceID1, SurfaceID2, SurfaceID3, UserFrame, ToolFrame)
```

Argument	Definition
SurfaceID1	Integer. The first surface ID to discard existing data from.
SurfaceID2	Integer. The second surface ID to discard existing data from.
SurfaceID3	Integer. The third surface ID to discard existing data from.
UserFrame	Integer. The user frame index to activate for the alignment.
ToolFrame	Integer. The tool frame index to activate for the alignment.

Example:

```
: CALL RSWALGNINIT (1,2,3,4,5);  
  
//Server discards existing data from  
//surface IDs 1, 2 and 3.  
  
//Robot activates UserFrame[4] and  
//ToolFrame[5] 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 user frame or a tool frame. The robot will write the frame result to the index you provide.

Syntax

```
RSWALGN321(SurfaceID1, SurfaceID2, SurfaceID3, FrameType, NewFrame)
```

Argument	Definition
SurfaceID1	Integer. The surface ID which contains position data for the first plane (P1, P2, P3).
SurfaceID2	Integer. The surface ID which contains position data for the second plane (P4, P5).
SurfaceID3	Integer. The surface ID which contains position data for the third plane (P6).
FrameType	Integer. 0 identifies a user frame, 1 identifies a tool frame.
NewFrame	Integer. The frame index where the robot should write the calculated frame data.

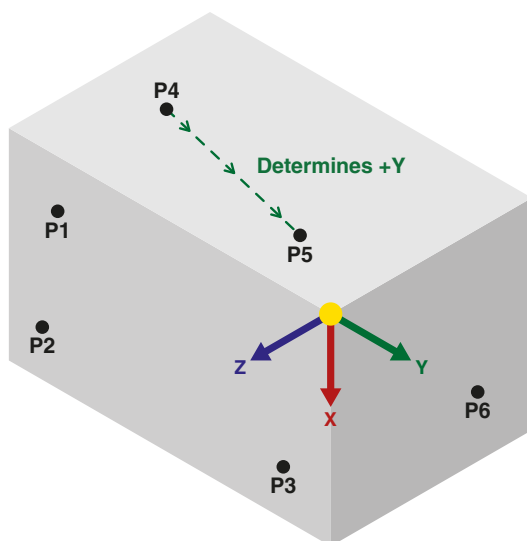
Example:

```
: CALL RSWALGN321(10,11,12,0,4);
```

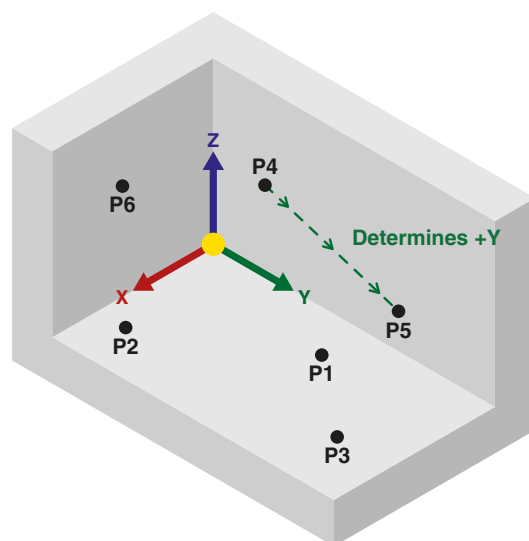
```
//Server calculates a user frame with a  
321 alignment.
```

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

```
//Robot writes frame result to UserFrame[4].
```



321 alignment outside of a box



321 alignment inside of a box

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 user frame or a tool frame. The robot will write the frame result to the index you provide.

Syntax

```
RSWALGNPPP(SurfaceID1, SurfaceID2, SurfaceID3, FrameType, NewFrame)
```

Argument	Definition
SurfaceID1	Integer. The surface ID which contains position data for the first plane (Plane 1).
SurfaceID2	Integer. The surface ID which contains position data for the second plane (Plane 2).
SurfaceID3	Integer. The surface ID which contains position data for the third plane (Plane 3).
FrameType	Integer. 0 identifies a user frame, 1 identifies a tool frame.
NewFrame	Integer. The frame index where the robot should write the calculated frame data.

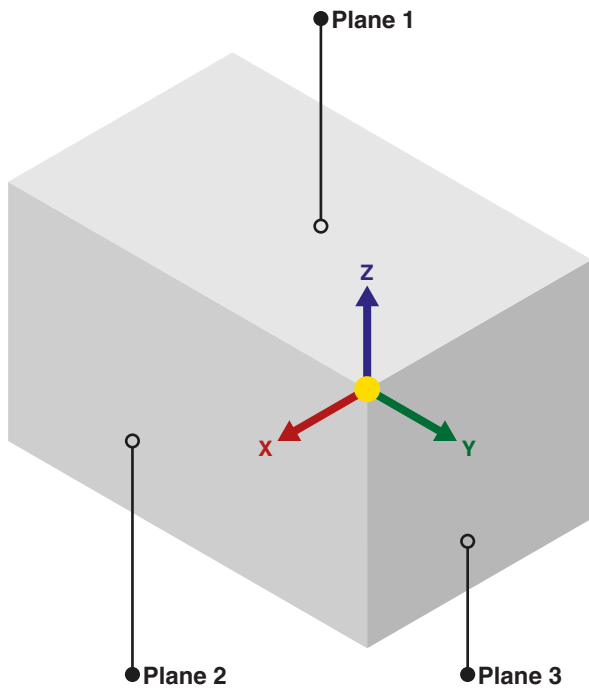
Example:

```
: CALL RSWALGNPPP(1,2,3,0,4);
```

