Zmotion RTBasic Program Manuals

Version 1.1.0

Foreword

Zmotion Technology is a national high-tech enterprise, which devotes to study motion control and general motion control products. Zmotion Technology has attracted experienced talents from famous companies or institutions, such as Huawei, ZET, Huazhong University of Science and Technology etc. Zmotion insists self- innovating and collaborating with comprehensive universities to research basic knowledge of motion control. Due to its concentration and hard work in motion control technology, ZMOTION already become one of the fastest growing industrial motion control companies in China, and it is also the rare company that has managed core technologies of motion control and real time industrial control software completely.

All Zmotion products development obeys standard IPD-CMM from Huawei, which means they have stability and reliability of telecom level, perfect compatibility and expansibility of software or hardware. Zmotion provides powerful and convenient ZDevelop development environment, and it supports ZBasic, ZPLC ladder diagram and ZHMI configuration second-development and hybrid program. Real time simulation and online-tracking Debug are available, also it supports all kinds of control systems call different program language's function library.

This manual is to better serve customers and provide customers with more comprehensive reference materials. ZMOTION is committed to the continuous optimization and improvement of products, so that customers can quickly understand our products. And the product manuals will also be updated all the way.

This manual includes the use of Zmotion ZDevelop software, detailed instruction, program operation logic description, motion buffer principle, expansion module, axis application, controller introduction and wiring reference between the controller and other components, and it can support multiple communication methods. In addition, application routines of typical industries are provided for programming reference.

Relative Program Manuals:

ZMotion PC Function Library Program Manual
ZMotion PLC Program Manual
ZMotion HMI Program Manual
ZMotion Robotic Arm Instruction Description Manual
ZDevelop Use Manual

These materials and other hardware manuals all can be downloaded from http://www.zmotionglobal.com

Safety Tips

1. Precautions

- The controller is highly integrated, and it is designed to be small and lightweight, that's easy to install, and users can use the space efficiently. The controller can be installed on a panel or standard rail, and it can be installed horizontally or vertically.
- As the basic rule for installing and arranging various equipment in the system, isolate low-voltage logic equipment such as controllers from heat radiation, high voltage and electrical noise, and keep away from dust, corrosive gases, water, oil, chemicals and other places.
- When configuring the layout of the controller on the panel, since the controller will generate heat when running for a long time, it should be considered to arrange the controller in a cooler area, and less exposure to high temperature environment will prolong the service life of electronic equipment. The high temperature environment may cause that the controller can't be used normally.
- Also, wiring of the equipment in the panel should be considered during installation. Avoid laying low-voltage signal lines and communication cables in slots with AC power lines and high-energy fast-switching DC lines.
- Please leave sufficient interspace around the controller for cooling and wiring of the controller. The controller can be cooled by natural convection. To ensure proper cooling, interspace must be left above and below the equipment.

2. Warnings

- The controller is a weak current device, so it needs to be installed in a place that is not easy to touch, such as the casing, control cabinet or electric control room, to avoid being touched by non-operators.
- For your personal safety, please do not approach the machine when it is running.
- Do not disassemble, repair or modify this product.
- External use of control circuit to form emergency stop circuit, interlock circuit, limit circuit is equal to the circuit related to safety protection.
- When installing or removing the controller, the power supply of the control system should be completely disconnected to avoid unnecessary losses caused by electric shock or accidental equipment operation.
- Failure to comply with the above requirements may result in serious personal injury and

property damage, and Zmotion Technology does not assume the corresponding risks and responsibilities.

3. Wiring Requirements

- Use screws to fix the controller to prevent the product from being dropped or subjected to abnormal shocks and cause malfunctions.
- In order to ensure stable communication, the communication cable of the controller should be a high-performance cable with shielding layer.
- Use 24V DC power supply to supply power to the controller, and the IO port needs to be powered separately, which is separate from the controller power supply.
- Each component of the controller network for safety reasons, ensure that all commons and grounds of the controller and related equipment are grounded at the same point, which should be directly connected to the system earth ground. When determining the grounding point, the safety grounding requirements and the proper operation of the protective interrupting device should be considered.
- After completing all wiring work, power on the control circuit, do not operate with power on.
- All line connections should be as short as possible to reduce interference and ensure communication quality.

Copyright statement

ZMC controller software involved in details as well as the introduction and routines of each instruction, please refer to ZBASIC software manual.

Information contained in this manual is only for reference. Due to improvements in design and functions and other aspects, Zmotion Technology reserves the final interpretation! Subject to change without notice!

Pay attention to safety when debug the machine! Be sure to design effective safety devices in the machine, and add the error handling procedures in software. Zmotion has no obligation or responsibility for the loss.

Content

Chapter I Introduction of Motion Control Products

1.1 Motion Control Product Overview

Motion control achieves real-time control of position, speed, acceleration, etc. of mechanical transmission components, so that it can complete corresponding motions according to expected trajectory and specified motion parameters.

The control system takes the processor, detection mechanism, and actuator as the core to realize logic control, position control, trajectory processing control, robot motion control, etc. The processor is usually a programmable controller, a single-chip microcomputer, or a motion controller, which is equivalent to the brain of the system. It is mainly responsible for logically processing the received signals, and issuing commands to the actuator to coordinate the normal operation of the system. The detection mechanism is usually composed of various sensors, which are equivalent to the eyes of the system. The purpose is to detect condition's changes in system and feed them back to the controller. The actuator is usually composed of servo units and valves, which are equivalent to the hands of the system. It is mainly for executing the commands issued by controller.

The motion controller is the core component of the motion control system. It is responsible for generating the control instructions of the motion path, and also it is used for the logical control of the equipment, assigning motion parameters to the axes that need motion, and responding to changes in the external environment of the controlled object in time.

General motion controllers usually provide a series of motion planning methods, based on the limitation of the magnitudes such as impact, acceleration and speed that can affect the accuracy of the dynamic trajectory, and provide the setting of motion parameters and motion-related instructions for the motion control process, so that it completes the corresponding actions according to the pre-specified motion parameters and the specified trajectory.

1.2 Motion Control Product Advantage

ZMOTION motion control products include pulse standalone motion controller, pulse network motion control card, fieldbus standalone motion controller, fieldbus PCI motion control card, etc. These can meet motion control requirements from all walks of life, a single controller supports 128 axes motion control.

Motion control products support multi-interpolation motions, such as, interpolation of linear, circular, space arc, ellipse, helical, etc. A single interpolation channel support most 16 axes joint interpolation. These products support speed look-ahead, electronic cam, electronic gear, pitch compensation, synchronous follow, motion superposition, virtual-axis, precision output, hardware position latch, continuous interpolation, motion pause and other functions. Some motion control products internally set more than 30 kinds of robot motion control algorithm, such as, SCARA, DELTA, 6-joint, etc. one controller can control several robotic arms, and it supports superpose multi-robot. Please see "ZMotion Robotic Arm Instrction" for details of robotic arm.

Fieldbus motion control products support EtherCAT, RTEX industrial Ethernet motion control bus. They lead in the performance and stability, and support EtherCAT bus, RTEX bus and pulse axes, these three kinds mixed use.

The first domestically launched dual-bus PCI control card and dual-bus motion controller that supports both EtherCAT bus and RTEX bus. The fastest EtherCAT bus cycle is 100 microseconds, and it also supports bus axis hardware position latch and position comparison output.

ZMOTION provides powerful ZDevelop software development environment, which is easy to learn and operate.

Motion controller supports Ethernet, U disk, CAN bus, RS485, RS232 serial port and other communication interfaces, and controller can link to ZMOTION expansion module through CAN bus or EtherCAT bus to expand inputs and outputs and pulse motion axes (a 120Ω resistor should be connected between CAN bus two terminals, CANL and CANH).

Advantages:

✧ The hardware composition is simple, the system can be composed by connecting the motion controller to the PC.

✧ Except ZDevelop software, there also supports all kinds of operation systems and program language to develop upper computer software (such as, VC, VB, C#, PYTHON, LABVIEW, etc.).

- ✧ Motion control software has wonderful code commonality and portability.
- ✧ It is easy to be learnt and developed, which means no need of too much training work, and it support several persons develop at the same time.

1.3 Controller Main Function Description

Item		Description
Task		Execute I/O refresh of specified condition and user procedure function, support multi-task run simultaneously, they don't interrupt each other, the maximum task number can be checked in ZDevelop software "Controller Status"
Debug		Support interruption point debug and single-step debug, and check task operation status
Interrupt		Support three kinds, externally interrupt, timer interrupt, power-off interrupt
Set Monitor Window		Monitor variable, constants, input and output, axis parameters, etc.
Program Language Type		ZDevelop program (BASIC, PLC, HMI), or other common upper computer program language
Online Command		Input instruction parameters in online command bar and then send them to controller for executing immediately
Communication Interface	Serial	232 serial and 485 serial ports, they support MODBUS_RTU protocol and self-defined communication
	Net	Fast communication speed, convenient wiring, it supports MODBUS_TCP protocol and self-defined communication
	U Disk	Insert U disk, data interaction
	CAN bus	Connect to ZIO expansion module, and controllers interconnect
	EtherCAT	Connect to EtherCAT drive or EtherCAT expansion module
	RTEX	Connect to RTEX drive
Data Type	Self-defined Array	Sets elements of the same data type, default floating point type
	Self-defined Variable	Default floating point type
	Self-defined Constant	It can be Boolean type, character string type, time type, date type, integer type, etc.
	Register	It comes with 4 kinds of registers, TABLE, MODBUS, VR, FLASH

Common Motion Control Functions	Point to Point	JOG point motion
	Interpolation	Interpolation of linear, circular, space, arc, ellipse, helical, support continuous interpolation
	Electronic Gear	Build electronic gear connection between main axis and slave axis.
	Electronic Gam	Cam watch motion and automatic cam
	Motion Superposition	Motion superposition of different axes
	Path Speed Look-ahead	Speed self-optimization according to lookahead parameters
	Position Latch	Memory axis position according to external signal trigger situation
	Position Comparison Output	Arrive comparison point, output OP signal, and compare continuously, respond rapidly
	Precision Output	OP respond rapidly

1.4 Applications of Controller

The motion control products of Zmotion Technology have been developed and applied by many partners for many years, and the products are widely used in 3C electronic semiconductors, dispensing equipment, laser processing, printing and packaging, special machine tools, robots, stage entertainment, medical equipment and other automation fields.

The electronic product processing industry includes placement machines, glue dispensers, printed circuit board drilling machines, winding machines, welding machines, loading and unloading robots, screw tightening machines and other equipment.

The textile and garment industry has warp knitting machines, dyeing machines, printing machines, industrial sewing machines, embroidery machines, cloth cutting machines, combing machines, twisting machines, shoe-making machines and other equipment.

The printing and packaging industry includes automatic blow molding machine, bag making machine, die-cutting machine, bronzing machine, unpacking machine, packing machine, labeling machine, automatic particle packaging machine, bag packaging machine, newspaper printing machine, gravure printing machine, etc.

Where there is automation equipment, there is motion control. With its excellent performance and perfect functions, Zmotion controllers can provide the best solutions for all walks of life.

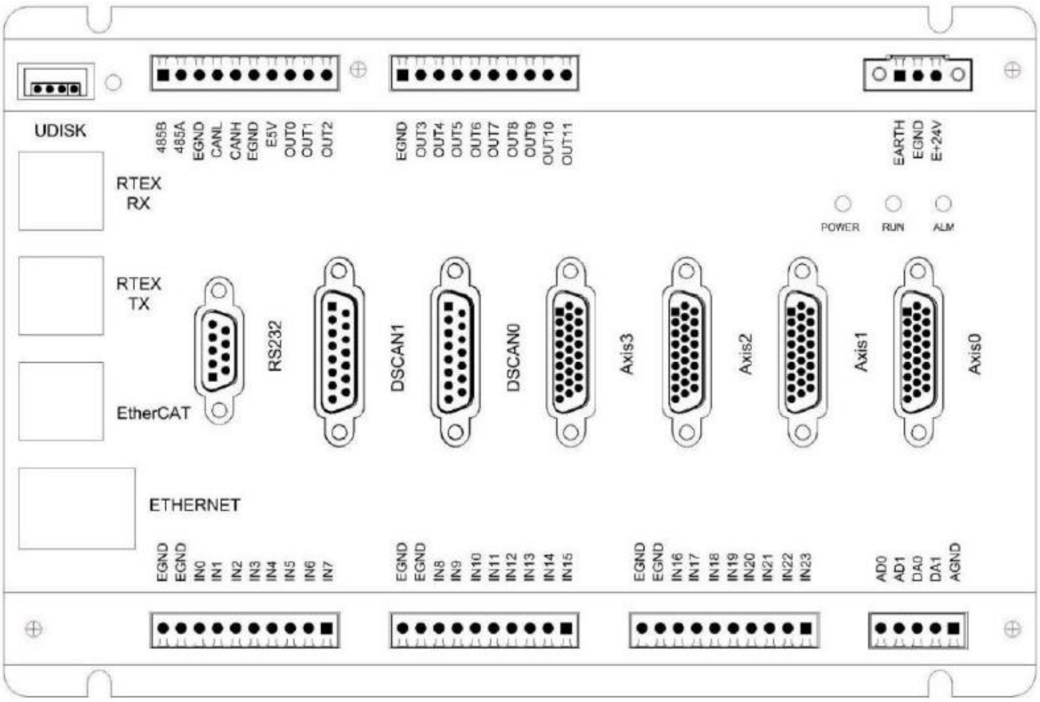
1.5 Controller Interface

Here the example is ZMC420SCAN bus motion controller

ZMC420SCAN bus motion controller supports EtherCAT bus and RTEX bus connection, and it supports at most 20 axes motion control, and several kinds of axes (pulse axis, EtherCAT bus axis, RTEX bus axis, encoder axis, galvanometer axis, virtual axis) can be hybrid interpolated. And it supports full-function motion control. For product specific parameters, please see hardware manuals.

ZMC motion controller supports Ethernet, U disk, CAN bus, RS485, RS232 serial port and other communication interfaces, and controller can link to ZMOTION expansion module through CAN bus or EtherCAT bus to expand inputs and outputs and pulse motion axes (a 120Ω resistor should be connected between CAN bus two terminals, CANL and CANH). “Expansion Module” can refer to the expansion methods.

ZMC420SCAN is like:



Interface Function:

Specification	Interface	Number	Description
RS232	232 serial-port	1	Use MODBUS_RTU protocol
485	485 serial-port	1	Use MODBUS_RTU protocol
CAN	CAN bus	1	Connect CAN expansion module or controller
ETHERNET	Net	1	Use MODBUS_TCP protocol, expand

			interface number through switch
EtherCAT	EtherCAT bus	1	Connect to EtherCAT drive or EtherCAT expansion module
RTEX	RTEX bus	1	Connect to RTEX drive
UDISK	U disk	1	Insert U disk equipment
E +24V	Main power	1	24V DC power supply
IN	Digital input	24	NPN type, internal 24V power
OUT	Digital output	12	NPN type, internal 24V power
AD	Analog input	2	Precision 12-bit, 0-10V
DA	Analog output	2	Precision 12-bit, 0-10V
DSCAN	Galvanometer axis interface	4	Connect to laser galvanometer, support XY2-100 protocol
Axis	Pulse axis interface	4	Each interface includes pulse output and encoder input

1.6 Controller's usage

✧ Prepare Work

Software: install ZDevelop program software or other upper computer program software supported by controller (VC, VB, C#, PYTHON, LABVIEW).

Equipment: select controller, computer, 24V DC power supply, drive, step motor or servo motor, wiring terminal, IO equipment, expansion module and other equipment according to specific requirements.

Connecting line: the connection line for communication between computer and controller, the connection line between drive axis interface and controller, and other connecting line of IO interface, power interface.

✧ Procedure design

1. System Diagram Design

Select the required components and connecting lines according to functional requirements, and please be familiar with the use of control instructions related to the function, and design the overall composition of the system software, including variable design, task design, program function design, etc.

2. Software Setup and Program

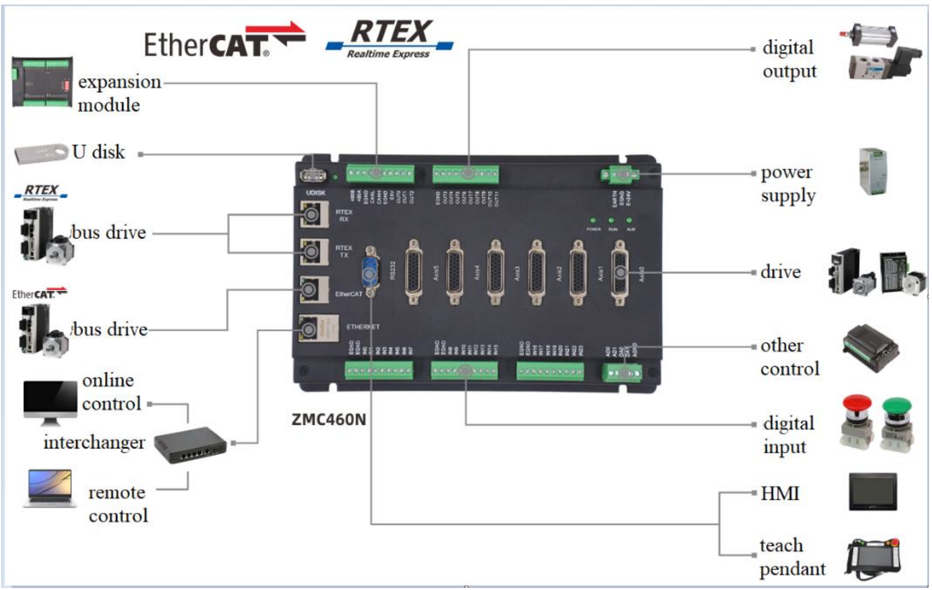
Use ZDevelop software to write programs according to the design in step 1. For quick use of the software, please refer to the "New Project" section of this article, or open the ZDevelop software menu bar "Help" - "ZDevelop Help" to view the introduction of the various functions of the software, writing tasks and program module for program simulation debugging.

Parameters that need to be set for programming: BASE select the axis numebr, ATYPE axis type, UNITS pulse equivalent, SPEED axis speed, ACCEL axis acceleration, DECEL axis deceleration and other basic axis parameters, and then send motion commands to the axis.

If the drive is connected by using the EtherCAT bus or the RTEX bus, a bus initialization operation is required during programming (see the "Bus initialization" routine). If expansion modules are required, such as expansion of axes or IO points, axis mapping needs to be performed on the extended axis resources during programming (see "Axis Mapping"). IO mapping is required for extended IO resources, and ZCAN expansion uses the DIP switch on the expansion board to set the number of the extended IO (refer to the chapter "ZCAN expansion module"), the EtherCAT bus extension uses the NODE_IO instruction to set the number of the extended IO, and the extended resources can be accessed through the IO number.

✧ Install and Wiring

Install various units, and connect each unit to the controller with appropriate cables. The wiring diagram of the controller is as follow:

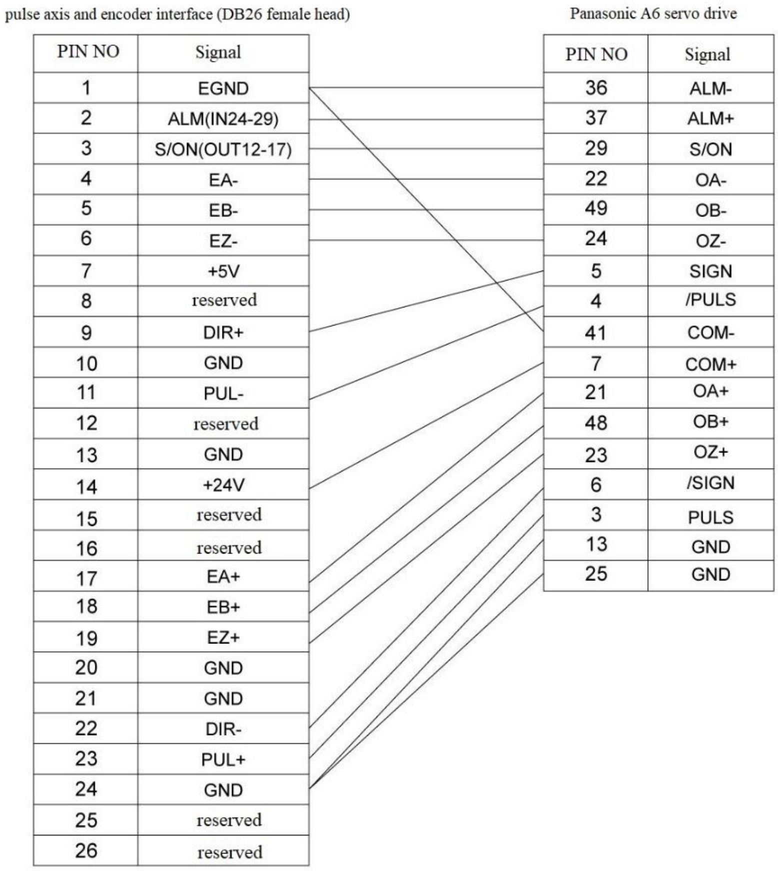


➢ Wiring between computer and controller:

Serial or network port can be used to communicate. When using the serial port communication, RS232 serial port of the controller should be connected. When using the network port communication, the EtherNET network port of the controller should be connected.

- Wiring between drive and controller:

  The driver can link to the pulse axis interface, EtherCAT bus interface, and RTEX bus interface of the controller. Refer to the figure below when the driver is connected to the pulse port. To use the bus to connect the driver, just use the network cable to directly insert the corresponding EtherCAT or RTEX interface.



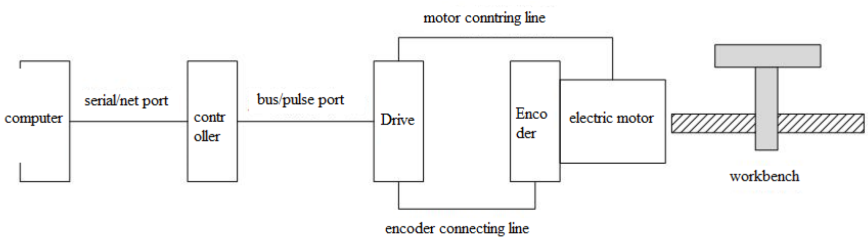
- Power wiring:

  Connect the positive pole of the +24V DC power supply to the 24V interface of the power supply module of the controller, the negative pole to the GND interface, the motor to the 220V AC power supply, and the IO device to the corresponding IO interface of the controller. Some models of the controller IO need to be powered by a 24V DC, and IO power is supplied separately, then can be used later.

- Expansion wiring:

  Support expand IO or pulse axis through CAN bus or EtherCAT bus. For details, please refer to the "Module Extension" chapter.

- Configuration reference:



✧ **Trail Running**

After confirming that the wiring is correct, then power on, download the debugged program to the controller, and start trial operation. Use the oscilloscope window or other parameter monitoring windows to confirm that the action is as desired.

Chapter II ZDevelop Software Program

2.1 Program Software Introduction

ZDevelop is a PC-side program development debugging and diagnosis software for ZMoiton series motion controllers. Through it, users can easily edit and configure the controller program, quickly develop applications, monitor the axis running parameters in real time, and real-time debug the running program of controller. And it supports Chinese and English bilingual environment.

ZDevelop programming software supports ZBasic, ZPLC ladder diagram, ZHMI configuration programming. ZBasic is the Basic programming language used by ZMotion motion controller, and provides all standard program grammar, variables, arrays, conditional judgments, loops and mathematical operations. This extended Basic instruction and function provides a wide range of motion control functions, such as single-axis motion, multi-axis synchronization and interpolation motions, as well as digital, analog and IO control.

ZBasic supports below functions:

- Self-define the SUB procedure, some general functions can be written as a self-defined SUB procedure, which is convenient for program writing and modification.
- SUB procedure with G code form, which supports G00, G01, G02, G03, G04, G90, G92 and other common instructions.
- Support global variables (GLOBAL), array and SUB procedure. Support file module variables, array and SUB procedure. Support local variables (LOCAL).
- Interruption procedure (power-off interruption, external interruption, timer interruption), such as, power-off interruption, save data through power off interruption, which can recover the power-off status.

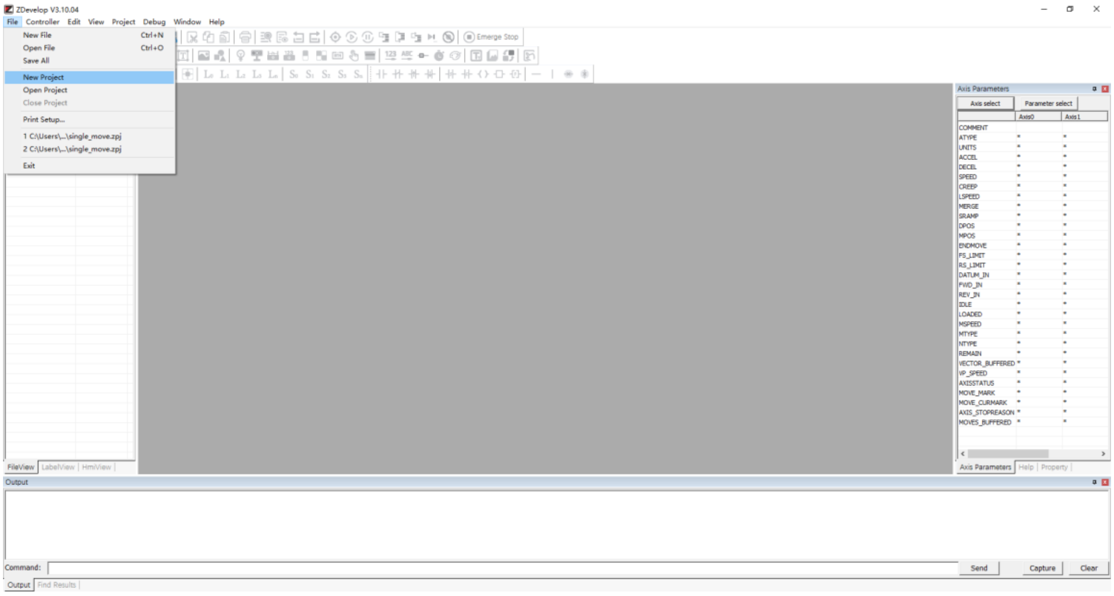
ZBasic has the real-time multi-task property, multi ZBasic procedures can build at the same time and multi-task real-time operation, which makes the complex application simpler.

PC online send Basic commands also can realize the same effect, the inner Basic program of controller and PC online Basic commands can multi-task run simultaneously.

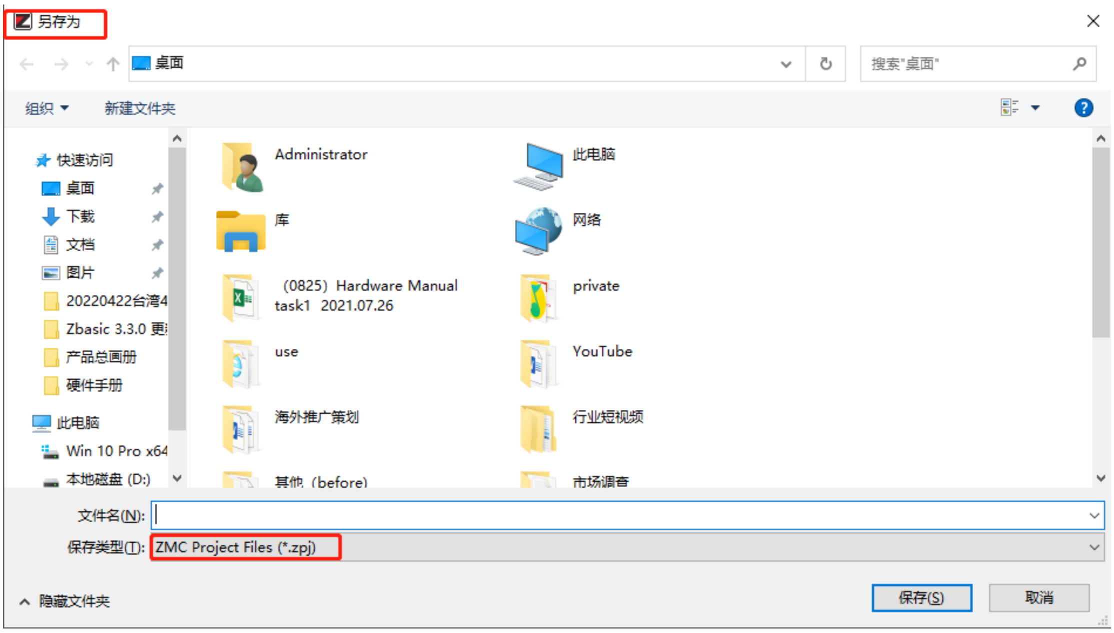
2.2 New Project

Please build a new folder to save the project that is to be built. Open ZDevelop programming software, here shows ZDevelop V3.10. Please visit ZMOTION website (www.zmotionglobal.com) to update software version.

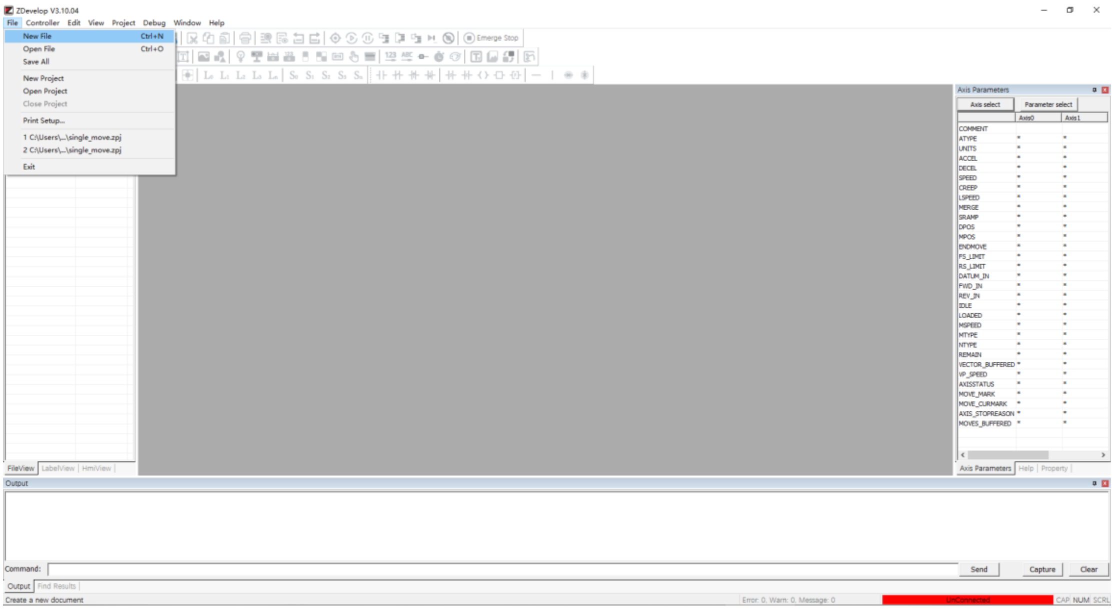
1. New build item: “File” in “Menu” → “New Project”.



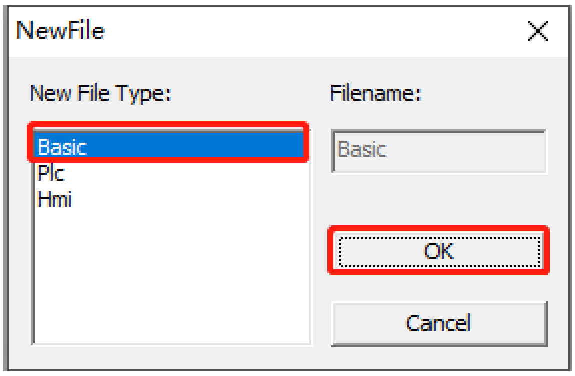
2. Click “New Project”, then “Save as…” will be jumped, select one folder and open it. Input folder’s name and save the project, pay attention to the suffix should be “.zpj”.



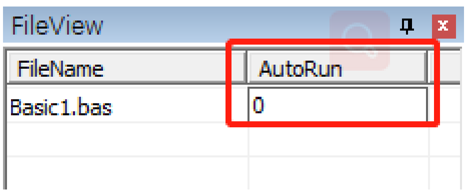
3. New build a file: “File” → “New File”.



After clicking “New Project”, below jumping window will appear, which supports Basic/PLC/Hmi hybrid programming. Here selects the “Basic” file type and click “OK”.



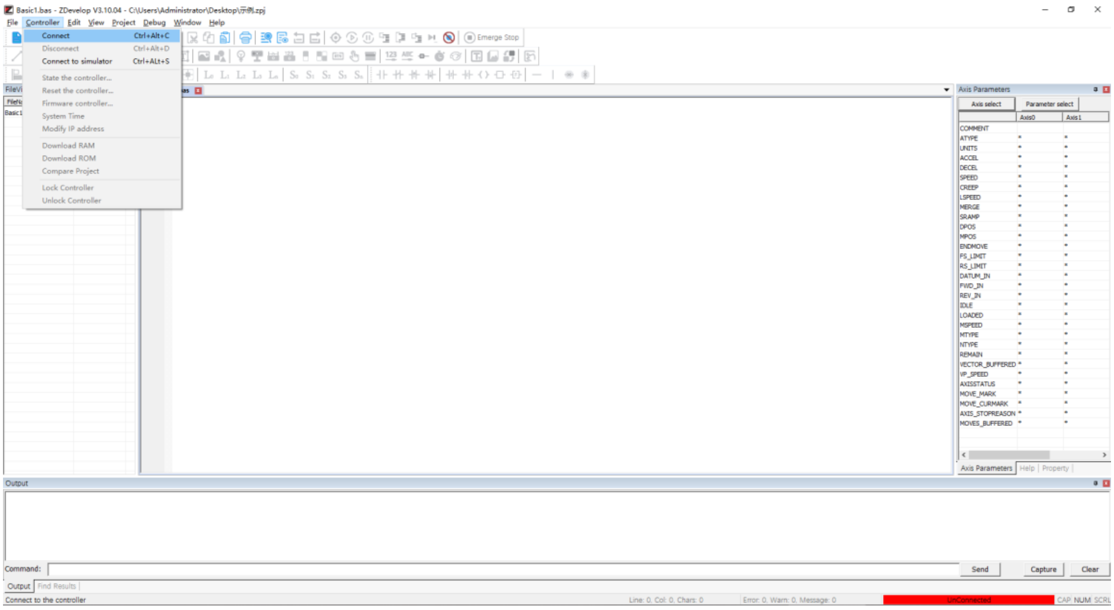
4. Set file as automatic operation: please see the below picture, double click the right position “AutoRun” of “File”, and input task number is “0”.



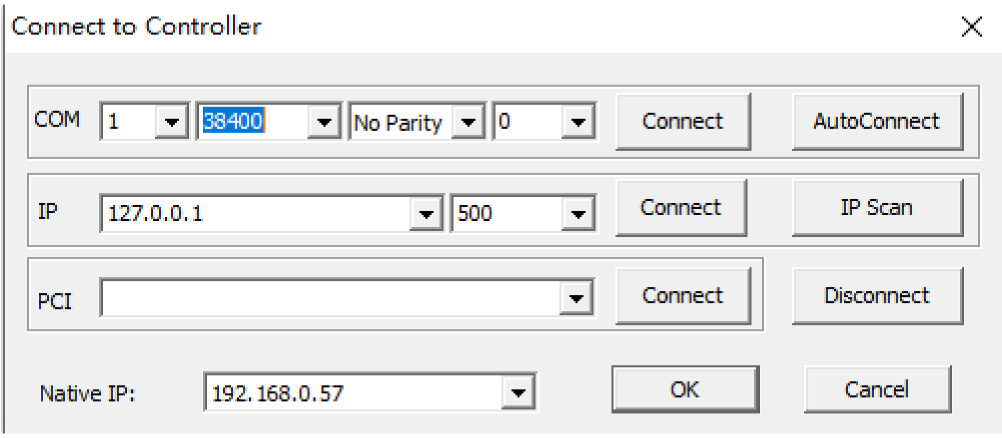
5. Program the procedure: when procedure is programmed, click “save” the file. New built Basic file will be saved automatically into the file in Project zpj.

6. Connect to Controller: program the procedure well in the input window, click “Controller” - “Connect”.

If there is no “Controller”, select connect to simulation, click “Connect” – “Connect to Simulator”. In this way, it can be connected to simulator, and there is hint showing simulator is connected successfully.



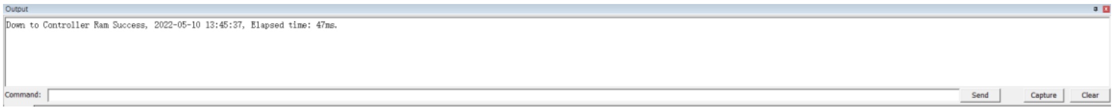
Click “Connect”, then “Connect to Controller” window will jump. And select serial port parameters or net port IP address, click “Connect”. When it is connected well, print information in Command and Output window: Connect to Controller: ZMC432 Version: 4.64-20170623.



For the detailed method of serial port connection and network port connection, please refer to the "Help" → "ZDevelop Help" document in the menu bar of ZDevelop software.

7. Download Program: click “Download RAM” or “Download ROM”. When it is downloaded successfully, Command and Output window will give a hint. Program is downloaded into controller and will run automatically.

✓ Download RAM:

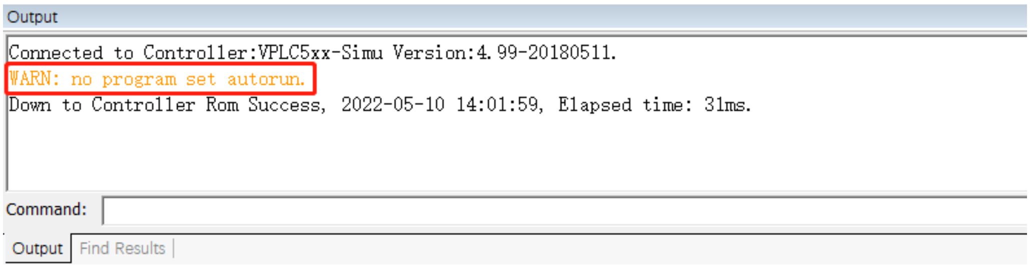


✓ Download ROM:

The program will not be saved after the RAM download is powered off, but the program will be saved after the ROM download is powered off. After the program downloaded to the ROM is connected to the controller next time, the program will automatically run according to the task number.

Precautions:

- When open the project item, select the item zpj file. If only the Bas file is opened, program can’t be downloaded into controller.
- ZMC00x series controller don’t support Download RAM.
- When project is not built, only Bas file can’t be downloaded into controller.
- AutoRun 0 means the task number, task number 0 runs the procedure. Task number doesn’t have priority.
- If all files of whole project are not set the task number, when downloading into controller, system will give the indication: WARN: no program set autorun.



2.3 Online Command and Output

The online command and output window can see and output various parameters of the controller, print program running results and program error information. The print output function given by the software developer in the program (output by commands such as, ?, PRINT, WARN, ERROR, TRACE, etc.).

Note: English symbols are used for question marks, and Chinese symbols are invalid. ERRSWITCH is the control switch of TRACE, WARN, and ERROR commands. Different parameter values correspond to different output effects:

0: TRACK, WARN, ERROR instructions all don’t output.

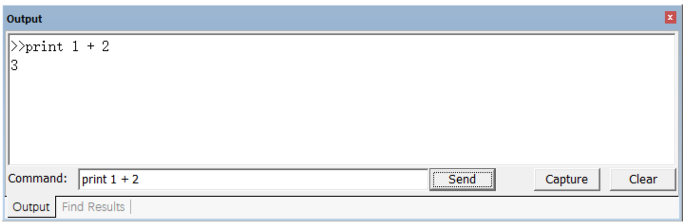
1: only output ERROR instruction.

2: output WARN, ERROR instructions.

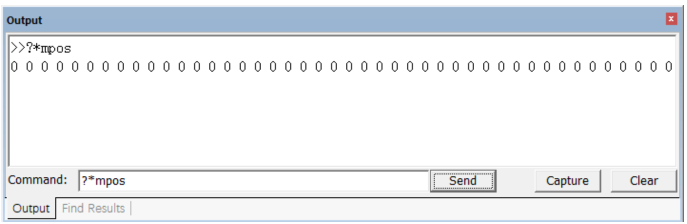
3: TRACE, WARN, ERROR instructions all output.

The online command and output window is shown below, “>>” represents the command input by ZDevelop online command, and the online command input “print 1+2” window will print the calculation result.

This function is valid when connecting to controller or simulator, it is not limited by program running status.



Use online command to see status of all axes, please see the below picture. Input “?*mpo”, window will print measurement positions of several axes mpos.



Common print and check commands:

?*SET: print all parameters' values

?*TASK: print task information

Normal	Only print task status
Error	Print task status, error task number, error line

?*MAX: print all specifications and parameters

?*FILE: print program file information

?*SETCOM: print the present serial port configuration information

?*BASE: print the present task BASE list

?*数组名: print all elements of array, the array length can't be so long.

