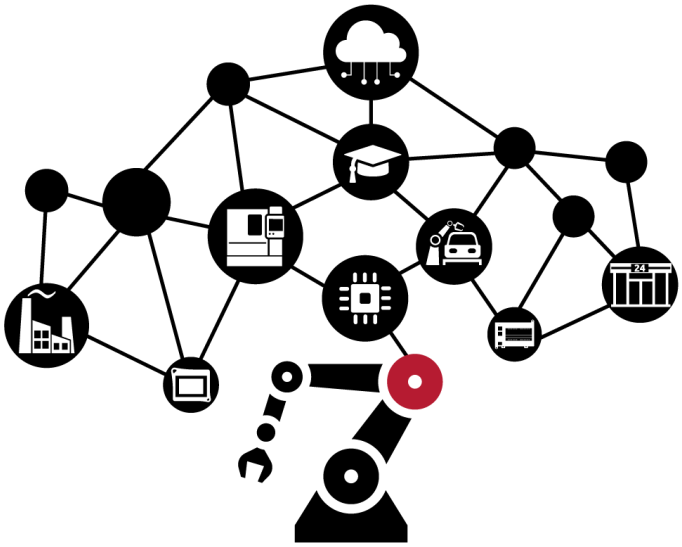
NexMotion Real-time Programming Language (NRPL)

Instructions Manual



Version: 1.1

Date: Oct. 7, 2019

NEXCOBOT Inc.

www.nexcobot.com

Copyright Statement and Disclaimer

The contents contained in this document are the proprietary property of NexCOBOT International Co., Ltd. (NexCOBOT hereafter) and is subject to the protection of intellectual property law (including, but not limited to the Copyright Act). The use of any material in relation to this document without the prior authorization of NexCOBOT is considered infringement. Without the written approval of NexCOBOT in advance, this document or any part of it shall not be photocopied, sold, distributed, modified, published, stored or otherwise used.

To keep this document and its contents correct and complete, NexCOBOT reserves the right to change or revise the document at any time without further notification.

Operating machine or equipment has a certain level of danger. It is the user's responsibility to pay special attention and have safety protection in place before operating any machine or equipment. NexCOBOT shall not be held for any and all direct or indirect damage or loss to the equipment mentioned in this document due to the use for a purpose other than the intended.

Revision History

Rev.	Description
1.0	First released.
1.1	Typos correct.

Contents

1. Data Type

1.1. Basic Data Type

NexMotion Real-time Programming Language (NRPL) provides the following basic data types:

Type	C/C++ Prototype	Description	Byte	Range
U8_T	unsigned char	Unsigned integer	1	0–255
U16_T	unsigned short	Unsigned integer	2	0–65535
U32_T	unsigned int	Unsigned integer	4	0–4294967295
I8_T	char	Signed integer	1	-128–127
I16_T	short	Signed integer	2	-32768–32767
I32_T	int	Signed integer	4	-2147483648–2147483647
F64_T	double	Double-precision float point	8	IEEE-754, it accommodates 15 digits after the decimal point.

1.2. Inner Structure Data Type

1.2.1. Structure: AP_T

AP_T is the description of the position of each axis of the robot group with the axis coordinate system.

- Structure definition

```
struct AP_T
{
    F64_T A0;
    F64_T A1;
    F64_T A2;
    F64_T A3;
    F64_T A4;
    F64_T A5;
    F64_T A6;
    F64_T A7;
};
```

- Member: F64_T A0~A7 describes the position of each axis of GROUP.

Structure: CP_T

CP_T is the description the position of tool endpoint of the robot group with the Cartesian coordinate system.

1.2.2. The tool endpoint of a coordinate system has maximum six degrees of freedom and the movement parameters are notated using X, Y, Z, A, B, and C.

Positions: X, Y, and Z follow the right-hand rule of the Cartesian coordinate system.

Orientation: A, B, and C are Euler angle of intrinsic Z-Y-X representation, sequentially rotate Z, Y, and X axes.

- Structure definition

```
struct CP_T
{
    F64_T X;
    F64_T Y;
    F64_T Z;
    F64_T A;
    F64_T B;
    F64_T C;
};
```

- Member

1.2.3.

Member	Cartesian Coordinate System Position	Unit of Length
F64_T X	X-axis	mm
F64_T Y	Y-axis	mm
F64_T Z	Z-axis	mm
Member	**Cartesian Coordinate System Rotation**	**Unit of Degree**
F64_T A	Z-axis	mm
F64_T B	Y-axis	mm
F64_T C	X-axis	mm

Structure: CFG_T

CFG_T is the description of the relationship between the tool endpoint position of the robot group and each axis coordinate.

You can use Cartesian coordinate (CP_T) to describe the position of a robot when you run motion commands such as PTP, LIN, etc. However, some Cartesian coordinates can be generated by the controller (reverse kinematics) to calculate more than one solution. This means that there can be several robot poses at the same endpoint position. In order to correctly select a specific pose from multiple sets of posture, additional information (CFG_T) is used to record the angular range of each axis of the robot.

CFG_T definition methods:

1. Angular range of axis
2. Types of robot posture

Also, the different mechanism configuration of a robot has a different definition of CFG_T. Refer to the table below for details.

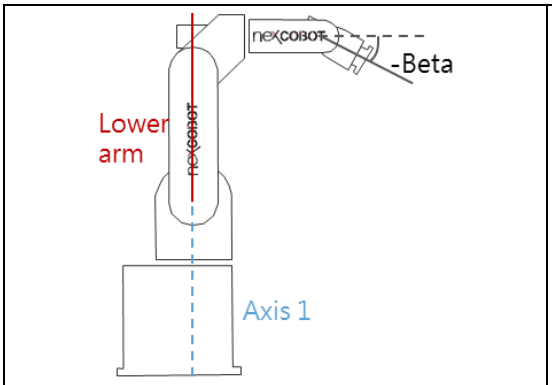
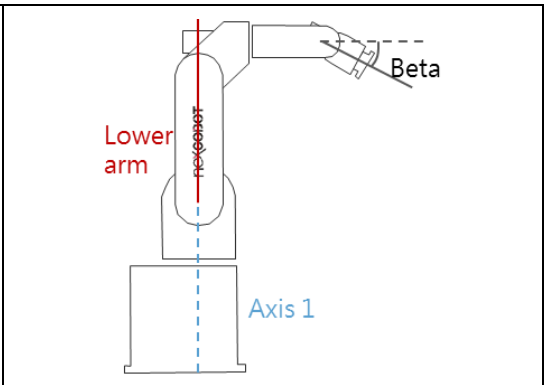
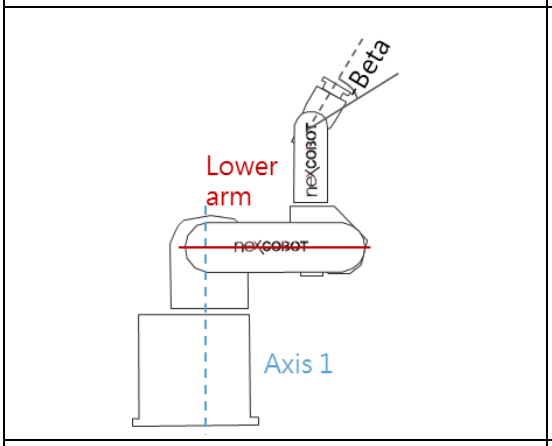
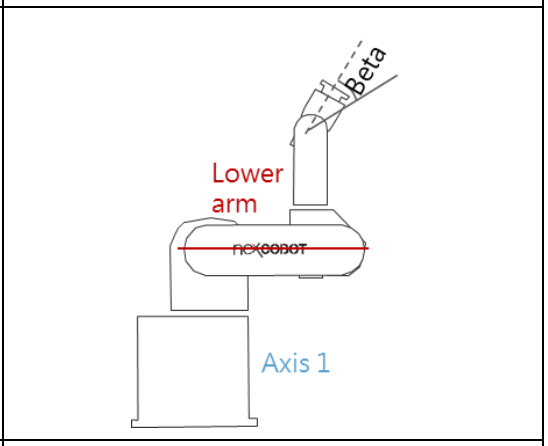
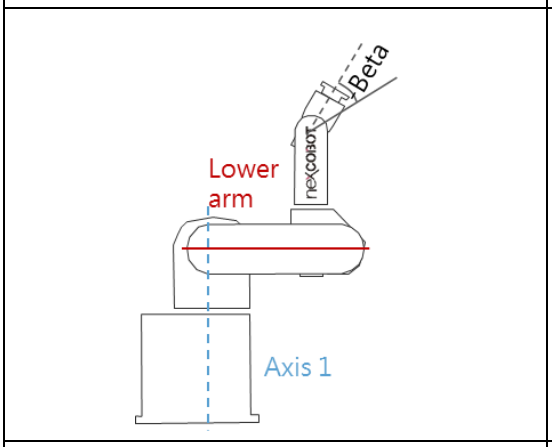
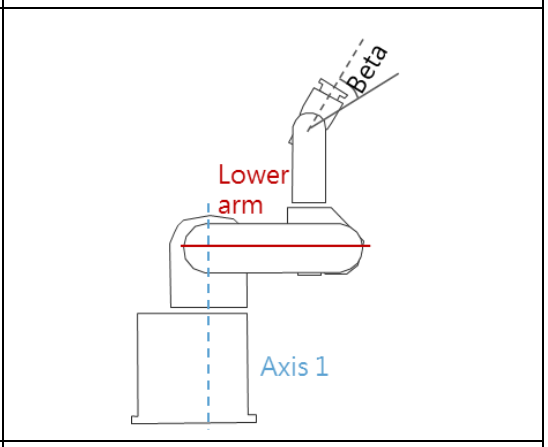
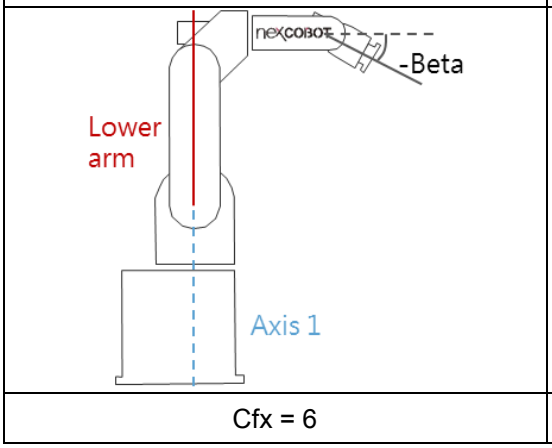
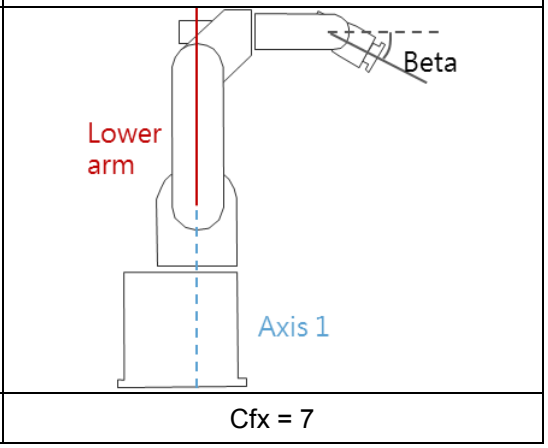
Mechanism Type	C1	C2	C3	C4
AR6	Joint 1 (Axis #0)	Joint 4 (Axis #3)	Joint 6 (Axis #5)	Posture type
SCARA	Joint 1 (Axis #0)	Joint 4 (Axis #3)	None	Joint 2 (Axis #1)
DELTA	None	Joint 4 (Axis #3)	None	None

If CFG_T:C1 is set to 10, it means that the CFG_T setting is not referenced, and the controller automatically selects an optimal posture when calculating.