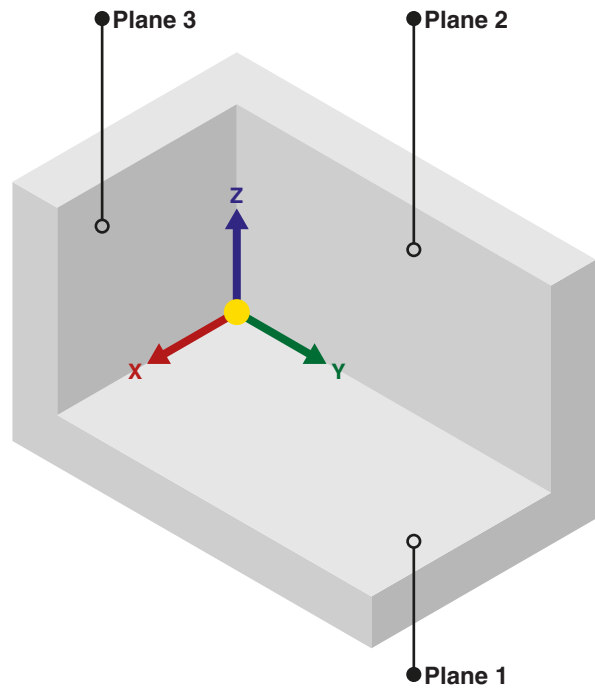
```
//Server calculates a user frame with a plane-  
plane-plane alignment.
```

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

```
//Robot writes frame result to UserFrame[4].
```



**Plane-plane-plane alignment
outside of a box**



**Plane-plane-plane alignment
inside of a box**

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 22).

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 user frame or a tool frame. The robot will write the frame result to the index you provide.

Syntax

```
RSWALGNPCC(SurfaceID1, SurfaceID2, SurfaceID3, FrameType, NewFrame)
```

Argument	Definition
SurfaceID1	Integer. The surface ID which contains position data for a plane (P1, P2, P3).
SurfaceID2	Integer. The surface ID which contains position data for the first cylinder (C2).
SurfaceID3	Integer. The surface ID which contains position data for the second cylinder (C3).
FrameType	Integer. 0 identifies a user frame, 1 identifies a tool frame.
NewFrame	Integer. The frame index where the robot should write the calculated frame data.

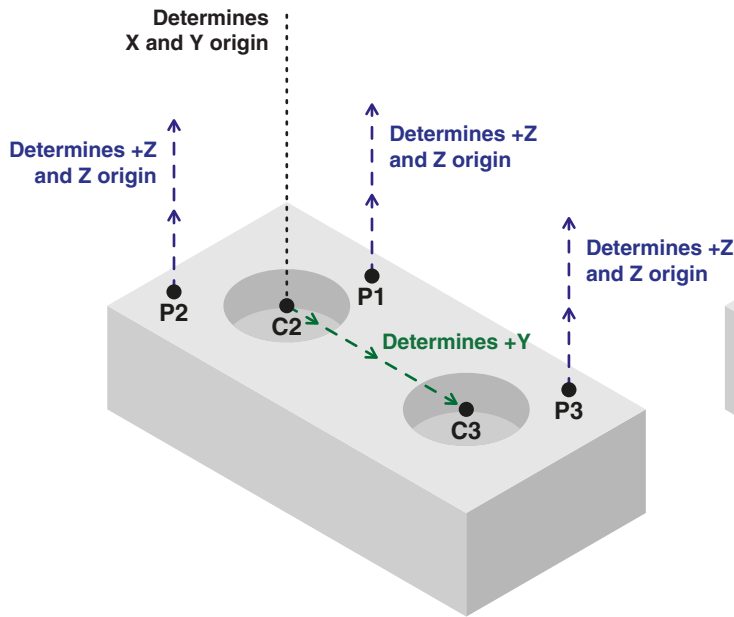
Example:

```
: CALL RSWALGNPCC (10,11,12,1,4);
```

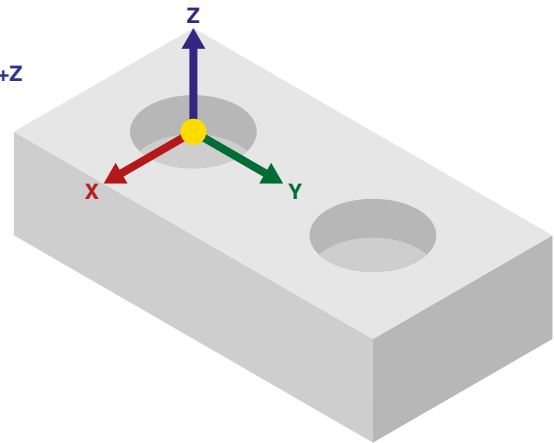
```
//Server calculates a tool frame with a  
plane-cylinder-cylinder alignment.
```

```
//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 ToolFrame[4].
```



Plane-cylinder-cylinder alignment



Frame result

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 user frame or a tool frame. The robot will write the frame result to the index you provide.