?*参数名: print single parameter of all axes

?*ETHERCAT: print EtherCAT bus connection setting status

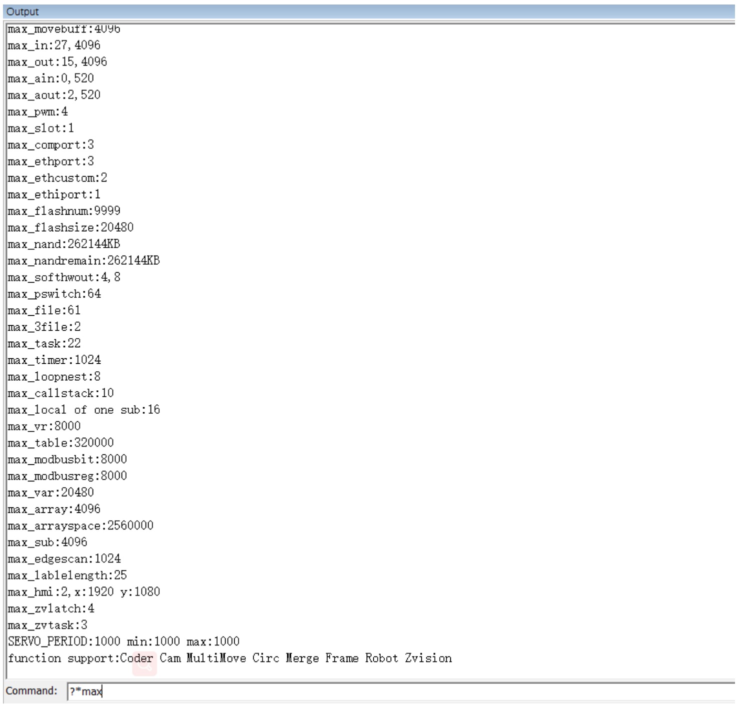
?*RTEX: print Rtex bus connection setting status

?*FRAME: print robot parameter, which needs 161022 or above firmware.

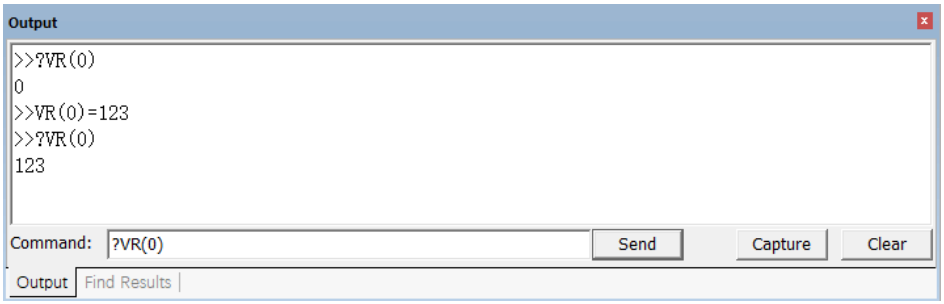
?*SLOT: print slot information of controller (RTEX, EtherCAT)

?*PORT: print all PORT communication ports

After connection to controller, use ?*max to print all specifications and parameters results of controller:



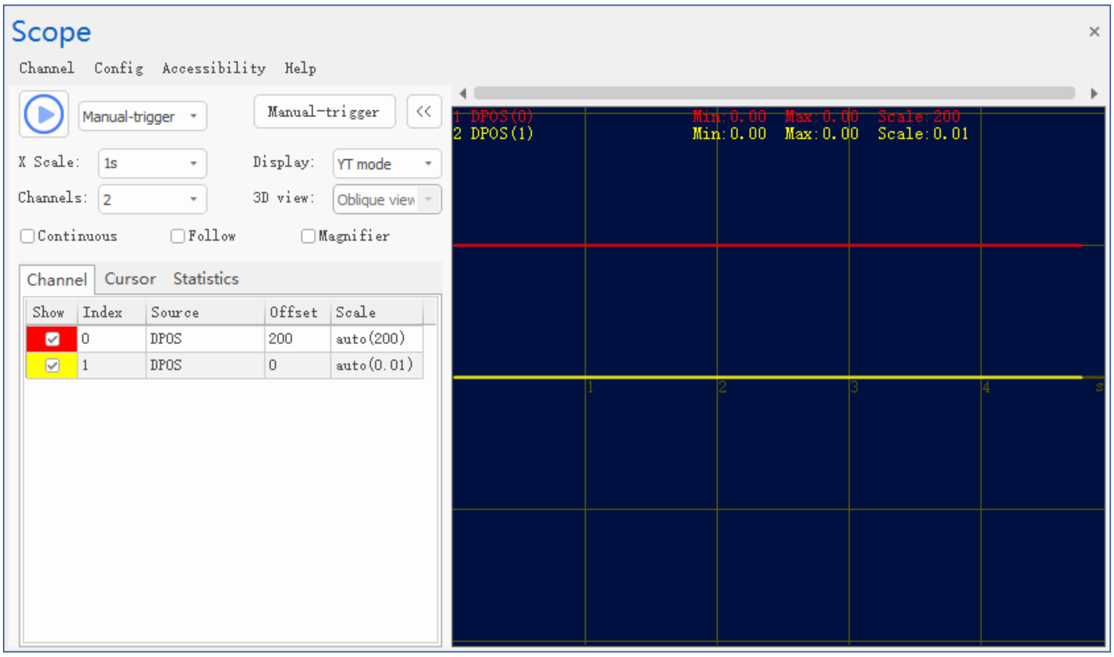
Modify the value of a variable. The setting and modification of VR variables, TABLE variables, MODBUS variables, global variables, system settings, axis parameters, and axis state variables can be realized through "Online Commands". The following figure is an example of modifying the VR variable value.



2.4 How to Use Oscilloscope

2.4.1 Scope Interface

Oscilloscope is extremely important of program debugging and running. It is used to transfer signals that can't be seen by naked eyes into graphics, so it is convenient to analyze change processes of all kinds of signals. Oscilloscope shows controller internal data in graph, it can display different signals, like, axis parameter, axis status, etc., click "Tool" – "Scope" to open the scope window.



Please see above SCOPE main interface, horizontal line means time, its time unit depends on horizontal scale, controller period, and space size, the corresponding calculation formular is "unit time = horizontal scale * space * controller period" (unit time: the time of each horizontal span), for example, if the horizontal scale is 1000, the space is 2, and period is 1000us, then each grid's time will be 2000ms. For vertical line, unit depends on specific selected data source.

--How to Operate--

After editing the program in RTSys, and connecting to controller / simulator, then open the scope, now you can set needed data source and corresponding No., select auto-trigger / manual trigger, next, click "" open button, and download the program into RAM/ROM again. At this time, if you use auto-trigger, it will sample after clicking ON, if you use manual trigger, after clicking ON, you need to click "Manual-trigger" to sample, then download to RAM/ROM, or download directly after clicking ON, then waiting for Basic to trigger (note, when waiting Basic trigger, "TRIGGER" command should be added in program).

--Scope Basic Buttons--

Buttons	Functions
Channel	Selected channel and superposition channel, comparison channel isn't shown.
Config	Open oscilloscope configuration window, set parameters.
Accessibility	Assist in observing waveforms, including searching waveforms, comparing waveforms, and importing and exporting waveforms.
Help	Display the mouse operation guide interface to prompt the mouse shortcut operations in each mode.
	Switch of oscilloscope. ON state, it is , but it will not trigger the oscilloscope.
Trigger Mode	In the drop-down menu, you can select auto-trigger or manual-trigger. When auto-trigger is selected, the manual-trigger button is unavailable. ● Auto-trigger: it will be triggered immediately after clicking the ON button. ● Manual-trigger: it is necessary to download to RAM/ROM after clicking ON button, then click the "Manual-trigger" button, or download directly to RAM/ROM after clicking "ON" button and wait for the Basic program to trigger (Note: when waiting for the Basic program to trigger, the "TRIGGER" instruction must be added to the program).
Manual-trigger	Trigger manually oscilloscope to sample.
<<	Press to hide the channel name and peak value, and display only the channel No.
X Scale	The scale of the horizontal axis. Select from the drop-down menu to manually enter the value and unit. The default input unit is ms, which is automatically converted to s after input. Place the mouse in the value box and scroll the mouse to zoom in and out of the horizontal scale. It is effective in YT mode, but becomes sensitivity in XYZ mode and XYZD mode, indicating the sensitivity of the left mouse button operation.
Display	There are four modes to switch, including YT mode, XY mode, XYZ mode and XYZD mode. When the number of channels is less than 2, the XY/XYZ/XYZD mode is not available, when the number of channels is less than 3, the XYZ/XYZD mode is not available, when the number of

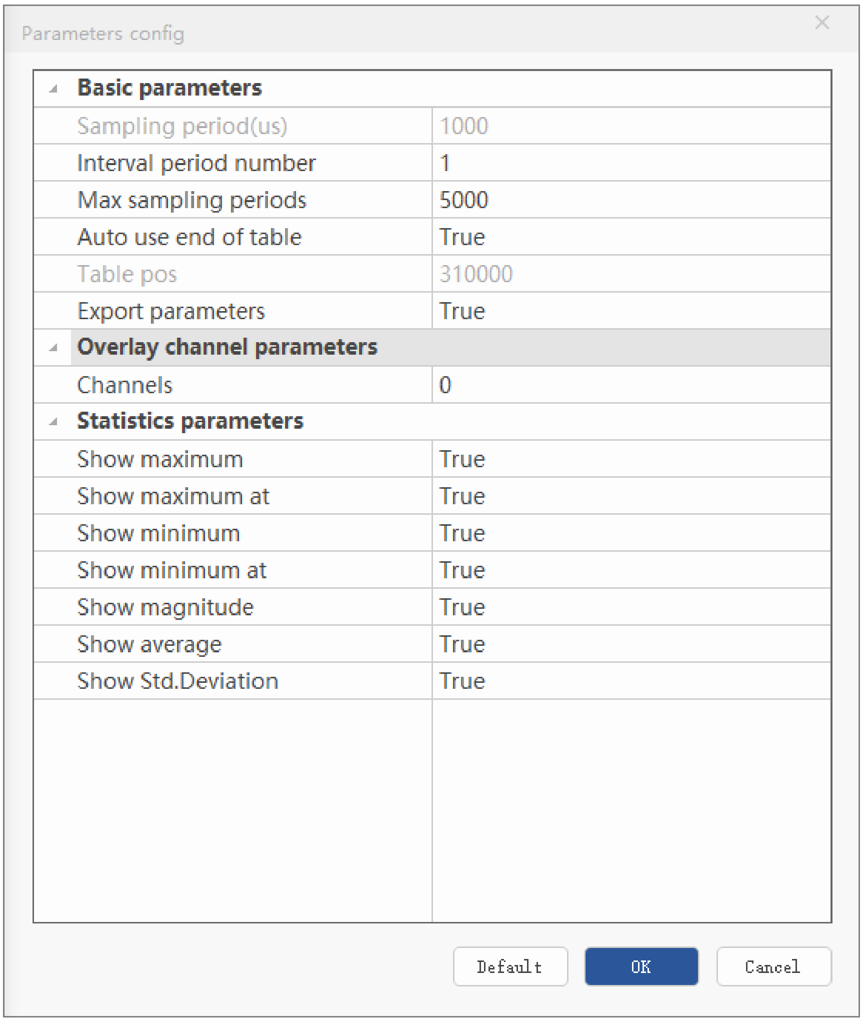
	channels is less than 4, the XYZD mode is not available.
YT Mode	The curves of different data sources changing over time, with each channel showing a waveform.
XY Mode	The XY plane displays the interpolated synthetic trajectory of the two axes, and two consecutive channels of the same type are grouped together to display a waveform.
XYZ Mode	XYZ 3D space displays the synthetic trajectory. Select the channel as the X, Y, and Z axis in turn. Three channels of the same type are grouped together to display a waveform (channel types include regular channel, overlay channel, contrast regular channel, and contrast overlay channel). Each type can display at most one waveform. Note: When using this mode, the OpenGL version of the display card must be 1.5 or above.
XYZD Mode	XYZD four-channel visualization display trajectory, where XYZ is the 3D space synthetic trajectory display, and D is the data source displayed in the form of dots. The calculation method is: dot diameter size = current D value ÷ D reference value × D reference size. Parameter modification is located in the "Observer Config" window. Select channels as X, Y, Z axis and D value channels in turn. Four channels of the same type are grouped to display a waveform (channel types include: regular channel, overlay channel, contrast regular channel and contrast overlay channel), and each type can display at most one waveform. Current D value: the size of the data source value at the current position. Note: When using this mode, the OpenGL version of the display card must be 1.5 or above.
Channels	Set the total number of regular channels to be sampled. It cannot be modified when ON. When the set number of channels is greater than the number of channels supported by the controller, a prompt message will pop up: Exceeding the maximum number of channels supported by the controller.
3D View	You can choose oblique angle, front angle, left angle and top angle. The default is oblique angle. XYZ mode and XYZD mode are valid.
Continuous	When continuous acquisition is not enabled, sampling stops after reaching the maximum acquisition cycle number, when continuous acquisition is enabled, the oscilloscope will continue sampling, and will continue sampling after reaching the maximum acquisition cycle number, that is, it will not stop sampling until the stop button is pressed. The acquired data will automatically overwrite the previous data. what’s more, all waveform sampling data acquired continuously can be exported (the continuous acquisition function is automatically canceled when using the serial port).
Follow	After turning on the follow, the horizontal axis automatically moves to the real-time sampling position and follows the waveform display.
Magnifier	When this is checked, and the magnified view will be automatically displayed at the lower right of the mouse when the mouse moves to the

	display area. The magnified view will follow the mouse movement and refresh. The magnifying glass parameters can be modified in the "Observer Config" window. YT mode is valid.
Show	Select whether to display the current channel curve. The oscilloscope has four types of channels, including regular channels 1 to 8, superimposed channels 1 to 4, regular channels 1 to 8 for comparison waveforms, and superimposed channels 1 to 4 for comparison waveforms.
Index	Select the data source No. to be collected, such as axis No., digital IO No., analog IO No., TABLE No., VR No., MODBUS No., etc. The number setting range is from 0 to the maximum number of axes of the controller, and the number can be entered manually.
Source	Select the data type to be collected. Click the left mouse button to manually enter the data type, or click the drop-down menu to select the type parameter. You can set the required parameter type in the "Data Source Design" window.
Offset	To set the waveform vertical axis offset, select the offset from the drop-down menu or enter it manually.
Scale	The scale of one grid on the vertical axis. When auto is selected, it indicates automatic scale, which is available when the oscilloscope is stopped. The scale value changes automatically according to currently acquired waveform, so that the waveform can be fully displayed on the current oscilloscope interface.
↑	It indicates loss may occur here, which is related to the maximum acquisition cycle number. After the oscilloscope starts continuous acquisition, it will re-trigger the acquisition at 80% of the maximum acquisition cycle. At this time, the TABLE data begins to be rewritten, and point loss may occur during this process. The "TRIGGER" command is effective in manual trigger mode, and it appears at about 80% of the maximum acquisition cycle number.
Note: to set the oscilloscope parameters, such as axis No., data source, and oscilloscope "Parameter Config" window, you must stop the oscilloscope first and then set them.	

2.4.2. How to Configure Scope

(1) Scope Config Window

Click menu above “Config” button, then click “parameter configuration”.

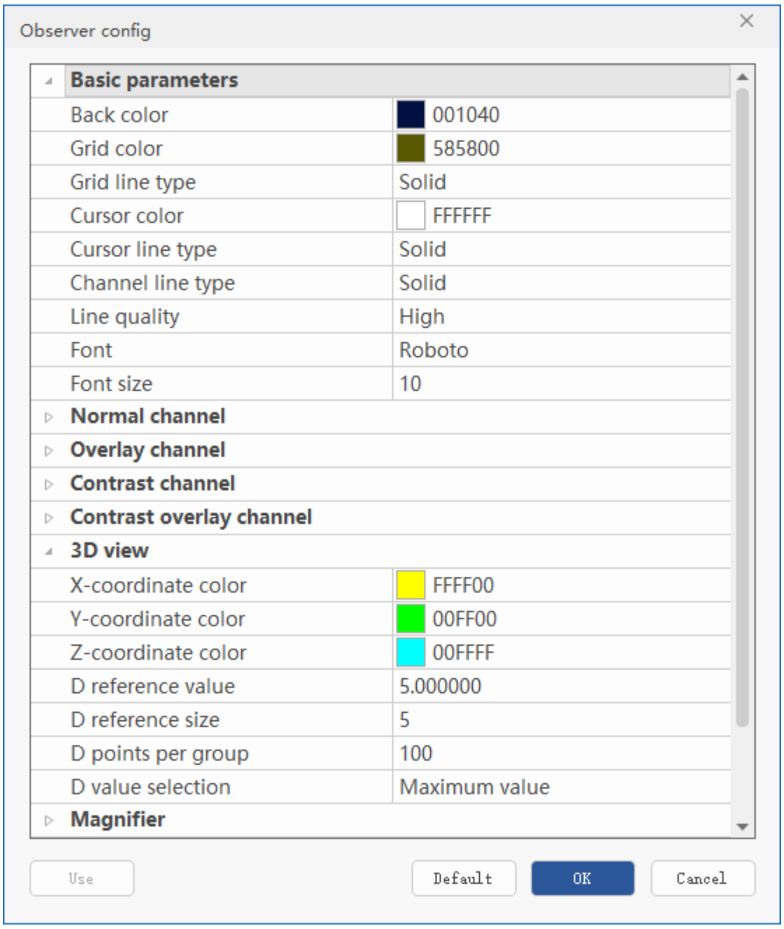


Parameter	Description
Basic parameters	
Sampling period (us)	Time interval between twice sampling by SCOPE, it can't be modified.
Interval period number	The sampling time interval, the unit is system cycles, which is related to the controller firmware version. The default value is 1ms. You can view it by SERVO_PERIOD. (For example, if the interval cycle number is set to 1, it means sampling once in 1 cycle. If the interval cycle number is set to 5, it means sampling once in 5 cycles, the cycle time depends on the controller firmware version.) Generally, the smaller the interval cycle, the more accurate the sampling data, and the larger the data volume per unit time.
Max sampling periods	The total number of sampled data. The larger the value, the larger the sampling range. (That is, the size of the table required for the data collected by one channel)
Auto use end of table	The position where saves the data, default is True.
Table pos	Set the location where the captured data is stored. Generally, the default is to automatically use the space at the end of the TABLE data. When "Auto use end of TABLE array" is set to False, you can

	customize the setting, but be careful not to overlap with the TABLE data area used by the program. There are three ways to check the size of the controller TABLE space: a. use the TSIZE instruction to read. b. view in the "Controller Status" window. c. print and view the online command? *max.
Export parameters	Select when you need to export oscilloscope channel parameter information. After checking, oscilloscope parameters are exported when exporting waveforms, including: basic parameters, overlay parameters, and channel configuration parameters (No., data source, offset, vertical scale). The default is True.
Overlay channel parameters	
Channels	Select how many channels that are overlayed, select from the drop-down menu.
Overlay channel 1 / 2	You can select the channel number for superimposition.
Overlay method	The overlay method between two channels, add or subtract.
Statistics parameters	
Statistics parameters	Set the parameter information displayed on the oscilloscope statistics page. The default value is True.

(2) Observer Configuration Window

Click menu above “Config” button, then click “obverse config”, then corresponding window will appear, after configured, click “use” to preview how it is after modified, then click “OK”.

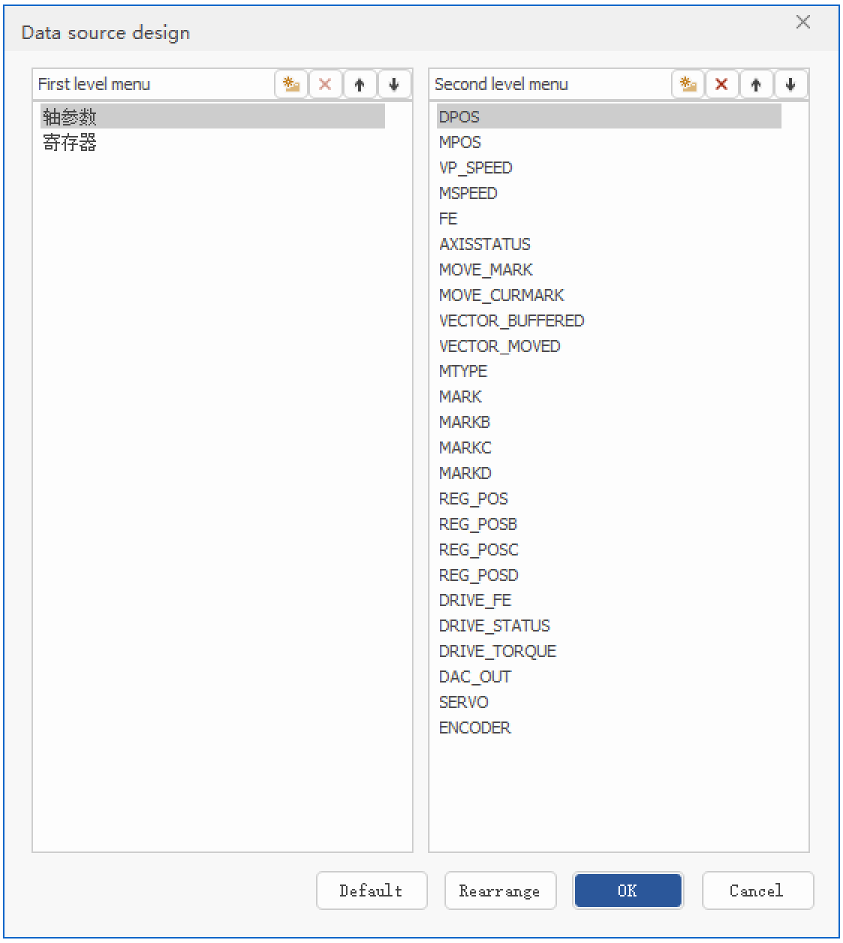


Parameter	Description
Back / Grid / Grid line / Cursor color	Set corresponding needed color.
Grid line type	Set the grid line type, there are solid or dashed lines.
Cursor line type	Set cursor line type, there are solid or dashed lines.
Channel line type	Set channel line type, there are point, solid, dashed lines. **For "point"**, scope will show data that are sampled by SCOPE in fixed period, "point size" parameter can be set. **For "solid / dashed lines"**, sampled points will become one smooth lines, then abnormal data can be easily checked, also, "line width" parameter can be set.
Line quality	Set channel waveform's line quality, when there are many data, recommend to use standard mode, which can accelerate scope performance.
Font / Font size	Set the font and font size of the channel No., channel name and peak value on the waveform display interface.
Normal / Overlay / Contrast / Contrast overlay channel	Set corresponding channel's line width, point size, and channel color.
D reference value / size	Used to calculate the dot diameter size in XYZD mode. The diameter size is related to the ratio of D reference size/D reference

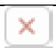
	value. The larger the ratio, the larger the dot diameter. The calculation formula is: Dot diameter size = current D value ÷ D reference value × D reference size. (The current D value is the value of "D value selection")
D points per group	Display a dot for every N sampling points. (For example, if "D points per group" is set to 100, a dot will be displayed for every 100 sampling points according to the value of "D value selection")
D value selection	The value of the current display dot size in N sampling points can be selected as the maximum value, minimum value and average value. (For example, if "D value selection" is set to the maximum value and "D points per group" is set to 100, the maximum value of every 100 sampling points will be used as the basis for calculating the current display dot diameter)
Magnifier	Set the width, height and magnification of the magnifier.
Search	Set the line width, point size, and channel color of the search results displayed when searching a waveform.

(3) Data Source Design Window

Click menu above "Config" button, then click "data source design".



Parameter	Description

First / Second level menu	Set corresponding needed color. When there is information in second level menu, the first level menu text is the type, the second level content is data source. When there is no information in second level menu, the first level menu is data source.
	“add” button, add information in first level or second level.
	“delete” button, deleted selected information. Note: axis parameter and register in first level can’t be modified.
	Up / down, used to sort.
Rerrange	Sort items of first level and second level according to characters from A to Z.

2.4.3. How to Import & Export Scope Data

a. Import Configuration

Import parameters related to scope, including parameter configuration, observer configuration, data source design, channel parameter configuration (show, No., data source, offset, vertical scale). And the file format of the imported data is .ini.

You only need to click “config” – “import config”, then select which file, when imported, new file data will cover before parameters.

b. Export Configuration

Export parameters related to scope, including parameter configuration, observer configuration, data source design, channel parameter configuration (show, No., data source, offset, vertical scale). And the file format of the imported data is .ini.

You only need to click “config” – “export config”, then select folder to save it.

2.4.4. How to Sample by SCOPE

A. Open project, connect to controller or simulator, then open the oscilloscope window (note: first, connect to controller or simulator, then operate the oscilloscope window).

B. Click “Scope Config” in oscilloscope window, select sampling period, max sampling period, sampling space, whether use END table, table position and show type, etc. Then, click “OK” for saving this configuration.

C. Select sampling Index and Source, then select auto-trigger or manual-trigger, click  button.

D. Download program into controller. When it is auto-trigger, sampling immediately after clicking button. When it is manual-trigger, click button first, then click “manual-trigger”, at last, download RAM/ROM, or if there is “TRIGGER” command in the program, you can click and download directly to wait for BASIC to trigger sampling.

E. If the waveform accuracy is not high or the display is incomplete, click the "" button and then open the "Scope Config", adjusting the sampling space and sampling depth, and perform the above sampling process again.

If the sampling time is long, start “Continuous acquisition” function. At this time, no relation between sampling time and max sampling period.

2.4.5. Scope Needs

- How to Calculate Scope Sampling Time:

For example, max period: 1000, space: 5

If system cycle SERVO_PERIOD=1000, which means it is 1ms trajectory planning cycle. Space 5 means sampling one data point per 5ms. Total sampling data number is 10000, so sampling time length is 50s.

- How to Calculate TABLE End Space:

Set the position where the captured data is stored. Generally, the space at the end of the TABLE data is automatically used by default, now starting space address is calculated automatically according to captured data space.

Calculation method: captured data space = channel numbers * max sampling periods

For example, if TABLE space of controller is 320000, there are 4 sampling channels, max sampling periods is 30000, each sampling point occupies one TABLE, so it will occupy 4*30000=120000 TABLE positions. 320000-120000=200000, which means starting position of TABLE is 200000.

If you don’t use TABLE end space, you also can self-define. Same condition as above, starting TABLE position can’t be more than 200000, because this space can’t be same as TABLE spaced used in program, otherwise, no way to run.

- How to Solve “Point Loss” Problem:

Generally, the “max sampling periods” is too low, “point loss” may appear. Then, you can set a bigger value.

- How to Solve “Polyline” under “Continuous Acquisition”:

Related to “max sampling periods”. Actually, the problem is “point loss”.

- How to Use “Continuous Acquisition” Function:

When continuous acquisition is not selected, the oscilloscope automatically stops sampling when the sampling depth is reached.

First select “Continuous acquisition” in “Scope Config”, then start oscilloscope, it will continue to sampling after triggered, and sampling even if it reached the depth. It will stop until press “Stop” button manually.

All waveforms and captured data from continuous acquisition can be exported.

2.4.6. Scope Usage Routine

Example 1: Continuous trajectory look-ahead application

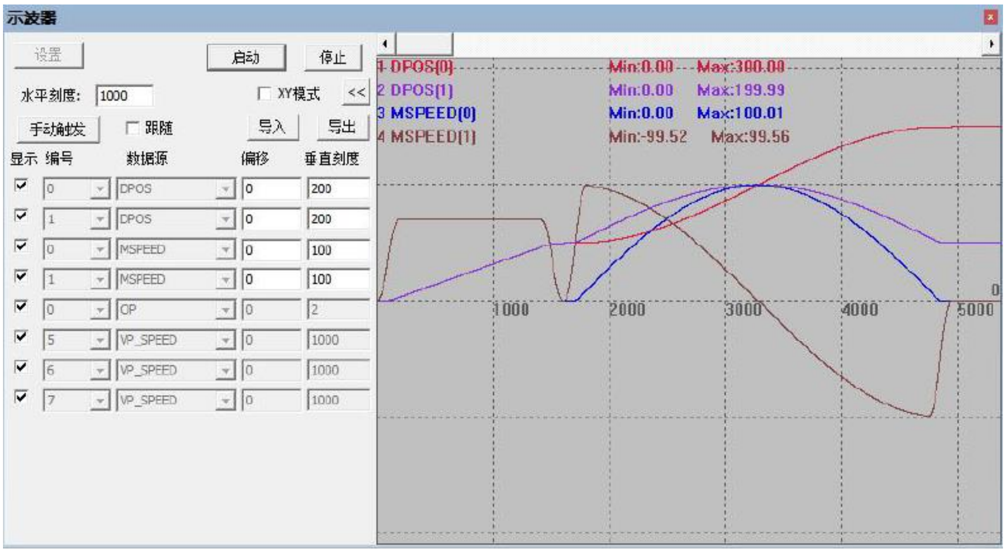
```
RAPIDSTOP(2)
WAIT IDLE(0)
WAIT IDLE(1)

BASE(0,1)
DPOS=0,0
ATYPE=1,1
UNITS=100,100
SPEED=100,100
ACCEL=1000,1000
DECEL=1000,1000
SRAMP=100,100
MERGE=ON
CORNER_MODE=2                  'start corner deceleration
DECEL_ANGLE = 15 * (PI/180)    'set angle of starting deceleration
STOP_ANGLE = 45 * (PI/180)     'set angle of ending deceleration
FORCE_SPEED=100                'it is valid when in equal deceleration
```

TRIGGER 'trigger oscilloscope automatically

MOVE(100,100)

MOVECIRC(200,0,100,0,1) 'Radius 100 draw a semi-circle clockwise, end coordinates (300,100)

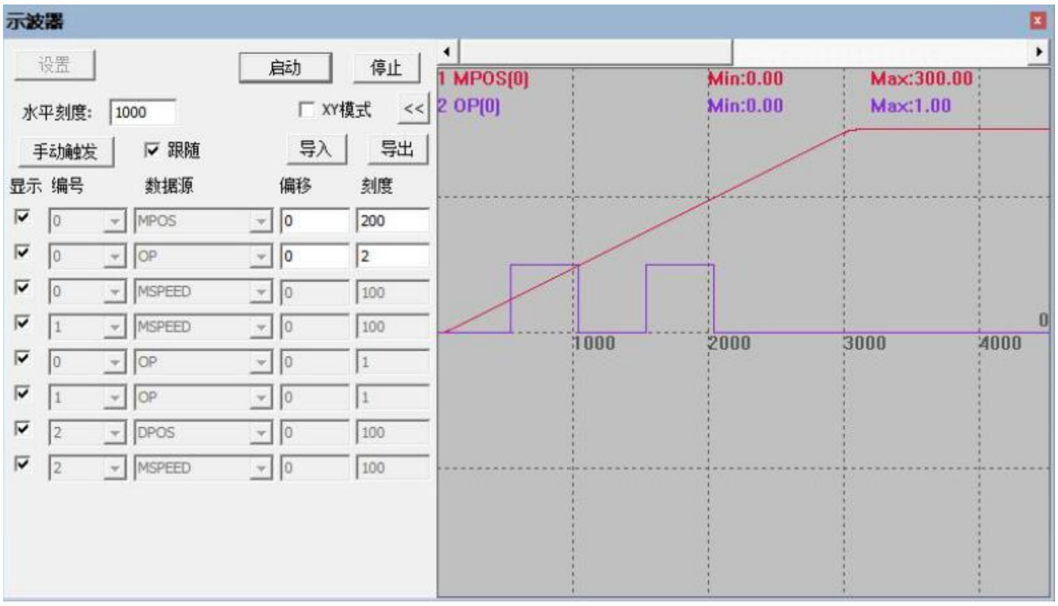
Speed and position curve of sampling axis 0 and axis 1:



Two-axis interpolation synthetic trajectory in XY mode:

Example 2: PSO position synchronization output, output OP signal when arriving comparison point

```
RAPIDSTOP(2)
WAIT IDLE(0)

BASE(0)

DPOS=0
MPOS=0
ATYPE=1
UNITS=100
SPEED=100
ACCEL=1000
DECEL=1000
OP(0,OFF)
TABLE(0,50,100,150,200)            'coordinate of comparison point
HW_PSWITCH2(2)                     'stop and delete incomplete comparison points
HW_PSWITCH2(1, 0, 1, 0, 3,1)       'compare 4 points, operate output 0
TRIGGER                            'trigger oscilloscope automatically
MOVE(300)
```

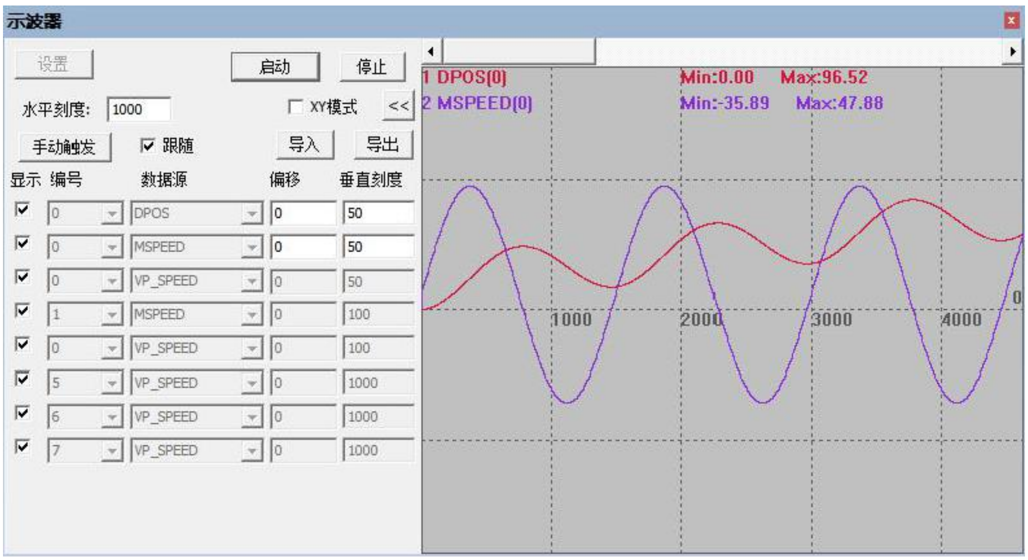


Example 3: Electronic Cam Application

```
RAPIDSTOP(2)
WAIT IDLE(0)

BASE(0)                    'select axis 0
ATYPE=1                    'pulse directional step or servo
```

```
DPOS = 0
UNITS = 100                     ′pulse equivalent
SPEED = 200
ACCEL = 2000
DECEL = 2000

′Calculate TABLE data
DIM deg, rad, x, stepdeg
stepdeg = 2      ′use this to modify line number, line is more, speed is more stable
FOR deg = 0 TO 360 STEP stepdeg
    rad = deg * 2 * PI/360                      ′convert to radian
    X = deg * 25 + 10000 * (1-COS (rad))        ′calculate offset of each small segment
    TABLE (deg/stepdeg, X)                      ′store TABEL
    TRACE deg/stepdeg, X
NEXT deg

TRIGGER                                 ′trigger oscilloscope sampling
WHILE 1                                 ′cycle motion
CAM (0, 360/stepdeg, 0.1, 300)          ′the virtual follow length is 300
WAIT UNTIL IDLE                         ′wait until motion stops
END
```

Motion trajectory: total time of each cam instruction = distance / speed = 300/200 = 1.5s



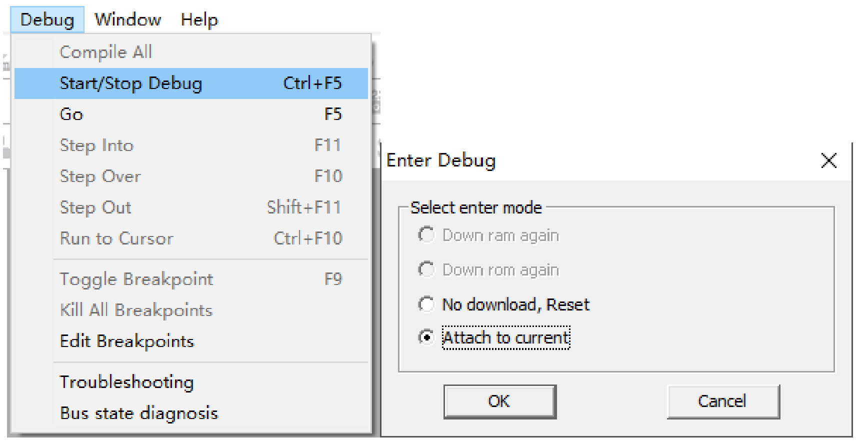
2.5 Program Debug

2.5.1 Enter Program Debug

Pay attention to safety when debug the machine! Be sure to design effective safety devices in the machine, and add the error handling procedures in software. Zmotion has no obligation or responsibility for the loss.

Debug function means it can debug the program rapidly, and check the running situation of all tasks in the program.

After ZDevelop connecting to controller, select “Debug” – “Start/Stop Debug”, then it will jump below window:

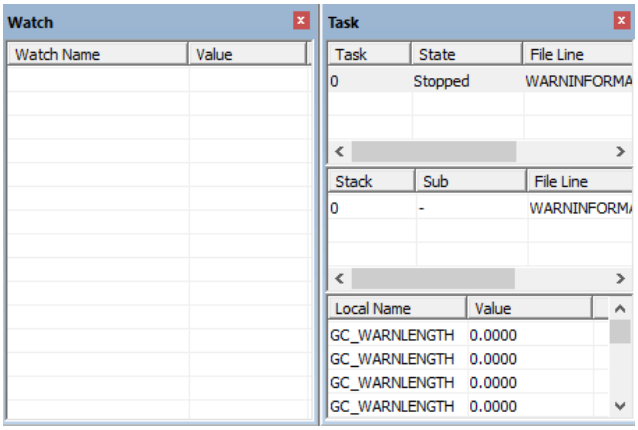


There are 4 kinds of debugging way when enters debug:

- Down ram again: it means the program is downloaded into RAM again, RAM fails to store when power-off.
- Down rom again: it means the program is downloaded into ROM again, RAM stores when power-off.
- No download, Reset: it means not to download the program, and run the program downloaded before, and open task window to see current running status.
- Attach to current: it means this time the program is not to be downloaded, only showing current running status when opened task window.

2.5.2 Task and Watch Windows

After selecting debug method, task and watch windows can be opened.

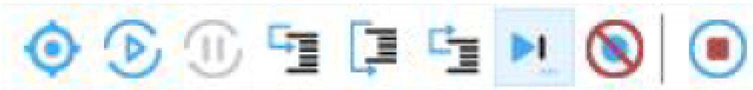


Task window is used to see running status of task, file and task running line number.

Valid expressions such as global variables and file module variables can be added into the “Watch” window. Local variables are not supported, and its parameter values are automatically obtained and displayed when the program is running. Also, under the debugging state, you can select variables in the program editing area and right-click "Add to Watch", or double-click the content name of the watch window to modify or add watch items.

2.5.3 Usage of Debug Tool Bar

After starting debugging, debug tool bar becomes valid.



From the left to the right:

- Reset: run from the starting position
- Run(F5): start to run automatically, pause the scan when encountering a breakpoint, and then click to resume the scan.
- Pause: pause the running
- Step Into(F11): run into program, press once, it will scan the next line
- Step Over(F10): run into next program
- Step Out: jump out of SUB subroutine to run
- Run to: run to the line specified by the cursor
- Toggle breakpoint: click to set, click to cancel again in the original position

➢ Emerge Stop: force stop all programs from running

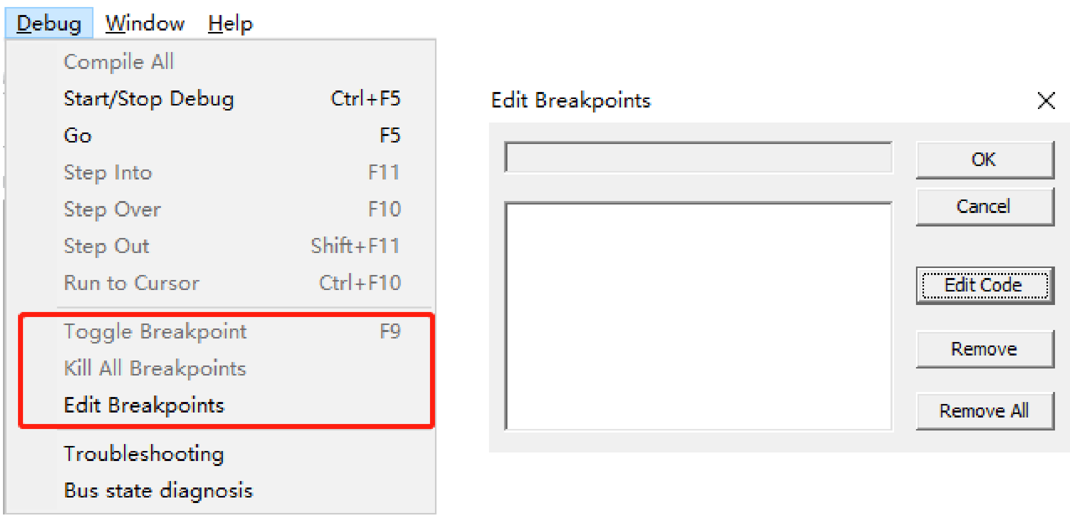
When the program is inconsistent with the controller or the program is not downloaded in time after re-modification, it will cause the line number specified by debugging to be offset. Motions that are currently submitted when paused are not paused.