- Structure definition

```
struct CFG_T
{
    I32_T C1;
    I32_T C2;
    I32_T C3;
    I32_T C4;
};
```

- Member
  - I32_T C1: Parameter of axis angular range. If set to 10, the CFG_T setting is ignored.
  - I32_T C2: Parameter of axis angular range
  - I32_T C3: Parameter of axis angular range
  - I32_T C4: Parameter of axis angular range or posture type

	
Cfx = 0	Cfx = 1
	
Cfx = 2	Cfx = 3
	
Cfx = 4	Cfx = 5
	
Cfx = 6	Cfx = 7

The parameters of **axis angular range** are defined as follows:

Value	Meaning
0	0 to 90 degree
1	90 to 180 degree
2	180 to 270 degree
3	270 to 360 degree
-1	0 to -90 degree
-2	-90 to -180degree
-3	-180 to -270degree

Posture Type

Value	Wrist center relative to axis 1	Wrist center relative to lower arm	Axis 5 angle
0	In front of	In front of	Positive
1	In front of	In front of	Negative
2	In front of	Behind	Positive
3	In front of	Behind	Negative
4	Behind	In front of	Positive
5	Behind	In front of	Negative
6	Behind	Behind	Positive
7	Behind	Behind	Negative

1.2.4.

Structure: EP_T

EP_T describes the external axis coordinates associated with the robot group.

- Structure definition

```
struct EP_T
{
    F64_T E0;
    F64_T E1;
    F64_T E2;
    F64_T E3;
    F64_T E4;
    F64_T E5;
};
```

- Member: F64_T E0~E5 describes the position of external axes.

Structure: RAP_T

RAP_T describes the position of each axis of the robot group and its associated external axis.

1.2.5.

- Structure definition

```
struct RAP_T
{
AP_T    AP;
EP_T    EP;
};
```

- Member
  - **AP_T AP** is the position of each axis of group. Please refer to the structure definition of AP_T.
  - **EP_T EP** is the position of external axis. Please refer to the structure definition of EP_T.

1.2.6. Structure: RCP_T

RCP_T describes the location of the Cartesian coordinates of the robot group and its associated external axis coordinates.

- Structure definition

```
struct RCP_T
{
    CP_T    CP;
    CFG_T   CFG;
    EP_T    EP;
};
```

- Member
  - **CP_T CP** is the endpoint position of Cartesian coordinate. Please refer to the structure definition of CP_T.
  - **CFG_T CFG** is the setting of relationship between Cartesian coordinates and axis coordinates. Please refer to the structure definition of CFG_T.
  - **EP_T EP** is the position of external axis. Please refer to the structure definition of EP_T.

Structure: RP_T

RP_T describes the composite position of the robot group, including the axis coordinates and endpoint of Cartesian coordinates, which are generally used in the point teaching.

1.2.7.

- Structure definition

```
struct RCP_T
{
    CP_T   CP;
    CFG_T CFG;
    EP_T   EP;
    AP_T   AP;
};
```

- Member
  - **CP_T CP** is the endpoint position of Cartesian coordinate. Please refer to the structure definition of CP_T.
  - **CFG_T CFG** is the setting of relationship between Cartesian coordinates and axis coordinates. Please refer to the structure definition of CFG_T.
  - **EP_T EP** is the position of external axis. Please refer to the structure definition of EP_T.
  - **AP_T AP** is the position of each axis of group. Please refer to the structure definition of AP_T.

2. Built-in Functions

2.1. System Functions

Digital I/O control

2.1.1.1. DOUT: Digital Output Setting

Usage

2.1.1 DOUT() is used to control the digital output (ON or OFF).

Example 1

```
PTP( P1 );
DOUT( 0, 1 ); //Ch#0 set ON
PTP( P2 );
DOUT( 0, 0 ); //Ch#0 set OFF
```

1. Move to P1 in PTP mode
2. Set DO Ch#0 to ON
3. Move to P2 in PTP mode
4. Set DO Ch#0 to OFF

DOUT() can also be used to set up the delay time (DT).

Example 2

```
PTP( P1 );
DOUT( 0, 1, DT=1000 ); //Delay 1000ms, Ch#0 set ON
PTP( P2 );
```

1. Move to P1 in PTP mode.
2. Set Ch#0 ON after a 1000 ms (milliseconds) delay.
3. Move to P2 in PTP mode.

After the DOUT() output setting is established, you can set the waiting time (WT) separately.

Example 3

```
PTP( P1 );
DOUT( 0, 1, WT=1000 );
PTP( P2 );
```

1. Move to P1 in PTP mode.
2. After setting the Ch#0 to ON, wait for 1000 ms and return.
3. Move to P2 in PTP mode.

Example 4

```
PTP( P1 );
DOUT( 0, 1, WT=1000, REV );
PTP( P2 );
```

1. Move to P1 in PTP mode.
2. After setting the Ch#0 to ON, wait for 1000 ms, set Ch#0 to OFF(REV) and return.
3. Move to P2 in PTP mode.

Example 5

```
PTP( P1 );
WAIT( 1000 ); //Wait 1000 ms
DOUT( 0, 1 ); //Set DO#0 ON
WAIT( 1000 ); //Wait 1000 ms
DOUT( 0, 0 ); //Set DO#0 OFF
PTP( P2 );
```

Example 5 can be simplified as below:

```
PTP( P1 );
DOUT( 0, 1, DT=1000, WT=1000, REV );
PTP( P2 );
```

Command

DOUT (CH, VAL, [DT], [WT] , [REV]);

Command	Notes
CH	• Specify the digital output channel number • Data type: I32_T • CH number: from 0 to the actual maximum number of the system
VAL	• Set to ON or OFF • Data type: I32_T • VAL=0 (OFF)，VAL=1 (ON)
[DT]	• Delay time. This parameter can be ignored. • Data type: I32_T • Millisecond. After command is executed, it will delay the specified time and then set DO. • If this parameter is set to less than 0 or not set, it means no delay.
[WT]	• Wait time. This parameter can be ignored. • Data type: I32_T • Millisecond • After command sets the DO output, a wait time can be set. It is equivalent to adding a WAIT() after this command. • If this parameter is set to less than 0 or not set, it means no delay.
[REV]	• The flag of reverse output signal. This parameter can be ignored. • Data type: Flag constant • If the wait time is set, the REV flag can be set separately. After wait time, the command will reverse the output signal.

Return Value

No return value.

Related Information

2.1.1.2. GDOUT: Get Digital Output

Usage

GDOUT() is used to read the status of the digital output.

Example

```
I32_T DO_0;

DO_0 = GDOUT( 0 );//Get DO#0 state


```

Command

I32_T GDOUT (CH);

Command	Notes
CH	• Specify the digital output channel number. • Data type: I32_T • CH number, from 0 to the actual maximum number of the system.

Return Value

Data type: I32_T

- 0: DOUT OFF
- 1: DOUT ON

Related Information

2.1.1.3. GDIN: Get Digital Input

Usage

GDIN() is used to get the status of digital input.

Example

```
I32_T DI_0;

DI_0 = GDIN( 0 );//Get DI#0 state


```

Command

I32_T GDIN (CH);

Command	Notes
CH	• Specify the digital output channel number. • Data type: I32_T • CH number, from 0 to the actual maximum number of the system.

Return Value

Data type: I32_T

- 0: DOUT OFF
- 1: DOUT ON

Related Information

2.1.1.4. WDIN: Wait For The Digit Input

Usage

WDIN() is used to wait for the digit input until the status matches the setting.

Example 1

```
1
2   …
3   WDIN( 5, 1 );//Wait DI#5 state turn ON
4   PTP( P0 );
5
```

After waiting for state of DI#5 to be ON, execute PTP(P0).

Also, WDIN() can be used to set the time (millisecond) for timeout.

Example 2

```
1   I32_T isTimeout;
2   …
3   isTimeout =WDIN( 5, 1, T=5000 );//Wait DI#5 state turn ON
4
5   if( isTimeout )
6   { // Wait DI#5 turn ON, timeout
7      PTP( P0 );
8   }else
9   { // DI#5 state turn ON
10     PTP( P1 );
11  }
```

Command

I32_T WDIN (CH, VAL, [T]);

Command	Notes
CH	• Specify the digital output channel number • Data type: I32_T • CH number: from 0 to the actual maximum number of the system
VAL	• Waiting for state of digital input • Data type: I32_T • VAL=0 (OFF)，VAL=1 (ON)
[T]	• Set time out time. This parameter can be ignored. • Data type: I32_T • Millisecond. . If you don’t specify the T or T is less than 0: wait for state to be matched.

Command	Notes
	• T equals to 0: check state of DI and return immediately. • T greater than 0: specify wait time. If the status meets the specified value within the waiting time, it will return immediately. If the waiting time exceeds the specified value, the timeout will be returned.

Return Value

Data type: I32_T

The return indicates whether the waiting is successful.

- 0: Successful
- 1: Timeout

Related Information

2.1.2.1. AOUT: Analog Output Setting

Usage

2.1.2. AOUT() controls the single analog output channel.

Example 1

```
PTP( P1 );
AOUT( 0, 5.0 ); //Set AO#0 to 5V
PTP( P2 );
```

1. Move to P1 in PTP mode.
2. Set AO Ch#0 to 5V.
3. Move to P2 with PTP.

AOUT() can also set the delay time (DT).

Example 2

```
PTP( P1 );
AOUT( 0, 3.3, DT=1000 ); //Delay 1000ms, AO#0 set 3.3V
PTP( P2 );
```

1. Move to P1 with PTP.
2. After delay 1000ms, set AO#0 to 3.3 Volt.
3. Move to P2 with PTP.

After the AOUT() output setting is established, you can set the waiting time(WT) separately.

Example 3

```
PTP( P1 );
AOUT( 0, -2.5, WT=1000 );
PTP( P2 );
```

1. Move to P1 with PTP.
2. After setting AO#0 to -2.5 Volt, delay 1000ms and return.
3. Move to P2 with PTP.

Example 4

```
PTP( P1 );
WAIT( 1000 );   //Wait 1000 ms
AOUT( 0, 5.0 ); //Set AO#0 to 5V
WAIT( 1000 );   //Wait 1000 ms
AOUT( 0, 0.0 ); //Set AO#0 to 0V
PTP( P2 );
```

Example 4 can be simplified as below:

```
PTP( P1 );
AOUT( 0, 5.0, DT=1000, WT=1000 );
AOUT( 0, 0.0 );
PTP( P2 );
```

Command

AOUT (CH, VAL, [DT], [WT]);

Command	Notes
CH	• Specify the digital output channel number • Data type: I32_T • CH number: from 0 to the actual maximum number of the system
VAL	• Set analog output value • Data type: F64_T • The analog output value range definition should be defined in accordance with the AIO device of the actual system. For example, the unit is Volt or mV.
[DT]	• Delay time. This parameter can be ignored. • Data type: I32_T • Millisecond • After the command is executed, it will delay the specified time and then set the AO. • If this parameter is set to less than 0 or not set, it means no delay.
[WT]	• Wait time. This parameter can be ignored. • Data type: I32_T • Millisecond • After the command sets the AO output, a wait time can be set. It is equivalent to adding a WAIT() after this command. • If this parameter is set to less than 0 or not set, it means no delay.

Return Value

No return value.

Related Information

2.1.2.2. GAIN: Get Analog Input Signal

Usage

GAIN() gets the analog input signal.

Example

```
1  F64_T AI_0;
2
3  AI_0 = GAIN( 0 );
4
5
```

1. Declare variable: F64_T AI_0.
3. Get the analog input channel #0 value and store it in variable AI_0.

Command

F64_T GAIN (CH);

Command	Notes
CH	• Specify the analog input channel number. • Data type: I32_T • CH number: from 0 to the actual maximum number of the system.

Return Value

Data type: F64_T

The analog input value range definition should be defined in accordance with the AIO device of the actual system. For example, the unit is Volt or mV.

Related Information

2.1.2.3. GAOUT: Get Analog Output Signal

Usage

GAOUT() gets the analog output signal.

Example

```
1  F64_T AO_0;
2
3  AO_0 = GAOUT( 0 );
4
5
```

1: Declare variable: F64_T AO_0

3: Get value of analog output channel #0 and store it in AO_0.

