ZMotion Basic programming manual

Version 2.7

ZBasic is Basic Programming language which is designed for ZMotion Motion Controllers

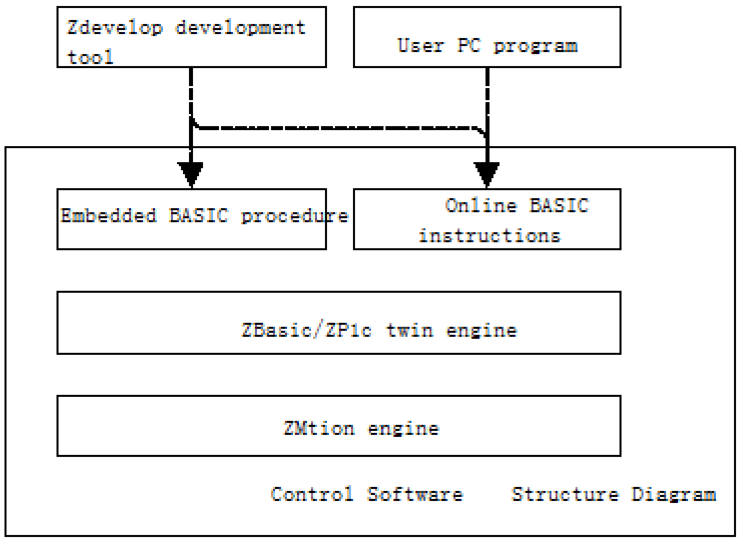
ZBasic provides all standard programming syntax: variable、 array、condition judgment、 circulation and mathematical operation. Besides, the expanded Basic instruction and function provide extensive motion control functions, such as single axis motion、 multi-axes synchronous motion and asynchronous motion ,meanwhile it also supports digital controland simulation I/O control .

ZBasic supports self-defined SUB process, compiling some common functions into self-defined process, which will benefit to compile and revise program; ZBasic also supports SUB process in G code.

ZBasic supports grobal variable, array and SUB process; file module variable, array and SUB process, and local variable.

ZBasic has real-time multi-tasking characteristics, whose multiple ZBasic program can construct and operate simultaneously, making the complex application easy to operate. ZBasic supports interrupt routine, for example, the powefailureinterruption through which the previous state willcan recover.

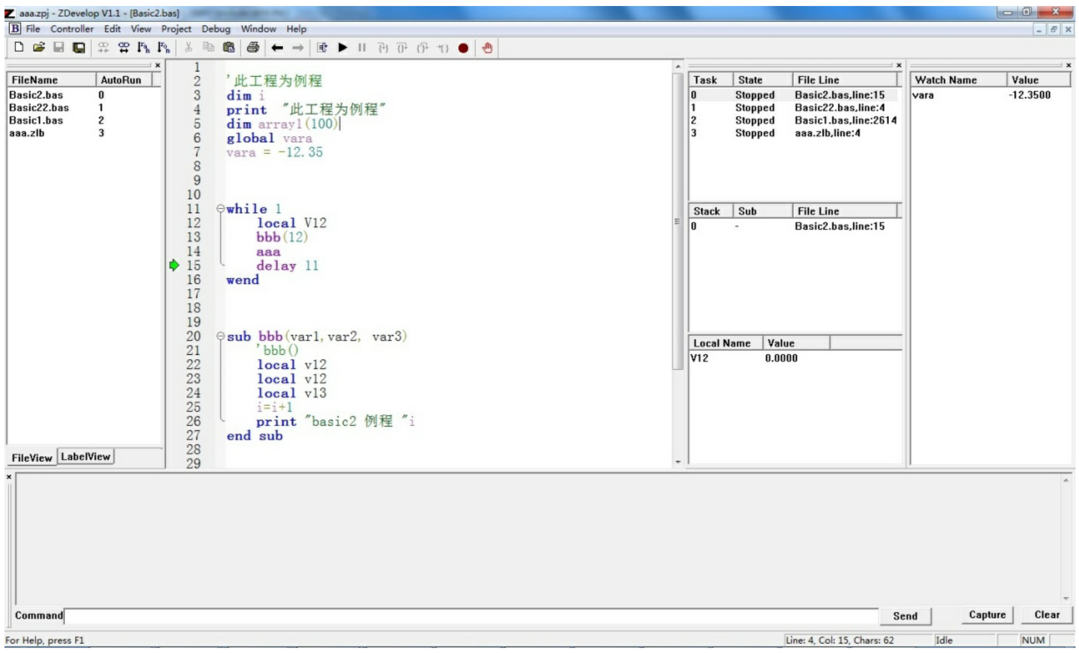
Through PC, sending BASIC instruction online can also achieve the same effect, controller's BASIC program and remote PC's BASIC instruction can operate simultaneously.


Zdevelop development tool
User PC program
Embedded BASIC procedure
Online BASIC instructions
ZBasic/ZPlc twin engine
ZMtion engine
Control Software Structure Diagram

Chapter 1　Introduction to ZBASIC Programming

Compiling and debugging ZBasic program requires ZDevelop development software, ZMC Motion Controller (or emulator)to send PC online instruction requires zmotion.dll dynamic database.

How to use ZDevelop Development Software


aaa.zpj - ZDevelop V1.1 - [Basic2.bas]
File Controller Edit View Project Debug Window Help
FileView LabelView
Command
Send
Capture
Clear
For Help, press F1
Line: 4, Col: 15, Chars: 62
Idle
NUM

FileName	AutoRun
Basic2.bas	0
Basic22.bas	1
Basic1.bas	2
aaa.zlb	3

```
'此工程为例程
dim i
print  "此工程为例程"
dim array1(100)
global vara
vara = -12.35



while 1
    local V12
    bbb(12)
    aaa
    delay 11
wend



sub bbb(var1,var2, var3)
    'bbb()
    local v12
    local v12
    local v13
    i=i+1
    print "basic2 例程 "i
end sub
```

Task	State	File Line
0	Stopped	Basic2.bas,line:15
1	Stopped	Basic22.bas,line:4
2	Stopped	Basic1.bas,line:2614
3	Stopped	aaa.zlb,line:4

Stack	Sub	File Line
0	-	Basic2.bas,line:15

Local Name	Value
V12	0.0000

Watch Name	Value
vara	-12.3500

Simple Steps:

1. Click “File” menu, set up projects.

2. Click “New” menu to set up BAS program file and Compile program code,then click “save” menu to save the same catalogue with project file.

3. Click “Project”-“add to project”,then choose the BAS file saved.

4. Double click the file list on the left side of the window to correspond to the BAS (AutoRun) option,then fill in digit 0.

5. Connect controller or imitation device,then click “download to RAM”menu,it will download the above procedures into controller and autorun.

⚠ When establish no project, only Bas file cannot be downloaded into controller.

What's Procedure?

Procedure consists of sequences, which tells computer how to finish a specific task. To meet users' need, software developers develop procedure by using programming language to describe suitable instruction (sentence) sequence for computer to execute.

A procedure should include the two following aspects.

1. The description to data.A procedure should have specific date type and organization form , that is date structure.(reference to: DIM GLOBAL CONST etc sentence description of variable definition)

2.The description to operation. That is operation steps or algorithm.

To compile algorithm ，usually, three descriptive methods should be applied: sequence、selection、circulation。

Sequence

Without condition and circulation,the procedure always starts from top to bottom.

Function module1
Function module2
Just like above, first, execute function block1, then function block2.
When set autorun, if there is a default local, the files will be executed in an order from top to bottom.
Under BASIC programming, the program is scanned from top to bottom just only one time.
Under PLC programming, the program is scanned from top to bottom in cycle.

Selection

Different executive conditions have different statements executions
The main alternative statements: IF THEN ，ON GOTO ，ON GOSUB etc.

```
dim aa
if aa= 0 then
Statement 1
elseif aa=1 then
```

```
Statement2
else
Statement3
endif
```

Circulation

Executing program repeatedly is called Circulation .

The main loop Statements: FOR NEXT ,WHILE WEND ,REPEAT UNTIL etc.

```
dim a = 0
for i = 1 to 10 step 1
a = a+1
print a
next
```

Multitask

Multitask means that multiple programs can’t interfere with each other and run simultaneously ,one task refers to a program is running .(reference to the instruction description such as RUN)

```
runtask 1, atask
runtask 2, btask
end

atask:
print "atask":ticks
delay(1000)
goto atask

btask:
print "btask":ticks
delay(1000)
goto btask
```

The definition of variable、array and SUB

The variable can be divided into three types : Local variable、File module variable(dim) and Global variable.

The array and SUB can be divided into two types: File module array and Global array.

The variable can be evaluated directly with no definition, the variable can be defined as the default file module variable.

The array and variable must be defined then can be used.

The definition format and function name of SUB () END SUB

Make sure the definition sentence of variable and array can be executed, them should be put in the front of the autorun file.

Put the process of SUB behind the main function END , the SUB will operate as long as the main function internal modulation is used.

Character string

Local parameters must use character string type constant or variable , all kinds of character strings can be combined by "+", and operate single byte by the array

relative instruction The relevant character string instructions list

DIM	The defined array can be applied as character string directly, one element can be regarded as one byte.
""	double quot defines the constant of the character strings
CHR	Convert ASCII into a character string constant, which has only one byte.
MODBUS_STRING	Standard MODBUS agreement defines character string as every 16bit register stores 2 bytes.
VRSTRING	VR list is applied as character string, one VR stores one byte.
+	Operator, combine two character strings
VAL	Convert number string into number.
TOSTR	Convert number into string.
STRCOMP	String comparison function
DMCPY	Array copy function, copy string.

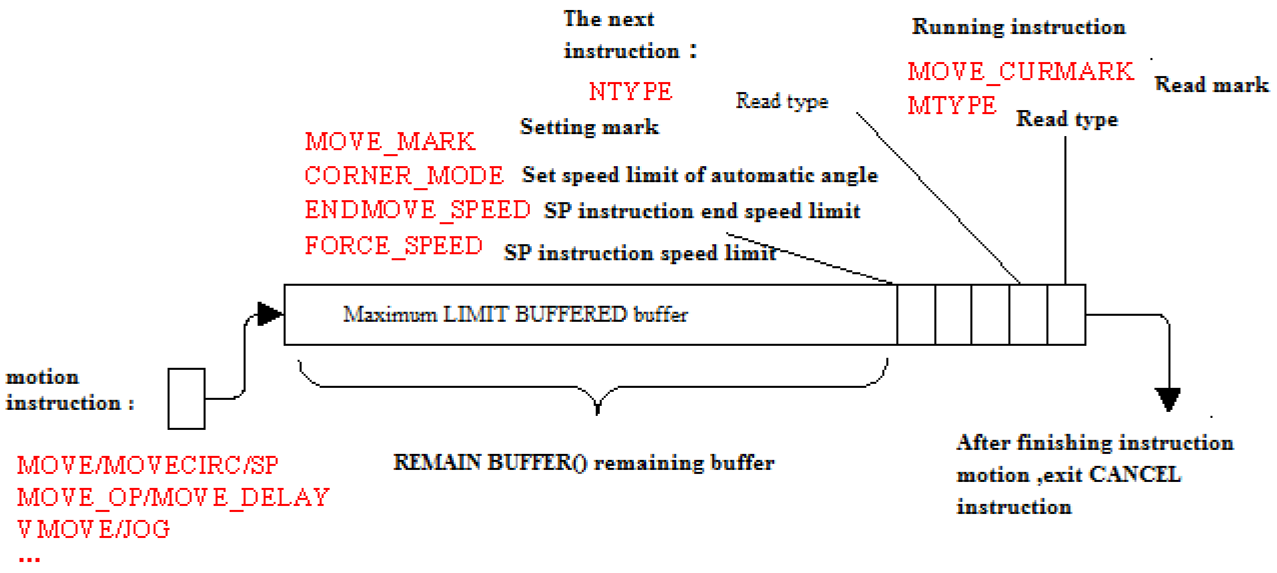
Motion instruction

Motion instruction can achieve point-to-point movement 、 linear interpolation、circular interpolation and cam movement etc.

Set shaft parameters before motion.

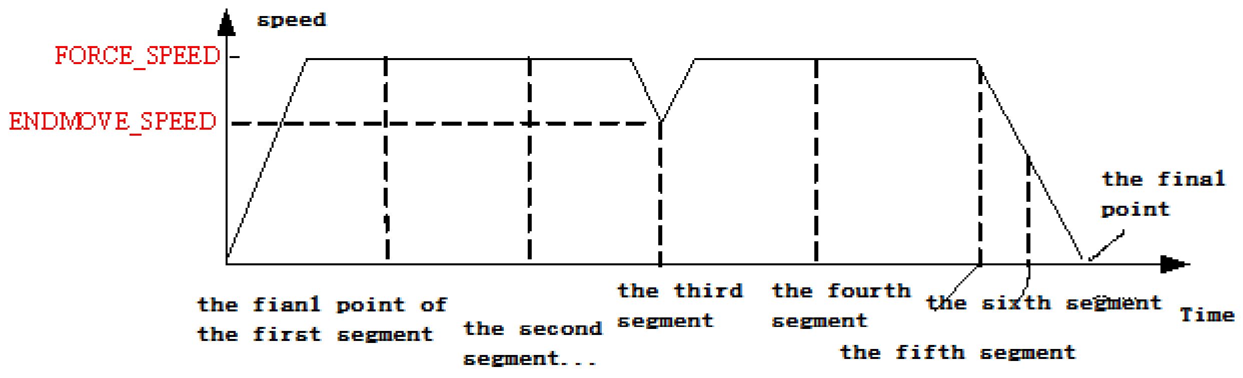
Motion buffer

ZMotion motion controller has multi-level buffers, when one motion instruction is operated, the later motion instruction can be filled into buffer to avoid block the program. You can also fill delay instruction into buffer by MOVE_OP, so the automatic time lag can occurs in motion instructions.


The next instruction：
NTYPE
Read type
Running instruction
MOVE_CURMARK
MTYPE
Read mark
Read type
MOVE_MARK
Setting mark
CORNER_MODE Set speed limit of automatic angle
ENDMOVE_SPEED SP instruction end speed limit
FORCE_SPEED SP instruction speed limit
Maximum LIMIT BUFFERED buffer
motion instruction :
MOVE/MOVECIRC/SP
MOVE_OP/MOVE_DELAY
VMOVE/JOG
...
REMAIN BUFFER() remaining buffer
After finishing instruction motion ,exit CANCEL instruction

Continuous interpolation

After setting MERGE=ON, interpolation has the same main shaft can be continuous automatically, and using SP instruction can set the speed of motion and end by handreference to : MERGE, SP, CORNER_MODE, ENDMOVE_SPEED, FORCE_SPEED



the end of the first stage

Parameter

Parameter includes shaft parameter、task parameter、system parameter etc. Almost parameters can be read and write(except a few only can be read).

Set shaft parameters before motion.

power failure storage

Controller contains battery protection register VR and multi-storage block Flash(?FLASH_SECTES parameter can examine),which is appliedto save power-off date.

ONPOWER power failure function can be used to record the power failure position into VR.

⚠ VR depends on battery to maintain ,it can be unlimited read and wrote, but the date may loss if power off for a long time, some key parameters in the machine should be stored in Flash.

⚠ Flash has write limitation on lifetime, it is rewritable in limited times, rewrited date frequently such as outpuissuggested to writeddate into VR.

Set electricity run in program

Program must be stored in ROM, and set autorun task number according to the correspondent autorun file, reference to the introduction of zdevelop tool software.

How to use EtherCAT

It must be a controller that supports EtherCAT before using the associate instructions

The Rtex bus uses a set of program instructions, but the functions are diff erent.

Please refer to the twelfth chapter bus instructions in detail.

Basic concept:

Node number---array in sequence of connection, starting from NO.0---Node number-1

Driver number--- array in sequence of connection, starting from NO.0---ecat diver number -1.only when there is AXIS_ADDRESS configuration, it plays a role.

Axle number---controller numbers axis, from No.0 to shaft number-1,**which** can reflect to any driver.

1. Use SLOT_SCAN insstruction to scan the devices, judging whether RETURN is ture or not.
2. Judge device type through NODE_INFO/ NODE_AXIS_COUNT.
3. Set AIXS_ADDRESS, ,ATYPE ,drive_profile ,drive_io etc one by one.
4. SLOT_START starts device.

SLOT_SCAN	Scan device
SLOT_START	Bus start
SLOT_STOP	Bus stop
ATYPE	Shaft type 65-ecat cycle position control
AXIS_ADDRESS	Axle address configuration
WDOG	Axis unified enable control
SERVO_PERIOD	Refresh rate
AXIS_ENABLE	Axis enable control
SDO_WRITE	SDO operation
SDO_READ	SDO operation
SDO_WRITE_AXIS	Driver SDO operation
SDO_READ_AXIS	Driver SDO operation
?*ETHERCAT	Bus information output
NODE_COUNT	Device number
NODE_STATUS	Device condition
NODE_AXIS_COUNT	Device belt drive number
NODE_IO	Device IO number
NODE_AIO	Device AIO number
NODE_INFO	Device information read
NODE_PROFILE	Device profile setting, select PDO message date
DRIVE_FE	Driving error
DRIVE_FE_LIMIT	Driving error setting
DRIVE_STATUS	Driving status word
DRIVE_TORQUE	Driver torque feedback
DRIVE_MODE	Driver motion model
DRIVE_PROFILE	Driver profile setting, select PDO message date
DRIVE_CONTROLWORD	Driver control word
DRIVE_CW_MODE	Driver control word Operation model

DRIVE_IO	Remote IO number setting

Program debugging

connect Zdevelop to the controller, download program into RAM, selecting from menu debugging or starting debugging, make sure Consistent between PC program file and controller program file ,if not, the cursor position may be wrong ;Only when controller UNLOCK, the program can be debugged.

Watch out ! when Debugging machine. Must set efficient safety protection device in the machine, and add error handling program , or ZMotion isn't responsible for the loss.

Connect HMI

Controller support MODBUS-RTU or MODBUS-TCP agreement to connect man-machine interface.

Use parameter ADDRESS to set agreement station number, default 1. the serial interface connection should configurate relevant serial interface number, when connecting Ethernet, the IP of PLC terminal which configurate man-machine interface is the IP of controller (imitation device IP 127.0.0.1), terminal number 502.

The setting diagram of Eview touch screen of Ethernet

Common problems

When some mistakes happen in program operation, ZDevelop software can show error information, if there isn't error information, you can check the error information again by inputting *task in command line, double click the error information can switch to program error site automatically. you can read "error code list " file to find the unfound Corresponding codes in the following list.

problems	possible reasons
2043:Unknown function is m	controller's unknown function
stop of error:2049: Line not ended.	1.local command must occupy the whole li 2.GSUB call needs no brackets
stop of error:2033: Unknown label is met.	1，Undefined variable and array 2， Undefined SUB process 3,Defined array, but defined sentence i executed ,

	4,Probably the file isn't set the autorun
2048:Function can only appliedin expression	Only read instructions, not operate valuation
2064:Param few	input a few parameters
2063:Param too many	input too many parameters
2072:Need = sign	forget to write equal
2060:Syntax format error	instruction grammatical errors
error:1010	Repetition pause
error:1011	no motion, no pause

Chapter 2 Motion Instruction

When the existing motion instruction is executing, the latter invoked motion instruction will be buffered automatically , each axis of Zmotion motion controller supports up to512 levels motion buffer (controller buffer number differs in different models), when all buffer are occupied, the latter invoked motion instruction will block the present task.until the buffer is vacant.

Every motion instruction has a MOVE_MARK parameter, using MOVE_CURMARK you can know where the the present buffer runs.

Multiaxial moving interpolation instruction like MOVE uses public axis parameter like SPEED, but it has relevant SP instruction, SP instruction appoints different parameter like speed for each motion, referring to *SP.

Axis parameter MERGE is appliedto set whether the multiaxial moving interpolation instruction can slow down to zero, when MERGE=ON, it has no deceleration, current axis parameter CORNER_MODE can set whether the multiaxial moving interpolation instruction decelerates automatically to necessary speed.

Zmotion motion controller supports move pause,referring to MOVE_PAUSE.

Zmotion motion controller supports move addax, referring to ADDAX.

BASE

Type	motion instruction
Description	Choose parameter and participation motive axis that are to be set. default value in oder is 0 , 1 , 2… The next BASE order that is executed in program is regarded as

	choose axis in according to the last BAST order. Each task has its independent axis list, it can remember BASE choose axis,which is applied in different machine control. When operating the interpolation motion, the motion parameter of the first axis is regarded as the interpolation parameter. See example one. If all axis are not listed in BASE instruction, the BASE instruction will arrange the remaining axis behind in order. Reference to example two.
grammar grammar	BASE(axis<,second axis><,third axis>...) axis the first axis second axis the second axis … Parameter at most has the axis number which is supported by controller, referring to relevant controller hardware manual .
Application controller	general use
example Example	example one BASE(0,1,2,3) ' Axial list option is 0,1,2,3 SPEED=100,10,20,30 Current axis 0,1,2,3 are set corresponding speeds,but when in interpolation motion condition, only the speed of the main axis :100 works. MOVE(100,100,100,100) Axis 0,1,2,3 combine interpolation, and the combined speed is 100,each axis's speed is component velocity. Example two BASE(1) Axial list option is 1 MOVE(100,100,100) ' Axis 1,2,3 run interpolation motion Example three BASE(0,2,5) 'Axial list option is 0,2,5 MOVE(100,100,100) Axis 0,2,5 run interpolation motion

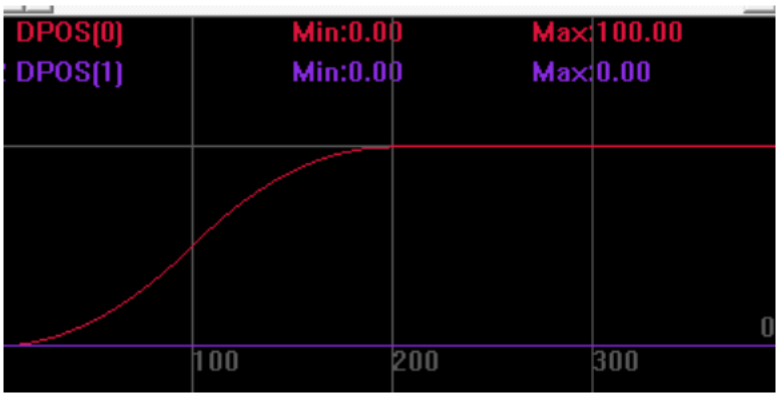
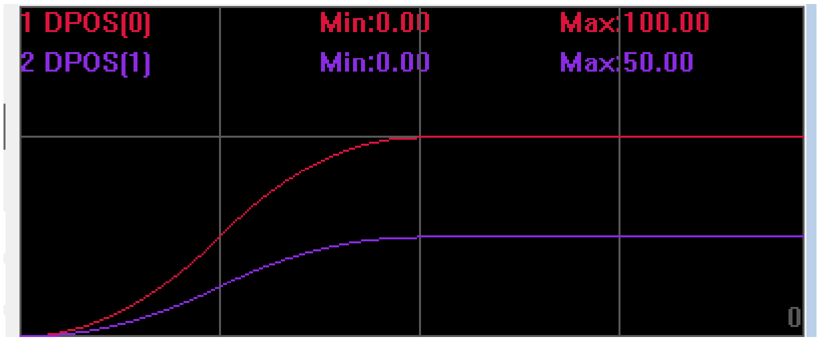
AXIS

type	auxiliary instruction

description	Temporarily revise a motion instruction or axis parameter into a appointed axis to execute. For axis parameter ,the AXIS can be omitted.
grammar	AXIS(expression) expression After executing new axis number which is revised temporarily, the axis option is Still subject to BASE instruction.
Application controller	general use
example	Example one BASE(0) MOVE(1000) AXIS(1) Currently Forced axis 1 motives 1000unit MOVE(100) ' Axis0 motives 100 Example two BASE(1) UNITS(0)=100 Forced appointed axis 0UNITS is set as 100 UNITS=10 Set axis 1UNITS as 10
relative instruction	BASE

ADDAX

type	single axis motion instruction
description	Motion add means adding the motion of a axis to another axis ADDAX instruction overlaps pulse number, not arranged units. Conversion Relation:overlap axis motion distance*overlap axis UNITS /overlapped axis UNTIS =overlapped axis motion distance. Assuming axis A's UNITS is 100, axis B's UNITS is 50, the overlap axis motion is 100 Overlap axis A's motion to axis B,the motion of the current axis A is 100, the axis B is 100*100/50=200. Overlap axis B's motion to axis A,the motion of the current axis B is 100, the axis A is 100*50/100=50. Avoid overlapping simultaneously and mutually between axes.
grammar	ADDAX(axis)
Application controller	general use

example	BASE(0,1) ATYPE=1,1 UNITS=100,200 ‘ Set axis 0 UNITS as 100,axis 1UNITS as 200 MOVE(100) Set the motion of axis0 is 100, current axis 1keep still. ADDAX(0)AXIS(1) According to pulse amount , overlap the motion of axis 0 to axis 1. MOVE(100) Set the motion of axis 0 is 100,the speed of current axis 1 is 100*100/200=50. Take the conversion between the UNITS of two axes into consideration. ADDAX(-1)AXIS(1) ‘Cancel overlap before overlapping  after overlapping 

DEFPOS

type	coordinate instruction
description	**Define the current position as a new absolute position point**
grammar	DEFPOS(pos1 [,pos2[, pos3[, pos4.....]]]) pos1: the absolute position, use unit to definite unit. pos2: the next axis absolute position, use unit to definite unit.
Application controller	general use
example	BASE(0,1) choose axis0, axis1 ATYPE=1,1 UNITS=100,100 'Set UNITS as 100 DPOS=0,0 Set DPOS as 0 MOVE(100,100) The motion of axis 0 and axis 1 is 100 ?DPOS(0),DPOS(1) Currently DPOS are 100 DEFPOS(0,10) 'set the current position ?DPOS(0),DPOS(1) Current DPOS is 0,100
relative instruction	DPOS

CANCEL

type	single axis motion instruction
description	BASE axis stops decelerating, if axis joins interpolation , it also stop interpolation motion. The deceleration of Mode0~2 according to the max value in FASTDEC and DECEL. After CANCEL, waiting WAIT IDLE pause to finish,then calling absolute position motion.
grammar	CANCEL(mode) parameter : mode parameter: mode 0 (default value): Cancel the current motion 1: cancel buffered motion 2: Cancel current motion and buffered motion

	<table><tr><td>3</td><td>Interrupt pulse sent immediately ,simultaneously set dpos as 0</td></tr></table>
Application controller	general use
example	example oneBASE(0) ATYPE=1 UNITS=100 SPEED=100 ACCEL=1000 DECEL=1000 'set the deceleration as 1000 FASTDEC=10000 Set quick deceleration as 10000 MOVE(1000) 'Current motion MOVE(-1000) 'Buffer motion CANCEL(1) Current axis only executed MOVE(1000), it's deceleration is 10000 example two BASE(0) ATYPE=1 DPOS=0 SPEED=100 MOVE(10000) Currently motive 10000 DELAY(2000) Delay two seconds CANCEL(3) 'Currently cutoff pulse send directly, axis pauses immediately, deceleration doesn't work,meanwhile set axis DPOS as 0. example three BASE(0,1) ATYPE=1,1 SPEED=1000 MOVE(1000,1000) Interpolation motion DELAY(1000) Delay one second CANCEL(2)AXIS(1) Pause axis1,and axis1 joins interpolation, and the interpolation

	motion also stops.
relative instruction	RAPIDSTOP ,DECEL,FASTDEC

RAPIDSTOP

type	Multi-axis motion instruction
description	All axises pause immediately, if axis joins interpolation ,and the interpolation motion also stop. The deceleration of Mode0~2 according to the max value in FASTDEC and DECEL After RAPIDSTOP, waiting WAIT IDLE pause to finish,then calling absolute position motion.
grammar	RAPIDSTOP (mode) parameter : mode 0 (default value): Cancel current motion 1: Cancel buffered motion 2: Cancel current motion and buffered motion 3: Interrupt pulse sent immediately ,simultaneously set dpos as 0.
Application controller	general use

example	example oneBASE(0,1,2)
	ATYPE=1,1,1
	UNITS=100
	SPEED=100 'The speed of interpolation Composition motion is 1000.
	ACCEL=1000
	DECEL=1000 Set deceleration as 10000
	FASTDEC=10000 Set quick deceleration as 10000
	MOVE(1000,1000,1000) 'Current motion

	MOVE(-1000,-1000,-1000) Buffer motion RAPIDSTOP (1) Current axis only executed MOVE(1000), it's deceleration is 10000. example two BASE(0,1) ATYPE=1,1 DPOS=0,0 SPEED=100 MOVE(10000,10000) Interpolation motion is 10000 DELAY(2000) Delay two seconds CANCEL(3) Currently cutoff pulse send directly, axis pauses immediately,deceleration doesn't work,meanwhile set axis DPOS as 0.
relative instruction	CANCEL ,DECEL,FASTDEC

FORWARD

type	single axis motion instruction
description	**BASE optional axis forward motion** Must CANCEL ,then switch to REVERSE.
grammar	Forward [axis(轴)]
Application controller	general use
example	example oneBase(0) FORWARD ' axis 0 vmotion in a forward direction WAIT UNTIL IN(1)=ON Wait until input 1 is valid CANCEL(2) example twoFORWARD AXIS(1) ' axis1has forward motion WAIT UNTIL IN(1)=ON Wait until input 1 is valid CANCEL(2)

relative instruction	REVERSE，VMOVE

REVERSE

type	single axis motion instruction
description	**BASE optional axis reverse motion** Must CANCEL ,then switch to FORWARD.
grammar	reverse [axis]
Application controller	general use
example	example one Base(0) REVERSE ’ axis 0 vmove in converse direction WAIT UNTIL IN(1)=ON ‘Wait until input 1 is valid CANCEL(2) example twoREVERSE AXIS(1) Axis 1 moves in converse direction WAIT UNTIL IN(1)=ON Wait until input 1 is valid CANCEL(2)
relative instruction	FORWARD，VMOVE

VMOVE

type	single axis motion instruction
description	When the last VMOVE motion doesn’t stop, the VMOVE instruction will replace the last VMOVE instruction and revise direction automatically,so it doesn’t need to CANCEL the last VMOVE instruction.
grammar	VMOVE(dir1) parameter parameter：dir1 -1 Converse motion 1 forward motio
Application controller	general use

example	BASE(0)
	ATYPE=1
	SPEED=100
	VMOVE(-1) Vmotive in converse direction
	WAIT UNTIL IN(1)=ON 'Wait until input 1 is valid
	VMOVE(1) Vmotive in forward direction

relative instruction	FORWARD,REVERSE

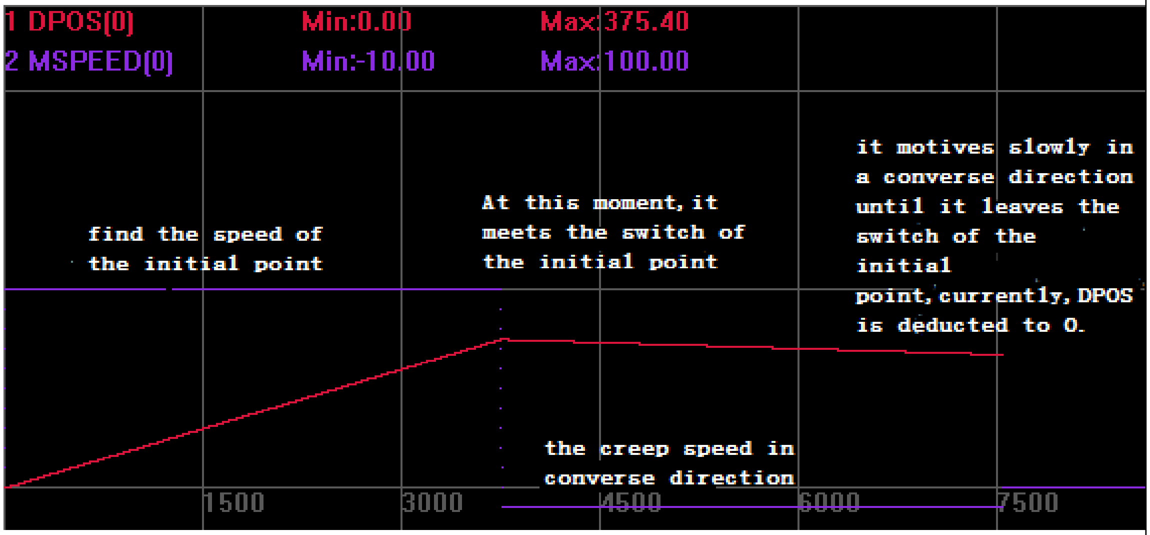
DATUM

type	single axis motion instruction
description	**Find zero-point motion** Input point of the origin point switch is decided by parameter DATUM_IN, positive and negative limit sensors switch is set through FWD_IN and REV_IN. And the trigger is valid when it is 0(can use INVERT_IN to inverted level signal), when the input condition is off, which means origin point or limit sensor reach. It must be configured with signal Z ATYPE to get signal Z back to zero.
grammar	DATUM (mode) , DATUM(21,mode2) parameter : mode back to zero mode, plus 10 means meeting limit sensor then appearing reverse lookup, but having no limit sensor stop, eg:13=3+limit sensor reverse lookup, which is applied in the condition :the origin point in the middle.

value	description
0	Clear error state of all axes
1	Axis forward motives at CREEP speed until signal Z happens It will stop if it meets limit sensor switch. The value of DPOS is replaced as 0,and meanwhile MPOS is revised .
2	Axis converse motives at CREEP speed until signal Z happens It will stop if it meets limit sensor switch. The value of DPOS is replaced as 0,and meanwhile MPOS is revised .
3	Axis forward motives at SPEED speed until meeting the origin point switch. Then axis converse motives at CREEP speed until leaving the origin point

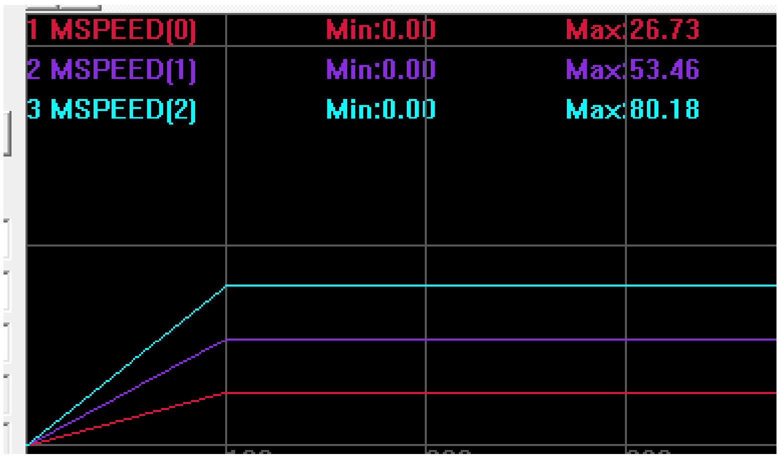
	switch. It will stop directly if meeting limit sensor switch. The value of DPOS is replaced as 0,and meanwhile MPOS is revised .
4	Axis converse motives at SPEED speed until meeting the origin point switch. Then axis forward motives at CREEP speed until leaving the origin point switch. It will stop directly if meeting limit sensor switch. The value of DPOS is replaced as 0,and meanwhile MPOS is revised .
5	Axis forward motives at SPEED speed until meeting the origin point switch. Then axis converse motives at CREEP speed until leaving the origin point switch, then continues to convert at crawl speed until meeting the signal Z. It will stop directly if meeting limit sensor switch. The value of DPOS is replaced as 0,and meanwhile MPOS is revised .
6	Axis converse motives at CREEP speed until meeting the origin point switch. Then axis forward motives at CREEP speed until leaving the origin point switch, then continues to convert at crawl speed until meeting the signal Z. It will stop directly if meeting limit sensor switch. The value of DPOS is replaced as 0,and meanwhile MPOS is revised .
8	Axis forward motives at SPEED speed until meeting the origin point switch. It will stop directly if meeting limit sensor switch.
9	Axis converse motives at SPEED speed until meeting the origin point switch. It will stop directly if meeting limit sensor switch.
21	Use Ethercat driver back to zero function, the time mode2 is valid. Set driver's back to zero mode(6098h),default is 0 which means using driver's current back to zero mode. Can use axis's SPEED, CREEP, ACCEL, set driver's 6099h , 609Ah

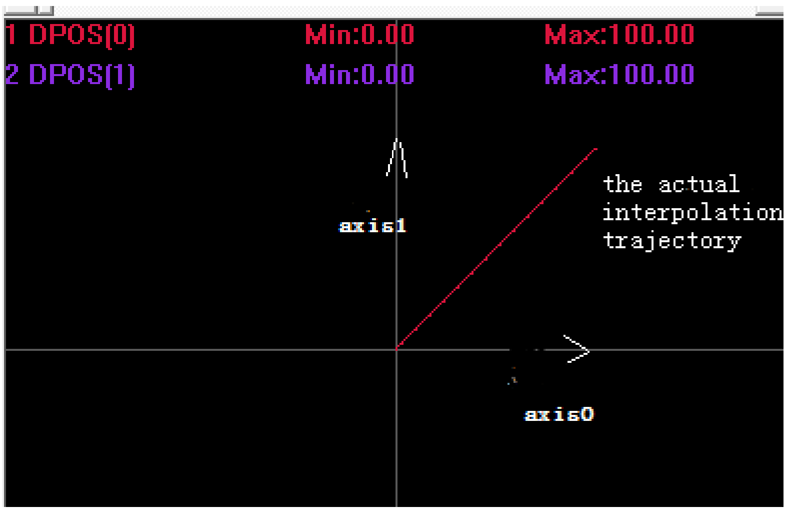
Mode2 means when mode=21,default is 0,when default isn't 0, set driver's back to zero mode,according to driver manual date dictionary 6098h to set value.