Syntax

RSWALGNPPPNT(SurfaceID1, SurfaceID2, SurfaceID3, FrameType, NewFrame)

Argument	Definition
SurfaceID1	Integer. The surface ID which contains position data for Plane 1.
SurfaceID2	Integer. The surface ID which contains position data for Plane 2.
SurfaceID3	Integer. The surface ID which contains position data for a point (P3).
FrameType	Integer. 0 identifies a user frame, 1 identifies a tool frame.
NewFrame	Integer. The frame index where the robot should write the calibrated frame data.

Example:

```
: CALL RSWALGNPPPNT(11,22,33,0,4);
```

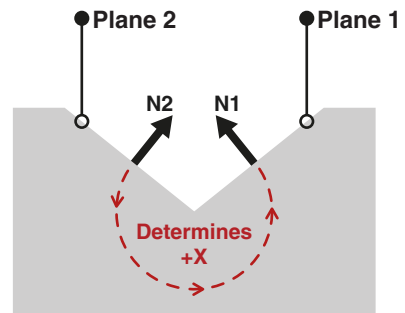
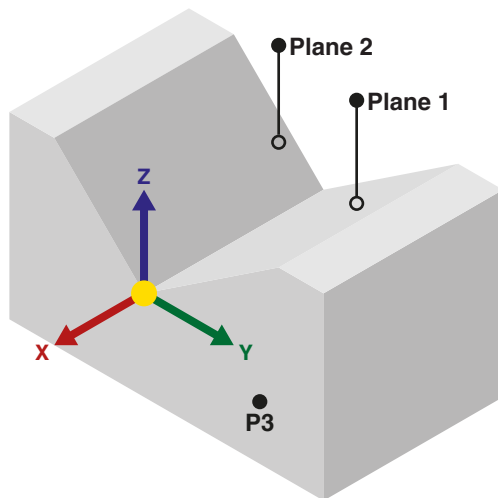
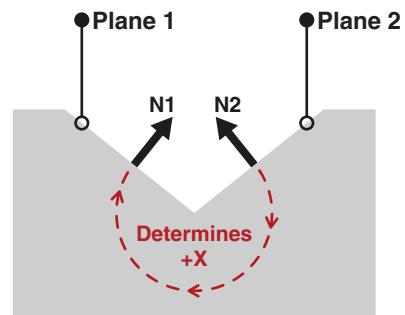
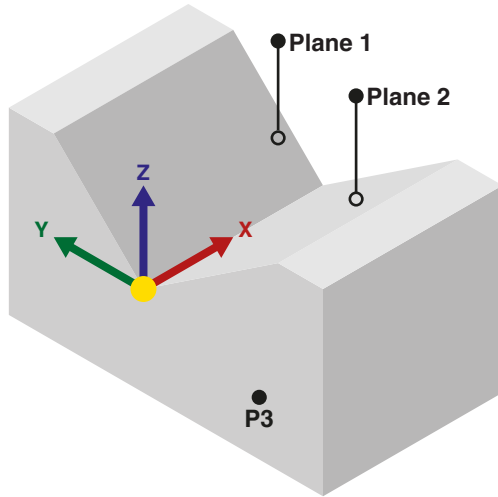
```
//Server calculates a user frame with a plane-  
plane-point alignment.
```

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

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

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

```
//Robot writes frame result to UserFrame[4].
```



Two examples of a plane-plane-point alignment

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

2.7.6 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 user frame or a tool frame. The robot will write the frame result to the index you provide.

Syntax

```
RSWALGNSSS(SurfaceID1, SurfaceID2, SurfaceID3, FrameType, NewFrame)
```

Argument	Definition
SurfaceID1	Integer. The surface ID which contains position data for the first sphere (S1).
SurfaceID2	Integer. The surface ID which contains position data for the second sphere (S2).
SurfaceID3	Integer. The surface ID which contains position data for the third sphere (S3).
FrameType	Integer. 0 identifies a user frame, 1 identifies a tool frame.
NewFrame	Integer. The frame index where the robot should write the calculated frame data.