Command

F64_T GAOUT (CH);

Command	Notes
CH	• Specify the analog output channel number. • Data type: I32_T • CH number: from 0 to the actual maximum number of the system.

Return Value

Data type: F64_T

The analog output value range definition should be defined in accordance with the AIO device of the actual system. For example, the unit is Volt or mV.

Related Information

2.1.2.4. WAIN: Waiting for the Analog Input Value Meets the Set Range

Usage

WAIN() waits for the analog input value to enter the set range.

Example 1

```
1
2   …
3   WAIN( 0, GT=5 );
4   PTP( P0 );
5
```

3: Wait until the AI#0 voltage (or current) value is greater than or equal to 5.

4: Execute PTP(P0).

WAIN() can also set the time (millisecond) for timeout:

Example 2

```
1   I32_T isTimeout;
2   …
3   isTimeout =WAIN( 0, GT=5, T=5000 );
4
5   if( isTimeout )
6   {
7      PTP( P0 );
8   }else
9   {
10     PTP( P1 );
11  }
```

3: Wait until the AI#0 voltage (or current) value is greater than or equal to 5 or timeout with 5000 ms.

5: If timeout occurs, execute 7: PTP(P0).

8: If timeout doesn't happen, execute 10" PTP(P1).

Command

I32_T WAIN (CH, [GT], [LT] [T]);

Command	Notes
CH	• Specify the analog input channel number • Data type: I32_T • CH number: from 0 to the actual maximum number of the system
[GT]	• Set the comparison range (greater or equal to). This parameter

Command	Notes
	can be ignored. • Data type: F64_T • Set value of GT. Example: GT=0, when the AI value >= 0, condition is triggered. GT=-5, when the AI value >= -5, condition is triggered. If [LT] is set, the [GT] and [LT] conditions must be satisfied at the same time.
[LT]	• Set the comparison range (less or equal to). This parameter can be ignored. • Data type: F64_T • Set value of LT. Example: LT=0, when the AI value >= 0, condition is triggered. LT=-5, when the AI value >= -5, condition is triggered. • If [GT] is set, the [GT] and [LT] conditions must be satisfied at the same time.
[T]	• Set timeout time. This parameter can be ignored. • Data type: I32_T • Millisecond. If you don't specify the T or T is less than 0, wait for the state to be matched. • T equals to 0: Check the AI condition and return. • T greater than 0: specify the wait time. If the status meets the specified value within the waiting time, it will return immediately. If the waiting time exceeds the specified value, the timeout will be returned.

Return Type

Data type: I32_T

The return indicates whether the waiting is successful.

- 0: Successful. The AI value is within the setting range.
- 1: Timeout. The AI value is not in the setting range within the waiting time.

Related Information

2.1.3.1. WaitEvent: Waiting Event

Usage

2.1.3. WaitEvent() is used to wait for a system event to be triggered.

Example 1

```
1
2  WaitEvent( 0, -1 );
3
4
5
```

2. Wait until event 0 is triggered.

Example 2

```
1  I32_T isTimeout;
2
3  isTimeout = WaitEvent( 1, 1000 );
4
5  if( isTimeout )
6  {
7
8  }
9
```

1. Declare a variable of I32_T.
3. Wait for 1 second and report if event 1 is triggered.
5-8. Determine what behavior to perform based on the value of 'isTimeout'.

Command

I32_T WaitEvent (EventID, TimeoutMs);

Command	Notes
EventID	• Specify the waiting event number. • Data type: I32_T • 8 Built-in events (numbers 0 to 7).
TimeoutMs	• Set timeout time, milliseconds. • Data type: I32_T • Greater than 0: waiting time. • Equal to 0: check event whether is triggered and return. • Less than 0: wait until the event is triggered.

Return Value

Data type: I32_T

- 0: Successful
- 1: Timeout

Related Information

SetEvent()

ResetEvent()

2.1.3.2. SetEvent: Set Event to Signaled State

Usage

SetEvent() sets the system event to the signaled state.

Example 1: two task synchronization.

- Task#0 controls Group#0, Task#1controls Group#1.
- After the first robot (Group#0) is in place, control the second robot (Group#1).

```
1  // Task#0
2
3  PTP( P0 );
4  SetEvent( 1 );
5
```

1. Task#0.
3. Move Group#0 to P0.
4. Trigger Event#1.

Example 2

```
1  // Task#1
2  WaitEvent( 1, -1 );
3  ResetEvent( 1 );
4  WAIT( 1000 );
5  PTP( P1 );
```

1. Task#1.
2. Wait until Event#1 is triggered.
3. Reset Event#1.
4. Wait a second.
5. Move Group#1 to P1.

Command

SetEvent (EventID);

Command	Notes
EventID	• Specify the setting event number. • Data type: I32_T • Eight built-in events, they are numbered from 0 to 7.

Return Value

No return value.

Related Information

WaitEvent()

ResetEvent()

2.1.3.3. ResetEvent: Reset Event to Un-signaled State

Usage

ResetEvent() resets the system event to the un-signaled state.

Example 1: two task synchronization

- Execute Task#0 controls Group#0, Task#1 controls Group#1.
- After the first robot (Group#0) is in place, then control the second robot (Group#1).

```
1  // Task#0
2
3  PTP( P0 );
4  SetEvent( 1 );
5
```

1. Task#0.
3. Move Group#0 to P0.
4. Trigger Event#1.

```
1  // Task#1
2  WaitEvent( 1, -1 );
3  ResetEvent( 1 );
4  WAIT( 1000 );
5  PTP( P1 );
```

1. Execute Task#1.
2. Wait until Event#1 is triggered.
3. Reset Event#1.
4. Wait one second.
5. Move Group#1 to P1.

Command

ResetEvent (EventID);

Command	Notes
CH	• Specify the setting event number. • Data type: I32_T • Eight built-in events, they are numbered from 0 to 7.

Return Value

No return value.

Related Information

WaitEvent()

ResetEvent()

2.1.4.1. cos

Usage

2.1.4. cos() is used to calculate cosine.

Example

```
1  F64_T value;
2
3  value = cos( 60.0 * 3.1415926 / 180.0 );
4  printf( “The cosine of %f degree is %f\n”, 60.0, value );
5
```

Output

```
The cosine of 60.0 degree is 0.50000
```

Command

F64_T cos (X);

Command	Notes
X	• To calculate the angle of cosine, expressed in radian. • Data type: F64_T • 1 rad = 180 / PI degrees

Return Value

Data type: F64_T

Return the cosine value of X.

Related Information

sin

tan

2.1.4.2. sin

Usage

sin() is used to calculate sine.

Example

```
1  F64_T value;
2
3  value = sin( 60.0 * 3.1415926 / 180.0 );
4  printf( “The sine of %f degree is %f\n”, 60.0, value );
5
```

Output

```
The sine of 60.0 degree is 0.866025
```

Command

F64_T sin (X);

Command	Notes
X	• To calculate the angle of cosine, expressed in radian. • Data type: F64_T • 1 rad = 180 / PI degrees

Return Value

Date type: F64_T

Return sine value of X.

Related Information

cos

tan

2.1.4.3. tan

Usage

tan() is used to calculate tangent.

Example

```
1  F64_T value;
2
3  value = tan( 60.0 * 3.1415926 / 180.0 );
4  printf( “The tangent of %f degree is %f\n”, 60.0, value );
5
```

Output

```
The tangent of 60.0 degree is 1.732051
```

Command

F64_T tan (X);

Command	Notes
CH	• To calculate the angle of cosine, expressed in radian. • Data type: F64_T • 1 rad = 180 / PI degrees

Return Value

Data type: F64_T

Return tangent value of X.

Related Information

cos

sin

2.1.4.4. acos

Usage

acos() is used to calculate arc cosine.

Example

```
F64_T value;

value = acos( 0.5 );
printf( “The arc cosine of %f is %f radian\n”, 0.5, value );

```

Output

```
The arc cosine of 0.5 is 1.047198 radian
```

Command

F64_T acos (X);

Command	Notes
CH	• To calculate the principal value of arc cosine. • Data type: F64_T • 1 rad = 180 / PI degrees

Return Value

Data type: F64_T

Return arc cosine value of X.

Return Value

asin

atan

2.1.4.5. asin

Usage

asin() is used to calculate arc sine.

Example

```
F64_T value;

value = asin( 0.866025 );
printf( “The arc sine of %f is %f radian\n”, 0.866025, value );

```

Output

```
The arc sine of 0.866025 is 1.047198 radian
```

Command

F64_T asin (X);

Command	Notes
CH	• To calculate the principal value of arc sine. • Data type: F64_T • 1 rad = 180 / PI degrees

Return Value

Data type: F64_T

Return the arc sine value of X.

Related Information

acos

atan

2.1.4.6. atan

Usage

atan() is used to calculate arc tangent.

Example

```
F64_T value;

value = atan( 1.732051 );
printf( “The arc tangent of %f is %f radian\n”, 1.732051, value );

```

Output

```
The arc sine of 1.732051 is 1.047198 radian
```

Command

F64_T atan (X);

Command	Notes
CH	• To calculate the principal value of arc tangent. • Data type: F64_T • 1 rad = 180 / PI degrees

Return Value

Data type: F64_T

Return the arc tangent value of X.

Related Information

acos

asin

atan2

2.1.4.7. atan2

Usage

atan2() is used to calculate arc tangent.

Example

```
1  F64_T value;
2
3  value = atan2( 10.0, -10.0 );
4  printf( “The arc tangent for ( x = %f, y = %f ) is %f radian\n”, -10.0,
5  10.0, value );
```

Output

```
The arc tangent for ( x = -10.000000, y = 10.000000 ) is 2.356194 radian
```

Command

F64_T atan2 (X, Y);

Command	Notes
X	• To calculate the **adjacent side** value of arc tangent. The unit is self-defined. • Data type: F64_T
Y	• To calculate the **opposite side** value of arc tangent. The unit is self-defined. • Data type: F64_T

Return Value

Data type: F64_T

Return the arc tangent value of X and Y.

Related Information

acos

asin

atan

2.1.4.8. cosh

Usage

cosh() is used to calculate hyperbolic cosine.

Example

```
1  F64_T value;
2
3  value = cosh( 1.0 );
4  printf( “The hyperbolic cosine of %f is %f\n”, 1.0, value );
5
```

Output

```
The hyperbolic cosine of 1.000000 is 1.543081
```

Command

F64_T cosh (X);

Command	Notes
X	• To calculate the hyperbolic angle of hyperbolic cosine. • Data type: F64_T

Return Value

Data type: F64_T

Return the hyperbolic cosine value of X.

Related Information

sinh

tanh

2.1.4.9. sinh

Usage

sinh() is used to calculate hyperbolic sine.

Example

```
1  F64_T value;
2
3  value = sinh( 1.0 );
4  printf( “The hyperbolic sine of %f is %f\n”, 1.0, value );
5
```

Output

```
The hyperbolic sine of 1.000000 is 1.175201
```

Command

F64_T sinh (X);

Command	Notes
X	• To calculate the hyperbolic angle of hyperbolic sine. • Data type: F64_T

Return Value

Data type: F64_T

Return the hyperbolic sine value of X.

Related Information

cosh

tanh

2.1.4.10. tanh

Usage

tanh() is used to calculate hyperbolic tangent.

Example

```
1  F64_T value;
2
3  value = tanh( 1.0 );
4  printf( “The hyperbolic tangent of %f is %f\n”, 1.0, value );
5
```

Output

```
The hyperbolic tangent of 1.000000 is 0.761594
```

Command

F64_T tanh (X);

Command	Notes
X	• To calculate the hyperbolic angle of hyperbolic tangent. • Data type: F64_T

Return Value

Data type: F64_T

Return the hyperbolic tangent value of X.

Related Information

cosh

sinh

2.1.4.11. exp

Usage

exp() is used to calculate exponent based on nature logarithm e.