Application controller	general use
example	Base(0) Speed = 100 Find the speed of origin point Creep = 10 Converse creep speed Datum_in=5 Input IN5 as origin point switch Invert_in(5,on) 'Inversion IN5 level signal, in general, often close signal can invert. datum(3) 'Axis 0, forward back to zero at 100units/s to find the origin point at 10units/s until leaving the origin point, meanwhile set axis DPOS as 0. **motion trajectory and velocity curve**  In order to visually display , therefore the creep process is longer,in practical application, the creep process is very short.
i	DATUM_IN, INVERT_IN

MOVE

type	multi-axis motion instruction
description	Thelinear interpolation motion, relevant motion for some distance. In interpolation motion ,only main axis speed parameter is valid. The continuous interpolation motion of self-defining speed can use SP suffix instruction, referring to the description of *SP. The distance of the interpolation motion X= $\sqrt{X_0^2+X_1^2+X_2^2+....+X_n^2}$ Motion time T=X/main axis SPEED

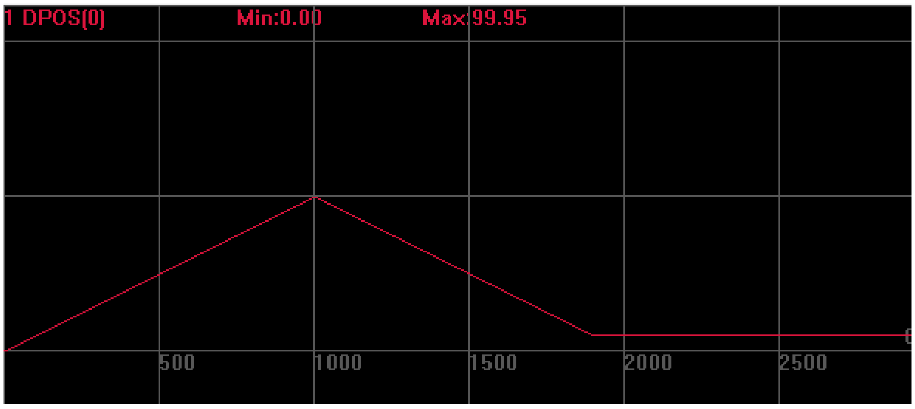
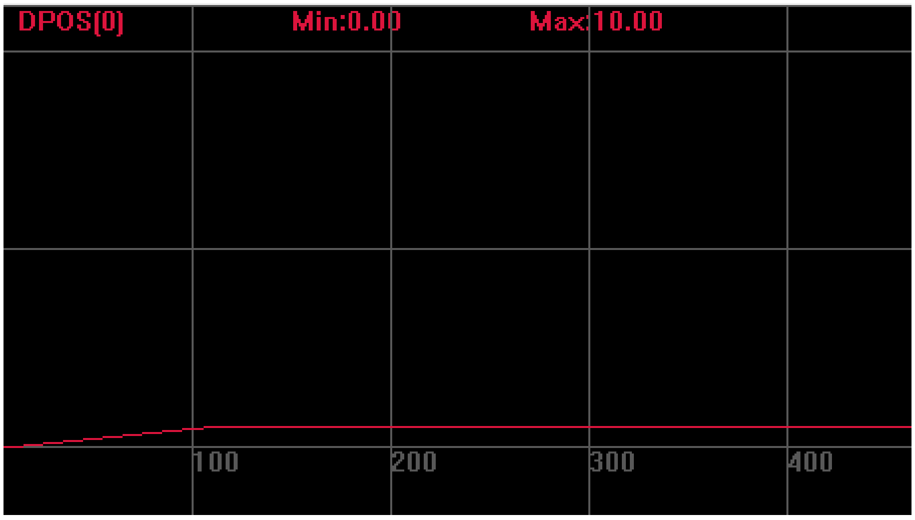
grammar	MOVE(distance1 [,distance2 [,distance3 [,distance4...]]]) parameter： distance1 the distance of the first axis motion distance2 the distance of the second axis motion
Application controller	general use
relative instruction	MOVEABS,*SP
example	example one Base(0,1,2,) ‘Main axis is axis 0 Speed=100,10,1000 ‘Only the speed of the main axis is 100,it works and is regarded as the speed of the component motion. Dpos = 0,0,0 move(500,1000,1500) ’ The relevant distance of axis 0 ,1 ,2 , 3 line interpolation . Wait idle ’ Wait until the motion stop Print *dpos 500,1000,1500 In the interpolation motion,the actual speed of each axis is the decomposing speed of each main axis.  1 MSPEED(0) Min:0.00 Max:26.73 2 MSPEED(1) Min:0.00 Max:53.46 3 MSPEED(2) Min:0.00 Max:80.18 example two Base(0,1) Speed=100,100 Dpos=0,0

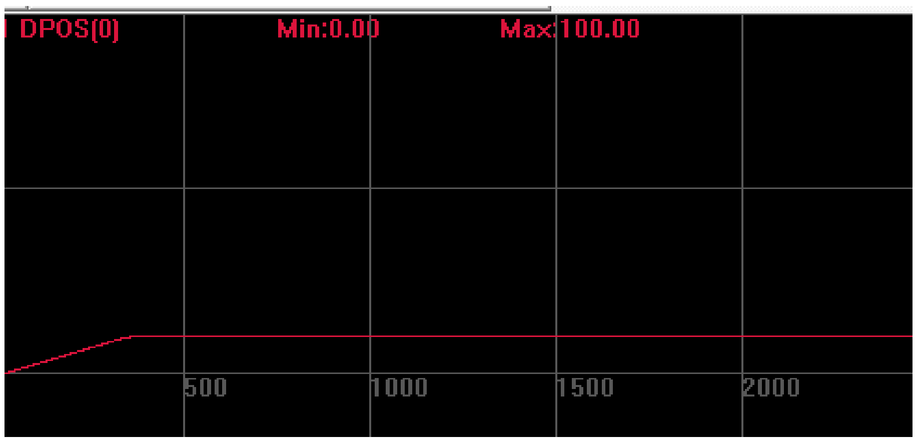
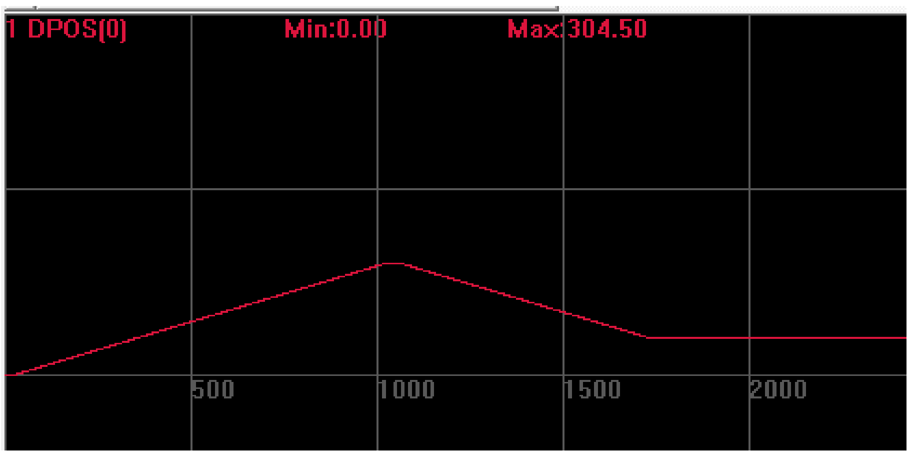
	Move(100,100) interpolation trajectory  1 DPOS(0) Min:0.00 Max:100.00 2 DPOS(1) Min:0.00 Max:100.00 axis1 the actual interpolation trajectory axis0

MOVEMODIFY

type	motion instruction
description	Motionless has the same effect with MOVEABS,but doesn't enter motion buffer state.see example one, need to wait instruction to use it correctly. See example two In continuous interpolation motion, if use MOVEMODIFY ,it will damage speed continuity. MOVEMODIFY corresponds to multi-axis motion simultaneously ,it may not line interpolation.
grammar	MOVEMODIFY (distance1 [,distance2 [,distance3 [,distance4...]]]) distance1 the distance of the first axis motion distance2 the distance of the second axis motion So far only support single axis revise
Application controller	general use

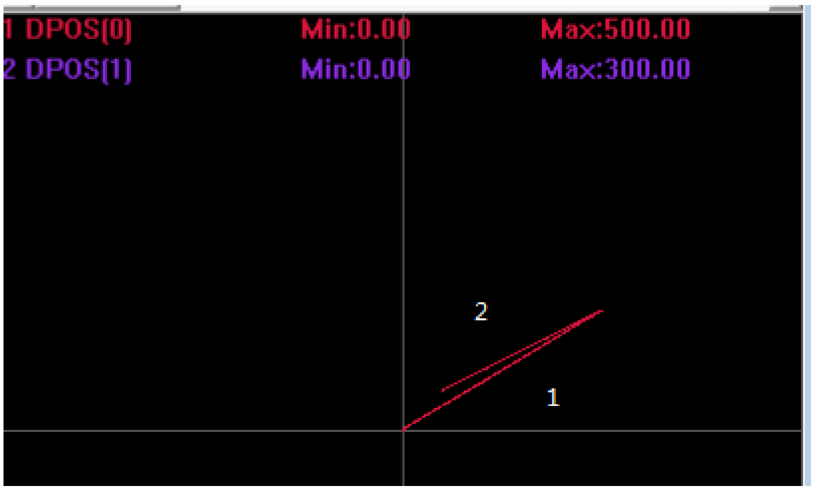
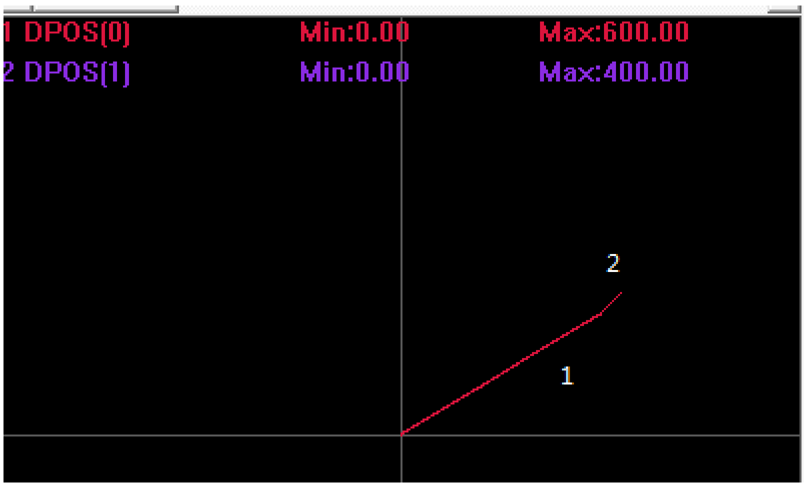
example	example oneBASE(0)
	DPOS=0
	SPEED=100
	MOVEABS(100)
	MOVEABS(10) At first, current axis can motives to100 ,then back to 10. '

	 MOVEMODIFY(100) MOVEMODIFY(10) Current axis motives to site 10 directly,MOVEMODIFY doesn't enter motion buffer.  example twoBASE (0) DEFPOS(0) MOVEABS(500) WAIT UNTIL DPOS >=300 'Wait until it motives to 300, then revise final position MOVEMODIFY (100) Revise target position to be 1000, currently, it will decelerate and stop,then it motives in converse direction. Don't use WAIT,motive to 100 directly.

	 use WAIT 

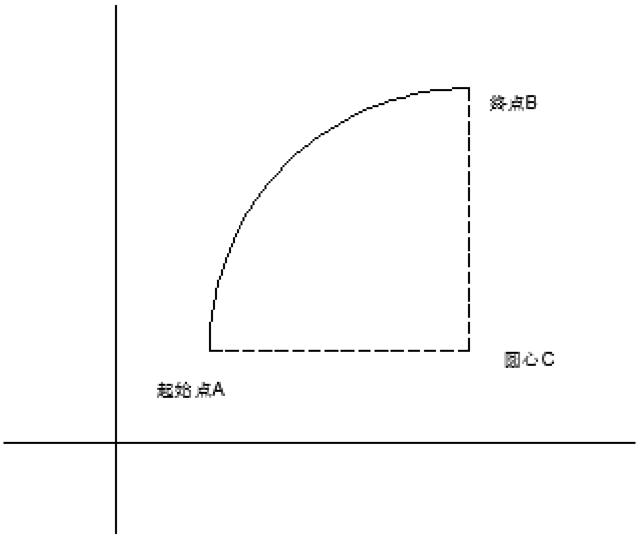
MOVEABS

type	multi-axis motion instruction
description	The line interpolation motion, absolute motion to appointed coordinate. The continuous interpolation motion of Self-defined speed can use SP suffix instruction,see the description of *SP.
grammar	MOVEABS(position1[, position2[, position3[, position4...]]]) parameter : position1 the coordinate of the first axis motion position2 The coordinate of the second axis motion
Application controller	general use
example	BASE(0,1)

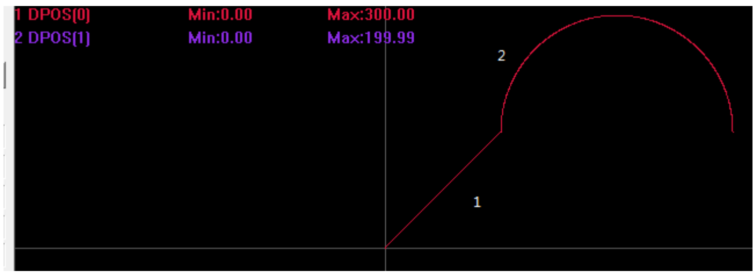
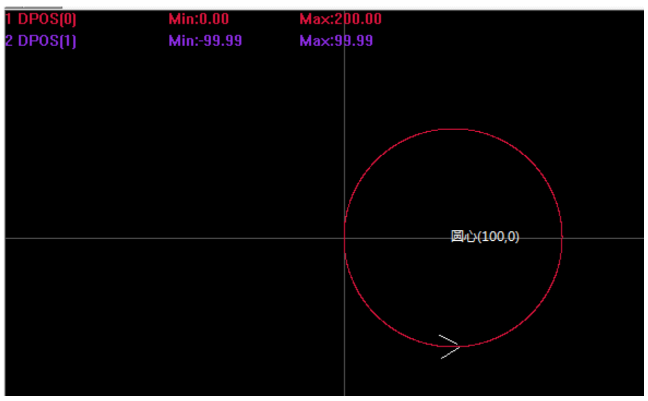
	DPOS=0,0 SPEED=100,100 MOVEABS(500,300) Axis0 motives to 500,axis1 motives to 300,interpolation motion. MOVEABS(100,100) Axis0 motives back to 100,axis1 motives back to 100 interpolation trajectory  1 DPOS(0) Min:0.00 Max:500.00 2 DPOS(1) Min:0.00 Max:300.00 2 1 if use MOVE relevant motion interpolation trajectory  1 DPOS(0) Min:0.00 Max:600.00 2 DPOS(1) Min:0.00 Max:400.00 2 1
relative instruction	MOVE,*SP

MOVECIRC

type	multi-axis motion instruction

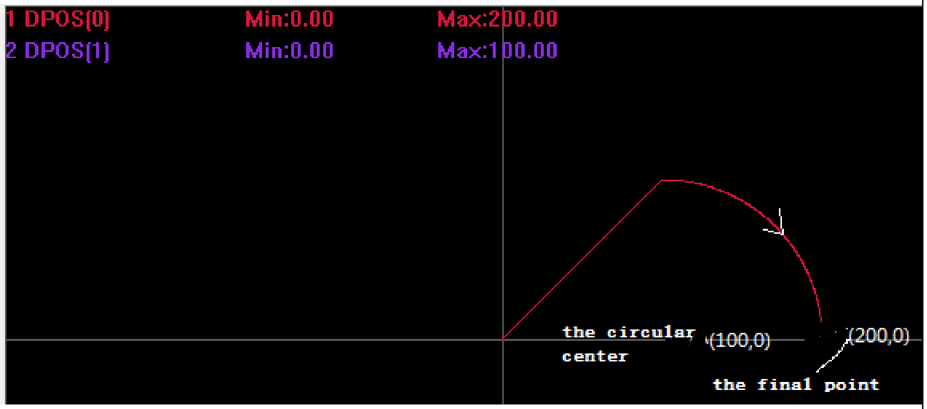
description	Two-Axis Circular arcs interpolation , circular center drawing arcs,relevant motion. BASE the first axis and the second axis operate the circular arcs interpolation,relevant move mode,when the distance of target is 0, it is a globalcircle. The continuous interpolation motion of Self-defined speed can use SP suffix instruction,see the description of *SP. Need to require the relevant initial point of circular center and the final point of circular arc. Make sure the coordinate position is correct,or practical motion trajectory maybe wrong.  Assuming that the coordinate of initial point A(100,100),circular center C（400,100）,final point B（400,400）. The coordinate of Circular center C relative to initial point is （300,0）,the coordinate of final point B relative to initial point A is （300,300）.
grammar	MOVECIRC(end1, end2, centre1, centre2, direction) parameter： end1 The coordinate of the first axis motion of final point ,relative to initial point. end2 The coordinate of the second axis motion of final point ,relative to initial point. centre1 The coordinate of the first axis motion of Circular center ,relative to initial point. centre2 The coordinate of the second axis motion of Circular center ,relative to initial point. direction 0-counterclockwise,-1clockwise
Application controller	general use

example	BASE(0,1)
	DPOS=0,0
	SPEED=100,100

	MOVE(100,100) Motive to site 100,100 MOVECIRC(200,0,100,0,0) Radius100 draws half circle in a counterclockwise direction, the coordinate of final point （300,100） interpolation trajectory  MOVECIRC(0,0,100,0,0) Radius 100 draws circle interpolation trajectory 
relative instruction	MOVECIRC，MOVECIRC2，*SP

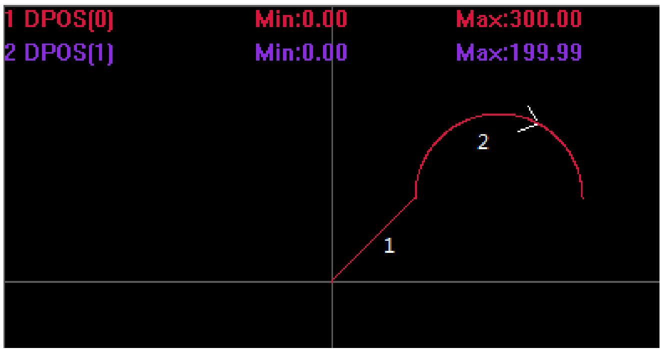
MOVECIRCABS

type	multi-motion instruction
description	Two-Axis Circular arcs interpolation，circular center drawing arcs,relevant motion. BASE the first axis and the second axis operate the circular arcs interpolation,absolute move mode. Can’t draw the full circle, use relevant MOVECIRC circular arc to solve the problem. The continuous interpolation motion of Self-defined speed can use SP suffix instruction,see the description of *SP.

grammar	MOVECIRCABS(end1, end2, centre1, centre2, direction) parameter : end1 The coordinate of the first axis motion of final point ,absolute position. end2 The coordinate of the second axis motion of final point ,absolute position. centre1 The coordinate of the first axis motion of Circular center ,absolute position. centre2 The coordinate of the second axis motion of Circular center, absolute position. direction 0-counterclockwise,-1clockwise Make sure the coordinate position is correct,or practical motion trajectory maybe wrong.
Application controller	general use
example	BASE(0,1) DPOS=0,0 SPEED=100,100 MOVE(100,100) Motive to site 100,100 MOVECIRCABS(200,0,100,0,1) Radius100 draws 1/4 circle in a clockwise direction, the coordinate of final point (200,0) interpolation trajectory 
relative instruction	MOVECIRC,MOVECIRC2ABS,*SP

MOVECIRC2

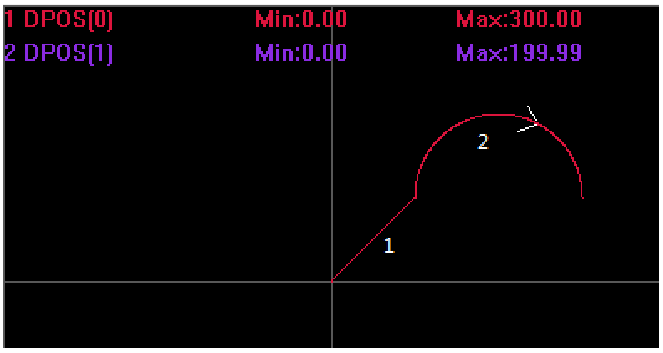
type	multi-axis motion instruction
description	Two-Axis Circular arcs interpolation , three points drawing arcs,relevant motion.

	BASE the first axis and the second axis operate the circular arcs interpolation,relevant move mode,which relative to the distance of the initial point. The continuous interpolation motion of Self-defined speed can use SP suffix instruction,see the description of *SP. Notice : the instruction can't be appliedto operate full circle interpolation motion,use relevant MOVECIRC circular arc to draw full circle or continue two similar instruction.
grammar	MOVECIRC2(mid1, mid2, end1, end2) parameter : mid1 The coordinate of the first axis motion of the intermediate point ,relative to initial point. mid2 The coordinate of the second axis motion of the intermediate point,relative to initial point. end1 The coordinate of the first axis motion of the final point ,relative to initial point. end2 The coordinate of the second axis motion of the final point, relative to initial point. Make sure the coordinate position is correct,or practical motion trajectory maybe wrong.
Application controller	general use
example	BASE(0,1) DPOS=0,0 SPEED=100,100 MOVE(100,100) 'Motive to site 100,100 MOVECIRC2(100,100,200,0) 'Three points draw half circle, relevant coordinate interpolation trajectory 
relative instruction	MOVECIRC2ABS , MOVECIRC , *SP

MOVECIRC2ABS

type	multi-axis motion instruction
description	Two-Axis Circular arcs interpolation , three points drawing arcs,absolute motion. BASE the first axis and the second axis operate the circular arcs interpolation,absolute motion mode. The continuous interpolation motion has a self-defined speed ,it can use SP suffix instruction,see the description of *SP. Notice : the instruction can't be appliedto operate full circle interpolation motion,use relevant MOVECIRC circular arc to draw full circle or continue two similar instruction.
grammar	MOVECIRC2ABS(mid1, mid2, end1, end2) parameter : mid1 The coordinate of the first axis motion of the intermediate point ,absolute position mid2 The coordinate of the second axis motion of the intermediate point,absolute position. end1 The coordinate of the first axis motion of the final point ,absolute position. end2 The coordinate of the second axis motion of the final point,absolute position. Make sure the coordinate position is correct,or practical motion trajectory maybe wrong.
Application controller	general use

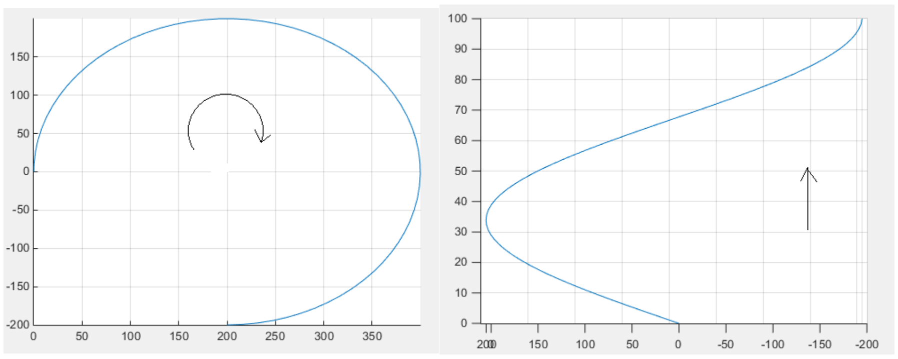
example	BASE(0,1)
	DPOS=0,0
	SPEED=100,100
	MOVE(100,100) 'Firstly motive to site 100,100
	MOVECIRC2ABS(200,200,300,100) Three points draw half circle, absolute coordinate
	interpolation trajectory

	 1 DPOS(0) Min:0.00 Max:300.00 2 DPOS(1) Min:0.00 Max:199.99 1 2
relative instruction	MOVECIRC2 , MOVECIRCABS , *SP

MHELICAL

type	multi-axis motion instruction
description	Helix Interpolation,circular center drawing arcs,relative movement The first axis and the second axis of BASE operate the circular arcs interpolation,the third axis operates helix interpolation,relative to initial point. The continuous interpolation motion has a self-defined speed ,it can use SP suffix instruction,see the description of *SP. Able to draw a full helical path which seems like a circle from the direction of Z Axis
grammar	MHELICAL(end1,end2,centre1,centre2,direction,distance3,[mode]) parameter : end1 The coordinate of the first axis motion of the final point ,relative to the initial point. end2 The coordinate of the second axis motion of the final point,relative to the initial point. centre1 The coordinate of the first axis motion of Circular center ,relative to the initial point. centre2 The coordinate of the second axis motion of Circular center,relative to the initial point. direction 0-counterclockwise,-1clockwise distance3 The motion distance of the third axis. mode The velocity calculation of the third axis: value | description 0(default) | participate the velocity calculation of the third axis

	1 — Not participate the velocity calculation of the third axis Make sure the coordinate position is correct,or practical motion trajectory maybe wrong.
Application controller	general use

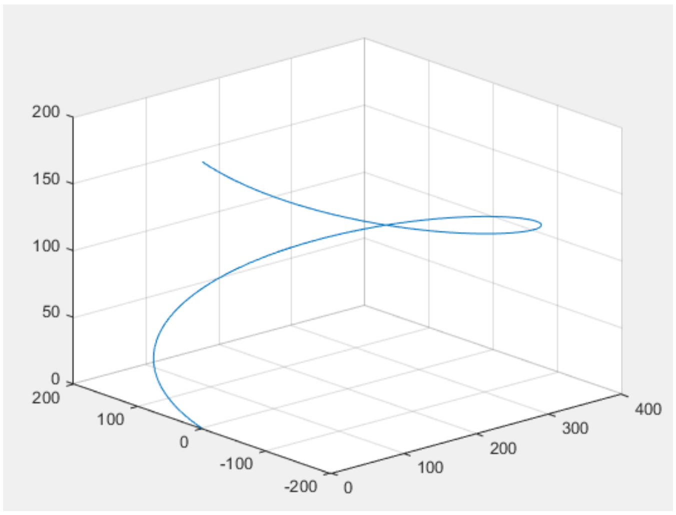
example	BASE(0,1,2)
	DPOS=0,0,0
	SPEED=100,100,100
	MHELICAL(400,0,200,0,1,100) 'Take the origin point as starting point, the center point(200,0),the final point(200,-200),clockwise,axis Zparticipate velocity calculation, motive 100.
	interpolation trajectory
	

relative instruction	MHELICAL2，MHELICALABS，*SP

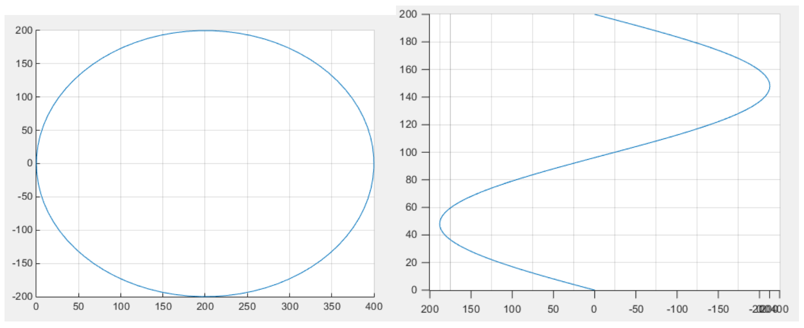
MHELICALABS

type	multi-axis motion instruction
description	Helical Interpolation,circular center drawing arcs,absolute movement .BASE the first axis and the second axis operate the circular arcs interpolation,the third axis operates helix interpolation,absolute motion mode. The continuous interpolation motion has a self-defined speed ,it can use SP suffix instruction,see the description of *SP. Able to draw a full helical path which seems like a circle from the direction of Z Axis
grammar	MHELICALABS(end1,end2,centre1,centre2,direction,distance3,[mode]) parameter： end1 标 the coordinate of the first axis

	motion end2 the coordinate of the second axis motion centre1 the Circular center of the first axis motion centre2 the Circular center of the second xis motion direction 0-counterclockwise,-1clockwise distance3 the coordinate of the third axis motion mode The velocity calculation of the third axis: Make sure the coordinate position is correct,or practical motion trajectory maybe wrong.
Application controller	general use

example	BASE(0,1,2)
	DPOS=0,0,0
	SPEED=100,100,100
	MHELICALABS(0,0,200,0,1,100) Take the origin point as starting point, the center point(200,0),the final point(200,-200),clockwise,axis Z participates velocity calculation, motive 200.
	interpolation trajectory
	

Value	description
0(Default)	participate the velocity calculation of the third axis
1	Not participate the velocity calculation of the third axis

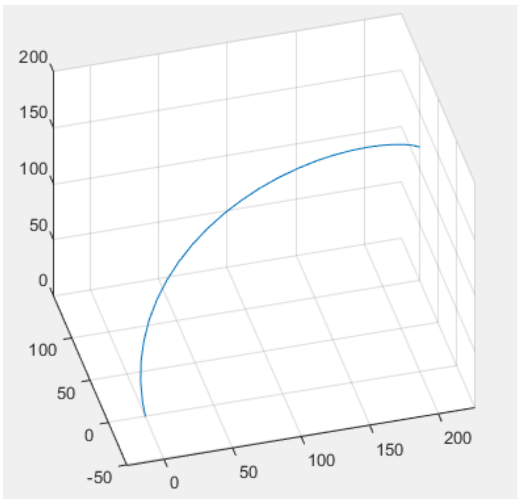
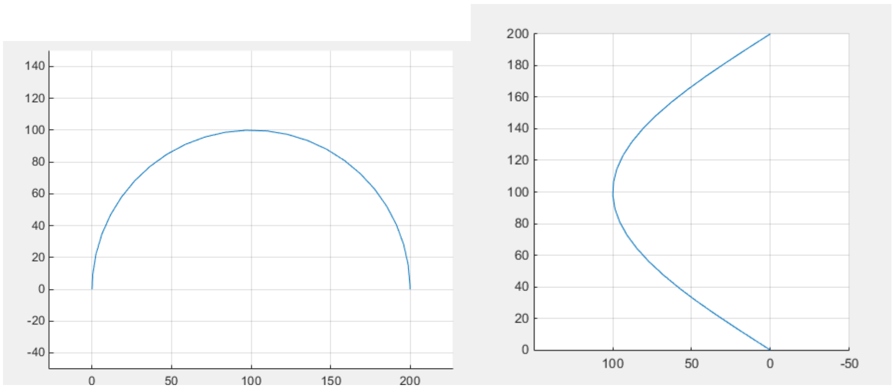
	
relative instruction	MHELICAL，MHELICAL2ABS，*SP

MHELICAL2

type	multi-axis motion instruction
description	Helical Interpolation，three points drawing arcs,absolute motion. BASE the first axis and the second axis operate the circular arcs interpolation,relative motion mode. The continuous interpolation motion has a self-defined speed ,it can use SP suffix instruction,see the description of *SP. Unable to draw a full helical path, use MHELICAL 或 MHELICALABS to solve the problem.
grammar	MHELICAL2(mid1, mid2, end1, end2, distance3,[mode]) parameter： mid1 The coordinate of the first axis motion of the intermediate point ,relative to initial point. mid2 The coordinate of the second axis motion of the intermediate point,relative to initial point. end1 The coordinate of the first axis motion of the final point ,relative to initial point. end2 The coordinate of the second axis motion of the final point, relative to initial point. distance3 The motion distance of the third axis,relative to initial point. mode velocity calculation of the third axis:

Value	description
0(Default)	participate the velocity calculation of the third axis
1	Not participate the velocity calculation of the third axis

	Make sure the coordinate position is correct,or practical motion trajectory maybe wrong.
Application controller	general use

example	BASE(0,1,2)
	DPOS=0,0,0
	SPEED=100,100,100
	MHELICAL2(100,100,200,0,200) Take the origin point as starting point, the center point(100,100),the final point(200,0),axis Z participates velocity calculation, motive 200 interpolation trajectory
	

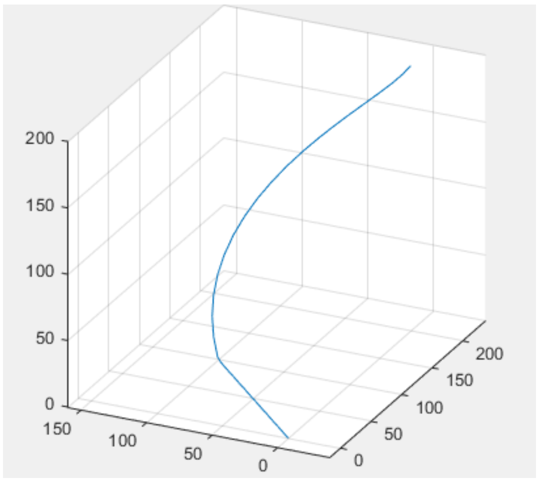
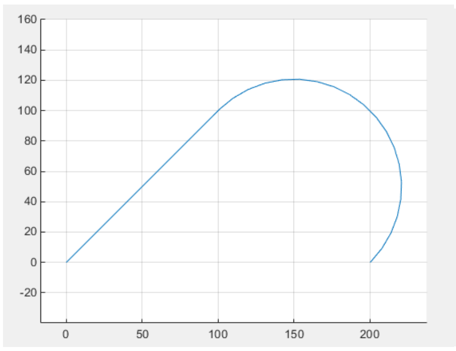
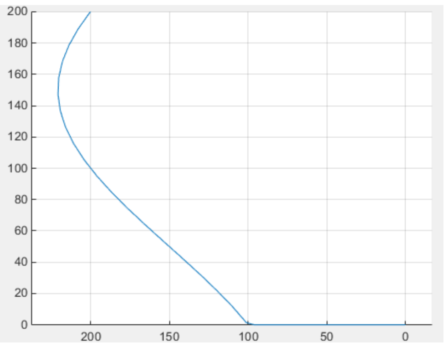
relative instruction	MHELICAL2ABS，MHELICAL，*SP

MHELICAL2ABS

type	multi-axis motion instruction
description	Helical Interpolation，three points drawing arcs,absolute

	motion. BASE the first axis and the second axis operate the circular arcs interpolation,relative motion mode. The continuous interpolation motion has a self-defined speed ,it can use SP suffix instruction,see the description of *SP. Unable to draw a full helical path, use MHELICAL or MHELICALABS to solve the problem.
grammar	MHELICAL2ABS(mid1, mid2, end1, end2, distance3,[mode]) parameter : mid1 The coordinate of the first axis motion of the intermediate point. mid2 The coordinate of the second axis motion of the intermediate point. end1 The coordinate of the first axis motion of the final point. end2 The coordinate of the second axis motion of the final point. distance3 The coordinate of the third of the final point,correction:the parameter is wrong in the version before 20150306, suggest to use relative instruction MHELICAL2mode The velocity calculation of the third axis: <table><tr><td>Value</td><td>description</td></tr><tr><td>0(Default)</td><td>The velocity calculation of the third axis:</td></tr><tr><td>1</td><td>Not participate the velocity calculation of the third axis</td></tr></table> Make sure the coordinate position is correct,or practical motion trajectory maybe wrong.
Application controller	general use

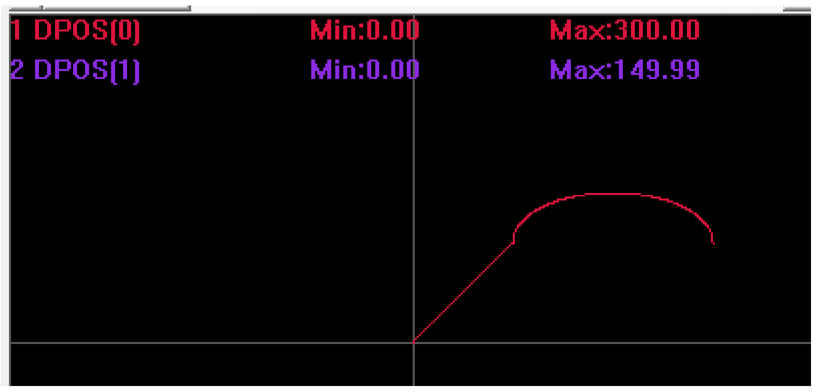
example	BASE(0,1,2)
	DPOS=0,0,0
	SPEED=100,100,100
	MOVE(100,100) ‘ Motive to (100,100)
	MHELICAL2ABS(200,100,200,0,200)
	The starting point(100,100), the center point(200,100),the final point(200,0),axis Z participates velocity calculation, motive 200 interpolation trajectory

	  
relative instruction	MHELICAL2，MHELICALABS，*SP

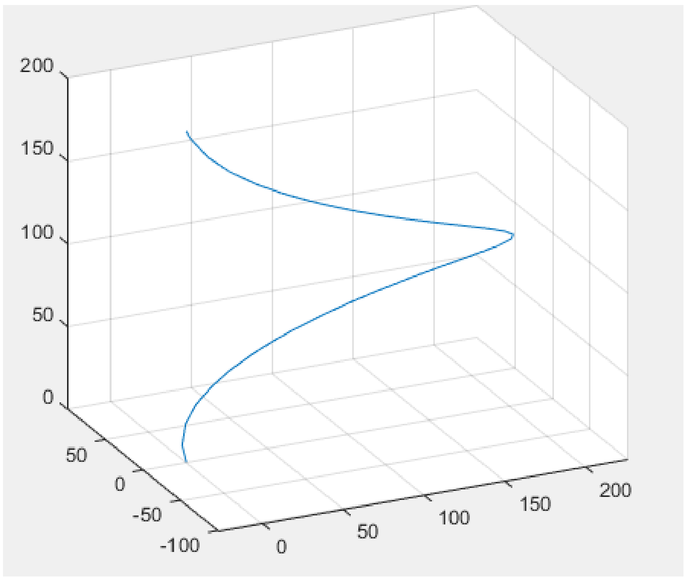
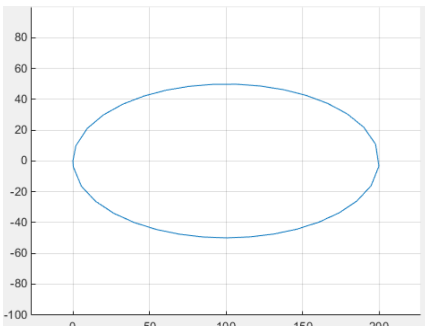
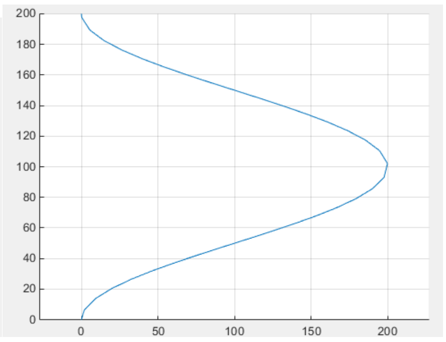
MECLIPSE

type	multi-axis motion instruction
description	Ellipse interpolation, center draws arc,relative motion, optional helix. BASE the first axis and the second axis operate ellipse interpolation,relative motion mode,able to choose the third axis helix simultaneously. The continuous interpolation motion has a self-defined speed ,it can use SP suffix instruction,see the description of *SP. Able to draw ellipse.
grammar	MECLIPSE (end1, end2, centre1, centre2, direction, adis, bdis[, end3]) parameter： end1 The coordinate of the first axis motion of final point ,relative to initial point. end2 The coordinate of the second axis motion of final point ,relative to initial point. centre1 The coordinate of the first axis motion of center

	,relative to initial point. centre2 The coordinate of the second axis motion of center ,relative to initial point. direction 0-counterclockwise,-1clockwise Value / description: 0 = counterclockwise; 1 = clockwise adis Elliptic radius of the first axis,the semi-major axis or the semi-minor axis is ok. bdis Elliptic radius of the second axis,the semi-major axis or the semi-minor axis is ok, when A equals to B, it can be regarded as helix or circular arc automatically end3 The distance of the third axis motion,need to fill helix.. Make sure the coordinate position is correct,or practical motion trajectory maybe wrong.
Application controller	general use

example	example one operate no helix
	BASE(0,1,2)
	DPOS=0,0,0
	SPEED=100,100,100
	MOVE(100,100)
	MECLIPSE(200,0,100,0,1,100,50) Center(200,100),final point(300,100),the semi-minor axis is 50,the semi-major axis is 100,draw half elliptic arc having no helix.
	interpolation trajectory
	
	example two operate helix
	BASE(0,1,2)

Value	description
0	counterclockwise
1	clockwise

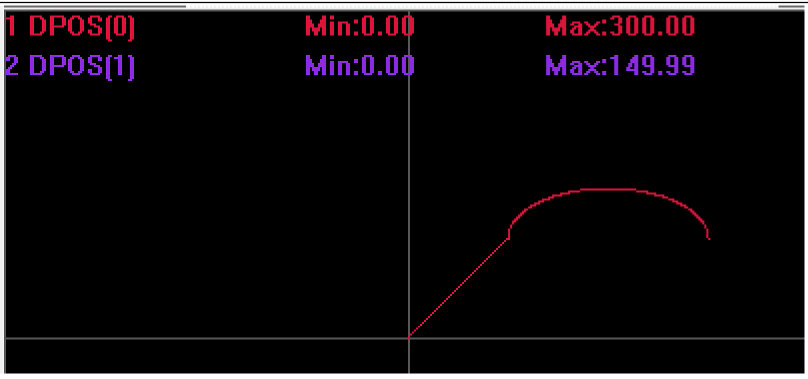
	DPOS=0,0,0 SPEED=100,100,100 MECLIPSE(0,0,100,0,1,100,50 , 200) Center(200,100),final point(300,100),the semi-minor axis is 50,the semi-major axis is 100,draw half elliptic arc and have helix. interpolation trajectory 
relative instruction	MECLIPSEABS , *SP

MECLIPSEABS

type	multi-axis motion instruction
description	Ellipse interpolation, center draws arc,absolute motion, optional helix. BASE the first axis and the second axis operate ellipse interpolation,relative motion mode,able to choose the third axis helix simultaneously. The continuous interpolation motion has a self-defined speed ,it can use SP suffix instruction,see the description of *SP.