Example:

```
: CALL RSWALGNSSS(10,11,12,1,4);
```

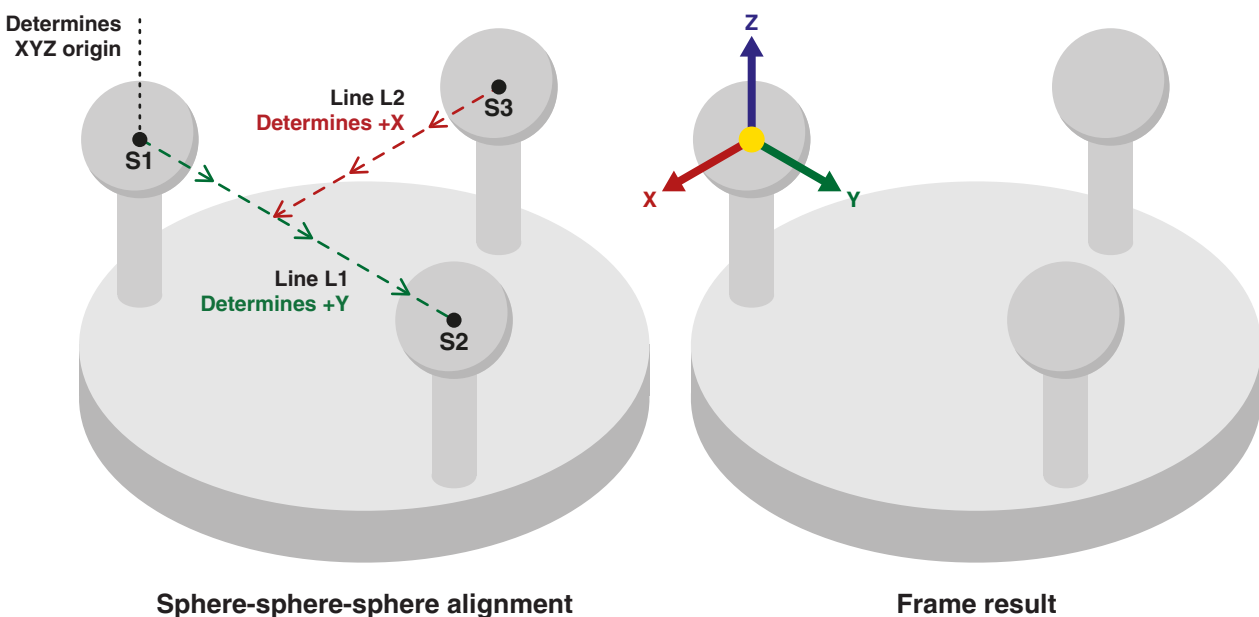
```
//Server calculates a user frame with a  
sphere-sphere-sphere alignment.
```

```
//Surface ID 10 contains S1 data.
```

```
//Surface ID 11 contains S2 data.
```

```
//Surface ID 12 contains S3 data.
```

```
//Robot writes frame result to ToolFrame[4].
```



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 user frame and tool frame for the TCP calibration. The robot sends all active frame data to the Server.

Syntax

RSWTCPINIT(SurfaceID, UserFrame, ToolFrame)

Argument	Definition
SurfaceID	Integer. The surface ID to discard existing data from.
UserFrame	Integer. The user frame index to activate for the TCP calibration.
ToolFrame	Integer. The tool frame index to activate for the TCP calibration.

Example:

```
: CALL RSWTCPINIT(20,10,12);
```

//Server discards existing data from surface ID 20.

//Robot activates UserFrame[10] and ToolFrame[12] 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 user frame. The robot will write the calculated TCP to the given tool frame index.

Syntax

```
RSWTCPPANE(SurfaceID, PlaneNormalX, PlaneNormalY, PlaneNormalZ, NewToolFrame)
```

Argument	Definition
SurfaceID	Integer. The surface ID which contains position data of the spherical tool measured on the disc stylus.
PlaneNormalX	Number. The X component of the disc stylus plane normal direction.
PlaneNormalY	Number. The Y component of the disc stylus plane normal direction.
PlaneNormalZ	Number. The Z component of the disc stylus plane normal direction.
NewToolFrame	Integer. The tool frame index where the robot should write the calibrated TCP data.

Example:

```
: CALL RSWTCPPANE(20,0,0,1,13);
```

```
//Server calculates a tool TCP using data  
stored in surface ID 20. The plane normal  
direction is +Z in the active user frame.
```

```
//Robot writes frame result to ToolFrame[13].
```

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 index.

Syntax

```
RSWTCPSPH(SurfaceID, ToolDiameter, NewToolFrame)
```

Argument	Definition
SurfaceID	Integer. The surface ID which contains position data of the spherical tool measured on the spherical stylus.
ToolDiameter	Number. The diameter of the spherical tool end in millimetres.
NewToolFrame	Integer. The tool frame index where the robot should write the calibrated TCP data.

Example:

```
: CALL RSWTCPSPH(20,25,13);
```

```
//Server calculates the TCP of a tool using  
data stored in surface ID 20. The tool has a  
spherical end with diameter 25 mm.
```

```
//Robot writes frame result to ToolFrame[13].
```

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 36)
- Complex alignments (see 3.1 on page 48)
- Probe calibration (see 2.6 on page 32)
- 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 registers you provide.

Syntax

RSWSTATRES(Average, Min, Max, AbsMax, StdDev, RMS, NbVal, MinPoint, MaxPoint)

Argument	Definition
Average	Integer. The register index to write the average residual error value.
Min	Integer. The register index to write the minimum residual error value.
Max	Integer. The register index to write the maximum residual error value.
AbsMax	Integer. The register index to write the absolute maximum residual error value (see 5.11.1 on page 69).
StdDev	Integer. The register index to write the standard deviation of residual errors.
RMS	Integer. The register index to write the root mean square (RMS) of the residual error values.
NbVal	Integer. The register index to write the total number of touch points used in the calculation.
MinPoint	Integer. The register index to write the touch point which had the minimum residual error value.
MaxPoint	Integer. The register index to write the touch point which had the maximum residual error value.

Example:

```

: CALL RSWPCRUN(15,13); //Server calculates a probe calibration using
                        //data stored in surface ID 15.

: CALL RSWSTATRES(1,2,3,4,5,6,7,8,9); //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 registers.

```

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 user frame and tool frame for the alignment. The robot sends all active frame data to the Server.