Example

```
1  F64_T value;
2
3  value = exp( 5.0 );
4  printf( “The exponential value of %f is %f\n”, 5.0, value );
5
```

Output

```
The exponential value of 5.000000 is 148.413159
```

Command

F64_T exp (X);

Command	Notes
X	• Value of the exponent • Data type: F64_T

Return Value

Data type: F64_T

Return the exponential value of X which is based on nature logarithm e.

Related Information

log

pow

2.1.4.12. ldexp

Usage

ldexp() is used to calculate the exponential value based on 2 and multiplying X.

Example

```
1  F64_T value;
2
3  value = ldexp( 2.5, 4 );
4  printf( “%f * 2^%d = %f\n”, 2.5, 4, value );
5
```

Output

```
2.500000 * 2^4 = 40.000000
```

Command

F64_T ldexp (X, Exp);

Command	Notes
X	• Value of multiplier • Data type: F64_T
Y	• Value of exponent • Data type: I32_T

Return Value

Data type: F64_T

Return the exponential value of X and Exp.

Related Information

log

pow

2.1.4.13. log

Usage

log() is used to calculate the logarithm value based on the nature logarithm “e”.

Example

```
1  F64_T value;
2
3  value = log( 2.5 );
4  printf( “log( %f )= %f\n”, 2.5, value );
5
```

Output

```
log( 2.500000 )= 0.916291
```

Command

F64_T log (X);

Command	Notes
X	• To calculate the natural argument of the natural logarithm, this argument value has to be greater than zero. • Data type: F64_T

Return Value

Data type: F64_T

Return nature logarithm value of X.

Related Information

log10

exp

pow

2.1.4.14. log10

Usage

log10() is used to calculate the logarithm value that is based on 10.

Example

```
1  F64_T value;
2
3  value = log10( 2.5 );
4  printf( “log10( %f )= %f\n”, 2.5, value );
5
```

Output

```
log10( 2.500000 )= 0.397940
```

Command

F64_T log10 (X);

Command	Notes
X	• To calculate the argument of the logarithm that is based on 10, this argument value has to be greater than zero. • Data type: F64_T

Return Value

Data type: F64_T

Return common logarithm value of X.

Related Information

log

exp

pow

2.1.4.15. pow

Usage

pow() is used to calculate the power exponent.

Example

```
1  F64_T value;
2
3  value = pow( 2.5, 5.5 );
4  printf( “pow( %f, %f )= %f\n”, 2.5, 5.5, value );
5
```

Output

```
log10( 2.500000, 5.500000 )= 154.408089
```

Command

F64_T pow (X, Y);

Command	Notes
X	• Base value of power • Data type: F64_T
Y	• Exponential value of power • Data type: F64_T

Return Value

Data type: F64_T

Return pow result of X and Y.

Related Information

log

exp

sqrt

2.1.4.16. sqrt

Usage

sqrt() is used to calculate square root.

Example

```
1  F64_T value;
2
3  value = sqrt( 2.5 );
4  printf( “sqrt( %f )= %f\n”, 2.5, value );
5
```

Output

```
sqrt( 2.500000 )= 1.581139
```

Command

F64_T sqrt (X);

Command	Notes
X	• To calculate the argument of the square root that is based on 10, this argument value has to be greater than zero. • Data type: F64_T

Return Value

Data type: F64_T

Return square root of X.

Related Information

log

pow

2.1.4.17. ceil

Usage

ceil() is used to calculate the smallest integer which is not less than the input argument.

Example

```
1  F64_T value;
2
3  value = ceil( -2.5 );
4  printf( “ceil of %.1f = %.1f\n”, -2.5, value );
5
```

Output

```
ceil of -2.5 = -2.0
```

Command

F64_T ceil (X);

Command	Notes
X	• Calculate the nearest value that is greater than the argument. • Data type: F64_T

Return Value

Data type: F64_T

The smallest integral value that is not less than X.

Related Information

floor

fabs

2.1.4.18. floor

Usage

floor() is used to calculate the largest integer which is not greater than the input argument.

Example

```
1  F64_T value;
2
3  value = floor( -2.5 );
4  printf( “floor of %.1f = %.1f\n”, -2.5, value );
5
```

Output

```
floor of -2.5 = -3.0
```

Command

F64_T floor (X);

Command	Notes
X	• Calculate the nearest value that is less than the argument. • Data type: F64_T

Return Value

Data type: F64_T

The largest integral value that is not greater than X.

Related Information

ceil

fabs

2.1.4.19. fmod

Usage

fmod() is used to calculate the residual value of numerator or denominator.

Example

```
1  F64_T value;
2
3  value = fmod( 10.5, 3.2 );
4  printf( “fmod of %.1f / %.1f is %.1f\n”, 10.5, 3.2, value );
5
```

Output

```
fmod of 10.5 / 3.2 is 0.9
```

Command

F64_T fmod (X, Y);

Command	Notes
X	• Numerator • Data type: F64_T
Y	• Denominator • Data type: F64_T

Return Value

Data type: F64_T

Return the fmod result of the X and Y residual value of numerator or denominator.

Related Information

fabs

2.1.4.20. abs

Usage

abs() is used to calculate the absolute value of the integer.

Example

```
1  I32_T value;
2
3  value = abs( -10 );
4  printf( “abs of %d is %d\n”, -10, value );
5
```

Output

```
abs of -10 is 10
```

Command

F64_T abs (X);

Command	Notes
X	• To calculate the argument of the absolute value, this argument must be an integer. • Data type: I32_T

Return Value

Data type: I32_T

Return the absolute value of X.

Related Information

fabs

2.1.4.21. fabs

Usage

fabs() is used to calculate the absolute value of the float-point.

Example

```
1  F64_T value;
2
3  value = fabs( -10.5 );
4  printf( “fabs of %.1f is %.1f\n”, -10.5, value );
5
```

Output

```
fmod of -10.5 is 10.5
```

Command

F64_T fabs (X);

Command	Notes
X	• To calculate the argument of the absolute value, this argument must be an integer. • Data type: F64_T

Return Value

Data type: F64_T

Return the fabs result of X.

Related Information

abs

2.1.4.22. deg2rad

Usage

deg2rad() is used to convert degree to radian.

Example

```
1  F64_T value;
2
3  value = deg2rad( 30.0 );
4  printf( “The degree of %f = %f radian\n”, 30.0, value );
5
```

Output

```
The degree of 30.000000 = 0.523599 radian
```

Command

F64_T deg2rad (X);

Command	Notes
X	• Calculates the degree of deg2rad in degrees. • Data type: F64_T • 1 rad = 180/ PI degree

Return Value

Data type: F64_T

Return the deg2rad result of X.

Related Information

rad2deg

2.1.4.23. rad2deg

Usage

rad2deg() is used to convert radian to degree.

Example

```
1  F64_T value;
2
3  value = rad2deg( 1.0 );
4  printf( “The radian of %f = %f degree\n”, 1.0, value );
5
```

Output

```
The radian of 1.000000 = 57.295780 degree
```

Command

F64_T rad2deg (X);

Command	Notes
X	• Calculates the radian of deg2rad in radians. • Data type: F64_T • 1 rad = 180/ PI degree

Return Value

Data type: F64_T

Return the degree value of X.

Related Information

deg2rad

2.1.4.24. sign

Usage

sign() is used to calculate the input argument as +1, -1 or 0.

Example

```
1  F64_T value;
2
3  value = sign( -3.5 );
4  printf( “The sign of %f is %f\n”, -3.5, value );
5
```

Output

```
The sign of -3.500000 is -1
```

Command

F64_T sign (X);

Command	Notes
X	• Calculates the augment of sign • Data type: F64_T

Return Value

Data type: F64_T

Return the sign value of X.

Related Information

2.1.5.1. printf: Message Output

Usage

2.1.5printf() outputs the formatted data to the system message window of the human machine interface. The total number of output characters cannot exceed 127.

Example

```
1
2  printf( “Hello World” );
3  printf( “Value=%d”, 32 );
4  printf( “Double value=%f”, 1.23 );
5
```

2. Output: Hello World
3. Output: Value=32
4. Output: Double value=1.23

Command

printf (Format,…);

Command	Notes
Format	• Text message to be output to the message window. • Data type: C-style string. Use double quotes, e.g. ”Hello World”. • The string can contain %d and %f. ○ %d: output I32_T integer ○ %f: output F64_T float point

Return Value

No return value.

Related Information

2.1.5.2. WAIT: Waiting For A Specific Time

Usage

WAIT() is used to wait for a specific time. The unit is millisecond.

Example

```
1
2  …
3  WAIT( 5000 );
4  PTP( P0 );
5
```

After waiting for 5 seconds, execute PTP(P0).

Command

WAIT (TimeMs)

Command	Notes
Format	• Timeout time setting. • Data type: I32_T • Unit: millisecond • Setting value to 0: return immediately • Setting value greater than 0: specify timeout time

Return Value

No return value.

Related Information

2.2. GROUP and Commands

GROUP is an NRPL built-in object that is automatically generated based on the number of robots currently supported by the system.

GROUP Motion Commands

2.2.1.1. PTP: Perform a Fast Point-to-point Moving of Each Axis of the Group

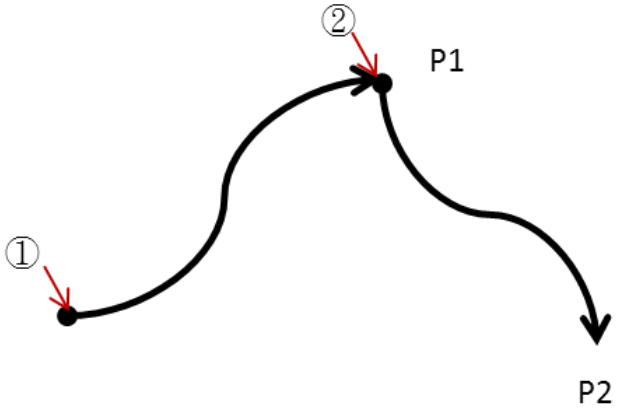
2.2.1. Usage

PTP() is used to perform a fast point-to-point moving of each axis of the robot group. The movement mode is planned with the shortest path of each axis position in the group. Therefore, the trajectory of its tool center point (TCP) may be an arbitrary path.

Example 1

```
1  PTP( P1 );
2  PTP( P2 );
3
```

After moving to P1 with PTP, then move to P2 in PTP mode.



Example 2

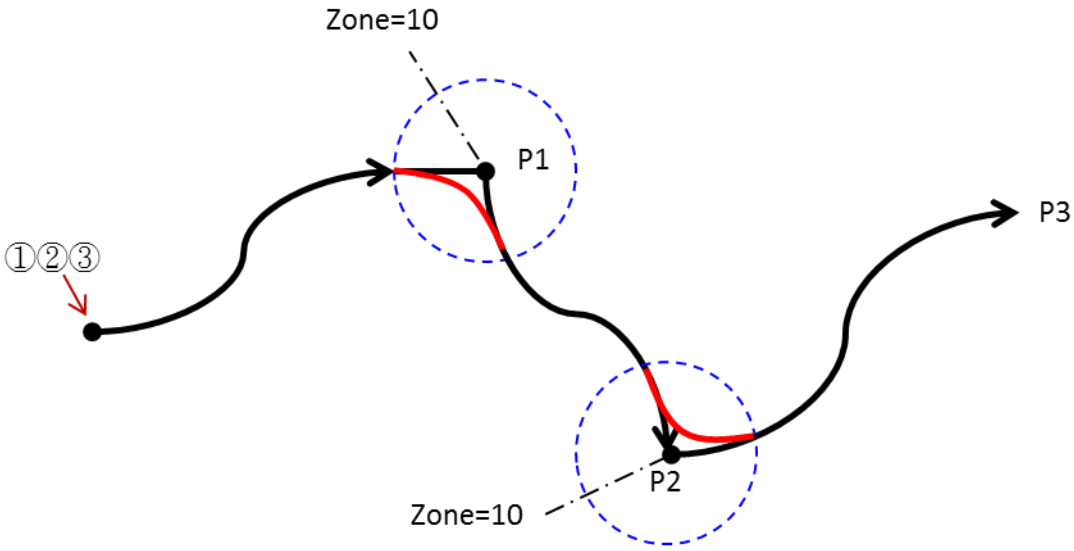
```
1  PTP( P1, VP=50 );
2
3
```

1. Move to P1 in PTP mode at 50% of the maximum speed.

Example 3

```
1  PTP( P1, Z=10 );
2  PTP( P2, Z=10 );
3  PTP( P3 );
4
5
```

1. Move to P1 in PTP mode, when distance of P1 is 10 mm, move to P2 in PTP mode.
2. When distance of P2 is 10 mm, move to P3 in PTP mode.
3. When P3 is in place, the task is done.