	Able to draw ellipse.
grammar	MECLIPSEABS(end1, end2, centre1, centre2, direction, adis, bdis[, end3]) parameter : end1 The coordinate of the first axis motion of final point . end2 The coordinate of the second axis motion of final point . centre1 The coordinate of the first axis motion of center. centre2 The coordinate of the second axis motion of center . direction 0-counterclockwise,-1clockwise Value | description 0 | counterclockwise 1 | clockwise adis Elliptic radius of the first axis,the semi-major axis or the semi-minor axis is ok. bdis Elliptic radius of the second axis,the semi-major axis or the semi-minor axis is ok, when A equals to B, it can be regarded as helix or circular arc automatically. end3 The distance of the third axis motion,need to fill helix. Make sure the coordinate position is correct,or practical motion trajectory maybe wrong.
Application controller	general use

example	Example one operate no helix
	BASE(0,1,2)
	DPOS=0,0,0
	SPEED=100,100,100
	MOVE(100,100)
	MECLIPSEABS(300,100,200,100,1,100,50)
	'Center(200,100),final point(300,300),the semi-minor axis is 50,the semi-major axis is 100,draw half elliptic arc and have no helix .
	
	interpolation trajectory

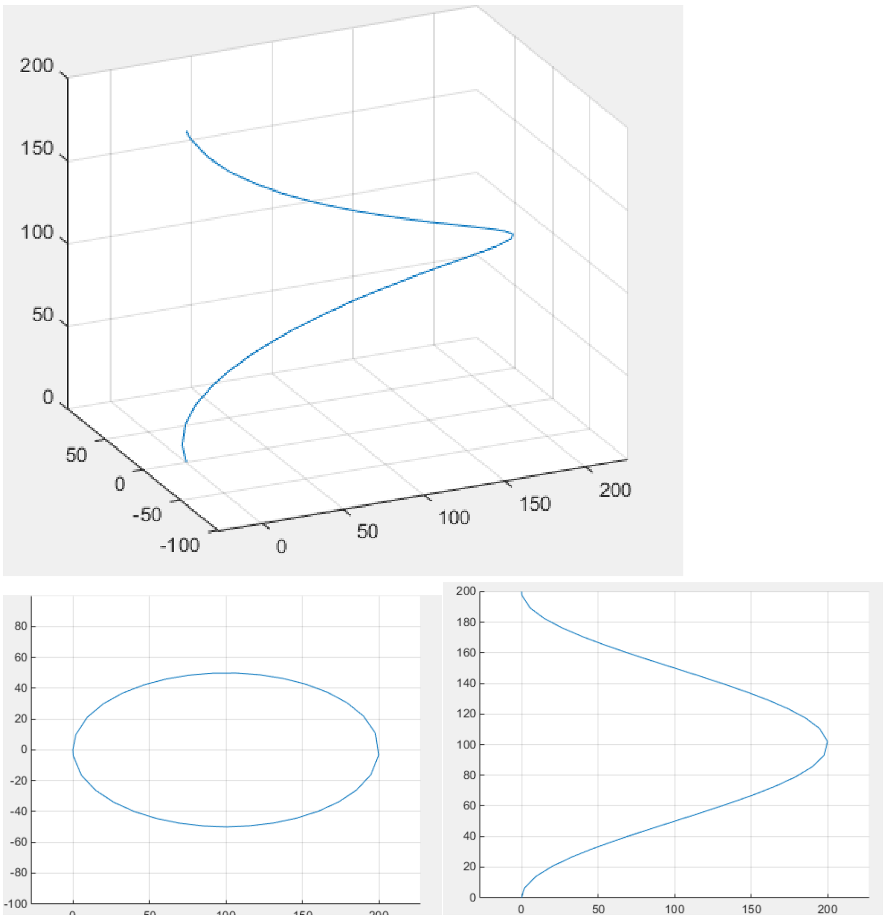


Example two operate helix

```
BASE(0,1,2)
DPOS=0,0,0
SPEED=100,100,100
```

MECLIPSEABS(0,0,100,0,1,100,50 , 200) Center(100,0),final point(0,0),the semi-minor axis is 50,the semi-major axis is 100,draw full elliptic arc and have helix simultaneously.

interpolation trajectory



relative instruction	MECLIPSE，*SP

MSPHERICAL

type	multi-axis motion instruction
description	Space arc interpolation motion,relative motion mode,optional helix. The continuous interpolation motion has a self-defined speed ,it can use SP suffix instruction,see the description of *SP
grammar	MSPHERICAL(end1,end2,end3,centre1,centre2,centre3,mode[,distance4][,distance5]) parameter： end1 The first axis motion distance parameter1 end2 The second axis motion distance parameter1 end3 The third axis motion distance parameter1 centre1 The first axis motion distance parameter2 centre2 The second axis motion distance parameter2 centre3 The third axis motion distance parameter2 mode appoint the the meaning of the former parameter.

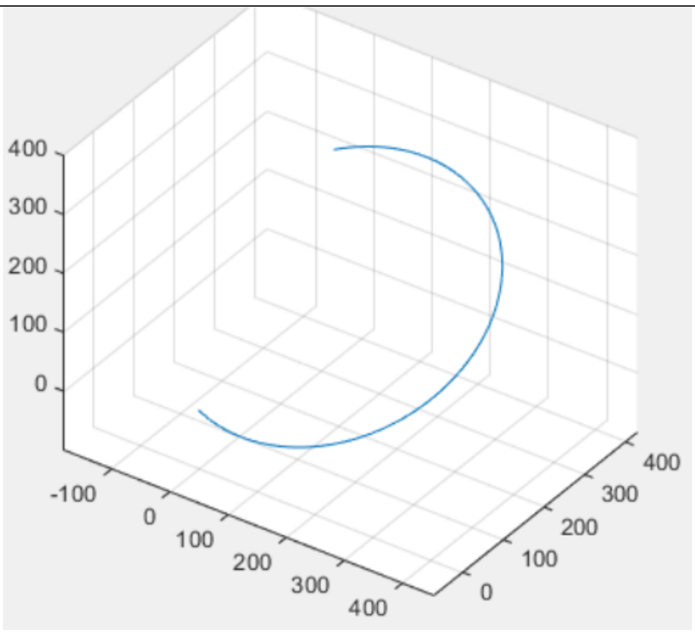
value	description
0	The current point, the intermediate point and final point determine the circle arc. Distance parameter1is final distance,distance parameter2is intermediate point distance.
1	The current point, the center point and final point determine the circle arc. Take the shortest circle arc. Distance parameter1is final distance,distance parameter2is center point distance.
2	The current point, the intermediate point and final point determine the circle. Distance parameter1is final distance,distance parameter2is intermediate point distance.
3	The current point, the center point and final point determine the circle. Take the shortest circle arc, then continue to finish the globalcircle. Distance parameter1is final distance,distance parameter2is center point distance.

Distane4 The function of the fourth axis helix,appoint the fourth axis relative distance, the axis doesn't participate velocity calculation.

Distane5 The function of the fifth axis helix,appoint the fifth axis relative distance, the axis doesn't participate velocity calculation.

Make sure the coordinate position is correct,or the practical motion trajectory will go wrong.

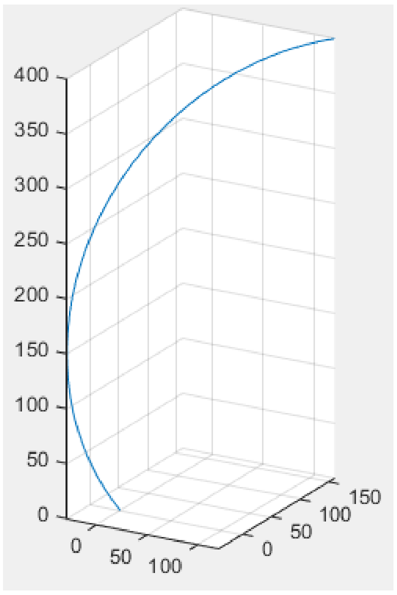
Application controller	general use
example	BASE(0,1,2) DPOS=0,0,0 SPEED=100 Use center point(120,160,150),the circle whose radius is 250 to demonstrate the motion trajectory of the four modes. **mode 0** MSPHERICAL(120,160,400,240,320,300,0) The final point(120,160,400), the intermediate point (240,320,300),mode0,three points draw circle. interpolation trajectory



mode 1

MSPHERICAL(120,160,400,240,320,300,0) The final point(120,160,400), the center point (120,160,150),mode0, take the shortest circle arc

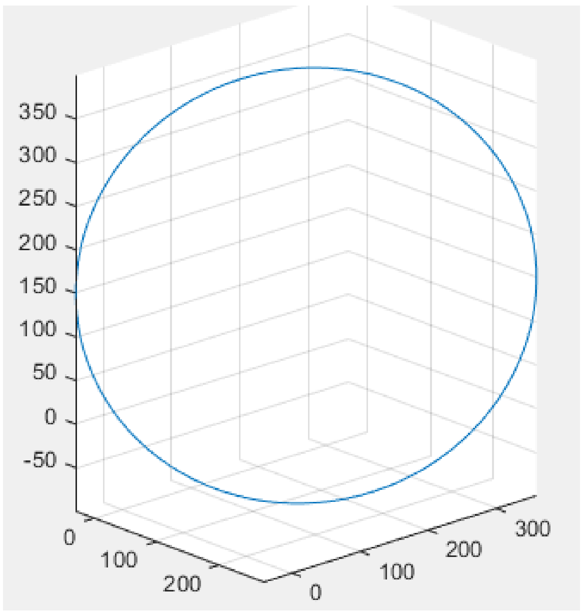
interpolation trajectory



mode 2

MSPHERICAL(120,160,400,240,320,300,2) The final point(120,160,400), the intermediate point (240,320,300),mode2,three points draw circle.

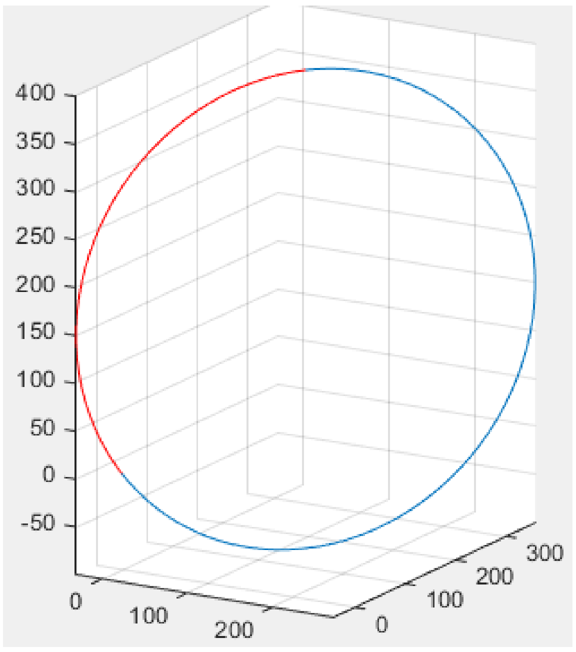
interpolation trajectory



mode 3

MSPHERICAL(120,160,400,120,160,150,3) 'The final point(120,160,400), the center point (120,160,150),mode3, take the shortest arc(red local),then finish the global circle.

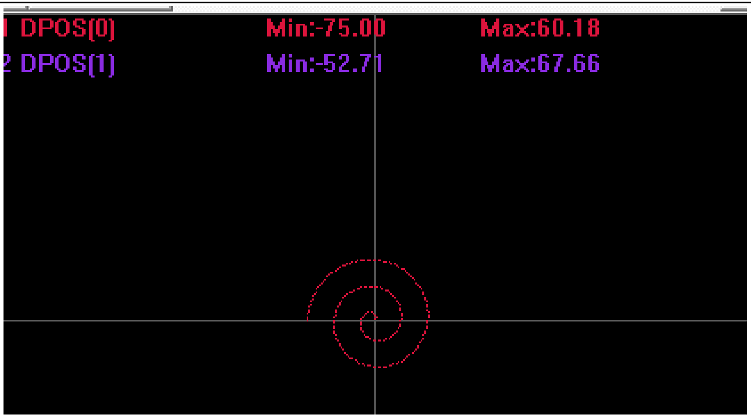
interpolation trajectory



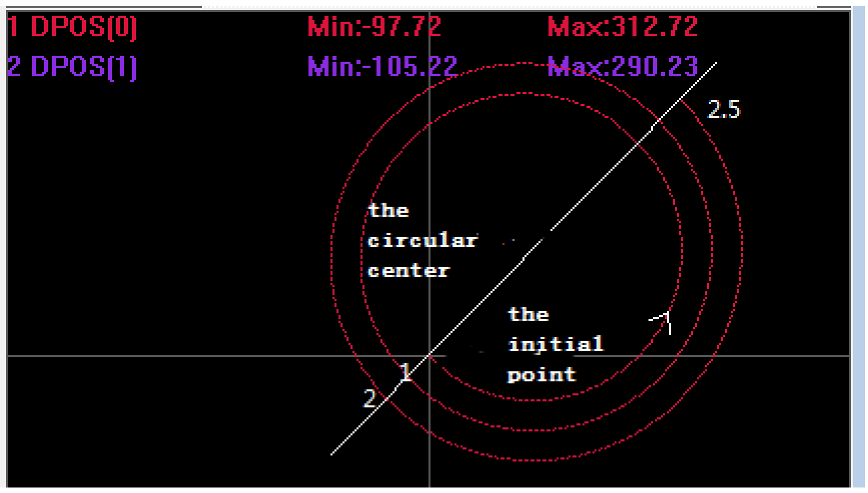
MOVESPIRAL

type	multi-axis motion instruction
description	involute curve interpolation motion,relative motion mode,optional helix. The current point and center point determine initial radius,when initial radius is 0,angle can't be determined, start from angle 0 directly.see example one. The continuous interpolation motion has a self-defined speed ,it can use SP suffix instruction,see the description of *SP.
grammar	MOVESPIRAL (centre1, centre2, circles, pitch [,distance3][,distance4]) parameter : centre1: The first axis relative distance of center centre2: The second axis relative distance of center circles: the number of circles, can be a small number of circles ,negative number stands clockwise, the final position of each circle lies in the point where the initial point and the center intersect. pitch: the diffusion distance of each circle,it can be negtive distance3 The helix function of the third axis, appoint the relative distance of the third axis, the axis doesn't participate velocity calculation. distance4 The helix function of the fourth axis, appoint the relative distance of the fourth axis, the axis doesn't participate velocity calculation.
Application controller	general use

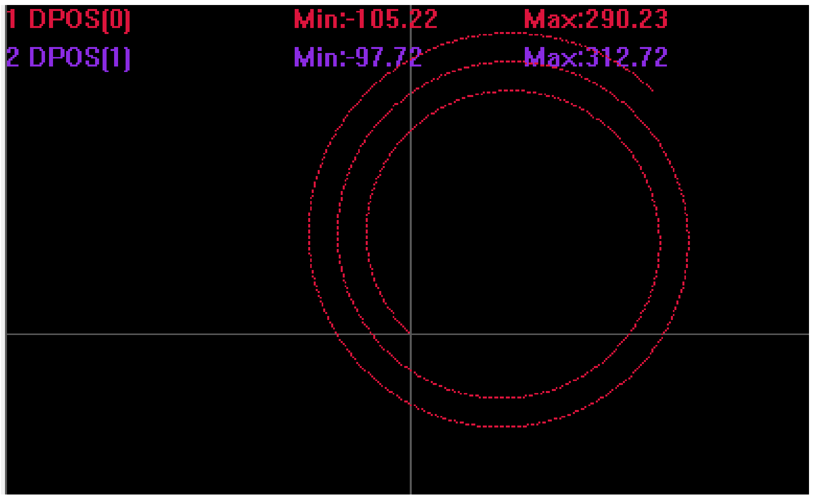
example	BASE(0,1,2)
	DPOS=0,0,0
	SPEED=100
	
	Example one helix from the initial point
	MOVESPIRAL(0,0,2.5,30) Currently,take the initial point as center ,rotate 2.5 circles in counterclockwise direction,each circle diffuses 30.
	interpolation trajectory

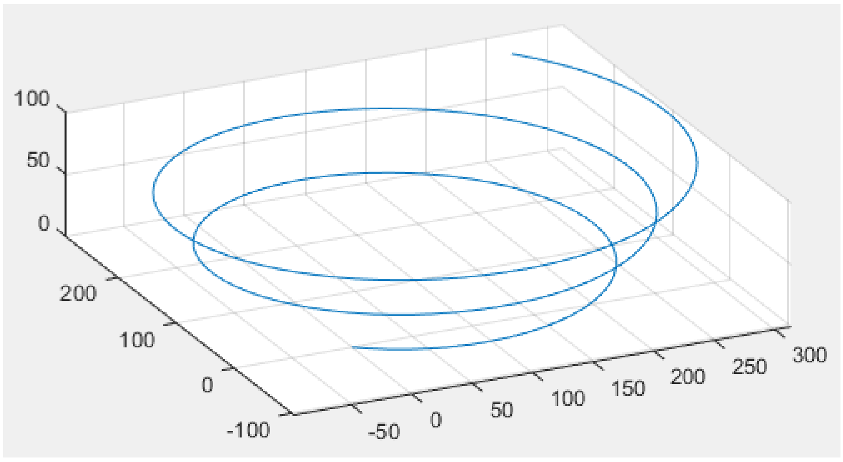


Example two operate no helix
MOVESPIRAL(100,100,2.5,30) Initial radius is 100,take point(100,100)as center point,rotate 2.5 circles in counterclockwise direction,each circle outward diffuses 30.
interpolation trajectory



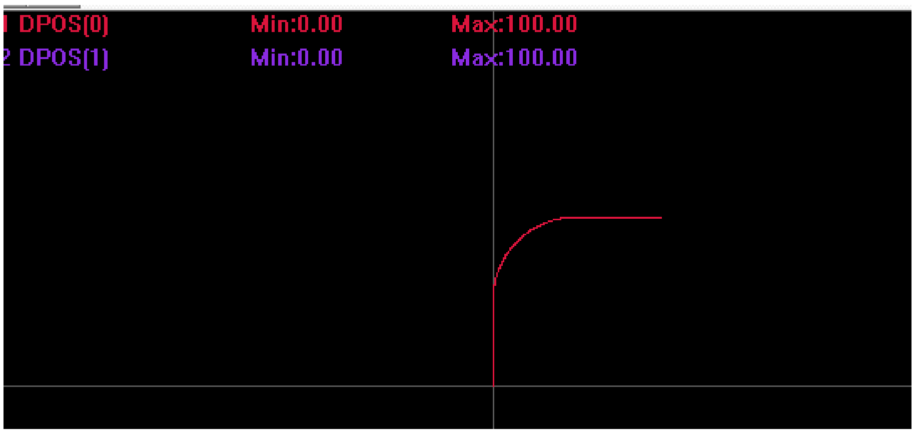
The number of rotation is negative number(-2.5), rotate in clockwise.



	Example three helix MOVESPIRAL(100,100,2.5,30,100) Initial radius is 100,take point(100,100)as center point,rotate 2.5 circles in counterclockwise direction,each circle outward diffuses 30,and axis Z motives upward to 100 . interpolation trajectory 

MOVESMOOTH

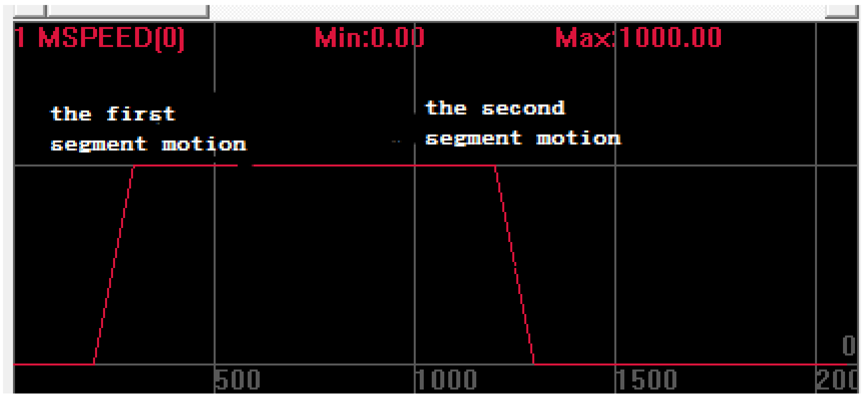
type	multi-axis motion instruction The instruction is developed in the early, have limitation in the axis number, you'd better use CORNER_MODE instruction.
description	**Spatial linear fillet motion** According to the absolute coordinate of the next linear motion,insert arc in the corner automatically,which makes the final point of motion and the final point of line inconsistent. The continuous interpolation motion has a self-defined speed ,it can use SP suffix instruction,see the description of *SP.
grammar	MOVESMOOTH (end1, end2, end3, next1, next2, next3, radius) parameter : end1 the absolute coordinate of the first axis motion. end2 the absolute coordinate of the second axis motion. end3 the absolute coordinate of the third axis motion. next1 the absolute coordinate of the first axis next linear motion. next2 the absolute coordinate of the second axis next linear motion. next3 the absolute coordinate of the third axis next linear motion.

	radius Insert the radius of arc,when the arc is too lager,it will reduce automatically.
Application controller	general use
example	BASE(0,1,2) MOVESMOOTH (0,100,0,100,100,0,50) after inserting arc, actually,it motives to (50,100,0). MOVEABS(100,100,0) interpolation trajectory 

MOVELIMIT

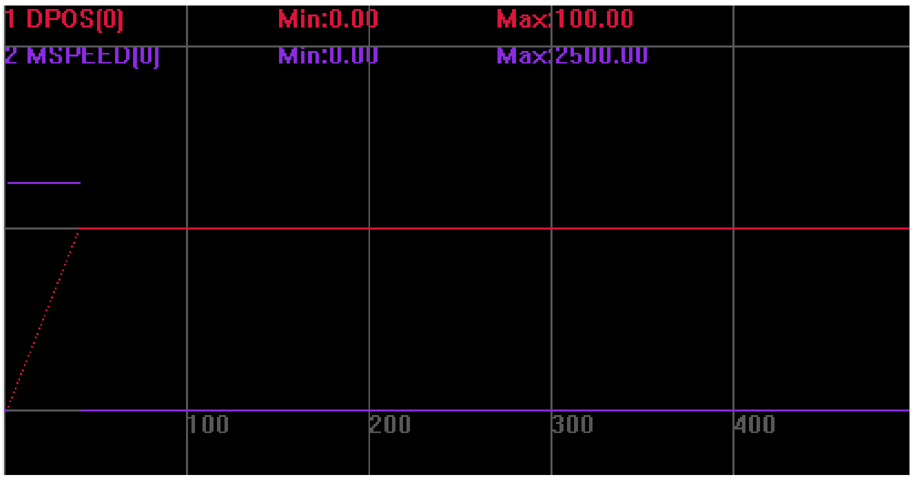
type	special motion instruction
description	**Add speed limitation in the end of the current motion to reduce speed in corner forcibly.**
grammar	MOVELIMIT(limitspeed) Limit speed
Application controller	general use

example	BASE(0)
	DPOS=0
	SPEED=1000
	MERGE=ON start continuous interpolation, Velocity Continuum during multistage motion
	MOVE(500) MOVELIMIT(100) velocity reduce to 100 between two motions
	MOVE(500)
	interpolation velocity
	when using MOVELIMIT to limit speed

	when setting no limitation in speed 
relative instruction	*SP

MOVE_PT

type	special motion instruction
description	Set distance by Drive motor motion during one time Generally, each circle of PC calculates corresponding coordinate,then send to controller. Don't motive too far within a very short time,or buffer rate will be too high,motor will be blocked,but you can decompose into small stage,and repeatedly transmit to solve above problem. The speed of motion=(DIS/TICKS)*1000 units/s
grammar	MOVE_PT (TICKS, DIS1,DIS2…) parameter : TICKS the length of time.Ticks can reduce1 continuously , 1TICKS about 1ms DIS1 the distance of motion
Application controller	general use
example	BASE(0) Dpos = 0 For i=0 to 9 MOVE_PT (4, 10) Next motive 10 units within 4tick,speed=2500 units/s print *dpos 100

	
relative instruction	MOVE_PTABS

MOVE_PTABS

type	special motion instruction
description	Drive motor motives some position during one time Generally, each circle of PC calculates corresponding coordinate,then send to controller. Don't motive too far within a very short time,or buffer rate will be too high,motor will be blocked,but you can decompose into small stage,and repeatedly transmit to solve above problem. The speed of motion=(DIS/TICKS)*1000 units/s
grammar	MOVE_PTABS (TICKS, DIS1,DIS2…) parameter： TICKS the length of time.Ticks can reduce1 continuously，1TICKS about 1ms DIS1 The absolute position of motion
Application controller	general use
example	Base(0,1) DPOS=0,0 MOVE_PTABS (3, 20,20) motive to 20,20 within 3tick print *dpos 20,20
relative instruction	MOVE_PT

MOVE_PAUSE

type	special motion instruction
description	Axis BASE motion pause It is valid only when it in single and multi-axis interpolation motion, when multi-axis movement,all motion pause together.
grammar	MOVE_PAUSE (mode) parameter： mode pause mode: 0(default) — Pause the current motion 1 — After finishing the current motion,and preparing to execute the next motion, it stops. 2 — After finishing the current motion,and preparing to execute the next motion,and the MARK of two instructions is inconsistent,it stops. The mode is applied in a motion which is finished by multiple instructions,after finishing the whole motion,it stops.
Application controller	general use
example	BASE(0) DPOS=0 SPEED=100 Example one mode 0 MOVE(1000) 'the current motion MOVEABS(-100) the buffer motion MOVE_PAUSE(0) mode0,pause the current motion ?DPOS(0) 0.02 Now ,the current motion operates for a very short time,it stops directly until scanning MOVE_PASEME. Example two mode 1 MOVE(1000) 'the current motion MOVEABS(-100) 'the buffer motion MOVE_PAUSE(1) mode1,finish the current motion then stop ?DPOS(0) 1000 Now, finish the current motion,DPOS is 1000 Example three mode 2

	MOVE_MARK=1 'set the mark as1 manually MOVE(1000) 'the current motion MOVE_MARK=1 set the same mark as the last motion MOVEABS(-100) buffer motion MOVEABS(100) Don't set the motion mark,add 1 automatically MOVE_PAUSE(2) mode2,finish the current motion then stop when the mark of the next motion instruction and current motion is inconsistent,then stop it. ?DPOS(0) -100 Now,finish the current motion,set the next motion the same mark as the current motion manually, continue to operate till the mark of the third motion is inconsistent, then stop it.
relative instruction	MOVE_MARK(axis parameter) ,MOVE_RESUME ,AXISSTATUS

MOVE_RESUME

type	special motion instruction
description	when BASE axis stops pause,continue to motive from the pause position. Can check whether there is a pause through AXISSTATUS
grammar	MOVE_RESUME
Application controller	general use
example	BASE(0) DPOS=0 SPEED=100 MOVE(100) the current motion MOVE(100) 'the buffer motion MOVE_PAUSE(1) 'finish the current motion then stop ?DPOS(0) 100 DELAY(1000) MOVE_RESUME 'continue to operate ?DPOS(0) 200
relative instruction	MOVE_PAUSE , AXISSTATUS

MOVE_OP

type	special motion instruction
description	Axis BASE motion buffer is inserted a output port operation Take no motion when operate the instruction LOAD,only operate output port. The grammar is the same as instruction OP.
grammar	MOVE_OP ([ionum],value) or MOVE_OP (ionum1, ionum2,value[,mask]) parameter : ionum Output number,0-,when there isn't the parameter,output 0-31. value Output state,when operate multiple output ports, demonstrate multiple states of port according to their position. ionum1 The first output channel that is to be operated. ionum2 The last output channel that is to be operated. mask Set value according to position,appoint which IO that need to be operated, when fill nothing,operate all channels from the first to the second.
Application controller	general use

example	BASE(0)
	DPOS=0
	SPEED=100
	MOVE(100)
	MOVE_OP (0,ON) after finishing the last motion,OUTO port outputs signal.
	MOVE_OP(1,4,15) OUTO-3 port outputs signal,15 corresponds binary1111.
	MOVE(100)

relative instruction	OP,MOVE_OP2

MOVE_OP2

type	special motion instruction
description	Axis BASE motion buffer is inserted a output port operation, pass appointed time,output state toggles. Take no motion when operate the instruction LOAD,only operate output port. The grammar is the same as instruction OP. The single axis only supports to output one buffer simultaneously,the second MOVE_OP2 instruction will close the

	buffer of the former instruction automatically.
grammar	MOVE_OP2(ionum,state,offtimems) parameter : ionum Output number,0-,when there isn't the parameter,output 0-31 value Output state offtimems Pass ms time,take rotation to produce the effect of buffer output. .
Application controller	general use

example	BASE(0)
	DPOS=0
	SPEED=100
	OP(0,OFF) close OUTO port
	MOVE(100)
	MOVE_OP2 (0,ON,100) after finishing the last motion,output port 0 outputs a buffer of 100ms,the time of buffer can't block the operation of the next motion.
	MOVE(100)

relative instruction	MOVE_OP , OP

MOVE_TABLE

type	special motion instruction
description	Axis BASE motion buffer is inserted a TABLE. Take no motion when operate the instruction LOAD,only revise TABLE. The current instruction of MTYPE and MOVE_OP is consistent.
grammar	MOVE_TABLE(tablenum, value) parameter : tablenum TABLE number Value the value need to be revised
Application controller	general use

example	BASE(0)
	DPOS=0
	SPEED=100
	TABLE(0)=0 'let TABLE(0)=0
	?TABLE(0) Print and conform the value of TABLE(0)
	0
	MOVE(100)

	MOVE_TABLE(0, 100) after finishing the last motion, valuate TABLE(0) 100 MOVE(100) WAIT IDLE ?TABLE(0) 'after Printing and revising, the value of TABLE(0) 100
relative instruction	TABLE

MOVE_TASK

type	special motion instruction
description	Axis BASE motion buffer is inserted start TASK Take no motion when operate the instruction LOAD,only revise TABLE. The current instruction of MTYPE and MOVE_OP is consistent. After starting task, there is a false alarm,but don't affect the operation of the program.
grammar	MOVE_TASK(tasknum, label) parameter : tasknum task number label function name or label name
Application controller	general use

example	BASE(0)
	DPOS=0
	SPEED=100
	MOVE(100)
	MOVE_TASK(1, task_move) after finishing the first motion,take task_move as the task1 then start.
	MOVE(100)
	
	Task_move:
	    Print "task_move"
	end

relative instruction	RUNTASK，RUN

MOVE_AOUT

type	special motion instruction

description	Axis BASE motion buffer is inserted a AUTO instruction. Take no motion when operate the instruction LOAD,only revise TABLE. The current instruction of MTYPE and MOVE_OP is consistent.
grammar	MOVE_AOUT(danum, value) parameter : danum DA number Value the value that needs to be revised
Application controller	general use,in fact,it only is applied in the controller that contains DA channel.
example	BASE(0) DPOS=0 SPEED=100 AOUT(0)=0 'valuate DA0 channel 0 ?AOUT(0) 'print and conform MOVE(100) MOVE_AOUT(0, 100) after finishing the first motion,valuate DA0 channel 100 MOVE(100) WAIT IDLE ?AOUT(0) 'print DA0 100
relative instruction	AOUT

MOVE_DELAY

type	special motion instruction
description	Axis BASE motion buffer is inserted a delay. Take no motion when operate the instruction LOAD,only delay appointed time. After finishing the former motion instruction,the speed will be reduced to 0 automatically.
grammar	MOVE_DELAY(timems) Alias：MOVE_WA(timems) parameter : timems delay, milliseconds amount
Application controller	general use

example	BASE(0)
	DPOS=0
	SPEED=100

	MOVE(100) MOVE_DELAY(1000) ‘ wait one second between two MOVE MOVE(100)
relative instruction	DELAY

MOVE_WAIT

type	special motion instruction
description	Axis BASE motion buffer is inserted a optional delay. Take no motion when operate the instruction LOAD,only delay appointed time. After finishing the former motion instruction,the speed will be reduced to 0 automatically.
grammar	MOVE_WAIT(PARANAME, paranum, eq, value) parameter : PARANAME choose the name of the parameter: DPOS MPOS IN AIN VPSPEED MSPEED MODBUS_REG MODBUS_IEEE MODBUS_BIT NVRAM VECT_BUFFED REMAIN Paranum parameter number or axis number Eq : 1 >= -1 《= It is invalid to the BIT parameter like IN. Value: compare value
Application controller	firmware above 150802 version,or above XPLC160405 version supports

example	BASE(0)
	DPOS=0
	SPEED=100
	MOVE(100)
	MOVE_WAIT(IN, 0, 0, 1) ‘wait till IN(0) has signal,then execute the next motion buffer
	MOVE(100)

relative instruction	WAIT

MOVE_TURNABS

type	multi-axis motion instruction

description	Rotating interpolation motion,make sure linear motion in rotating table.(the right hand rule) Rotating function means work platform rotates in a plane that parallels XY,rotating forward and XY's forward are consistent Rotating parameter is stored in TABLE list,storing the number of axis R ,the number of a circle buffer of axis R,the number of axis x and axis y,the center of X and Y in orderly. The continuous interpolation motion has a self-defined speed ,it can use SP suffix instruction,see the description of *SP.
grammar	MOVE_TURNABS(tablenum, position1[, position2[, position3[, position4...]]]) parameter : tablenum The table number of stored rotating parameter. position1 The coordinate of the first axis. position2 The coordinate of the next axis.
Application controller	general use
example	Base(0,1,2,3) Table(0, 3, 3600, 0,1, 0,0) Set the parameter of the rotating table Move_turnabs(0,100,200) Linear interpolation to target position Wait idle Wait till the motion stops
relative instruction	MCIRC_TURNABS

MCIRC_TURNABS

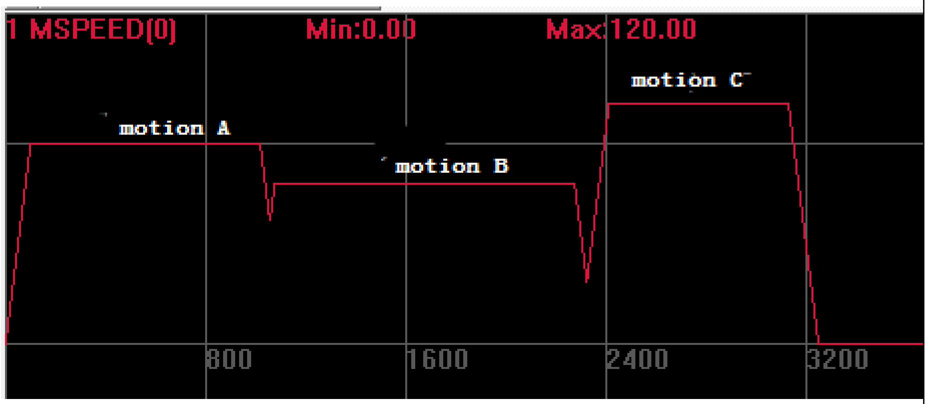
type	multi-axis motion instruction
description	Rotating interpolation motion,make sure circular motion in rotating table.(the right hand rule) Rotating function means work platform rotates in a plane that parallels XY,rotating forward and XY's forward are consistent Rotating parameter is stored in TABLE list,storing the number of axis R ,the number of a circle buffer of axis R,the number of axis x and axis y,the center of X and Y in orderly. The continuous interpolation motion has a self-defined speed ,it can use SP suffix instruction,see the description of *SP.
grammar	MCIRC_TURNABS(tablenum, refpos1, refpos2, mode,end1, end2 [, dis3, dis4, dis5])

	parameter : tablenum store the table number od rotating parameter refpos1 The table number of stored rotating parameter. refpos2 The reference point of the first axis,absolute position Mode 1-reference point before the current point,2-reference point behind the final point,3-reference point in the middle, using the three points to draw circle. end1 The final point of the first axis,absolute position end2 The final point of the second axis,absolute position dis3 The final position of the helix axis
Application controller	general use
example	Base(0,1,3) Table(0, 3, 3600, 0,1, 0,0) Set the parameter of the rotating table TURN_POSMAKE(0,100,200,5,10) MCIRC_TURNABS(0,table(10),table(11),3,200,300,10) The three axes of arc rotate simultaneously. Wait idle
relative instruction	MOVE_TURNABS，TURN_POSMAKE