Syntax

RSWDBINIT(DatumConfigID, UserFrame, ToolFrame)

Argument	Definition
DatumConfigID	Integer. 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.
UserFrame	Integer. The user frame index to activate for the alignment.
ToolFrame	Integer. The tool frame index to activate for the alignment.

Example:

```
: CALL RSWDBINIT(15,2,3); //Server discards existing data from
                           //surface IDs referenced in DatumConfig15.

                           //Robot activates UserFrame[2] and
                           //ToolFrame[3] 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.

Syntax

RSWDBRUN (DatumConfigID, FrameType, NewFrame)

Argument	Definition
DatumConfigID	Integer. 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.
FrameType	Integer. 0 identifies a user frame, 1 identifies a tool frame.
NewFrame	Integer. The frame index where the robot should write the calculated result.

Example:

```
: CALL RSWDBRUN(15,0,4); //Server calculates a co-ordinate system
                           using surfaces referenced in the file
                           DatumConfig15.

                           //Robot writes frame result to UserFrame[4].
```

3.1.3 Run the datum builder to a position register

This macro tells the Server to calculate a co-ordinate system. You can define the co-ordinate system within the World frame, or the active user frame. 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 position register.

If you generate your robot program using RCS Software Suite, you will need to edit it to include this macro. Open the program and replace the RswDbRun macro with RswDbRunPR. Edit the argument values within the macro if required.

Syntax

RSWDBRUNPR (DatumConfigID, FrameOrigin, NewPR)

Argument	Definition
DatumConfigID	Integer. 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.
FrameOrigin	Integer. 0 to set the co-ordinate system within the World frame, 1 to set it within the active user frame.
NewPR	Integer. The position register index where the robot should write the calculated result.

Example:

```
: CALL RSWDBRUNPR(15,0,4); //Server calculates a co-ordinate system
                             using surfaces referenced in the file
                             DatumConfig15.

                             //Robot writes the result to PR[4], set within
                             the World 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 18) 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

RSWBFINIT (SurfaceID)

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

Example:

```
: CALL RSWBFINIT(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.

The robot will write the fitted plane position to a position register (PlanePR). The robot will write the fitted plane normal as a vector in the XYZ components of another position register (NormalPR). The Server will store the best fit result in the best fit ID you provide for future reference.

Syntax

RSWBFPLN(SurfaceID, BestFitID, PlanePR, NormalPR)

Argument	Definition
SurfaceID	Integer. The surface ID which contains position data for a plane.
BestFitID	Integer. The best fit ID where the Server should store the fitted plane data for future reference.
PlanePR	Integer. The position register index where the robot should write the fitted plane position.
NormalPR	Integer. The position register index where the robot should write the fitted plane normal.

Example:

```
: CALL RSWBFPLN(15,102,10,11);
```

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

```
//Robot writes the fitted plane position  
to PR[10], and the fitted plane normal to  
PR[11].
```

```
//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.

The robot will write the fitted sphere centre to a position register (CentrePR). The robot will write the fitted sphere diameter to a register (DiameterReg). The Server will store the best fit result in the best fit ID you provide for future reference.

Syntax

```
RSWBFSPHR(SurfaceID, BestFitID, BoreBoss, CentrePR, DiameterReg)
```

Argument	Definition
SurfaceID	Integer. The surface ID which contains position data for a sphere.
BestFitID	Integer. The best fit ID where the Server should store the fitted sphere data for future reference.
BoreBoss	Integer. -1 refers to a bore sphere, 1 refers to a boss sphere.
CentrePR	Integer. The position register index where the robot should write the fitted sphere centre.
DiameterReg	Integer. The register index where the robot should write the fitted sphere diameter in millimetres.

Example:

```
: CALL RSWBFSPHR(16,103,1,10,11);
```

```
//Server calculates a best fit for a boss  
sphere using data stored in surface ID 16.
```

```
//Robot writes the fitted sphere centre to  
PR[10], and the fitted sphere diameter to  
R[11].
```

```
//Server stores the best fit result in  
best fit ID 103.
```

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.

The robot will write the fitted cylinder centre to a position register (CentrePR). The robot will write the fitted cylinder direction as a vector in the XYZ components of another position register (DirectionPR). The robot will write the fitted cylinder diameter to a register (DiameterReg). The Server will store the best fit result in the best fit ID you provide for future reference.

Syntax

RSWBFCYL(SurfaceID, BestFitID, BoreBoss, CentrePR, DirectionPR, DiameterReg)

Argument	Definition
SurfaceID	Integer. The surface ID which contains position data for a cylinder.
BestFitID	Integer. The best fit ID where the Server should store the fitted cylinder data for future reference.
BoreBoss	Integer. -1 refers to a bore cylinder, 1 refers to a boss cylinder.
CentrePR	Integer. The position register index where the robot should write the fitted cylinder centre.
DirectionPR	Integer. The position register index where the robot should write the fitted cylinder direction.
DiameterReg	Integer. The register index where the robot should write the fitted cylinder diameter in millimetres.

Example:

```
: CALL RSWBFCYL(17,104,1,10,11,12);
```

//Server calculates a best fit for a boss cylinder using data stored in surface ID 17.

//Robot writes the fitted cylinder centre to PR[10], and the fitted cylinder axis direction to PR[11].

//Robot writes the fitted cylinder diameter to R[12].

//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 user frame or tool frame.