Example 4

```
1  PTP( P1, BS=1 );
2  PTP( P2, BS=1 );
3
```

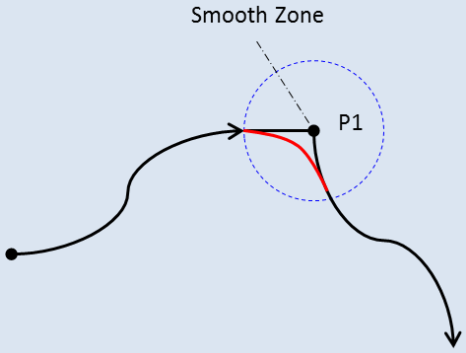
After moving to P1 with PTP, then move to P2 in PTP mode.

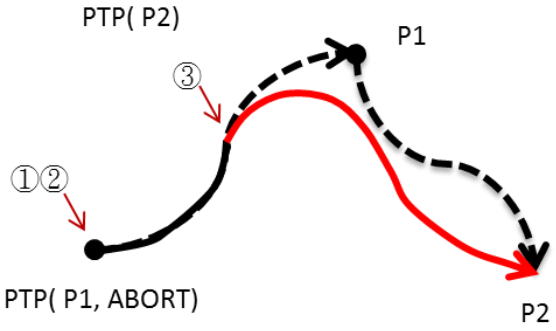
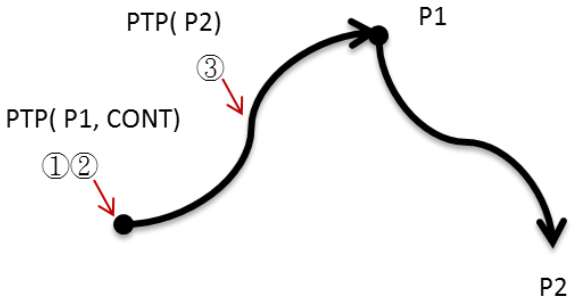
P1 and P2 data type are RP_T. If PTP command specifies the Tool:[TL] or Base:[BS], the point will refer to the position of Cartesian coordinate.

Command

PTP (TargetPos, [VP], [TL], [BS], [Z], [ABORT], [CONT]], [OW])

Command	Notes
TargetPos	• Absolute target position • Data type: RP_T or RCP_T or RAP_T • Set target position of point-to-point movement of the group, can use any of the three data types below: ○ RP_T: Robot position of composite coordinate which contains Cartesian coordinate and axis coordinate. ○ RCP_T: Robot position of Cartesian coordinate ○ RAP_T: Robot position of axis coordinates • If RP_T(composite position) is used, position will refer to axis coordinates. • If the [TL]:tool number or [BS]:base number is specified, the Cartesian coordinate is used as a reference.
[VP]	• Percentage of the maximum speed per axis. The parameter can be ignored. • Data type: F64_T • Speed planning basis for point-to-point motion: parameter of the maximum speed x VP(%) • If this parameter is ignored, the motion planning will be scheduled at the maximum speed. • Refer to Group:AXVEL() on setting the system parameter of each axis.
[TL]	• Specifies Tool number. The parameter can be ignored. • Data type: I32_T • If the Target position uses PCP_T: The robot Cartesian coordinate position will set the tool number. If the target position uses axis coordinate, this parameter will be ignored. • If this parameter is ignored, the tool number is set by the system parameter. • Refer to Tool() and learn how to set the system parameter tool number.
[BS]	• Specifies the Base number. The parameter can be ignored. • Data type: I32_T • If the Target position uses PCP_T: The robot Cartesian coordinate position will set the base number. If the target position uses axis coordinate, this parameter will be ignored. • If this parameter is ignored, the base number is set by the system

Command	Notes
	parameter. • Refer to Base() and learn how to set the system parameter of base number.
[Z]	• Specifies the smooth zone and length unit. The parameter can be ignored. • Data type: F64_T • Specifies a smooth area with a target position as the center, the smooth zone as the radius (as shown below). When a point-to-point (PTP) motion enters the smooth zone, the smooth path is automatically planned to connect with the next motion command, and the speed dose not decrease to zero.  • If the smooth zone is set in a PTP mode, the NRPL program has to execute the next motion command immediately. The behavior is the same as using [CONT]. • If this parameter is ignored, the subsequent command will be executed after the PTP motion is accurately in place.
	• This PTP command can be aborted by subsequent command. The parameter can be ignored. • Data type: flag constant. • If ABORT is declared, it means that the PTP command can be interrupted by subsequent motion command and execute the next motion command immediately. If a zone is specified, the ABORT flag can be ignored. • If CONT flag is declared, the PTP command will not be changed by subsequent motion command. The differences between ABORT and CONT are compared in the following two examples. ○ Example A

Command	Notes
	`` 1 PTP(P1, ABORT); //Can be aborted 2 WAIT(2000); 3 PTP(P2); 4 5 ```  ○ Example B ``` 1 PTP(P1, CONT); //Continue 2 WAIT(2000); 3 PTP(P2); 4 5 ``` 
CONT	• Execute the next command continuously. The parameter can be ignored. • Data type: flag constant. • The PTP command is preset as blocked type, and it waits until the PTP motion command to be accurately in place, then execute the subsequent command. • Under some application conditions, it is desirable to run the subsequent command immediately after the PTP command is issued. Therefore, the PTP command can be declared as a non-blocked type with the CONT flag.
OW	• Overwrites current motion command. The parameter can be ignored. • Data type: flag constant.

Command	Notes
	• Will force-overwrite the current motion command

Return Value

No return value.

Related Information

2.2.1.2. LIN: Execute Linear Motion of TCP In the Cartesian Coordinate

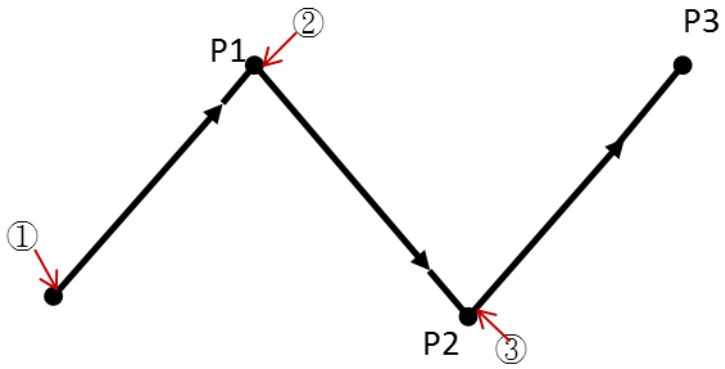
Usage

LIN() is used to execute the endpoint TCP linear motion of the GROUP (robot) in the Cartesian space.

Example 1

```
1  LIN( P1 );
2  LIN( P2 );
3  LIN( P3 );
4
5
```

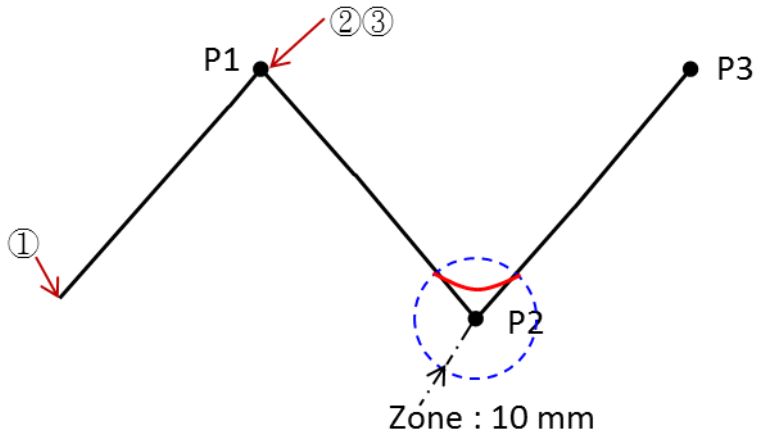
1. Move to P1 in linear mode.
2. Move to P2 with LIN() when P1 is in place.
3. After P2 is in place, execute LIN() to move to P3.



Example 2

```
1  LIN( P1 );
2  LIN( P2, Z=10 );
3  LIN( P3 );
4
5
```

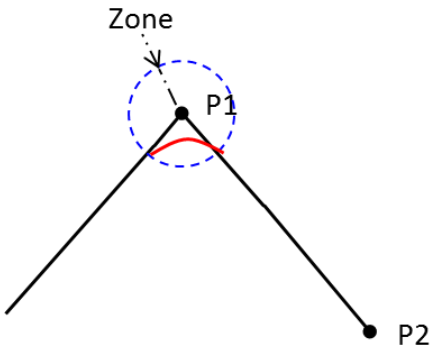
1. Move to P1 in linear mode.
2. Move to P2 with LIN() when P1 is in place. When distance between current point and P2 is 10 mm (Zone=10), P3 is moved with LIN().
3. Wait until P3 is in place.



Command

LIN (TargetPos, [V], [OV], [TL], [BS], [Z], [ABORT], [CONT]], [OW])

Command	Notes
TargetPos	• Absolute target position • Data type: RP_T or RCP_T • Sets the target position of linear motion of the group, can use any of the two data types below: ○ RP_T: Robot position of composite coordinate which contains Cartesian coordinate and axis coordinate. ○ RCP_T: Robot position of Cartesian coordinate • If RP_T is used, the position of Cartesian coordinate is used as a reference.
[VP]	• Specifies the maximum linear velocity (mm/s) of endpoint (TCP) of GROUP. The parameter can be ignored. • Data type: F64_T • If this parameter is ignored, the motion planning will be scheduled at the maximum linear velocity of the system parameters. • Refer to Group:VEL() on setting the system parameter.
[OV]	• Specifies the maximum orientation velocity (Degree/s) of endpoint (TCP) of GROUP. The parameter can be ignored. • Data type: F64_T • If this parameter is ignored, the motion planning will be scheduled at the maximum orientation velocity of the system parameters. • Refer to Group:OVEL() and learn how to set the system parameter.
[TL]	• Specifies Tool number. The parameter can be ignored. • Data type: I32_T • If this parameter is ignored, the tool number is set by the system

Command	Notes
	parameter. Refer to Tool() and learn how to set the system parameter's tool number.
[BS]	• Specifies the Base number. The parameter can be ignored. • Data type: I32_T • If this parameter is ignored, the base number is set by the system parameter. • Refer to Base() and learn how to set the system parameter of base number.
[Z]	• Specifies the smooth zone and length unit. The parameter can be ignored. • Data type: F64_T • Specifies an area with the target position as the center and the smooth zone as the radius (as shown below). When a linear motion enters the smooth zone, the smooth path is automatically planned to connect with the next motion command, and the velocity does not decrease to zero.  • If the smooth zone is set in a linear motion mode, the NRPL program has to execute the next motion command immediately. The behavior is the same as using [CONT]. • If this parameter is ignored, the subsequent command will be executed after the linear motion is accurately in place.
[ABORT]	• This PTP command can be aborted by subsequent command. The parameter can be ignored. • Data type: flag constant • The connection of the motion commands is in the buffer mode; the motion command will be pushed into the buffer and wait for the previous command to complete. Therefore, if ABORT is declared, it means that the LIN() command can be interrupted by subsequent motion command and executes the next motion command immediately.