*SP

type	multi-axis motion instruction.
description	speed and final speed of each segment motion. The multi-axis motion instruction has its corresponding SP motion instruction, now you can use axis parameter FORCE_SPEED and ENDMOVE_SPEED ,STRATMOVE_SPEED to set the speed and final speed of each motion,when there is no need of setting speed,there is no need of using SP instruction.
grammar	SP instruction includes：MOVESP，MOVEABSSP MOVECIRCSP, MOVECIRCABSSP ， MHELICALSP ， MHELICALABSSP ， MECLIPSESP，MECLIPSEABSSP， MSPHERICALSP
Application controller	general use
example	Example one BASE(0)

DPOS=0
MERGE=ON start continuous interpolation
SPEED=100 100 operating speed 100
FORCE_SPEED=80 limited speed 80
STARTMOVE_SPEED=60 the initial speed 60
ENDMOVE_SPEED=30 the final speed 30
MOVE(100) Motive A,needn't use SP to limit speed
MOVESP(100) Motive B,needn't use SP to limit speed
FORCE_SPEED=120 limited speed120
ENDMOVE_SPEED=30 final speed 30
MOVESP(100) Motive C, use SP to limit speed
velocity trajectory



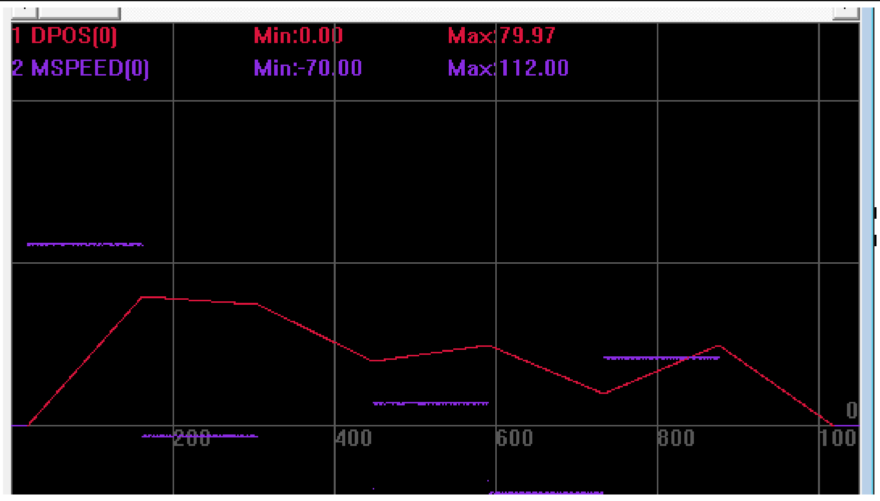
When there is no need of using instruction SP,the operating speed is SPEED,after using instruction SP,the operating speed is FORCE_SPEED.
When setting STARTMOVE_SPEED and ENDMOVE_SPEED simultaneously,STARTMOVE_SPEED takes effect.
example two
BASE(0,1)
MERGE =ON
CORNER_MODE=2+8 'Start corner deceleration and small circle limitation speed automatically.
DECEL_ANGLE = 15 * (PI/180) Set start deceleration angle
STOP_ANGLE = 45 * (PI/180) Set stop deceleration angle
FULL_SP_RADIUS = 5 set the max radius of small limitation speed Speed = 1000 ,1000
Startmove_speed = 1000 'Set a larger initial speed
Endmove_speed = 1000 Set a larger final speed
Force_speed = 50 'the speed of the each segment is limited by speed , that is set speed a larger value
Movesp(0,100) 'The speed of the segment is 50units/s
Force_speed = 70 the speed of the each segment is limited by speed , that is set speed a larger value
Movesp(100,0) The speed of the segment is 50units/s

relative instruction	FORCE_SPEED，ENDMOVE_SPEED ,STRATMOVE_SPEED，CORNER_MODE

CAM

type	simultaneous motion instruction
description	Instruction CAM determines the motion of the axis according to the dates stored in the TABLE, the corresponding position of these date value is the absolute position of the initial point of motion. Notice:two or multiple CAM instruction can use the TABLE date area in the same segment to operate. The total time of motion is determined by setting speed and the fourth parameter, match the actual speed of motion automatically,according to table trajectory and time. Table date need to be set manually,the first date is the guiding point, you'd better to set it 0. The proportion of Table date * table multiplier=the sent buffer number
grammar	CAM(start point, end point, table multiplier, distance) parameter : start point The number of the start point TABLE,stored in the position of the first point. end point The number of the end point TABLE table multiplier The position *the proportion,it is set pulse equivalent in general. distance According to the distance of the motion,calculate the total time of motion. The total time=distance/axis SPEED
Application controller	general use

example	BASE(0)
	UNITS=100
	DPOS=0
	SPEED=100
	TABLE(10,0,80,75,40,50,20,50,0) table starts to store store date from number 10
	CAM(10,17,100,500) Motion trajectory is table 10 to 17,the total time is 500/100=5s
	trajectory and velocity

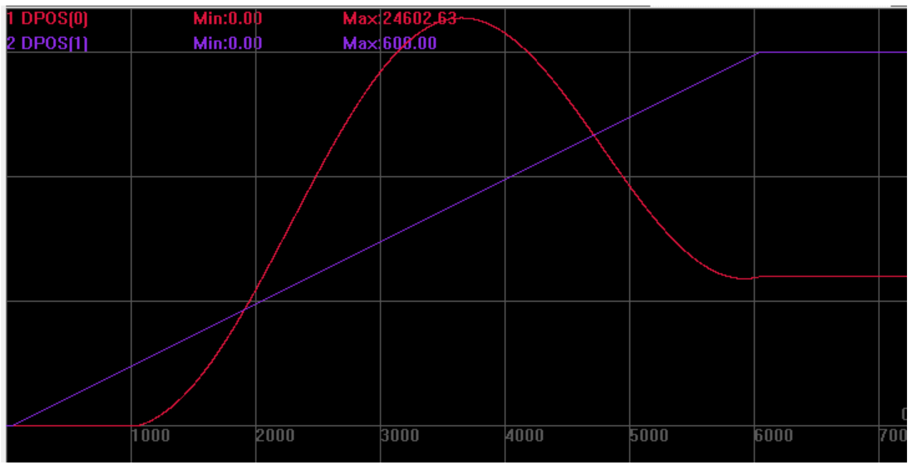
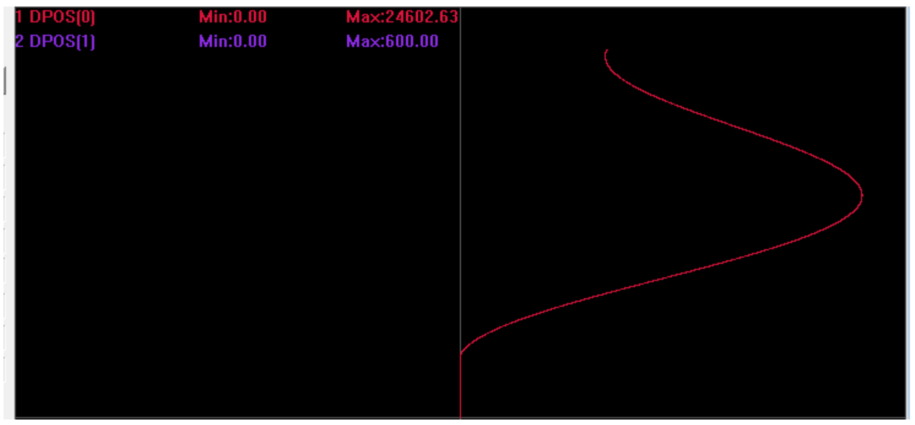
	
relative instruction	CAMBOX

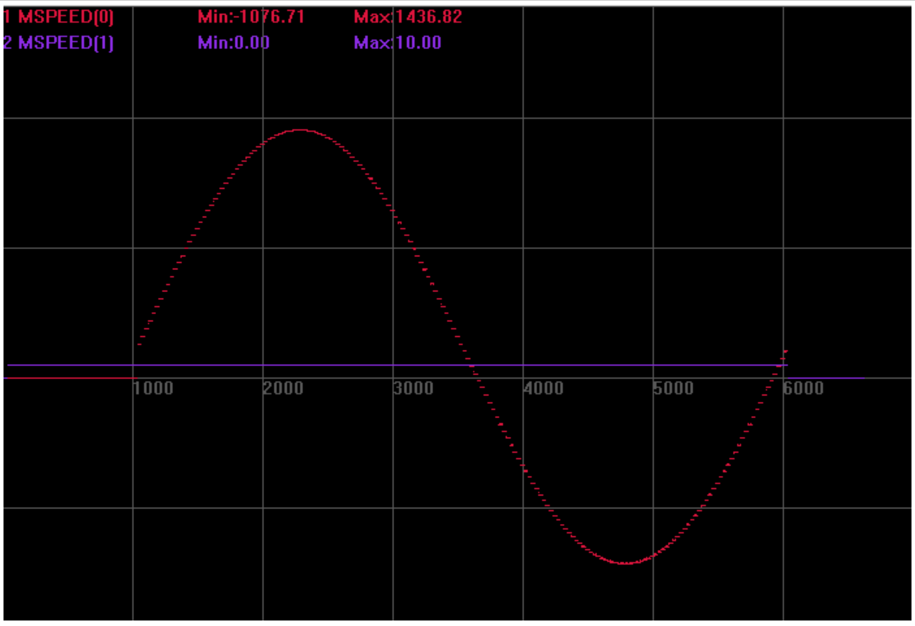
CAMBOX

type	simultaneous motion instruction
description	Instruction CAMBOX determines the motion of the axis according to the dates stored in the TABLE, the corresponding motion trajectory position of these date value is the distance of the initial point of motion. Notice:two or multiple CAMBOX instruction can use the TABLE date area in the same segment to operate. The proportion of Table date * table multiplier=the sent buffer number The total time of the motion is determined by reference motive distance and velocity of the axis, match the actual speed of motion automatically. Table date need to be set manually,the first date is the guiding point, you'd better to set it 0. The proportion of Table date * table multiplier=the sent buffer number
grammar	CAMBOX(start_point, end_point, table_multiplier, link_distance , link_axis[, link_options][, link_pos]) parameter： start_point: The number of the start point TABLE,stored in the position of the first point. end_point: The number of the end point TABLE table_multiplier: The position *the proportion,it is set pulse equivalent in general.

	link_distance: According to the distance of the axis motion,calculate the time of motion. link_axis: Reference axis link_options: The linking mode of the reference axis,different bits represent different meaning. Bit 0 When the Regist signal event of the reference axis is triggered,the current axis and reference axis start connective motion. Bit 1 When the reference axis motives to set absolute position,the current axis and reference axis start connective motion. Bit 2 Repeat continuous bi-directional motion automatically.(through setting REP_OPTION=1 to cancel repetition) Bit 5 Only the forward motion of the reference axis can connect. link_pos: When the parameter of link_options is set as 2, the parameter represents connecting the start -up absolute position .
Application controller	general use

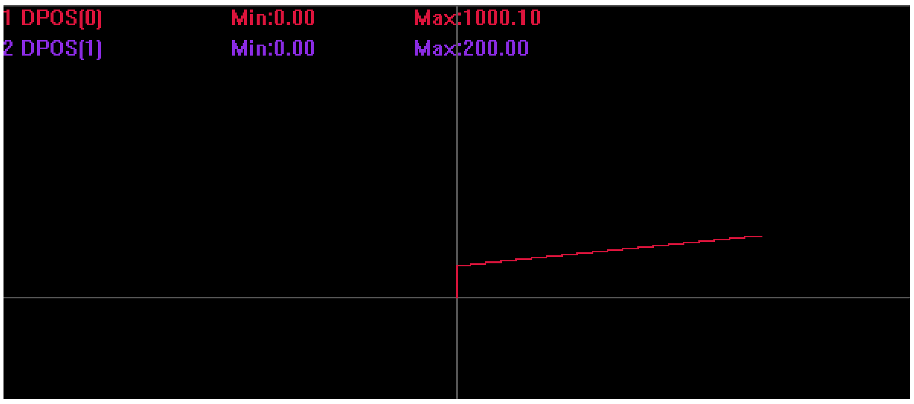
example	ERRSWITCH = 3
	
	base(0,1) Choose the 0th axis
	
	atype=1,1 ' buffer mode stepper or servo
	dpos = 0,0
	units = 100,100 pulse equivalent, each mm100 buffer
	speed = 200,200
	accel = 2000,2000
	decel = 2000,2000
	
	calculate the date of table
	dim deg, rad, x, stepdeg
	stepdeg = 2 ' Can revise segment number through this,the more segments,the more stable speed
	FOR deg=0 TO 360 STEP stepdeg
	  rad = deg * 2 * PI/360 convert to radian
	  x = deg * 25 + 10000 * (1-COS(rad))
	  TABLE(deg/stepdeg,x) 'store table
	  trace deg/stepdeg,x
	NEXT deg
	
	  while 1 ' cycle the motion

	if in(0) = on then input 0 to start motion effectively dpos = 0,0 CAMbox(0,360/stepdeg, 100, 500, 1,2,100) ' Start until axis1motives to the position 100 move(600) axis(1) wait until idle and idle(1)' wait the motion to stop delay(100) ' end if wend end ’ end the current task interpolation trajectory   velocity curve

	
relative instruction	CAM

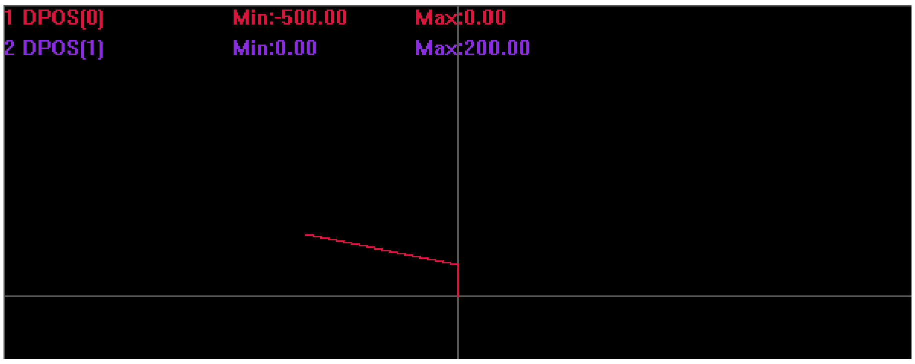
CONNECT

type	simultaneous motion instruction
description	Connect the target position of the current axis and the measurement position of driving_axis by the Electronic Gear . What connect is buffer number, so take the UNITS' proportion of different axes into consideration. Assuming,the UNIST of axis0 which is connected is 100,the UNIST of axis1 which is connected is 10. Use CONNECT to connect,ratio is 1,when axis0 motives s1=100, the motion of axis1=s1*UNITS(0)*ratio/UNITS(1). Ratio can vary dynamically by calling instruction repeatedly. Is often applied in hand wheel. Use CANCEL to cancel connection.
grammar	CONNECT（ratio，driving_axis） parameter： ratio: Ratio: can be positive or negative ,pay attention to the ratio is buffer number driving_axis: The axis number of connective

	axis, when using hand wheel, it is encoder axis
Application controller	general use
example	BASE(0,1) ATYPE=1,1 UNITS=10,100 SPEED=100,100 MOVE(100)AXIS(1) axis1 motives to 100,currently, axis0 keeps motionless WAIT ILDE(1) CONNECT(1,1)AXIS(0) When Axsi0 connects axis1,the ratio is 1 MOVE(100)AXIS(1) Axis1 motives 100,Axis0 motives 1000 composing trajectory 
relative instruction	CLUTCH_RATE，CONNPATH

CONNPATH

type	simultaneous motion instruction
description	Connect the target position of the current axis and interpolation vector length of driving_axis by the Electronic Gear . What connect is buffer number, so take the UNITS' proportion of different axes into consideration. Ratio can vary dynamically by calling instruction repeatedly. Use CANCEL to cancel connection.
grammar	CONNPATH（ratio，driving_axis） parameter： ratio: Ratio: can be positive or negative ,pay attention to the ratio is buffer number

	driving_axis: The axis number of connective axis, when using hand wheel, it is encoder axis
Application controller	general use
example	BASE(0,1) ATYPE=1,1 UNITS=10,100 SPEED=100,100 MOVE(100)AXIS(1) 'axis1 motives to 100,currently, axis0 keeps motionless WAIT ILDE(1) CONNECT(-0.5,1)AXIS(0) When Axsi0 connects axis1,the ratio is-0.5 MOVE(100)AXIS(1) 'Axis1 motives 100,Axis0 motives -500 composing trajectory 
relative instruction	CLUTCH_RATE, CONNECT

CONNFRAME

Type：simultaneous motion instruction
Grammar：CONNFRAME（frame，tablenum, viraxis0, viraxis1,[…]）
Description：Connect the target position of current joint coordinate system and the position of the virtual position;when CLUTCH_RATE=0,the motion speed and acceleration of the joint coordinate system is limited by the parameter like SPEED.

⚠ When the joint axis receives wrong warning, the motion will be CANCELED.

⚠ Don't CANCEL this motion which motives at high speed in virtual axis ,or will cause the axis motion stops.

⚠ The order LOAD can revise the coordinate of virtual axis automatically,which coincides with joint axis,so after calling the order ,need to WAIT LOSDED to start motion.

⚠ Don’t revise the coordinate of virtual axis ,or call the motion like DATUM that may change coordinate during connecting,which will cause joint axis motives to new virtual position quickly.

⚠ After CONNFRAME taking effect,the MTYPE of joint axis is 33,now can’t operate joint axis,must operate joint axis indirectly by operating virtual axis.

parameter：

frame: The type of coordinate system,1- scara ，(if need for a localicular type of Manipulator,please contact manufacturer)

tablenum: Store conversion parameter table position,when frame=1,store in sequence:the length of the first joint axis,the length of the second joint axis,the number of pulse of one circle of the first joint axis，the number of pulse of one circle of the second joint axis。viraxis0： The first axis of the virtual axis system

viraxis1： The second axis of the virtual axis system

[…] The Nth axis of the virtual axis system, the motion axis can be actual axis.

⚠When the virtual axis and the actual axis are rotating axis,such as the end rotating axis, its pulse equivalent of the virtual axis and the actual axis should be consistent.

FRAMEdescriptiondescription： (if need for a particular type of mechanical construction,please contact manufacturer)

Frame = 1 ，SCARA ， As shown,the rotating axis is joint axis,the end corresponding position is virtual position.

TABLE(tablenum, L1,L2, Pules1OneCircle, Pules2OneCircle[,Pules3OneCircle,L3][,ZDis])

store in sequence:

L1: the length of the first joint axis,L2:,the length of the second joint axis,

Pules1OneCircle:.pulse amount of one circle of the first joint axis，

Pules2OneCircle: pulse amount of one circle of the second joint axis。

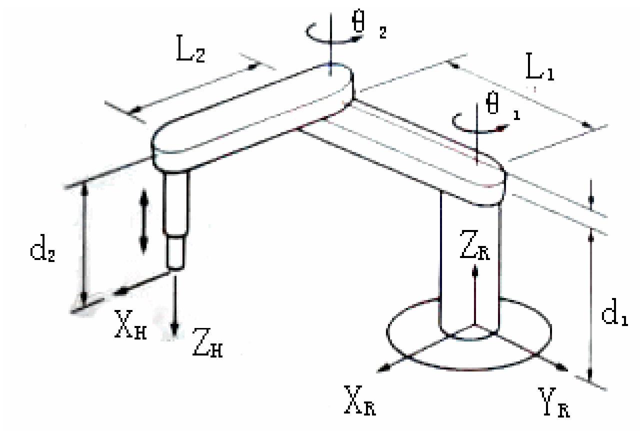
Pules3OneCircle: pulse amount of the end rotating axis W,the setting of virtual axis viraxis2 is the same as this,the parameter is optional.

L3: The shift length of the end rotating axis W,bit0,it parallels X.

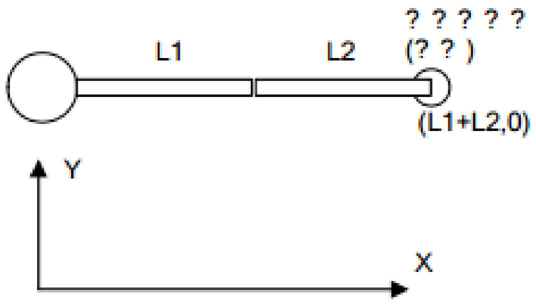
ZDis: The connective mode of rotation and axis Z, need to appoint the distance of axis Z which rotating one circle.

BASE(axis1, axis2[, axisu][, axisz])

axis1： the number of Big joint axis
axis2： the number of small joint axis
axisu： the number of axis of end rotating axis
axisz： the number of axis Z, which is need when rotation and lifting.
CONNFRAME（1 ，tablenum, viraxis0, viraxis1[,viraxis2] [,viraxis3]）
viraxis0： The number of the virtual axis X
Viraxis1： The number of the virtual axis Y
Viraxis2：the number of axis of the virtual end rotating axis,optional,when there is no need to fill, it is only appliedas a converse between two joint axes and XY.
Viraxis3： the number of the virtual axis Z, which is need when rotation and lifting.



Coordinate direction:two joint axes become a line when they are 0 point,now point to the positive direction of the virtual axis X,when joint axis is in 0 ,the coordinate of the virtual axis is（l1+l2,0）, the positive direction of two joint axes must be consistent.
When using the end rotating axis W, the positive direction of W and joint axis are consistent.As shown.



the end rotating axis（optional）:
For example,the following dates L1,L2 and equivalent and etc can be revised according to the practical situation.
L1=10’ the length of the first joint axis
L2=10’ the length of the second joint axis
BASE(0,1) ' assuming the number of the joint axis is 0/1

UNITS=10,10 here,measured in degrees
DPOS=0,0' Set the position of the joint axis,here it can be revised according to the practical situation
BASE(6,7)
ATYPE=0,0 ' set as the virtual axis
TABLE(0,L1,L2,3600,3600) 'The parameter is stored in TABLE0 start-up position, one circle of motor is 360 buffer.
UNITS=1000,1000' The buffer equivalent should be set ahead of time, can't change halfway。
BASE(0,1) ' assuming the number of the joint axis is 0/1
CLUTCH_RATE=0,0 ' Use the limitation of the speed and acceleration of the joint axis
SPEED=10000,10000' set speed limitation of the joint axis
ACCEL=100000,100000' set acceleration limitation of the joint axis
CONNFRAME(1,0,6,7) ' Take the sixth or seventh axis as the virtual axis XY ,start connect
WAIT LOADED Wait the motion loading,now adjusting the position of the virtual axis automatically.
the third step:motion virtual axis
BASE(6,7)
MOVEABS(10,10) ' the joint axis can motives to position 0,90

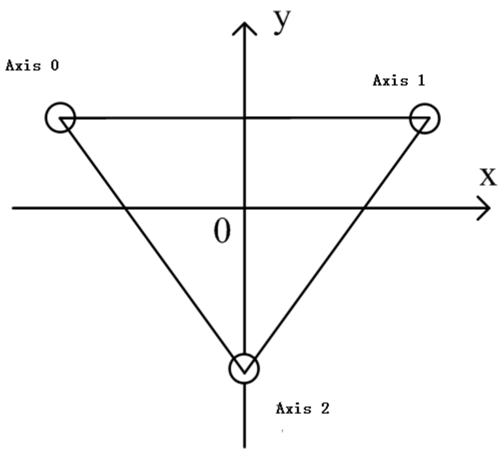
Frame = 2/12 ，Delta ，Only the type of suffix-R controller supports,2 is the mode of axis3,12 is the mode of axis4.

TABLE(tablenum, R, r, L1,L2, Pules1OneCircle,Pules2OneCircle,Pules3OneCircle, Lx, Ly, Lz, [Pules4OneCircle]))

store in sequence:
R:。The radius of up distance
r: The radius of down distance
L1: The length of up pole
L2: The length of down pole
Pules1OneCircle: Joint axis1(left upper axis) the pulse amount
Pules2OneCircle: Joint axis2(right upper axis) the pulse amount
Pules3OneCircle: Joint axis3(down axis) the pulse amount
Lx: The X shift between the end point and the down center point,(the negative direction of X is negative)
Ly: The Y shift between the end point and the down center point,(the negative direction of Y is negative)
Lz: The Z shift between the end point and the down center point,(the negative direction of Z is negative)
Pules4OneCircle ：When frame=12,the pulse amount of the end rotating axis

BASE(axis1, axis2, axis3)
CONNFRAME(2/12 ，tablenum, viraxisx, viraxisy, viraxisz[,viraxisu])

viraxisx：The number of the virtual axis X
viraxisy： The number of the virtual axis Y
viraxisy： The number of the virtual axis Z
Viraxisu： When frame=12, the axis number of the end rotating axis.it is a actual axis.



The order of the axis and the position of zero point

The zero point direction of the virtual axis:as shown in the figure,Z motives up is positive direction.
The zero point direction of the joint axis:as shown in the figure, the up joint horizontal direction is 0,the downward rotation is positive direction.

Frame = 3， stack

TABLE(tablenum, LargeX,LargeZ, PulesLargeOneCircle, L1,L2, Pules1OneCircle, Pules2OneCircle,SmallX,SmallZ [,PulesSmallOneCircle])

store in sequence:
LargeX: a horizontal offset of the base
LargeZ:。a vertical offset of the base
PulesLargeOneCircle: The pulse amount of the pan axis
L1: the length of the main mechanical arm axis
L2: the length of the small mechanical arm axis
Pules1OneCircle: The pulse amount of the main mechanical arm axis
Pules2OneCircle: The pulse amount of the main mechanical arm axis
SmallX：.a horizontal offset of the end
SmallZ： a vertical offset of the end ,generally, it is negative,
PulesSmallOneCircle：The pulse amount of the ending axis,the setting of virtual axis viraxis3 is the same as this,the parameter is optional.

BASE(axisLarge,axis1, axis2[, axisu])
axisLarge： Big rotating axis
axis1： the number of Big joint axis
axis2： the number of small joint axis
axisu： the axis number of the end rotating axis.

CONNFRAME(3 ，tablenum, viraxis0, viraxis1,viraxis2[,viraxis3])

viraxis0： The number of the virtual axis X

Viraxis1： The number of the virtual axis Y

Viraxis2： The number of the virtual axis Z

Viraxis3： The number of the virtual and small rotating axis,optional,when there is no need to fill it,it only appliedas a converse between XYZ,now,the corresponding date in the TABLE needn’t to be filled .

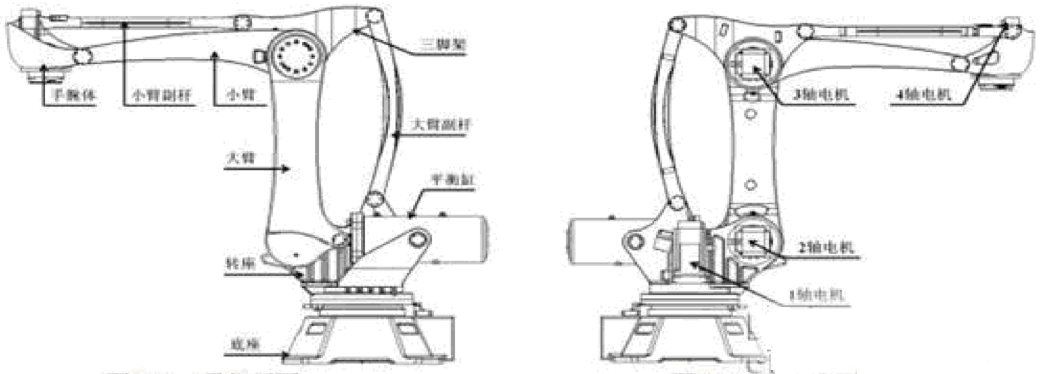


图 1(a)　零位正面　　　图 1 (b) 零位背面

码垛机器人三维模型（零位）

The zero point direction of the coordinate

The base:the direction points to virtual axis X is 0,turn to the direction of the virtual axis Y is positive direction.

Big arm:the vertical is 0,the downward rotation is positive direction.

Small arm:the vertical is 0,the downward rotation is positive direction. When using the end of the small rotating axis W, the positive direction of W is consistent with the big rotating axis,urn to the direction of the virtual axis Y is positive direction.

For example：

BASE(0,1,2,3) ' assuming the number of the joint axis is 0/1/2/3

UNITS=10,10,10,10' here,measured in 10degrees

Dpos=0,0,0,0’ Set the position of the joint axis,here it can be revised according to the practical situation

BASE(6,7,8,9)

atype=0,0 ,0,0’ set as the virtual axis

TABLE(0,10,10,3600,100,100,3600,3600,10,-10,3600) 'The parameter is stored in TABLE0 start-up position, one circle of motor is 360 buffer.

UNITS=1000,1000,1000,10' The buffer equivalent should be set ahead of time, can’t change halfway。

BASE(0,1,2,3) the number of the joint axis is 0/1/2/3

speed=10000,10000,10000,10000' set speed limitation of the joint axis

accel=100000,100000,100000,10000’ set acceleration limitation of the joint axis

CONNFRAME(3,0,6,7,8,9) ' Take the sixth,seventh ,eighth and ninth axis as the virtual axis XYZW,start connect

WAIT LOADED Wait the motion loading,now adjusting the position of the virtual axis automatically.
the third step:motion virtual axis
BASE(6,7,8,9)
MOVEABS(100,100,100)

Frame = 5， The rotation and resizing mechanical arm

TABLE(tablenum, L0, Pules1OneCircle, Pules2OneUnit[,Pules3OneCircle])

store in sequence:

L0: The length of the rotation and resizing axis when the axis is 0.

Pules1OneCircle: the pulse amount of the first joint axis,the big rotating axis

Pules2OneUnit: the pulse amount of the first joint axis,the telescopic axis,must keep consistent length unit with the virtual axis XY.

Pules3OneCirclePulse amount of the ending axis,the setting of virtual axis viraxis3 is the same as this,the parameter is optional.

BASE(axis1, axis2[, axis3])

axis1： The number of the big rotating axis

axis2： The number of the telescopic axis

axis3： The number of the ending rotating axis

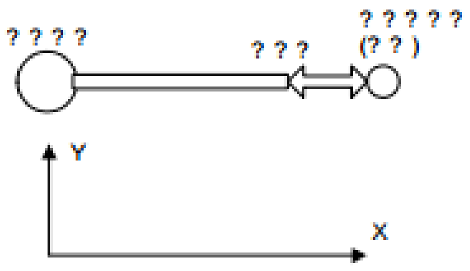
CONNFRAME（5，tablenum, viraxis0, viraxis1[,viraxis2]）

viraxis0： The number of the virtual axis X

Viraxis1： The number of the virtual axis Y

Viraxis2 ： The number of the virtual and ending rotating axis,optional,when there is no need to fill it,it only appliedas a converse between XY.

Coordinate direction:two joint axes become a line when they are 0 point,now point to the positive direction of the virtual axis X,when joint axis is in 0 ,the coordinate of the virtual axis is （l0,0）, the rotating direction and the XY's counterclockwise must be consistent.As shown in the
Figure.



When using the end of the small rotating axis W, the positive direction of W is consistent with the big rotating axis,

For example：

the following date L0,the equivalent can be revised according to the practical situation.

L0=10'

BASE(0,1) ' assuming the number of the joint axis is 0/1
UNITS=10,100 ‘here,measured in degrees
DPOS=0,0’ Set the position of the joint axis,here it can be revised according to the practical situation
BASE(6,7)
ATYPE=0,0 ’ set as the virtual axis
TABLE(0,L0,3600,100) ' The parameter is stored in TABLE0 start-up position, one circle of motor is 360 buffer.
UNITS=1000,1000' The buffer equivalent should be set ahead of time, can’t change halfway。
BASE(0,1) ' assuming the number of the joint axis is 0/1
CLUTCH_RATE=0,0 ' Use the limitation of the speed and acceleration of the joint axis
SPEED=10000,10000' set speed limitation of the joint axis
ACCEL=100000,100000' set acceleration limitation of the joint axis
CONNFRAME(5,0,6,7) ' Take the sixth or seventh axis as the virtual axis XY ,start connect
WAIT LOADED Wait the motion loading,now adjusting the position of the virtual axis automatically.
the third step:motion virtual axis
BASE(6,7)
MOVEABS(10,10) 'The joint axis can motive to the position 0,90

The special type : Frame = 100 ，5,vertical coordinate plus a vertical rotation and a ending swing rotation.

TABLE(tablenum, Lw, PulesVOneCircle, PulesWOneCircle, Lvwx, Lvwy)

store in sequence:

Lw: The length of the swing axis,when it is a negative which means the forward of Z is downward

PulesVOneCircle: The pulse amount of vertical rotating axis

PulesWOneCircle:。The pulse amount of the swing rotating axis

Lvwx: The X shift between the zero point and V rotating center point,(the negative direction of X is negative)

Lvwy: The Y shift between the zero point and V rotating center point,(the negative direction of X is negative)

BASE(axisx, axisy, axisz, axisv, axisw)

axisx： The number of the axis X
axisy： The number of the axis Y
axisz： The number of the axis Z
axisv: The number of the ending rotating axis
axisw: The number of the ending swing axis

CONNFRAME(100 ，tablenum, viraxisx, viraxisy,viraxisz,viraxisv,viraxisw)

viraxisx：The number of the virtual axis X
viraxisy： The number of the virtual axis Y

viraxisz： The number of the virtual axis Z

viraxisv： The number of the vertical rotating axis, it isn’t a vertical axis

viraxisw： The number of the swing rotating axis, it isn’t a vertical axis

The zero point direction :the zero of the rotating axis makes the forward of the swing axis and X consistent , the forward of the swing axis points the forward of the axis Y from the forward of the X. when the swing axis is 0, it points to directly below.

The special type : Frame = 100 ，5,vertical coordinate plus a vertical rotation and a ending swing rotation.

If there is need for materials, please contact us.

Frame = 102 ，2 The composing instruction of axis Frame = 102 ，2

TABLE(tablenum, R, r, L1,L2, Pules1OneCircle,Pules2OneCircle, Lx, Ly))

store in sequence:

R: The radius of up distance

r: The radius of down distance

L1: The length of up pole

L2: The length of down pole

Pules1OneCircle: The joint axis 1(left axis) the pulse amount

Pules2OneCircle: The joint axis2(right axis) the pulse amount

Lx: The X shift between the end point and the down center point,(the negative direction of X is negative)

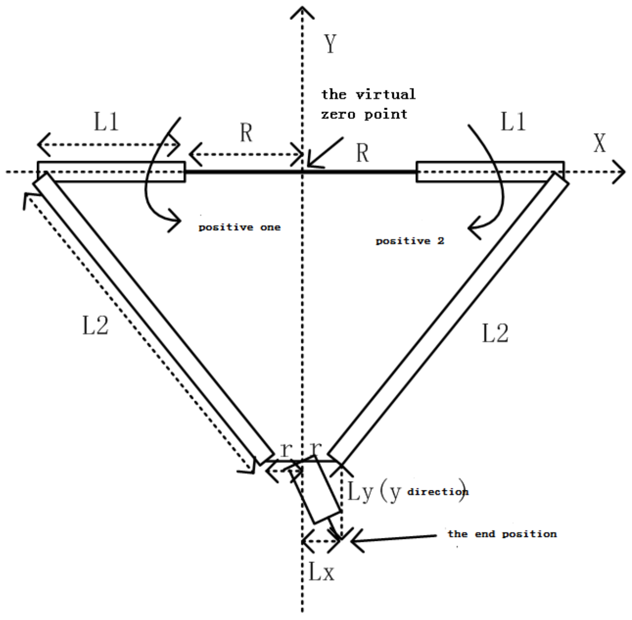
Ly: The Y shift between the end point and the down center point,(the negative direction of Y is negative)

BASE(axis1, axis2)

CONNFRAME(102 ，tablenum, viraxisx, viraxisy)

viraxisx： The number of the virtual axis X

viraxisy： The number of the virtual axis Y



The zero direction of the virtual axis:right and left is X,right is positive; up and down is Y,the up is positive;the zero point lies in the middle of the horizontal line.
The zero direction of the joint axis:the first axis in the left,the horizon is 0,turn down is positive; the second axis in the right,the horizon is 0,turn down is positive.

CONNREFRAME

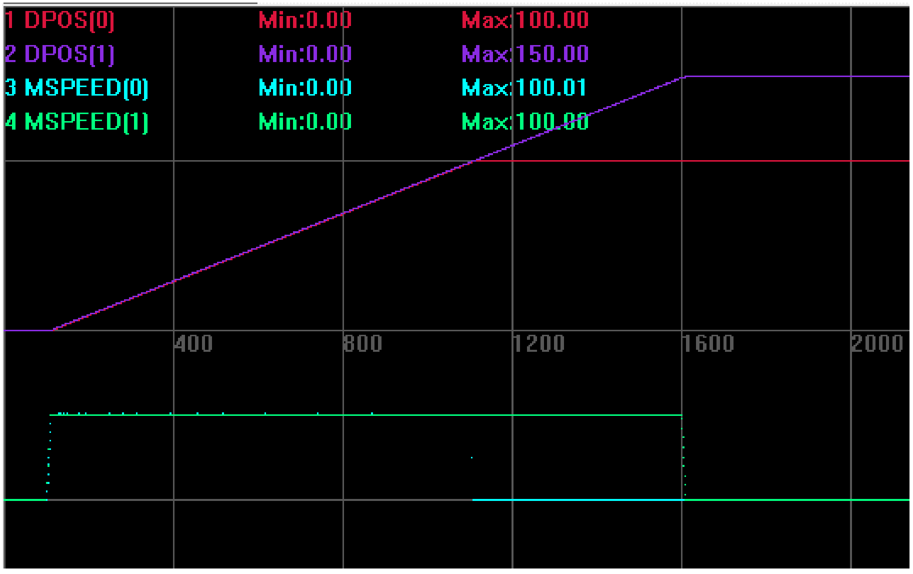
type： simultaneous motion instruction
grammar：CONNREFRAME（frame，tablenum, axis0, axis1,[…]）
description： Connect the coordinate of the virtual axis and joint axis,after the motion of joint axis,the virtual axis goes to the coordinating position ,the instruction is the reverse motion instruction of CONNFRAME.