2.5.4 Breakpoint Debug

Program can be obtained and paused through adding breakpoints

Breakpoint debugging can view the specific running process of program, which is mainly used to judge program logic errors. With watch content and axis parameter changes, you can view the impact of each step of the program execution on registers, variables, arrays, etc.

Breakpoint shortcut key F9 add, add or delete breakpoint button or menu bar "Debug" → "Toggle Breakpoint", multiple breakpoints can be added, menu bar "Debug" → "Kill All Breakpoint" is used to clear the project file at one time all breakpoints. “Edit Breakpoint” window can quickly remove the target breakpoint or navigate to the breakpoint to edit the code.



After the program stops at the breakpoint, you can perform step-by-step debugging, press the shortcut key F11, and press the program once to execute one step down.

As shown in the figure below, the debugging cursor stops at line 17. At this time, the statement on line 17 is not executed, and the statement on line 16 has been executed. Press F11 once to execute line 17.

If the breakpoint is set in the loop, the next time the loop runs to the breakpoint, the program will still be stopped.

After the program is debugged, all breakpoints should be cleared first, then download the program to the controller. Otherwise, print information prompting Warn file: "Basic1.BAS" line: 17 task: 0, Paused. The program after the breakpoint will not be scanned for the time being.

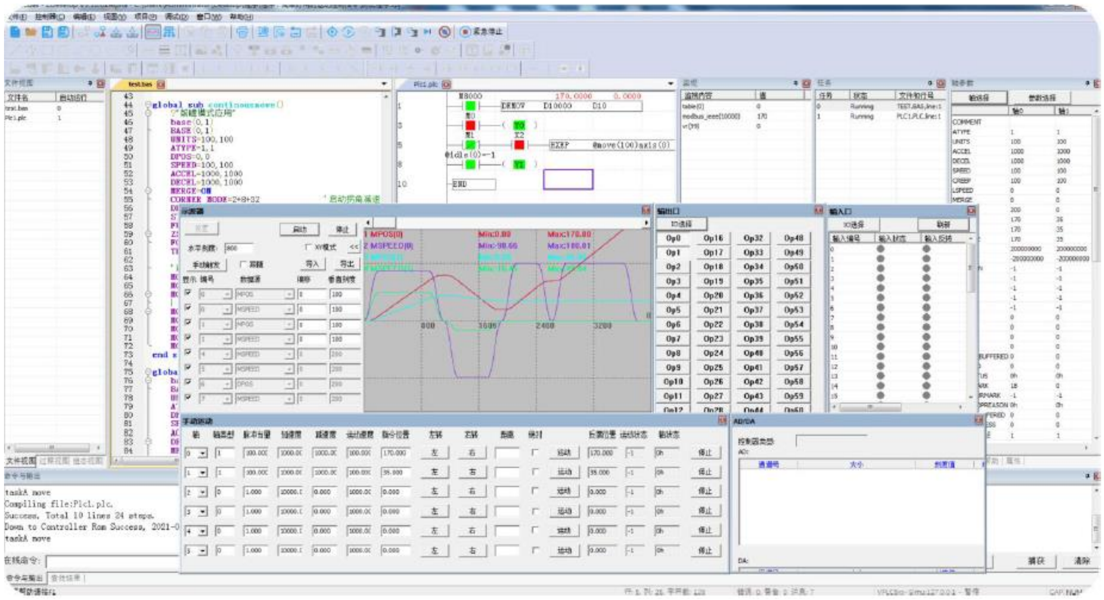
When the program is running, a warn warning appears, still it can continue to run. After the program is downloaded, it will stop running if it prints an ERROR error.

2.6 the View Window

ZDevelop software has a variety of view windows, users can easily edit and configure the controller program, develop applications quickly, monitor the axis running parameters in real time, and debug the running program of the motion controller in real time.

For example, the axis parameter window can monitor common parameters in motion control, and the readable and writable axis parameters can be directly modified after double-clicking in the window, the read-only parameters do not support modification. The input and output window monitors the status of IO, and the manual motion window quickly debugs the running status of the axis.

For more view windows and their function descriptions, please refer to the help menu of ZDevelop software, and open the "ZDevelop User Manual" to view.



Chapter III Basis of Basic Programming

This manual takes the Basic programming language as an example for detailed description. For customers who use PC host computer programming, please refer to Zmotion "Zmotion PC Function Library Programming Manual" for more information.

3.1 Programming Basic Knowledge

3.1.1 Program

Procedure consists of code sequence, telling computer how to execute a specific task. A program is a sequence of instructions (statements) developed by software developers according to user needs and described in a programming language that is suitable for computer execution.

ZBasic is not case sensitive, all punctuation marks of instructions in the program should be in English format.

Two aspects should be included in one procedure as follow:

1. To describe the data properly. In the procedure, the data type and organization form should be defined well, namely, the data structure. (For Reference: DIM, Global, Const)
2. To describe the operation procedure well. That is, the operation steps, or the algorithm, combined with the motion control is the process of motion and action.

■ Common Program Structure

To write an algorithm, we generally use the following program structure description methods: sequence, selection, loop, delay, wait, and sub-procedure calling. See the next section for sub-procedure calling.

◆ Sequence

In the absence of conditions and loops, the program always moves from top to bottom. When set to run automatically, the files are executed sequentially from the beginning of the file down by default.

Function 1

Function 2

Like above, execute function 1 firstly, then execute function 2.

Under BASIC programming, the program scans once from top to bottom.

Under PLC programming, the program scans periodically from top to bottom.

◆ Selection

Select different commands to execute according to execution conditions. There includes; IF THEN, ON GOTO, ON GOSUB, etc.

Routine 1:

```
DIM    aa
aa = 1
IF   aa = 0   THEN
       Command 1
ELSELF   aa = 1   THEN
       Command 2
ELSE
       Command 3
ENDIF
END
```

Routine 2:

```
DIM    a
a = 100
ON    a > 10   GOTO   label1
a = 1000
END           'main program ends

Lable1:
PRINT   a
END           'goto jump can't return
```

◆ Loop

Program is executed repeatedly, which means loop. There are main loop commands, FOR NEXT, WHILE WEND, REPERAT UNTIL, etc.

Routine 1:

```
DIM    a
```

```
a = 0
FOR i = 1 TO 10 STEP 1
    a = a + 1
  PRINT  a
NEXT
END
```

Routine 2:

```
DIM   a
a = 0
WHILE  IN(1) = OFF       'wait until input 1 is valid, exit loop
    a = a + 1
    PRINT  a
    DELAY (1000)
WEND
END
```

◆ Delay

When program encounters DELAY command, it will stop for a relative time, then continue to executing.

Routine:

```
PRINT  1
DELAY(2000)            'delay 2000ms
PRINT2               'print 1, after delaying 2000ms, print 2
END
```

◆ WAIT

When program encounters WAIT command, it will stop here, then execute until meeting WAIT conditions.

Routine:

```
BASE(0,1)
MOVE(100,100)
```

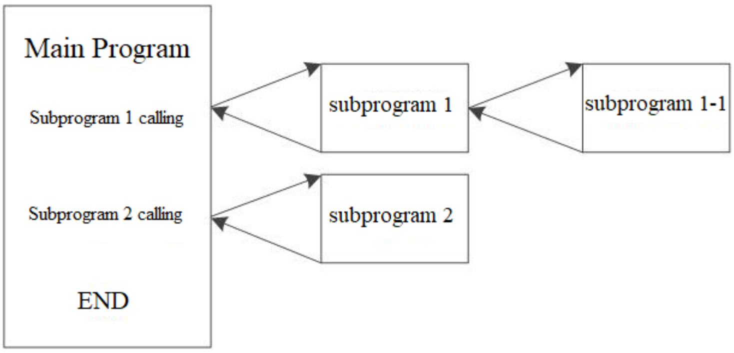
```
WAIT IDLE          'wait until the current interpolation motion ends
PRINT          'motion finishes
```

Except the WAIT and the DELAY commands, the program will block. When the motion instruction is scanned, if the motion buffer of the axis is full, the program will stop at the current motion command line until the current motion is completed. When the buffer has one black space, the program will continue to execute. See Motion Buffer Instructions for buffers.

■ Sub-procedure

Subprograms are often used in the programming process (subprograms are defined by the SUB instruction in Basic). Using subprograms can modularize programming. The relationship between each module is as simple as possible, and the functions are relatively independent, which is equivalent to simplifying the main program, so programming becomes more efficient and easier to read, and it can effectively decompose a complex program system design task into many subroutines and subtasks that are easy to control and process, which is convenient for development and maintenance.

→ Main program and subprogram execution logic:



The SUB subprogram can be opened as a subprogram, it returns to the main program after running END SUB. It can also be opened by using RUNTASK instruction to run independently as a task. After the task is opened, it has no relation with the main program. After the operation is completed, the subprogram task ends. Not return to the main program.

The main program calls subprograms nested up to 8 levels.

There are Global SUB, File Module SUB. Global SUB can be applied in all files, but File Module SUB only can be used in the current file. Subprogram also can pass parameters and returns parameters.

Example:

```
SUB sub1()              'define process SUB1, which is only used in the current file.
    ?1
    …
END SUB                 'self-define SUB process ends

GLOBAL SUBg_sub2()      'define global process g_sub2, which can be used in any file.
    ?2
    …
```

```
END SUB                    'self-define SUB process ends

GLOBAL SUBg_sub3(para1,para2)     'define global process g_sub3, passing 2 parameters
    ?para1, para2
    …
    RETURN para1 + para 2        'parameter return functions add
END SUB                    'self-define SUB process ends
```

3.1.2 Data

■ Data Definition

◆ Variable Definition

Variable is the parameter that can be self-defined by users. It is used to temporarily save communication data with external equipment or data that's processed by task inside. Namely, it saves data that is with property, like, name or data type, etc. There is no need to assign address allocation between variables and memory addresses.

Variable definition instruction: global variable (GLOBAL), file module variable (DIM), local variable (LOCAL).

Global variable (GLOBAL): it can be used in any file of project.

File module variable (DIM): it only can be used in file inside project.

Local variable (LOCAL): it is mainly used in the SUB, which means it is invalid in other files.

Variable can be assigned without definition, now variable is the DIM by default.

Example:

```
GLOBAL   g_var2              'define the global variable g_var2
DIM   VAR1                   'define file module variable VAR1

SUB   aaa()
     LOCAL   v1              'define local variable V1
     v1=100
END SUB
```

◆ Constant Definition

The value of a variable varies depending on the data that is substituted for that variable. The relative fixed value is a constant. Once the value of the constant is defined, it cannot be modified, which means it can only be read.

CONST defines a constant once time, and the definition and assignment must be the same line. Constant can be defined as global constant GLOBAL CONST. GLOBAL is used in any file, but there is no way to write LOCAL CONST. Constant is different from variable, it doesn't save the information in memories. There are many common constants, such as, Boolean type, Character String type, Time type, Date type, Integer type, etc.

Example:

CONST   MAX_VALUE = 100000   'define file constant

GLOBAL CONST   MAX_AXIS = 6   'define global constant

◆ Array Definition

Array assignment means that the data of the same attribute are collectively defined, and the number of data is designated. The pieces of data that make up the array are called "elements".

GLOBAL and DIM are relative instructions, but LOCAL definition is not supported.

Pay attention to array space designation, it can't be over definition range. Otherwise, program will appear error that indicates the array space limits.

Example:

DIM array(15)   'define file array, valid 15 arrays, number 0~14

GLOBAL array2(10)   'define global array, valid 10 arrays, number 0~9

?*max can check the max array size parameter "max_arrayspace", and it equals to the value that is gained by adding self-defined array and TABLE. However, the space except TABLE is real max space can be used by self-defined array, the max number of arrays to be self-defined is determined by max_array parameter.

■ Data Type

- ✧ Inside a computer, data is stored and operated in binary form, and a bit in binary data is the smallest unit in which a computer stores data.
- ✧ A binary bit can only represent two states of 0 or 1. To represent more information, it is necessary to combine multiple bits into a whole, generally 8-bit binary constitutes a basic unit byte (Byte).
- ✧ Byte is the most basic unit of computer data processing, and mainly interprets information in bytes. In general, one ASCII code occupies one byte, and one Chinese character international code occupies two bytes. Different computer models have different word lengths. Commonly used word lengths are 8, 16, 32 and 64 bits.
- ✧ Unit conversion: 1Byte=8bit, 1KB=1024B, 1MB=1024KB, 1GB=1024MB.
- ✧ Common bases are binary, octal, decimal, and hexadecimal. The parameters of various motion instructions are decimal data by default.

Name	Description
Bit	Bit is the most basic unit of binary value, its state is 0/1.
Nibble	It consists of 4 consecutive bits (such as bit3 ~ bit0), one bit represents decimal numbers 0 ~ 15 or hexadecimal 0 ~ F
Byte	It consists of 2 consecutive nibbles (8 bits, bit7 ~ bit0). Represent decimal numbers 0 ~ 255 or hexadecimal 00 ~ FF
Word	It consists of 2 consecutive bytes (16 bits, bit15 ~ bit0). Represent decimal numbers 0 ~ 65535 or 4 bits hexadecimal 0000 ~ FFFF
Double Word	It consists of 2 consecutive words (32 bits, bit31 ~ bit0). Represent decimal numbers 0 ~ 2^{32}-1 or 8 bits hexadecimal 00000000 ~ FFFFFFFF

- ✧ The data type refers to the specific provisions on the form and range of the value represented by the variable. When the variable is declared, the size of the data type is determined according to the size of the data range in the memory. The larger the data range in the memory, the larger the range of values that can be represented.

Data Type	Description
Boolean	Value is 0/1
Integer	Value is integer
Real number	Value is real number
Date	In date form, DD: MM: YYYY
Time	In time form, hh: mm: ss
Character	Value is character string

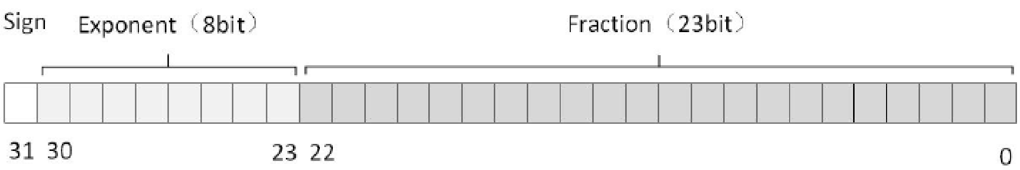
- ✧ The data types of variables input or output by instructions are determined by instruction.

✧ The data type of self-defined variable belongs to dynamic type. When integer is assigned to variable, the variable is integer type. When floating type is assigned to variable, the variable is floating point type.

✧ Self-defined array's data types are single-precision floating point and double-precision floating point. Please refer to below floating point introduction.

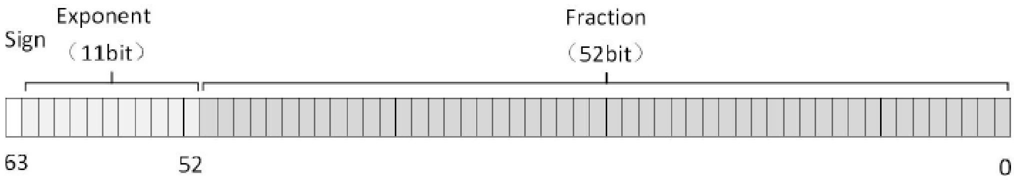
Single-precision floating point 32-bit:

Single-precision floating point data format: VR, MODBUS_IEEE, TABLE and self-defined array and variable (ZMC3XX series controller and former series)



Double-precision floating point 64-bit:

Double-precision floating point data format: TABLE and self-defined array and variable (ZMC4XX series controller and following series)



Common register data type form:

Register Type	Data Type	Value Range
MODBUS_BIT	Boolean	0 or 1
MODBUS_REG	16 bits integer	-32768 to 32767
MODBUS_LONG	32 bits integer	-2147483648 to 2147483647
VR_INT		
MODBUS_IEEE	32 bits floating	-3.4028235E+38 to -1.401298E-45
VR		
TABLE and array (ZMC3XX series and its former)		
TABLE and array (ZMC4XXseries and its after)	64 bits floating	1.7E-308 to 1.7E+308
VRSTRING	character	one character occupies one VR address
MODBUS_STRING	character	One character occupies 8 bits

✧ The memory capacity required for all data does not match the total data size (capacity value) of each data because the head position of the data allocated to the memory is automatically allocated to the multiple position of the "calibration value (boundary value)" for each whitespace occurs between data types. Even if the kinds of data types are the same, the overall occupied data size still varies depending on the order of the data types.

■ Data Operation

Pay attention the data type when operating data of different types. Below problems will appear if types are not matched:

◆ Data Loss

Decimal part will loss when data type is from floating to integer.

Routine:

```
VR(0)=10.314
MODBUS_REG(0)=0
MODBUS_REG(0)=VR(0)
?MODBUS_REG(0)                'the result is 10
```

◆ Force Conversion

After the integer type is stored in the floating-point type register, it will become a floating-point type, and then using the integer type to manipulate the data may be incorrect.

◆ Common Usage Problem

When obtaining the date, do not use single-precision floating-point storage, because the date format is 8-bit, and the single-precision floating-point number has only 6 valid bits. It is recommended to directly use the 32-bit integer MODBUS_LONG to store.

Some parameters must use string type constants or variables, various strings can be combined by "+", and the operation of a single byte of a string needs to be performed using an array.

Instructions related to character string:

Instruction	Description
DIM	Defined array can be used as character string directly, each element represents a byte.

“”	Use “” to define constant type character string directly.
CHR	Convert ASCII to a character string, it only occupies one byte.
MODBUS_STRING	Standard MODBUS protocol defines character string, each 16-bit register stores 2 bytes.
VRSTRING	VR list acts as character string, 1 VR stores 1 byte.
+	Operational character, which is used to combine two characters.
VAL	Convert Number character string to numerical.
TOSTR	Convert numerical to number character string.
STRCOMP	Compare different character strings
DMCPY	Array copy function, also can copy character string.
HEX	Return hexadecimal value, only for print purpose.
DATE$	Return date in “dd: mm: yyyy” format.
DAY$	Return the English name of today's week
TIME$	Return the current time of 24 hours type in “hh:mm:ss” format.

◆ Parameter

✧ There are axis parameters, task parameters, system parameters, etc. Parameters can be read or be written (except a little parameter only be read).

✧ Configure axis parameters (axis type, pulse equivalent, axis speed, etc.) well before motion. Relative safety configuration (positive and negative hardware/software position limitation, alarm signal, emergency stop, deceleration, etc.) should also be set well.

✧ There are two types, auto-save and nonauto-save.

→ For auto-save parameter, it will be saved after modification, and won’t recover the default value when powers on again. Relative instructions: axis parameter instruction, IP_ADDRESS, APP_PASS and LOCK these kinds of password instructions, CANIO_ADDRESS, etc.

→ For nonauto-save parameter, it will recover default value when it powers on again, which means it needs to be modified. For example, use SETCOM instruction to set serial port parameters, needing to set again after power-on each time, so SETCOM instruction should be put the beginning of program.

◆ Power Failure Storage

✧ The controller has protection on register VR and multiple sector storage FLASH blocks when power-down.

✧ Check FLASH sector amounts through ZDevelop online command “?FLASH_SECTES”. Command “?FLASH_SECTSIZE can see the size of sector, and can save power-down data.

- ✧ ONPOWEROFF when power-down interrupting, written program can be used to record the position of power-off to VR. When system powers on again, use program for recovering VR data into current position, because executing time is very short when power-down, it is recommended to only save several data.
- ✧ Use SETCOM instruction to match VR with MODBUS_REG registers, and set instruction parameter “variable”. Please see SETCOME instruction for details.

Routine:

Set variable = 3, and one VR_INT should be mapped into two MODBUS_REG addresses. Conversion Relation: VR_INT(num) =MODBUS_REG(num)*2^16+MODBUS_REG(num+1)

```
SETCOM(38400,8,1,0,0,0,3)        'configure as power failure storage
VR_INT(0)=0

MODBUS_REG(0)=1             'low 16-bit value is 1
MODBUS_REG(1)=2             'high 16-bit value is 2
?VR_INT(0)              'result: 131073
END
```

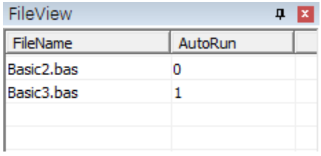
- ✧ VR is not easily to lose when power down, it can be read and written infinite times. The data storage time is 10 years. It is recommended to store the key parameters of the machine and equipment in FLASH. The FLASH space is larger. When the power is turned on, the data is read from the FLASH and written to each variable.
- ✧ FLASH has a write life limit and cannot be erased and written indefinitely. It is recommended to write to VR for frequently rewritten data.

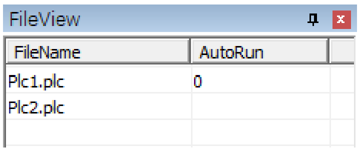
3.2 Three Programming Methods of Zdevelop

3.2.1 Hybrid Programming

- ✧ ZDevelop software supports 3 kinds of programming methods, they are ZBASIC, ZPLC ladder diagram and ZHMI configuration. It also supports these 3 languages hybrid programming. The programmed procedure through ZDevelop can be downloaded to ZMOTION motion controller.
- ✧ ZBSIC, ZPLC and ZHMI can run multi-task among them. ZBASIC can run multi-task, ZPLC and ZHMI both only can run one task.
- ✧ For example, see the below, two different BASIC files in one project can set different task

numbers to run separately. PLC/HMI file in the same project only can have one task number.





✧ Both ZPLC programming and ZBASIC programming are easy to understand and clear in logic structure, which can meet various programming requirements and are widely used at present. The HMI configuration programming is suitable for ZMOTION ZHD series teaching box, and teach pendant of other companies can also be connected to the controller, please use the teaching box programming software provided by the company.

3.2.2 PLC and BASIC Call Each Other

Relation of PLC and BASIC registers:

PLC		BASIC	
Input relay X	X0-X7	Input port IN MODBU_BIT (10000-10527)	IN(0)-IN(7)
	X10-X17		IN(8)-IN(15)
	X20-X27		IN(16)-IN(23)
	X1770-X1777		IN(1016)-IN(1023)
Output relay Y	Y0-Y7	Output port OP (20000-20527)	OP(0)-OP(7)
	Y10-Y17		OP(8)-OP(15)
	Y20-Y27		OP(16)-OP(23)
	Y1770-Y1777		OP(1016)-OP(1023)
Auxiliary relay M	M0	MODBUS_BIT (0-4095)	MODBUS_BIT(0)
	M1		MODBUS_BIT(1)
	M1023		MODBUS_BIT(1023)
Special relay D	D0	MODBUS_REG MODBUS_LONG MODBUS_IEEE	MODBUS_REG(0)
	D1		MODBUS_REG(1)
	D1023		MODBUS_REG(1023)
Floating register DT	DT0	TABLE (0-5999)	TABLE(0)
	DT1		TABLE(1)
	DT1023		TABLE(1023)
State register S	S0 ~ S999	MODBUS_BIT (30000-30999)	
Analog output register	D13000 ~ D13127	MODBUS_REG (13000-13127)	
Analog input register	D14000 ~ D14255	MODBUS_REG (14000-14255)	
PLC Command		EXE @BASIC Command	

- ✧ Input Relay X is related to IN, under PLC programming, X is octal system (X0~X7, X10~X17, …), but controller's input port IN is decimal system, so decimal conversion is needed when programming. For example, IN2 is relative to X24, IN8 is relative to X10.
- ✧ Output Relay Y is related to OP, under PLC programming, Y is octal system (Y0~Y7, Y10~Y17, …), but controller's output port OUT is decimal system, so decimal conversion is needed when programming. For example, OUT2 is relative to Y24, OUT8 is relative to Y10.
- ✧ Auxiliary Relay M is related to MODBUS_BIT.
- ✧ Special Relay D is related to MODBUS_REG.
- ✧ Floating Register DT is related to TABLE, which can be used to transfer data between ZBASIC.
- ✧ "EXE@BASIC instruction expression" in PLC can be used to call BASIC instructions.
- ✧ Basic can use command "RUN "xxx.plc", task number" to start PLC task.
- ✧ "CALL SUB_FUNC" or "RUNTASK_RUNC" can be used to call PLC subprogram LBL. Please see "ZMotion PLC Programming Manual" for more details.

3.3 Register

There are several main registers of controller, such as, TABLE, FLASH, VR, MODBUS, etc. After connecting ZDevelop software to controller, size of each register on this controller can be checked through ZDevelop software "Controller" – "State the controller". Also it can input "?*max" in online and output window to see the amount of each register. Different controllers have different store space.

3.3.1 Table

TABLE is a very large array that comes with the controller, the data type is 32-bit floating point (4 series and above are 64-bit floating point), and it will not be saved when power off. When writing a program, the TABLE array does not need to be defined again and can be used directly. The index subscript starts from 0.

Some instructions of ZBasic can directly read the values in TABLE as parameters, such as CAM, CAMBOX, CONNFRAME, CONNREFRAME, MOVE_TURNABS, B_SPLINE, CAN, CRC16, DTSMOOTH, PITCHSET, HW_PSWITCH, etc.

Parameters sampled by the oscilloscope are also stored in TABLE. Therefore, in the development and application, pay attention to the allocation and use of multiple TABLE areas, and do not overlap with the data storage area sampled by the oscilloscope.

1) TABLE instruction reads and writes data:

TABLE(0) = 10 ′TABLE(0) assigns 10

TABLE(10,100,200,300) ′Mass assignment, assign TABLE(10) as 100, assign TABLE(11) as 200, assign TABLE(12) as 300

2) TABLE size can be read by TSIZE instruction, and can be modified (can’t be over TABLE max space).

PRINT TSIZE ′print controller TABLE size

TSIZE = 10000 ′set TABLE size, which can’t be over max controller TABLE size

3) TABLESTRING instruction prints data in TABLE according to character string format.

TABEL(100,68,58,92)

PRINT TABLESTRING(100,3) ′print data in string form, then convert to ASCII code.

PTINT RESULT: D:\

When TABLE is used as parameter to pass, uses are basically same. Next take CAM as the example:

CAM(start point, end point, table multiplier, distance)

start point: the starting point TABLE number, where the first point is stored

end point: the end point TABLE number

table multiplier: the position is multiplied by this ratio, generally set to the pulse equivalent

distance: the distance of the reference movement

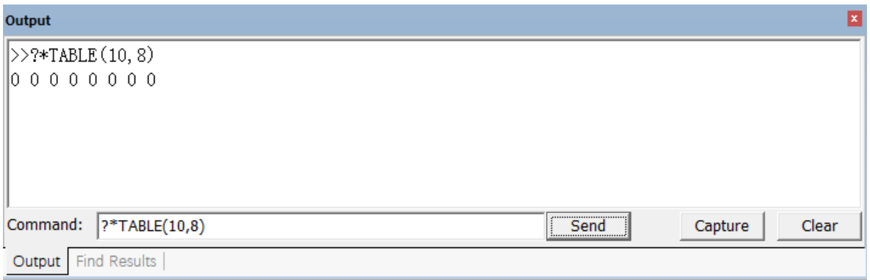
Example of usage:

TABLE(10,0,80,75,40,50,20,50,0) 'TABLE starts to store data from 10, assign TABLE (10) as 0, assign TABLE (11) as 80

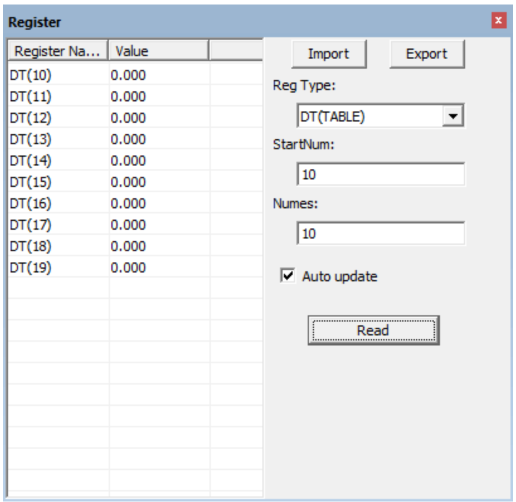
CAM(10,17,100,500) 'Motion track is from TABLE(10) to TABLE(17)

There are two ways to view the data in TABLE:

→ enter “?*TABLE(10,8)” on the online command, starting from TABLE(10), 8 data in turn.



→ Check the DT (TABLE) data in the register, starting from 10, and there are 8 numbers.



3.2.2 FLASH

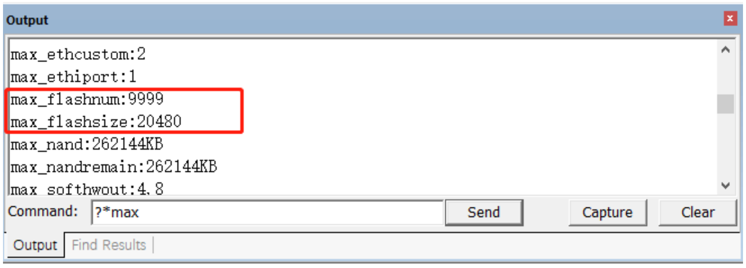
✧ Strictly speaking, FLASH is closely related to the register, but is is not a register, so it is described in this chapter.

✧ FLASH has a power-down storage function, and the number of reading and writing limit is 100,000 times, and data will not be lost if it is not powered on for a long time. It is generally used to store large data that does not require frequent reading and writing, such as processing files.

✧ When reading and writing, pay attention to ensure that the names and order of variables, arrays, etc. to be operated are highly consistent. If they are inconsistent, data will be cluttered.

✧ When FLASH is used, it is viewed according to the block number, and the number of blocks is checked through FLASH_SECTES instruction. The number of FLASH blocks and block data sizes of different controllers are different, and the data size of each block is checked through FLASH_SECTSIZE instruction.

✧ Also view it on the online command line, as shown below.



- ✧ Parameters set by CAN communication, IP address, APP_PASS, LOCK password and other system parameters are stored in FLASH.
- ✧ Note: FLASH must be written before reading, otherwise an alarm WARN will be prompted.
- ✧ How to use FLASH:

GLOBAL VAR 'variable definition

GLOBAL ARRAY1(200) 'array definition

DIM ARRAY2(100)

'data is stored in FLASH block: Write VAR, ARRAY1, ARRAY2 data into FLASH block 1 in turn

FLASH_WRITE 1, VAR, ARRAY1, ARRAY2

'FLASH block data read: read the data of FLASH block 1 into VAR, ARRAY1, ARRAY2 in sequence

FLASH_READ 1, VAR, ARRAY1, ARRAY2

'The reading order is consistent with the writing order

3.3.3 VR

- ✧ The VR register has a power-down storage function and can be read and written infinitely, but the data space is small, generally only 1024 or less. The VR space of the latest series of controllers is 8000, which is used to save data that needs to be modified continuously, such as axis parameters, coordinates, etc., the data type is 32-bit floating point (4 series and above are 64-bit floating point).
- ✧ Use VR_INT to force an integer, and VRSTRING to force a string. VR, VR_INT, VRSTRING share a space, and the address space is overlapping. VR and VR_INT have the same read and write methods. VRSTRING saves ASCII code, and one character occupies one VR.
- ✧ The principle of VR's power-off storage is that the controller has a power shortage memory inside, but the data capacity is small, so the data with a large amount of data or data that needs to be saved for a long time is best to be written into the FLASH block or exported to a U disk.

✧ The VR register can also be used for the RTEX controller to transmit reading and writing data, write the DRIVE_WRITE parameter, and read the DRIVE_READ parameter. For details, see Chapter 16 RTEX instruction.

✧ Use CLEAR instruction to clear all data in VR, CLEAR_BIT instruction will set a certain position of VR to 0, READ_BIT instruction will read a certain bit data of VR register, SET_BIT instruction will set a certain position of VR to 1.

Example 1: VR usage method

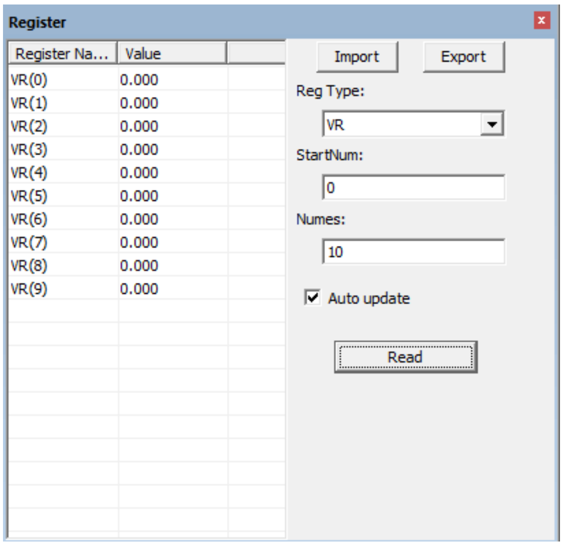
```
VR(0)                    'assign
aaa = VR(0)              'read
```

Example 2: data conversion in VR register

```
VR(100)=10.12
VR_INT(100) = VR(100)        'data conversion
?VR_INT(100)                 'print result: 10, from floating to integer type
```

Example 3: VRSTRING stores character string

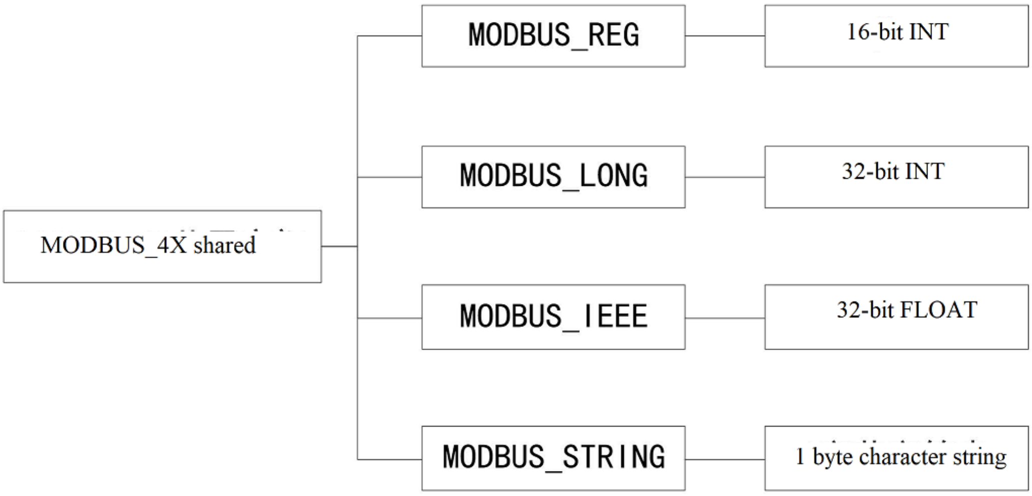
```
VRSTRING(0,4) = "abc"        'save character string, starting from VR(0)
PRINT VRSTRING(0,4)          'print result: abc
```



3.3.4 MODBUS

✧ MODBUS register conforms to MODBUS standard communication protocol, there are bit register and word register. MODBUS register doesn't support power failure storage.

✧ Bit register: MODBUS_BIT, for touch screen, it is called MODBUS_0X, Boolean type.
Word register: MODBUS_REG, MODBUS_LONG, MODBUS_IEEE, MODBUS_STRING. For touch screen, it is called MODBUS_4X, see the below:



- ✧ The MODBUS word register in the controller occupies the same variable space, one LONG occupies two REG addresses, and one IEEE also occupies two REG addresses. When using, pay attention to stagger the word register number address.

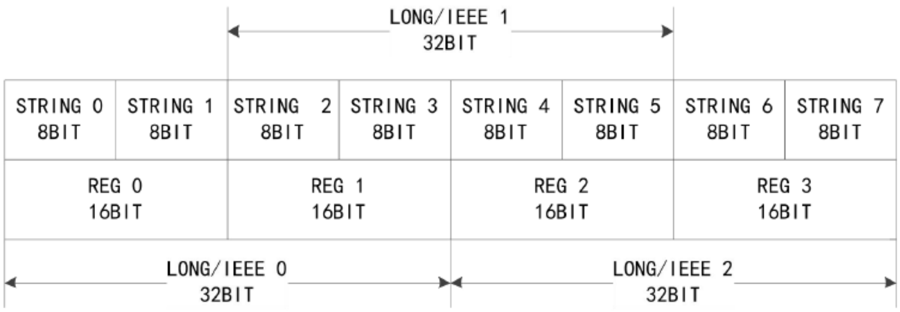
→ MODBUS_LONG(0) occupies two REG addresses, MODBUS_REG(0) and MODBUS_REG(1).

→ MODBUS_LONG(1) occupies two REG addresses, MODBUS_REG(1) and MODBUS_REG(2).

→MODBUS_IEEE(0) occupies two REG addresses, MODBUS_REG(0) and MODBUS_REG(1).

→MODBUS_IEEE(1) occupies two REG addresses, MODBUS_REG(1) and MODBUS_REG(2).

- ✧ So pay attention not to overlap MODBUS_REG, MODBUS_LONG, MODBUS_IEEE addresses in users application programs.
- ✧ Calculation method: MORBUS_REG(1) is high bit, MODBUS_REG(0) is low bit, MODBUS_LONG (0) = MODBUS_REG(1) * 2^16 + MODBUS_REG(0).
- ✧ 4X space diagram:



Routine:

MODBUS_REG(0)=0 'initialize as 0

MODBUS_REG(1)=0 'initialize as 0

MODBUS_LONG(0)=70000 'assign modbus_long as 70000, range of modbus_reg is

32768~32767

?MODBUS_REG(0),MODBUS_REG(1)

'print reg(0) is 4464, reg(1) is 1, long(0)=reg(1)*2^16+reg(0)

✧ In the process of serial port setting (SETCOM parameter), when the register is selected as VR, a VR is mapped to a MODBUS_REG at this time, where VR is a 32-bit floating point type, and MODBUS_REG is a 16-bit integer type with signs. The data transmitted from VR to MODBUS_REG will lose the fractional part. When VR data exceeds plus or minus 15 digits, the MODBUS_REG data will be changed. MODBUS_REG transmits data to VR without problems, see the following routines, and see the SETCOM instruction for more information.

Routine:

VR(0)=0	'initialize VR(0) and MODBUS_REG(0) as 0
MODBUS_REG(0)=0	
SETCOM(38400, 8,1,0,0,4,0)	'VR is mapped into MODBUS_REG
VR(0)=100.345	'set VR(0) = 100.345
?MODBUS_REG(0)	'print result is 100, VR had been mapped to REG, but REG is integer type, which means fractional part will lose
MODBUS_REG(0)=200	'set REG(0) as 200
?VR(0)	'print result is 200, REG changes, VR also changes.

✧ When using the MODBUS protocol to communicate with other devices, it is necessary to transfer data in the MODBUS register, such as communication with a touch screen. When MODBUS communication is not performed, the MODBUS register can also be used as a local array of the controller.

✧ The controller directly corresponds to the input IN port from the MODBUS_BIT address 10000, 20000 corresponds to the output OUT port (note that the read IO is the original state, the INVERT_IN inversion input instruction does not work), 30000 corresponds to the S register programmed by the PLC.

✧ MODBUS_IEEE addresses starting from 10000 correspond to the axis DPOS range, starting from 11000 correspond to the axis MPOS range, starting from 12000 correspond to the axis VP_SPEED range, MODBUS_REG addresses starting from 13000 correspond to the analog DA output range, and starting from 14000 correspond to the analog AD input range.

MODBUS_BIT Address	Meaning
0~7999	Customized use for users
8000~8099	special M register programmed by PLC
8400~8199	IDLE signs of axis 0-99
8200~8299	BUFFER reminding signs of axis 0-99
10000~14095	Relative input IN port
20000~24095	Relative output OUT port
30000~34095	Relative S register programmed by PLC

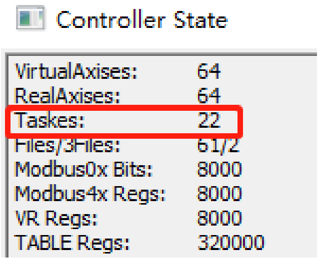
MODBUS Word Register Address	Meaning
0~7999	Customized use for users, MODBUS_REG, MODBUS_IEEE and MODBUS_LONG can be used together
8000~8099	special D register programmed by PLC
10000~10198	Corresponds to DPOS of each axis, use MODBUS_IEEE to write and read
11000~11198	Corresponds to MPOS of each axis, use MODBUS_IEEE to write and read
12000~12198	Corresponds to VPSPEED of each axis, use MODBUS_IEEE to read
13000~13127	Analog output AOUT, use MODBUS_REG to read and write
14000~14255	Analog input AIN, use MODBUS_REG to read

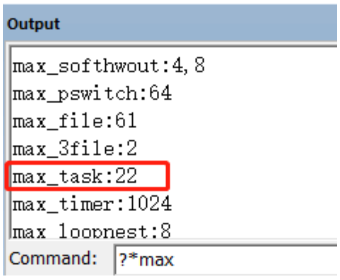
3.4 Multi-task Program

3.4.1 Concept of Muti-task

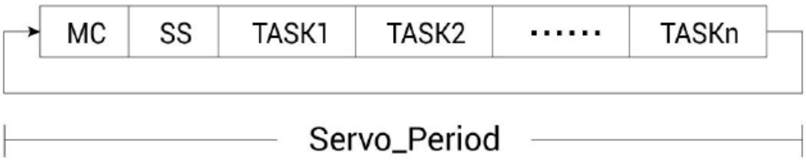
- ✧ Task is the function to execute a series of instructions processing, such as, I/O refresh, user program, etc. One task means one program that is running.
- ✧ If multiple program modules can run at the same time without interruption, which is called multi-task. And multi-task program can be achieved in the ZDevelop software.
- ✧ Multi-task takes a complex program apart several parts, which means it starts task separately and tasks are executed simultaneously, each task is in independent. In this way, the complicated motion process of equipment will be simpler, programming is more flexible. Program only can be executed in sequence when there is no multi-task, the executing efficiency is extremely low.
- ✧ ZMC motion controller supports multi-task programming, every task has own unique number. These numbers don't have priority, they are just identification that the task of the current program.
- ✧ Different models support different task amounts. After connecting to controller, "State the controller" in ZDevelop menu bar can check the exact task amounts. Also, it can be known through sending "?*max" in "command". As shown in the figure below, the controller

supports 22 tasks, and the task number range is 0-21.