Command	Notes
	• If the zone is specified, the ABORT flag can be ignored.
CONT	• Executes the next command continuously. The parameter can be ignored. • Data type: flag constant • LIN() command is preset as a blocked type. It waits until the LIN() motion command accurately in place, then execute the subsequent command. Under some applied conditions, it is desirable to execute subsequent command immediately after the LIN() command is issued. Therefore, the LIN() command can be declared as a non-blocked type with the CONT flag.
OW	• Overwrites the current motion command. The parameter can be ignored. • Data type: flag constant • Will force-overwrite the current motion command

Return Value

No return value.

Related Information

2.2.1.3. ARC: Execute Arc Motion of TCP Under the Cartesian Coordinate

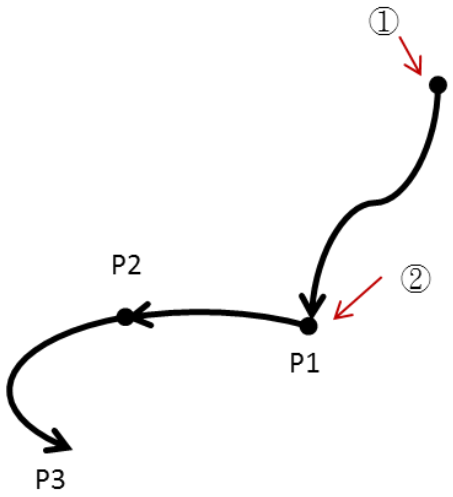
Usage

ARC() is used to execute the endpoint (TCP) arc motion of GROUP (robot) in the Cartesian space.

Example 1

```
1  PTP( P1 );
2  ARC( P2, P3 );
3
4
5
```

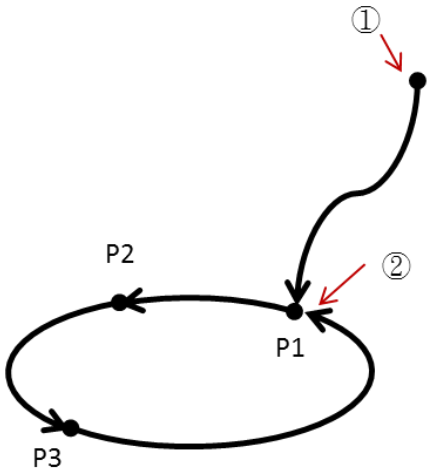
1. Move to P1 in PTP mode
2. After the position is reached, the ARC() command is executed in the arc motion mode to reach P3 through P2.



Example 2

```
1  PTP( P1 );
2  ARC( P2, P3, ANG=360 );
3
4
5
```

1. Move to P1 in PTP mode.
2. After the position is reached, the ARC(P2, P3, ANG=360) is executed in the arc motion mode and reaches P1 after moving 360 degrees through P2 and P3.

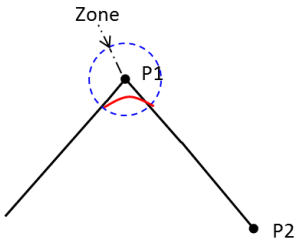


Command

ARC (RPOS_BP, RPOS_TP, [ANG], [V], [OV], [TL], [BS], [Z], [ABORT], [CONT], [OW])

Command	Notes
RPOS_BP	• Absolute target position: through the reference point of the arc • Data type: RP_T or RCP_T • Sets the absolute position of the target passing point of the Group's arc motion, use either of the following two data types: ○ RP_T: Robot position of the composite coordinate which contains Cartesian coordinate and axis coordinate. If RP_T is used, the position of the Cartesian coordinate is used as a reference. ○ RCP_T: Robot position of the Cartesian coordinate • When setting the command, make sure the passing point position of the arc is not too close to the start or end of the arc. The best setting is to use the middle position between the start and end points. If the passing point position is too close to the start or endpoint, the controller may not be able to determine the arc path and return the error. • **Note:** RPOS_BP and RPOS_TP must use the same data type and cannot be used together.
RPOS_TP	• Absolute target position: end point of the arc • Data type: RP_T or RCP_T • Sets the absolute position of the target passing point of the Group's arc motion, use either of the following two data types: ○ RP_T: Robot position of the composite coordinate which contains Cartesian coordinate and axis coordinate. If RP_T is used, the position of the Cartesian coordinate is used as a reference. ○ RCP_T: Robot position of Cartesian coordinate

Command	Notes
	• If arc degree [ANG] parameter is specified, the RPOS_TP parameter becomes the second reference passing point of the arc. • **Note:** RPOS_BP and RPOS_TP must use the same data type and cannot be used together.
[ANG]	• Specifies the arc degree in the arc motion. The parameter can be ignored. • Data type: F64_T • If you want to perform an arc motion of 360 degrees or more, you can directly specify the arc degree by setting this parameter. After setting this parameter, RPOS_TP parameter becomes the second reference passing point.
[V]	• Specifies the maximum linear velocity (mm/s) of the GROUP's endpoint (TCP). The parameter can be ignored. • Data type: I32_T • If this parameter is ignored, the motion planning will be scheduled at the maximum linear velocity of system parameters. Refer to Group:VEL() and learn how to set the system parameter.
[OV]	• Specifies the maximum orientation velocity (Degree/s) of the GROUP's endpoint (TCP). The parameter can be ignored. • Data type: F64_T • If this parameter is ignored, the motion planning will be scheduled at the maximum orientation velocity of the system parameters. Refer to Group:VEL() and learn how to set the system parameter.
[TL]	• Specifies the Tool number. The parameter can be ignored. • Data type: I32_T • If this parameter is ignored, the tool number is set by the system parameter. Refer to Tool() and learn how to set the system parameter.
[BS]	• Specifies the Base number. The parameter can be ignored. • Data type: I32_T • If this parameter is ignored, the base number is set by the system parameter. Refer to Base() and learn how to set the system parameter.
[Z]	• Specifies the smooth zone and length unit. The parameter can be ignored. • Data type: F64_T • Specifies an area with the target position as the center and the smooth zone as the radius (as shown below). When a linear motion enters the smooth zone, the smooth path is automatically planned to connect with

Command	Notes
	the next motion command, and the velocity does not decrease to zero.  • If the smooth zone is set in a linear motion mode, the NRPL program has to execute the next motion command immediately. The behavior is the same as using [CONT]. • If this parameter is ignored, the subsequent command will be executed after the linear motion is accurately in place.
[ABORT]	• This PTP command can be aborted by subsequent command. The parameter can be ignored. • Data type: flag constant • The connection of the motion commands is in the buffer mode; the motion command will be pushed into the buffer and wait for the previous command to complete. Therefore, if ABORT is declared, it means that the ARC() command can be interrupted by subsequent motion command and executes the next motion command immediately. • If the zone is specified, the ABORT flag can be ignored.
CONT	• Executes the next command continuously. The parameter can be ignored. • Data type: flag constant • All of the motion commands are preset as the blocked type, they wait until the motion command to be accurately in place, and then execute the subsequent command. Under some applied conditions, it is desirable to execute subsequent command immediately after the ARC() command is issued. Therefore, the ARC() command can be declared as a non-blocked type with the CONT flag.
OW	• Overwrites the current motion command. The parameter can be ignored. • Data type: flag constant • Will force-overwrite the current motion command

Return Value

No return value.

Related Information

2.2.2.1. AXVEL: Velocity

Usage

2.2.2. AXVEL () is used to set the maximum axis velocity of GROUP parameters. Once the setup is complete, it will be recorded in the GROUP internal parameters until the program ends or the AXVEL() command is called again. PTP() plans the motion based on the maximum velocity of each axis.

Example 1

```
1  AXVEL( A0=10 );
2
3
```

1. Set the maximum velocity of GROUP axis 1 to 10 unit/sec.

Example 2

```
1  AXVEL( A0=20, A2=25, A4=30  );
2
3
```

1. Set the maximum velocity of the GROUP axes:
   - Axis 1 to 20 unit/sec.
   - Axis 3 to 25 unit/sec.
   - Axis 5 to 30 unit/sec.

Command

AXVEL ([A0], [A1], [A2], [A3], [A4], [A5], [A6], [A7]);

Command	Notes
[A0]~[A7]	• The parameter of each GROUP axis (0-7). This parameter can be ignored. • Data type: F64_T • Set the maximum axis velocity of GROUP. The unit is unit/sec. • The unit type alters with the motion type of the GROUP axis, the revolute joint's unit is degree; the prismatic joint's unit is mm. • **Note:** There should be at least one axis is set up between [A0]-[A7].

Return Value

No return value.

Related Information

2.2.2.2. AXACC: Acceleration

Usage

AXACC() is used to set the maximum axis acceleration of the GROUP parameters. Once the setup is complete, it will be recorded in the GROUP internal parameters until the program ends or the AXACC() command is called again. PTP() plans the motion based on the maximum acceleration of each axis.

Example 1

```
1  AXACC( A0=50 );
2
3
```

1. Set the maximum acceleration of GROUP axis 1 to 50 unit/sec^2.

Example 2

```
1  AXACC( A0=20, A2=25, A4=30  );
2
3
```

1. Set the maximum acceleration of GROUP axes:
   - Axis 1 to 20 unit/sec^2
   - Axis 3 to 25 unit/sec^2
   - Axis 5 to 30 unit/sec^2

Command

AXACC ([A0], [A1], [A2], [A3], [A4], [A5], [A6], [A7]);

Command	Notes
[A0]~[A7]	• The parameter of each GROUP axis (0-7). This parameter can be ignored. • Data type: F64_T • Set the maximum axis velocity of GROUP. The unit is unit/sec^2. • The unit type alters with the motion type of the GROUP axis, the revolute joint's unit is degree; the prismatic joint's unit is mm. • **Note:** There should be at least one axis is set up between [A0]-[A7].

Return Value

No return value.

Related Information

2.2.2.3. AXDEC: Deceleration

Usage

AXDEC() is used to set the maximum axis deceleration of the GROUP parameters. Once the setup is complete, it will be recorded in the GROUP internal parameters until the program ends or the AXDEC() command is called again. PTP() plans the motion based on the maximum deceleration of each axis.

Example 1

```
1  AXDEC( A0=50 );
2
3
```

1. Set the maximum deceleration of GROUP axis 1 to 50 unit/sec^2.

Example 2

```
1  AXDEC( A0=20, A2=25, A4=30  );
2
3
```

1. Set the maximum deceleration of GROUP axes:
   - Axis 1 to 20 unit/sec^2
   - Axis 3 to 25 unit/sec^2
   - Axis 5 to 30 unit/sec^2

Command

AXDEC ([A0], [A1], [A2], [A3], [A4], [A5], [A6], [A7]);

Command	Notes
[A0]~[A7]	• The parameter of each GROUP axis (0-7). This parameter can be ignored. • Data type: F64_T • Set the maximum axis velocity of GROUP. The unit is unit/sec^2. • The unit type alters with the motion type of the GROUP axis, the revolute joint's unit is degree; the prismatic joint's unit is mm. • **Note:** There should be at least one axis is set up between [A0]-[A7].

Return Value

No return value.

Related Information

2.2.2.4. AXJERK: Jerk

Usage

AXJERK() is used to set the maximum axis jerk of the GROUP parameters. Once the setup is complete, it will be recorded in the GROUP internal parameters until the program ends or the AXJERK() command is called again. PTP() plans the motion based on the maximum jerk of each axis.

Example 1

```
1  AXJERK( A0=50 );
2
3
```

1. Set the maximum jerk of GROUP axis 1 to 50 unit/sec^3.

Example 2

```
1  AXJERK( A0=20, A2=25, A4=30  );
2
3
```

1. Set the maximum jerk of GROUP axes:
   - Axis 1 to 20 unit/sec^3
   - Axis 3 to 25 unit/sec^3
   - Axis 5 to 30 unit/sec^3

Command

AXDEC ([A0], [A1], [A2], [A3], [A4], [A5], [A6], [A7]);

Command	Notes
[A0]~[A7]	• The parameter of each GROUP axis (0-7). This parameter can be ignored. • Data type: F64_T • Sets the maximum GROUP axis velocity . The unit is unit/sec^3. • The unit type alters with the motion type of the GROUP axis, the revolute joint's unit is degree; the prismatic joint's unit is mm. • **Note:** There should be at least one axis is set up between [A0]-[A7].