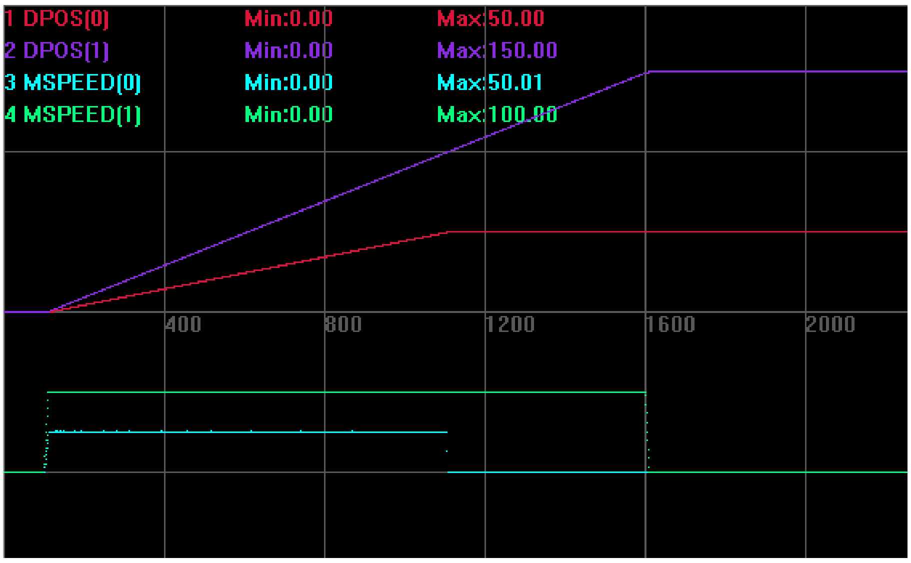
⚠ When the virtual axis CONNREFRAME operates LOAD, the motion of CONNFRAME of the joint axis will be CANCELED automatically.

⚠ When the joint axis CONNREFRAME operates LOAD, the motion of CONNFRAME of the virtual axis will be CANCELED automatically.
parameter： reference to the parameter CONNFRAME,the axis BASE and the axis in parameter is opposite in position.
For example：
L1=10 the length of the first joint axis
L2=10' the length of the second joint axis
BASE(0,1) assuming the number of the joint axis is 0/1
UNITS=10,10 ' here,measured in degrees
DPOS=0,0' Set the position of the joint axis,here it can be revised according to the practical situation
BASE(6,7)
ATYPE=0,0 set as the virtual axis
TABLE(0,L1,L2,3600,3600) '
The parameter is stored in TABLE0 start-up position, one circle of motor is 360 buffer.
UNITS=1000,1000The buffer equivalent should be set ahead of time, can't change halfway。
CONNREFRAME(1,0,0,1) According to the coordinate of 0/1 to calculate the coordinate of CONNREFRAME(1,0,0,1) '6/7
The third step: motion joint axis
BASE(0,1)
MOVEABS(90,0) the coordinate of the virtual axis turns to 0,20

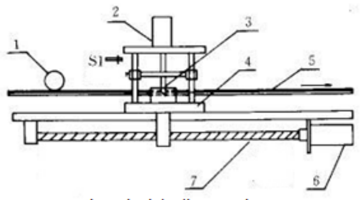
MOVELINK

type	simultaneous motion instruction
description	The instruction is appliedin self-defined cam motion,the motion has settable acceleration/deceleration processing. Connected axis is the basic axis,connective axis is the following axis. The distance of the coupling axis is divided into three stages which is applied in the motio of the basic axis.those locals includes the accelerated local,normanl local and decelerated local. In order to match speed in accelerated and decelerated stage,Link distance (the motion distance of the basic axis) must is double as the distance (the motion distance of the following axis)
grammar	MOVELINK (distance, link dist, link acc, link dec, link axis[, link options] [, link start]) parameter： distance: The connect from the start to the completion,the motion distance of the following axis, adopting user unit. link dist: The absolute distance of the basic axis during the globalprocess of connecting,adopting units unit. link acc: During the accelerated stage of the current axis, the absolute distance of the basic axis,adopting units unit. link dec: During the decelerated stage of the current axis, the absolute distance of the basic axis,adopting units unit. Notice :if the sum of parameter3 and parameter4 is greater than the second parameter, they will be reduced in proportion automatically,to1 make equal of the value of the sum and the second parameter. link axis: the axis number of the reference axis link options: Different bits have different meaning. 1 Bit0, precisive connection starts from when the regiser event of the reference axis which is triggered 2 bite1, connection starts from when the basic axis reaches a absolute position.(referring the description of the parameter link pos) 4bit2,when the position is set,MOVELINK will execute repeatedly and run in the converse direction.(the mode can set the first bit of axis parameter REP_OPTION as 1 to cancel) 32 bit5 it can link only when the basic axis operates in

	positive direction. link pos: When the parameter link options is set as 2, the parameter denotes the basic axis is the value in the absolute position, the link starts.
Application controller	
example	Axis 0 is the following axis, axis 1 is the basic axis. Example one BASE(0,1) ATYPE=1,1 DPOS=0,0 SPEED=100,100 MOVELINK(100,100,0,0,1) when in the stage where setting no acceleration and deceleration,the effect ia the same as the CONNECT,the differnce lies in the difference of UNINTS is no importance,and don't accelerate errors..curently,the motion ratio is 1:1.MOVE(150)AXIS(1) axis1 motives 100,axis0 motives 100 too. interpolation trajectory and velocity curve.  MOVELINK(50,100,0,0,1)时



example two Application of the Flying Shears



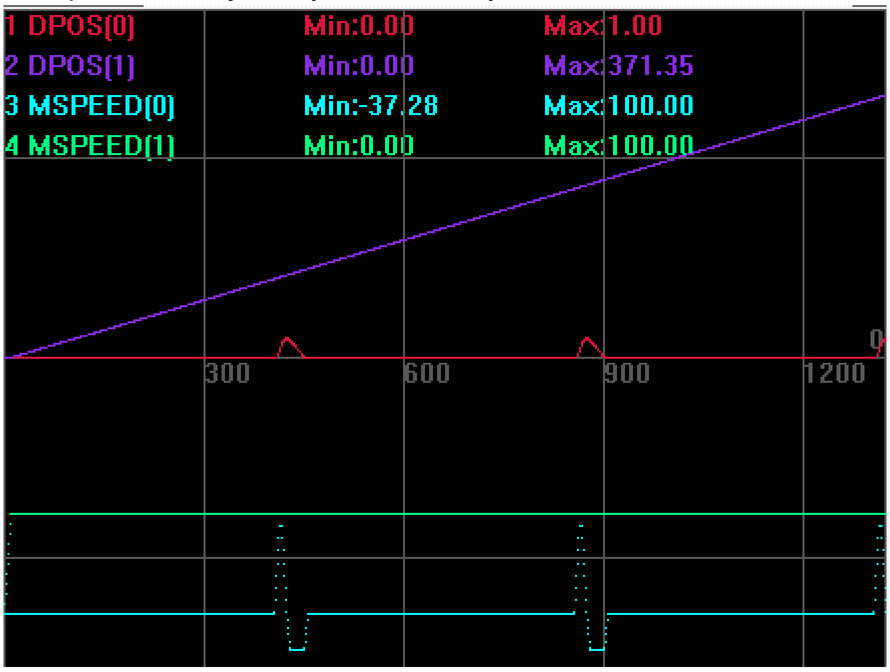
the principle diagram the Flying Shears system

1-length measuring wheel
2-hydraulic cylinder 3-cutter 4-worktable
5-profile 6- servo motor 7-screw

Profile motives continuously, the worktable keeps still firstly untill the profile motives continuously to some distance,then the worktable starts to accelerate ,wait the speed of the worktable and profile are consistent, then the switch S1 operates tool cut,after cutting,the tool recovers. then the worktable starts to decelerate,then back to the initial point. Repeat the process until get the profile of the setting length by cutting.

Assuming,the length of the cutting profile is 50 unit,the worktable operates one distance,the basic axis is axis1(profile),axis0 is the follwing axis(worktable),OUTO port controls tool,the following is the local Flying Shears procedure.

```
BASE(0,1)
VMOVE(1)AXIS(1)        profile motives continuously
WHILE 1
```

	MOVELINK(0,45,0,0,1) Before the profile motives 45 units,the woektable keeps still. MOVELINK(0.4,0.8,0.8,0,1) the stage of the worktable acceleration MOVELINK(0.2,0.2,0,0,1) The stage of the speed follow MOVE_OP2(0,on,1000) The tool cut,then recover after 1s. MOVELINK(0.4,0.8,0,0.8,1) 'the stage of the worktable deceleration MOVELINK(-1,3.2,0.5,0.5,1) the worktable back to the initial point WEND interpolation trajectory and velocity curve.  The motion distance of the worktable(the following axis):.0.4(accelerate) +0.2(follow)+0.4(deceleration)=1unit , then-1back The motion distance of the profile(the basic axis): 45+0.8+0.2+0.8+3.2=50units , the velocity is constant all the time.
S12	

MOVESLINK

type	simultaneous motion instruction
description	The instruction is applied in self-defined cam motion,the motion plans the intermediate curve,don't calculate the cam table. Connected axis is the basic axis,connective axis is the following axis.

	In order to match speed in accelerated and decelerated stage,Link distance (the motion distance of the basic axis) must is double as the distance (the motion distance of the following axis)
grammar	MOVESLINK (distance, link dist, start sp, end sp, link axis[, link options] [, link start]) distance: The connect from the start to the completion,the motion distance of the following axis, adopting user unit. link dist: .The absolute distance of the basic axis during the globalprocess of connecting,adopting units unit. start sp: When start up,the velocity proportion of the following axis and the basic axis, units/units unit,the negative number means the following axis motives in the negative direction. end sp: When end,the velocity proportion of the following axis and the basic axis, units/units unit,the negative number means the following axis motives in the negative direction. Notice ：when start sp = end sp = distance/ link dist,匀速运动。the velocity is constant. link axis: the axis number of the reference axis link options: Different bits have different meaning. 1 Bit0, precisive connection starts from when the regiser event of the reference axis which is triggered . 2 bite1, connection starts from when the basic axis reaches a absolute position.(referring the description of the parameter link pos) 4bit2,when the position is set,MOVELINK will execute repeatedly and run in the converse direction.(the mode can set the first bit of axis parameter REP_OPTION as 1 to cancel) link pos: When the parameter link options is set as 2, the parameter denotes the basic axis is the value in the absolute position, the link starts.
Application controller	general use
example	The function is the same as MOVELINK, only the difference in parameter setting. Example one： BASE(0) MOVESLINK(50,100,0,1,1) ‘The axis 0 follows the axis 1,the axis 0 starts to accelerate untill it reaches the same speed as the axis1.

	MOVESLINK(100,100,1,1,1) The axis 0 follows the axis 1at constant speed 100units. MOVESLINK(50,100,1,0,1) ‘The axis 0 follows the axis 1 , the speed reduces to 0. Example two : BASE(0) MOVESLINK(startdis,100,0,1,1) ‘The axis 0 follows the axis 1,the axis 0 starts to accelerate untill it reaches the same speed as the axis1. While 1 ‘ cyclic motion MOVESLINK(100,100,1,1,1)‘The axis 0 follows the axis 1at constant speed 100units. MOVESLINK(dis1,dis2,1,1,1)‘ Plan the motion of the axis,make sure the speed of two axis is consistent in the end. WEND
relative instruction	MOVELINK

REGIST

type	positional regist instruction
description	**The instruction REGIST is appliedto measure feedbacks position of regist axis,only when the axis has encoder properties, it supports lock.** When regist happens,the situation MARK of axis will be set as ON,simultaneously,is registed position will be stored within the parameter REG_POS. Each axis has input signal,the signal R0,R1, EZcan use regist function.when using two signals to regist,the second regist signal use MARKB and REG_POSB. Generally, input R0,R1coresponding to the input port respectively 0and 1,if need detail information,please of comtroller .
grammar	REGIST(mode) paramete : mode: regist mode When Z pulses up,the absolute position of Z is sent to REG_POS. When Z pulses down,absolute position of Z is sent to REG_POS.

	When the input signal R0 up, the absolute position is sent to REG_POS. When the input signal R0 down , the absolute position is sent to REG_POS. When the input signal R0 down , the absolute position is sent to REG_POS. The signal Z up ,the absolute position is sent to REG_POSB When the input signal R0 up , the absolute position is sent to REG_POS.the signal Z down ,the absolute position is sent to REG_POSB When the input signal R0 down , the absolute position is sent to REG_POS. The signal Z up ,the absolute position is sent to REG_POSB When the input signal R0 down , the absolute position is sent to REG_POS. The signal Z down ,the absolute position is sent to REG_POSB When the input signal R0 up , the absolute position is sent to REG_POS. The signal R1 up ,the absolute position is sent to REG_POSB When the input signal R0 up , the absolute position is sent to REG_POS. The signal R1 down ,the absolute position is sent to REG_POSB When the input signal R0 down , the absolute position is sent to REG_POS. The signal R1 up ,the absolute position is sent to REG_POSB When the input signal R0 down , the absolute position is sent to REG_POS. The signal R1 down ,the absolute position is sent to REG_POSB After 150804's version supports,every regist channel is independent. The signal R1 up ,the absolute position is sent to REG_POSB The signal R1 down ,the absolute position is sent to REG_POSB The signal Z up ,the absolute position is sent to REG_POSB The signal Z down ,the absolute position is sent to REG_POSB Up and down is for the inner situation of the controller,the input port of different type may be inconsistent,need to make sure the edgeof the actual regist.
R1	general use
example	example1 ： The input signal R0 of the regist axis 0 BASE(0) REGIST(3) WAIT UNTIL MARK

	PRINT REG_POS Example 2： PC exchanges position to regist,generally it is appliedin motion capture, to get the actual position in capture through the position of regist. global g_start global g_posx, g_posy while 1 wait until g_start=1 ' wait PC to start up regist(4) axis(0) ' input register 0,24v turns 0v regist(4) axis(1) wait until mark(0) and mark(1) g_start=0 g_posx=reg_pos(0) g_posy=reg_pos(1) 'print g_posx, g_posy wend
relative instruction	MARK, MARKB, REG_POS, REG_POSB

ENCODER_RATIO

type	motion setting instruction
description	**encoder inputs gear ratio ,default value (1,1)**
grammar	ENCODER_RATIO(mpos_count, input_count) parameter： mpos_count the numerator , don’t surpass 65535 input_count the denominator, don’t surpass 65535
Application controller	general use
example	ENCODER_RATIO(4,1) ‘Encoder four times input
relative instruction	PP_STEP

STEP_RATIO

type	motion setting instruction
description	**Set Step and output gear ratio, default value (1,1)**

	Don't revise casually, it can achieve the same effect by revising pulse equivalent.
grammar	STEP_RATIO(output_count, dpos_count) parameter : output_count the numerator , don't surpass 65535 input_count the numerator , don't surpass 65535
Application controller	general use
example	STEP_RATIO (16,1) 'The pulse number of 16times outputs,it can achieve the same effect through the pulse equivalent multiplicates 16.

BACKLASH

type	motion setting instruction
description	The opposite clearance of the setting axis, the extended axis in invalid.
grammar	BACKLASH(enable [,dist[, speed, acceleration]])
Application controller	general use
example	BACKLASH(0) ' Close the function of the opposite clearance BACKLASH(1, 0.1) 'Set the opposite clearance 0.1mm

.

PITCHSET

type	motion setting instruction
description	The Pitch Compensation of the setting axis, the extended axis in invalid. The pulse compension number of each point is stored in TABLE.
grammar	PITCHSET(enable [, startpos, disone, maxpoint, tablenum])
Application controller	general use
example	PITCHSET(0) ' Close the function. PITCHSET(1,0,10,100,0) Set 100 points,start to store from TABLE(0). TABLE(0)

Chapter 3 Program Structure And Program Procedure

type： special character
description：The following is notation, start until the next line._
type： special character
description：Change the following line then continue.

When conditional judgment statement, Memory Storage,print output,don't use as soon as possible.

CONST

type： grammar instruction
Grammar ： CONST CVARNAME = value
the alias ：no
description ： The defining symbol represents the constant ,which avoids using the value to represent.

parameter：	CVARNAME	the name of constant
	value	the value of constant

for example： CONST MAX_VALUE = 100000' definite the constant

DIM

type： grammar instruction
grammar：DIM VarName, ArrayName(space)
description： The definition file mode variety,array;when the variety is valued without defining,it will be defined as the definition file mode automatically.
The definition file mode variety can only be applied inside the program file.
Array can be applied as string.

parameter：	VarName	the name of variety
	ArrayName	the name of array
	space	the length of array

reference to: MODBUS_STRING, VRSTRING string function
for example：

```
DIM ARRAY1(100)' defined array ARRAY1
DIM VAR1    defined variety VAR1
VAR2 = 100 valuating sentence will define the file mode variety automatically.
ARRAY1 = "asdf"
ARRAY1(0,100,200,300)  'valuate continuously the array
```

LOCAL

type： grammar instruction
grammar：LOCAL localname
description： Define localial variety,localial variety
parameter： localname the name of the localial variety
for example：
 LOCAL V1 define variety V1

⚠There is the number limitation in the localial variety of the same SUB,the input of parameter will turn to localial variety automatically during the SUB procedure .

⚠ In the same SUB procedure with the different tasks which calls and forms different localial variety,In the SUB procedure with the same tasks whose Recursive Calls form different localial variety too.
GLOBAL
type： grammar instruction
grammar1： GLOBAL VAR1
grammar2： GLOBAL SUB SUB1()
grammar3： GLOBAL CONST CVARNAME = value
description： Define the globalvariety,array;define the globalSUB procedure.
parameter： VAR1：the name of variety
SUB1： the name of procedure
CVARNAME： the name of constant
Value： the value of constant

for example：
 GLOBAL SUB g_sub2() Define the global procedure g_sub2，can use in arbitrary file.
 GLOBAL CONST g_convar = 100 'Define the applied constants in the whole
 GLOBAL g_var2'define the variety of the whole g_var2

SUB

type：grammar instruction

grammar：SUB label([para1] [,para2]…)
 END SUB
description： the procedure of the user's self-definition,can increase GLOBAL description in the front to define the USB procedure of the global use.
reference to：GOSUB, GLOBAL

parameter：
    label: the name of procedure,can’t conflict with the existing key words.
    para1： The transporting in parameter in procedure call , the parameter is taken as the localial variety of LOCAL automatically.
    para2： The transporting in parameter in procedure call , the parameter is taken as the localial variety of LOCAL automatically. …
for example：
        SUB sub1() ’ The defined procedure SUB1, which is only appliedin the existing file.
        End sub
        GLOBAL SUB g_sub2() ’ Define the globalprocedure g_sub2 ，can use in arbitrary file.
        End sub

GSUB

type：grammar instruction：GSUB label([char1] [,char2]…)
        End sub
description： the procedure of the user’s self-definition,can increase GLOBAL description in the front to define the USB procedure of the global use. When the GSUB procedure calls,it uses G code grammar without bracket.
reference to：GOSUB, GLOBAL
parameter：
    label: the name of procedure,can’t conflict with the existing key words.
    char1： The transporting in parameter in procedure call , the parameter is taken as the localial variety of LOCAL automatically.
    char2： The transporting in parameter in procedure call , the parameter is taken as the localial variety of LOCAL automatically.
    …for example：
    G01 X100 Y100 Z100 U100 ' calls G10
    GLOBAL GSUB G01(X, Y, Z, U)’ define the GSUB procedureG10
    ...
    End sub

:

type：grammar instructure
grammar：label:
description： The user process ;label,the label can be appliedas SUB process without parameter.
reference to：SUB， GOTO, GOSUB
parameter：
    label: The name of the label,can’t conflict with the existing key words.

for example：

    goto label1

        1.1.1label1: ' plus:the defined label

    end

DMINS

type：grammar instructure

grammar：DIMINS   ArrayName(pos)
the alias ：no
description： The Linked list opration of the array,after inserting,the existing elements and all the following elements will move afterward one opsition.
parameter：   ArrayName the name of the array
        pos   The index array
for example：
    DMINS ARRAY1(0) 'Insert element 0,all the initial elements will move one opsition.
    ARRAY1(0) = newdata 'valuate the inserted element

⚠ Be caucious to operate the Super-long array,especially the array TABLE.

DMDEL

type：grammar instruction
grammar：DMDEL   ArrayName(pos)
the alias ：no
description： The Linked list opration of the array,delete array elements,after that,the existing elements and all the elements will move forward.
parameter：   ArrayName the name of the array
        pos   The index of the array
for example：
        DMDEL   a(0)' delete the first element of the array A.

Be caucious to operate the Super-long array,especially the array TABLE.

DMCPY

type：grammar instruction

grammar :DMCPY ArrayDesName(startpos) , ArraySrcName(startpos)[, size]
the alias ：no
description： Array copy,copy Src to Des.
parameter： ArrayName the name of the array

startpos	The initial index of the array
size	The number of the copy,surpass the max then it will decrese automatically.

for example：

```
dmcpy aaa(0), bbb(0),100
```

Be caucious to operate the Super-long array,especially the array TABLE.

IF

type： program structure
grammar： IF <condition1> THEN

```
commands
ELSEIF <condition2> THEN
commands
ELSE
commands
ENDIF
```

description： Conditional judgement,the same standard BASIC structure

parameter： condition1 condition
condition2 condition
for example1 ：

```
dim a ’ defined variety
a=12 ’ valuate
if a>11 then ‘Conditional judgement
trace "the val a is bigger then 11"
elseif a<11 then
trace "the val a is less then 11"
end if
```

for example2 ：

```
if in(0) then out(0,on) '
  When it runs  individually,can no end if
```

THEN

type： program structure
reference to：IF

ENDIF

type： program structure
reference to：IF

ELSEIF

type： program structure
reference to：IF

FOR

type： program structure
grammar：FOR variable=start TO end [STEP increment]

```
        ...
        'block of commands
        ...
        NEXT variable
```

description： Loop sentence,use standard BASIC grammar.

parameter： variable	the name of the variety
start	the start loop value
end	the end loop value
increment	loop the incremental steps,it is optional

for example：

```
        local a
        for a=1 to 100'1 loop to 100
          print a, '
        next
```

TO

type： program structure
reference to：FOR

STEP

type： program structure
reference to：FOR

NEXT

type： program structure
reference to：FOR

EXIT

type： program structure
grammar：EXIT FOR , EXIT WHILE
description： Loop the entry sentence.
for example：

```
local a
for a=1 to 100'1 loop to 100
  print a, '
  if a> 20 then exit for 'Must use the way,or IFand ENDIF are
  unmatched.
next
```

GOSUB

type： program structure
grammar：GOSUB label
the alias ：CALL
description： Call the SUB process,only call the SUB process of the file and the global

Call the SUB process directly, can omit the GOSUB sentence.

When the parameter isn't in SUB process,can omit bracket.

After GOSUB, the existing content can push stack,can't access the existing localial variety when calling the SUB process.when RETURN,it out the stack.

for example：

```
GOSUB sub1()
sub2(1,2) transport 1 to para1,transport 2 to para 2.
call sub3
END

SUB sub1()
  a=100
  print "sub1"
return

SUB sub2(para1,para2)
```

```
    a=200
    print “sub2”,para1,para2
return

GLOBAL SUB sub3()   can exist in the another pragram file
    a=30
    print “sub3”
return
```

CALL

type： program structure
reference to：GOSUB

GOTO

type： program structure
grammar：GOTO label
description：
Jump forcibly, GOTO differs from GOSUB in GOTO can’t push stack.
for example：

```
a=100
goto label1     Jump forcibly to label1
a=1000
label1:
print a      the end is 1=100
End
```

ON GOSUB

type： program structure
grammar：ON expression GOSUB label
description：when the condition expression is ture,calling process label
for example：

```
        a=100
on a>10 gosub label1    when a>10,call process label
        a=1000
        print a
        end

        label1:
        print a
```

```
return   'SUB process will return
```

ON GOTO

type： program structure
grammar：ON expression GOTO label
description：Conditional jump,when the condition expression is ture, jump
for example：

```
a=100
on a>10 goto label1
a=1000

label1:
print a
end   'The jump of the goto can't return
```

REPEAT

type： program structure
grammar：REPEAT commands UNTIL condition
description： Excute circularly commands untill the condition is ture,then exit loop.
for example： loop sentence,

```
a=0
repeat
print a
a=a+1
delay(1000)
until IN(4)=ON untill input number 4 is valid
```

WHILE

type： program structure
grammar： WHILE condition

```
…
WEND
```

the alias ：no
description：loop，condition when the condition is met,execute loop
parameter： condition
For example：

```
a=0
while NOT IN(4)=ON     ' untill input number 4 is valid
```

```
a=a+1
print a
delay(1000)
wend
```

WEND

type： program structure
reference to：WHILE

END SUB

type： program structure
grammar：END SUB
reference to：SUB
description： The user self-defined SUB process finishes.
for example：

```
sub sub1()
  …
END SUB
```

RETURN

type： program structure
grammar：RETURN
reference to：SUB
description： The user self-defined SUB process returns or return value,yhe return value default 0,can read the last the return value of the SUB call by RETURN outside.
for example：

```
Call sub1
?return ‘the result is 111

sub sub1()
  …
  return 111

end sub
```

UNTIL

type： program structure
reference to：REPEAT WAIT

DELAY

type： grammar instruction
grammar：DELAY(delay time)
the alias：WA
description： delay time， The unit :millisecond
parameter： delay time the millisecond amount
for example： DELAY(100) ‘delay 100ms

WAIT UNTIL

type： grammar instruction
grammar：WAIT UNTIL condition
description：Wait until the condition is met
for example：

WAIT UNTIL DPOS(0) > 0’ Wait the position of axis 0 surpasses 0.

WAIT IDLE

type： grammar instruction
grammar： WAIT IDLE
description： Wait the axis BASE to finish motion, when it doesn’t finish,cann’t execute the later progarm. Which is equivalent to WAIT UNTIL IDLE.as a axis parameter,IDLE supports the grammar of the parameter.
for example1：

```
Base(0,1)
Move(100,100)
Wait idle    Wait to finish the existing interpolation motion.
```

for example2：

```
Base(0,1)
Move(100,100)
Base(2,3)
Move(200,200)
Wait unitl idle(0) and idle(1) and idle (2) and idle(3) ‘等待轴 0，1，2，3 停止
```

Wait the finish of the axis 0,1,2,3.

WAIT LOADED

type： grammar instruction
grammar： WAIT LOADED
description： Wait the axis BASE motion buffer is empty,but the existing motion can execute truly, the instruction can block and can't execute the later program. Which is equivalent to WAIT UNTIL LOADED.as a axis parameter,lOADED supports the grammar of the parameter.

for example：

```
BASE(0)
WAIT LOADED  Wait the axis BASE motion buffer is empty.
```

Chapter 4 Task-related

ZBasic supports real-time multi-task at once time, synchronous multi-task running is also available in one file. The task will run from the first line of the file, through the RUN orderable to appoint any USB process to start to run running through RUNTASK

BASE_MOVE

Type: task parameters
Grammar: ：VAR1 = BASE_MOVE，BASE_MOVE = value.
Description: the BASE spindle used to force the specified interpolation motor function. This parameter does not modify the actual motion.
The default value is -1, which does not take effect at this time.
More than 20160326 firmware version support.
Each task has an independent BASE_MOVE parameter.
MOVE MOVEABS MOVECIRC MOVE_OP MOVE_TASK and other interpolation type functions are valid, and the cam point and other single axis functions are invalid.
For example: BASE_MOVE=2 "behind the mandatory interpolation tasks using 2 Axis axis, 2 axis speed parameter using

```
MERGE (2) =ON 'axis 2 start continuous
SPEED (2) =100
ACCEL (2) =1000
BASE (0,1)
MOVE (100100) 0/1'
 MOVE (100, 100) 'axis 0/1 interpolation, axis 2 is also forced as
 the main axis, but the distance of movement is 0
```

```
MOVE_OP (1,1)
  BASE (1)
MOVE (100) '1 axis motion 100, also use axis 2 as the main axis
 BASE_MOVE = -1 'cancels the mandatory principal axis of this task
```

RUN

Tape: task instruction
Grammar：RUN “filename”[, tasknum]
Description: creates a new task to execute a file on the controller. Multi task
operating instructions:
END: the current task is finish normally
STOP: STOP specifying the file
STOPTASK: stops specifying the task
HALT: stop all tasks
RUN: start file execution
RUNTASK: start task executes on a SUB
Parameter: filename: program file name, BAS file cannot expand tasknum:
task number, and the default search is the first valid
Example: run "aaa", 1 "run aaa. Bas in task 1

RUNTASK

Type: Task instructions
Grammar：RUNTASK tasknum, label
Description: a SUB process or label is executed as a new task
Parameter： tasknum: task number.
Label: custom SUB procedure (with no Parameters), or label
Example: runtask 1, taska 'start task 1 to track the print position

```
MOVE (1000100)
MOVE (1000100)
end
taska:' loop print location
while 1
  print *mpos
  delay(1000)
wend
end
```

FILE3_RUN

Type: Task instructions

Grammar：FILE3_RUN “filename”, tasknum。
Description: the startup command for the 3 program file must be supported by a large capacity storage controller and more than 2015 firmware versions.
Parameter: filename: the file name of 3 programs must be downloaded to the controller in advance.
Tasknum: task number, defaults to finding the first valid
Reference to： FILE3_ONRUN, FILE
For example: FILE3_RUN "aaa. Z3p", 1 "in task 1 running 3 times aaa. Z3p

FILE3_ONRUN

Type: callback function
Grammar ：GLOBAL SUB FILE3_ONRUN:
Description: a SUB process or label is executed as a new task
Parameter: tasknum：Task number.
Label： Custom SUB procedure (with no Parameters), or label
For example:

```
 FILE3_RUN "aaa.z3p", 1 "in task 1 run 3 times aaa.z3p
 END
GLOBAL SUB FILE3_ONRUN: 'three tasks are automatically invoked when
the task starts
 Print
 IF     1= PROCNUMBER THEN
        Base (0, 1, 2) 'select the running axis list of the three files
        SPEED=1000
        ACCEL=10000
 ELSE
        Base(4,5,6)
 END IF
 end sub
```

END

Type: Task instructions
Description: the current task is finish
Reference to：RUN, RUNTASK

STOP

Type：Task instructions
Grammar：STOP program name, [tasknum]

Description: the program is forced to stop. When a file has multiple tasks, it is
recommended to use the STOPTASK command
Parameter: program name: program file name, bas file does not need to have
extension.
Tasknum: task number, when the program file has multiple tasks,
the default task number is the smallest task
For example:
RUN aaa, 1 'execute aaa. Bas
RUN aaa, 2
STOP aaa STOP task 1
STOP aaa, 2 STOP task 2

STOPTASK

Type: task instructions
Grammar：STOPTASK [tasknum]
Description：the task is force to stop
Parameter： tasknum: task number，default current task
For example ： STOPTASK 2’stop task 2

HALT

Type: task instructions
Description：Stop all tasks.
For example ：
HALT 'stops all tasks

PAUSE

Type：task instructions
Description: pause all tasks, usually only PC software call, breakpoint break
will also enter a suspended state.
For example:
PAUSE "pauses all tasks

PAUSETASK

Type: task instructions
Description: Suspend a task
Parameter: tasknum: task number, default current task
Reference to：PROC, PROC_STATUS,RESUMETASK
For example:

PAUSETASK 1 'pause task 1

RESUMETASK

Type：task instructions
Description: suspend a task
Parameter: tasknum: task number, default current task
Reference to：PROC, PROC_STATUS,PAUSETASK
For example: RESUMETASK 1 'continue to run task 1

PROC

Type：Task correction auxiliary instruction
Grammar：PROC(tasknum)
Description: other tasks can be specified when accessing task parameters, task states.
Parameter：tasknum: task number
Reference to: task parameters, task status
Example: Print ERROR_LINE PROC (1) print error line for task 1

PROCNUMBER

Type：the task is in a special state, system state
Grammar ：VAR1 = PROCNUMBER
Description: The task number of the current task. when it does not know the current task number accessed by this program. This state cannot be corrected by PROC.
Reference to ：PROC
Example:

Print PROCNUMBER ‘print the current task number

ERROR_LINE

Type：Task status
Grammar：VAR1 = ERROR_LINE
Description：error line number of the current task ，。
Reference to：PROC
For example：

Print ERROR_LINE: ‘print the error line of current task
Print ERROR_LINE(1): ‘print the error line of task1

PROC_LINE

Type：Task status
Grammar ：VAR1 = PROC_LINE
Description：The current line of the task
Example:

```
Print proc_line (0) print task 0, run to that line
>>print PROC_LINE
100
```

PROC_STATUS

Type：Task status
Grammar：VAR1 = PROC_STATUS
Description：The state of the current task
0: task stop
1: task is running
3: task is pause
For example ：

```
Print PROC_STATUS(0) ’print task is in zero state
>>print PROC_STATUS(0)
1 ‘in service
```

RUN_ERROR

Type ： task status
Grammar ：VAR1 = RUN_ERROR
Description ：error number of the task
Example ：

```
Print running error(1)   ‘print the error number of the task1
0
```

TICKS

Type：Task parameters
Grammar：VAR1 = TICKS，TICKS = value
Description：Counting cycles number of the current task, each cycle minus one
Each task has its own dependent TICKS parameter, cycle of ZMC00X series and ZMC1XX series is one millisecond
System refresh cycle modification does not affect the timing of

this TICKS.

For example:

```
ticks = 1000
wait until ticks < 0 'Wait for the ticks < 0
move(100)
```

Chapter 5 Operator And Mathematical Function

ZBASIC supports all operators of standard BASIC, and also uses standard BASIC priority.

Priority order: arithmetic operator > comparison operator > calculation logic operator. Operators of the same priority are computed from left to right.

arithmetic operator		comparison operator		ogical operator	
describe	**symbol**	**describe**	**symbol**	**describe**	**symbol**
exponentiation	**^**	equal to	**=**	Logical NON	Not
minus	**-**	not equal to	**<>**	Logical AND	And
multiply	*****	less-than	**<**	Logical OR	Or or /
divide	**/**	greater than	**>**	Xor	Xor
be divided with no remainder	\	less than or equal to	**<=**	Logical equivalence	Eqv
complementation	Mod or %	be equal or greater than	**>=**		
add	**+**				
subtract	**-**				
shift left	**<<**				
shift right	**>>**				

+

Type: operator
Grammar: expression1 + expression2
Description: add two expressions together
Parameter: expression1: any valid expression
expression2: any valid expression
For example：
```
>>print 1+2
3
```

–

Type: operator
Grammar: expression1 - expression2
Description: subtracts expression 1 from expression 2
Parameter:
expression1: any valid expression
expression2: any valid expression
For example：
```
>>print 2-(2-1)
1
```

*

Type: operator
Grammar: expression1 * expression2
Description: multiplies expression 1 with expression 2:
Parameter
expression1: any valid expression
expression2: any valid expression
For example：
```
>>print 10*(1+2)
30
```

/

Type: operator
Grammar: expression1 / expression2
Description: expression 1 divided by expression 2

Parameter:
Expression1: any valid expression
expression2: any valid expression
For example：

```
>>print 10/5
2
```

=

Type: operator
Grammar: expression1 = expression2
Description: comparison operator: if expression 1 equals expression 2, return TRUE, otherwise FALSE
Assignment operator: assign the expression 2 to the preceding variable or parameter. etc.
Parameter:
expression1: any valid expression
expression2: any valid expression
Example 1:
ON IN(0)=ON GOTO label1
If the input channel 0 is ON, the program jumps to the line beginning with the identification "label1:" to begin execution
Example 2:
VAR1 = 100
Modify VAR1 to 100.

<>

Type: operator
Grammar: expression1 < > expression2
Description: if the expression 1is not equal to the expression 2, return TRUE, otherwise it returns FALSE.
Parameter:
expression1: any valid expression
expression2: any valid expression
For example：
ON MODBUS_BIT(0)<>0 GOTO label1
If the MODBUS bit register 0 is nonzero, the program jumps to the line beginning with the identification "label1:" to begin execution

>

Type: operator
Grammar: expression1 > expression2
Description: if the expression 1 is greater than expression 2, returns TRUE, otherwise returns FALSE.
Parameter:
expression1: any valid expression
expression2: any valid expression
For example：
WAIT UNTIL MPOS>100
This statement will cause the program to wait until the feedback position is greater than 100

>=

Type: operator
Grammar: expression1 > expression2
Description: if the expression 1 is greater than or equal to the expression 2, returns to the TRUE, otherwise returns to FALSE.
Parameter:
expression1: any valid expression
expression2: any valid expression
For example：
VAR1=1>=0
Because is greater than 0, the value of VAR1 will equal TRUE (-1)

<

Type: operator
Grammar: expression1 < expression2
Description: if expression 1 is less than expression 2, returns TRUE, otherwise returns FALSE.
Parameter:
expression1: any valid expression
expression2: any valid expression
For example：
VAR1=1<0
Because is greater than 0, so the value of VAR1 will equal FALSE (0)

<=

Type: operator
Grammar: expression1 > expression2
Description: if expression 1 is less than or equal to expression 2, returns TRUE, otherwise returns FALSE.
Parameter:
expression1: any valid expression
expression2: any valid expression
For example：
VAR1=1<=1
The value of VAR1 will be equal to TRUE (-1).

>>

Type: operator
Grammar: expression1, expression2
Description: right shift.
Parameter:
expression1: any valid expression
expression2: any valid expression
For example：

```
>>print 8>>1
4
```

<<

Type: operator
Grammar: expression1, expression2
Description: left shift.
Parameter:
expression1: any valid expression
expression2: any valid expression
For example：

```
>>print 8<<1
16
```

\

Type: operator
Grammar: expression1\ expression2

Description: integer division.
Parameter:
expression1: any valid expression
expression2: any valid expression
For example：

```
>>print 10 \ (1+2)
3
```

$

Type: special character
Grammar: $hexnum
Description: indicates that the following data is in the 16 hexadecimal format
For example

```
>>print $F
15
```

AND

Type: operator
Grammar: expression1 AND expression2
Description: by bitwise and operator, only the integer part is operated

AND		Result
0	0	0
0	1	0
1	0	0
1	1	1

Parameter:
expression1: any valid expression
expression2: any valid expression
For example：

```
>>print 1 and 2
0
```

OR

Type: operator
Grammar: expression1, OR, expression2
Description: by bit or operator, only the integer part is operated

OR		Result
0	0	0
0	1	1

1	0	1
1	1	1

For example：
```
>>print 1 and 2
3
```

NOT

Type: operator
Grammar: NOT expression2
Description: bitwise negation operator，only can modify integers part, and be careful when “NOT” is used in some special integers, such as "ON".
For example：
```
>>print NOT 1
-2
```

XOR

Type: operator
Grammar: expression1, XOR, expression2
Description: logical XOR or operator, only operate integer part

XOR		Result
0	0	0
0	1	1
1	0	1
1	1	0

For example：
```
>>print 1 xor 1
0
```

EQV

Type: operator
Grammar: expression1 EQV expression2
Description: by bit or operator, only operate the integral part.