✧ Each motion control cycle (Servo Period) of the motion controller includes the operation of MC, SS, and user multi-task program, as shown in the following figure:



→ MC: achieve Motion Control, EtherCAT communication and interruption. Motion Control includes: single-axis motion control, multi-axis interpolation motion, robot positive and negative algorithm. EtherCAT communication includes PDO and SDO.

→ SS: System Service includes RS232, RS485 serial communication, CAN, EtherNET (MODBUS master and slave communication and ZDevelop service of corresponding software).

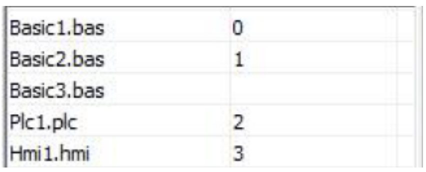
→ TASK1, …, TASKn: this relates to operation of each task, from task 1 to task n.

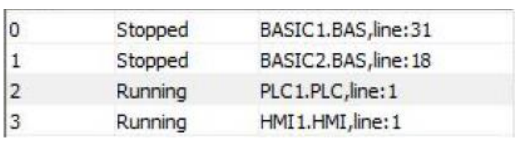
→ In one control period, if tasks execute different instructions currently, then occupied time also is different, it is not totally the same. There is no priority of task in default situation, but one certain task can be set the priority through PROC_PRIORITY instruction.

✧ All tasks in Basic are scanned to run once (unless there is an endless loop in the program). A Basic file under one project supports multiple auto-run tasks at the same time.

✧ The PLC main task is executed cyclically, and the PLC subprogram task only runs once. It is recommended to set only one Auto-run task number in the PLC file under one project.

✧ The HMI program needs to set the auto-run task number, and the initialization function only scans and executes once, and the periodic function scans cyclically. One HMI file is supported under one project, and the configuration program can run only by setting the auto-run task number for the HMI file.





✧ The controller processed 4 tasks at the same time, like the above figure. Among task 0, 1, 2, 3, they don’t disturb each other. After controller downloaded the program, 4 tasks start simultaneously, and when file task executing, SUB subprogram task or marking task will start by using task instruction. Once SUB subprogram task or marking task are opened, they become no relation with main program. Tasks can be triggered to execute again after task stopped.

✧ Advantages of controller multi-task:

→ Program modular: user can write several small and specific programs to achieve assigned functions that are consistent with customer’s equipment.

→ Concurrency: every task can run independently. When task starts, it won’t be influenced by other tasks.

→ Simplify the error process: Error handling becomes simple after dividing the multitasking operation, and only the task with error is processed.

→ Command interaction: when program is running, users can do command interaction in any time, such as, online modify motion parameters, send commands in online command bar, etc. And other programs don’t be affected.

3.4.2 Check Multi-task Status

Task has three states, they are Running, Stopped and Paused. Followings are 3 ways to see the state.

➢ Task instruction

PROC_STATUS: which means checking the task status, parameter only can be read. Return value: 0-task stops, 1-task is running, 3-task pauses.

Example:

```
PRINT   PROC_STATUS(0)          'print status of task 0
?*PROC_STATUS                   'print status of all tasks supported by controller
```

➢ Task window

Open task window through “Debug” – “Start/Stop Debug”, like the below figure.

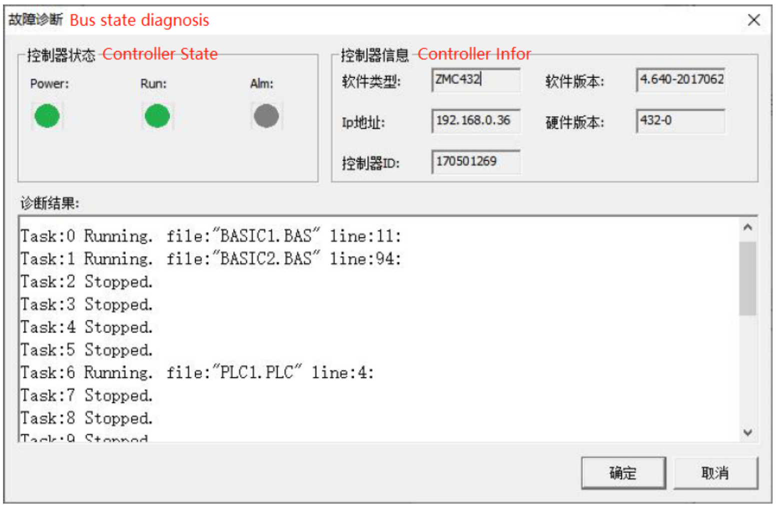
Task number and running status of started task, current file and operation line number can be viewed through this window, but tasks that don’t start can’t be known.

When Basic tasks finished scanning in the program, the task will become Stopped state. But PLC main task is always the Running state because it scans round.

➢ Open menu bar "Debug" – "Bus state diagnosis" window

Status of all task numbers, current file and running line number all can be checked.

This window also shows all tasks error information.



3.4.3 Multi-task Start and Stop

➢ First, multi-task operation instructions

END: the current task ends normally.

STOP: stop the running task of assigned file.

STOPTASK: stop assigned task.

HALT: stop all tasks.

RUN: start a new task and run a file.

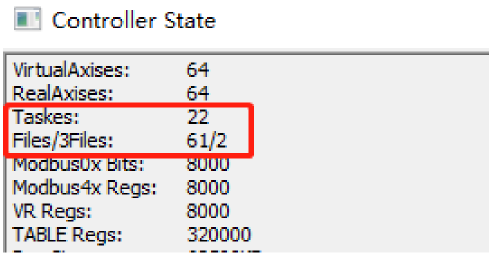
RUNTASK: start a new task and run one SUB or one program with labels.

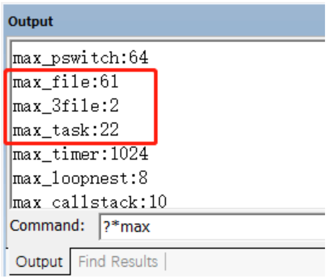
PAUSETASK: pause assigned task.

RESUMETASK: resume assigned task, then task will execute from that pause position.

Task operations in Basic and PLC both use above instructions.

“State the controller” and “?*MAX”: check task total amounts and file total amounts supported by controller.





- Second, start multi-task

There are 3 methods, they are auto-running task number configuration, RUN instruction and RUNTASK instruction. When using instructions to start task, task will be opened after this instruction is scanned by program.

Pay attention to the task number writing when starts task, tasks can’t be opened repeatedly.

1) Auto-running task number:

set auto-running task number through "FileView" window. After the controller is powered on, the file with the auto-running task number will be executed first. Basic file can set several AutoRun task numbers, but only one PLC file and HMI file are supported. The auto-run files are run in parallel, and they are turned on at the same time after power-on.

2) The file as one task is turned on through RUN instruction:

Example:

RUN "TuXing_001.bas",2 'set the file TuXing_001.bas as task 2, and start

3) SUB subprogram or signed program are set as one task and are turned on through RUNTASK instruction. Start global SUB subprogram through cross-file, and the label program that needs to start task only can exist in this file.

Example:

RUNTASK 1,task_home 'set as task 1 to start the task_home subprogram

- Stop multi-task

Instructions to stop multi-task: STOPTASK, STOP, HALT.

Task stops, then restarts it, it will execute from the beginning.

When starts task, usually use STOPTASK to stop the task firstly. Then start through RUNTASK for avoiding errors caused by start repeatedly.

1) STOPTASK supports stop file taskm SUB subprogram task and labelled task.

Example:

STOPTASK2 'stop task 2

2) STOP instruction supports stop Basic file task. It is recommended to use STOPTASK instruction, because the operation is simpler.

3) HALT instruction stops all tasks.

Example:

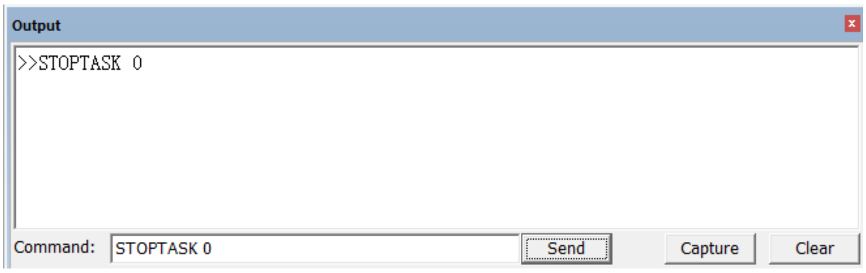
HALT 'stop all tasks in project

Also “Emerge Stop” button can be used to stop all tasks rapidly.

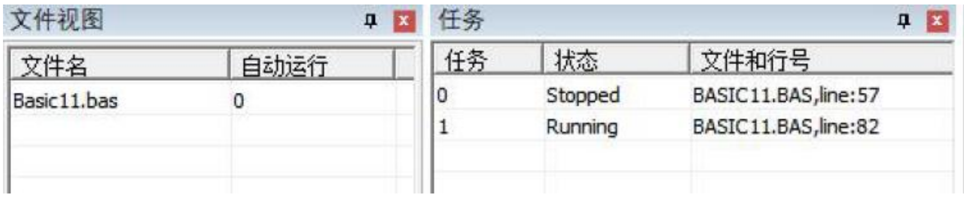
Example:

There are 2 tasks in project, after they are downloaded, task 0 and task 1 are running.

Send online command: STOPTASK 0



Stop task 0



When restarts the task, program can be downloaded again.

The above program cannot use the RUN command to start the auto-running file task 0, because the automatically opened task 1 in task 0 is still running. If the command is used to start task 0 again, it will cause task 1 to be opened repeatedly. If task 1 is stopped, start task 1 independently through RUNTASK instruction.

3.4.4 Pause and Resume of Task

Use PAUSETASK command to pause task, and use RESUMETASK command to resume task. After resuming, the task continues to execute from where it was suspended. And paused tasks support stopping.

1) PAUSETASK: pause assigned task

Example:

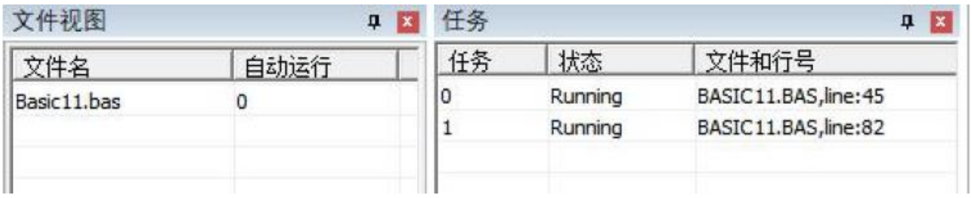
PAUSETASK 1 'pause task 1

2)RESUMETASK: resume assigned task

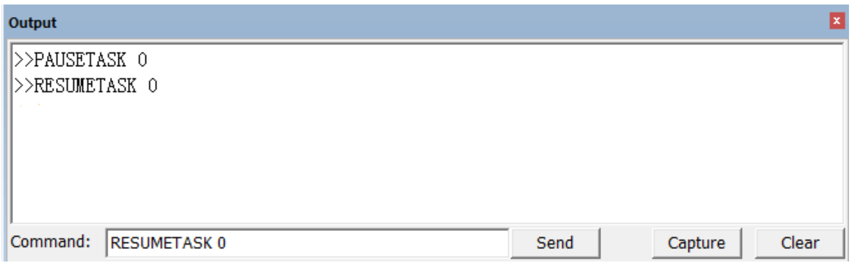
Example:

RESUMENTASK 'continue to running task 1

Example: there are 2 tasks in project, after they are downloaded, task 0 and task 1 are running.



Send online command to control task is paused or resumed.



Send: PAUSETASK 0

Task 0 is paused.

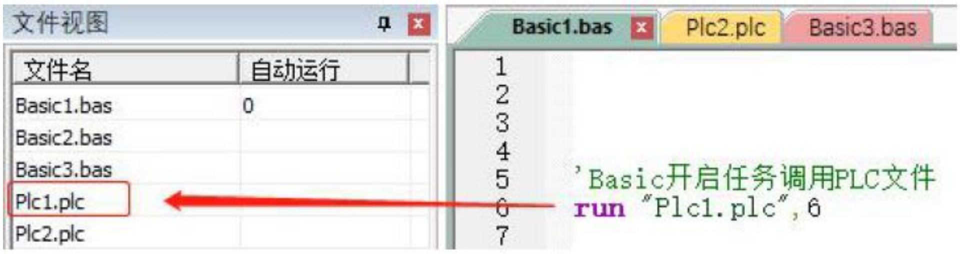
Send: RESUMETASK 0

Task 0 resumes the operation state.

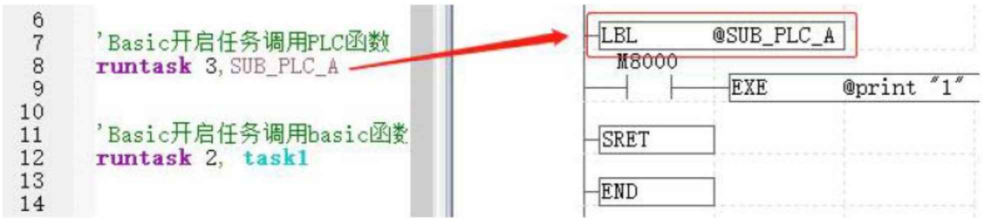
3.4.5 Basic and PLC Task Call Each Other

➢ First, Basic calls PLC task.

1) Basic file uses RUN instruction to call PLC file.

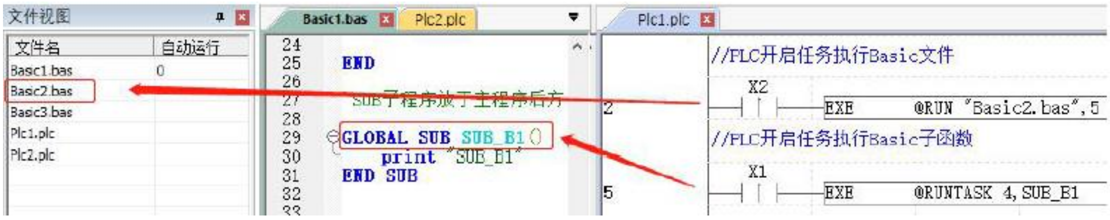


2) Basic file uses RUNTASK instruction to call subprogram defined by LBL instruction in PLC.



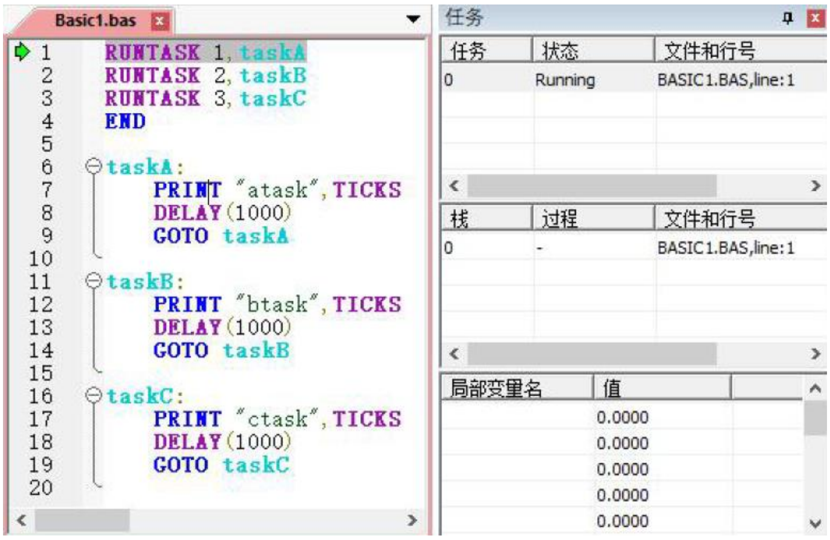
➢ Second, PLC calls Basic task.

For PLC, using EXE or EXEP (pulse execution) instructions to call Basic task, then calling Basic file task or subprogram task.

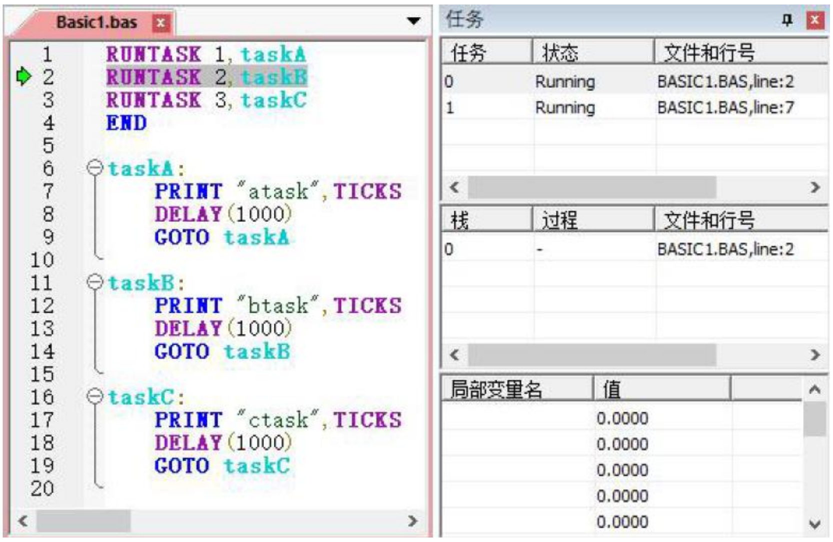


3.4.6 Multi-task Routine

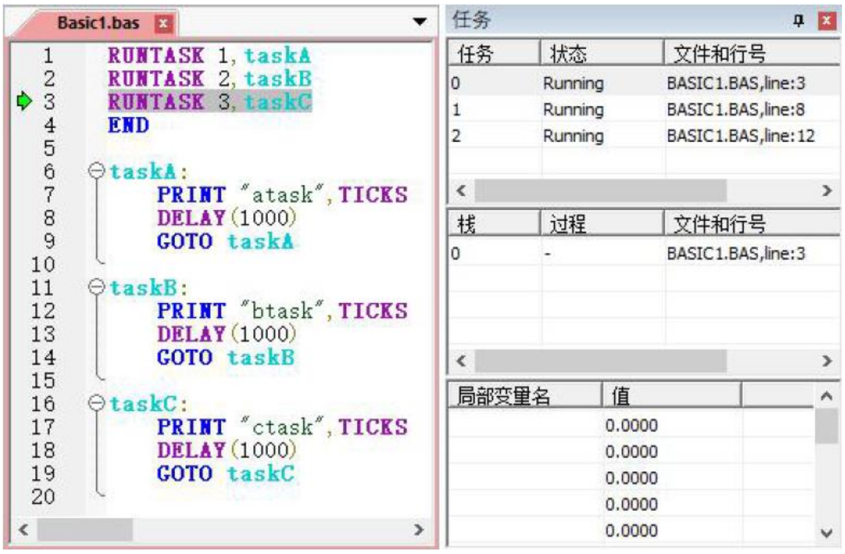
The following routine, there are 4 tasks, one main file task 0 and 3 module files 123. Start single-step debugging, check effects of multi-task running, and observe the direction of the cursor on the left. After the program scans to RUNTASK1, it starts the task taskA. After it starts, it continues to scan the next line, RUNTASK2, task B also starts, RUNTASK3 starts task taskC, and it will stop scanning until meeting END main task 0. taskA, taskB, and taskC are executed separately as independent tasks. The program execution can be seen in the program debug window.



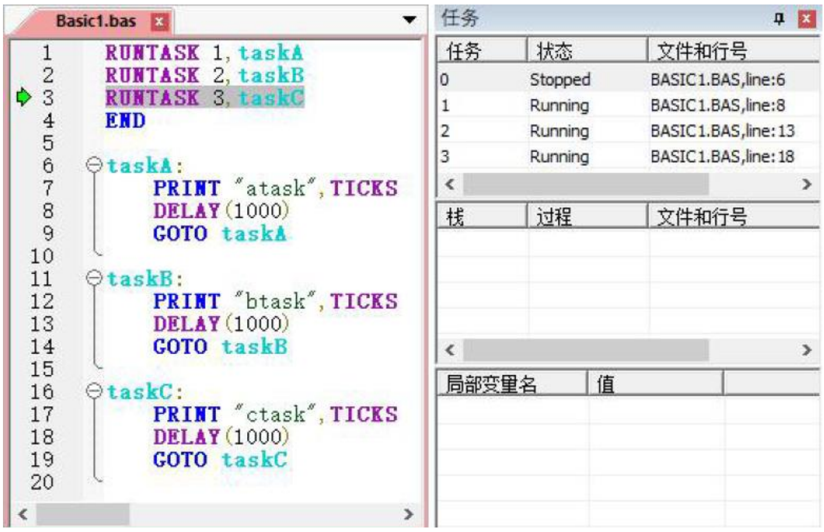
There is only auto-task 0 when power-on



Task 0 starts task 1 to run



Task 0 starts task 1 and task 2 to run



Task 0 starts task 1, task 2 and task 3 to run

3.5 Three Kinds of Interruption

✧ There are 3 types of ZBasic interruption, power failure interruption, external interruption and timer interruption.

✧ Main switch of interruption must be turned on before using interruption, in this way, entering interruption when program has initialized well. And the interruption switch is closed state by default when controller powers on.

✧ When these three kinds of interruptions are running, the interruption function independently occupies one task number, which means there isn't push stack situation.

➢ Precautions of interruption usage

→ There is no priority among these interruptions.

→ Interrupt nesting is supported, multiple interrupts can be executed at the same time, but too many interrupt functions should not be processed at the same time.

→ There is only one task inside the controller that processes all interrupt signal responses, and there is a fixed interruption task number. If interruptions handle too many functions and the code of the interrupt handling function is too long, all interrupt responses will be slowed down, or even interruption blocked, affecting execution of other interruptions.

➢ Solutions:

→ Decrease the number of interruption in a way, actually many applications can be achieved through cyclic scan.

→ If the interruption processes an extremely long function, it is recommended to call one independent task to handle the complex task in interruption, then other interruption responses won't be blocked.

3.5.1 Power Failure Interruption

- ✧ It must be a global SUB function. The controller has only one power failure interruption. The execution time of power failure is very limited, and only a few commands can be written to store the data in the VR.
- ✧ Relative function: INT_ENABLE, ONPOWEROFF.

Example:

```
INT_ENABLE=1
DPOS(0)=VR(0)             'read saved value when power-on, recover coordinate
DPOS(1)=VR (1)
DPOS(2)=VR(2)
END                       'main program ends

GLOBAL SUB ONPOWEROFF()   'power failure interruption
    VR(0) = DPOS(0)       'save coordinate into VR
    VR(1) = DPOS(1)
    VR(2) = DPOS(2)
END SUB
```

3.5.2 External Interruption

- ✧ Rising edge trigger or falling edge trigger can be set, it must be a global SUB function, currently only interrupt IN ports 0-31 can be used. Only firmware that supports PLC function can be used. For details, please consult ZMOTION technicians.
- ✧ Relative function: IN_ONn, INT_OFFn.

Example:

```
INT_ENABLE=1              'Open interruption
END                       'main program ends

GLOBAL SUB INT_ON0()      'external rising edge interrupt program
    PRINT "triggered when meeting rising edge of IN0"
END SUB

GLOBAL SUB INT_OFF0()     'external falling edge interrupt program
    PRINT " triggered when meeting falling edge of IN0"
END SUB
```

2.5.3 Timer Interruption

- ✧ The function to be executed after reaching the set time must be a global SUB function. Timer interruptions can start several functions simultaneously. And the number is determined by the number of timers. The number of timers depends on the controller model. Use ?*max to print and view.
- ✧ Relative function: ONTIMERn.

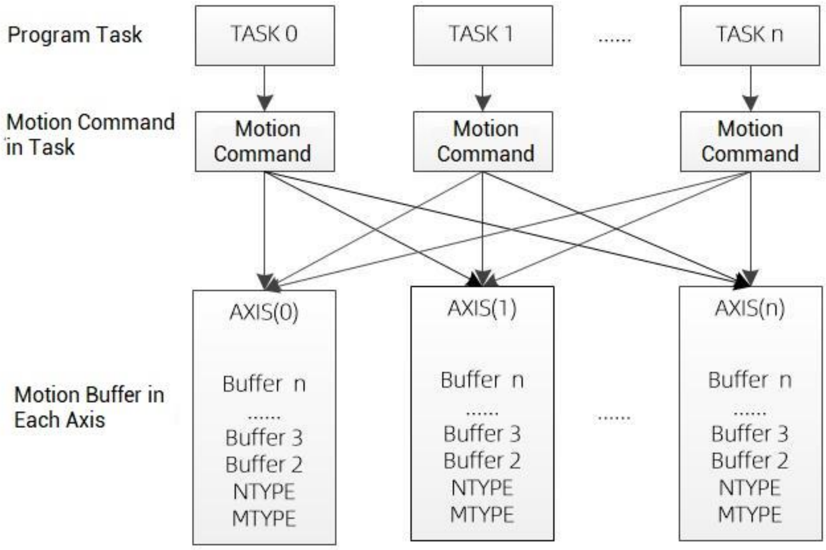
Example:

```
INT_ENABLE=1                'start interruption
TIMER_START(0,100)          'timer 0 open, cycle time is 100ms
END                         'main program ends

GLOBAL SUB ONTIMER0()       'interruption program
    PRINT "ontimer0 enter"
  'TIMER_START(0,100)    'execute interruption periodically, open timer again in SUB
END SUB
```

3.6 Motion Buffer

3.6.1 The Concept of Motion Buffer

- ✧ When running the motion instruction, in order to prevent the program from being blocked, the controller provides a buffer to save the motion buffer queue entering the motion buffer. This function is called motion buffer, so that the program can scan down normally without blocking.
- ✧ ZMotion motion controller has multilevel motion buffer. When the motion buffer is turned on, and when the program scans and recognizes the first motion instruction of the program task, it will assign the motion instruction to the motion buffer of the specified axis, and the motor starts to move. At this time, the program continues to scan down to the second motion, then it is stored in the motion buffer, and while the motion instructions are continuously scanned and stored, the motion commands are sequentially taken out from the motion buffer and executed.
- ✧ MTYPE is the current running motion instruction and NTYPE is the first buffer motion instruction.
- ✧ Motion instructions of any program can enter motion buffer of any axis, which is assigned by axis number.
- ✧ Motion buffer areas of each axis are independent, they don’t bother each other.



3.6.2 Motion Buffer

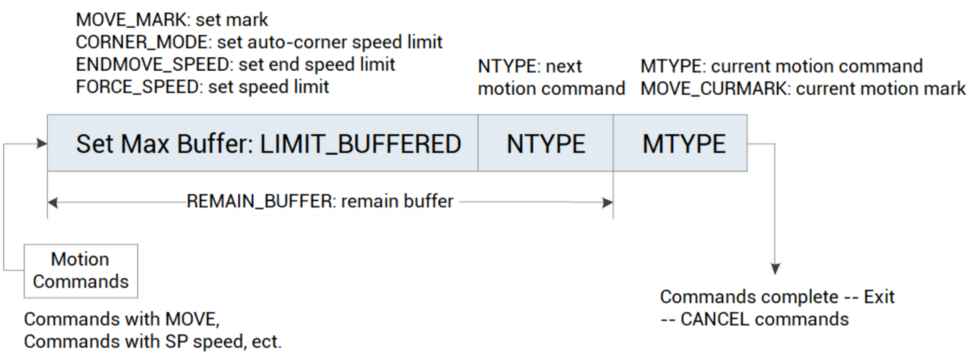
✧ During the program scanning, the scanned motion instructions are stored in the motion buffer of the corresponding axis, and the motion instructions are fetched and executed from the motion buffer in the order of first-in, first-out. In addition, it also includes a series of output instructions in motion buffer.

✧ MOVEMODIFY and MOVEMODIFY2 are special, they will not enter the motion buffer.

✧ Interpolation motion buffer is in the motion buffer of main axis.

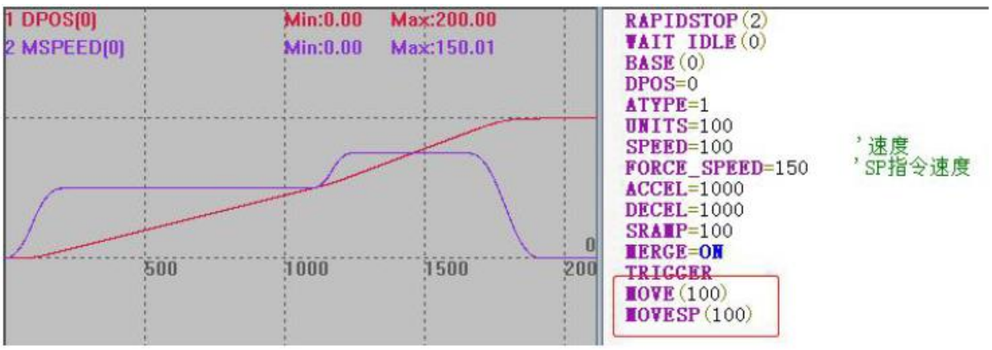
✧ When buffering multiple motion instructions, in order to judge the current executing motion instruction, there are MOVE_MARK motion label and MOVE_CURMARK current motion label instructions to check. MOVE_MARK motion label will add one when scanned one motion instruction; MOVE_CURMARK instruction is the current motion label, indicating which motion instruction the current motion reaches, and -1 after all motions are completed.

✧ When the current motion finished, it will automatically execute the next motion of motion buffer. When all instructions are executed, the motion buffer is blank, or use CANCEL/RAPIDSTOP instruction to clear motion buffer.

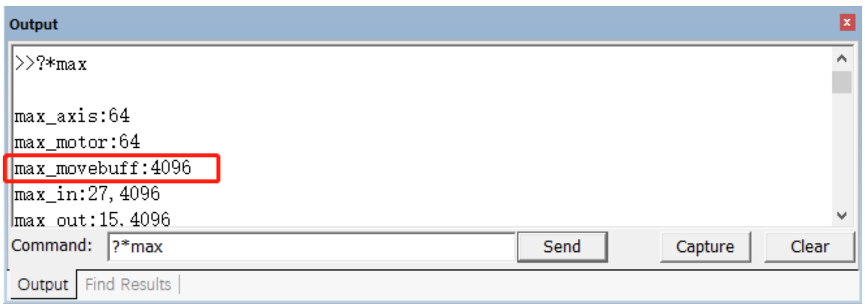


✧ SP instruction is also called SP motion instruction, when using SP motion instructions (MOVESP, MOVECIRCSP, etc. Only add SP behind the motion instruction directly.), the motion of SP instruction moves as the SP speed, not the SPEED speed. SP speed includes FORCE_SPEED, ENDMOVE_SPEED and STARTMOVW_SPEED, they will follow SP motion instructions to be written into motion buffer.

✧ The operation effect of SP instruction and non-SP instruction is as follows, the speed of MOVE(100) is SPEED=100, and the speed of MOVESP(100) is FFORCE_SPEED=150.



✧ Each axis of the ZMC4 series motion controller can support up to 4096 motion buffers (the number of buffers varies for different models of controllers, see the controller hardware manual for details or use ?*max to print to view the max_movebuff parameter). And LIMIT_BUFFERED motion buffer limit can be set manually.



✧ Each axis' motion buffer is independent, they won't disturb each other, and the size of axis' motion buffer are the same. The number of remain buffer of one certain axis can be checked through REMAIN_BUFFER(MTYPE) AXIS(N).

```
>>?REMAIN_BUFFER(0) AXIS(0)
4096
>>?REMAIN_BUFFER(1) AXIS(0)
4096
>>?REMAIN_BUFFER(4) AXIS(0)
376
```

✧ The buffer space occupied by different motion commands is different, and the more complex motion occupies, the more motion buffer space is occupied. For example, ZMC432 controller, the size of the motion buffer is 4096, and the number of MOVE linear interpolation instructions and MOVECIRC circular interpolation instructions that can be buffered at one time in the buffer is different.

3.6.3 Motion Buffer Blocked

✧ Since the motion buffer space of each axis is limited, when too many motion instructions are scanned into the motion buffer, the multi-level motion buffer will be full. If the program continues to scan more motion instructions, the program will also be blocked. Until the motion commands are completed and exited in sequence, and the motion buffer has a vacancy, the motion command will continue to enter the motion buffer.

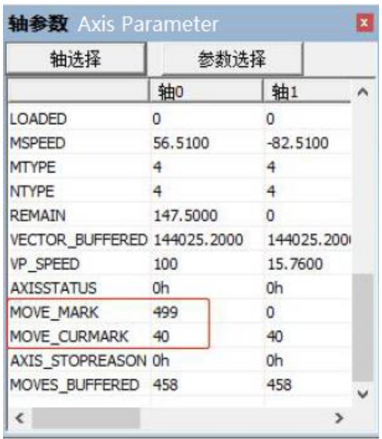
Example:

Take V3.10 version simulator as an example, the default is 4096 motion buffers, the routine in the following figure shows that the motion buffer of the controller can store up to 459 circular interpolation instructions, and the value of i is 485 after downloading the program, which means that the current FOR loop has not been executed and the program is blocked.

Motion Buffer Block Effect – Circular Interpolation

Motion Buffer Block Effect – Linear Interpolation

✧ As shown in the figure below, when some arc motion commands are taken out from the motion buffer and are executed, the buffer has space, FOR loop continues to execute, and saves the motion command into the motion buffer. After the instruction is executed and exits the motion buffer, as long as there is enough space in the motion buffer, new motion instructions will be stored in the motion buffer one by one.



✧ In order to prevent the program from being unable to continue to scan down due to the blockage of the motion buffer, we can add a judgment processing program when scanning motion instructions to confirm that there is space in the buffer before scanning motion instructions.

3.6.4 Output in Motion Buffer

- ✧ The output command in the motion buffer can enter the motion buffer. In the motion buffer, it can start the OP port, delay, output parameters, output PWM, start tasks, etc. For detailed instructions, please refer to the Chapter Motion Instruction.
- ✧ The difference between normal output and output in motion buffer:

  → Ordinary output instruction program scans this line of instructions and executes the output.

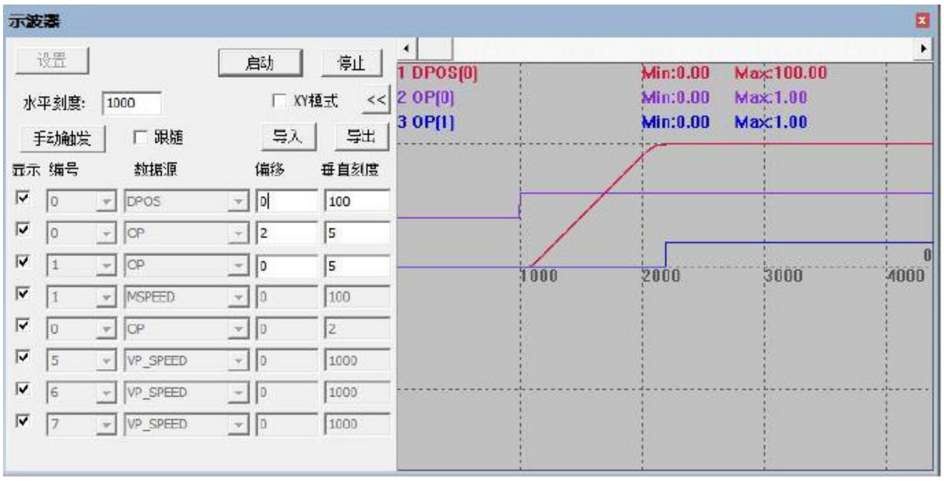
  → The output instruction in the motion buffer is stored in the motion buffer after the program is scanned, and the motion buffer is fetched and executed in the order of first-in, first-out, and the output will not be executed until the output instruction is fetched.

Example:

```
RAPIDSTOP(2)
WAIT IDELE(0)

BASE(0)                         'select axis 0
DPOS=0
UNITS=100            'pulse equivalent
SPEED=100            'speed
ACCEL=1000           'acceleration
DECEL=1000           'deceleration
SRAMP=100            'S curve
TRIGGER              'Trigger oscilloscope sampling
OP(0,3, $0)          'close output port 0-3
DELAY(1000)          'delay
MOVE(100)
MOVE_OP (0,ON)       'output in motion buffer
OP(0,ON)             'output normally
END
```

Running effect of the example:

After delaying 1s, program scans OP instruction, then output 0 is executed to output immediately. MOVE_OP fills the IO operation command into the motion buffer, so after MOVE(100) is executed, the output port 1 will be output.



Chapter IV Communication Method

4.1 Serial Port Communication

4.1.1 The Serial Port Type

Controller includes 3 types of serial ports, RS232, RS485 and RS422. And all controllers have RS232, most controllers have RS485, only a few controllers have RS422.

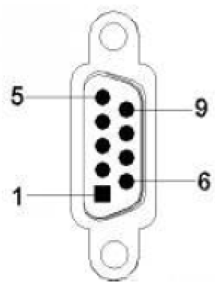
Controller's serial port protocols are MODBUS_RTU, they are as slave station by default. RS232 and RS485 can be set as master station through SETCOM instruction, and communication rate and other parameters are configured through SETCOM instruction.

Controller serial port default parameter: Baud rate 38400, data-bit 8, stop-bit 1, no parity bit, not support power failure storage.

➢ RS232

The RS232 interface of the controller can be used as a MODBUS master station or a slave station, supporting 1 master station to send data and 1 slave station to receive data. When used as the master station, it can be connected to a driver, inverter, temperature controller, etc. to control data read and write. When used as a slave station, it can be connected to the HMI to monitor the running status, and is often used to connect to a PC or HMI.

RS232 controller uses DB-9 interface, below is the pin signal description:

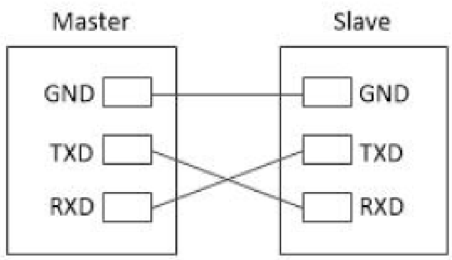


Pin Number	Name	Description

2	RXD	Receive data pin
3	TXD	Send data pin
5	EGND	Power ground
9	E5V	External power 5V output

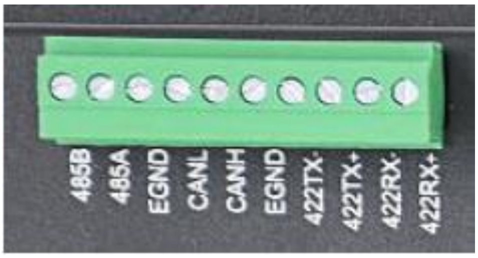
RS232 only needs to wire 3 cables, 2 data signal TXD and RXD, 1 ground cable GND. Data signal RXD and TXD are cross connected, then connecting with GND together.

Wiring reference:



➢ RS485

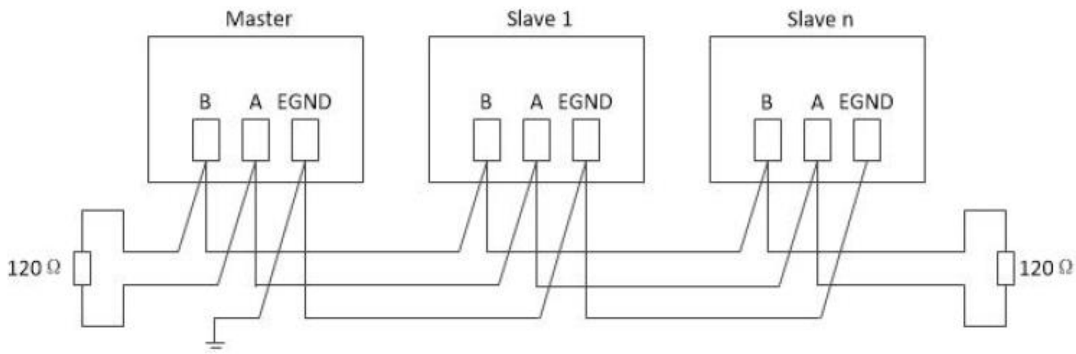
It mainly provides the connection of multiple communication devices of the master/slave station, and theoretically supports 128 nodes, one master and multiple slaves. When it is used as a master station, it can be connected to drives, converters, temperature controllers, etc. to control data read and write; when used as a slave station, it can communicate with PLC, and can be connected to a HMI to monitor the running status.