The robot will write the fitted circle centre to a position register (CentrePR). The robot will write the fitted circle diameter to a register (DiameterReg). The Server will store the best fit result in the best fit ID you provide for future reference.

Syntax

RSWBFCIR(SurfaceID, BestFitID, BoreBoss, NormalDirectionX, NormalDirectionY, NormalDirectionZ, CentrePR, DiameterReg)

Argument	Definition
SurfaceID	Integer. The surface ID which contains position data for a 2D circle.
BestFitID	Integer. The best fit ID where the Server should store the fitted circle data for future reference.
BoreBoss	Integer. -1 refers to a bore circle, 1 refers to a boss circle.
NormalDirectionX	Number. The X component of the circle plane normal direction.
NormalDirectionY	Number. The Y component of the circle plane normal direction.
NormalDirectionZ	Number. The Z component of the circle plane normal direction..
CentrePR	Integer. The position register index where the robot should write the fitted circle centre.
DiameterReg	Integer. The register index where the robot should write the fitted circle diameter in millimetres.

Example:

```
: CALL RSWBFCIR(18,105, (-1), 0, 0, 1, 4, 2); //Server calculates a best fit for a bore 2D circle using data stored in surface ID 18.

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

//Robot writes the fitted circle centre to PR[4].

//Robot writes the fitted circle diameter to R[2].

//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.

The robot will write the fitted cone centre to a position register (CentrePR). The robot will write the fitted cone direction to another position register (DirectionPR). The robot will write the fitted cone diameter and half angle to registers (DiameterReg and AngleReg). The Server will store the best fit result in the best fit ID you provide for future reference.

Syntax

RSWBFCONE (SurfaceID, BestFitID, BoreBoss, CentrePR, DirectionPR, DiameterReg, AngleReg)

Argument	Definition
SurfaceID	Integer. The surface ID which contains position data for a cone.
BestFitID	Integer. The best fit ID where the Server should store the fitted cone data for future reference.
BoreBoss	Integer. -1 refers to a bore cone, 1 refers to a boss cone.
CentrePR	Integer. The position register index where the robot should write the fitted cone centre.
DirectionPR	Integer. The position register index where the robot should write the fitted cone direction.
DiameterReg	Integer. The register index where the robot should write the fitted cone diameter in millimetres.
AngleReg	Integer. The register index where the robot should write the fitted cone half angle in degrees.

Example:

```

: CALL RSWBFCONE (19,106,1,10,11,12,13);           //Server calculates a best fit for a boss cone
                                                    using data stored in surface ID 19.

                                                    //Robot writes the fitted cone centre to
                                                    PR[10], and the fitted cone axis direction to
                                                    PR[11].

                                                    //Robot writes the fitted cone diameter to
                                                    R[12], and the half angle to R[13].

                                                    //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. The robot will write the result to the register index you provide.

Syntax

```
RSWANLSANGLE (BestFitID1, BestFitID2, AngleReg)
```

Argument	Definition
BestFitID1	Integer. The best fit ID which contains calculated data for the first best fit feature.
BestFitID2	Integer. The best fit ID which contains calculated data for the second best fit feature.
AngleReg	Integer. The register index where the robot should write the calculated angle between features in degrees.

Example:

```
: CALL RSWBFCYL (15, 115, 1, 10, 11, 12, 13);           //Server calculates a best fit for a cylinder. It
                                                         stores the result in best fit ID 115.

: CALL RSWBFCYL (16, 116, 1, 20, 21, 22, 23);           //Server calculates a best fit for another
                                                         cylinder. It stores the result in
                                                         best fit ID 116.

: CALL RSWANLSANGLE (115, 116, 30);                     //Server calculates the angle between the
                                                         two best fitted cylinders using data stored in
                                                         best fit IDs 115 and 116.

                                                         //Robot writes the result to R[30].
```


3.3.2 Distance between features

This macro tells the Server to calculate the distance between two best fitted features. The robot will write the result to the register index you provide.

Syntax

RSWANLSDIST(BestFitID1, BestFitID2, DistanceReg)

Argument	Definition
BestFitID1	Integer. The best fit ID which contains calculated data for the first best fit feature.
BestFitID2	Integer. The best ID which contains calculated data for the second best fit feature.
DistanceReg	Integer. The register where the robot should write the calculated distance between features in millimetres.

Example:

```

: CALL RSWBFCYL(15,115,1,10,11,12,13);           //Server calculates a best fit for a cylinder. It
                                                    stores the result in best fit ID 115.

: CALL RSWBFCYL(16,116,1,20,21,22,23);           //Server calculates a best fit for another
                                                    cylinder. It stores the result in
                                                    best fit ID 116.

: CALL RSWANLSDIST(115,116,31);                   //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 R[31].
  
```

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

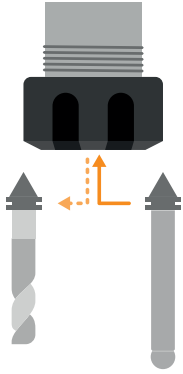
For robotic applications which use a cutting tool, accurate tool vectors and lengths must be set. This ensures tools will follow cutting paths and machine features correctly.

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 22). 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 22). This is used in “4.1.8 Set spindle cutting tool diameter” on page 66.