Return Value

No return value.

Related Information

2.2.2.5. VEL: Linear Velocity

Usage

VEL() is used to set the linear velocity of the GROUP parameters. Once the setup is complete, it will be recorded in the GROUP internal parameters until the program ends or the VEL() command is called again. LIN() or ARC() plans the linear motion based on the linear velocity.

Example

```
1  VEL( 100 );
2
3
```

1. Set the linear velocity of GROUP axis 1 to 100 mm/sec.

Command

VEL (Vel);

Command	Notes
Vel	• GROUP linear velocity • Data type: F64_T • Set the GROUP linear velocity. The Unit is mm/sec.

Return Value

No return value.

Related Information

2.2.2.6. ACC: Linear Acceleration

Usage

ACC() is used to set the linear acceleration of the GROUP parameters. Once the setup is complete, it will be recorded in the GROUP internal parameters until the program ends or the ACC() command is called again. LIN() or ARC() plans the linear motion based on the linear acceleration.

Example

```
1  ACC( 100 );
2
3
```

1. Set the linear acceleration of GROUP axis 1 to 100 mm/sec^2.

Command

ACC (Acc);

Command	Notes
Acc	• GROUP linear acceleration • Data type: F64_T • Set the GROUP linear acceleration. The Unit is mm/sec^2.

Return Value

No return value.

Related Information

2.2.2.7. DEC: Linear Deceleration

Usage

DEC() is used to set the linear deceleration of the GROUP parameters. Once the setup is complete, it will be recorded in the GROUP internal parameters until the program ends or the DEC() command is called again. LIN() or ARC() plans the linear motion based on the linear deceleration.

Example

```
1  DEC( 100 );
2
3
```

1. Set the linear deceleration of GROUP axis 1 to 100 mm/sec^2.

Command

DEC (Dec);

Command	Notes
Dec	• GROUP linear deceleration • Data type: F64_T • Set the GROUP linear deceleration. The Unit is mm/sec^2.

Return Value

No return value.

Related Information

2.2.2.8. JERK: Linear Jerk

Usage

JERK() is used to set the linear jerk of the GROUP parameters. Once the setup is complete, it will be recorded in the GROUP internal parameters until the program ends or the JERK() command is called again. LIN() or ARC() plans the linear motion based on the linear jerk.

Example

```
1  JERK( 100 );
2
3
```

1. Set the linear jerk of GROUP axis 1 to 100 mm/sec^3.

Command

JERK (Jerk);

Command	Notes
Jerk	• GROUP linear jerk • Data type: F64_T • Set the GROUP linear jerk. The Unit is mm/sec^3.

Return Value

No return value.

Related Information

2.2.2.9. OVEL: Orientation Velocity

Usage

OVEL() is used to set the orientation velocity of GROUP parameters. Once the setup is complete, it will be recorded in the GROUP internal parameters until the program ends or the OVEL() command is called again. LIN() or ARC() plan the linear motion based on the orientation velocity.

Example

```
1  OVEL( 50 );
2
3
```

1. Set the orientation velocity of GROUP is 50 deg/sec.

Command

OVEL (OVel);

Command	Notes
OVel	• GROUP orientation velocity • Data type: F64_T • Set the GROUP orientation velocity. The Unit is mm/sec^3.

Return Value

No return value.

Related Information

2.2.2.10. OACC: Orientation Acceleration

Usage

OACC() is used to set the orientation acceleration of the GROUP parameters. Once the setup is complete, it will be recorded in the GROUP internal parameters until the program ends or the OACC() command is called again. LIN() or ARC() plans the linear motion based on the orientation acceleration.

Example

```
1  OACC( 50 );
2
3
```

1. Set the orientation acceleration of GROUP is 50 deg/sec^2.

Command

OACC (OAcc);

Command	Notes
OAcc	• GROUP orientation acceleration • Data type: F64_T • Set the GROUP orientation acceleration. The Unit is mm/sec^2.

Return Value

No return value.

Related Information

2.2.2.11. ODEC: Orientation Deceleration

Usage

ODEC() is used to set the orientation deceleration of the GROUP parameters. Once the setup is complete, it will be recorded in the GROUP internal parameters until the program ends or the ODEC() command is called again. LIN() or ARC() plans the linear motion based on the orientation deceleration.

Example

```
1  ODEC( 50 );
2
3
```

1. Set the GROUP orientation deceleration to 50 deg/sec2.

Command

ODEC (ODec);

Command	Notes
ODec	• GROUP orientation deceleration • Data type: F64_T • Set the GROUP orientation deceleration. The Unit is mm/sec^2.

Return Value

No return value.

Related Information

2.2.2.12. OJERK: Orientation Jerk

Usage

OJERK() is used to set the orientation jerk of the GROUP parameters. Once the setup is complete, it will be recorded in the GROUP internal parameters until the program ends or the OJERK() command is called again. LIN() or ARC() plan the linear motion based on the orientation jerk.

Example

```
1  OJERK( 50 );
2
3
```

Set the orientation jerk of GROUP is 50 deg/sec^3.

Command

OJERK (OJerk);

Command	Notes
OJerk	• GROUP orientation jerk • Data type: F64_T • Set the GROUP orientation jerk. The Unit is mm/sec^3.

Return Value

No return value.

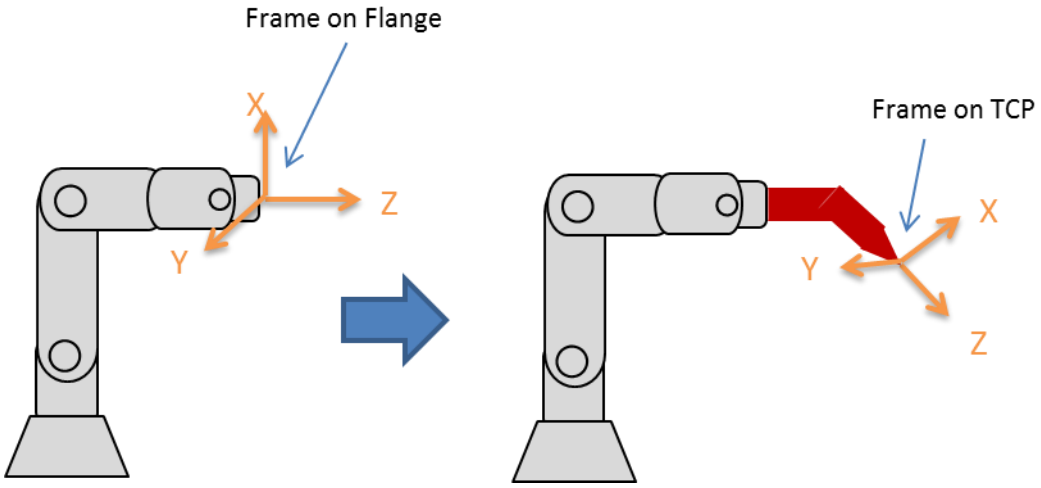
Related Information

2.2.2.13. TOOL: Tool Number

Usage

TOOL() is used to set the tool number of the GROUP parameters. Once the setup is complete, it will be recorded in the GROUP internal parameters until the program ends or the TOOL() command is called again.

If a tool number is not set, the target position of the motion command will represent the position and orientation of the flange's endpoint. When the tool parameter is used, the tool center point (TCP) coordinate system is attached to the endpoint of the tool based on the conversion relationship.



Each GROUP supports up to 16 sets of tools. Each set of tool parameters can be set up through the teach pendant user interface (TPUI) or by calling the TDAT() command settings.

When planning or teaching the position points, you must pay attention to the Tool and Base's current numbers. When using the NRPL, you must confirm all the points and Tool numbers match with the Base number. If they do not match, the path of the robot may not as you expected.

Example 1

```
1  TOOL( 0 );
2  LIN( P1 );
3
```

1. Set the GROUP system parameter: Tool number is 0.
2. Execute LIN() command to move to P1 in linear motion in the Cartesian space. The P1 position is based on the coordinate system

of tool number 0.

Example 2

```
1  TOOL( 0 );
2  LIN( P1 );
3  LIN( P2, TL=1 );
4  LIN( P3 )
5
6
```

1. Set the GROUP system parameter. The tool number is 0.
2. Execute LIN() to move to P1 in linear motion in the Cartesian space. The P1 position is planned based on the coordinate system of tool number 0.
3. Execute LIN() to move to P2 in linear motion in the Cartesian space. The P2 position is planned based on the coordinate system of tool number 1.
4. Execute LIN() to move to P3 in linear motion in the Cartesian space. The P3 position is planned based on the coordinate system of tool number 0.

Command

TOOL (Tool);

Command	Notes
Tool	• GROUP Tool number • Data type: F64_T • Set the GROUP tool numbers (from -1 to 15) • -1: Do not specify the tool, the position is the flange endpoint of the robot. • 0-15: From the 1st set of tool (No. 0) to the 16th set of tool (No. 15).

Return Value

No return value.

Related Information

2.2.2.14. TDAT: Tool Setting

Usage

TDAT() is used to set the GROUP tool parameter. Once the setup is complete, it will be recorded in the GROUP internal parameters until the program ends or the TDAT() command is called again.

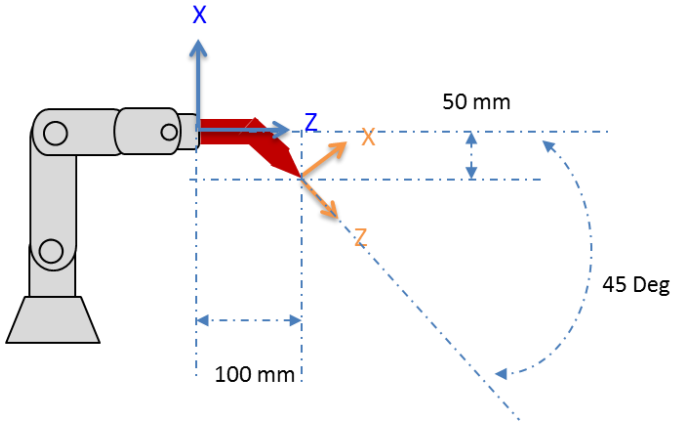
Each GROUP supports up to 16 sets of tools. Each set of tool parameters can be set up through the teach pendant user interface (TPUI) or by calling the TDAT() command settings.

Content of Settings

Data Type	Variable Name	Data Type	Description
TDAT_T	TYPE	I32_T	Always 0
	TRAN.X	F64_T	Offset along flange x-axis
	TRAN.Y	F64_T	Offset along flange y-axis
	TRAN.Z	F64_T	Offset along flange z-axis
	TRAN.A	F64_T	Rotation angle about flange z-axis
	TRAN.B	F64_T	Rotation angle about flange y-axis
	TRAN.C	F64_T	Rotation angle about flange x-axis

If a tool number is not set, the target position of the motion command will represent the position and orientation of the flange's endpoint. When the tool parameter is used, the tool center point (TCP) coordinate system is attached to the endpoint of the tool based on the conversion relationship.

Example: Set a tool parameters as shown below:



Setting Value

Data Type	Variable Name	Value	Description
TDAT_T	TYPE	0	Always 0
	TRAN.X	-50	Offset along flange x-axis
	TRAN Y	0	Offset along flange y-axis
	TRAN.Z	100	Offset along flange z-axis
	TRAN.A	0	Rotation angle about flange z-axis
	TRAN.B	-45	Rotation angle about flange y-axis
	TRAN.C	0	Rotation angle about flange x-axis

```
1  TDAT_T  toolData;
2
3  toolData.TYPE   = 0;
4  toolData.TRAN.X = -50.0;
5  toolData.TRAN.Y = 0.0;
6  toolData.TRAN.Z = 100.0;
7  toolData.TRAN.A = 0.0;
8  toolData.TRAN.B = -45.0;
9  toolData.TRAN.C = 0.0;
10
11 TDAT( 0, toolData );
12
13 LIN( P1, TL=0 );
```

1. Declare a TDAT_T variable.

3-9. Set up Tool parameters.