RS485 interface uses differential transfer method, and judges the high-level electricity or low-level through voltage difference between A and B.

Pin Name	Description
485B	485-
485A	485+
EGND	Power ground

The RS485 interface of the controller adopts a simple wiring method. As shown in the figure below, 485A, 485B, and GND ground wires of the controller are respectively connected to the A, B, and ground wires of the first slave station, and then connected to the second slave station. A, B, ground wire (A to A, B to B, signals share ground), and 485A and 485B of the controller and the last slave station should be connected in parallel with 120Ω resistance to prevent signal reflection, the cable needs to use shielded twisted pair, to avoid signal interference, the distance of each node branch line should be less than 3m.



➢ RS422

Some model controllers of ZMC3XX series have RS422.

RS422 data transmission characteristics are the same as 485. RS422 adopts a four-wire system, marked as RX+/RX- (receive signal), RT+/RT- (send signal), one signal ground wire, a total of 5 wires. The four-wire interface uses separate sending and receiving channels, therefore it is not necessary to control the data direction.

Pin Name	Description
422TX-	Send data -
422TX+	Send data +
422RX-	Receive data -
422RX+	Receive data +
EGND	Power ground

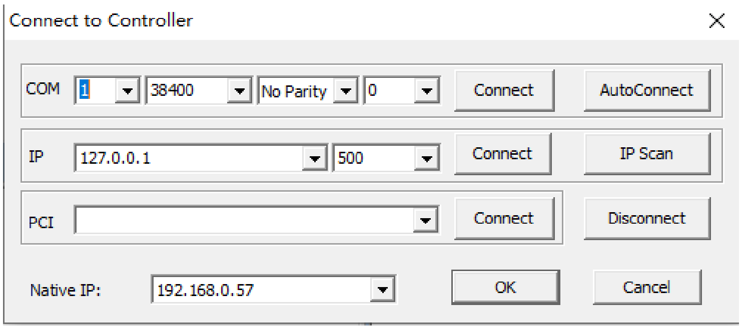
The RS422 interface of the controller adopts a simple wiring method, but compared with RS485 and RS232, the wiring cost is high, and the wiring is easy to make mistakes. The RS422 interface of the controller only supports access to one device.

4.1.2 Serial Connection Method

The serial port supports the MODBUS communication protocol RTU mode, which is often used to connect to a computer or touch screen. When communicating, pay attention to the matching of the serial port parameters. No matter which serial port, except for the port number and wiring method, the default parameters and operation instructions are the same.

Below is the way for PC to use serial connect and control:

→ Connect the cable first, click "Controller" → "Connect" in the ZDevelop menu bar, open the following window to connect to the controller, it will automatically list the serial port numbers available on this computer, select the serial port number to be connected, and set the baud rate, parity bit and stop bit, click connect, and the connection status and the corresponding information will be print out automatically in the software output window.



The default parameters of the serial port of the controller: Baud rate 38400, data bit 8, stop bit 1, no parity bit. If the serial port connection fails, check whether the serial port number is correct, and modify the configuration of the communication port COM of the computer to make it match the default parameters of the controller.

The serial port parameters are set through SETCOM instruction. The serial port parameters don’t support power failure storage. After the controller is powered on again, the SETCOM parameters will recover their default values, so please write SETCOM configuration at the beginning of the program.

The serial port is MODBUS slave by default. It can be set as master station by modifing the MODE=14 of the SETCOM instruction, or set MODE=0 to open the serial port custom communication, namely, no protocol mode. In the serial port custom communication mode, use the GET # command to read data from the customized serial channel. PRITNT # command outputs strings from the customized serial channel, PUTCHAR # command outputs characters (ASCII code) from the customized serial channel.

SETCOM instruction mode parameters configuration protocol:

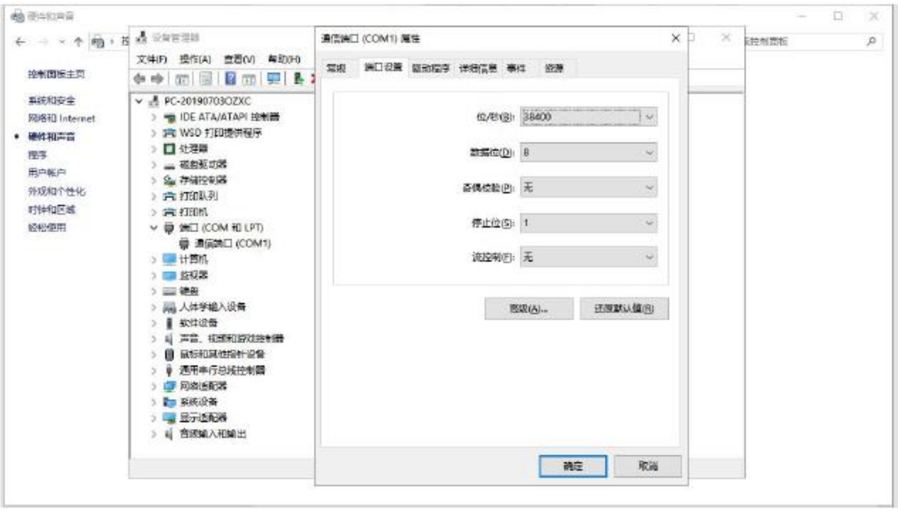
Mode Value	Description
0	RAW data mode, no protocol, at this time, data can be read through GET#. Send data through PRINT#. PUTCHAR# instructions.
4 (default)	MODBUS master station (16-bit integer)
14	MODBUS slave station (16-bit integer)
15	Direct command execution mode, now input character string command from serial port directly (newline ends)

If the connection fails, check the following methods:

→ Check whether the serial port cable is a crossover cable.

→ Whether the COM port number and parameters in "Connect to Controller" are correct.

→ Open the computer "Device Manager" - "Port" - "Communication Port (COM)" - "Port Setting" to check whether the COM port setting is correct. The default parameters of the controller serial port: Baud rate 38400, data bit 8, stop bit 1, no parity bit.



The com port number can be changed in the "Port Settings" - "Advanced" option, which can be selected through the drop-down list.

→ When connecting to the controller through the serial port, the corresponding serial port of the controller must be configured as MODBUS slave protocol mode (default mode), which can be restored after power off and restart.

→ Whether the COM port is occupied by other programs, such as serial debugging assistant, etc.

→ Whether there is enough serial port hardware on the PC side.

→ Replace the serial cable/computer test.

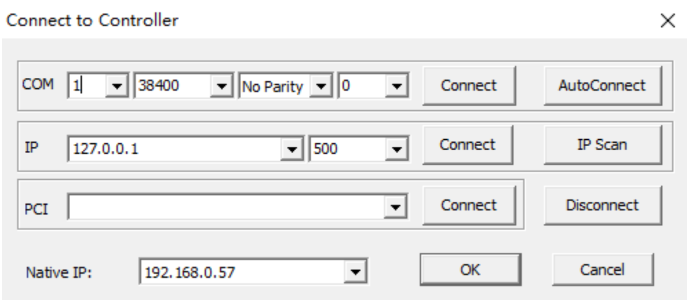
4.2 Net Port Communication

Controller network port is EtherNET interface, which supports MODBUS_TCP communication protocol, and it is usually used to connect to computer or touch screen. Generally, the controller has one EtherNET interface, but there are at least 2 network port channels at the bottom. When net port is needed to connect to multiple equipment, the switch can be used. The number of net port channel can be viewed through ?*PORT instruction.

It is recommended to use twisted pair cable with shield layer for good quality of communication. In ZDevelop menu bar, click "Controller" – "Connect", open the window like the below figure, and IP address is selected from the list. It will automatically find valid IP address within the current LAN, then select correct IP and click "Connect".

When using network port, controller IP address and computer IP address must be the same network segment, which means first three segments are the same and the last segment is different. Otherwise, it will fail to connect, for this situation, modifying controller IP or computer IP.

The controller factory IP is 192.168.0.11, and if IP address is modified, it will be stored forever.



Controller network port also supports self-defined communication, using OPEN # instruction to open the self-defined network port communication, using GET # instruction to read data from the channel, using PRINT # instruction to output character string from the channel, using PUTCHAR #instruction to output character string (AACII code) from self-defined net port channel.

Item	Trouble Description	Solutions
1	When the controller connected with power, POWER and RUN indicator lights don't light.	✚ Check the power supply. ✚ If the power is normal, check whether the controller is burned out or not.
2	When the controller powered on and connected cable, but the net port indicator light doesn't work.	✚ Check whether the two sides of net segment are plugged well or not. ✚ Check whether the net cable is damaged or

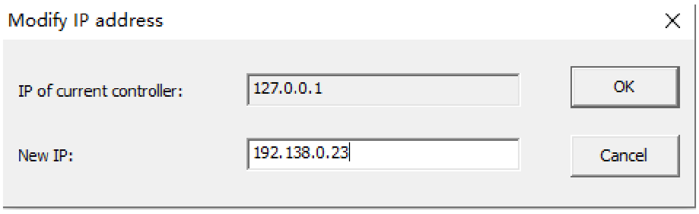
		not. ✚ Check whether the cable insert slot is damaged. ✚ Note: the cable is connected to EtherNET, not EtherCAT.
3	Fail to connect to the controller.	✚ Check whether controller IP address is the same as PC IP address, they must be at the same net segment, see next for details of modification. ✚ If controller net port channels are occupied, it is recommended to close some channels that are not used now, trying to connect again. ✚ If the computer is stuck seriously, please try several times to connect with software. ✚ Restart the controller, then do above connection steps again.
4	Succeed in connecting, but sometimes it loses connection when using.	✚ It is recommended to use metal interface with shielded cable. In the case of serious interference, using a crystal head network cable with no shielding layer will cause unstable communication and occasional disconnection.

✧ Modify controller IP address:

Use serial port to connect with controller firstly, then obtain controller IP address, modifying it now.

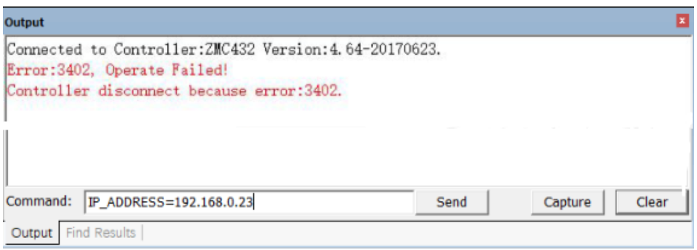
→ Method 1:

"Controller" – "Modify IP address" window can modify controller IP address directly.



→ Method 2: send online command to modify through IP_ADDRESS command.

After modifying successfully through instruction, it will disconnect automatically. Then online command print controller connection error information, using net port connection and selecting new IP address 192.168.0.23 to connect with controller again. Modified IP address is valid forever.



✧ Modify computer IP address

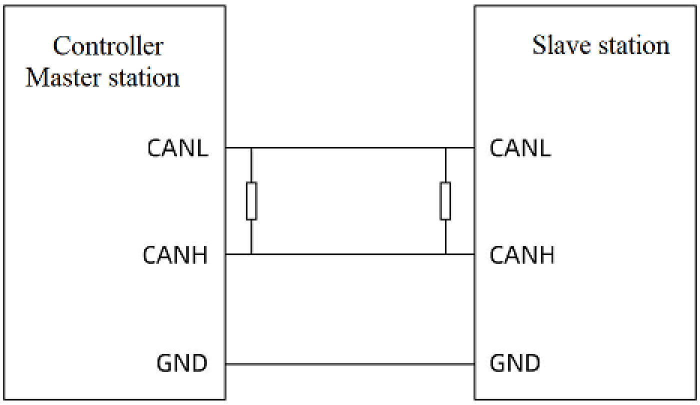
First, see the address 4 of computer native IP protocol is 192.168.0.xxx or not, the first three segments are the same as controller and the last segment can’t be the same, the controller factory default IP address is 192.168.0.11. If the third address is different, relative subnet mask should be modified as 0.

Then, open “Connect to controller” window to connect again.

4.3 CAN Bus Communication

4.3.1 CAN Wiring

Controller CAN bus interface is used to connect with ZCAN expansion module or controller. And their connection method is the same, the difference is the board doesn’t integer a 120Ω resistor, so CANH and CAHL two sides both need one 120Ω resistor. For wiring reference of CAN bus and expansion module, please see “Chapter ZCAN Expansion Module”.



When controller uses CAN bus connection, now communication between controllers can be achieved through CAN instruction, and data is transferred through TABLE.

Controller is the master station of CAN communication by default. When controllers are doing CAN communication, one controller should be configured as slave station, using CANIO_ADDRESS command to configure the master station and slave station, CANIO_ADDRESS=32 means master station, when CANIO_ADDRESS=other values, which means slave station.

Command Grammar: CAN (channel, function, tablenum)

→ channel: CAN channel, 0 means the first channel, -1 means the default channel.

→ function: function number (see the below form, mode 6/7 suits to standard frame, mode 16/17 suits to expand frame)

→ tablenum: TABLE position of saving data

Value	Description
6	Receive, when there is no data, identifier < 0
7	Send
16 (needs to upgrade firmware)	Support receive with expansion, when there is no data, identifier < 0
17 (needs to upgrade firmware)	Send expanded data, use 7 to send normal data

Example:

```
'send station: the first controller
TABLE(0,1,8,1,2,3,4,5,6,7,8)          'send cobid = 1, 8 bytes (1-8)
CAN(0,7,0)                            'send data
'receive station: the second controller
CANIO_ADDRESS = 1                     'set as CAN slave station, and this only is set once
CAN(0,6,0)                            'receive data
?TABLE(0)
```

4.4 U Disk Interface

- ✧ Most ZMOTION motion controllers have one standard U disk interface.
- ✧ When there is no controller, new build one udisk folder in ZDevelop root directory for simulating U disk, then connect to simulator, debug U disk instruction.
- ✧ U disk interface has three aspects usage:

1. Program Update

Download packaged zar program package through U disk interface, which is convenient for customer to update system program.

Before program updating, zar program package should be downloaded into program. To load U disk file through FILE instruction successfully, then zar program will operate automatically.

Example:

```
DIM  result                    'define variable
IF U_STATE=TRUE THEN           'U disk plug and judge
     result = FILE "find_first","zar",10      'scan the first zar file, and the fie name is saved VR
     IF result=TRUE THEN       'scan file and judge successfully
      File"load_zar", VRSTRING(10,20)  'download the same filename file as zar file stored   in VR
   ENDIF
 ENDIF
 END
```

2. Upload three file

Use FILE command to upload the three file that is saved in U disk and execute it.

Example:

```
IF   U_STATE=TRUE THEN          'judge U disk is plugged or not
   FILE "FIND_FIRST","Z3P",800      'find Z3P file
   ?"file name: "VRSTRING(800,20), "wait to downloading"
   FILE "COPY_TO", VRSTRING(800,20), VRSTRING(800,20)   'download Z3P file
   ?"complete to download Z3P file"
ENDIF
```

3. U disk and register data interaction

The U disk supports reading and writing variables and arrays.

FLASH data copy: The data stored in FLASH in multiple controllers can be transferred to each other through U disk.

Data in VR register, TABLE register and U disk can be transferred to each other.

The read-write file type is SD(filenum).BIN or SD(filenum).CSV, and the file types that can be operated by different commands are different.

Different types of controllers have the same usage of the U-disk interface. Just insert the U-disk into the UDISK port on the controller. After the controller is powered on, when a U-disk is inserted, the U-disk indicator will light up.

Before operating U disk, first to judge U disk status through U_STATE instruction for ensuring successful communication, then use U disk relative instructions to operate.

Example:

```
DIM   a,array1(2)                    'variable, array definition
a=123
array1(0)=10
array1(1)=20
IF  U_STATE = TRUE THEN              'judge U disk is plugged or not
    U_WRITE 0,a,array1               'write variable and array into U disk SD0 file
  a=456
    array1(0)=11
    U_READ  0,a,array1               'read U disk file SD0 data
    PRINT a,array1(0)                'result: 123, 10
    IF a <> 123 THEN                 'judge U disk is written and read successfully
        PRINT "U disk read wrongly"
    ELSE
        PRINT "U disk succeeds in reading"
    ENDIF
ELSE
    PRINT "U disk isn't inserted"
ENDIF
END
```

For more operations of U disk, please refer to “chapter U disk relative instructions”

4.5 EtherCAT Bus communication

4.5.1 EtherCAT Bus Initialization

The EtherCAT bus interface can be used to connect the EtherCAT servo drive and the EtherCAT expansion module. No matter what module is connected, the EtherCAT bus needs to write an EtherCAT bus initialization program to enable the motor and the EtherCAT expansion module. The application after enabling is the same as that of the pulse motor, and the motion instructions are the same.

Normal process of initialization program:

1. Use SLOT_SCAN to scan the device to determine whether the RETURN is correct, and no error will be reported when the device is not connected.

2. The device type and information are judged by NODE_INFO/ NODE_AXIS_COUNT, etc.

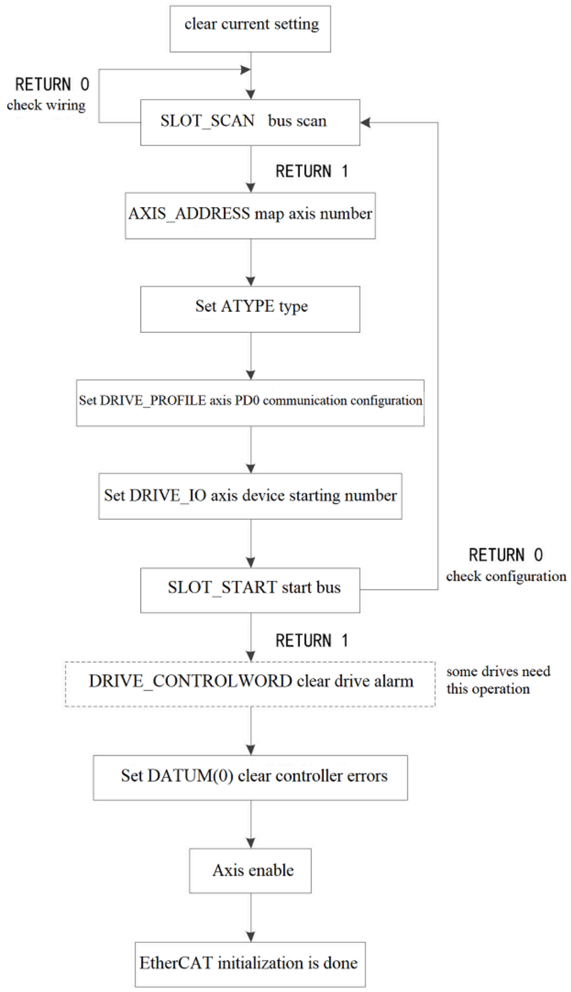
3. Set AIXS_ADDRESS, ATYPE, DRIVE_PROFILE, DRIVE_IO, etc. in turn.

4. Start the device through SLOT_START.

5. Set AXIS_ENABLE=1 for each axis to enable the axis, and set WDOG=1 to enable all axes. (some drives need use DRIVE_CONTROLWORD instruction to clear drive alarms)

6. After building connection, master station and slave station can exchange data periodically.

The general process of EtherCAT initialization program: refer to routine.



Basic concepts involved:

1. NODE node number: arranged according to the wiring sequence of EtherCAT devices, the number starts from 0 and decreases by 1 for the number of devices.

2. drive number: according to the wiring sequence of EtherCAT devices, the number starts from 0 to the number of EtherCAT drives minus 1. It only works when AXIS_ADDRESS is configured, that is, only the drive device is counted, other extended IO and other devices are not counted in the drive number.

3. axis number: the number set by the controller to the drive motor device connected to the controller. The number starts from 0 to the total number of connected devices minus 1. The axis number can be mapped to any connected drive device through the AXIS_ADDRESS command (pulse type controllers do not support local axis number mapping).

Precautions when setting:

1. The device number is sequentially incremented according to the connection sequence followed by the number 0, which needs to support the EtherCAT bus.

2. The axis number of the pulse axis on the bus controller is fixed. The bus can also call the axis number of the pulse axis. When ATYPE=65, the bus is called. Note that DPOS will change

when switching, so it is best not to make the axis number conflict.

3. After clearing the drive alarm through DRIVE_CONTROLWORD (axis number), a delay of 200ms is required, and then DATUM(0) is used to clear the controller alarm. Without the delay, it may be necessary to download the program twice to clear the alarm.

4. If DRIVE_CONTROLWORD (axis number) is used to operate a drive, then all drives after this drive will inherit this setting. So it's better to set it once for each drive.

5. Connecting or disconnecting a device during operation requires rescanning to update the status. For example, NODE_COUNT(0) returns the number of currently connected devices, and the number returned will change after rescanning.

6. AXIS_ADDRESS(i)=1, the first drive is selected, not the first device. For example, after connecting two expansion boards and then connecting two drives, the device numbers are expansion board A: 0, expansion board B: 1, drive A: 2, and drive B: 3. At this time, AXIS_ADDRESS(i)=1 selects drive A, and AXIS_ADDRESS(i)=2 selects drive B.

Please do continuous selection in order, don't choose 2 first and then choose 1, don't choose 1 skip 2 and choose 3.

The controller must support EtherCAT to use related commands. It uses a set of program instructions with the RTEX bus, but the functions are different. For details, please refer to the description of the bus instructions chapter.

Related EtherCAT instructions:

Instruction	Meaning
SLOT_SCAN	Scan the devices
SLOT_START	Start the EtherCAT connection
SLOT_STOP	Stop the EtherCAT connection
ATYPE	Axis Type, when value is 65, it stands for EtherCAT period position control
AXIS_ADDRESS	Axis address configuration
WDOG	Total enable control of axes
SERVO_PERIOD	Refresh rate
AXIS_ENABLE	Axis enable
SDO_WRITE	SDO writing
SDO_READ	SDO reading
SDO_WRITE_AXIS	SDO writing of drives
SDO_READ_AXIS	SDO reading of drives
?*ETHERCAT	Bus information output
NODE_COUNT	Devices quantity
NODE_STATUS	Devices status
NODE_AXIS_COUNT	Drive quantity the devices contain
NODE_IO	IO serial number setting of devices

NODE_AIO	AIO serial number setting of devices
NODE_INFO	Get devices information
NODE_PROFILE	Set profile of device, choose PDO message.
DRIVE_FE	Error of drive
DRIVE_FE_LIMIT	Set error of drive
DRIVE_STATUS	Status of drive
DRIVE_TORQUE	feedback of drive torque
DRIVE_MODE	motion mode of the drive
DRIVE_PROFILE	Set profile of drive, choose PDO message.
DRIVE_CONTROLWORD	Control word of drive
DRIVE_CW_MODE	operation mode of drive control word.
DRIVE_IO	serial number setting of remote IO
AXISSTATUS	Axis status

The EtherCAT configuration matches the actual connection mechanism:

After the master station initialized the slave station, no matter whether the number of slave stations configured in the software is consistent with the actual connection of the EtherCAT communication port, the successfully configured slave station can be controlled by the master station. If two EtherCAT slave stations are configured in the software, and one is actually connected, the connected one can be controlled by motion commands. After the master station and the slave station are connected, even if another slave station configured in the software is connected to the network, the master station also does not establish a connection with the slave station that is later connected to the network.

EtherCAT slave disconnection and recovery mechanism:

After the EtherCAT slave station is connected to the master station, if the communication of some EtherCAT slave stations is dropped due to external reasons such as the unplugging of the communication cable, the master station will not re-establish the connection with the dropped slave station, and the dropped slave station can't be controlled through motion instructions. Still connected EtherCAT slaves are not affected. If it is unable to communicate with the bus due to internal reasons such as drive error, check whether the error can be cleared. After clearing, re-enable it and use it. If it cannot be cleared, power off and re-execute the bus initialization process.

If the disconnected slave needs to reconnect to the master, unplug the EtherCAT cable between the controller and the first servo drive and plug it in again, or power on the controller again. If the above operation is performed, the normal running of slave station will be affected. And normal running slave station will re-establish the connection with the master station. If any axis is running, it will cause the axis to stop immediately.

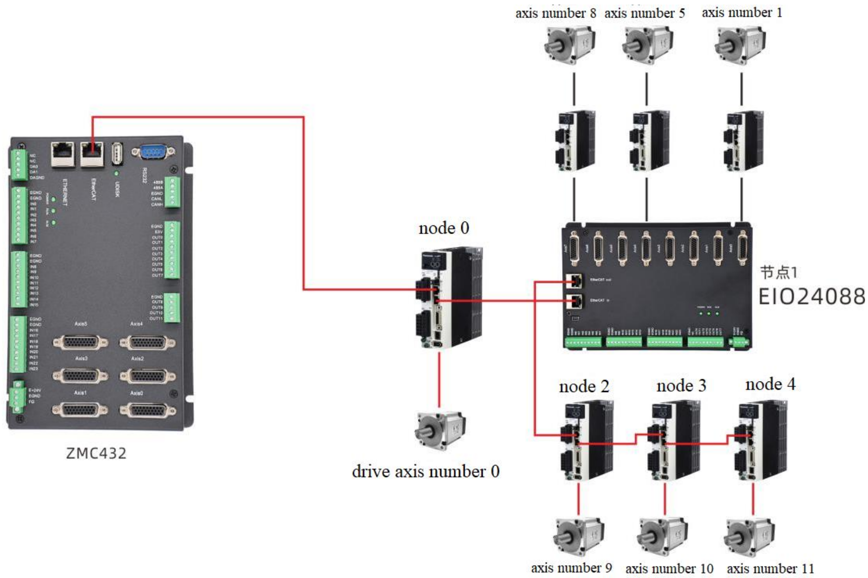
4.5.2 EtherCAT Bus and Drive Communication

The rule for wiring of servo drive is the same as the controller or the EtherCAT expansion module, using a network cable to connect the EtherCAT bus port of the controller to the EtherCAT port of other devices.

Note that there are two EtherCAT ports on the servo drive. Some drives can be connected to these two ports at will, and some are divided into EtherCAT IN and EtherCAT OUT. The IN port is connected to the upper-level device and the OUT port is connected to the next-level device. The two cannot be mixed. Pay attention to the connection order.

In multi-axis control, the EtherCAT OUT port of the servo drive is connected to the EtherCAT IN port of the next-level drive device, and so on.

The wiring configuration of the EtherCAT bus is as follows:



EtherCAT bus slot number is 0 by default.

The device number (node), also known as the node, refers to the number of all devices connected to a slot, starting from 0, and automatically numbered according to the connection order of the devices on the bus.

The controller will identify the drive on the slot, starting from 0, and it makes number automatically according to connection sequence of drive on the bus. The drive number is different from device number, it only makes number for device on the slots, ignoring others.

The rule for making numbers of EtherCAT bus and RTEX bus are the same.

The motor connected to EtherCAT bus needs to write one EtherCAT bus initialization program to enable. After enabled, if it is ATYPE=65 position mode, the usage is the same as pulse motor, including motion instructions. If it is ATYPE=66 speed mode or ATYPE=67 torque mode, now motion instructions can’t be used, but DAC command can be used to control axis continuous motion, when DAC=0, it will stop.

4.5.3 EtherCAT Bus Connect to Expansion Module

EtherCAT expansion module can expand IO and pulse axis. And the wiring rule is the same as EtherCAT Drive, wiring refers to the former content. Please note that EtherCAT IN and EtherCAT OUT on expansion module can’t be mixed when wiring.

After the wiring is completed as required, first initialize the EtherCAT expansion module. During the initialization process, it is necessary to map the extended IO and extended pulse axis resources before they can be used. The extended resource mapping is performed after the bus scan and before the bus is turned on according to the following method.

The IO mapping on the EtherCAT bus is set through NODE_IO instruction (digital quantity) and NODE_AIO instruction (analog quantity), and the axis mapping uses the AXIS_ADDRESS instruction to map the axis number.

Expanded resource mapping method:

1. IO mapping

The slot number and node device number are automatically numbered from 0 according to the sequence of connection with the controller.

The NODE_IO instruction sets the starting number of the digital IO of the device, and the starting number of the input and output of a single device is the same. It is set after bus scan, and the NODE_AIO instruction is basically the same as the NODE_IO instruction.

Grammar:

NODE_IO(slot, node)=iobase

slot: the slot number, 0 means default number.

node: the device number, starts from 0

ioBASE: the starting number of mapping IO, and the result is only the multiple of 8.

NODE_AIO(slot, node[,idir])=aiobase

slot: the slot number, 0 means default number.

node: the device number, starts from 0

idir: AD/DA selection. O means default, AIN and AOUT are set simultaneously, only AIN is read. 3-AIN, 4-AOUT.

IO mapping example:

```
SLOT_SCAN(0)                    'scan bus
IF NODEZ-COUNT(0)>0 THEN        'judge whether the slot number 0 has device
```

NODE_IO(0,0)=32 'set IO starting number of interface device 0 slot 0 is 32

NODE_AIO(0,1,3)=8 'set AIO starting number of interface device 1 slot 0 is 8

ENDIF

2. Axis mapping

The bus axis needs to perform axis mapping operation through AXIS_ADDRESS command. The operation method is as follows:

AXIS_ADDRESS(axis number)=(slot number<<16)+drive number+1

The axis map is written in the bus initialization routine, after scanning the bus and before turning on the bus.

Example:

AXIS_ADDRESS (0)=(0<<16)+0+1 'The first ECAT drive, drive number 0, bound as axis 0

AXIS_ADDRESS (2)=(0<<16)+1+1 'The second ECAT drive, drive number 1, bound as axis 2

AXIS_ADDRESS (1)=(0<<16)+2+1 'The third ECAT drive, drive number 2, bound as axis 1

ATYPE(0)=65 'Set to ECAT axis type, 65-position 66-speed 67-torque ATYPE(1)=65

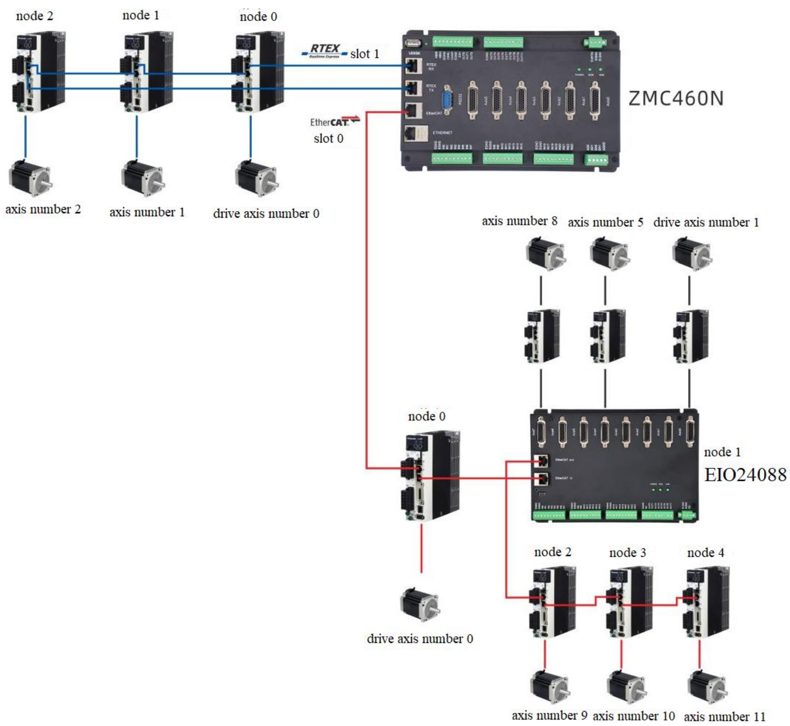
ATYPE(2)=65

4.6 RTEX Bus Communication

RTEX is a high-end bus technology independently developed by Panasonic to meet the high-speed real-time requirements of motion control. By simplifying data communication packets, high-speed communication can be achieved with a speed of 100Mbps. Only supports connecting to Panasonic drives. Before using it, a bus initialization program should be written to enable it.

The controller RTEX bus communication has two interfaces, namely RX and TX. When wiring, controller RX——driver TX, controller TX——driver RX.

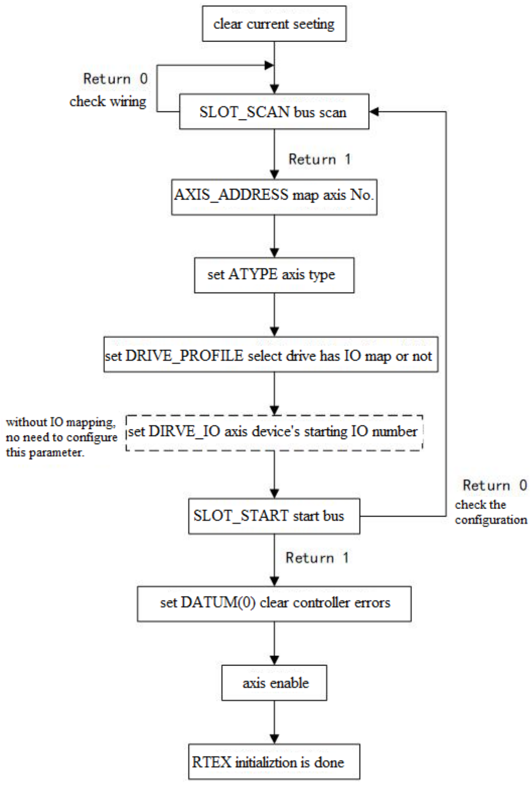
RTEX wiring method refers to below figure:



The control word of the RTEX driver will be set automatically, please set DRIVE_CW_MODE first if needs to set it manually.

The initialization process is completed by the controller, which means users don't need to operate, and the program starts to execute after power-on.

The general process of the initialization program: see the routine for the initialization program



1. Use SLOT_SCAN to scan the device to determine whether the RETURN is correct, and an error will be reported when the device is not connected.

2. The device type and information are judged by NODE_INFO/ NODE_AXIS_COUNT, etc.

3. Set AIXS_ADDRESS, ATYPE, DRIVE_PROFILE, DRIVE_IO, etc. in turn.

4. SLOT_START start the device.

5. After the connection is established, the master and slave can exchange cyclic data.

Basic concepts involved:

1. NODE node number: arranged according to the wiring sequence of RTEX devices, the number starts from 0 and decreases by 1 for the number of devices.

2. drive number: according to the wiring sequence of RTEX devices, the number starts from 0 to the number of RTEX drives minus 1. It only works when AXIS_ADDRESS is configured, that is, only the drive device is counted, other extended IO and other devices are not counted in the drive number.

3. axis number: the number set by the controller to the drive motor device connected to the controller. The number starts from 0 to the total number of connected devices minus 1. The axis number can be mapped to any connected drive device through the AXIS_ADDRESS command (pulse type controllers do not support local axis number mapping).

The controller must support RTEX to use related commands. It uses a set of program instructions with the EtherCAT bus, but the functions are different. For details, please refer to the description of the bus instructions chapter.

Related RTEX instructions:

Instruction	Meaning
SLOT_SCAN	Scan the devices
SLOT_START	Start the EtherCAT connection;
SLOT_STOP	Stop the EtherCAT connection;
ATYPE	Axis Type, when value is 50, it stands for position control in period(RTEX)
AXIS_ADDRESS	Axis address configuration
WDOG	Total enable control of axes
SERVO_PERIOD	Refresh rate
AXIS_ENABLE	Axis enable
?*Rtex	RTEX information output
NODE_COUNT	Devices quantity
NODE_STATUS	Devices status
NODE_AXIS_COUNT	Drive quantity the devices contain
NODE_IO	IO serial number setting of devices
NODE_AIO	AIO serial number setting of devices

NODE_INFO	Get devices information
DRIVE_STATUS	Status of drive
DRIVE_TORQUE	feedback of drive torque
DRIVE_PROFILE	Set profile of drive, choose PDO message.
DRIVE_CONTROLWORD	Drive control word
DRIVE_CW_MODE	Control mode of drive control word
DRIVE_IO	serial number setting of remote IO
DRIVE_CLEAR	Alarm of drive error clear, if no error, the controller will warn.
DRIVE_READ	parameters of drive writing
DRIVE_WRITE	parameters of drive reading
AXISSTATUS	Axis status

RTEX slave drop and recovery mechanism is the same as EtherCAT, please refer to the last description about EtherCAT.

Chapter V Motion Control Function

5.1 Common Motion Mode

There are three common motion modes:

- 1. Jog motion: only the end position is required, and it has nothing to do with the intermediate process of the movement, that is, the movement trajectory. The positioning speed is required to be fast, and different acceleration and deceleration control strategies are used in the acceleration section and deceleration section of the movement, which are divided into three categories: JOG jogging, MOVE inching motion and VMOVE continuous motion.
- 2. Continuous trajectory motion: this control is also called interpolation. In the case of high-speed motion, the system must not only ensure the contour accuracy of the system processing, but also ensure that the movement speed of the axis is not affected. When processing small line segments, there is a preprocessing function of trajectory look-ahead. For the description of interpolation motion, please refer to the next section.
- 3. Synchronous motion: it refers to the coordinated motion control of multiple axes, which can be synchronized during the entire motion of multiple axes, or local speed synchronization during the motion process. It is mainly used in system control with electronic gear and electronic cam functions. In the industry, there are printing and dyeing, printing, papermaking, steel rolling, synchronous shearing and other industries.

5.1.1 Single-axis Jog Motion

Related instructions:

Instruction	Description	Usage
VMOVE	Continuous motion, positive or negative direction	VMOVE(motion direction)
CANCEL	Single-axis stops, 4 kins of stop mode	CANCEL(mode)
JOGSPEED	The motion speed when JOG	JOGSPEED=speed value
FWD_JOG	Positive JOG input relative input number	FWD_JOG=input number
REV_JOG	Negative JOG input relative input number	REV_JOG=input number
FHSPEED	Keep the speed when FHOLD_IN is buttoned	FHSPEED=speed value
FHOLD_IN	Keep inputting relative number	FHOLD_IN=input number
FAST_JOG	Jog fast, input number	FAST_JOG=input number

VMOVE is a single-axis continuous motion command. After VMOVE is executed, it will keep running unless CANCEL or RAPIDSTOP is used to clear the motion buffer. Otherwise, it will operate all the time.

When the preceding VMOVE motion does not stop, the following VMOVE instruction will automatically replace the preceding VMOVE instruction and modify the direction, so there is no need to CANCEL the preceding VMOVE instruction.

In JOG motion mode, each axis can independently set target speed, acceleration, deceleration, speed smoothing and other motion parameters, and can move or stop independently.

The JOG movement is controlled by the switch signal, and it can move in the positive and negative directions. The FWD_JOG command maps the positive jog switch, and the REV_JOG command maps the negative jog switch. When the controller receives the signal input, the relative axis will move slowly according to the JOGSPEED speed. The axis decelerates and stops when the signal input is interrupted. When continuous JOG movement is required, keep the input state of the switch.

The controller also supports fast jog. The FAST_JOG instruction sets the fast jog switch. When the fast jog switch is pressed, the axis moves with SPEED. When the switch is not pressed, the axis moves with JOGSPEED.

The FHOLD_IN mapping keeps the input setting. When there is an input signal, the speed of the motion axis continues to execute the current motion according to the speed parameter of FHSPEED. When FHOLD_IN cancels the input, the axis continues to move, but the motion speed changes back to the SPEED speed.

When REV_JOG and FWD_JOG both have signal input at the same time, the axis runs in the forward direction according to FWD_JOG.

The MOVE instruction controls the inching motion of the axis, sets the distance of inching motion, and sends a limited number of pulses to the target axis. It can be used in single-axis or multi-axis motion occasions. It can send pulses to multiple axes at one time. The controller for jog motion only can control each axis to move independently, cannot interpolate jointly.

Single axis jog example 1: VMOVE continuous motion

```
RAPIDSTOP(2)
WAIT IDLE(0)

BASE(0)                  'select axis number
ATYPE=1                  'axis type configuration
UNITS=100                'pulse equivalent configuration
SPEED=100                'speed configuration
ACCEL=1000               'acceleration configuration
DECEL=1000               'deceleration configuration
```

```
SRAMP=100                'S curve
DPOS=0                  'the current position clear out
TRIGGER

WHILE 1                  'cycle motion
    IF MODBUS_BIT(0) = ON THEN      'MODBUS_BIT(0) valid motion to left
        VMOVE(-1)
    ELSEIF MODBUS_BIT(1) = ON THEN  'MODBUS_BIT(1) valid motion to right
        VMOVE(1)
    ELSEIF MODBUS_BIT(0) = OFF OR MODBUS_BIT(1) = OFF THEN
        CANCEL               'MODBUS_BIT is invalid, stop motion ENDIF
WEND
```

Single axis jog example 2: JOG jog motion

Note that after mapping the JOG switch, the level of INVERT_IN must be reversed, because the OFF signal is valid for the ZMC series controller. If it is not reversed, the signal will be OFF when it is connected. The controller judges that there is an input and immediately controls the axis movement.