EQV		Result
0	0	1
0	1	0
1	0	0
1	1	1

For example：
>>print 1 eqv 1
1

MOD

Type: operator
Grammar: expression1, MOD, expression2
Description: find the remainder
Parameter
Expression1: any valid expression that takes an integer part.
Expression2: any valid expression that takes an integer part.
For example：
>>print 10 MOD (1+2)
1

ABS

Type: mathematical function
Grammar: ABS (expression)
Description: Take the absolute value.
Parameter: expression: any valid expression
For example：
ABS (-11): The result is 11

TAN

Type: mathematical function
Grammar: TAN (expression)
Description: the tangent trigonometric function, the input parameter is the radian unit.
Parameter: expression: any valid expression
For example:
Tan (pi/3) "the result is 1.732

ATAN

Type: mathematical function
Grammar: ATAN (expression)
Description: the arc tangent trigonometric function, the return value is radian unit.

Parameter: expression: any valid expression
For example: atan (1) - the result is 0.7854 = (45/180) *pi

ATAN2

Type: mathematical function
Grammar: ATAN2 (y, x)
Description: the arc tangent trigonometric function, the return value is radian unit
Parameter：expression: any valid expression
For example：

print atan2(1,0) The result is 1.5708

ASIN

Type: mathematical function
Grammar: ASIN (expression)
Description: anti sine trigonometric function. The return value is radians
Parameter: expression: any valid expression
For example: "print asin (0.5)", the result is 0.52360

SIN

Type: mathematical function
Grammar: SIN (expression)
Description: sine trigonometric function, the input parameter is radian unit
Parameter: expression: any valid expression
Example: "print sin (pi/6)", the result is 0.5000

COS

Type: mathematical function
Grammar: COS (expression)
Description: cosine trigonometric function, the input parameter is radian unit
Parameter: expression: any valid expression
For example: print cos (pi/3). The result is 0.5000

CRC16

Type: mathematical function
Grammar: CRC16 (arrayname, index, size[, inital] [[poly]])

Description: CRC16 CCITT calculation
Parameter：arrayname: The array in which the data is stored, one byte in one position
Index: The initial index of the array where the data store is located
Size: Number of bytes calculated
Initial: CRC calculates initial values, defaults to $FFFF
Poly: Polynomial, only supports Modbus $A001 and CCITT $1021, the default $A001
For example：
TABLE (0, $FE, $48, $06, $00, $6D, $00, $00, $00) "8 data stored in TABLE CRCVALUE = CRC16 (TABLE, 0, 8)" the calculation of CRC, results of $1A0D TABLE (8) = CRCVALUE 256 computing "CRC plus behind the data, big endian mode (TABLE 9) = CRCVALUE and $ff

ACOS

Type: mathematical function
Grammar: ACOS (expression)
Description: cosine trigonometric function, the return value is radian unit.
Parameter: expression: any valid expression
Example: "print ACOS (0.5)", the result is 1.04720

EXP

Type: mathematical function
Grammar: exp ([base, [expvalue])
Description: exponential function
Parameters: base：base number , the default value is e
Expvalue： index
Example 1:
Print exp (2,4) - the result is 16 (2*2*2*2)
Example 2:
Print exp (1). The result is 2.7183

SQR

Type: mathematical function
Grammar: SQR (expression)
Description: square root function
Parameter: expression: any valid expression
Example: a= SQR (4)
Print a 'the result is 2

LN

Type: mathematical function
Grammar: LN (expression)
Description: natural logarithmic function
Parameter: expression: any valid expression
For example: a= ln (1)
Print a "the result is 0

LOG

Type: mathematical function
Grammar: LOG(expression)
Description: Logarithm, the base is 10.
Parameter: expression: any valid expression
For example: a= log (100)
Print a 'the result is 2

CLEAR_BIT

Type: mathematical instruction or function
Instruction Grammar: CLEAR_BIT (bit#, vr#)
Function Grammar: CLEAR_BIT (bit#, int)
Description: bit operations, only for integers, modify the corresponding bit to 0; the function only returns the result after being modified.
Parameter: bit#: bit number: 0-31
Vr: to operate the VR variable number
Int: The expression needs to be operated, taking the integer part
Example 1:

```
      a=1
        Print clear_bit (0, a) ’returns the value after cleaning the bit 0 in
        ,value of the a will not
     Change
     0       'result is 0, a is 1,
    Print a
     1
```

For example, 2:

```
    clear_bit (0,23)
    The zeroth bit of VR (23) will be cleared (set to 0)
```

READ_BIT

Type: mathematical function
Grammar: READ_BIT (bit#, vr#)
Description: bit operations, only for integers, read the corresponding bit
Parameter: bit#: bit number: 0-31
vr#: VR variable number to be operated
For example：
dim a
a=read_bit (0,13): reads the zeroth bit of VR (13) and is assigned to a
print a: to output the a value

READ_BIT2

Type: the mathematical function, provides support for the firmware version after 20130813
Grammar: READ_BIT2 (bit#, int)
Description: bit operations, only read the corresponding bit status on integers
Parameter: bit#: bit number: 0-31
Int: The expression needs to be operated, taking the integer part
For example：
dim a,b b=1
a=read_bit2 (0, b): reads the zeroth bit of the B and grants it to a
print **a:** to output the value of **a**

SET_BIT

Type: a mathematical instruction or function
Instruction Grammar: SET_BIT (bit#, vr#)
Function Grammar: SET_BIT (bit#, int)
Description: bit operations, only for integers, modify the corresponding bit to 1
Parameter: bit#: bit number: 0-31
vr#: VR variable number to be operated
int: The expression needs to be operated, taking the integer part
For example 1:
a=1
Print set_bit (1, a) returns the result of setting the zeroth bit of a, and the value of a remains unchanged.
3
Print a
1

for example 2:
set_bit (0,23): The zeroth bit of VR (23) will be changed to 1

FRAC

Type: mathematical function
Grammar: FRAC (expression)
Description: returns the portion of the decimal that is always greater than 0
Parameter: expression: the number of operating
For example：
```
        a=frac(1.235)
        Print a 'the result is 0.235
```

INT

Type: mathematical function
Grammar: INT (expression)
Description: returns the integer part
Parameter: expression；any valid expression
For example：
```
     a=int (1.235)
     print a ', and the result is 1.
       Int (-1.1)'
     -2', Because the fractional part always be positive number
```

SGN

Type: mathematical function
Grammar: SGN (expression)
Description: returns the symbol
            1：More than 0
            0：Equal to 0
          -1：less than 0
Parameter: expression：any valid expression
For example：
```
      a=SGN(-1.235)
     Print a "the result is -1
```

TURN_POSMAKE

Type: mathematical function
Grammar: TURN_POSMAKE (tablenum, posx, posy, disR, tableout)

Description: the midpoint of the rotation function is calculated using the function, the rotation of the forward is consistent with the positive of the XY (right hand rule):
Parameter:
Tablenum: rotation function stores the table number
Posx: coordinates of the X directions
Posy: coordinates Y direction.
disR: The relative deviation of axis of rotation.
Tableout: Coordinate storage after computation.
Reference to: MCIRC_TURNABS
For example：

B_SPLINE

Type: mathematical function
Grammar: B_SPLINE (type, data_start, points, data_out, ratio)
Description: B data smoothing in TABLE.
Parameter:
Type: type, currently only supported 1-b spline.
Data_start: graphic data in the starting position of TABLE.
Points: number of graphic data.
Data_out: smoothed graphics data in the starting position of TABLE
Ratio: smoothness ratio of the function of B_SPLINE. The number of smoothness is points * ratio.
For example：
B_SPLINE (1,0,10100,10) "smoothing a 10 point graphical data, the source graphics point is in the Table position from"
0 to 9, smoothing the data at 100, and the smoothed data is stored from the Table address 100

CHR

Type: String Functions
Grammar: CHR (expression)
Description: return ASCII print, only for PRINT.
Parameter: expression: any valid expression
For example：

```
>>PRINT CHR(66);
  B
```

TOSTR

Type: String Functions
Grammar: TOSTR (VAR1, [N], [DOT])
Description: formatted output function any valid expression
Parameter: VAR1: any valid expression
N: output total digits, including decimal points and symbol bits. When N sets negative numbers, it is right justified
DOT: The number of decimal output, when the N is too small no decimal position, not decimal output.
For example：

```
>> print tostr(-100,6,2)
   -100.0
```

HEX

Type: String Functions
Grammar: HEX (expression)
Description: returns sixteen decimal format, used only for PRINT.
Parameter: expression: any valid expression, taking only the integer part
For example：

```
>>PRINT HEX (15);
  F "Sixteen
```

STRCOMP

Type: String Functions
Grammar: STRCOMP (STR1, STR2)
Description: string comparison function, returns “>0”,“=0” or“<0”,based on the situation of two strings
Parameter: STR1: string 1
Str2: string 2
For example：

```
DIM AAA (10) AAA = "ABC""
>>PRINT strcomp (AAA, ABC) 0
```

VAL

Type: String Functions
Grammar: VAL (STR1)
Description: string conversion to numerical

Parameter: STR1: string symbol
For example：

```
VAR1 = VAL("123")
?VAR1
123
```

IEEE_IN

Type: mathematical function
Grammar: IEEE_IN (byte0, byte1, byte2, byte3)
Description: combine four bytes into a single precision floating-point number

Parameter: byte0 - byte3 four bytes
For example：

```
VAR = IEEE_IN(VR(10),VR(11),VR(12),VR(13))
```

IEEE_OUT

Type: mathematical function
Grammar: byte_n = IEEE_OUT (VAR, n)
Description: take a byte from a single precision floating-point number
Parameter: VAR: single precision floating-point number
N 0-3: the bytes will be taken
For example：

```
VAR = IEEE_OUT(VR(1),2)
```

PI

Type: constant
：3.14159

TRUE

Type: constant
：-1

FALSE

Type: constant
: 0

ON

Type: constant
: 1

OFF

Type: constant
: 0

SCAN_EVENT

Type: mathematical function
Grammar: RET = SCAN_EVENT (expression)
Description: detects the content change of an expression, off- on returns 1, on-off returns -1, and returns 0. (more than 150810 firmware version support)
Parameter: expression: any valid expression, and the result is converted to BOOL type
For example:

```
While 1
     Iret = scan_event(in(0))
     If Iret > 0 then
          Print "IN0 On""
     Else if Iret <0 then
          Print "IN0 Off"" End if
     Wend
 End if
```

Chapter 6   Axis Parameters And AXES Status

Axis parameter modification Grammar: speed = value 'BASE axis. You can also use speed axis (axisnum) = value to force a particular axis , in which axis can be omitted and changed to speed (axisnum) = value.

You can go through SPEED=value1, value2... To set parameters for multiple BASE axes at once.

The axis parameters can be written and readable. When read, the axis can also be omitted: VAR1 = speed (axisnnum).

The axis status is generally read-only, and the values will change at any time, except for MPOS, DPOS, etc, which can be modified directly.

When interpolation is performed, the parameters of the spindle (BASE axis) are used as interpolation parameters.

ACCEL

Type: axis parameter
Grammar ; VAR1 = ACCEL, ACCEL = expression
Description: axis acceleration, in units of units/s/s, is used as the acceleration of interpolation motion when multi axis motion is applied.
For example:

```
BASE(1,2,3,4)
ACCEL=100, 100, 100, 100 ’set the axes 1,2,3,4 acceleration
  ACCEL(0)=200    ’set the axes 0 acceleration
  PRINT ACCEL(0)
```

⚠ It is suggested that acceleration and deceleration should be set before the movement, and the movement should not be modified. The adjustment in the motion will lead to the change of the speed curve.

ADDAX_AXIS

Type: axis state
Grammar: VAR1 =ADDAX_AXIS
Description: The axis number of the axis superimposed on the ADDAX instruction. -1 indicates on overlap.
For example:

```
PRINT ADDAX_AXIS(0):’View axis number superimposed on axis 0.
```

ALM_IN

Type: axis parameter
Grammar: VAR1=ALM_IN, ALM_IN = expression
Description: drive alarm corresponding to the input number, -1 invalid
Reference: FWD_IN, REV_IN, FS_LIMIT,RS_LIMIT
For example：

```
ALM_IN = 10 ’define the alarm signal to the input port 10.
  INVERT_IN(10,ON)
```

⚠ DATUM_IN, JOG, FWD_IN, REV_IN, ALM_IN, FHOLD_IN and other special input signals, are in the input OFF, that there is a signal input, to the contrary, the effect can be used INVERT_IN. (ECI series controller except).

ATYPE

Type: axis parameter
Grammar: VAR1 = ATYPE, ATYPE = expression
Description: axis type setting, can be set only to have the characteristics of axis (axis characteristics can query the hardware manual or using ZDevelop software after connecting the controller view controller state). It is better to set the ATYPE during the procedure initialization, To use the ZCAN extension axis should set the AXIS_ADDRESS first, after that a delay of 2 tick is needed before calling motion instruction, due to bus bandwidth limit, Setting ZCAN extension axes number must not exceed 2.

⚠ For some of the products with independent encoders, can use the corresponding virtual axis as axis encoder axis, such as ZMC206 , its encoder can be controlled by axis 6-11,the details of controller state can be checked after connecting the Zdevelop to the controller.

Atype type	description
0	Virtual axis
1	Stepping or servo in pulse direction mode
2	Servo of analog signal control mode
3	Quadrature encoder
4	Step+encoder
6	Pulse direction encoder, used for hand wheel input
7	Pulse direction mode stepping or servo + EZ signal input
8	ZCAN extended pulse direction mode stepping or servo
9	ZCAN extended orthogonal encoder
10	ZCAN encoder with extended pulse direction mode
65	ECAT pulse type
66	ECAT speed closed loop
67	ECAT moment closed loop
70	ECAT custom operation, only read the encoder

Reference to: AXIS_ADDRESS,INVERT_STEP
For example；

```
BASE(0 , 1 , 2 , 3)
ATYPE = 1,1,1,1
```

AXISSTATUS

Type: axis status
Grammar: VAR1 = AXISSTATUS

Description: various states of the axis.

position	illustrate	value
1	Follow-up error overrun alarm	2
2	Error communicating with remote axis	4
3	Remote drive error reporting	8
4	Forward hard limit	16
5	Reverse hard limit	32
6	Find the origin	64
7	HOLD speed hold signal input	128
8	Error of follow up error overrun	256
9	Over forward soft limit	512
10	Over negative to soft limit	1024
11	CANCEL in execution	2048
12	Pulse frequency exceeding MAX_SPEED limit. Need to modify speed down or modify MAX_SPEED	4096
14	Manipulator coordinate error	16384
18	Power exception	262144
22	Alarm signal input	4194304
23	The axis is in a state of suspension	8388608

For example：

```
    VAR = READ_BIT2(4,AXISSTATUS)
     Print VAR
     1        'A hard limit has occurred
```

AXIS_ADDRESS

Type: axis parameter

Grammar: VAR1 = AXIS_ADDRESS, AXIS_ADDRESS = expression

Description: Axis axis extended address configuration, ZCAN extension axis, CAN ID (low 5 bits combined by the dial), more than 6 extended local axis number module.

Bit 0-4	Expansion board CANID, 0-31
Bit 6-7	The local axis number on the extension board is.0-1

ECat axis configuration:

Bit 0-15	Drive number +1, 0- automatically specified
Bit 16-31	SLOT number

The 4 series controller supports local pulse or encoder axis remapping, firmware more than 160608 version support.

Bit 0-15	Mapped local pulse axis number

Bit 16-31	-1

Drive number - in line order, number 0 from -1 to ECAT drive.
Reference to: ATYPE
For example, 1:
AXIS_ADDRESS (6) =2 + (32*1) 'set axis 6 is mapped to the ZCAN extension module ID2 axis 1
ATYPE (6) =8

⚠ Must be set before the AXIS_ADDRESS, and then set the ZCAN extension axis type ATYPE modified must be reset ATYPE.
For example, 2:
AXIS_ADDRESS (0) =0 +1 ’slot0 first drive, drive number 0.
ATYPE (0) =65 ' axis 0 is configured as a ECAT binding mode, the first drive, drive number 0
For example, 3:
AXIS_ADDRESS (0) =0 ‘Automatically specify the slot0 drive (not recommended, recommended example 2)
ATYPE (0)=65 'Axis 0 is configured as ECAT mode
For example, 4:
AXIS_ADDRESS (16) = (-1<<16) + 0 ’ binding local pulse axis are 0
ATYPE (16) =65 ' Axis 16 is configured as the pulse axis and USES local pulse axis 0

AXIS_ENABLE

Type: axis parameter
Grammar: AXIS_ENABLE = 1/0
Description: axis settings.
Reference to: WDOG
For example:
AXIS_ENABLE = 1

AXIS_STOPREASON

Type: axis state
Grammar: AXIS_STOPREASON = 0
Description: historical reasons to stop axis latch, write 0 clear, according to a latch, meaning the same with AXIS_STOPREASON. More than 20150731 version support it.
Reference：AXISSTATUS
For example:
If AXIS_STOPREASON and (512+1024) then
Print “axis el stopped”

End if

CLUTCH_RATE

Type: axis parameter
Grammar: clutch_rate= value
Description: connect: connection speed, ratio/ seconds; when set to 0, according to the acceleration parameters of axis speed / track connection, more suitable for hand motion (when the speed is not high enough so may lead to continue the campaign for a period of time to the end).
Reference to: CONNECT
For example:

clutch_rate = 0 "tracks connections based on the speed / acceleration parameters of the axis."

CLOSE_WIN

Type: axis parameter
Grammar: CLOSE_WIN = POS
Description: the end of coordinate range of latch trigger.
Reference to: REGIST, OPEN_WIN

CORNER_MODE

Type: axis parameter
Grammar: CORNER_MODE = mode
Description: corner, deceleration, and other mode settings.
Mode: different bits represent different meanings.

Positio n	value	Description
0	1	Reserve
1	2	Automatic corner deceleration, this parameter is effective before the MOVE function call
2	4	Reserve
3	8	Automatic small round speed limit
4	16	Reserve
5	32	Automatic chamfer setting. This parameter is modified before the MOVE function call. This MOVE motion is

		automatically processed with the preceding MOVE motion, and the chamfer radius is referenced to ZSMOOTH. This chamfer is for all axes of the interpolation, and firmware supports after 20150701.

Reference to: MERGE, *SP, STOP_ANGLE, DECEL_ANGLE, FULL_SP_RADIUS, ZSMOOTH. Deceleration speed is limited by FORCE_SPEED

For example, 1:

```
BASE (0)
  CORNER_MODE=2 +8 "start with a small round corner deceleration
  speed
  DECEL_ANGLE = 15 * (PI/180)" set the      starting deceleration angle
STOP_ANGLE = 45 * (PI/180) "set the end deceleration angle
FULL_SP_RADIUS = 5" set the maximum radius of small speed
```

For example, 2:

```
BASE (0)
CORNER_MODE=32'starts the chamfer
ZSMOOTH = 10    'chamfer reference radius
MOVE (100)
 MOVE (0,100)' automatic chamfer between two straight lines above
```

CREEP

Type: axis parameter
Grammar: VAR1 = CREEP, CREEP = expression
Description: Axis return zero low speed, unit is units/s.
Reference to: DATUM
For example:

```
Base (0)
Speed = 100,
  Creep = 10
  Datum (4) 'First press 100 to get back zero, after the origin, press 10 to
  leave the origin
```

DAC

Type: axis parameter
Grammar: VAR1 = DAC, DAC = expression
Description: servo axis DA direct control, unit DA module scale, 12 or 16 bits.
Reference to: SERVO

DATUM_IN

Type: axis parameter
Grammar: VAR1 = DATUM_IN, DATUM_IN = expression
Description: input number of original switch signal, and -1 is invalid.
Reference to: DATUM
For example：

```
    Base (0,1,2,3)
   Datum_in = 6 ,7,8,9   The axes 0, 1, 2, and 3 origin inputs correspond to
                         input ports 6, 7, 8, 9
      Invert_in(6,on)
      Invert_in(7,on)
      Invert_in(8,on)
      Invert_in(9,on)
```

⚠ DATUM_IN, JOG, FWD_IN, REV_IN, FHOLD_IN and other special input signals, are in the input OFF, that there is a signal input, to the contrary, the effect can be used INVERT_IN. (ECI series controller except).

DECEL

Type: axis parameter
Grammar: VAR1 = DECEL, DECEL = expression
Description: axial speed, units/s/s, when the multi axis movement, as interpolation movement deceleration. When set to 0, the acceleration is equal to the acceleration ACCEL value, and the symmetrical acceleration and deceleration is performed.
Reference to: ACCEL, FASTDEC
For example:

```
    BASE (1,2,3,4)
    DECEL =100, 100, 100, 100 'Set the axis 1, 2, 3, 4 deceleration
    DECEL (0)=200 'Set the deceleration of the axis 0
    PRINT DECEL (0)
```

⚠ It is suggested that the acceleration and deceleration should be set before the movement, and the movement should not be modified. The adjustment in the motion will lead to the change of the speed curve.

DECEL_ANGLE

Type: axis parameter
Grammar: VAR1 = DECEL_ANGLE, DECEL_ANGLE = expression

Description: the minimum corner of the start deceleration, in radians
Reference to：STOP_ANGLE，CORNER_MODE
For example：

```
CORNER_MODE=2
DECEL_ANGLE = 25 * (PI/180)
STOP_ANGLE = 90 * (PI/180)
```

DPOS

Type: axis status (writable)
Grammar: VAR1 = DPOS
Description: the virtual coordinate position of the axis, or the demand location, writes that the DPOS will automatically be converted to OFFPOS offset.
Reference to: MPOS, ENDMOVE, OFFPOS, DEFPOS
For example:

```
Dpos(0) = 0 ‘Axis 0 coordinate offset to 0
?*dpos ’Command line to view current DPOS
0 0 0 0 0 0 0 0 0 0 0 0
```

D_GAIN

Type: axis parameter
Grammar: VAR1 = D_GAIN
Description: differential gain, non-analog servo does not support.
Reference to: P_GAIN, I_GAIN, OV_GAIN, VFF_GAIN

ENCODER

Type: axis status
Grammar: VAR1 = ENCODER
Description: original values of encoder hardware register, raw values, internal parameters, only can be read when the ATYPE is configured to encoder request use the encoder's ATYPE to read.
Reference to：ATYPE MPOS
For example：

```
?*encoder
0 0 0 0 0 0 0 0 0 0 0 0
```

ENCODER_STATUS

Type: axis status

Grammar: VAR1 = ENCODER_STATUS
Description: encoder EA, EB, EZ status

Position	value	Description
0	1	EA state
1	2	EB state
2	4	EZ state

Reference to: ATYPE MPOS

ENDMOVE

Type: axis status
Grammar: VAR1 = ENDMOVE
Description: the axis of the current movement of the final target position.
Reference to: DPOS, MPOS, ENDMOVE_BUFFER
For example:

```
    Base(0)
    Speed = 10
    Dpos = 0
    Move(100)
    Move(200)
     Print ENDMOVE(0)    'When running to this statement, the current
  movement is move (100)
    100
```

⚠ VMOVE, DATUM and other non-fixed distance movements, ENDMOVE is not accurate, or always varies.

ENDMOVE_BUFFER

Type: axis state
Grammar: VAR1 = ENDMOVE_BUFFER
Description: the axis of the current and buffering of all movements of the final target position, which can be used for absolute and relative position conversion.
Reference to: DPOS, MPOS, ENDMOVE
For example:

```
   Base(0)
   Speed = 10
    Dpos = 0
    Move(100)
    Move(200)
    Print ENDMOVE_BUFFER (0)
    300
```

⚠ For VMOVE, DATUM and other non-fixed distance movements, ENDMOVE_BUFFER is not accurate, or always varies.

ENDMOVE_SPEED

Type: axis parameter
Grammar: VAR1 = ENDMOVE_SPEED, ENDMOVE_SPEED = expression
Description: Custom speed SP, movement end velocity, this parameter is
brought into motion buffer.
Reference to：FORCE_SPEED，*SP，STARTMOVE_SPEED
For example：

```
FORCE_SPEED=150
ENDMOVE_SPEED=10
MOVESP(20)
```

ERRORMASK

Type: axis Parameter
Grammar: VAR1 = ERRORMASK, ERRORMASK = expression
Description: and AXISSTATUS do and operation to determine which errors
need to be closed WDOG.
Reference to: AXISSTATUS, WDOG,

FAST_JOG

Type: axis parameter
Grammar: VAR1 = FAST_JOG, FAST_JOG = expression
Description: the number of the fast JOG input, and -1 is invalid.
Reference to: REV_JOG, FWD_JOG, SPEED, JOGSPEED

⚠ DATUM_IN, JOG, FWD_IN, REV_IN, FHOLD_IN and other special input signals, are in the input OFF, that there is a signal input, to the contrary, the effect can be used INVERT_IN. (ECI series controller except).

FASTDEC

Type: axis parameter
Grammar: VAR1 = FASTDEC, FASTDEC= expression
Description: fast deceleration, unit units/s/s, automatically used when
CANCEL or abnormal stop, automatically set as DECEL when set to 0
value.

Reference to：DECEL
For example：

```
    Base(0)
    FASTDEC = 1000
```

FE
Type: axis status
Grammar: VAR1 = DPOS
Description: follow error, the value equal to DPOS-MPOS.
Reference to: MPOS, DPOS

FE_LIMIT

Type: axis parameter
Grammar: VAR1 = FE_LIMIT, FE_LIMIT = expression
Description: maximum allowed follow error value.
Reference to: FE, FE_RANGE

FE_RANGE

Type: axis parameter
Grammar: VAR1 = FE_RANGE, FE_RANGE = expression
Description: follow error when warning.
Reference to: FE, FE_LIMIT

FHOLD_IN

Type: axis parameter
Grammar: VAR1 = FHOLD_IN, FHOLD_IN = expression
Description: keep input corresponding input point number, -1 invalid.
Reference to: FHSPEED

⚠ DATUM_IN, JOG, FWD_IN, REV_IN, FHOLD_IN and other special input signals, are in the input OFF, that there is a signal input, to the contrary, the effect can be used INVERT_IN. (ECI series controller except).

FHSPEED

Type: axis parameter
Grammar: VAR1 = FHSPEED, FHSPEED = expression
Description: AXIS holding speed, the speed at which FHOLD_IN is pressed, in units of units/s,.
Reference to: FHOLD_IN, SPEED

FORCE_SPEED

Type: axis parameter
Grammar: VAR1 = FORCE_SPEED, FORCE_SPEED = expression
Description: forced speed custom speed SP movement, the movement parameters were brought into the buffer, the unit is units/s, when the FORCE_SPEED is greater than SPEED, SPEED will also force limited top speed (after 140716 version of SPEED does not work, if the time required to enter) the FORCE_SPEED has been reduced to the corresponding speed, set the STARTMOVE_SPEED.
Reference to: ENDMOVE_SPEED, *SP, SPEED
For example：

```
BASE(0)
SPEED = 1000
FORCE_SPEED=150
ENDMOVE_SPEED=10
MOVESP(20)              The speed is 150
```

FS_LIMIT

Type: axis parameter
Grammar: VAR1 =FS_LIMIT, FS_LIMIT = expression
Description: positive soft limit position, the unit is units, when the FS_LIMIT is greater than REP_DIST, parameter has no effect, positive soft limit is prohibited. When you cancel the soft limit, it is recommended that you do not modify the value of the REP_DIST and set the FS_LIMIT to a larger value
Reference to：RS_LIMIT, FWD_IN, REV_IN，
For example：

```
FS_LIMIT = 200 'Set the forward soft limit to 200
FS_LIMIT = 200000000 'Cancel forward soft limit (REP_DIST default initialization value is 100000000 )
```

FULL_SP_RADIUS

Type: axis parameter
Grammar: VAR1 = FULL_SP_RADIUS, FULL_SP_RADIUS = expression
Description: maximum radius small speed, the unit is units; set up automatic chamfering for reference radius chamfer.
Reference to: CORNER_MODE, FORCE_SPEED
For example:

```
CORNER_MODE=8 'Start small round speed limit
FULL_SP_RADIUS = 5      'Set small round speed maximum radius
```

FWD_IN

Type: axis parameter
Grammar: VAR1 = FWD_IN, FWD_IN = expression
Description: forward hardware limit switch, corresponding input point number, -1 invalid.
Reference to: REV_IN, FS_LIMIT, RS_LIMIT
For example：

```
 Base(0,1,2,3)
FWD_IN = 6 ,7,8,9    'axes 0, 1, 2, 3, forward hardware limits correspond to
inputs 6, 7, 8, 9
```

DATUM_IN, JOG, FWD_IN, REV_IN, FHOLD_IN and other special input signals, are in the input OFF, that there is a signal input, to the contrary, the effect can be used INVERT_IN. (ECI series controller except).

FWD_JOG

Type: axis parameter
Grammar: VAR1 = FWD_JOG, FWD_JOG= expression
Description: forward JOG input corresponding input point number, -1 invalid.
Reference to: REV_JOG

_IN, JOG, FWD_IN, REV_IN, FHOLD_IN and other special input signals, are in the input OFF, that there is a signal input, to the contrary, the effect can be used INVERT_IN. (ECI series controller except).

I_GAIN

Type: axis parameter
Grammar: I_GAIN = expression
Description: integral gain, non-analog servo, non-support.
Reference to: P_GAIN, D_GAIN, OV_GAIN, VFF_GAIN

IDLE

Type: axis status
Grammar: VAR1 = IDLE
Description: a single axis movement ends and returns TRUE.

Reference to: LOADED, WAIT, IDLE, MTYPE

For example 1 ：

```
     If IDLE(0) then      ‘If the axis 0 stops
        Base(1)
        Move(100)
   End If
```

For example 2 ：

```
    Base(0,1)
    Move(100,100)
    Base(2,3)
    Move(200,200)
     Wait unitl idle(0) and idle(1) and idle (2) and idle(3)    ’Wait for axes 0, 1, 2,
   3 to stop
```

INVERT_STEP

Type: axis parameter

Grammar: INVERT_STEP = mode

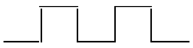
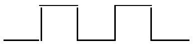
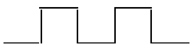
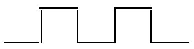
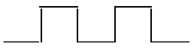
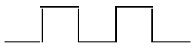
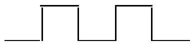
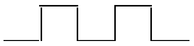
Description: step pulse output mode setting.

Parameter: mode (default 0)

   low 8 bit (bit 0- bit 7), the mode value is as follows.

  0-3 pulse direction mode. 4-7 double pulse mode (partial controller version is not supported, detailed consult manufacturer), 8/9 AB output (part of controller customization)

The levels corresponding to each mode are as follows:

Mode value	Forward motion		Negative motion	
	Pulse line	Direction line	Pulse line	Direction line
0		High		Low
1		High		Low
2		Low		High
3		Low		High
4		High	High	
5	High			High
6		Low	Low	
7	Low			Low

8	AB output, customization			
9	AB output, customization			

High 8 bit (bit 8- bit 15) indicates directional change protection time, unit microsecond: 0-255.

If the mode setting is incorrect, the stepper motor may lose a position of one step when it changes the direction in the commutation, and when if it is not certain how to set   the stepper motor is not set, a the protection time setting around of about 100 microseconds can be set is suitable .

Reference to：STEP_RATIO

For example：

INVERT_STEP = 256*100 ’Set the protection time of 100 microseconds, the mode is 0.

INTERP_FACTOR

Type: axis parameter

Grammar: INTERP_FACTOR = 0/1

Description: when the interpolation axis is involved in the speed calculation, default participation (1). This parameter acts only on the third axes of the arbitrary axis of the straight line and the helix. You can't set all the axes of the actual movement of the interpolation 0, which will cause the actual speed to be infinite.

Reference to：MOVE MOVEABS

For example：

INTERP_FACTOR = 0

JOGSPEED

Type: axis parameter

Grammar: JOGSPEED= value

Description: the speed at JOG, units/s. When the REV_JOG/FWD_JOG is set and pressed corresponding to the input point, the motor will move at a slow speed of JOGSPEED.

Reference to: REV_JOG, FWD_JOG

LINKAX

Type: axis status

Grammar: VAR1 = LINKAX

Description: returns the reference axis number of the current connection

movement.
Reference to: CAMBOX, MOVELINK, CONNECT
For example:

```
base(0)
cambox(0,300,1,80,2,2,100)
val = linkax
print val
2
```

LOADED

Type: axis status
Grammar: VAR1 = LOAED
Description: in addition to the current movement, there is no buffered instruction, returning ture at this point.
Reference to: IDLE, WAIT, LOADED

LSPEED

Type: axis parameter
Grammar: VAR1 = LSPEED, LSPEED = expression
Description: axis starting speed, at the same time to stop the default rate, 0, unit is units/s, when the multi axis motion, as initial velocity interpolation motion.
When you need to pursue efficiency, you can consider setting the starting speed
For example：

```
Base(0,1)
Lspeed = 0,0
>>print lspeed
0
```

HOMEWAIT

Type: axis parameter
Grammar: VAR1 = HOMEWAIT, HOMEWAIT = expression
Description: pulse mode of servo drive, return to origin movement, when looking back, waiting for some time. The wait time for this parameter setting is in milliseconds.
For example：

```
>>print HOMEWAIT
  100
```

MARK

Type: axis status
Grammar: VAR1 = MARK
Description: Returns whether the latch event is generated.
Reference to: REG_POS, REGIST, MARKB
For example：

```
 BASE(0)
 MOVE(100)
 REGIST(3) 'Rising edge R0
 WAIT UNTIL MARK
```

MARKB

Type: axis status
Grammar: VAR1 = MARKB
Description: returns the latch event that corresponds to the second latch location.
Reference to: MARK, REG_POSB, REGIST

```
   BASE(0)
   MOVE(100)
   REGISTB(3) 'Rising edge R0
   WAIT UNTIL MARKB
```

MAX_SPEED

Type: axis parameter
Grammar: MAX_SPEED = value
Description: the maximum frequency limit for pulse output, once found above this setting, forces, and sets AXISSTATUS.
For example：

```
     Base(0)
     MAX_SPEED =500000    ‘Set pulse speed limit 500K
```

MERGE

Type: axis parameter
Grammar: MERGE = ON/OFF
Description: the movement of the front and back buffers is connected together without slowing down, used for continuous interpolation.
Reference to: *SP, CORNER_MODE

For example：
```
Base(0,1,2,3)
MERGE =ON    ‘Open the continuous interpolation state of the axis 0.
```

MOVES_BUFFERED

Type: axis status
Grammar: VAR1 = MOVES_BUFFERED
Description: returns the number of currently moving instructions that are buffered.
Reference to：LOADED, LIMIT_BUFFERED，REMAIN_BUFFER
For example：
```
Print MOVES_BUFFERED
0
```

⚠ Do not use MOVES_BUFFERED to reduce this parameter to determine if there is a residual buffer space. Special motion instructions may take up multiple buffer spaces, and REMAIN_BUFFER state functions are used to determine more accurately.

MOVE_CURMARK

Type: axis status
Grammar: VAR1 = MOVE_CURMARK
Description: returns the MOVE_MARK label for the current movement of the current axis.
Reference to：MOVE_MARK, MOVE_PAUSE
For example：
```
  MOVE_MARK =1
  MOVE(100)
  MOVE_MARK =2
  MOVE(200)
  MOVE_MARK =3
  MOVE(300)
   WAIT UNTIL MOVE_CURMARK = 2 ‘Wait for MOVE (200) to start, open
   the output port 1.
OP(1,ON)
```

MOVE_MARK

Type： axis parameter
Grammar：VAR1 = MOVE_MARK，MOVE_MARK = expression

Description: The MARK label of the next movement instruction, which is
written to the motion buffer along with the motion instructions.
With the MOVE_PAUSE command, you can pause at different
boundaries of MARK.

Reference to：MOVE_CURMARK, PAUSE

For example：

```
MOVE_MARK =1
MOVE(100)
MOVE_MARK =1
MOVE(100)
MOVE_MARK =2
MOVE(200)
MOVE_PAUSE (2) ‘Pause before MOVE(200)
```

MPOS

Type: axis status (write)

Grammar: VAR1 = MPOS

Description: axis measurement feedback position, unit is units, write MPOS
will automatically converted to OFFPOS offset.

Reference to：DPOS，ENDMOVE，OFFPOS，DEFPOS

For example：

```
Mpos(0) = 0 ‘MPOS offset to 0
?*mpos
0 0 0 0 0 0 0 0 0 0 0 0
```

MSPEED

Type: axis status

Grammar: VAR1 = MSPEED

Description: axis measurement, feedback, position, speed, unit units/s.

Reference to: MPOS, VP_SPEED

MTYPE

Type: axis status

Grammar: VAR1 = MTYPE

Description: the current type of motion instruction that, when interpolating
linkage, moves from the axis always back to the main axis of the
instruction type.

MTYPE	Motor instruction type

0	IDLE (no move)
1	MOVE
2	MOVEABS
3	MHELICAL
4	MOVECIRC
5	MOVEMODIFY
6	MOVESP
7	MOVEABSSP
8	MOVECIRCSP
9	MHELICALSP
10	FORWARD
11	REVERSE
12	DATUMING
13	CAM
14	FWD_JOG
15	FWD_JOG
20	CAMBOX
21	CONNECT
22	MOVELINK
23	CONNPATH
25	MOVESLINK
26	MSPIRAL
27	MECLIPSE/ MECLIPSEABS/ MECLIPSESP/ MECLIPSEABSSP
28	MOVE_OP/MOVE_OP2 MOVE_TABLE MOVE_TASK
29	MOVE_DELAY
31	MSPHERICAL/ MSPHERICALSP
32	MOVE_P
33	CONNFRAME
34	CONNREFRAME

Reference to: REMAN_BUFFER, NTYPE

For example:

```
    if MTYPE = 1 then
    cancel
    End if
```

NTYPE

Type: axis status
Grammar: VAR1 = NTYPE
Description: the first instruction type following the current motion instruction.
When the linkage is interpolated, the movement type of the spindle is always returned to the spindle.
Reference to：MTYPE
For example：

```
    if NTYPE = 1 then
    cancel（2）
    End if
```

OFFPOS

Type: axis parameter
Grammar: VAR1 = OFFPOS, OFFPOS = expression
Description: offset modifies all coordinates, and this does not affect motion.
When the modification is complete, OFFPOS is reduced to 0.
Reference to：DPOS, MPOS, ENDMOVE, ENDMOVE_BUFFER,
For example：

```
   Base(0)
   Moveabs(1000)
   Wait idle
   OFFPOS = -1000 ’Coordinate offset 1000
   PRINT Dpos(0)
   >>PRINT Dpos(0)
     0
```

OPEN_WIN

Type: axis parameter
Grammar: OPEN_WIN = POS
Description: the end of coordinate range of latch trigger.
Reference to: REGIST, CLOSE_WIN

OV_GAIN

Type: axis parameter
Grammar: VAR1 = OV_GAIN, OV_GAIN = expression
Description: speed gain, non-analog servo does not support.
Reference to: P_GAIN, I_GAIN, D_GAIN, VFF_GAIN

P_GAIN

Type: axis parameter
Grammar: VAR1 = P_GAIN, P_GAIN = expression
Description: proportional gain, non-analog servo, non-support.
Reference to: D_GAIN, I_GAIN, OV_GAIN, VFF_GAIN

PP_STEP

Type: axis parameter
Grammar: PP_STEP = value
Description: the encoder input will be multiplied by this proportion, and this parameter effect is superimposed with the ENCODER_RATIO.
Reference to: ENCODER_RATIO

REG_POS

Type: axis status
Grammar: VAR1 = REG_POS
Description: returns the latched measurement feedback position (MPOS), units unit.
Reference to：REG_POSB，REGIST，MARK, MARKB
For example：

```
MOVE(100)
REGIST(3) 'Rising edge R0
WAIT UNTIL MARK
PRINT REG_POS
```

REG_POSB

Type: axis status
Grammar: VAR1 = REG_POSB
Description: returns the measurement feedback position (MPOS) of the latch register 2, and the units unit.