11. Set tool number 0 tool parameters to GROUP system.

13. Execute the LIN() command to move to P1 in linear motion in the Cartesian space. The P1 position is planned based on the coordinate system of Tool number 0.

Command

TDAT (Tool, TDat);

<table>
<tr><th>Command</th><th>Notes</th></tr>
<tr><td>Tool</td><td>• GROUP Tool number
• Data type: F64_T
• Set the GROUP tool numbers (from 0 to 15)
• 0-15: From the 1st set of tool (No. 0) to the 16th set of tool (No. 15).</td></tr>
<tr><td>TDat</td><td>• GROUP Tool number
• Data type: F64_T
• Set the tool parameters of the numbered GROUP tool
<table>
<tr><th>Data Type</th><th>Variable Name</th><th>Data Type</th><th>Description</th></tr>
<tr><td rowspan="8">TDAT_T</td><td>TYPE</td><td>I32_T</td><td>Always 0</td></tr>
<tr><td>TRAN.X</td><td>F64_T</td><td>Offset along flange x-axis</td></tr>
<tr><td>TRAN.Y</td><td>F64_T</td><td>Offset along flange y-axis</td></tr>
<tr><td>TRAN.Z</td><td>F64_T</td><td>Offset along flange z-axis</td></tr>
<tr><td>TRAN.A</td><td>F64_T</td><td>Rotation angle about flange z-axis</td></tr>
<tr><td>TRAN.B</td><td>F64_T</td><td>Rotation angle about flange y-axis</td></tr>
<tr><td>TRAN.C</td><td>F64_T</td><td>Rotation angle about flange x-axis</td></tr>
</table></td></tr>
</table>

Return Value

No return value.

Related Information

2.2.2.15. BASE: Base Number

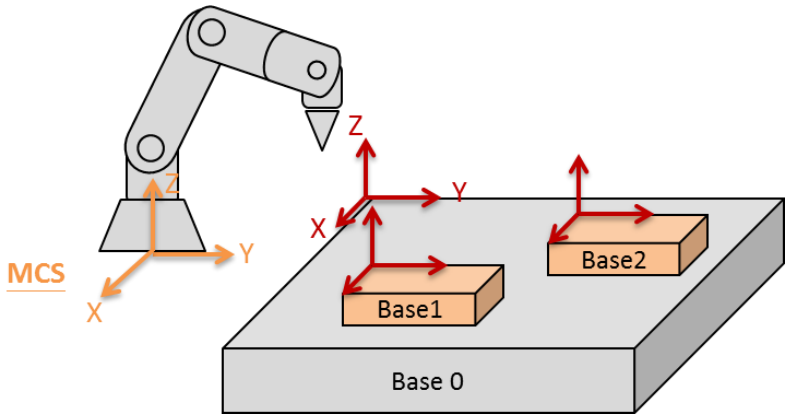
Usage

BASE() is used to set the base number of the GROUP parameters. Once the setup is complete, it will be recorded in the GROUP internal parameters until the program ends or the BASE() command is called again.

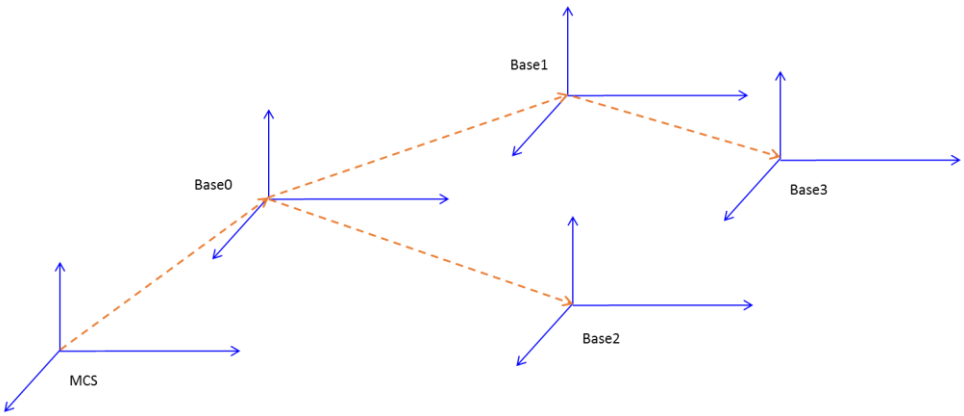
Specifications of base coordinate of GROUP:

- Support up to 32 sets of base coordinates
- Up to 3 levels of hierarchies

The advantage of setting the base coordinate is that the coordinates of the base referenced when teaching the path points are relative and variable. For example, we can set the work table as a set of reference base. When the work table moves(the actual point position changes), only a new reference base coordinates needs to be set. All of point positions do not need to reteach and can be used immediately.



The base coordinate setting is hierarchical, as shown above. Base0 is relative to the Machine Coordinate System (MCS), also known as the robot coordinate system, and then Base1 and Base2 are relative to Base0. When Base0 changes, Base1 and Base2 also change.



The setting method is to set the parameters of 0xC0-0xDF (total 32 sets).

Setting Definitions

Param. Num.	Sub. Index	Data Type	Description
0xC0-DF	0	F64_T	Offset along reference base x-axis
	1	F64_T	Offset along reference base y-axis
	2	F64_T	Offset along reference base z-axis
	3	F64_T	Rotation angle about reference base z-axis
	4	F64_T	Rotation angle about reference base y-axis
	5	F64_T	Rotation angle about reference PCS x-axis
	6	I32_T	Reference base index

Each GROUP supports up to 32 sets of bases. Each set of the base can be setup by the teach pendant user interface (TPUI) or by calling the BDAT() command.

When planning or teaching the position points, user must pay attention to the Tool and Base number currently used. When using the NRPL, you must confirm all the positions and Tool numbers match with the Base numbers. If they do not match, the path of the robot may not as you expected.

Example 1

```
1 BASE( 0 );
2 LIN( P1 );
3
```

1. Set the GROUP system parameter: Base number is 0.
2. Execute the LIN() command to move to P1 in linear motion in the

Cartesian space. The P1 position is based on the coordinate system of base No. 0.

Example 2

```
1 BASE( 0 );
2 LIN( P1 );
3 LIN( P2, TL=1 );
4 LIN( P3 )
5
6
```

1. Set the GROUP system parameter: Base number is 0.
2. Execute the LIN() command to move to P1 in linear motion in the Cartesian space. The P1 position is based on the coordinate system of base No. 0.
3. Execute the LIN() command to move to P2 in linear motion in the Cartesian space. The P2 position is based on the coordinate system of base No. 1.
4. Execute the LIN() command to move to P3 in linear motion in the Cartesian space. The P3 position is based on the coordinate system of base No. 0.

Command

BASE (Base);

Command	Notes
Base	• GROUP Base number • Data type: F64_T • Set the GROUP base numbers (from -1 to 31) • -1: Do not specify the base, use the MCS (robot coordinate system) as the reference of the base coordinate. • 0-31: From the 1st set of tool (No. 0) to the 32nd set of tool (No. 31).

Return Value

No return value.

Related Information

2.2.2.16. BDAT: Base Setting

Usage

BDAT() is used to set the GROUP base parameters. Once the setup is complete, it will be recorded in the GROUP internal parameters until the program ends or the BDAT() command is called again.

Each GROUP supports up to 32 sets of bases. Each set of the base can be setup by the TPUI or by calling the BDAT() command.

Content of Settings

Data Type	Variable Name	Data Type	Description
BDAT_T	TYPE	I32_T	Always 0
	TRAN.X	F64_T	Offset along flange x-axis
	TRAN.Y	F64_T	Offset along flange y-axis
	TRAN.Z	F64_T	Offset along flange z-axis
	TRAN.A	F64_T	Rotation angle about flange z-axis
	TRAN.B	F64_T	Rotation angle about flange y-axis
	TRAN.C	F64_T	Rotation angle about flange x-axis
	REF_BASE	I32_T	Reference base index

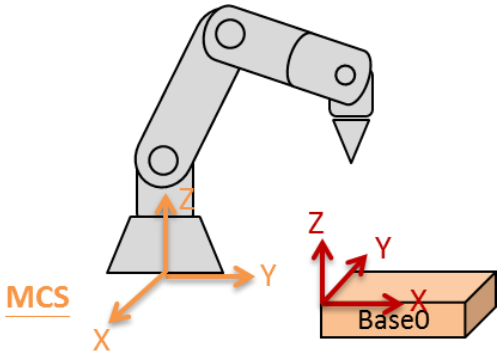
When planning or teaching the position points, you must pay attention to the current Tool and Base numbers. When using the NRPL, you must confirm all the positions and Tool numbers match with the Base numbers. If they do not match, the path of the robot may not as you expected.

Example: Setting the BASE parameters

Joe wants to set the Base coordinate origin (Index = 0) that is relative to the MCS's X, Y, Z axes to the following value:

- X axis: 10 mm
- Y axis:15 mm
- Z axis: 5 mm

Also, the Base0 coordinate system is rotated 90 degrees with respect to the Z axis of the MCS.



xyz@MCS =(10, 15, 5)

Setting Value

Data Type	Variable Name	Value	Description
BDAT_T	TYPE	0	Always 0
	TRAN.X	10	10 mm relative to the GROUP MCS X axis
	TRAN.Y	15	15 mm relative to the GROUP MCS Y axis
	TRAN.Z	5	5 mm relative to the MCS GROUP Z axis
	TRAN.A	90	90 deg. relative to the MCS GROUP Z axis
	TRAN.B	0	0 deg. relative to the MCS GROUP Y axis
	TRAN.C	0	0 deg. relative to the MCS GROUP X axis
	REF_BASE	-1	Relative to MCS

```
1  BDAT_T  baseData;
2
3  baseData.TYPE      = 0;
4  baseData.TRAN.X    = 10.0;
5  baseData.TRAN.Y    = 15.0;
6  baseData.TRAN.Z    = 5.0;
7  baseData.TRAN.A    = 90.0;
8  baseData.TRAN.B    = 0.0;
9  baseData.TRAN.C    = 0.0;
10 baseData.REF_BASE  = -1;
11 BDAT( 0, baseData );
12
13 LIN( P1, BS=0 );
```

1. Declare a BDAT_T variable.

3~9. Setup the Base parameters.

11. Set the base parameters to the GROUP system parameters: Base parameters of the base No. 0.

13. Execute the LIN() command to move to P1 in linear motion in the Cartesian space. The P1 position is based on the coordinate system of base No. 0.

Command

BDAT (Base, BDat);

<table>
<tr><th>Command</th><th>Notes</th></tr>
<tr><td>Tool</td><td>

GROUP Base number
Data type: F64_T
Set the GROUP base numbers (from 0 to 31)
0-31: From the 1st set of base (No. 0) to the 32nd set of base (No. 31).

</td></tr>
<tr><td>BDat</td><td>

GROUP Base number
Data type: BDAT_T
Set the base parameters of the numbered GROUP base:

<table>
<tr><th>Data Type</th><th>Variable Name</th><th>Data Type</th><th>Description</th></tr>
<tr><td rowspan="9">TDAT_T</td><td>TYPE</td><td>I32_T</td><td>Always 0</td></tr>
<tr><td>TRAN.X</td><td>F64_T</td><td>Offset along flange x-axis</td></tr>
<tr><td>TRAN.Y</td><td>F64_T</td><td>Offset along flange y-axis</td></tr>
<tr><td>TRAN.Z</td><td>F64_T</td><td>Offset along flange z-axis</td></tr>
<tr><td>TRAN.A</td><td>F64_T</td><td>Rotation angle about flange z-axis</td></tr>
<tr><td>TRAN.B</td><td>F64_T</td><td>Rotation angle about flange y-axis</td></tr>
<tr><td>TRAN.C</td><td>F64_T</td><td>Rotation angle about flange x-axis</td></tr>
<tr><td>REF_BASE</td><td>I32_T</td><td>Reference base index</td></tr>
</table>
</td></tr>
</table>

Return Value

No return value.

Related Information