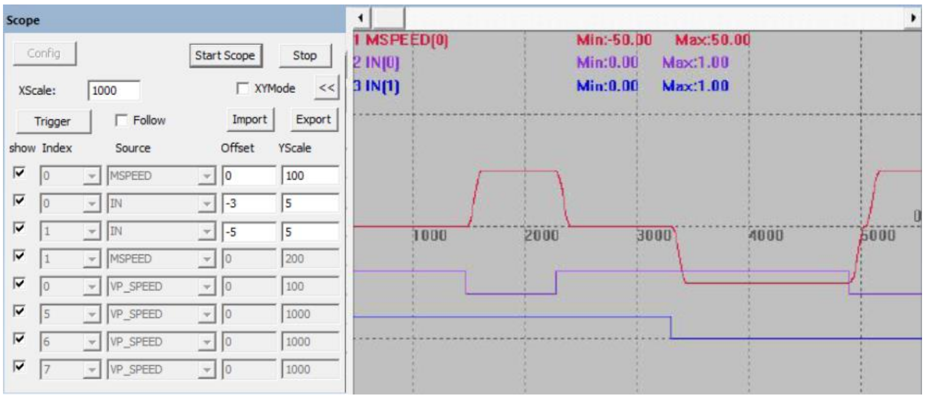
```
BASE(0)                 'select axis 0
ATYPE=1                 'pulse axis type
DPOS=0                  'the coordinate clears out
UNITS=100               'pulse equivalent
SPEED=100               'main axis speed
ACCEL=1000              'acceleration
DECEL=1000              'deceleration
SRAMP=100               'S curve
TRIGGER                 'trigger oscilloscope automatically
JOGSPEED=50             'JOG speed is 50
FWD_JOG=0               'IN0 is the positive JOG switch
REV_JOG=1               'IN1 is the negative JOG switch
INVERT_IN(0,ON)         'input 0 signal invert
INVERT_IN(1,ON)         'input 1 signal invert
```

Operation effect:

→ When input 0 has signal input, axis 0 moves in positive direction, the speed is 50.

→ When input 1 has signal input, axis 0 moves in negative direction, the speed is 50.

→ When input 0 and input 1 both have signal input, axis 0 moves in positive direction.

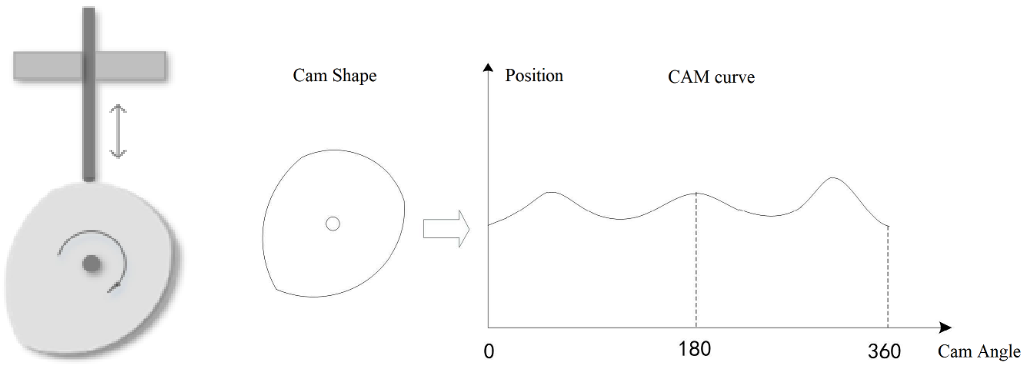


5.1.2 Electronic CAM

Related instructions:

Instruction	Description	Usage
CAM	Cam watch moves	CAM(start and end position of CAM watch, proportion, motion distance)
CAMBOX	Follow cam watch	Refer to instruction chapter
TABLE	Save cam watch data	TABLE(data starting address, data area)
MOVELINK	Automatic cam	Define reference axis, follow axis and synchronous motion mode
MOVESLINK	Automatic cam 2	

The mechanical cam is mainly composed of an active part and a driven part, which converts the rotary motion into a linear motion. The basic structure is as shown in the left figure below. The active part is the contour curve processed on the metal plate, which generally rotates at a uniform speed; the driven part is generally in contact with the contour of the cam. When the active part is in motion, the driven part reciprocates, and the motion trajectory is determined by the cam contour.



Because the mechanical cam mechanism is a high-level mechanism, there is point or line contact between the cam and the driven part, which is inconvenient for lubrication, easy to wear, and noisy, and the processing and manufacturing requirements of the disc are relatively high, the maintenance is complicated. Therefore, in many applications, mechanical cams are replaced by electronic cams.

The electronic cam means users construct the cam curve, as shown in the right figure above, then the mechanical cam can obtain a rotation angle and the movement trajectory of the processing position according to the contour of the cam. This trajectory is an arc, and the arc is decomposed into countless linear or circular trajectory. The arc trajectory can be combined to obtain a series of motion trajectories that are close to the arc. Then electronic cam directly loads the motion parameters of this trajectory into the motion command, and can control the axis to walk out of the target trajectory to reach the relationship between the driving part and the driven part. The cam motion can be completed without the assistance of additional mechanical structure. The software is used to control the signal, and the motion curve can be changed by changing the relevant motion parameters of the program. This application is extremely flexible, reliable and easy to operate.

ZMotion controller provides multiple electronic cam motion form for users:

CAM: cam watch motion

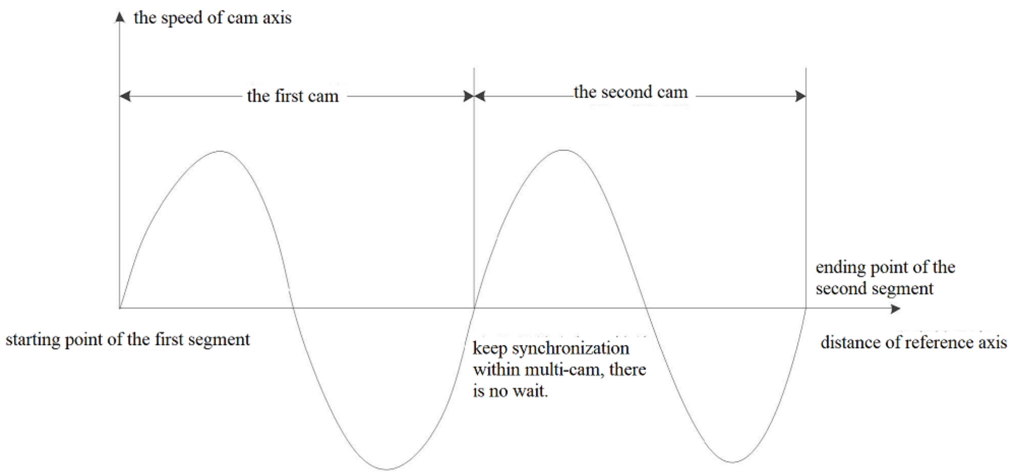
CAMBOX: follow cam watch motion

MOVELINK, MOVESLINK: Specific track cam motion, also known as flying shearing motion, which is generally used in flying shearing applications.

FLEXLINK: Specific track cam motion, also known as chasing shear motion, is generally used in chasing shear applications.

Motion pause is not supported during the cam motion, and the cam instruction will stop after the execution of the cam instruction. Use the CANCEL or RAPIDSTOP instruction to forcibly cancel it.

After the multi-segment cam command is stored in the motion buffer, the next cam will start immediately after the previous cam is completed to ensure synchronization.



If the cam command is not stored in the motion buffer, there will be a waiting time between multi-segment cams, and the motion will not be continuous. See: CAM, CAMBOX.

5.1.3 Electronic Gear

Related instructions:

Instruction	Description	Usage
CONNECT	connection axis motion	CONNECT (rate, axis number of connection axis)
CONNPATH	connection axis motion	CONNPATH (rate, axis number of connection axis)
CANCEL	Cancel connection	It can be used directly without parameters

Gear is different from the cam, the connection of the electronic gear is linear. And the electronic gear function is used for the connection of two axes. The master axis and the slave axis are connected according to a constant gear ratio. No physical gear is required. Set electronic gear ratio directly through the instruction. Since it is implemented in software, the electronic gear ratio can be changed at any time, and one master can drive multiple slave axes.

The electronic gear function is realized by the commands CONNECT and CONNPATH. One axis is connected to the other axis according to a certain ratio for follow-up motion. One motion command can drive the operation of two motors, then test these two motor-axis' movement numbers. Feedback the displacement deviation to the controller and obtain synchronous compensation, so that the displacement deviation between the two axes can be controlled within the allowable range of accuracy.

The electronic gear is connected to the number of pulses. For example, the master-slave axis connection ratio is 1:5, and one pulse is sent to the master axis. At this time, 5 pulses are sent to the slave axis.

The role of electronic gear:

1. Pulse compensation to reduce the burden on the upper computer (because the components currently used to send pulses have limitations on the frequency of sending pulses).

2. Matching the number of pulses sent by the motor and the minimum movement of the machine, the movement of the workpiece (or motor) corresponding to one pulse of command input can be set to any value. It can realize the stepless speed change of the motor. When the motor starts and stops, it can prevent the motor from losing step and overshooting, so that the potential of the motor can be fully utilized.

3. It transmits synchronous motion information, realizes the linkage of coordinates, the transformation between motion forms (rotation-rotation, rotation-straight line, straight line-straight line), simplified control, etc.

The same point between CONNPATH and CONNECT:

the syntax of the two is the same, the number of pulses is connected, and the effect of connecting CONNPATH to the motion of a single axis is the same as that of CONNECT.

Difference between CONNPATH and CONNECT:

CONNECT connects the target position of a single axis. CONNPATH connects the vector length of the interpolated axis. At this time, it needs to be connected to the main axis of the interpolated motion, and it cannot follow the interpolated motion when connected to the interpolated slave axis. CONNPATH tracks XY interpolated vector length changes, not individual X or Y axes.

5.1.4 Handwheel

Related instructions:

Instruction	Description	Usage
CONNECT	Connect to handwheel	CONNECT (rate, axis number of connection axis)
CANCEL	Cancel connection	It can be used directly without parameters

Handwheel is also called pulse generator, hand pulse, hand pulse generator, etc. It belongs to a kind of encoder and is used for zero correction and signal division of CNC machine tools, printing machinery, etc. When the handwheel rotates, the encoder generates a signal corresponding to the movement of the handwheel, selects the coordinates and locates the coordinates.

The handwheel function refers to using a specific encoder as the handwheel pulse change input source to detect the encoder pulse input change, use the CONNECT command to establish the connection between the handwheel axis and the follow axis, and drive the handwheel to follow the axis. This function is mainly used for auxiliary motion in interpolation motion. The encoder can be the encoder on the terminal board or the encoder module on the EtherCAT bus.

The handwheel following movement belongs to the position-following type, that is, the handwheel pulse changes n, and the following axis follows n*ratio pulses, and the speed and acceleration are planned according to the parameters of the main axis.

Conditions for entering the handwheel motion:

the axis following the handwheel is in a static state means there is no movement. The encoder is in an unbound axis means it can be used for axis position feedback. The following axis is in the enabled state. The following axis is not set to handwheel mode.

To exit handwheel mode by using CANCEL command. The handwheel connection ratio can be switched at any time.

Use process:

Set the type of handwheel axis and follow axis, set various basic motion parameters, use the

CONNECT command to connect the handwheel and follow axis according to a certain ratio, then the handwheel can drive the follow axis to move, and CANCEL cancels the handwheel connection after the movement is completed.

Routine: Follow handwheel motion

```
RAPIDSTOP(2)
WAIT IDLE(0)
ERRSWITCH = 3
CONST axishand = 0
BASE(axishand)              'select axis 0 connect to handwheel
ATYPE=6     'pulse + directional handwheel, for quadrature input handwheel, using 3
BASE(1)              'axis 1 is controlled by handwheel
ATYPE=1              'configure as pulse axis
DPOS = 0,0
UNITS = 100,100              'pulse equivalent, 100 pulses per mm100
SPEED = 200,200
ACCEL = 2000,2000
DECEL = 2000,2000
SRAMP = 20
CLUTCH_RATE = 0              'use speed and acceleration to limit
DIM poslast
poslast = DPOS
WHILE 1
    IF IN(0) = ON AND IN(1) = OFF THEN
      CONNECT(1, axishand)     'link to axis 0, the ratio is 1
    ELSEIF IN(1) = ON AND IN(0) = OFF THEN
        CONNECT(1, axishand)     'link to axis 0, the ratio is 10
    ELSEIF IN(0)= ON AND IN(1) = ON THEN
        CONNECT(50, axishand)     'link to axis 0, the ratio is 50, for step motor, if the
                                   ration is too high, it will lose steps or it ends for a
                                   long time.
    ELSEIF MTYPE = 21 THEN
      CANCEL              'cancel CONNECT
    ENDIF
    IF poslast <> DPOS THEN
        poslast = DPOS
```

```
        TRACE DPOS
    ENDIF
WEND
END
```

5.2 Interpolation Motion

5.2.1 Concept of Interpolation

The interpolation is the process to determine the tool movement path by using machine tool CNC system according to certain methods. Interpolation is a real-time data densification process. No matter what kind of interpolation algorithm, the operation principle is basically the same. Its function is to perform digital calculation according to the given information, and continuously calculate the feeding instruction of each coordinate axis participating in the movement, and then drive their respective execution components to produce coordinated motions, so that the controlled mechanical components move according to the ideal route and speed.

The most common interpolations are linear interpolation and circular interpolation. The interpolation at least needs 2 axes, and when interpolating, first to build the coordinates, then map axis into corresponding coordinate, the motion controller controls each axis' motion to achieve required motion path according to the coordinate mapping relation.

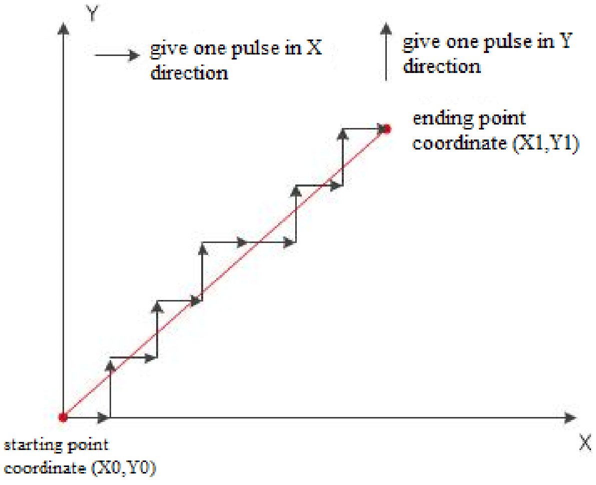
The interpolation motion instruction is stored into motion buffer, then be taken out in turn from the buffer for executing, until all interpolations are executed.

- Linear Interpolation:

In the linear interpolation, the interpolation between two points closes to each other along the group point of line. Firstly, assume the real contour starting point moves a short distance in X direction (give one pulse equivalent, the axis will move a fixed distance). Then, find the terminal point is below the real contour, the next segment will move s short distance in Y direction, if the terminal is also below now, continuing to move in Y, it will move s short distance in X until it is above the real contour. The motion cycles like this, until it arrives the contour terminal point. In this way, the actual contour is formed by splicing a segment of polylines. Although it is a polyline, each segment of the interpolation segment is very small within the allowable accuracy range, so this segment of the polyline can still be approximately regarded as a straight line, which is linear interpolation.

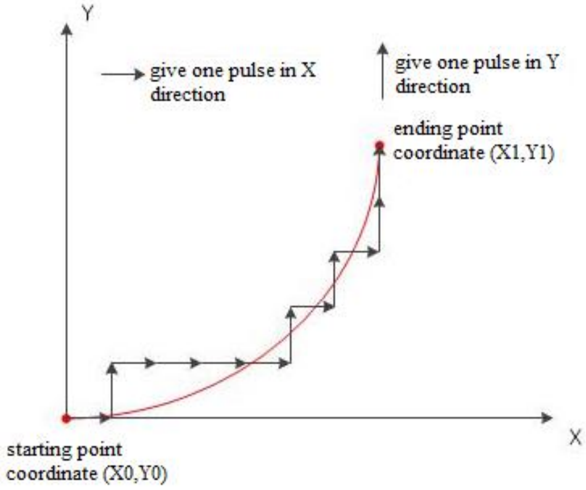
ZMotion controller uses hardware interpolation, and the interpolation precision is within one pulse, so it is still smooth when the path is amplified.

If the axis needs to move from point (X0,Y0) to point (X1,Y1) in the XY plane, the processing process of linear interpolation is like the below figure:



➢ Circular Interpolation

Circular interpolation is similar to linear interpolation. The interpolation digital information between the two ends is given, and the point group approximating the actual arc is calculated by a certain algorithm. The control axis moves along these points to process arc curves. Circular interpolation can be a plane arc (at least two axes) or a space arc (at least three axes). Assuming that the axis needs to travel an inverse arc in the first quadrant of the XY plane, the center of the circle is the starting point, and the processing process of the arc interpolation is shown in the figure below.



The space arc interpolation function of the controller is based on the current point and the end point and the middle point (or center point) set by the arc command parameters to determine the circular arc and realize the spatial circular arc interpolation movement. The coordinates are three-dimensional coordinates, and at least three axes are required to move along the X axis, the Y axis and the Z axis respectively.

- Interpolation mode of motion controller

Motion controller's interpolation motion mode has below functions:

1) it can achieve interpolation of linear, circular, space arc, helical, ellipse, etc.

2) it can do multi-axis interpolation in several coordinate systems multiple channels, single channel only supports at most 16 axes combined interpolation.

3) each axis has motion buffer, which can be used to achieve motion pause, resume and other functions. And the interpolation motion of one axis stops, other axes also stop.

4) it has delay in buffer and synchronously output digital in buffer functions.

5) it has the pre-process function, the controller analyzes and calculates target trajectory automatically, so that high speed and smooth continuous motion in small segment can be achieved.

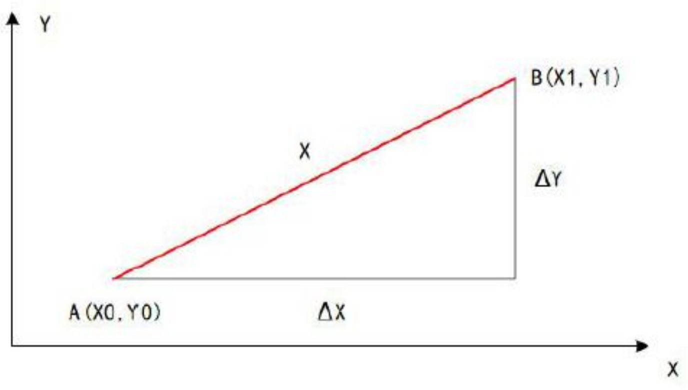
- Two-axis linear interpolation

Axis 0 and axis 1 participate in the linear interpolation motion, as shown in the figure below. The linear interpolation motion of these 2 axes moves from point A to point B, the XY axes start at the same time, and reach the end point at the same time. Set the motion distance of axis 0 as ΔX, and the motion distance of axis 1 is ΔY, the main axis is the first axis of BASE (now the master axis is axis 0), the motion speed of the main axis interpolation is S (the set speed of the main axis), and the actual speed of each axis is the sub-speed of the main axis, but it is not equal to S, at this time:

Main axis' motion distance: X=[(ΔX)2+(ΔY)2]½

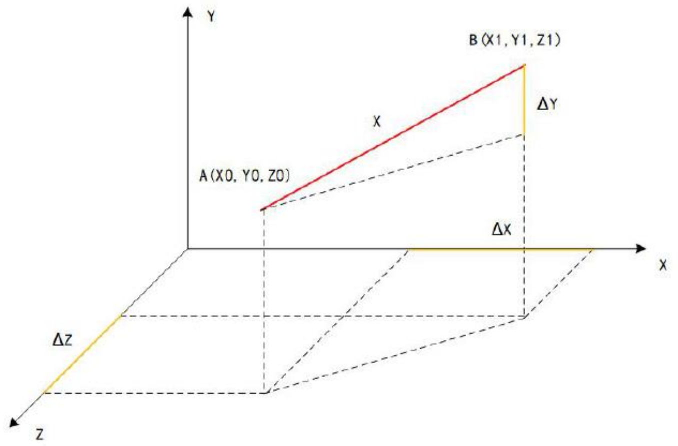
Axis 0 actual speed: S0=S×ΔX/X

Axis 1 actual speed: S1=S×ΔX/X



- Three-axis linear interpolation

Axis 0, axis 1 and axis 2 participate in the linear interpolation motion, as shown in the figure below. The linear interpolation motion of these 3 axes moves from point A to point B, the XYZ axes start at the same time, and reach the end point at the same time. Set the motion distance of axis 0 as ΔX, and the motion distance of axis 1 is ΔY, the main axis is the first axis of BASE (now the master axis is axis 0), the motion speed of the main axis interpolation is S (the set speed of the main axis), and the actual speed of each axis is the sub-speed of the main axis, but it is not equal to S, at this time:



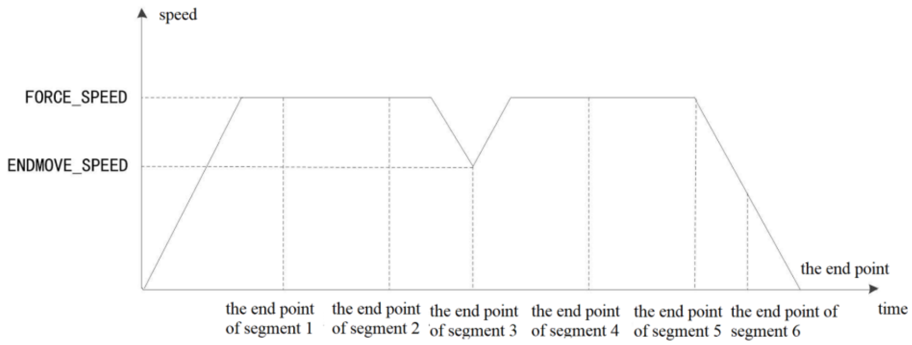
- Multi-axis linear interpolation

Multi-axis linear interpolation can be understood as multiple degrees of freedom of an axis, which is linear interpolation in three-dimensional space. Taking four-axis interpolation as an example, generally three axes run a straight line on the XYZ plane, and the other axis is a rotation axis, which does follow motion with a certain proportional relationship.

5.2.2 Continuous Interpolation

If the MERGE continuous interpolation is not turned on, after the previous interpolation movement is completed, when executing the next interpolation, it will first decelerate to stop, and then re-accelerate to execute the interpolation movement. In actual application, this situation will lead to low processing efficiency, so it is necessary not to use deceleration between consecutive interpolation movements, which is the continuous interpolation function.

To make the interpolation action continuous, after setting MERGE=ON, the interpolation motion of the same main axis will be automatically continuous, and there will be no deceleration between two consecutive motions, and the SP instruction can manually set the motion speed and end speed. Refer to some instructions, MERGE, SP, CORNER_MODE, ENDMOVE_SPEED, FORCE_SPEED, etc.



5.3 Look-ahead processing

Related instructions:

Instruction	Description	Usage
CONNER_MODE	Corner mode configuration	COERNER_MODE=mode value
MERGE	Continuous interpolation	MERGE=ON
DECEL_ANGLE	Start deceleration angle	When using,
STOP_ANGLE	Stop deceleration angle	
ZSMOOTH	Chamfer radius	ZSMOOTH=chamfer radius value
FULL_SP_RADIUS	Speed limit radius	
FORCE_SPEED	Force speed	

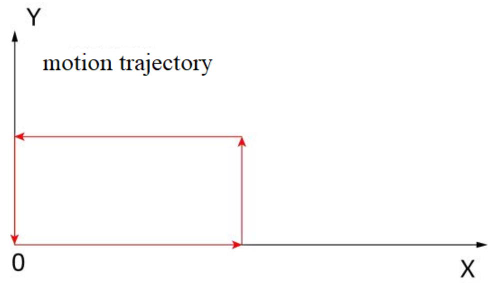
In the actual processing process, continuous interpolation will be turned on for pursing processing efficiency. If the corner of the motion trajectory is not decelerated, and when the corner is large, it will cause a greater impact on the machine and affect the machining accuracy. If the continuous interpolation is turned off and the deceleration at the corner is 0, although the machine is protected, the machining efficiency is greatly affected, so a look-ahead command is provided to automatically determine whether to reduce the corner speed to a reasonable value at the corner, which will not affect the processing accuracy but also improve the processing speed. This is the role of the trajectory look-ahead function.

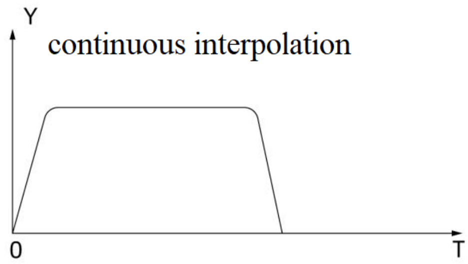
The trajectory look-ahead of the motion controller can automatically calculate a smooth speed plan according to the user's motion path, reduce the impact of the machine, and improve the machining accuracy. The inflection point will appear when automatically analyzing the command trajectory of the motion buffer, and automatically calculate the motion speed at the corner according to the corner conditions set by the user, and also calculate the speed plan according to the maximum acceleration value set by the user, so that acceleration and deceleration value in any acceleration and deceleration process in the machine do not exceed ACCEL and DECEL, so as to prevent the damage to the mechanical part.

Speed planning situation with trajectory look-ahead or without trajectory look-ahead:

If the motion trajectory is like the left figure, it moves a rectangle trajectory, and there are 4 linear interpolation motions.

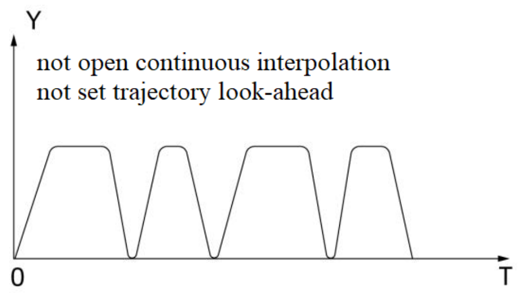
Mode 1: after opening continuous interpolation, obtained the speed of main axis changes with time, please see the right figure. The speed of main axis is consecutive, and it doesn't decelerate at the corner position. The corner has a big shock when running in high-speed.

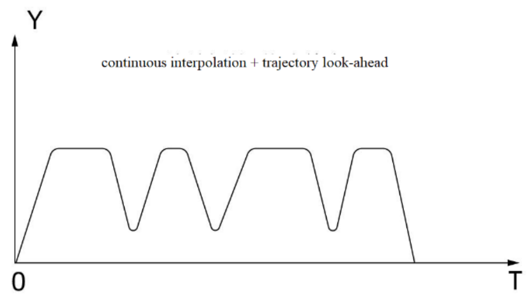




Mode 2: under the condition of mode 1, close continuous interpolation, obtained changing curve of main axis speed with time is the below left figure. A straight line is completed, it will decelerate to 0, then start the second straight motion, therefore the processing efficiency is not high.

Mode 3: under the condition of mode1, open continuous interpolation, and set trajectory look-ahead parameters, obtained changing curve of main axis speed with time is the below right figure. The corner position decelerates according to one certain proportion, the processing efficiency is higher than mode 2.





For above modes, speed is set as S curve through SPAMP instruction, so that the speed curve will be softer.

The main command CORNER_MODE of trajectory look-ahead is used for speed planning at corners. There are three commonly used modes, and different modes are selected according to the actual requirements of the processed trajectory.

This parameter takes effect before the interpolation motion command is called. Generally, the corner mode is set in the parameter initialization. Because the look-ahead motion parameters varying with the motion commands are stored in the motion buffer together. The look-ahead motion parameters can be called multiple times, and different modes can also be mixed, such as CONNER_MODE=2+8, which means automatic corner deceleration and small circle speed limit are used at the same time, set the appropriate look-ahead mode according to the requirements of the trajectory segment, and automatically optimize the trajectory when executing the motion command.

Note: Once the CORNER_MODE mode is set, the parameters will be stored in the controller. Set CORNER_MODE=0 to cancel it. Otherwise, CORNER_MODE set before will take effect.

CORNER_MODE instruction parameter description:

Bit	Value	Description
0	1	Reserve
1	2	Decelerate automatically at the corner position. Accelerate and decelerate as ACCEL and DECEL. This parameter takes effect before calling MOVE function. The deceleration angle is set through DECEL_ANGLE and STOP_ANGLE instructions. The reference speed of deceleration corner refers to FORCE_SPEED, so please set reasonable FORCE_SPEED.
2	4	Reserve
3	8	Automatic small circle speed limit, speed limit when the radius is less than the set value, no speed limit when the radius is greater than the limit value. This parameter is modified before the MOVE function is called. The speed limit is related to FORCE_SPEED. Limit speed = FORCE_SPEED * actual radius/FULL_SP_RADIUS Speed limit radius FULL_SP_RADIUS setting.
4	16	Reserve
5	32	Automatic chamfer settings. This parameter is modified before the MOVE function is called. This MOVE motion is automatically chamfered with the previous MOVE motion, and the chamfer radius refers to ZSMOOTH.

COR_MODE=2 corner deceleration application: do not change motion trajectory, just automatically judge whether there is deceleration at the corner, which is usually used to improve shaking problem of machine. For those places need trajectory precision and no speed requirements.

CORNER_MODE=8 small circle speed limit application: do not change the motion trajectory, generally used in arc processing, calculate the current limit speed according to the radius of the arc.

CORNER_MODE=32 automatic chamfering application: change the motion trajectory, this will not slow down the speed. For the occasions with large track corners, the motion track at the chamfer is automatically smoothed, so it is generally used in the occasions where the speed is fast and the track accuracy is not high.

See the CONNER_MODE instruction for the application routine of the track look-ahead.

5.4 Origin Point Homing

Related instructions:

Instruction	Description	Usage
DATUM	Origin point homing mode selection	DATUM(homing mode value)
DATUM_IN	The switch to map the origin point	DATUM_IN=input number
FWD_IN	Mapping positive limit position switch	FWD_IN=input number
REV_IN	Mapping negative limit position switch	REV_IN=input number
SPEED	Fast speed for finding the origin point	Set the value of speed
CREEP	Inverse cramp speed for finding origin point	Set the value of speed
INVERT_IN	Input signal inverse	INVERT_IN= (input number, ON/OFF)

High-precision automation equipment has its own reference coordinate system. The movement of the workpiece can be defined as the movement on the coordinate system. The origin of the coordinate system is the starting position of the movement. All kinds of processing data are based on the origin as the reference point. Therefore, before starting the controller to execute the motion command, the device must perform the zero-returning operation to return to the origin of the set reference coordinate system. If it is not performed, the subsequent motion trajectory will be wrong.

Zmotion controller provides a variety of homing methods, which are set through the DATUM instruction. Different mode values can choose different homing methods. Each axis automatically returns to the origin according to the homing method set before. This command is a single-axis homing command. When multi-axis homing, the DATUM command needs to be used for each axis.

When returning to zero, the platform needs to be connected to the origin switch (indicating the position of the origin) and the positive and negative limit switches (both are sensors., after the sensor detects a signal, it indicates that there is an input signal, which will be sent to the controller for processing).

When single axis finds the origin, the origin switch is set by DATUM_IN, and the positive and negative limit switches are set by FWD_IN and REV_IN respectively. After the positive/negative limit signal of the controller takes effect, the axis will stop immediately, and the stop deceleration is FASTDEC.

When the ZMC motion controller is 0, the trigger is valid, and when the input is in the OFF state, it means that the origin/limit is reached. The normally open type signal needs to use the

INVERT_IN inversion level.

DATUM instruction supports below homing mode:

Value	Description
0	Clear error status of all axes.
1	The axis runs forward at CREEP speed until the Z signal appears. It will stop directly when it touches the limit switch. The DPOS value is reset to 0 and the MPOS is corrected.
2	The axis runs reversely at CREEP speed until the Z signal appears. It will stop directly when it touches the limit switch. The DPOS value is reset to 0 and the MPOS is corrected.
3	The axis runs forward at SPEED until it touches the home switch. Then, the axis reverses at CREEP speed until it leaves the home switch. In the process of finding the origin, it will stop directly when it encounters the positive limit switch. When the crawling stage encounters the negative limit switch, it will stop directly. DPOS value reset to 0 while correcting MPOS
4	The axis runs reversely at SPEED until it touches the home switch. Then, the axis runs forward at CREEP speed until it leaves the home switch. In the process of finding the origin, it will stop directly when it encounters the positive limit switch. When the crawling stage encounters the negative limit switch, it will stop directly. DPOS value reset to 0 while correcting MPOS
5	The axis runs forward at SPEED until it touches the home switch. Then, the axis reverses at CREEP speed until it leaves the home switch. In the process of finding the origin, it will stop directly when it encounters the positive limit switch. It will stop directly when it touches the limit switch. The DPOS value is reset to 0 and the MPOS is corrected.
6	The axis runs reversely at SPEED until it touches the home switch. Then, the axis runs forward at CREEP speed until it leaves the home switch. In the process of finding the origin, it will stop directly when it encounters the positive limit switch. It will stop directly when it touches the limit switch. The DPOS value is reset to 0 and the MPOS is corrected.
8	The axis runs forward at CREEP speed until touching the origin switch. It will stop directly when it touches the limit switch.
9	The axis runs reversely at CREEP speed until touching the origin switch. It will stop directly when it touches the limit switch.
21	Use the zero-return function of the EtherCAT drive, and now mode2 is valid. Set the drive's zero-return mode (6098h). The default value of 0 means to use the drive's current zero return mode. It will use the SPEED, CREEP, ACCEL, DECEL of the axis, multiply it by UNITS, and automatically set the 6099h, 609Ah action sequence of the drive: 6098h homing mode→6099h speed→609Ah acceleration→6060h switch the

	current mode

Z signal homing must be configured with Z signal ATYPE.

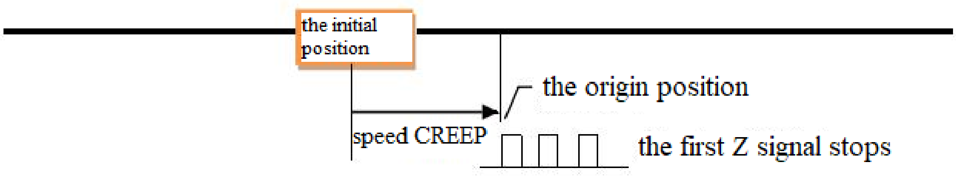
For the case the origin is in the middle of the positive and negative limits, add 10 to each mode, which means that the movement will not be canceled if the limit is encountered during the zero-return process, but will continue to search for the origin in the reverse direction. Other conditions are the same as the original mode. Since the origin is between the positive and negative limit switches, one limit switch only is met during homing. The following zero-return methods 5~8 all plus 10 modes.

After the bus controller uses the above controller to find the origin mode, it needs to manually reset the MPOS. Add 100 to zero return mode (modes 100+n and 110+n correspond to n and 10+n respectively), indicating that MPOS can be automatically cleared after connecting to the encoder (only for 4 series, ATYPE=4)

Detailed explanation of common zero return methods:

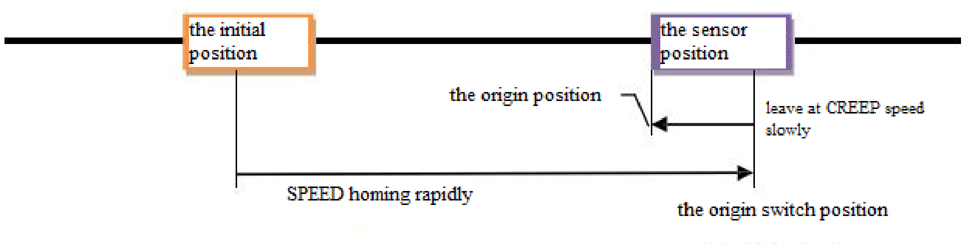
Mode 1: Z signal mode

The axis runs at CREEP speed until the Z signal appears. The DPOS value is automatically reset to 0 and the MPOS is corrected. It is only valid when ATYPE is set to 4 or 7 and the corresponding axis encoder Z is connected. It stops directly when it encounters the positive and negative limit switches on the way of returning to zero. When mode=1, it returns to zero in positive direction, and when mode=2, it returns to zero in negative direction.



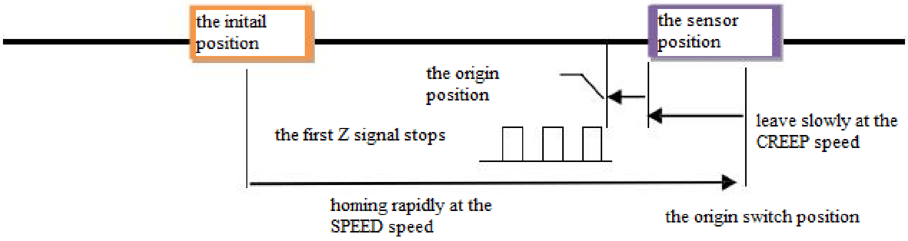
Mode 2: the origin + find inversely mode

The axis runs to the origin point at SPEED speed until touching the origin switch. Then, the axis runs inversely at CREEP speed until leaving the origin switch. The DPOS value is automatically reset to 0 and the MPOS is corrected. It stops directly when it encounters the positive and negative limit switches on the way of returning to zero. When mode=3, it returns to zero in positive direction, and when mode=4, it returns to zero in negative direction.



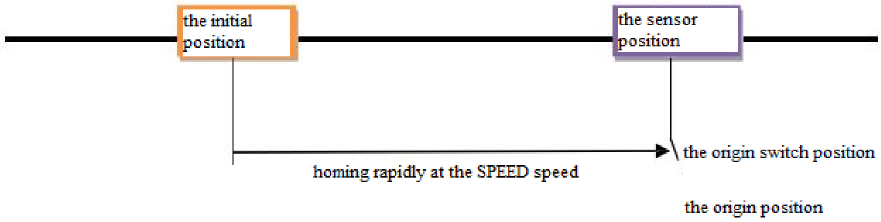
Mode 3: mode = 5, the origin + find inversely + Z signal mode

The axis travels towards the origin at SPEED until it touches the home switch. Then, the axis reverses at CREEP speed until it leaves the home switch, and continues to reverse at CREEP speed until it touches the Z signal. The DPOS value is automatically reset to 0 and the MPOS is corrected. It is only valid when ATYPE is set to 4 or 7 and the corresponding axis encoder Z is connected. It stops directly when it encounters the positive and negative limit switches on the way of returning to zero. mode=5 returns to zero in positive direction, mode=6 returns to zero in negative direction.



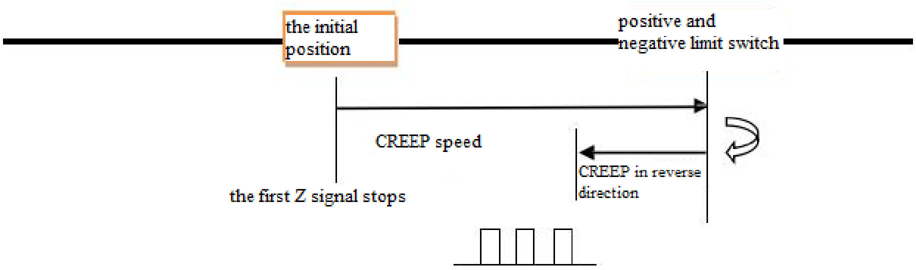
Mode 4: mode=8, the origin point returns to zero once mode

The axis runs at the speed of SPEED until it hits the origin switch. The DPOS value is automatically reset to 0 and the MPOS is corrected. It stops directly when encountering the positive and negative limit switches on the way to zero. mode=8 returns to zero in positive direction, mode=9 returns to zero in negative direction.



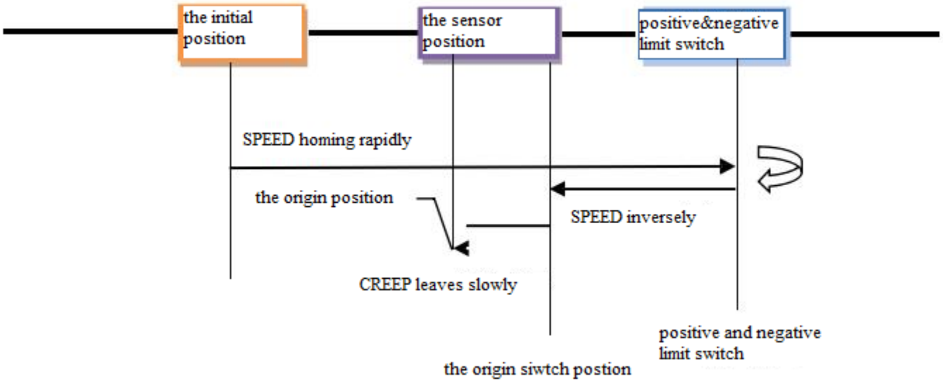
Mode 5: mode=11, Z signal mode,

The axis runs at the CREEP speed, it won’t stop when encountering the limit switch, and it will continue to run at the CREEP speed direction until the Z signal appears.



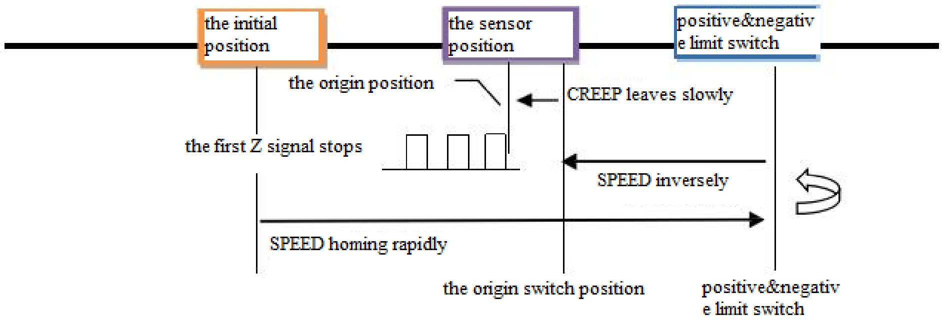
Mode 6: mode=13 forward running, origin + reverse search mode + limit reverse

The axis runs to the origin at SPEED speed, and it does not stop when meeting the forward limit switch, and then runs in reverse at SPEED speed until it hits the limit switch. Then, the axis moves slowly at CREEP speed until it leaves the origin switch.