Reference to: REG_POS, REGIST, MARK, MARKB
For example：
```
    MOVE(100)
    REGIST(6) 'Rising edge EZ
    WAIT UNTIL MARKB
    PRINT REG_POSB
```

REMAIN

Type: axis status
Grammar: VAR1 = REMAIN
Description: returns the axis of the current movement that has not yet been completed, units unit.
Reference to：VECTOR_BUFFERED
For example：
```
     Print REMAIN
```

REMAIN_BUFFER

Type: special axis status
Grammar：VAR1 = REMAIN_BUFFER ([mtype]) AXIS(AXISNUM)
Description: returns the number of buffers that can be used. This state is special, has its own parameters, Therefore, AXIS cannot be omitted when correcting the axle number. When return 0, it means the current axis's buffer space is full. At this point, if you continue calling to call the motion instructions of the current axis, the task will block until the buffer has enough space .
Parameter: mtype: the MTYPE type of motion, if not filled, it will be calculated according to the instructions that occupy the most buffer.
Reference：LOADED, LIMIT_BUFFERED
For example：
```
     MOVETIMES = 0
     MOVETIMES < 1000
        IF REMAIN_BUFFER(1) > 0 THEN    "If there is a spare line space, a
        straight line motion instruction is called
       MOVE(10)
       MOVETIMES= MOVETIMES+1
     END IF
     WEND
```

REP_DIST

Type: axis parameter
Description: automatically rotate the axis DPOS and MPOS coordinates according to the REP_OPTION settings.
Reference to: REP_OPTION

REP_OPTION

Type: axis parameter
Grammar: VAR1 = REP_OPTION, REP_OPTION = opt
Description: coordinate repetition setting, currently only using bit 1.
Reference to: opt, bit by bit to mean different meanings.

position	value	description
0	1	0- loop range - REP_DIST to + REP_DIST, 1- loop range from 0 to REP_DIST.
1	2	Write 1 prohibits the repetitive movement of CAMBOX and MOVELINK, forbidden to restore to 0 after the entry into force.
2	4	Reserve
4	16	1- does not use REP_DIST, 0- uses REP_DIST.

Reference to：CAMBOX，MOVELINK，REP_DIST
For example：

```
MOVELINK (10, 10, 5, 5, 1, 4)
 WAIT UNTIL IN(0) = ON 'When the input 0 is valid, the loop is not
 allowed。
REP_OPTION = REP_OPTION and 2
```

REV_IN

Type: axis parameter
Grammar: VAR1 = REV_IN, REV_IN= expression
Description: negative to the hardware limit switch corresponding to the input point number, -1 invalid.
Reference to: FWD_IN, FS_LIMIT, RS_LIMIT
Reference to：

```
Base(0,1)
Rev_in = 0,1
Invert_in(0,on)
Invert_in(1,on)
```

⚠ DATUM_IN, JOG, FWD_IN, REV_IN, FHOLD_IN and other special input signals, are in the input OFF, that there is a signal input, to the contrary, the effect can be used INVERT_IN. (ECI series controller except).

REV_JOG

Type: axis parameter
Grammar: VAR1 = REV_JOG, REV_JOG= expression
Description: negative to the JOG input corresponding input point number, -1 invalid.
Reference to: FWD_JOG

⚠ DATUM_IN, JOG, FWD_IN, REV_IN, FHOLD_IN and other special input signals, are in the input OFF, that there is a signal input, to the contrary, the effect can be used INVERT_IN. (ECI series controller except).

RS_LIMIT

Type: axis parameter
Grammar：VAR1 = RS_LIMIT ，RS_LIMIT = expression
Description: negative to the soft limit position, the unit is units, when RS_LIMIT is less than -REP_DIST, the parameter does not work, the negative soft limit is prohibited. When you cancel the soft limit, it is recommended that you do not modify the value of the REP_DIST and set the FS_LIMIT to a smaller value.
Reference to：FS_LIMIT, FWD_IN, REV_IN ，
For example：

```
RS_LIMIT = -200 'Set negative to soft limit -200
 RS_LIMIT = -200000000      'Cancel the negative soft limit (REP_DIST
 initialization value is 100000000)
```

SERVO

Type: axis parameter
Grammar: VAR1 = SERVO, SERVO = ON/OFF
Description: closed loop switch setting.
Reference to: WDOG

SPEED

Type: axis parameter
Grammar: VAR1 = SPEED, SPEED = expression
Description: Axis speed, unit as units/s, when there is a multi-axes motion, it

will be regarded as a speed of interpolation movement. It will take effect as soon as the SPEED is modified it means the dynamic speed change is allowed. When the FORCE_SPEED of the SP command is higher than SPEED, the speed of SPEED will also take effect.

Reference：FORCE_SPEED

For example：

```
 BASE(1,2,3,4)
  SPEED =100, 100, 100, 100 ’Set the speed of axis 1, 2, 3, and 4 SPEED
  (0)=200 ’ Set the speed at axis 0
PRINT SPEED (0)
200
```

SRAMP

Type: axis parameter

Grammar: VAR1 = SRAMP, SRAMP = smoothms

Description: S curve setting.

Parameter: smoothms 0-250 MS, after setting, the acceleration and deceleration process will be longer, the corresponding time.

For example：

```
SRAMP = 30
```

STARTMOVE_SPEED

Type: axis parameter

Grammar: VAR1, =STARTMOVE_SPEED, STARTMOVE_SPEED = expression

Description: Custom speed SP, movement end velocity, this parameter is brought into motion buffer.

Reference to: FORCE_SPEED, *SP, ENDMOVE_SPEED

For example：

```
FORCE_SPEED=150
ENDMOVE_SPEED=10 ‘The end requires speed less than 10
 STARTMOVE_SPEED=10 ‘When entering, requires speed below 10,
 continuous interpolation will slow down ahead of time.
MOVESP(20)
```

STOP_ANGLE

Type: axis parameter

Grammar: VAR1 = STOP_ANGLE, STOP_ANGLE= expression

Description: slow down to the lowest minimum corner, in radians.

Reference to：DECEL_ANGLE，CORNER_MODE
For example：

```
CORNER_MODE=2
DECEL_ANGLE = 25 * (PI/180)
STOP_ANGLE = 90 * (PI/180)
```

TRANS_DPOS

Type: axis status
Description: reserved.

UNITS

Type: axis parameter
Grammar: VAR1 = UNITS, UNITS = expression
Description: pulse equivalent, specify the number of pulses per unit, support 5 bit decimal precision, modified coordinates, velocity and acceleration will change.
Reference to：SPEED, ACCEL, DPOS, MPOS, STEP_RATIO, ENCODER_RATIO
For example：

```
Units=10000/9 "Motor 10000 pulses per cycle, 9mm pitch per turn.
```

VECTOR_BUFFERED

Type: axis status
Grammar: VAR1 = VECTOR_BUFFERED
Description: returns the axis of the current currents movement and the buffer movement has not yet been completed, units unit, the multi axis interpolation is the vector distance.
Reference to：REMAIN

VECTOR_MOVED

Type: axis state
Grammar: VAR1 = VECTOR_MOVED VECTOR_MOVED=0
Description: return the axis of the current movement distance, units units, the multi axis interpolation is vector distance, with the best manual cleared.
Reference to：VECTOR_BUFFERED
For example：

```
VECTOR_MOVED=0
 MOVE(100)
```

```
WAIT IDLE
? VECTOR_MOVED
100
```

VFF_GAIN

Type: axis parameter
Grammar: VAR1 = VFF_GAIN, VFF_GAIN = expression
Description: feedforward gain of speed feedback, non analog servo does not support.
Reference to: P_GAIN, D_GAIN, I_GAIN, OV_GAIN

VP_SPEED

Type: axial state
Grammar: VAR1 = VP_SPEED
Description: returns the speed of the current movement of the axis, units/s, and when the axis is moving, the spindle returns the speed of the interpolation motion.
Reference to：MSPEED， SPEED
For example：

```
Print VP_SPEED
```

ZSMOOTH

Type: axis parameter
Grammar: VAR1 = ZSMOOTH, ZSMOOTH = smooth distance
Description: smooth distance.
For example:

```
ZSMOOTH = 5 'sets 5mm smooth distance
```

Chapter 7 Input-output Correlation

AIN

Type: input and output function
Grammar: AIN (channel)
Description: read Analog Input, then return the scale value of AD conversion modules.The scale value range from 0 to 4096 has its corresponding voltage range from 0 to 10V

Parameters: channel: AIN channel is 0—127
The AD channel number of expansion board is related with dial code, the staring value is (8 + code combination value *8)
For xample:
```
        a=AIN(1)        ‘reading AD of the channel 1
        a=AIN(1)*4096/10    ‘magnitude of voltage of channel 1
```

AOUT

Type: input and output instruction or function
Instruction Grammar: AOUT (channel) = value
Function Grammar: AOUT (channel)
Alias: DA
Description: analog channel output. The scale value range 0~4096 corresponds to the 0~10V voltage
Parameters: channel: analog output channel 0-63 expansion board, DA channel number and dial code related, the starting value is (4 +dial code combination value *4)
For example:
```
   AOUT (1) = 0. Close the output DA channel 0
    AOUT (1) = 4096 'DA1' output 10V voltage
```

IN

Type: input and output function
Grammar: IN([channel1],[channel2])
Description: read input, return 0-31 status without parameter. Read the state after INVERT_IN flip. The expansion board IO channel number is related to the dialing code. The initial value is (16 +code combination value *16)
Parameter: Channel1: to read the start input channel
channel2: to read the end of the input channel, no end of the channel, the status of a single input returned.
For example:
```
    a=IN(1) ‘   read the input of channel 1
```

INFILTER

Type: system parameter
Grammar: VAR1 = INFILTER ，INFILTER= expression
Description: The filter parameter of the local input, the greater the value, the longer the filter time, 2-9, default 2.

For example:

```
   INFILTER= 5 "increase filtering time in severe interference situations."
```

IN_SCAN

Type: input and output function
Instruction Grammar: IN_SCAN([channel1],[channel2])
Function Grammar: VAR1=IN_SCAN([channel1],[channel2])
Description: scan input change, return value true- change, false- no change.
The function must be fixed for continuous scanning, returning a change between two scans and reading the specific changes by IN_EVENT. Using the INVERT_IN flip state
This support is available only after firmware version 20140214 is available, the scan range is limited by width, and the 00x series of controllers can only be used in a single task
Parameter：channel1: the initial input channel to read
Channel2: to read the end of the input channel, the individual input is scanned without the end of the channel.
For example：

```
while 1
  if in_scan(0,23) then
      if in_event(0) > 0 then
          print "in0 up", in_buff(0)
      Else if in_event(0) < 0 then
          print "in0 down", in_buff(0)
      End if
    End if
Wend
```

IN_EVENT

Type: input output function
Function Grammar: VAR1 = IN_EVENT (IONUM)
Description: Read the input changes, the rising edge of 1-, the falling edge of -1-, and no change in 0-. This function must be used in conjunction with the IN_SCAN.
Parameter: IONUM: The “IO” number must be the input range of the IN_SCAN.
For example:
Reference to: IN_SCAN

IN_BUFF

Type: input output function
Grammar: IN_BUFF([channel1],[channel2])
Description: Reads the current input of the IN_SCAN buffer and returns the 0-31 state without Parameters. Read the state after the INVERT_IN flip
Parameter: Channel1: the initial input channel to read must be the input range of the IN_SCAN.
Channel2: reads the end of the input channel and returns the status of a single input without ending the channel.
For example：
Reference to: IN_SCAN

INVERT_IN

Type: special instruction
Grammar： INVERT_IN(channel, state); VAR1= INVERT_IN(channel)
Description: Reverses the input status and can read to there is whether an inversion
parameter： channel: Input channel
state: ON/OFF。
Reference to：IN
For example：

```
INVERT_IN (1, ON) special signal, default OFF valid, inversion avoids the
limit signal (except for ECI series controller)
FWD_IN=1
```

OP

Type: input and output instructions and functions
Instruction Grammar：OP([ionum],value) or OP(ionum1, ionum2,value[,mask])
Function Grammar: OP([firstnum[,[finalnum])
Alias: OUT
Description: output or read output status, automatically used as an expression, syntax for function. The expansion board IO number is related to the dialing code. The start number of the extension board is (16 code combination value *16)
Parameter :
Ionum: Output number, 0-, output 0-31 without this parameter.
Value: The output state, where multiple output ports are operated, indicate the status of multiple ports by bit

ionum1: The first output channel to operate
ionum2: The last output channel to be operated
mask: Bits are used to specify which IO needs to operate, and when the first channel is not filled, the operation is done to the last channel
firstnum: Output number, 0-,
finalnum: The output number, 0-, reads the status of a single output without this parameter

For example：

```
Flip the outlet 0
IF OP (0) = ON THEN
    OP (0, OFF)
ELSE
    OP (0, ON)
END IF
```

READ_OP

Type: input output function
Grammar: READ_OP ([firstnum[,[finalnum])
Description: read the output status, with the OP, a number of output operations, bit by point to indicate the number of state of the port.
Parameter: firstnum: Output number, 0-,
Fianalnum: The output number, 0-, reads the status of a single output without this parameter.

For example:

```
Turn the output 0
IF READ_OP (0) = ON THEN
     OP (0, OFF)
ELSE
    OP (0, ON)
END IF
```

PWM_FREQ

Type: PWM control function
Grammar： PWM_FREQ (index, freq) PWM_FREQ (index)=freq
Description: PWM: frequency setting or read, note that only support PWM function of the controller to support this function.
Parameter: index: Number, 0-
Freq: frequency

For example：

```
PWM_FREQ (0)=1000
?PWM_FREQ (0)
```

PWM_DUTY

Type: PWM control function
Grammar： PWM_DUTY(index, duty) PWM_DUTY(index)=duty
Description: PWM: duty cycle setting or read, note that only support PWM function of the controller to support this function.
parameter：index: number,0-
Duty: The duty cycle is 0-1, and when is set, the PWM is turned off.
For example：

```
PWM_DUTY(0)=0.5
?PWM_DUTY(0)
```

PSWITCH

Type: input / output instruction
Instruction Grammar: PSWITCH(num,enable,[,axis,op num,op state,set pos,reset pos])
Description: operate the output port by location comparison.
parameter： num: The serial number of the comparator, the ZMC1xx series has 16 comparators, numbers: 0-15
enable: Operation comparator enable - ON start / OFF cancellation
Axis: Specifies the axis number to get the location
op num: The IO number of the operation
op state: The state of the output, 1, indicates that in the range below, the output is ON, and 0 indicates the output in the range below OFF.
set pos: Set the start position of the output and use the units unit
reset pos: Set the output reset position using the units unit
For example:
PSWITCH (0, ON, 0,1, ON, 0200) open the output port 1 within the 0-200 position range of axis 0

ZSIMU_IN

Type: emulator specific instruction

Grammar： IN([channel,]state)
Description: emulation external input
parameter： channel: Channel, modify 0-31 channels while not filling
State: State.
For example:
ZSIMU_IN(0,1) ’The simulation input 0 is valid

ZSIMU_AIN

Type: emulator specific instruction
Grammar： AIN(channel,state)
Description: analog channel input
Parameter: channel: passage way
State: state
For example：
ZSIMU_AIN(0,1024) ‘Simulation channel 0

ZSIMU_ENCODER

Type: emulator specific instruction
Grammar： ZSIMU_ENCODER (axisnum,value)
Description: simulation encoder input. The axis must be configured with encoder mode.
parameter： axisnum: Axis number
value: The input of the underlying encoder is automatically modified to ENCODER.
For example：
ZSIMU_ENCODER(0,1024) ’ENCODER = 1024

Chapter 8 Related- communication

ADDRESS

Type: system parameter
Grammar: ADDRESS = value
Description: MODBUS serial number of all serial ports of controller, station number 1-127, default =1.
Reference to: SETCOM, PORT, PROTOCOL
Distance:
Print ADDRESS

CANIO_ADDRESS

Type: system parameters are automatically stored in FLASH and will be effective after reboot
Grammar: CANIO_ADDRESS = value
Description: CANID and speed settings on board CAN
Low 8 (0-7 CANID) said, 0-32 said, the default 32,32 based controller.
The high 8 bit (bit 8-15) represents the CAN speed

Highest 8 bit value	baud rate value CAN:
0	500 kHz(default value)
1	250kHz
2	125kHz
3	1MHz

Reference to: CANIO_ENABLE
For example：
CANIO_ADDRESS = 32, set the main end, CAN baud rate 500K,
CANIO_ADDRESS = 32+ 1024, set the main end, CAN baud rate is 125KHZ,
CANIO_ADDRESS = 1, set CANID to 1, at this time from the end, 500K, cannot connect to the IO board

⚠ At a network, do not configure multiple main controller.

⚠ The CAN address and speed settings must be effective after reboot

CANIO_ENABLE

Type: system parameter
Grammar: CANIO_ENABLE = ON/OFF
Description: enables or disables the internal CAN main end function, which is enabled by default when CANIO_ADDRESS sets 32.
Reference to: CANIO_ADDRESS
For example：
CANIO_ENABLE = ON

CANIO_STATUS

Type: system status
Grammar: CANIO_STATUS (cardnum)
Description: get “IO” plate current state, return ON/OFF (more than 20140325 firmware support it):

For example1:

    *CANIO_STATUS? "All output status of IO board

For example 2:

    If CANIO_STATUS (1) =0 then 'judgment connection status of IO board
    Print "extension module 1 is not connected properly"
End if

CAN

Type: system instruction

Grammar: CAN (channel, function, tablenum)

Description: directly through the CAN bus to send and receive data, you can achieve multiple controllers communication via CAN, but with a CAN network only supports one main terminal (CANIO_ADDRESS=32); Such feature is only available in a new firmware version, if not, please contact the manufacturer directly.

Parameter：

Channel: CAN channel ,0 represents the first channel, -1 represents the default channel。

Function: Function Code

value	describe
6	Receive, when there is no data, identifier<0
7	send

Tablenum： The TABLE location of data storage, storage in turn:

identifier：Can, Cob, Id, 11 bit, high level data reservation used by ZCAN, suggested value: 0-511. if is below 0, it means there is no data received.

bytes： data region bytes, up to 8.

data： Data: data region, bytes (0-FF)

Reference to: CANIO_ADDRESS

For Example：

    Transmitting terminal
    TABLE (0,1,8,1,2,3,4,5,6,7,8) sends cobid=1,8 bytes, followed by 1-8.
    CAN(0,7,0)
    Receiving terminal:
    CANIO_ADDRESS=1 'is set as non - Master. set This parameter one time is
enough.
    CAN (0,6,0)
   ?TABLE(0)

GET

Type: system instruction
Grammar 1: GET, #PORT, VARIABLE
The Grammar of 2: GET #PORT, ARRAY[(StartIndex) [maxchares]].
Grammar 3: charesget = GET, #PORT, VARIABLE
Grammar 4:charesget = GET #PORT Grammar (StartIndex), ARRAY[[maxchares]].
Description: reads a byte from the RAW communication port, stored in a variable, not to read the grammar of 1/2 blocking, this function calls in multi task; grammar 3,4 will return the number of bytes read; the previous 20150522 version only supports grammar 1.
Reference to: PORT, SETCOM, PRINT #
For example 1:

```
    DIM VAR1
    SETCOM (38400,8,1,0,0,0) ‘start RAW mode
    GET #0, VAR1
```

For example 2:

```
     DIM ARRAY1(101)
     SETCOM (38400,8,1,0,0,0) ‘open RAW mode
     CHARES = GET, #0, ARRAY1, 100, ‘read up to 100 characters at most
    If CHARES > 0 THEN
       ARRAY1(CHARES) = 0 set end o
       PRINT ARRAY1 'string printed out
     ENDIF
```

OPEN#

Type: system instruction
Grammar: OPEN, #PORT, mode, portnum[, ipaddress]
Description: Custom Ethernet communication. The new firmware version provides this feature
Parameters: PORT communication channel, see PORT description.
"Mode": communications master, TCP_CLIENT, -, TCP_SERVER, -Lord
Portnum: TCP port number
IPAddress: strings, which are available from the end
Reference to: PORT, SETCOM, PRINT #
For example:

```
  open #11, "TCP_SERVER", 10 "
  Open #10, "TCP_CLIENT", "192.168.1.112" from the 6000.
```

PRINT

Type: system instruction
Grammar: PRINT #PORT, "string"
Description: from the RAW mode of communication inside the output string, hit 0 end
Reference to: PORT, SETCOM, GET #
For example：

```
DIM VAR(10)
VAR = "AAAA"
SETCOM (38400,8,1,0,0,0) ‘start RAW mode
PRINT #0,VAR
```

PUTCHAR

Type: system instruction
Grammar: PUTCHAR #PORT, the string,
Description: from the RAW communication port which can output string array representing the number of strings.
Reference to: PORT, SETCOM, GET #
For example：

```
DIM VAR1, ARRAY1(10) VAR1 = $FE
ARRAY1 = "ABCDEFGHIJ""
SETCOM(38400,8,1,0,0,0) ‘start RAW way
   PUTCHAR #0, VAR1
   PUTCHAR #0, ARRAY1
```

PRINT

Type: print output function
Grammar: PRINT expression, “string”
Description: prints the information to the output window of the PC ZDEVELOP software. Through * special strings can print some special information
Parameter: expression: valid expression
*SET: prints all parameter values
*TASK: print job information.
*MAX: print all specification parameters
*FILE: print program file information
*SETCOM: prints the configuration information for the current serial port
*BASE: print the current task list (BASE 140123 version supports).

* array name: all elements of an array name print array, the array length can not be too long.

* parameter name: print a single parameter for all axes

Example 1:

```
 >>print 1+2
   3
```

Example 2:

```
>>print *task
Task:0 Running. file:"hmi.bas" line:280: Task:1 Stopped.
Task:2 Stopped.
Task:3 Stopped.
Task:4 Stopped.
Task:5 Stopped.
Task:6 Stopped.
```

Example 3:

```
>> print *mpos
21872.400 0 0 0 0 0 0 0 0 0 0 0
```

TRACE

Type: print output function

Grammar: same as PRINT

Description: prints the information to the output window of the ZDEVELOP software on the PC side

Controlled by the **ERRSWITCH** switch

For example：

```
ERRSWITCH = 3
Dpos(0) =0
Trace "dpos (0) =" dpos (0):
Result: dpos (0) = 0
```

WARN

Type: print output function

Grammar: same as PRINT

Description: prints the information to the output window of the ZDEVELOP software on the PC side

Controlled by the ERRSWITCH switch

For example：

ERROR

Type: print output function
Grammar: same as PRINT
Description: the output window for printing information to the PC side of the ZDEVELOP software.
Controlled by the ERRSWITCH switch

ERRSWITCH

Type: system parameter
Grammar: ERRSWITCH = switch
Description: debug output switch control TRACE WARN ERROR debug output statement is whether really output.
Parameter: switch debugging output switch,
1- :TRACE WARN ERROR instruction does not output
1 -: only output ERROR instructions
2 -: output WARN ERROR instructions
3-:trace WARN ERROR instructions all output

PORT_STATUS

Type: channel parameter, read only
Grammar: VAR1 = PORT_STATUS (port)
Description: returns the status of the current channel. The serial port always returns 1. The new firmware version provides this functionality.

Value	The protocol
0	No connection
1	Connected to use

Reference to: PORT, SETCOM, ADDRESS

PORT

Type: channel correction auxiliary instruction
Grammar: PORT (portnum)
Description: other channels can be specified when accessing channel parameters.
Parameter: portnum: channel number
The number of channels supported by different types of controllers is different
ZMC00x series

The channel number	The protocol
0	Serial port A
1	Serial port B

The ZMC1-2xx series supports simultaneous two Ethernet connections, port number 502, and touch-screen MODBUS-TCP support
Protocol connection

The channel number	The protocol
0	RS232 Serial port
1	RS485 Serial port
2	Ethernet interface, connection 1
3	Ethernet interface, connection 2
10	Custom network communication channel 1
11	Custom network communication channel 2

ZMC3xx series: with 3 serial ports

The channel number	The protocol
0	RS232 serial port
1	RS485 serial port
2	RS422 serial port
3	Ethernet interface, connection 1
4	Ethernet interface, connection 2
10	Custom network communication channel 1
11	Custom network communication channel 2

ZMC4xx series: with 2 serial ports.

The channel number	The protocol
0	RS232 serial port
1	RS485 serial port
2	Ethernet interface, connection 1
3	Ethernet interface, connection
10	Custom network communication channel 1
11	Custom network communication channel 2

ECI series: only one serial port

The channel number	The protocol
0	RS232 serial port
1	Ethernet interface, connection 1
2	Ethernet interface, connection 2

Reference to: FILE_PORT

FILE_PORT

Type: channel parameters
Grammar: VAR1 = FILE_PORT, FILE_PORT = filenum
Description: returns or sets the default file number of the current channel.
After setting, you can access the file modules, variables, or arrays of the corresponding files through the online command
Parameter: filenum: the number of program file, which can be viewed through the PRINT *FILE
Reference to: PORT
For example：
">>FILE_PORT = 0" the FILE_PORT parameter of the channel that is currently connected

⚠ This parameter is automatically used by the ZDevelop integrated development environment. Do not modify this parameter when using ZDevelop

PROTOCOL

Type: channel parameter, read only
Grammar: VAR1 = PROTOCOL (port)
Description: returns the communication protocol for the current channel

Value	protocol
0	RAW data format, no protocol
3	MODBUS protocol controller from the end (the default).
14	MODBUS protocol, controller based
15	Direct command.

Reference to: PORT, SETCOM, ADDRESS

SETCOM

Type: system instruction
Grammar: SETCOM (baudrate, databits, stopbits, parity, port[, mode], variable], timeout])
Description: serial port configuration
Parameter: baudrate: Serial baud rate: 96001920048001 1520, 38400 (default)
Databits:8 data bits
Stopbits:Stop: only set 0/1/2
Parity: whether to Verify?:

value	Description

0(default)	none
1	odd
2	Even parity check

Port: Serial port PORT number 0-1, see the PORT description, different controllers are different。

Mode: protocol

Value	Description
0	RAW data schema, no protocol At that time can use GET, PRITNT #
4(default)	MODBUS slave end (16 bit integer)
14	MODBUS main end (16 bit integer)
15	Direct instruction execution mode, at this time can be straight Connect the string instruction from the serial port (line feed end)

Variable: register selection, 0-vr, 1-table, 2-system MODBUS register.
The variable parameter is the global setting and all ports have one

	describe
0	VR, at this point, a VR is mapped to a MODBUS_REG.
1	TABLE, at this point, a TABLE data is mapped to one MODBUS_REG. (not recommended)
2 (default)	The MODBUS register system, the VR and MODBUS mail The sink is two separate intervals
3	VR_INT mode, at which point a VR_INT is mapped to two MODBUS_REG。

⚠ The variable parameter is the global setting and all ports have one.

⚠ When set to VR or TABLE, the generic export will map to the location of MODBUS_BIT (0) and lose. The entry will be mapped to the location of MODBUS_BIT (1000), so do not use MODBUS_BIT as the buttons of touch screen at this time, etc.

Timeout: mode=14 is message timeout time, milliseconds, default 1000

Reference to: ADDRESS, PROTOCOL, MODBUS_STRING, MODBUS_IEEE, PORT

For example：

Setcom (38400, 8,1,0,0,4,2) "set the serial port 0 for Modbus from the end of 38400, the baud rate"?

>>?*setcom "print out the serial port to the current configuration

information

Setcom (38400, 8,1,0,1,14,21000) sets the serial port 1 to the Modbus main end, and the baud rate 38400

Setcom (38400, 8,1,0,0,15) "set the serial 0 direct string instruction.

⚠ The ZMC00x series does not support the MODBUS master end

MODBUS_BIT

Type: MODBUS bit register
Command Grammar: MODBUS_BIT (first, [last]) = value
Instruction Grammar: MODBUS_BIT (first, [last])
Description: modify or read the BIT register, the touch screen is generally called the 0x register.
The ZMOTION motion controller specifically features 2048 MODBUS bit registers
Another special MODBUS 0x register, through these registers can be directly read from the touch screen and set the IO,
Notice that the read IO is the original state, INVERT_IN does not work:
Reference to: ADDRESS

Register （zero based）	meaning
10000-	Enter IN, each IN accounts for 1 registers
20000-	Output OP, each OP occupies one register

Parameter: first bit register number, number starts from 0. Last bit register number, number starts from 0

for example：
```
DIM VAR
MODBUS_BIT(100) = 1
VAR = MODBUS_BIT(10)
```

MODBUS_IEEE

Type: MODBUS word register
command Grammar: MODBUS_IEEE (Regnum) = value
Function Grammar: MODBUS_IEEE (Regnum)
Description: modify or read the word register, 32 bit floating point mode. The touch screen is generally called the 4x register.
The ZMOTION motion controller is specifically equipped with the MODBUS word register that will take up the address of two registers. When using the system MODBUS word register, set the SETCOM parameter to use the system register
Another: special MODBUS 4x registers, through which registers can be read directly from the touch screen:

Register（zero based）	type	meaning
10000-	IEEE	DPOS read, each axis accounted for two Register
11000-	IEEE	MPOS read
12000-	IEEE	VP_SPEED read

Parameter:

Regnum register number, starting from 0

For example：

```
DIM VAR
MODBUS_IEEE(100) = 100.10
VAR = MODEBUS_IEEE(100)
```

MODBUS_LONG

Type: MODBUS word register
Instruction Grammar: MODBUS_LONG (Regnum) = value
Function Grammar: MODBUS_LONG (Regnum)
Description: modify or read the word register, a 32 bit integer, will occupy two register address. The touch screen is generally called the 4x register.

The ZMOTION motion controller is specially equipped with the MODBUS word register. When using the system MODBUS word register, please set the SETCOM parameter to use the system register
Parameter: Regnum: register number, starting from 0.
For example:

```
DIM VAR
MODBUS_LONG(100) = 1000
VAR = MODBUS_LONG(100)
```

MODBUS_REG

Type: MODBUS word register
Command Grammar: MODBUS_REG (Regnum) = value
Function Grammar: MODBUS_REG (Regnum)
Description: modifying or reading word registers, the touchscreen is generally called the 4x register

The motion controller of ZMOTION is specially equipped with MODBUS word register, and when using the system MODBUS word register, please set it. The SETCOM parameter is used to use the system register. Different types of controllers have different word registers.

Another: special MODBUS 4x registers, through which registers can be read directly from the touch screen:

Register（zero based）	type	meaning

13000-	16	The output of DA
14000-	16	AD read

Parameter: Regnum: register number, starting from 0
For example：

```
DIM VAR
MODBUS_REG(100) = 100
VAR = MODBUS_REG(100)
```

MODBUS_STRING

Type: MODBUS word register, string function
Grammar: MODBUS_STRING (index, chares)
Description: string read MODBUS register area, in bytes to read. The touch screen is generally called the 4x register.
Parameters: index: initial register number of MODBUS, numbers start at 0. Different types of controllers have different word registers.
Chares: read the total number of characters.
Reference to: DIM, VRSTRING
For example：

```
DIM ARR(8)
>> MODBUS_STRING (0, 8) = abc,
>>print, MODBUS_STRING (0, 8)
Abc
ARR = MODBUS_STRING(0,8)
```

MODBUSM_DES

Type: communication instruction
Grammar: MODBUSM_DES (address[, port]), ADDRESS1 = MODBUSM_DES ([port])
Description: set or read the main end of each other of the MODBUS
Parameters: address: Modbus protocol to end station number.
Port: current MODBUS master communication port number,
Reference to: ADDRESS, PROTOCOL, PORT, SETCOM
For example：

```
MODBUSM_DES (1,1) 'set 485 port, the other station No. 1
```

MODBUSM_STATE

Type: communication status

Grammar: VAR1 = MODBUSM_STATE
Description: MODBUS master communication status

price	describe
0	in a normal state
1	Waiting for response
2	Wait timeout
3	Answer error

Reference to: PROTOCOL, PORT, SETCOM
For example：

```
Print MODBUSM_STATE
0
```

MODBUSM_REGSET

Type: communication instruction
Grammar: MODBUSM_REGSET (startreg, num, local_reg)
Description: set the local MODBUS register to the opposite end
Parameter: startreg: The start number of the terminal register starts at 0.
Num: The number of registers
Local_reg: takes the initial number from the local system MODBUS register.
Reference to: ADDRESS, PROTOCOL, PORT, SETCOM
For example：
MODBUSM_REGSET ("0,10,0") copies the local register 0-9 to the communication terminal register 0-9

MODBUSM_REGGET

Type: communication instruction
Grammar: MODBUSM_REGGET (startreg, num, local_reg)
Description: copy the MODBUS register on the opposite side to the local
parameter: startreg: The start number of the terminal register starts at 0.
Num: Number of registers
Local_reg: takes the initial number from the local system MODBUS register.
Reference to: ADDRESS, PROTOCOL, PORT, SETCOM
For example：
MODBUSM_REGGET (0,10,0) 'to register communication on end 0-9 is

copied to the local register 0-9

MODBUSM_BITSET

Type: communication instruction
Grammar: MODBUSM_BITSET (startreg, num, local_reg)
Description: sets the local MODBUS bit register to the opposite end
Parameter: startreg: The start number of the terminal register starts at 0
Num: Number of registers
Local_reg: takes the initial number from the local system MODBUS register.
Reference to: ADDRESS, PROTOCOL, PORT, SETCOM
For example：
MODBUSM_BITSET ("0,10,0") copies the local bit register 0-9 to the communication terminal register 0-9

MODBUSM-BITGET

Type: communication instruction
Grammar: MODBUSM_BITGET (startreg, num, local_reg)
Description: copy the MODBUS bit register on the opposite end to the local
Parameter: ：startreg: The start number of the terminal register starts at 0.
Num: Number of registers
Local_reg: takes the initial number from the local system MODBUS bit register.
Reference to: ADDRESS, PROTOCOL, PORT, SETCOM
For example：
MODBUSM_BITGET (0,10,0) 'communication to end a copy to the local register 0-9 register 0-9

The chapter 9 Related-system

All dates, times, Parameters, or instructions are not allowed to be modified after LOCK

APP_PASS

Type: system Instruction
Grammar: APP_PASS (pass)
Description: the user can set an application password to the controller. When you download the ZAR package, you can choose to verify the

password. The check error will not be downloaded. And LOCK cannot be modified after APP_PASS.
Parameter: pass letters or numbers, "_" and a few special symbols, a total of not more than 16 characters
For example:

```
APP_PASS(Zmotion)
```

⚠ APP_PASS was not reversible encryption algorithm, once forgot, will not be informed, pass cannot set a variable or expression,
Otherwise, the variable name will be considered a password

CLEAR

Type: system instruction
Grammar: CLEAR ()
Description: clear all VR content.
For example:

```
CLEAR()
```

CONTROL

Type: system only read parameter
Grammar: CONTROL
Description: returns controller type.
For examples:

```
>>print CONTROL
106
```

DATE

Type: system parameters
Grammar: DATE= DD:MM:YYYY
Description: sets the date of the system, or returns the number of days after 2000
For example

```
DATE=27:2:13
>>print DATE
4806
```

DATE$

Type: String Functions

Grammar: DATE$
Description: string function, returns current date in DD/MM/YYYY format.
For example:

```
>>print DATE$
27:02:201
```

DAY

Type: system parameter
Grammar: DAY
Caption: system set the system clock for week is 0-6,0 stands for Sunday.
For example 1:

```
DAY = 3
```

For example 2

```
>>print DAY
 3
```

DAY$

Type: String Functions
Grammar: DAY$
Description: string function that returns the current week:
For example:

```
>>print DAY$
Wednesday
```

RTC_DATE

Type: system parameters
Grammar: RTC_DATE = YYYYMMDD
Description: setting up or getting system dates.
For examples:

```
RTC_DATE = 20130227
>>print RTC_DATE
20130227
```

⚠ That way of attention and DATE instruction is not the same.

DISABLE_GROUP

Type: system instruction
Grammar: DISABLE_GROUP (AXIS1, AXIS2),...)