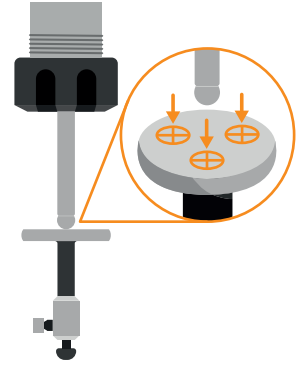
The Server stores the touch point data in the surface IDs you provide. You can reference these in the spindle calibration macros.



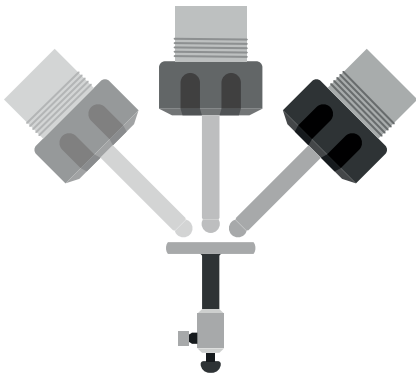
1 Swap tool with spindle artefact



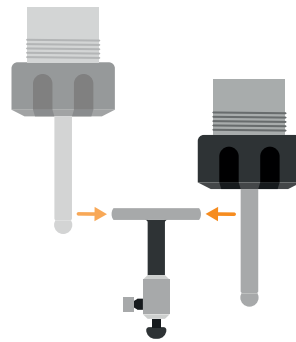
2 Manually set a nominal tool frame where +Z points out of the artefact



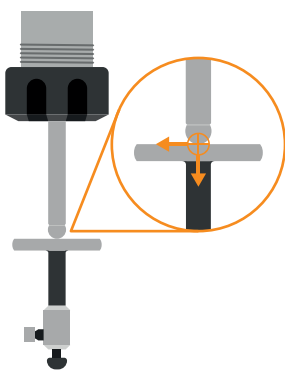
3 Take three touch points around top of disc stylus



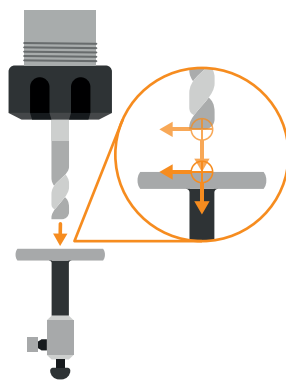
4 Use artefact on top of disc stylus in several orientations



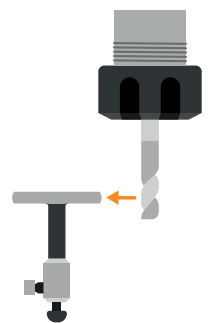
5 Use artefact on sides of disc stylus at different heights



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 user frame 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

```
RSWSPNCALINI(DiscID, ArtefactID, DirectionID, UserFrame, ToolFrame)
```

Argument	Definition
DiscID	Integer. The first surface ID to discard existing data from.
ArtefactID	Integer. The second surface ID to discard existing data from.
DirectionID	Integer. The third surface ID to discard existing data from.
UserFrame	Integer. The user frame index to activate for the spindle calibration.
ToolFrame	Integer. The tool frame index to activate for the spindle calibration.

Example:

```
: CALL RSWSPNCALINI (21,22,23,5,6);           //Server discards existing data from  
                                              surface IDs 21, 22 and 23.  
  
                                              //Robot activates UserFrame[5] and  
                                              ToolFrame[6] 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 user frame 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 user frame 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 user frame result to the user frame index you provide (UserFrame).
- Write the spindle artefact tool frame to the tool frame index you provide (ToolFrame).
- Write the reference tool frame at the spindle mounting point to the tool frame index you provide (RefToolFrame).

Syntax

RSWSPNCALRUN(DiscID, ArtefactID, DirectionID, ArtefactLength, NewUserFrame, NewToolFrame, RefToolFrame)

Argument	Definition
DiscID	Integer. The surface ID which contains the disc stylus plane position data.
ArtefactID	Integer. The surface ID which contains the artefact TCP position data.
DirectionID	Integer. The surface ID which contains the artefact direction position data.
ArtefactLength	Number. The length of your spindle artefact. Measure your artefact in place from the spindle mounting point to the centre of the artefact sphere.
NewUserFrame	Integer. The user frame index where the robot should write the tool setter disc stylus user frame.
NewToolFrame	Integer. The tool frame index where the robot should write the spindle artefact tool frame.
RefToolFrame	Integer. The tool frame index where the robot should write the reference tool frame.

Example:

```
: CALL RSWSPNCALRUN(21,22,23,98.5,5,6,7);
```

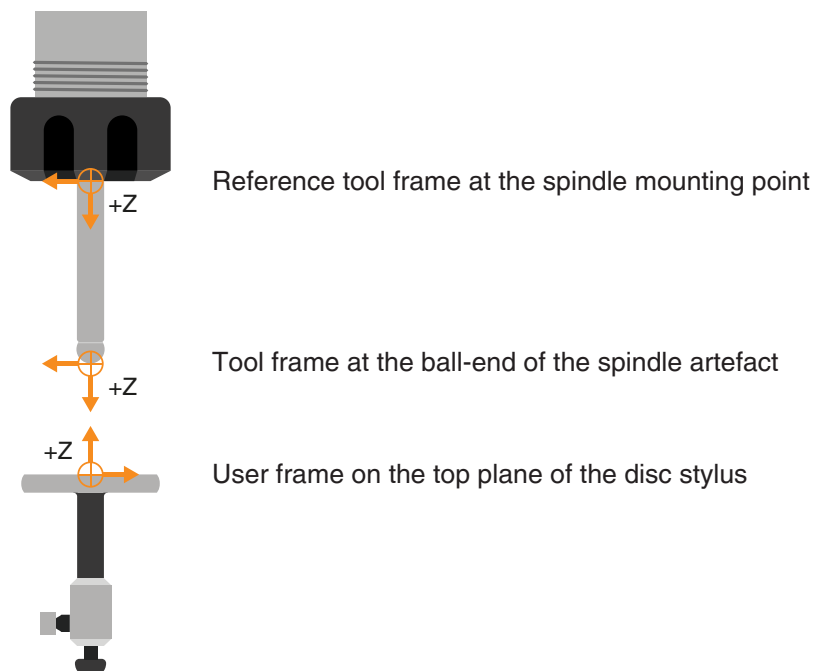
//Server uses data from surface IDs 21, 22 and 23 to calculate: a user frame for the disc stylus and a tool frame for the artefact TCP.

//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 UserFrame[5].

//Robot writes the tool frame result to ToolFrame[6].

//Robot writes the reference tool frame result to ToolFrame[7].



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 user frame. The active user frame should be the tool setter disc stylus which was calculated by running the spindle artefact calibration (see 4.1.2 on page 60).

Syntax

RSWSPNALIGN

Example:

```
: CALL RSWSPNALIGN; //Robot rotates the active tool TCP so that  
//its –Z direction aligns with the +Z direction  
//of the active user frame.
```

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 user frame 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, UserFrame, ToolFrame)

Argument	Definition
SurfaceID	Integer. The surface ID to discard existing data from.
UserFrame	Integer. The user frame index of the tool setter disc stylus.
ToolFrame	Integer. The tool frame index to activate for the measurement. This may be the spindle artefact TCP or spindle cutting tool TCP.

Example:

```
: CALL RSWSPNDLINIT(31, 6, 8); //Server discards existing data from  
//surface ID 31.  
  
//Robot activates UserFrame[6] and  
//ToolFrame[7] 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 user frame 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 given user frame index, which can be the same index 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

RSWSPNCALREF(SurfaceID, UserFrame)

Argument	Definition
SurfaceID	Integer. The surface ID which contains data of a single touch on the disc stylus plane using a calibrated spindle artefact.
UserFrame	Integer. The user frame index of the tool setter disc stylus. The robot will update the frame values to set the disc reference height.

Example:

```
: CALL RSWSPNCALREF(21,8);
```

```
//Server uses the data stored in  

  surface ID 21 to calculate a reference user  

  frame on top of the disc stylus.
```

```
//Robot writes the result to UserFrame[8].
```

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 (RefToolFrame) along the tool Z axis by the tool length you provide. The robot will write the result to the tool frame index 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)
```

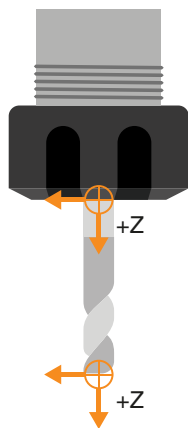
Argument	Definition
ToolLength	Number. The nominal length of your spindle cutting tool.
RefToolFrame	Integer. The tool frame index which contains the reference tool frame data.
NewToolFrame	Integer. The tool frame index where the robot should write the nominal spindle cutting tool TCP.

Example:

```
: CALL RSWSPNDLNOM(175, 7, 8);
```

```
//Server calculates a nominal TCP. It moves  
the reference tool frame ToolFrame[7] along  
the tool Z axis by 175 mm.
```

```
//Robot writes the tool frame result to  
ToolFrame[8].
```



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 index, which can be the same index you used for “4.1.6 Set nominal spindle cutting tool TCP” on page 64.

Syntax

RSWSPNDLTCP(SurfaceID, NewToolFrame)

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

Example:

```
: CALL RSWSPNDLTCP(66,8);
```

//Server calculates an accurate TCP using data stored in surface ID 66.

//Robot writes the tool frame result to ToolFrame[8].

4.1.8 Set spindle cutting tool diameter

This macro tells the 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 22) to do this. This is step 8 on page 58.

The Server will calculate the diameter of the spindle cutting tool using data stored in the surface ID you provide. The robot will write the result to the register you provide.

Syntax

```
RSWSPNDLSIZE(SurfaceID, DiameterReg)
```

Argument	Definition
SurfaceID	Integer. The surface ID which contains the spindle cutting tool diameter data.
DiameterReg	Integer. The register index where the robot should write the calculated diameter in millimetres.

Example:

```
: CALL RSWSPNDLSIZE(31,21); //Server calculates a diameter using data  
                               stored in surface ID 31.  
  
                               //Robot writes the result to R[21].
```

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 user frame to perform its work.

5.4.2 User frame

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.

Once set, all cutting and probing co-ordinates are referenced relative to the user 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 48). Datums can be used to locate user 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 22. 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).

This page is intentionally left blank.


This page is intentionally left blank.

www.renishaw.com/rcs-support



#renishaw

 +44 (0) 1453 524524

 industrialautomation@renishaw.com

© 2021–2024 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-6852-8026-00-A

Issued: 08.2024