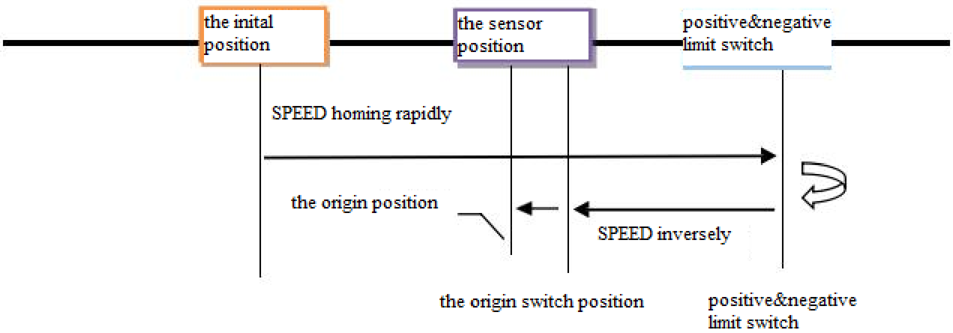
Mode 7: mode=15, origin + reverse search + Z signal mode

The axis runs to the origin at the speed of SPEED, and it does not stop when encountering the limit switch, but it continues to move in the reverse direction at the speed of SPEED until it hits the origin switch. Then, the axis moves to the origin at the speed of SPEED. The CREEP speed reverses movement until it leaves the home switch, and then continues to reverse at the creep speed until the Z signal is encountered.



Mode 8: mode=18, one-time homing mode

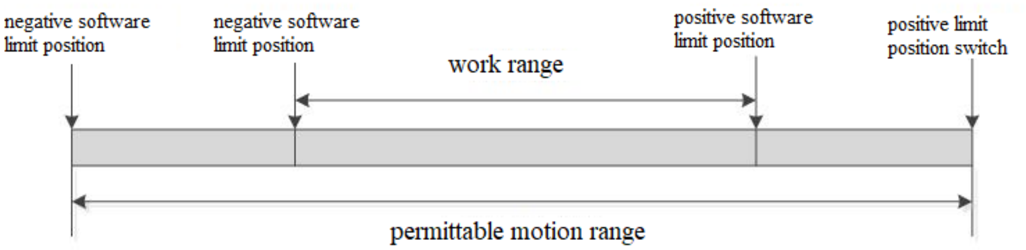
The axis runs at the speed of SPEED, and it does not stop when encountering the limit switch, but it continues to move in the reverse direction at the speed of SPEED. It stops when it hits the home switch.



5.5 Related Limit Position Instructions

Instruction	Description	Usage
FS_LIMIT	Positive software limit setting	FS_LIMIT = positive limit position
RS_LIMIT	Negative software limit setting	RS_LIMIT= positive limit position
FWD_IN	Mapping the positive limit input	FWD_IN = input number
REV_IN	Mapping the negative limit input	REV_IN = input number

The motion controller can limit the motion range of each axis by installing limit switches or setting software limits. Hardware limit switches and software limit switches are used for the permissible movement range and working range of the axes of the technology object.



Hardware limit switches are limit switches that limit the maximum "permissible travel range" of an axis. A hard limit switch is a physical switching element, and it is mapped to the corresponding input switch signal through instruction. According to whether the switch signal is normally open or normally closed to determine whether to flip the signal. After it is set, and when hitting the hardware limit switch, the corresponding axis stops immediately, and the stop deceleration is FASTDEC.

The soft limit switch id different from the hard limit switch, it is only realized by the software program setting, without the help of external switching elements. The software limit switch will limit the "working range" of the axis, and the limit position is directly set by the instruction. After the axis reaches the set position, the motion will stop with deceleration FASTDEC immediately. They should be located inside the relevant hardware limit switch that limits the travel range of the machine tool. Since the position of the software limit switch is more flexible, the working range of the axis can be adjusted according to the current running track and specific requirements.

When the worktable hits the limit switch or the planned position exceeds the software limit, the motion controller stops the motion of the worktable in an emergency. After the limit is triggered, the axis cannot continue to move. At this time, the position of the axis needs to be adjusted so that it is far away from the limit position to restart the movement.

The axis will only generate a stop signal when it hits the limit. At this time, since it takes a certain time to decelerate, the actual position of the axis will exceed the limit by a certain distance.

Assume that the SPEED speed is v0 when it stops, and the fast deceleration FASTDEC is a. The calculation formula: vt2-v02=2as, bring in the following data: 0-1002=2*(-1000)*s, the overshoot distance s=5, which can be known, reduce overshoot by increasing FASTDEC and decreasing SPEED.

Example:

```
BASE(0)                 ′select axis 0
ATYPE=1                 ′axis type setting
UNITS=100               ′pulse equivalent 100
SPEED=100               ′speed 100units/s
ACCEL=500               ′acceleration
DECEL=500               ′deceleration
FASTDEC=1000            ′fast deceleration 100units/s/s
DPOS=0                  ′the coordinate is cleared
FS_LIMIT=200            ′set positive software limit position is 200units
MOVE(300)               ′moves 300units
WAITIDLE(0)             ′wait until axis stopped
?DPOS(0)                ′print result: 205units
```

The value of positive/negative software limit FS_LIMIT and RS_LIMIT needs to be between -REP_DIST and +REP_DIST, the software limit parameters will take effect. Otherwise, the positive/negative software limit setting will be invalid. When canceling the software limit, it is recommended not to modify the value of REP_DIST, but to set FS_LIMIT and RS_LIMIT to a larger value.

Routine: the application of positive and negative limit position

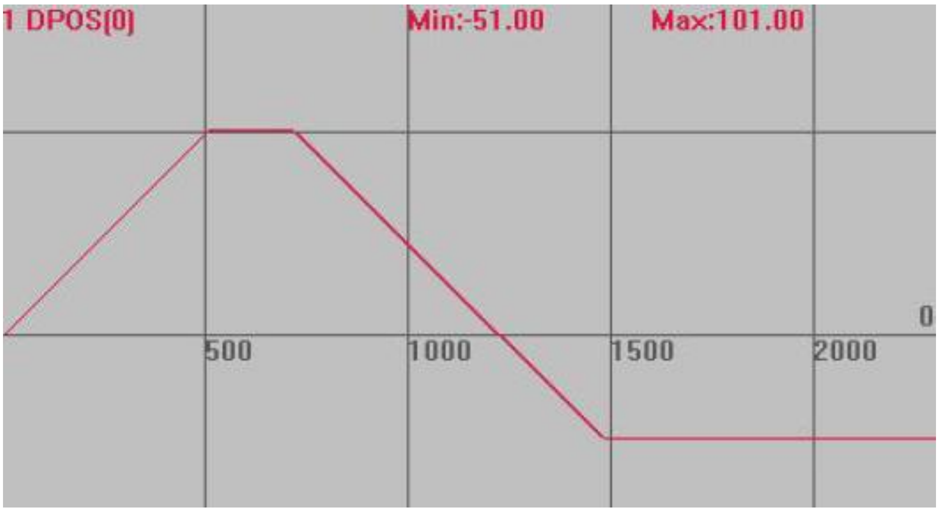
```
ERRSWITCH = 3
RAPIDSTOP(2)
WAIT IDLE
BASE(0)                 ′select axis X move
DPOS = 0
ATYPE=1                 ′pulse stepper or servo
UNITS = 100             ′pulse equivalent, 100 pulses per mm
SPEED = 200
ACCEL = 20000
DECEL = 20000
TRIGGER
'set software limit position
```

```
REP_DIST = 100000000              'the default, don't modify this value
RS_LIMIT = -50    'negative software limit position, it takes effect when it exceeds -REP_DIST
FS_LIMIT = 100    'positive software limit position, it takes effect when it is below -REP_DIST
VMOVE(1)                          'continue to move in positive direction
WAIT UNTIL AXISSTATUS AND (512)   'judge positive limit position generates or not
PRINT "SOFTLIMT FS", *DPOS
DELAY(200)
VMOVE(-1)                         'continue to move in negative direction
WAIT UNTIL AXISSTATUS AND (1024)  'judge negative limit position generates or not
PRINT "SOFTLIMT RS", *DPOS
RS_LIMIT = -200000000             'close software limit position
FS_LIMIT = 200000000
END
```

Print result:

```
Axis:0 AXISSTATUS:200h,FSOFT
SOFTLIMT FS 101
Axis:0 AXISSTATUS:400h,RSOFT
SOFTLIMT RS -51
```

The motion trajectory is as follows, the positive software limit is set to 100, so that the axis is forced to stop after moving to the 100, and the negative software limit is set to -50, the axis cannot continue to move after moving to this position in the negative direction.



5.6 Position Latch

Related instructions:

Instruction	Description	Usage
REGIST	Set latch method	REGIST (method value)
REG_INPUTS	Latch input mapping	REG_INPUT=$input number
MARK	Judge latch is triggered or not	WAIT UNTIL MARK
MARKB	Judge the second latch is triggered or not	WAIT UNTIL MARKB
MARKC	Judge the third latch is triggered or not	WAIT UNTIL MARKC
MARKD	Judge the forth latch is triggered or not	WAIT UNTIL MARKD
REG_POS	Save latched measurement feedback position	Print REG_POS
REG_POSB	Return latch 2 measurement feedback position	Print REG_POSB
REG_POSC	Return latch 3 measurement feedback position	Print REG_POSC
REG_POSD	Return latch 4 measurement feedback position	Print REG_POSD
OPEN_WIN	Latch triggered start coordinate range point	OPEN_WIN=POS
CLOSE_WIN	Latch triggered end coordinate range point	CLOSE_WIN=POS

The latch function of the controller is mainly used to latch the position of the encoder MPOS (the latest firmware of the 4 series and above controllers supports virtual axis and pulse axis latch). When the latch signal is triggered, the current position information is immediately captured in the position latch, and clear the previous latched position coordinates. When reading latch position information, the position information latched when the last latch signal is triggered will be read. The number and position of the latched channel ports of different types of controllers are different, please refer to the hardware manual of the corresponding type of controller.

It should be noted that the operation interface of position latch is accessed according to the axis number. Different types of axes have different latch parameters. Before use, first determine the type of axis. The types of axes that support latch are divided into the following types: local pulse axis, EtherCAT axis, RTEX axis, also need to pay attention to the number of latch ports to avoid errors caused by overflow of latch data.

The pulse axis type generally adopts three latches of R0, R1, and Z pulse; the bus axis type generally adopts R2, R3 latch.

In addition to supporting controller latching, the EtherCAT bus controller also supports driver latching. At this time, the driver IO points are used to achieve latching. For the specific mode, see the command syntax. RTEX only supports controller latches.

When supporting the simultaneous use of EtherCAT drive latch and controller latch, 4 latch channels are required. The 4 channels refer to MARK, MARKB, MARKC, MARKD, and the latch channel corresponding to the latch input port is specified by REG_INPUTS. When the latch is generated, the axis state MARK will be set to ON, and the latched position will be stored in the parameter REG_POS.

The input signals R0, R1 and Z signals of each axis can use the latch function, and the R0 and R1 inputs generally correspond to input ports 0 and 1. When using two signal latches, the second signal latch uses MARKB and REG_POSB, MARK and REG_POS need to be paired, that is, the numbers are the same.

The rising edge/falling edge refers to the internal state of the controller. Different types of input ports may be inconsistent, and it is necessary to confirm the actual latched edge.

How to use the latch function:

1) Determine whether the current hardware conditions meet the latching requirements, and determine the axis that needs to be latched;

2) Set the latch input mapping port REG_INPUT, the input port needs to support the latch function;

3) Set the latch mode REGIST and wait for the latch to trigger MARK;

4) Latch completes print latch position information REG_POS;

5) The start and end coordinates of the latched position can be read, and the latched position can be called by other instructions.

Refer to the description of the REGIST instruction for the latch method of the controller.

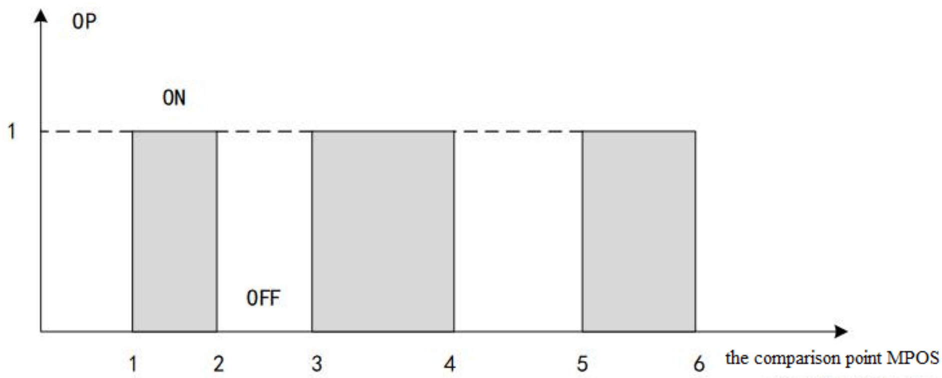
5.7 Hardware Comparison Output

Related instructions:

Instruction	Description	Usage
HW_PSWITCH	Hardware position comparison output	Set the comparison point
HW_PSWITCH2	Bus hardware position comparison output	Set the comparison point and the comparison output port
HW_TIMER	Hardware timer output	Output in periodically

There is a position comparison unit in the motion controller. The hardware comparison output is to operate the output port action by comparing whether the axis reaches the set position. Generally, the encoder position is compared with the set position. When the encoder position reaches a set position When comparing the position, trigger the level of the corresponding output port to flip once.

As shown in the figure below, when the set position 1 is reached, the level of the specified output port is flipped, the level of the designated output port is flipped again when it reaches position 2, and the level is flipped again when it reaches position 3. After all points are compared, the level remains the same as after the last flip. state.



Hardware comparison output needs to be supported by some models of 3 series and 4 series and above controllers. It is necessary to operate the output port that supports this function. The controller supports software comparison output PSWITCH command, hardware comparison output HW_PSWITCH command (only supports pulse axis), HW_PSWITCH2 command (Both pulse axis and bus axis are supported).

For the pulse axis, the difference between HW_PSWITCH and HW_PSWITCH2 is that there is a one-to-one correspondence between the axis and output of HW_PSWITCH, and there is no need to specify the output axis number; HW_PSWITCH2 can be specified in the output port that supports this function. The HW_PSWITCH command can operate multiple output ports at the same time to output simultaneously. The HW_PSWITCH2 instruction supports more controller models.

Comparing the feedback MPOS of the encoder, the position accuracy is higher. When the encoder is connected (the pulse axis axis type ATYPE is 4 or the bus axis type), the encoder feedback position MPOS is compared. When the encoder is not connected (the pulse axis axis type ATYPE is 1 or 7) Compare the target position DPOS.

If the comparison position is a large number of continuous equidistant outputs, the HW_TIMER hardware timing output can be used. At this time, it is necessary to set the starting comparison output position, interval distance and repetition period.

If the comparison position is a non-equidistant position value, use the HW_PSWITCH and HW_PSWITCH2 commands to specify the position in the TABLE table for output, and store the position data that needs to be compared and output in the TABLE table. At this time, it is necessary to ensure that the TABLE position data is not modified before all comparison points are completed, and the data in the TABLE table is a monotonically increasing positive distance value or a monotonically decreasing negative distance value, otherwise an error will occur.

When comparing the spindle with encoder input, the encoder position is automatically used to trigger, and the MOVEOP_DELAY parameter can be used to adjust the output exact moment. Different bus drivers may have different effects, which can also be adjusted by the

MOVEOP_DELAY parameter.

5.8 Precision Output

Related instructions:

Instruction	Description	Usage
MOVE_OP	Output in buffer	MOVE_OP(number, state)
AXIS_ZEST	Start precision output	Set the function according to bit
MOVEOP_DELAY	Delay output in buffer	Output in advance or delay

The MOVE_OP instruction defaults to normal output. The normal output operation needs to wait for one controller cycle to execute, while the precise output operation can respond within one pulse sent by the motor, which can greatly improve the accuracy of the process. At the same time, the MOVEOP_DELAY instruction can be used to adjust the response time (earlier or later).

The minimum error of precision output pulse output mode is 1 pulse, and the minimum error of bus control mode is within 1us.

Only controllers that support the hardware comparison output function can use the precision output function, and both use the same hardware resources.

Use the AXIS_ZSET command to set whether to enable precise output, and then use the MOVE_OP command to enable the precise output to take effect. Note that the output channel should select the channel that supports precise output, that is, the channel that supports hardware comparison output. The number of different models is different, generally special function starts from IO number 0.

There are two trigger modes for precision output, target position DPOS trigger or encoder feedback MPOS trigger.

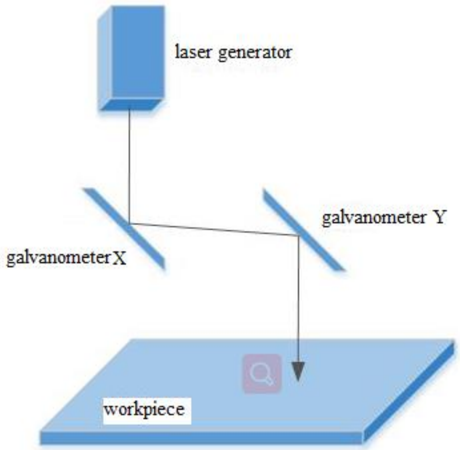
When there is no encoder feedback, the precision output function automatically uses the command position DPOS to compare the trigger. The motor always has a certain following error (following error = DPOS-MPOS). When the encoder feedback is used, the encoder feedback MPOS trigger will be more accurate. Precisely, whether to start the encoder position is also configured through the AXIS_ZSET instruction. According to the different effects of different drives, you can also use the MOVEOP_DELAY parameter to adjust the exact timing of the IO output.

5.9 Galvanometer Control System

5.9.1 The Description of Galvanometer

1. **The galvanometer working principle**

   Laser galvanometer is a special motion device specialized for laser processing field. It reflects the laser through two galvanometers, forming the motion in XY plane. Laser galvanometer is different from general motor, the inertia is extremely small, and the load is small in motion. There are two small reflection lens, X and Y use different motors to control deviation, the system response is very fast.

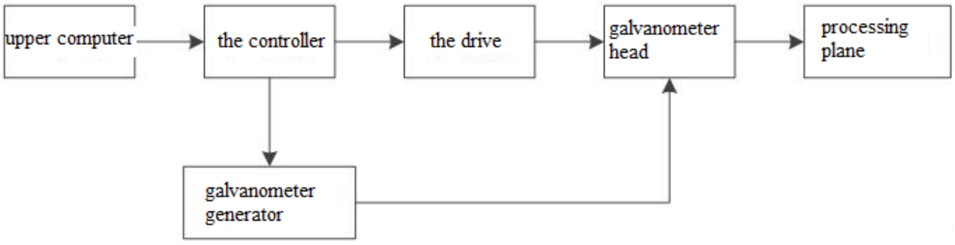


There are two basic movements of laser galvanometer movement: jump movement and the marking movement.

During the jump movement, the axis moves to the position to be processed, and the laser is turned off, which does not affect the processing of the trajectory, so it can move at a high speed. During the marking movement, the laser is turned on to process the trajectory, so the user needs to set the appropriate movement speed according to the actual processing requirements.

Galvanometer is an excellent vector scanning device. It is a special oscillating motor (laser galvanometer), the basic principle is that the energized coil generates torque in the magnetic field, but different from the rotating motor, the rotor is added with a reset torque by mechanical springs or electronic methods, the size and the angle deviating from the equilibrium position is proportional. When the coil is supplied with a certain current and the rotor is deflected to a certain angle, the electromagnetic torque is equal to the restoring torque, so it cannot rotate like an ordinary motor, but can only be deflected. The deflection angle is proportional to the current.

2. **Basic Structure of Galvanometer Control System**



The galvanometer system consists of the above parts to form a basic system, in which the main components of the galvanometer are two X/Y reflection lens, two motors that control the rotation of the X/Y mirrors respectively, and a man-machine operating system, encoder and others can also be added according to actual needs.

3. Basic Requirements for The Controller

Because the laser marking machine relies on the deflection of the X/Y galvanometer, the laser is reflected on the work surface for precise engraving. The control of the galvanometer is controlled by the open-loop controller, so it must be linear, that is, there is a linear relationship between the input signal and the deflection angle. Because the galvanometer is a fast and precise machine, it is required that the acceleration be as large as possible from one working state to another, so that the marking space time is infinitely small.

The galvanometer movement adopts the buffer movement method, that is, the user needs to transmit the movement and process data to the axis movement buffer, and then start the buffer movement, and the motion controller will continuously execute the movement data transmitted by the user in sequence until all the movement data are complete.

In the laser galvanometer motion control system, there are not only motion control, but also laser control. How to effectively deal with the cooperation between the galvanometer movement and the laser switch is a very important issue. Coordinating the relationship effectively between the laser and the movement, the movement trajectory can be precise.

Motion control: During the marking movement, the laser will move along the given marking trajectory at the set marking speed. When executing the relevant marking instructions, the laser galvanometer motion controller will automatically turn on the laser. If the next is still a marking instruction, the laser is always on until the last marking instruction ends, or instructions in buffer area are executed. The laser will be turned off automatically if encountering the jump instruction in buffer area. The laser will be turned on again only when meeting the marking instruction. Before starting the movement, the galvanometer coordinates should be adjusted to ensure the correct marking trajectory, and the buffer should be cleared at the same time.

Laser control: It mainly includes controlling the on/off control of the laser and the duration of the laser, and using the OP command to control the on-off of the laser. The laser energy can be controlled according to the difference of the laser, corresponding to the analog quantity, digital quantity output port, and the duty cycle of output port PWM correspond to the amount of control energy.

Pin No.	Signal	Description
1	CLOCK-	Clock signal -
2	SYNC-	Synchronization signal -
3	X channel-	Galvanometer X channel signal -

4. Applications

It is mainly used for laser marking, including laser cutting, stage lighting control, laser drilling, etc. It is a non-contact, non-polluting and non-wearing new marking process. It adopts automatic control and greatly improves the reliability. Laser marking uses a high-energy-density laser beam. With the regular movement of the laser beam on the surface of the material, the on-off of the laser beam is controlled at the same time, so that physical or chemical changes occur on the surface of the target material, and the laser beam can be processed a specified pattern on the surface of the material.

Compared with the traditional marking process, laser marking has the following advantages:

The marking speed is fast and the handwriting is clear.

Non-contact processing, pollution is less and no wear.

It is convenient to work and has strong anti-counterfeiting ability.

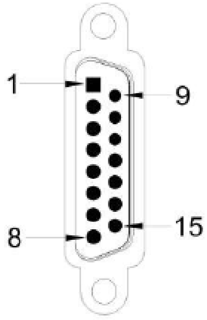
High-speed automatic operation, low production cost and reliable operation.

5. ZMC120SCAN Controller Galvanometer Interface Signal

ZMC420SCAN is the controller that supports laser galvanometer control, each general output of the controller all supports PWM function.

The local axis 4/5 can be configured as the first galvanometer through ATYPE=21, The local axis 6/7 can be configured as the second galvanometer through ATYPE=21, and the axis number can be changed through AXISZ_ADDRESS instruction.

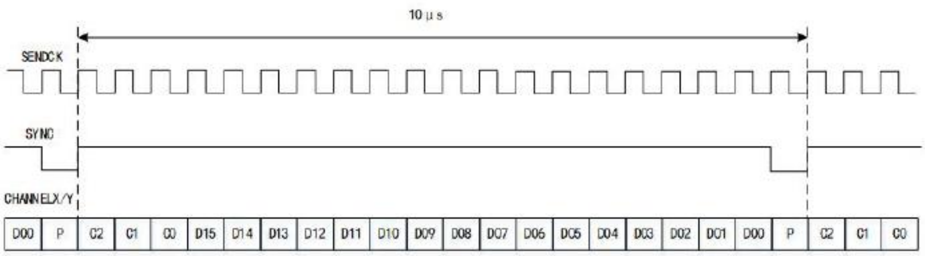
4	Y channel-	Galvanometer Y channel signal -
5	NC	Reserve
6	STATUS	Galvanometer status signal -
7	NC	Reserve
8	GND	Digital ground
9	CLOCK+	Clock signal +
10	SYNC+	Synchronization signal +
11	X channel+	Galvanometer X channel signal +
12	Y channel+	Galvanometer Y channel signal +
13	NC	Reserve
14	STATUS+	Galvanometer status signal +
15	GND	Digital ground



6. XY2-100 Galvanometer Protocol

ZMC420SCAN supports XY2-100 galvanometer protocol.

In the galvanometer control system, the XY2-100 protocol is widely used as the interface definition and communication protocol of the digital laser scanning galvanometer. Communication protocol refers to the rules and conventions that must be followed by both entities to complete communication or services. The XY2-100 protocol includes four signals: SENDCK (clock signal), SYNC (synchronization signal), CHANNEL X (X channel data), CHANNEL Y (Y channel data), these four signals are a synchronous serial transmission process.



The SENDCK signal is a clock signal with a frequency of 2MHz. When it transitions from low level to high level, the data bit is written; when it transitions from high level to low level, the data bit is reflected by the system sampling.

The SYNC signal is used to provide synchronization information for data conversion. When it goes from low level to high level, the first bit of data is sent; when it goes from high level to low level, the last bit of parity is sent.

CHANNEL X/Y is the data signal, which consists of 20 bits, among which C2, C1, C0 are the moving direction value of the galvanometer, the reference value is 001, D15 ~ D0 are the data bits, which are 16-bit binary numbers, used to control the vibration. The angle that the mirror rotates, the last bit P is the parity check bit, when there is an even number of "1" in the sent data, the corresponding check bit is "0", and when there is an odd number of "1" in the sent data, the corresponding check digit is "1"

7. Galvanometer Correction

The galvanometer is generally realized by correcting the galvanometer to control the exact position distance of the galvanometer. The galvanometer correction is actually to establish a corresponding relationship between the theoretical galvanometer moving distance and the actual galvanometer moving distance, and then the corresponding moving distance is combined with the established relationship in the process of moving, so as to achieve the purpose of accurately moving the galvanometer and achieve the effect of galvanometer correction.

Below are galvanometer correction instructions:

ZSCAN_CORRECT(ixy,imode,imaxline,imaxrow,x1,y1,x2,y2,tableindex)

ixy: the value is 0/1, there are 2 galvanometers to be selected: 0-the first galvanometer, 1- the second galvanometer.

imode: 0-turn off the correction function, 1- use correction for different areas.

imaxline: line number, the point in Y direction is the line number

imaxrow: row number, the point in X direction is the row number

x1, y1, x2, y2: the theoretical coordinates of the lower left corner and the lower right corner

tableindex: table index is to be stored by measured real coordinate, first X, then Y, the first line (stored as the row number), then the next line.

Galvanometer supports a maximum of 64*64 correction points to establish the theoretical coordinates of the lower left corner and the upper right corner, and the theoretical coordinates and the measured actual graphic coordinates written in the corresponding TABLE array are processed correspondingly. The galvanometer axis currently connected to the galvanometer interface. The galvanometer correction parameters are not saved after power off, so it should be noted during use that the galvanometer needs to be corrected again after the power is turned on again.

The galvanometer is an absolute value system. After the power is turned on, the controller is always in the state of communication with the motor. Modifying the DPOS of the galvanometer axis will cause the offset of the galvanometer. Therefore, do not modify the DPOS value of the galvanometer axis casually during the use of the galvanometer. It can move to the corresponding position through MOVEABS.

5.9.2 Galvanometer Application Process

1. When using the galvanometer axis, please set the axis type of corresponding galvanometer

axis 4, axis 5, (axis 6, 7) to be connected as 21.

2. Set the axis parameters for the corresponding galvanometer axis. The set axis pulse will affect the movement distance of the galvanometer during motion. Therefore, the pulse equivalent can be fixed as a size, and then the galvanometer axis at the current position can be corrected to the correct distance through the galvanometer correction command.
3. If the laser needs to be switched on and off during the movement of the galvanometer, the high-speed output port should be selected to control the switch light, and the corresponding precise output setting should be turned on, so that the output port can emit light in a short time after reaching the position, and achieve accurately control for the laser.
4. If the galvanometer axis needs to return to zero, the galvanometer axis can be moved to position 0 through MOVEABS command, and the DPOS value of the galvanometer axis cannot be modified casually during the movement of the galvanometer, otherwise it will cause the offset of the galvanometer axis, the corresponding galvanometer motor will also vibrate.
5. The galvanometer axis can be exchanged by the axis mapping instruction AXIS_ADDRESS, and the galvanometer axis can be operated by other axis numbers to change the axis. In addition, the current direction of the galvanometer axis cannot be modified by the command. In order to correct the direction of the galvanometer, it is necessary to invert the coordinates of the galvanometer that need to be reversed in the correction part of the galvanometer, and then correct it again to modify the direction of the galvanometer axis.
6. It can operate the galvanometer axis and the common motor axis to establish continuous interpolation, establish the linkage between the galvanometer axis and the ordinary axis, and realize hybrid interpolation.

Laser Control Notes:

The energy control of the laser has the following control methods:

1. The analog quantity controls the energy: the precision of analog is 10 bits, 0-10V. The value of 0-4096 controls corresponding energy.

2. Digital signal combination to control energy: it is combined with output signals, the energy selects the energy corresponding to each combination.

3. Control energy output through PWM duty cycle.

Example: The energy combination of Lianpin laser mopa laser is as follows:

Pin No.	Setting 1	Setting 2	Setting 3	Setting 4	Setting 5
Pin 1	0	0	0	0	1
Pin 2	0	0	0	0	1
Pin 3	0	0	0	0	1

Pin 4	0	0	0	0	1
Pin 5	0	0	0	1	1
Pin 6	0	0	1	1	1
Pin 7	0	1	1	1	1
Pin 8	1	1	1	1	1
Current	50%	75%	87.5%	93.75%	100%
Laser work	52%	77%	89%	93%	100%

Galvanometer Routine:

Example 1: two galvanometer axes interpolation

Description: two galvanometer axes achieve mark 5 5mm small segment round in one line.

```
′set axis number of galvanometer axis, and configure the axis type
BASE(4,5)
ATYPE=21,21
′set basic parameters
UNITS=300,300
SPEED=5000,5000
ACCEL=SPEED*20,SPEED*20
DECEL=SPEED*20,SPEED*20
MOVEABS(0,0)
FORCE_SPEED=5000
'start continuous interpolation
MERGE=ON
AXIS_ZSET(4)=3                    'start MOVE_OP precision output function
'set frequency
PWM_FREQ(2)=2000
WHILE 1
    IF MODBUS_BIT(0)=ON THEN
        MODBUS_BIT(0)=OFF
            OP(0,OFF)
            BASE(4,5)
            'energy switch
            OP(11,ON)
            'MO switch
            OP(1,ON)

            'marking 5 small segment round, the trajectory moving data
    FOR j = 0 TO 4
```

```
MOVE(-15, 0)
	MOVE_OP(0,ON)
	MOVE(-0.038, 0.434)
	MOVE(-0.113, 0.421)
	MOVE(-0.184, 0.395)
	MOVE(-0.250, 0.357)
	MOVE(-0.308, 0.308)
	MOVE(-0.357, 0.250)
	MOVE(-0.395, 0.184)
	MOVE(-0.421, 0.113)
	MOVE(-0.434, 0.038)
	MOVE(-0.434, -0.038)
	MOVE(-0.421, -0.113)
	MOVE(-0.395, -0.184)
	MOVE(-0.357, -0.250)
	MOVE(-0.308, -0.308)
	MOVE(-0.250, -0.357)
	MOVE(-0.184, -0.395)
	MOVE(-0.113, -0.421)
	MOVE(-0.038, -0.434)
	MOVE(0.038, -0.434)
	MOVE(0.113, -0.421)
	MOVE(0.184, -0.395)
	MOVE(0.250, -0.357)
	MOVE(0.308, -0.308)
	MOVE(0.357, -0.250)
	MOVE(0.395, -0.184)
	MOVE(0.421, -0.113)
	MOVE(0.434, -0.038)
	MOVE(0.434, 0.038)
	MOVE(0.421, 0.113)
	MOVE(0.395, 0.184)
	MOVE(0.357, 0.250)
	MOVE(0.308, 0.308)
	MOVE(0.250, 0.357)
```

```
                MOVE(0.184, 0.395)
                MOVE(0.113, 0.421)
                MOVE(0.038, 0.434)
                WAIT IDLE
                MOVE_OP(0,OFF)
        NEXT
        ENDIF
WEND
```

Motion efficiency figure:

Example 2: Mixed interpolation motion of galvanometer axis and common axis

Description: The galvanometer axis and the rotation axis establish an interpolation two-axis coordinated motion to mark and clean the graphics.

The cleaning length is 58, and the cleaning width is 30. The workpiece to be cleaned is placed on the rotating axis 0, the axis 5 controls the laser movement, and the Y axis reciprocates for cleaning.

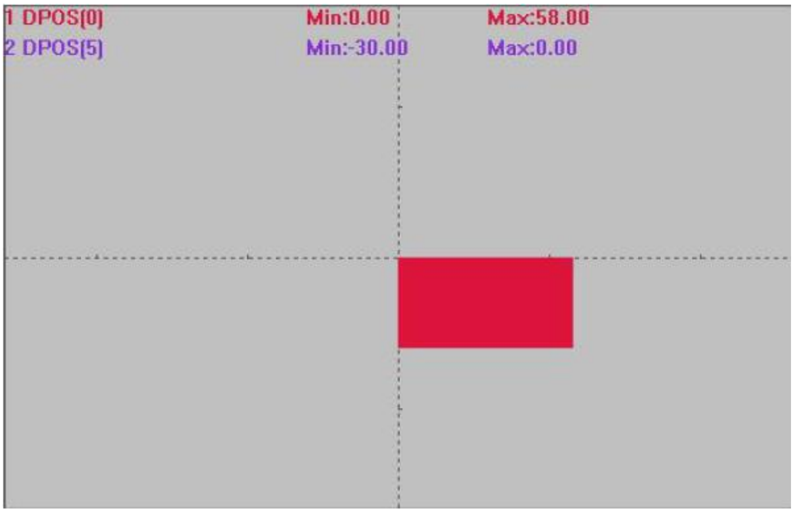
```
BASE(0,5)
ATYPE=7,21
UNITS = 10000/360,10000/18
SPEED=1000,5000
ACCEL=SPEED*5, SPEED*5
DECEL=SPEED*5, SPEED*5
MOVEABS(0,0)
MERGE=ON                'start continuous interpolation
AXIS_ZSET(0)=3          'start main axis MOVE_OP precision output function
OP(12,ON)               'enable pulse axis axis 0
```

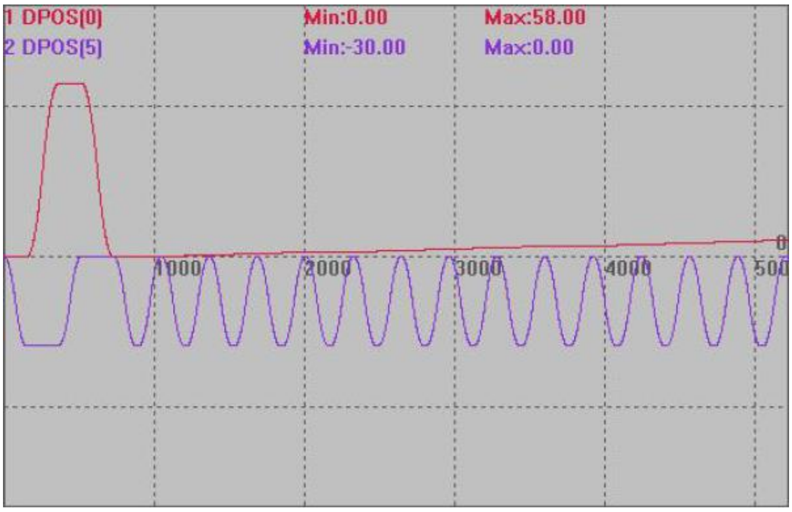
```
PWM_FREQ(2)=2000          'set DB25 the frequency of external control laser
PWM_DUTY(11)= 0.8         'set energy
PWM_FREQ(11) = 2000

WHILE 1
    LOCAL i               'cycle condition
    LOCAL sum             'accumulate the rotation angle
    IF MODBUS_BIT(0)=ON THEN
      sum = 0
      MODBUS_BIT(0)=off
        OP(0,OFF)
        OP(11,ON)         'energy switch
        OP(1,ON)          'mo switch
        WA 100

        MOBE_OP(0,ON)
        TRIGGER
        MOVE(0,-30)
        WAITIDLE
        MOVE(58,0)
        WAITIDLE
        MOVE(0,30)
        WAITIDLE
        MOVE(-58,0)
        WAITIDLE

        'when rotation axis rotated one certain angle, clean the marked graphics on rotation axis.
        FOR i = 0 TO 57.6 STEP 0.4    'clean
          sum = sum + 0.4
            MOVE(0,-30)
            MOVE(0.4,0)
            MOVE(0,30)

    NEXT
    ?"cylinder rotation angle", sum
    MOVE_OP(0,OFF)
    MOVE(-58,0)
    ENDIF
```

WEND

END

Motion efficiency figure:





5.10 Robotic Arm

Zmotion controller supports more than 30 kinds of manipulator algorithms. It can be used after establishing a manipulator connection according to the type of manipulator frame. It can control the motion of the manipulator smoothly and accurately. For detailed instructions, please refer to the "ZMotion Robot Manual Instruction".

5.10.1 Related Concept of Robot

1. Joint-axis and Virtual-axis

1) Joint axis

The joint axis refers to the rotation joint in the actual mechanical structure, and in the program it is generally displayed the rotation angle. Since there is a reduction ratio between the motor and the rotating joint, the units should be set according to the actual joint rotation for one circle. At the same time, when filling in the structural parameters in the table, the calculation should be based on the center of the rotating joint instead of the center of the motor axis.

2) Virtual axis

The virtual axis does not actually exist, it is abstracted as 6 degrees of freedom of the world coordinate system, which are X, Y, Z, RX, RY, RZ in sequence. It can be understood as the three linear axes of the space rectangular coordinate system and the three rotation axes around the axes, which are used to determine the processing track and coordinates of the working point at the end of the manipulator.

2. Coordinate System

1) Joint coordinate system

The absolute angle of each axis is relative to the origin position, including all joints of the robot, each joint is independent of each other, and the coordinate unit is angle. Manipulating one of the joints does not affect the other joints.

2) Cartesian coordinate system

World coordinate system: The world coordinate system is a standard Cartesian coordinate system fixed in space, the chassis of the robot is the coordinate origin, and its position is determined according to the type of robot. The virtual axis is operated according to the world coordinate system. At this time, each joint will automatically calculate the angle that needs to be rotated.

User coordinate system: the Cartesian coordinate system defined by the user for each work space, which is used for teaching and executing the position register, executing the position compensation command, etc. When not defined, the coordinate system will be replaced by the world coordinate system.

The main purpose of the manipulator algorithm is to connect the joint coordinate system with the Cartesian coordinate system.

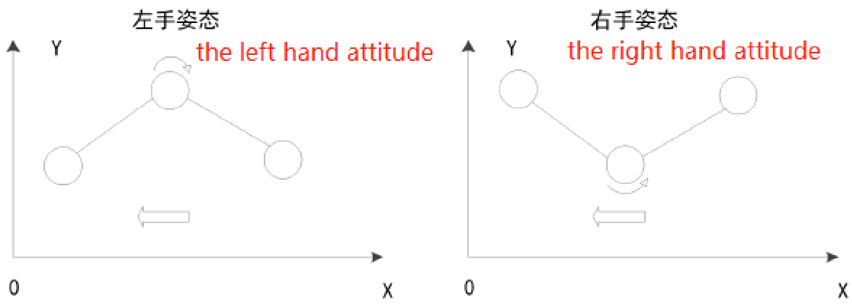
Coordinate system transformation refers to the transformation from the original coordinate system to another coordinate system when describing the same space. In the use of the manipulator, it is often used to determine the coordinate system of the workpiece.

The workpiece coordinate system is a Cartesian coordinate system fixed on the workpiece, and there is a transformation between the workpiece coordinate system and the world coordinate system. Each manipulator can have a Nuogan workpiece coordinate system to represent different workpieces, or to represent the same workpiece at different positions.

Virtual axis of the robot type meeting XYZ three axes supports this function.

3. Attitude

Mathematically speaking, the attitude of the manipulator is the solution of multiple sets of joint axes from the same set of virtual axis values. That is, when the manipulator moves to a certain coordinate point in the Cartesian coordinate system, it can have various motion trajectories, and these motion trajectories correspond to different attitudes. For the two attitudes of SCARA as shown in the figure below, when moving in the X direction, joint-axis has two ways to do motion.