Description: the axis is arranged as a group, will close all make within the group can drive after an alarm (ECAT product support).
Reference to: ?*DISABLE_GROUP, WDOG, AXIS_ENABLE, ERRORMASK, AXISSTATUS
For example：

```
      DISABLE_GROUP (-1) "cancel all set the alarm off, WDOG.
       DISABLE_GROUP (0,1,2) axis 0,1,2 sets a group, usually a machine
```

ERROR_AXIS

Type: the state of the system
Grammar: ERROR_AXIS
Description: the first axis number of the error

ERROR_SET

Type: system instruction
Grammar: ERROR_SET (output port register address [modbus, ERRSUBName])
Description: when the BASIC program is running error, automatically open output. And write the corresponding error message MODBUS register, output reduction when the BASIC program will automatically re run the.20130906 above support. Register the length of at least 32 bytes.
ERRSUBName: More than 150721 version provides; sets up a SUB process for temporary processing; Grammar error stops. This function is called automatically when it is stopped. This procedure does not use instructions such as WAIT that can be blocked, and must be streamlined. Grammar errors occur in this process and will no longer be processed
For example：

```
    ERROR_SET(1,200)
   Mov (30) 'spelling mistakes, at runtime error, the output port 1
   open.Modbus_string (200,32)
        = sample_move.bas, 6, e2043"
```

MOTION_ERROR

Type: the state of the system
Grammar: MOTION_ERROR
Description: list of axis errors, each bit represents an axis. A 0-n axis 0-n
For example：

```
    Print MOTION_ERROR 0
```

FRAME

Type: System Parameter
Description: reserved

IP_ADDRESS

Type: system parameters, automatic storage to FLASH
Grammar: IP_ADDRESS = dot.dot.dot.dot
Description: controller IP address, only with Ethernet interface controller support, read with 32 bit integer return. See: IP_GATEWAY IP_NETMASK
For example：

```
>>IP_ADDRESS=192.168.1.100
>>PRINT IP_ADDRESS
  23593316
```

IP_GATEWAY

Type: system parameters, automatic storage to FLASH
Grammar: IP_GATEWAY = dot.dot.dot.dot
Description: IP gateway of controller, only supported by the controller with Ethernet interface, when read it will return a value in by a 32 bit integer.
Reference to: IP_ADDRESS IP_ADDRESS
Example: IP_GATEWAY =192.168.1.1

IP_NETMASK

Type: system parameters, automatic storage to FLASH
Grammar: IP_NETMASK = dot.dot.dot.dot
Description: the controller IP network mask is supported only by controller with Ethernet interface. It is returned with 32 bit integers when reading
Reference to: IP_GATEWAY IP_ADDRESS
For example: IP_NETMASK = 255.255.255.0

LOCK

Type: system instruction
Grammar: LOCK (pass)

Description: lock controller, to debug and modify.
Parameter: letters or numbers, "_" and a few special symbols, a total of not more than 16 characters.
Reference to: UNLOCK
For example：

```
LOCK(passwd)
```

⚠ LOCK password is encrypted by an irreversible algorithm. Once forgotten, it is not known, and “pass” cannot set variables or expressions style, otherwise the variable name will be considered a password

SERIAL_NUMBER

Type: system only read parameter
Grammar: SERIAL_NUMBER
Description: returning the controller ID is a unique sequence number that can also be bound to the ID when the ZAR packet is generated, so that this ZAR can only be used for this controller
Reference to: APP_PASS
For example：

```
Print SERIAL_NUMBER
13080011
```

SERVO_PERIOD

Type: system parameter
Grammar: SERVO_PERIOD = value
Description: Servo update cycle, the default of 1000 microseconds, some models of controller not to change.
Reference to: SERVO

TABLE

Type: system array
Grammar: TABLE (index) = value, VAR1 = TABLE (index), TABLE (index [, value1..])
Description: global array created by system default. All programs can be accessed
Reference to: VR, TSIZE
For example：

```
TABLE(0) = 10
```

TSIZE

Type: system only read parameter
Grammar: TSIZE
Description: the number of all elements of the TABLE.
Reference to: TABLE
For example：

```
    Print TSIZE
```

TIME

Type: system parameters
Grammar: TIME = hh:mm:ss
Description: set the system clock time, returns the number of seconds after 0.
For example 1:

```
    TIME= 11:14:40
```

For example 2

```
   >>print TIME
     40541
```

TIME$

Type: String Functions
Grammar: TIME$
Description: 24 hour format string function, hh:mm:ss returns the current time
For example:

```
    >>print TIME$
     11:29:46
```

TRIGGER

Type: system instruction
Grammar: TRIGGER
Description: data capture for oscilloscopes, the version after 150723supports, and zdevelop oscilloscope functions.
For examples:

```
     TRIGGER   ‘grasping movement every moment following speed function
 storage oscilloscope settings of the TABLE interval.
    MOVE(10000)
```

RTC_TIME

Type: system parameters
Grammar: RTC_TIME = hhmmss
Description: setting or getting system time.
For examples:

```
RTC_TIME = 113706
>>print RTC_TIME
113706
```

⚠ That way of attention and TIME instruction is not the same.

UNLOCK

Type: system instruction
Grammar: UNLOCK (pass)
Description: Unlock controller
Parameters: pass: LOCK password.
Reference to: LOCK

⚠ The LOCK password is encrypted with an irreversible algorithm, and once forgotten, it will not be known

LIMIT_BUFFERED

Type: system parameter
Grammar: LIMIT_BUFFERED = value
Description: limit the number of buffers and cannot exceed the maximum value of the controller
Reference to: MERGE, CORNER_MODE

VERSION

Type: system status
Grammar: VAR1 = VERSION
Description: system software version number

VR

Type: system instruction
Grammar: VR (index) = value, VAR1 = VR (index)

Alias: NVRAM
Description: Save the storage register group and pay attention to the number
of different types of controllers
Reference to: FLASHVR
For example：

```
VR(0) = 10
Aaa = VR(0)
```

WDOG

Type: system parameter
Grammar: WDOG
Description: controls the power of all axes
Reference to: AXIS_ENABLE

VRSTRING

Type: String Functions
Grammar: VRSTRING (index[, chares])
Description: store strings with VR
Reference to: index: initial VR number, starting at 0
Chares: Read the total number of characters
For example：

```
  VRSTRING (0, 8) = "abc"
>>print VRSTRING (0, 8)
   abc
```

ZSIMU_IN

Type: emulator specific instruction
Grammar: ZSIMU_IN[([ionum, [value])
Description: analog input port
Parameter: ionum: input number, 0-, without this parameter output 0-31.
Value: input status
Reference to: IN

ZSIMU_AIN

Type: emulator specific instruction
Grammar: ZSIMU_AIN (ionum, value)
Description: analog input port
Parameter: ionum: AD number, 0-

Value: AD input
Reference to: AIN

ZSIMU_ENCODER

Type: emulator specific instruction
Grammar: ZSIMU_IN (axis, num, value)
Description: analog input port
Parameter: axis num: axis number, 0-
Value: ENCODER simulation value
Reference to: ENCODER

Chapter 10 Storage Related Instruction

The ZMotion motion controller has internal FLASH memory, and part of the model controller has an external memory interface that can connect to the U disk or SD card, as well as the hardware manual of the specific controller

The U disk or SD card is formatted as FAT

FILE

Type: U disk file instruction
Grammar: value = FILE "function"",...
Description: load U disk files, or search files
Parameter：function: FS(Function Selection)

“LOAD_ZAR”	FILE “LOAD_ZAR”,“filename” Load the ZAR upgrade file inside the usb drive, and the upgrade failed to return 0 and will be WARN output failed, and the ZAR file will be automatically started when the upgrade is successful. This return value TRUE is not meaningful ⚠After the upgrade, the breakpoint in the original debug state will be cleared
“FIND_FIRST”	FILE “FIND_FIRST”, tySearch U disk file. Type: 1- file / 2- folder /.Extend file suffix name Vr: VRSTRING (VR) stored search results, exceeding the maximum VR, use the MODBUS_STRING。
“FIND_NEXT”	FILE “FIND_NEXT”, vr Vr: VRSTRING (VR) stored search results, exceeding the

	maximum VR, use the MODBUS_STRING。
“FIND_PREV”	FILE “FIND_PREV”, vr Vr: VRSTRING (VR) stored search results, exceeding the maximum VR, use the MODBUS_STRING。
“FLASH_FIRST”	FILE “FLASH _FIRST”, type, vr To search for FLASH files, only supports BIN and Z3P files. Type: 1- file / 2- folder /.Extend file suffix name Vr: VRSTRING (VR) stored search results, exceeding the maximum VR, use the MODBUS_STRING。
“FLASH_NEXT”	FILE “FLASH _NEXT”, vr Vr: VRSTRING (VR) stored search results, exceeding the maximum VR, use the MODBUS_STRING。
“FLASH_PREV”	FILE “FLASH _PREV”, vr Vr: VRSTRING (VR) stored search results, exceeding the maximum VR, use the MODBUS_STRING。
“COPY_FROM”	FILE "COPY_FROM", "FLASH FILE name", "U disk FILE name"] Copy the FLASH file to the U disk, only supports BIN and Z3P files.
“COPY_TO”	"COPY_TO" FILE "COPY_TO", "U disk file name", "copy the FLASH file name] U disk file to FLASH, only support BIN and Z3P files.

For example：

"FILE?" LOAD_zar "," main "and" U disk inside have been stored in the main.zar upgrade file -1

```
>>?FILE "FIND_FIRST", ".ZAR" ,0 -1
>>?VRSTRING(0)
AAA.ZAR
```

FLASHVR

Type: Storage instructions
Grammar：FLASHVR (function)
Description: copy RAM data into FLASH
Parameter: function: Function selection

-1	The entire TABLE is stored in FLASH and is automatically read to the TABLE when it is on.
-2	When FLASH is cancel, the TABLE is read

Reference to：FLASH_WRITE
For example
FLASHVR (-1)

⚠The ZMC00x controller stores the TABLE data to the last block of FLASH, with FLASH_WRITE, The FLASH_READ instruction can also operate on this block to avoid conflicts.

STICK_WRITE

Type: Storage instruction
Grammar: value = STICK_WRITE (filenum, table_start [, length, [format]]])
Description: copy the TABLE data to the external memory, value=TRUE said the operation is successful, otherwise the operation failed.
Parameters: filenum: file number, corresponding to SD [filenum].BIN
table_start: The start TABLE number of the start operation.
Length: the number of TABLE elements to be operated, and the default 128
Format: writes to the file format

Value	Description
0(default)	Float format storage
1	Text format storage

Reference to: U_WRITE
For example：
STICK_WRITE (0,0128,0) ‘copy the first 128 elements of the table into external memory in float format

⚠ Does not have the external memory interface controller this command is not supported.

STICK_READ

Type: Storage instruction
Grammar: value = STICK_READ (filenum, table_start [, format])
Description: copy data from external memory to TABLE. Value=TRUE indicates operation is successful, otherwise operation failed
Parameter：filenum: File number, corresponding to SD [filenum].BIN
table_start: The start TABLE number of the start operation
format: Format for writing files

Value	Description
0(default)	Float format storage
1	Text format storage

Reference to: U_READ
For example：
STICK_READ (0,0128,0) copy external memory 0.bin file data to TABLE

⚠Does not have the external memory interface controller this command is not supported.

STICK_WRITEVR

Type: Storage instruction
Grammar: value = STICK_WRITEVR (filenum, vr_start, length, format]])
Description: copying data from VR to external memory. Value=TRUE indicates successful operation, otherwise operation fail
Parameter: filenum: file number, corresponding to SD [filenum]. BIN
Vr_start: starts the starting VR number.
Length: operate the number of elements, default 128
Format: writes to the file format

Value	Description
0(default)	Float format storage
1	Text format storage

For example：
STICK_WRITEVR (0,0128,0) copy the first 128 elements of the VR into external memory in float format

⚠Does not have the external memory interface controller this instruction is not supported.

STICK_READVR

Type: Storage instruction
Grammar: value = STICK_READVR (filenum, vr_start [, format])
Description: copy data from external memory to VR. Value=TRUE indicates operation is successful, otherwise operation failed
Parameter: filenum: File number, corresponding to SD [filenum].BIN
vr_start: The start VR number of the start operation
format: file format

Value	Description
0(default)	Float format storage
1	Text format storage

For example：
STICK_READVR (0,0,0) copy external memory 0.bin file data to VR

⚠Does not have the external memory interface controller this command is not supported.

FLASH_WRITET

Type: Storage instruction
Grammar: FLASH_WRITE [sect_num, varname] [arrayname], [arrayname], [(a), arrayname (a, length)]
Description: store variables or arrays, individual or part elements of an array into the internal FLASH
Parameter: sect_num: FLASH block number, different types, FLASH_SECTES represents the total number of blocks.
Varname: variable name
Arrayname: array name, which can be TABLE, VR.
a: The array index of operation
length: The number of array elements of the operation
Reference to: FLASHVR, FLASH_READ
For examples: FLASH_WRITE, VAR, ARRAY1, ARRAY2 (1)

⚠ Internal FLASH using sequential storage mode, the read sequence must be consistent with the order of the storage time.

⚠ Internal FLASH has limited storage time, do not cycle operation at random.

FLASH_READ

Type: Storage instruction
Grammar: FLASH_READ [sect_num, varname] [arrayname], [arrayname (a) []], arrayname(a,length)]
Description: read the data to te vatiable or array from the internal FALSH.
Parameter: sect_num: FLASH block number, different types are not the same, FLASH_SECTES refers to the total number of blocks.。
Varname: variable name
Arrayname: array name, can be defined as TABLE or, VR.
a: The array index a operations
length: Number of array elements to be operated
Reference to: FLASHVR, FLASH_WRITE
For examples: FLASH_READ 1, VAR, ARRAY1, ARRAY2 (1)

⚠ Internal FLASH using sequential storage mode, the read sequence must be consistent with the order of the storage.

FLASH_SECTSIZE

Type: system status function
Grammar: value = FLASH_SECTSIZE
Description: read the internal FLASH, a block can store the number of variables.
Parameters: value: number, different controller types are difference
Reference to: FLASH_SECTES
For example：
FLASH_SECTSIZE 20480, ZMC1xx series, 1024, ZMC00x series

FLASH_SECTES

Type: system status function
Grammar: value = FLASH_SECTES
Description: the total number of blocks read the internal FLASH.
Parameters: value: number of block, different controller types are difference.
Reference to: FLASH_SECTSIZE
For example：

```
?FLASH_SECTES
1000 'ZMC1xx series
16 "ZMC00x series.
```

⚠ For ZMC00x series controller, FLASHVR will use the last block to avoid conflict.

U_WRITE

Type: Storage instruction
Grammar: U_WRITE sect_num [[varname]] [arrayname] [[arrayname (a)] [[arrayname (a, length)]]
Description: store variables or arrays, individual or part elements of an array to external memory
Parameter: sect_num: File number, corresponding to SD [filenum].BIN.
Varname: variable name
Arrayname: array name, can be TABLE, VR.
a: Array index of operations
length: Number of array elements to be operated on
Reference to: FLASHVR，FLASH_WRITE，STICK_WRITE，U_READ
For examples: U_WRITE 1, VAR, ARRAY1, ARRAY2(1)

⚠:Does not have the external memory interface controller this instruction is not supported.

⚠Format is 32 bit floating-point IEEE sequential storage, a variable or an array element occupying a float, can to do the file in advance through PC, and then read it with the U_READ instruction.

U_READ

Type: Storage instruction
Grammar: U_READ sect_num [[varname]] [arrayname] [[arrayname (a)] [[arrayname (a, length)]]
Description: read data from an external memory into a variable or array
Parameter: sect_num: FLASH block number, different types, FLASH_SECTES represents the total number of blocks
Varname: variable name
Arrayname: array name, can be TABLE, VR.
a: The array index operations
length: the operation of array elements number
Reference to: FLASHVR, U_WRITE
For example: U_READ 1, VAR, ARRAY1, ARRAY2 (1)

⚠ Does not have the external memory interface controller this command is not supported.

⚠The file format is 32 bit floating-point IEEE sequential storage, a variable or an array element occupying a float, can
to do the file in advance through PC, and then read it with the U_READ instruction

U_STATE

Type: system state function
Grammar: U_STATE
Description: check if the U disk is inserted, insert, return TRUE, or return FALSE.,
For example:

```
If U_STATE = TURE then
    U_READ 1, VAR, ARRAY1, ARRAY2(1)
End if
```

⚠ Does not have the external memory interface controller this command is not supported.

⚠Don't put too many files in the U disk

Chapter 11 Interrupt Correlation

INT_ENABLE

Type: system parameter
Grammar: INT_ENABLE = switch
Description: interrupt switch. In order to avoid program initialization and interrupt, interrupt switch is closed by default

Value	Description
0(default)	Close
1	Open

Reference to：ONPOWEROFF
For example：

```
>>?int_enable
  0
```

ONPOWEROFF

Type: interrupt
Grammar: GLOBAL ONPOWEROFF:
Description: power down interruption program entrance, must be SUB global.
For example:

```
INT_ENABLE = 1
END

GLOBAL, SUB, ONPOWEROFF (),
    VR (0) = dpos (0), save coordinates
    VR (1) = dpos (1)
    vr(2) = dpos(2)
END SUB
```

⚠Power-off interrupt the execution time is limited, can only write a few sentences.

ONTIMERn

Type: interrupt

Grammar: GLOBAL ONTIMERn:
Description: timer interrupt entrance, must be SUB global. There are currently 3 global timer.
Reference to: TIMER_START, TIMER_STOP, TIMER_IFEND
For example：

```
INT_ENABLE=1
TIMER_START(0,100)
GLOBAL SUB ONTIMER0
    (print) ontimer0 enter"
END SUB
```

TIMER_IFEND

Type: system function
Grammar: value = TIMER_IFEND (timernum)
Description: returns the end of the timer
Parameters: timernum: Number 0 - number of timers - 1
Reference to: ONTIMER, TIMER_START

TIMER_START

Type: system instruction
Grammar: TIMER_START (timernum, time_ms)
Description: start the system timer.
Parameter: timernum: Number 0 - number of timers - 1,
time_ms: timer length, unit millisecond
Reference to: ONTIMER, TIMER_IFEND

TIMER_STOP

Type: system instruction
Grammar: TIMER_STOP (timernum)
Description: stop the system timer.
Parameters: timernum 0- timer number -1
Reference to: ONTIMER, TIMER_IFEND
Chapter twelve EtherCAT related

SLOT_SCAN

Type: Bus instruction
Grammar: SLOT_SCAN (slot)
Description: the bus scan is successfully returned by RETURNBus Scanning，

the result of success or not will be return through using the instruction RETURN. After scanning, you can use the NODE instruction to read the relevant device information and configure it using DRIVE related instructions of DRIVE
Parameter: slot slots number 0- default
Reference to: SLOT_START

⚠ Encounter an unsupported device type, RETURN returns FALSE.

SLOT_START

Type: bus instruction
Grammar: SLOT_START (slot)
Description: the bus is started and returned successfully via RETURN.
Parameter: slot: slot number 0-, default
Reference to : SLOT_STOP, SLOT_SCAN

SLOT_STOP

Type: bus instruction
Grammar: SLOT_STOP (slot)
Description: the bus stopped and returned successfully via RETURN.
Parameter: slot: slot number 0-, default
Reference to: SLOT_START

SDO_WRITE

Type: bus instruction
Grammar: SDO_WRITE (slot, node, index, subindex, type, value)
Description: SDO write. Returns success by RETURN
Parameter: slot: slot number 0- default
Node: device number 0-
Index: data dictionary number
Subindex: sub number

Type	data type
1	Boolean
2	Integer 8
3	Integer 16
4	Integer 32
5	Unsigned 8
6	Unsigned 16
7	Unsigned 32

Value: data values
Reference to：

SDO_WRITE_AXIS

Type: bus instruction
Grammar: SDO_WRITE_AXIS (axis, index, subindex, type, value)
Description: SDO write. Returns success by RETURN
Parameters: Axis axis number
Index data dictionary number
Subindex sub number

Type	data type
1	Boolean
2	Integer 8
3	Integer 16
4	Integer 32
5	Unsigned 8
6	Unsigned 16
7	Unsigned 32

Value: data values.
Reference to: SDO_WRITE

SDO_READ

Type: bus instruction
Grammar: SDO_READ (slot, node, index, subindex, type, tablenum)
Description: SDO read. Success returned by RETURN
Parameter: slot: slot number 0- default
Node: device number 0-
Index: data dictionary number
Subindex: sub number

Type	data type
1	Boolean
2	Integer 8 3 Integer 16
4	Integer 32
5	Unsigned 8
6	Unsigned 16
7	Unsigned 32

The TABLE location of the data read by the tablenum
Reference to:

SDO_READ_AXIS

Type: bus instruction
Grammar: SDO_READ_AXIS (axis, index, subindex, type, tablenum)
Description: SDO read. Success returned by RETURN
Parameter: Axis: axis number
Index: data dictionary number
Subindex: sub number

Type	data type
1	Boolean
2	Integer 8
3	Integer 16
4	Integer 32
5	Unsigned 8
6	Unsigned 16
7	Unsigned 32

Tablenum: reads the TABLE location of data stored
Reference to: SDO_READ

?*ETHERCAT

Type: Assisted Instruction
Grammar: *ETHERCAT?
Description: used at debugging time to display the important status of each NODE.
Parameters:
Reference to: PRINT

NODE_COUNT

Type: bus instruction
Grammar: VAR = NODE_COUNT (slot)
Description: total number of equipment.
Parameters:
Slot: slot number, 0-, default see: NODE_INFO

NODE_STATUS

Type: bus instruction
Grammar: VAR = NODE_STATUS (slot, node)
Description: state of the equipment.

BIT	meaning
0	Specify whether node exist; 1-
1	Communicat ion state; 1- error;
2	Node state; 1- error;

Parameter:
Slot: slot number, 0-, default
Node: device number 0-
Reference to: NODE_INFO

NODE_AXIS_COUNT

Type: bus instruction
Grammar: VAR = NODE_AXIS_COUNT (slot, node)
Description: device drive number read.
Parameters:
Slot: slot number, 0-, default
Node: device number 0-
Reference to: NODE_INFO

NODE_IO

Type: bus instruction
Grammar: VAR = NODE_IO (slot, node) NODE_IO (slot, node) =iobase
Description: device IO start number setting, same number of start and output for individual devices.
Parameter:
Slot: slot number, 0-, default
Node: device number 0-
Reference to: NODE_AIO

NODE_AIO

Type: bus instruction
Grammar: VAR = NODE_AIO (slot, node[, idir]), NODE_AIO (slot, node[, idir]), =iobase
Description: AD/DA start numbering equipment settings.
Parameters:
Slot: slot number, 0-, default

Node: device number 0-
Idir: AD/DA, select
1- defaults to setting ain and aout at the same time. Only read ain.
3- ain
4-aout
Reference to: NODE_IO

NODE_INFO

Type: bus instruction
Grammar: VAR = NODE_INFO (slot, node, SEL)
Description: information read by bus device, must be read by bus scan:
Parameter:
Slot: slot number, 0-, default
Node: device number 0-
Sel: information number
0- VENDER, the number of manufacturers
1 - DEVICE, device number
2- VERSION, version
3 - ALIAS, alias, is commonly used to identify drives
4 - Reserve
The following is the number of IO:
10-, in numbers
11-, Op numbers
The number of 12-ain
13-aout numbers
14- reserved
Reference to: SLOT_SCAN

NODE_PROFILE

Type: bus command
Grammar: NODE_PROFILE (slot, node) = iprofile[, reserve]
Description: the profile settings for the bus device are reserved and cannot be modified after the bus is started.
Reference to:

NODE_PRESET

Type: bus instruction
Command Grammar 1:NODE_PRESET (slot, node, manuid, productid)
The command Grammar 2:NODE_PRESET (slot, -1) clears all preset Functions

Grammar 1: VALUE = NODE_PRESET (slot, node), and whether or not there is a preset
Functions Grammar 1: VALUE = NODE_PRESET function grammar (slot) returns the maximum number of preset value.
Description: the bus device is pre configured, the device has not been hung up, and the bus has been started in advance. The bus can not be modified after starting.
After setting up, the device can be judged by NODE_STATUS whether it is hung or not
The type of default value is different from reality and will not start the bus
The middle of the peripheral does not have a default value, there is no scanning to, nor start the bus.
Support more than 20160601 version
Reference to: NODE_STATUS
For example：

```
    Node_preset (0, -1) 'clears the original preset node_preset (0,0, $83, 5)
and sets the first NODE as the OMRON drive
    slot_scan(0)
   "Scan result:", "RETURN", "Max", and "node_count (0)" will automatically
   display 1 totals at this time
   for i= 0 to node_count(0) -1
   ? "node", i, "status",node_status(0,i), "manu:" ,NODE_INFO(0,i,0), "dev:"
 ,NODE_INFO(0,i,1), "motor:", NODE_AXIS_COUNT(0,i)
Next
```

DRIVE_FE

Type: axis status
Grammar: VAR = DRIVE_FE (axis)
Description: the current error of the drive is out of bounds. You must set the correct ATYPE before you can set it
Reference to: ATYPE

DRIVE_FE_LIMIT

Type: axis parameter
Grammar: DRIVE_FE_LIMIT = value
Description: the current error overrun setting of the drive. You must set the correct ATYPE before you can set it
Reference to: ATYPE

DRIVE_STATUS

Type: axis status
Grammar: toq = DRIVE_STATUS (axis)
Description: the current state of the drive corresponds to the data dictionary 0x6041. You must set the correct ATYPE before you can read it
Reference to: ATYPE, DRIVE_PROFILE

DRIVE_TORQUE

Type: axis status
Grammar: VAR = DRIVE_TORQUE (axis)
Description: the current torque of the drive must set the correct ATYPE and DRIVE_PROFILE before you can read it
Reference to: ATYPE, DRIVE_PROFILE

DRIVE_MODE

Type: axis parameter
Grammar: DRIVE_MODE = value
Description: the drive control mode corresponds to the data dictionary 0x6060. You must set the correct ATYPE before you can manipulate this parameter
Reference to: ATYPE, DRIVE_PROFILE

DRIVE_PROFILE

Type: axis parameter
Grammar: DRIVE_ PROFILE = value
Description: send PDO each axis to receive PDO configuration options, you must set the correct ATYPE before you can operate this parameter. Please consult the manufacturer for detailed configuration

-1 - Driver default settings need controller version 20160601 and above

0- defaults

{0x60400010, 0x607a0020, 0x60600008}, {0x60410010 / control word target location mode, 0x60640020} / / word position feedback

1-

{0x60400010, 0x607a0020, 0x60600008}, //The position of the target mode control word

{0x60410010, 0x60640020, 0x60770010}, // state word position feedback

current moment
2-
{0x60400010, 0x607a0020, 0x60600008, 0x60b80010}, probe / control word target position setting mode
{0x60410010, 0x60640020, 0x60770010, 0x60b90010, 0x60ba0020},
/ / state word position feedback current moment probe state probe
3-
{0x60400010, 0x607a0020, 0x60600008, 0x60b80010, 0x60720010},
The position of the target mode control word / probe set the torque limit
{0x60410010, 0x60640020, 0x60770010, 0x60b90010, 0x60ba0020}, / /
state word position feedback current moment probe probe position
4-
{0x60400010, 0x607a0020, 0x60600008}, mode / control word target position {0x60410010, 0x60640020, 0x60770010, 0x60fd0020},
State feedback input current position / torque drive IO (using only 16 input)
5- firmware version 160504 supports {0x60400010, 0x607a0020, 0x60fe0120,0x60600008},
The position of the target word control / / IO output driver (32 output mode)
{0x60410010, 0x60640020, 0x60770010, 0x60fd0020}, state feedback input current position / torque drive IO (32 input)
6- special drive dedicated
7- special drive dedicated
8- special drive dedicated
9- firmware version 160504 support
{0x60400010, 0x607a0020, 0x60fe0120, 0x60b80010, 0x60720010, 0x60600008}, IO / control word target position output (32) probe set the torque limit mode
{0x60410010, 0x60640020, 0x60770010, 0x60fd0020, 0x60b90010, 0x60ba0020}, state feedback input current position / torque drive IO (32) probe probe position
10- above for custom range
-1, which indicates the built-in default PDO list using the drive, is supported by more than 20160601. Default PDO does not
Datum (21) zeroing instructions cannot be used when 0X6060 is used

⚠ The drive IO sets the IO number of the mapping through the DRIVE_IO

Reference to: ATYPE, DRIVE_IO, REGIST

DRIVE_CONTROLWORD

Type: axis parameter
Grammar: DRIVE_CONTROLWORD = value
Description: drive control word, you must set the correct ATYPE to operate this parameter. When ATYPE=65, the control word will switch

automatically according to WDOG/AXIS_ENABLE to enable the drive

For example：

DRIVE_CONTROLWORD=0 DELAY 100

DRIVE_CONTROLWORD=6 DELAY 100 DRIVE_CONTROLWORD=15

Reference to: DRIVE_CW_MODE

DRIVE_CW_MODE

Type: axis parameter

Grammar: DRIVE_CW_MODE = value

Description: you must set the correct ATYPE before you can operate this parameter. Some versions are always easy to use and can always be straight connection operate MODE.

For example：

Reference to: DRIVE_CONTROLWORD

DRIVE_IO

Type: axis parameter

Grammar: DRIVE_IO = value

Description: the start number of the DRIVE configuration of the IO is directly configured, and the inputs and outputs are equally numbered. DRIVE_PROFILE Function while reading the drive IO.

For example:

Reference to: NODE_IO, DRIVE_PROFILE

Chapter 12 Simple Routines

IO operation

```
While 1      'cycle
    if  in(0) = on    then      ‘ 'input 0' is valid
    op(2, off)    "the closure of the output port 2
    else
      op(2, on)‘open the output port 2
   endif
 wend
```

String use and custom communication

```
Routine 1:
SETCOM(38400,8,1,0,0,0)
DIM TEMPVAR
DIM VALUE
DIM CHLIST(10)
For i=0 to 9
    GET #0, TEMPVAR
    CHLIST(i)=TEMPVAR
Next
TRACE CHLIST "debugging
VALUE = VAL (CHLIST) change into variables
PRINT #0 TOSTR (CHLIST) "into a string output
```

SP instruction continuous interpolation

```
ERRSWITCH = 3 'full information output
base (0, 1, 2, 3) 'choose X, Y, and Z
rapidstop
Wait idle

dpos = 0,0,0,0
atype = 1, 1, 1 'pulse mode step or servo
units = 100, 100, 100 'pulse equivalent, every mm100 pulse
speed = 200200200200" this speed will limit FORCE_SPEED = accel =
2000200020002000
decel = 2000200020002000, merge = on 'starts continuous interpolation
CORNER_MODE = 0 'does not start automatic corner deceleration, manually
setting STARTMOVE_SPEED,
'ENDMOVE_SPEED'DECEL_ANGLE = 15 * (PI/180) start deceleration angle 15
degrees'STOP_ANGLE = 45 * (PI/180) to minimum speed angle 45 degrees

While 1 'cycle movement
    "If in (0) = on then" enter 0 valid startup movement
         TRACE "start movesp""
     Take a box and the speed and stopping speed of each section are
 different
       FORCE_SPEED = 100', Section 1 speed 100
       ENDMOVE_SPEED = 10'at the end of the first paragraph 10,
       MOVESP(100,0)

     FORCE_SPEED = 150'second segment speed 150
```

```
        ENDMOVE_SPEED = 15
        STARTMOVE_SPEED = 15 'the starting speed of the second segment is
15, because it's faster than the end of the first segment
'10, so the actual initial velocity is 10
        MOVESP(0,100)
        FORCE_SPEED = 200'third segment speed 200
        ENDMOVE_SPEED = 20
        STARTMOVE_SPEED = 20 'the starting speed of the third segment is 20,
because it is faster than the end of the second segment'15, so the actual
starting velocity is 15
        MOVESP(-100,0)
        FORCE_SPEED = 300'fourth segments, speed 300, subject to SPEED limit,
in fact 200
      ENDMOVE_SPEED = 30,
      STARTMOVE_SPEED = 30, the starting velocity of the third segment is 30
because it is higher than the end velocity of the second segment
Degree'20, so the actual starting speed is 20
      MOVESP(0,-100)
      Wait idle 'wait for movement to stop
      delay (100)' delay
    end if
  wend
End
```

The handwheel

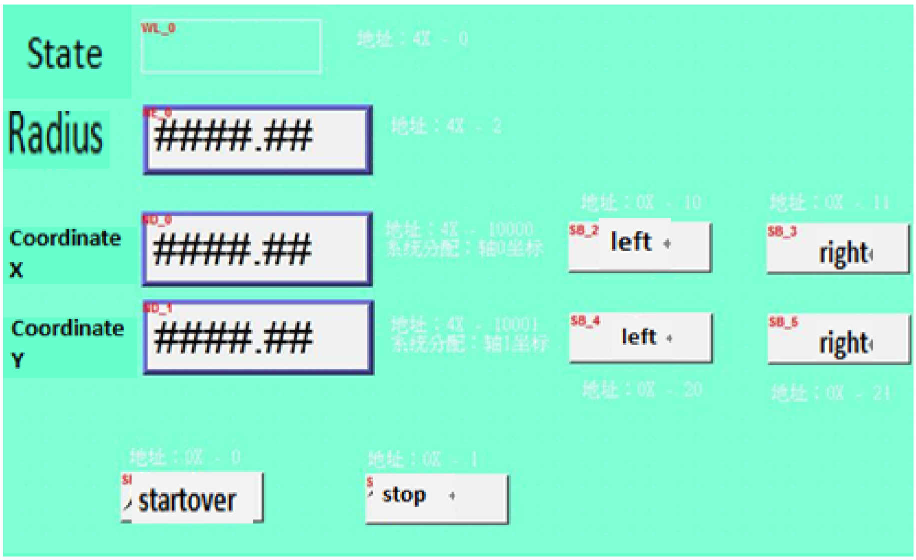
```
ERRSWITCH = 3 'all information output
const axishand = 0
base (axishand) select zeroth axis handwheel
Atype=3 'pulse + direction handwheel, quadrature input handwheel using 3

Base (1) 'axis 1 handwheel control
atype=1' step
dpos = 0
Units = 100 'pulse equivalent, per mm100 pulse
speed = 200
accel = 3000
decel = 3000
sramp = 20
Clutch_rate = 0'is limited by speed and acceleration
Dim poslast        'records a position variable
poslast = dpos
while 1
        if in(0) = on and in(1) = off then
            Connect (1, axishand) 'link to the axis 0, the ratio of 1
        elseif in(1) = on and in(0) = off then
            Connect (10, axishand) is linked to axis 0, magnification 10
        elseif in(0)= on and in(1) = on then
```

Connect (50, axishand) link to axis 0, magnification 50, to step, magnification is too high, there will be lost step or to end a long time

```
Elseif mtype = 21 then 'Cancel connect
     cancel
endif
if poslast <> dpos then
  poslast = dpos trace dpos
  endif wend
endif
wend
end
```

Touch screen communication

The routine touch screen address starts at 0, and some touch screen addresses begin at 1, with 0. of the 1 corresponding to the touch screen from 1



```
ERRSWITCH = 3      "all information output
base(0,1)    ' 'select X Y
rapidstop
wait idle
units = 100,100
speed = 100,100
accel = 1000,1000
decel = 1000,1000
```

```
dim radius  '' radius
radius = 100 'default size
modbus_ieee(2) = radius

dim run_state    'running state'
run_state = 1     '1    Stop, 2, run
modbus_reg(0) = run_state

dim stop_flag    'Stop sign

runtask 2, guidetask        guidetask'start manual scanning task
modbus_bit(0) = 0         Button reset
modbus_bit(1) = 0         Reset 'button
While1 ' scan MODBUS input
If modbus_bit (0) = 1 then 'start button
   Modbus_bit (0) = 0 'button to return
   if run_state = 1 then
         trace "move start"
         Stoptask 1        'Insurance
         Runtask 1, movetask      'starts running tasks
     end if

Elseif modbus_bit (1) = 1 then 'stop button press
            trace "move stop"
            Modbus_bit (1) = 0 'button reset
            stop_flag = 1
        endif
  wend
  end

Movetask:    'run
Base (0,1)    'select X Y
stop_flag = 0

run_state =2
modbus_reg(0) = run_state
Radius = modbus_ieee (2)    'read radius

Movecirc (0,0, radius*cos (pi/4), radius*sin (pi/4), 0) walk a whole circle
Wait until idle or (stop_flag = 1)    'wait for the end or stop
if (stop_flag = 1) then
      cancel
endif
```

```
run_state = 1
modbus_reg(0) = run_state
end

Guidetask: 'manual operation
while 1
    if run_state <> 2 then
       base(0)
       If modbus_bit (10) = 1 then 'left
       vmove(-1)
elseif modbus_bit(11) = 1 then  'right

              vmove(-1)
          elseif modbus_bit(11) = 1
          then                            'right
             vmove(1)
          elseif mtype = 10 or mtype = 11     The 'non vmove
          then                                movement
              cancel
          endif

          base(1)
          if modbus_bit(20) = 1
          then                            'left
              vmove(-1)
          elseif modbus_bit(21) = 1
          then                            'right
             vmove(1)
          elseif mtype = 10 or mtype = 11     The 'non vmove
          then                                movement

              can

          cel endif

     endif

wend end
```

Custom G code

```
ERRSWITCH = 3 'all the information output
base (0,1,2,3) X Y Z U' G01, which has specified, cannot modify the rapidstop
wait idle

dpos = 0,0,0,0
```

```
Atype=1,1,1,1 'pulse mode stepping or servo
units = 100100100100' pulse equivalent, per mm100 pulse
speed = 200200200200
Accel = 2000200020002000
decel = 2000200020002000
merge = on 'start continuous interpolation
CORNER_MODE = 2', start corner deceleration,
DECEL_ANGLE = 15 * (PI/180),
STOP_ANGLE = 45 * (PI/180)
G_init ()          'G initialization
While 1             'cycle movement
   If in (0) on = then 'input 0 effective start movement
      Take a box
      G91 'relative
      G01 X100 Y0
      G01 X0 Y100
      G01 X-100 Y0
      G01 X0, Y-100,
      wait, idle 'wait for movement to stop
      delay (100) delay
  end if
 wend
end
global sub g_init()
dim coor_rel
Coor_rel = 1 'relative
end sub
End 'make this file run automatically when exiting
global gsub G01(X Y Z U)
    Trace "G01, entered, distance:", sub_para (0), sub_para (1), sub_para (2),
sub_para (3) "debug output."
   if coor_rel then
     Move (sub_para (0), sub_para (1), sub_para (2), sub_para (3)) is quite
else
   local xdis, ydis, zdis, udis
   If sub_ifpara (0) then "have parameters
      xdis = sub_para(0)
   else
      xdis = dpos(0)
   endif
   if sub_ifpara(1) then
       ydis = sub_para(1)
  else
     ydis = dpos(1) endif
```

```
      if sub_ifpara(2) then
          zdis = sub_para(2)
     else
 zdis = dpos(2)
 endif

            if sub_ifpara(3) then
                  udis = sub_para(3)
            else
                  udis = dpos(3)
            endif
            moveabs(xdis,ydis,zdis,udis)                 'absolutely
      endif
end sub

'absolutely
global gsub G90()
      trace "G90 entered"
      coor_rel = 0
end sub

'Relative
global gsub G91()
      trace "G91 entered"
      coor_rel = 1
end sub

'delay
global gsub G04(P)
      trace "G04 entered"
            if sub_ifpara(0) then
                  delay (sub_para(0))
            else

            endif
end sub

global gsub G00(X Y Z U)
                                                                        Debugtransf
      trace "G00 entered, distance:"                                    er
      sub_para(0),sub_para(1),sub_para(2),sub_para(3)                     go out

      if coor_rel then
            pmove(sub_para(0),sub_para(1),sub_para(2),sub_para(3))
      else
```

```
        local xdis, ydis, zdis,
        udis if sub_ifpara(0)
        then
            xdis = sub_para(0)
        else
            xdis =
        dpos(0) endif
        if sub_ifpara(1) then

            ydis = sub_para(1)
        else
            ydis = dpos(1)
        endif
        if sub_ifpara(2) then
            zdis =
            sub_para(2)
        else
            zdis = dpos(2)
        endif
        if sub_ifpara(3) then
            udis =
            sub_para(3)
        else
            udis = dpos(3)
        endif
        pmoveabs(xdis,ydis,zdis,u
    dis) endif
end sub
```