4. Singularity

In the inverse solution mode, when the robot moves to a certain position, it will lose a certain degree of freedom, and this position is called the singular point, and it should be avoided to move to the singular point in the actual use process. For example, when the SCARA manipulator is fully straightened, it cannot translate in the X direction at this time. And when it needs to operate to move in the negative direction of X, the structure cannot judge which posture motion to use, and the manipulator cannot move at this time. Adjust the position of the joint axis in the forward solution mode, and then switch to the inverse solution mode for use.

5.10.2 Forward and Inverse Solution Motion

The establishment of the manipulator is set by the CONNREFRAME (positive solution) instruction and the CONNREFRAME (inverse solution) instruction. The virtual axis MTYPE (motion type) value is 34 in CONNREFRAME, and the joint axis MTYPE value is 33 in CONNREFRAME. Check whether a specific axis is located in the corresponding mode through MTYPE.

Joint axes and virtual axes are specified by the CONNREFRAME or CONNFRAME instructions, and the controller supports multiple robots as long as the number of axes is sufficient.

The program can control the movement of the joint axis or the virtual axis through the motion command, but only the virtual axis or the joint axis can be operated at the same time, and the two cannot be operated at the same time.

When operating the joint axis movement, the virtual axis needs to be in the CONNREFRAME mode, so that it automatically points to the current spatial coordinate. When operating the virtual axis movement, the joint axis needs to be in the CONNREFRAME mode, so as to automatically point to the current joint axis coordinate.

Robot mode can be cancelled through CANCEL or RAPIDSTOP instruction.

1. Inverse Solution Motion

The motion corresponding to CONNFRAME is the inverse solution motion, and this instruction acts on the joint axis. At this time, only the virtual axis can be operated. The virtual axis can be moved in the Cartesian coordinate system such as straight line, circular arc, space arc, etc. The joint axis will automatically move to the position after the inverse solution under the action of CONNFRAME.

The inverse motion modes refer to the two motion modes of the controller. Under the premise of ensuring the accurate position of the end point, the manipulator will make a trade-off between the accurate trajectory of the motion process and the smooth speed.

Inverse solution motion mode is achieved by connecting to speed through CLUTCH_RATE, the CLUTCH_RATE default value of the controller is 1000000.

CLUTCH_RATE of joint axis	Motion mode description
0	Smooth mode: In this mode, the joint axis uses its own speed and acceleration for speed planning, and the trajectory will be deformed at high speed. It is suitable for occasions where the precision of motion trajectory is not high.
Non-0	Forced mode: In this mode, the joint axis is completely planned according to the speed and acceleration of the virtual axis. This mode can accurately return to the set position, but it will shake when moving at high speed.

2. Forward Solution Motion

The motion corresponding to CONNREFRAME is the positive solution motion, and this instruction acts on the virtual axis. At this time, only the joint axis can be operated, and the joint axis can also perform various movements, but the actual movement trajectory is not a straight arc. This mode is generally used to manually adjust the joint position or return the power-on point to zero.

The joint interpolation motion is the interpolation motion of the manipulator in the positive solution mode, which controls the end point to go straight line, circular arc, etc.

5.10.3 Functions Supported by Robot

1. Robot control

Control the end point of the manipulator to move in the world coordinate system. Multiple manipulator types are supported, and one controller can control multiple manipulators at the same time. The manipulator has several motors, which are called several-joint manipulators. The motor axis that controls the actual mechanical joint movement is called the joint axis of the manipulator. All the joint axes consist of the joint coordinate system, and the joint axis rotates according to the angle in this coordinate system.

2. Coordinate System Rotation

The coordinate system of the movement of the manipulator's working point is rotated and offset with reference to the world coordinate system. A user coordinate system can be constructed. Control the end point of the manipulator to move in the world coordinate system. The coordinate axis of the world coordinate system is assumed to be a virtual axis and moves according to distance units.

3. Mechanical Parameter Correction

The current manipulator parameters are automatically corrected according to the coordinates and characteristics taught by the manipulator joints.

4. Robotic Calculation

Calculation between the coordinates of the end work point and the coordinates of the joint axis.

5. Manipulator Motion Simulation

ZRobotView simulation software shows the movements of the manipulator.

6. Controller Simulate

Support offline simulation, which means it can be used when there is no controller.

5.10.4 Application Cases of Robot

Generally speaking, the inverse solution mode is selected during production and processing, and the movement of the robot is controlled by sending the coordinate position to the virtual axis. During the movement of the robot, corners will appear. It is necessary to set the corner deceleration to prevent the machine from shaking at high speed.

Programming reference steps:

1. Parameter definition: Define the joint length and the distance between each axis, and set the pulse equivalent of each axis.

2. Joint axis setting: Select the axis number of the joint axis, set the axis type, pulse equivalent (the pulse equivalent of the joint axis needs to be converted into an angle), speed parameters, set the inverse motion mode (CLUTCH_RATE), corner deceleration, etc.

3. Virtual axis setting: Select the virtual axis number, set the axis type (ATYPE=0) and pulse equivalent.

4. Store robot parameters in TABLE.

5. Establish the forward and reverse connection of the robot.

Six degrees of freedom manipulator routine:

```
''''''''''''''''''''''''''parameter definition''''''''''''''''''''''''''
DIM LargeZ          ′vertical height of the base
DIM L1              'The X offset from axis 1 to axis 2; the offset from the center of the
                        turntable to the center of the large swing arm
DIM L2              'the length of large swing arm
DIM L3              'the distance between axis 3 center and axis 4 center
DIM L4              'The distance from axis 4 to axis 5.
DIM D5              '5 means turn one cycle, 6 means the number of turns, 0 means no
                        association.
DIM PulesVROneCircle    'the pulse number when virtual attitude axis turns one round
DIM SmallZ          'The vertical distance from the end to the axis 5
DIM SmallX, SmallY      'XY offset of end to center of turntable
DIM InitRx, InitRy, InitRz   'Initial attitude, (0, 0, 0) points to z positive direction
''''''''''''''''''''''''''parameter assignment''''''''''''''''''''''''''
LargeZ=50
L1=0
L2=100
L3=0
L4=60
D5=0
SmallZ=10
SmallX=0
SmallY=0
InitRx=0
InitRy=0
InitRz=0 PulesVROneCircle=360*1000
DIM u_m1            'The number of pulses per round of the motor 1
```

```
DIM u_m2           'The number of pulses per round of the motor 2
DIM u_m3           'The number of pulses per round of the motor 3
DIM u_m4           'The number of pulses per round of the motor 4
DIM u_m5           'The number of pulses per round of the motor 5
DIM u_m6           'The number of pulses per round of the motor 6
u_m1=3600
u_m2=3600
u_m3=3600
u_m4=3600
u_m5=3600
u_m6=3600
DIM i_1            'transmission ratio of joint 1
DIM i_2            'transmission ratio of joint 2
DIM i_3            'transmission ratio of joint 3
DIM i_4            'transmission ratio of joint 4
DIM i_5            'transmission ratio of joint 5
DIM i_6            'transmission ratio of joint 6
i_1=1
i_2=1
i_3=1
i_4=1
i_5=1
i_6=1
DIM u_j1           'The actual number of pulses per round of joint 1
DIM u_j2           'The actual number of pulses per round of joint 2
DIM u_j3           'The actual number of pulses per round of joint 3
DIM u_j4           'The actual number of pulses per round of joint 4
DIM u_j5           'The actual number of pulses per round of joint 5
DIM u_j6           'The actual number of pulses per round of joint 6
u_j1=u_m1*i_1
u_j2=u_m2*i_2
u_j3=u_m3*i_3
u_j4=u_m4*i_4
u_j5=u_m5*i_5
u_j6=u_m6*i_6
```

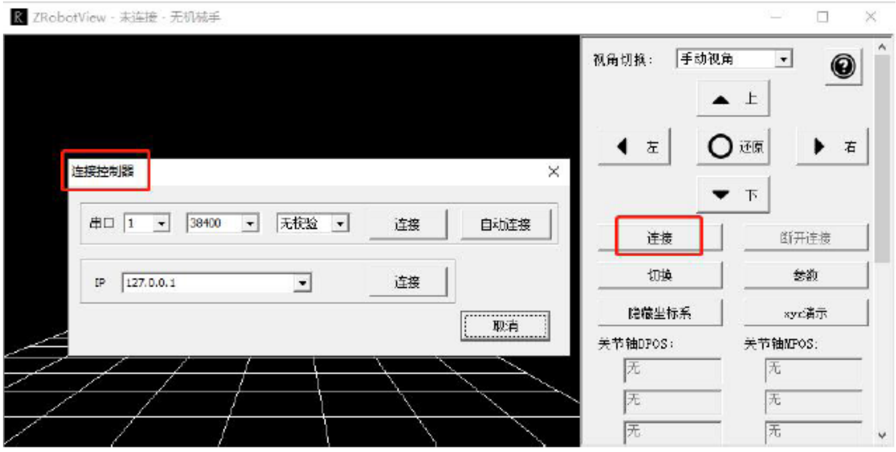
```
''''''''''''''''joint axis setting''''''''''''''''
BASE(0,1,2,3,4,5)          'select joint axis 0,1,2,3,4,5
ATYPE=1,1,1,1,1,1          'set axis type as pulse axis
UNITS = u_j1/360,u_j2/360,u_j3/360,u_j4/360,u_j5/360 ,u_j6/360
                                    'set as pulse per º
DPOS=0,0,0,0,0,0           'set joint axis position, now it should be modified according to actual
                                    situation
SPEED=100,100,100,100,100,100   'speed parameter setting
ACCEL=1000,1000,1000,1000,1000,1000
DECEL=1000,1000,1000,1000,1000,1000
CLUTCH_RATE=0,0,0,0,0,0        'use speed and acceleration of joint axis to for limitation
MERGE=ON                   'start continuous interpolation
CORNER_MODE = 2            'start corner deceleration
DECEL_ANGLE = 15 * (PI/180)   'start deceleration angle 15 degrees
STOP_ANGLE = 45 * (PI/180)     'reduce the angle to the lowest speed 45 degrees
''''''''''''''''virtual axis setting''''''''''''''''
BASE(6,7,8,9,10,11)
ATYPE=0,0,0,0,0,0          'set as virtual axis
TABLE(0,LargeZ,L1,L2,L3,L4,D5,u_j1,u_j2,u_j3,u_j4,u_j5,u_j6,PulesVROneCircle,SmallX,
SmallY,SmallZ,InitRx,InitRy,InitRz)  'fill the parameters according to manual
UNITS=1000,1000,1000,1000,1000,1000   'motion precision is set before, it can't change
                                              during the process
''''''''''''''''establish robot connection''''''''''''''''
WHILE 1
   IF    SCAN_EVENT(IN(0))>0 THEN  'input o, falling edge trigger
       BASE(0,1,2,3,4,5)           'select joint axis number
            CONNFRAME(6,0,6,7,8,9,10,11) 'start reverse solution connection
            WAIT LOADED     'Wait for the motion to load, now the position of the virtual
                                    axis is automatically adjusted.
            ?"reverse solution mode"
ELSEIF    SCAN_EVENT(IN(0))<0 THEN   'input 0, falling edge trigger
     BASE(6,7,8,9,10,11)              'select virtual axis number
            CONNREFRAME(6,0,0,1,2,3,4,5)      'start forward solution connection
            WAIT LOADED                  'wait for the motion to load
            ?"forward solution mode"
```

ENDIF

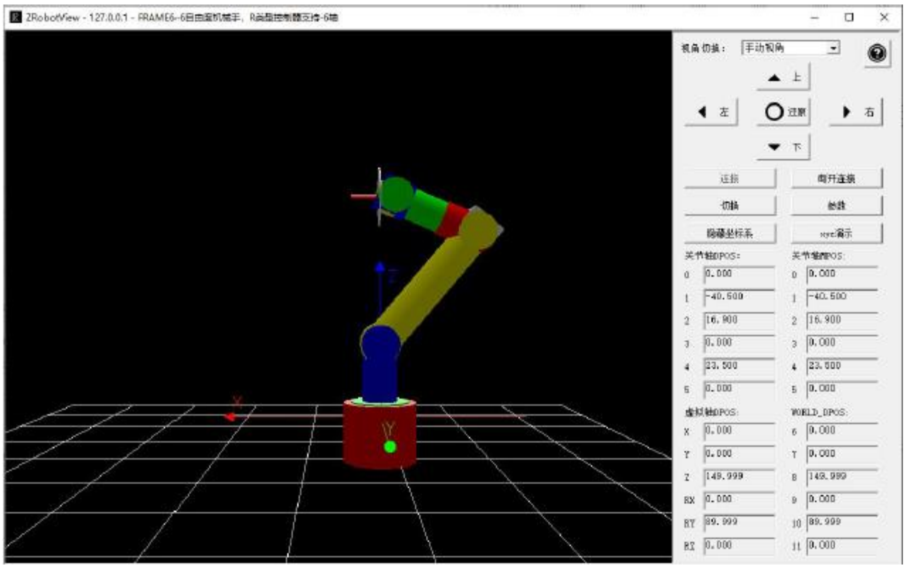
WNED

END

The ZRobotView software simulation can be enabled. How to use it: After downloading this program to the controller, first establish a forward or reverse solution connection (the robot cannot be displayed on the ZRobotView software if it is not established), open the ZRobotView software, and click the "Connect" button on the right, select the connection method with the controller and confirm the connection (select the same IP network segment as the controller for network port communication, and select the same serial port number and baud rate as the controller for serial port communication), now the simulation robot will be built automatically for simulation motion. Also, it can use the "manual motion" function of ZDevelop software, in this interface, simulating the motion of the robot by manually changing the coordinates of the axes.



→6 DOF robot ZRobotView software simulation graphic:



5.11 G Code

As a multi-axis motion controller, ZMC series motion controller supports standard Computerized Numerical Control (CNC) function to realize simple CNC machine tool control, and it can be applied to other positioning and paths through G codes planning occasion.

G-code (G-code) is the most widely used computer numerical control programming language, and it is mainly used in computer-aided manufacturing to control automatic machine tools.

ZBasic supports SUB procedure in G code form and supports G code in standard form. The G code function can be customized according to the actual processing requirements, and the CNC file can be parsed in the form of GSUB. It supports NC machining codes generated by various CAD/CAM software such as UG, MasterCam, ArtCAM, etc., which can be applied to machine tool processing occasions such as engraving and milling machines, precision engraving machines, drilling and tapping centers and machining centers.

For the usage of G code, please refer to the chapter "Self-defined G code" in the simple routine.

Chapter VI Description Related to Axis

6.1 The Concept of Axis

In the motion control system, the object controlled by motion is called “axis”, and the motion platform controlled by one motor is called a motion axis. Each motion axis only has one DOF, which can do linear interpolation or rotation motion. Below is the classification of axis:

Axis Type	Description
Motor axis	Use controller’s pulse axis interface, EtherCAT bus or RTEX bus interface to connect to drive, then assign the axis number for equipment, one motor is used as one axis.
Virtual axis	The virtual axis built in motion controller, not to use actual drive, or as a virtual spindle for synchronous control and as a Cartesian axis in the robot algorithm.
Encoder axis	Use the controller native encoder axis interface, and assign it as actual encoder input for using.

Motor axis: active operation, the motor moves according to the pulses sent by the controller, the number of pulses sent is determined according to the movement parameter change *UNITS, and the target demand position is reflected by the DPOS parameter.

Encoder axis: passive operation, the encoder rotates with the motor, generates pulses, and feeds back to the controller. The number of pulses received by the controller is determined by checking the ENCODER command, and the encoder feedback position is reflected by the MPOS parameter.

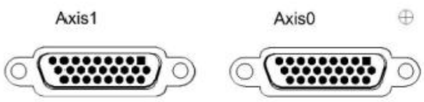
6.2 Axis Number Description

1. Pulse axis number

Pulse motor axis: it runs actively and moves according to the pulses sent by the controller. It is generally divided into servo motors and stepper motors. The number of pulses sent is determined according to the movement parameter variation *UNITS, and the target demand position is reflected by the DPOS parameter.

Encoder axis: it runs passively, follows the motor rotation, generates pulses and feedbacks to the controller, the number of pulses received by the controller is determined by checking the ENCODER command, and the encoder feedback position is reflected by the MPOS parameter.

When using the pulse axis, the motor axis number is the number of the DB axis terminal interface connected to it, which is printed on the shell, in the form of Axis0, Axis1... (If there is no DB interface, please check the corresponding controller hardware manual to determine the axis number).

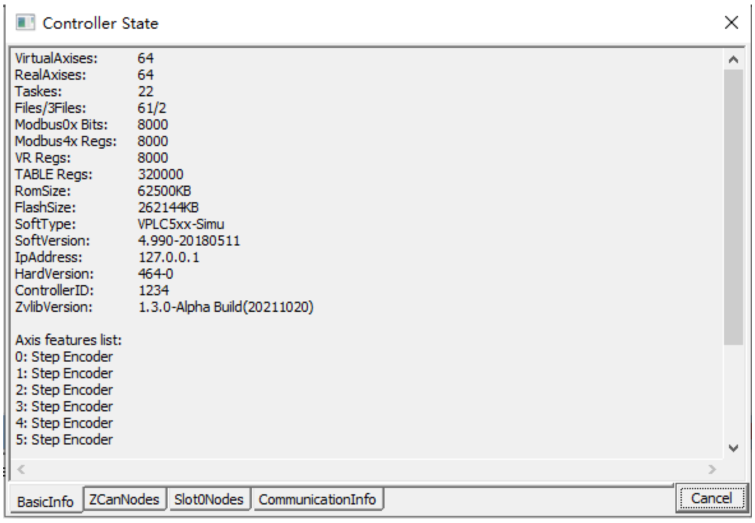


Take the following controller status as an example. For what type of axis each pulse axis interface supports, please refer to the description of the Axis features list in the "State the Controller" window. Step is pulse output, and Encoder is encoder feedback.

If the axis number is marked as “Step Encoder”, it can be configured to have both pulse output Step and encoder feedback input Encoder. When ATYPE=4, the pulse output and encoder feedback are on the same axis number. At this time, DPOS and MPOS are real. When ATYPE=1 or 7, there is only pulse output at this time, and the feedback of the connected encoder is on other axis numbers. See the rules below, DPOS is true, and MPOS copies the value of DPOS.

If there only is "Encoder" behind the axis number, which means it feedbacks occupied axis number. For example, axis 6, the default ATYPE of the feedback axis is 3 (when ATYPE is 3, it corresponds to the quadrature encoder, which can be changed to 6, corresponding to the pulse direction type Encoder).

As shown in the figure below, the motor axis number is axis 0, the corresponding encoder axis number is axis 6, the motor axis 1 corresponds to encoder axis 7, and so on. Assuming that the motor pulse and encoder are both connected to the Axis0 interface, then the motor axis number is 0, the encoder axis number is mapped to axis 6. Assuming the motor is connected to the Axis1 interface, encoder is connected to Axis2 interface, then the motor axis number is 1, and the encoder axis number is 8.



2. Bus Axis Number

Axis number of bus axis maps to axis number of connected drive equipment through AXIS_ADDRESS instruction. Pulse axis number is the same as pulse controller axis number, and the motor and encoder share one axis number.

3. Ways to modify the motor motion direction

Pulse axis:

1) select pulse mode through INVERT_STRP instruction

2) set denominator as negative value through STEP_RATIO

3) drive modifies the round direction

Bus axis:

1) set denominator as negative value through STEP_RATIO

2) drive modifies the round direction

6.3 Axis Status

Check various states of axis through AXISSTATUS instruction. It shows the value in decimal system, and it judges the state according to relative value in binary system, several errors can be made at the same time.

Axis parameter window shows the value in octal system, but the value printed by PRINT command is decimal system.

Bit	Description	Print value

1	Follow-up error over-limit alarm	2	2h
2	Error communicating with remote axis	4	4h
3	Remote drive error	8	8h
4	Forward hardware limit position	16	10h
5	Reverse hardware limit position	32	20h
6	Be finding the origin point	64	40h
7	HOLD speed, keeping signal input	128	80h
8	Follow-up error over-limit error	256	100h
9	Over the forward software limit position	512	200h
10	Over the reverse software limit position	1024	400h
11	CANCEL in the execution	2048	800h
12	When pulse frequency exceeds MAX_SPEED limits, deceleration or MAX_SPEED should be modified.	1096	1000h
14	Robot instruction coordinate error	16384	4000h
18	Power appears error	262144	40000h
19	Precise output buffer overflow	524288	80000h
21	Fail to trigger special motion instruction in motion	2097152	200000h
22	Alarm signal input	4194304	400000h
23	Axis enters pause state	8288608	800000h

AXIS_STOPREASON the historical stop reason of the axis is latched, write 0 to clear it, latch by bit, and latch the information of AXISSTATUS.

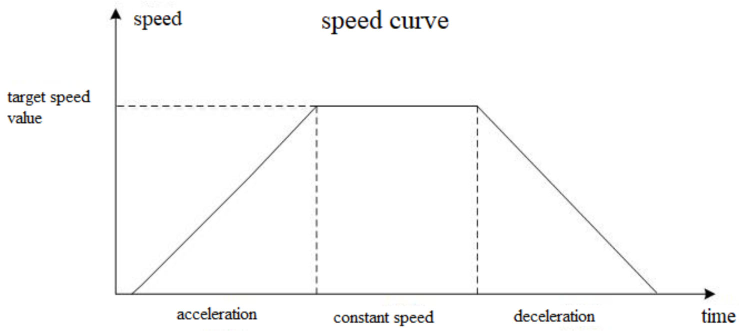
The IDLE command is used to judge whether the motion command added to the axis is completed. It returns 0 during motion and -1 when the motion ends. Generally, the WAIT IDLE (axis number) statement is used in the program to judge the state of the axis.

The MTYPE instruction is used to judge the current motion type of the axis. For example, the return value of MTYPE is 1, which means that the MOVE motion is in progress.

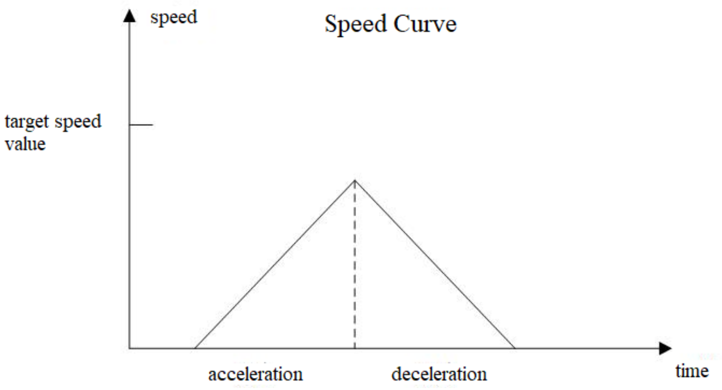
6.4 Axis Speed

6.4.1 Speed Curve

There are 3 stages, acceleration stage, constant speed stage and deceleration stage.



When the displacement is short, there may not be a constant speed stage, but only an acceleration and deceleration stage, as shown in the figure below.



Commonly used speed commands include SPEED motion speed, ACCEL acceleration, DECEL deceleration, FASTDEC rapid deceleration, etc., which are set when the axis parameters are initialized and used as the speed reference for motion commands.

1. Trapezoidal Curve

If SRAMP is not set (set SRAMP equal to 0), the speed curve is a trapezoidal curve. In this speed planning mode, the speed curve changes according to a trapezoidal curve. Keep the parameters such as speed, acceleration and deceleration unchanged.

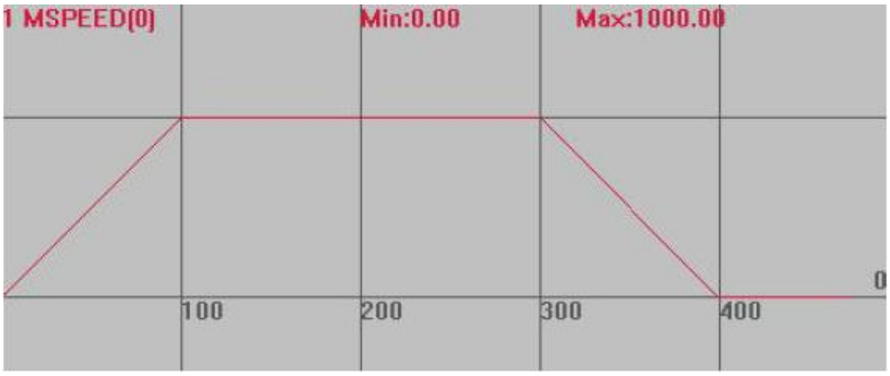
After the speed reaches the set value, it will move at a constant speed. If only the acceleration is set, when the deceleration is 0, the deceleration will be automatically equal to the acceleration value. Generally, the corresponding acceleration and deceleration are set before the movement. Do not modify it during the movement. The adjustment during the movement will cause the movement track to change.

Below is the routine:

```
RAPIDSTOP(2)
WAIT IDLE(0)
BASE(0)
MPOS=0
DPOS =0
UNITS = 100
SPEED = 1000
ACCEL = 10000
DECEL = 10000
SRAMP=0
TRIGGER
```

MOVE(300)

At this time, obtain the below speed curve: now, the acceleration and deceleration process is faster, and the speed change has a greater impact on the machine tool.



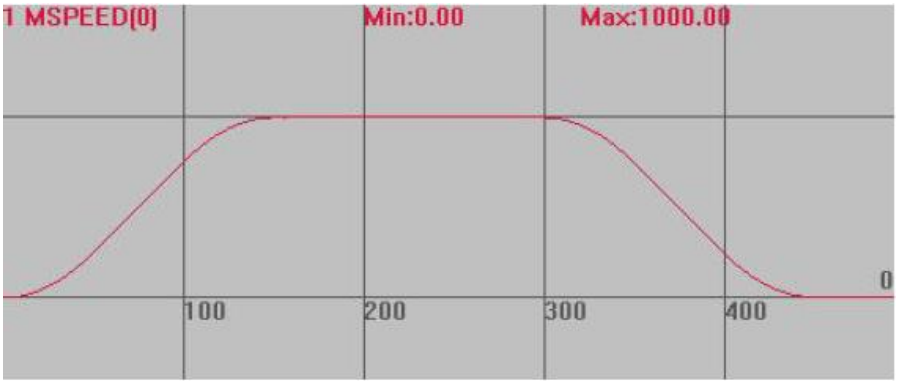
2. S Curve

By setting the value of SRAMP to set the appropriate rate of change of acceleration and deceleration, the speed curve will be smooth, and the jitter is reduced during mechanical start-stop or acceleration and deceleration. The range of SRAMP value is between 0-250 milliseconds. After setting, the acceleration and deceleration process will be longer correspondingly. The longer the time, the smoother the speed curve. If the setting time exceeds 250 milliseconds, it will be smoothed according to 250 milliseconds.

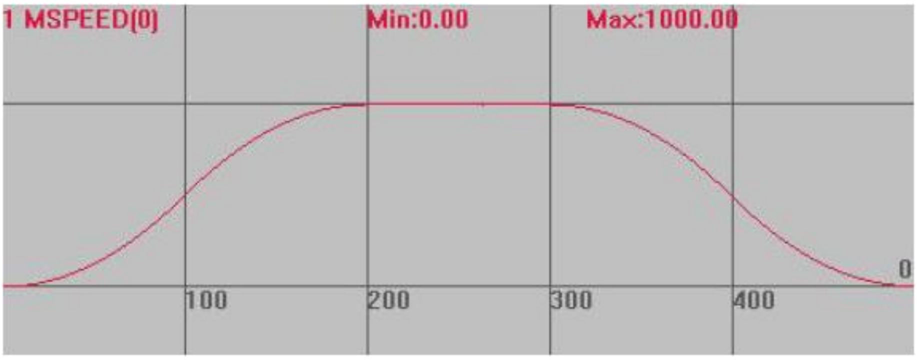
Routine:

```
RAPIDSTOP(2)
WAIT IDLE
BASE(0)
DPOS = 0
MPOS = 0
UNITS = 100
SPEED = 1000
ACCEL = 10000
DECEL = 10000
SRAMP=50
TRIGGER
MOVE(300)
```

When SRAMP=50, obtain the below S curve: it is softer when accelerating and decelerating.

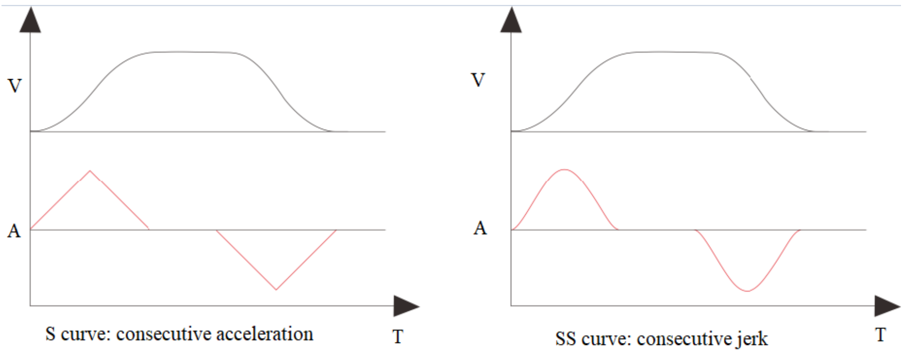


When SRAMP=100, obtain the below S curve: acceleration and deceleration process become longer.



3. SS curve

S curve and SS curve both can smooth the speed parameter, difference refers to below graphics. "jerk" parameter value of S curve is constant in acceleration and deceleration stages, but SS curve makes jerk parameter change according to acceleration and deceleration stages, speed curve is smoother than S curve, which means it can decrease axis shake. SS curve is configured by VP_MODE instruction, there are several modes to be selected.



S curve: consecutive acceleration SS curve: consecutive jerk

Routine: compare S curve with SS curve

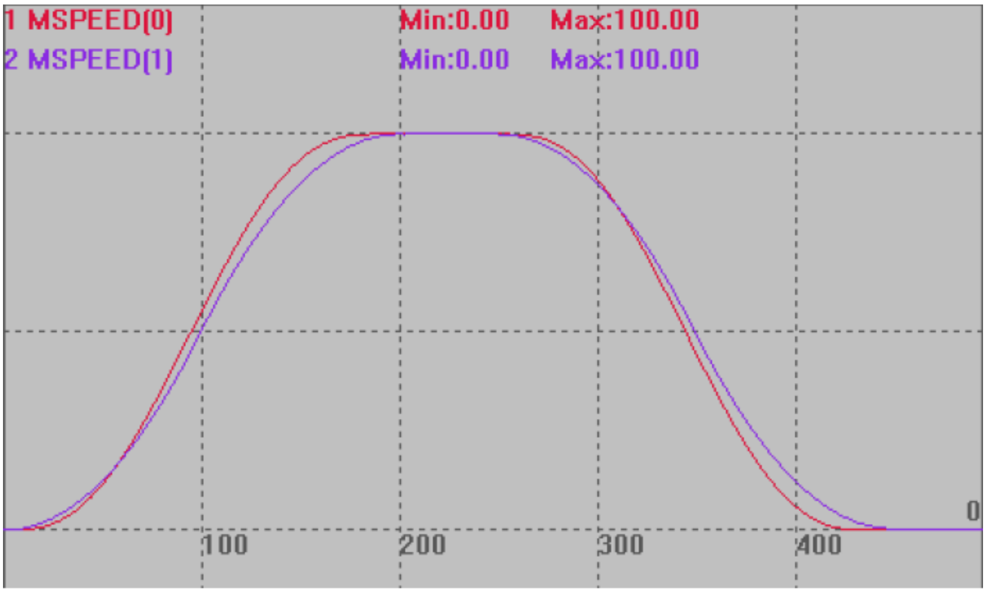
```
BASE(0,1)
ATYPE=1,1
UNITS-100,100
DPOS=0,0
```

```
MPOS=0,0
SPEED=100,100
ACCEL=1000,1000
DECEL=1000,1000
SRAMP=100,100
VP_MODE=7,0
TRIGGER
MOVE(25)   AXIS(0)
MOVE(25)   AXIS(1)
END
```

Speed curve: mode7 processed acceleration and deceleration stages.

MSPEED(0) = 50 (vertical scale), start and end stages of SS curve acceleration and deceleration are smoother.

MSPEED(0) = 50 (vertical scale), S curve.



6.4.2 SP Speed

The SP speed is applied to the interpolated motion commands with SP suffixes (such as MOVESP, MOVECICRSP), and the motion speed uses the FORCE_SPEED parameter instead of the SPEED parameter.

Start speed STARTMOVE_SPEED: the start speed of the SP movement of the custom speed.

End speed ENDMOVE_SPEED: the end speed of the SP movement of the custom speed.

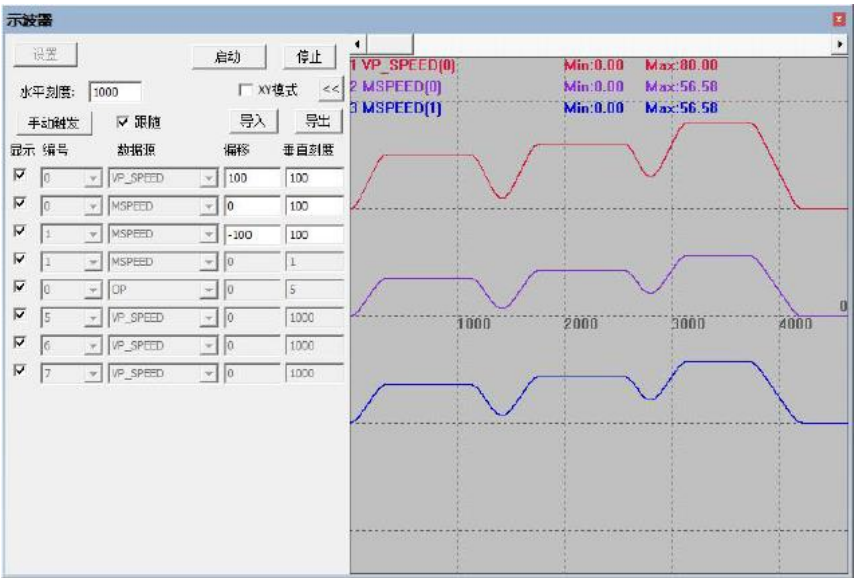
Forced speed FORCE_SPEED: forced speed of SP motion for custom speed.

The above three parameters are valid only when the motion command with SP is used, and all parameters are brought into the motion buffer.

When not in use, please set STARTMOVE_SPEED and ENDMOVE_SPEED to a larger value, otherwise the next motion instruction will continue to use this parameter.

```
RAPIDSTOP(2)
WAIT IDLE(0)
WAIT IDLE(1)
BASE(0,1)                 'select XY axis
DPOS = 0,0
MPOS = 0,0
ATYPE=1,1                 'pulse step or servo
UNITS = 100,100           'pulse equivalent
SPEED = 100,100
ACCEL = 200,200
DECEL = 200,200
SRAMP=100,100             'S curve
MERGE= ON                 'start continuous interpolation
TRIGGER
'the first segment
FORCE_SPEED=50            'the first speed is 50
STARTMOVE_SPEED=20        'the first start speed is 20
ENDMOVE_SPEED=10          'the first end speed is 10
MOVESP(40,40)
'the second segment
FORCE_SPEED=60            'the second speed is 60
STARTMOVE_SPEED=30        'the second start speed is 30
ENDMOVE_SPEED=40
MOVESP(50,50)
'the third segment
FORCE_SPEED=80            'the third speed is 80
STARTMOVE_SPEED=30        'the third start speed is 30
ENDMOVE_SPEED=20
MOVESP(60,60)
END
```

Speed change curve: start moving from speed 0, STARTMOVE_SPEED = 20 of the first segment has no effect, and the end speed of the first segment ENDMOVE_SPEED = 10 means that the first segment of motion is completed after the speed drops to 10. The second segment of motion actually starts the movement at the speed of 10 and end at ENDMOVE_SPEED = 40. The start speed of the third segment is STARTMOVE_SPEED = 30, which is less than the end speed of the second segment of 40. After the second segment is completed, the speed will drop to 30. After the third segment is completed, there is no movement command behind it. So the speed drops to 0 and ENDMOVE_SPEED has no effect.



6.5 Axis Mapping

When using the local pulse axis of the controller, no axis mapping is required, and the default axis number can be used, please refer to the section on axis number description.

When using the bus axis and the extended pulse axis, the bound axis number should be mapped before use. If you want to change the default axis number of the pulse axis, you can remap and configure the axis number. The mapped axis number uses the AXIS_ADDRESS axis mapping command, the grammar for axis mapping is different.

The axis numbers can be mapped at will, but they must be within the range of the number of axes supported by the controller, and the mapped axis numbers cannot be repeated. Generally, they are mapped in sequence, which is not easy to make mistakes. Different types of axis channel numbers are sorted independently, and the axis numbers are all from 0 to start.

Supports mixed interpolation of local pulse axis and EtherCAT axis. After the axis number is mapped, the extended axis resource can be called.

The axis number of EtherCAT and the axis number of the local pulse axis are independent coding sequences. For example, in a certain configuration, two local pulse axes and two EtherCAT axes need to be used. The axis mapping relationship during configuration is as follows:

AXIS 0——local pulse axis 0

AXIS 1——local pulse axis 1

AXIS 2——EtherCAT axis 0

AXIS 3——EtherCAT axis 1

Before configuration, set AXIS 0-3 as virtual axis ATYPE=0, and then use AXIS_ADDRES instruction to map the axis number of the drive. After the configuration is completed, configure ATYPE according to the characteristics of the axis, and then send commands to axes 0-3.

The default configuration file is configured according to the total number of channels of the connected hardware resources. If the hardware resources are greater than the software resources, the default mapping is to map all the software resources to the corresponding hardware resources in sequence, and the redundant unmapped hardware resources are uncontrollable.

Note that multiple motors can be connected to a multi-axis drive, one motor represents one axis, and each motor requires axis number mapping, which is equivalent to that the drive can control multiple axes.

6.6 Axis Type

Use the ATYPE instruction to configure the axis type according to the characteristics of the current axis. When the user program is initialized, the configuration of the axis type should be completed as soon as possible. If the type does not match, an error will be reported.

All unassigned axes default to virtual axes, and the value of ATYPE is 0.

The axis types supported by the controller are as follows:

Atype Type	Description
0	Virtual axis
1	Servo or stepper of pulse direction
2	Servo analog signal control method
3	Quadrature encoder
4	output in pulse direction + quadrature encoder input
5	output in pulse direction + quadrature encoder input in pulse direction
6	Encoder of pulse direction
7	Servo or stepper of pulse direction + EZ input
8	Servo or stepper of pulse direction through ZCAN
9	Quadrature encoder through ZCAN
10	Encoder of pulse direction through ZCAN
20	Galvanometer type with galvanometer status feedback. If galvanometer links with AXISSTATUS bit2 unsucessfully, it will set,

	ENCODER returns to the original sending position, pulse unit. ZMC408SCAN supports.
21	Galvanometer axis type, it needs support of controller. The default system period is 250us, galvanometer refresh period is 50us, which are related to firmware. All motion control instructions of ordinary axes can be used, and support galvanometer axis mixed interpolate with other axis types.
22	Galvanometer type with galvanometer status feedback. If galvanometer links with AXISSTATUS bit2 unsucessfully, galvanometer warnning AXISSTATUS bit3 will set. MPOS returns to reflection position, and does the reverse correction. ENCODER returns to original feedback position, pulse unit. ZMC408SCAN supports.
24	Remote encoder axis type. ZHD 500X handwheel using, need 5 series controllers with firmware version above 20180404.
50	RTEX period position mode, available in RTEX controller
51	RTEX period speed mode, available in RTEX controller.
52	RTEX period torque mode, available in RTEX controller. Do close 2 DOF mode in connected drive, and set speed limit.
65	ECAT period position mode, available in EtherCAT controller
66	ECAT period speed mode, available in EtherCAT controller. DRIVE_PERIOD should be set as 20 or above.
67	ECAT period torque mode, available in EtherCAT controller. DRIVE_PERIOD should be set as 30 or above..
70	Sef-defined ECAT mode, only read encoder value. available in EtherCAT controller.

1. ATYPE=0 Virtual axis

It can be the main axis when in the multi-axis synchronization motion, and slave axes all follow this virtual axis.

As the superposition axis for other axes, it superposes a virtual axis to axes that really move. These virtual axes can be set through ADDAX command (axis superposition), then the motion of each virtual axis is superposed to actual-axis.

2. ATYPE=1 or 7 Pulse axis

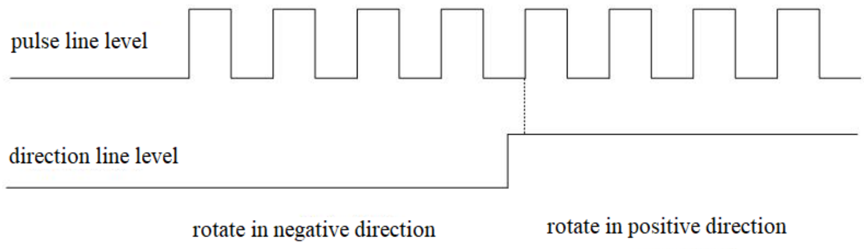
The motion of axis is controlled by pulse sent from the controller, and the direction of pulse determine the direction of motor rotation. The axis motion speed (fast or slow) is controlled according to frequency for sending pulse.

There are 3 modes of controller pulse output: pulse + direction, dual-pulse, orthogonal pulse. They are configured through INVERT_STEP instruction, the default is pulse + direction mode.

1) pulse + direction mode

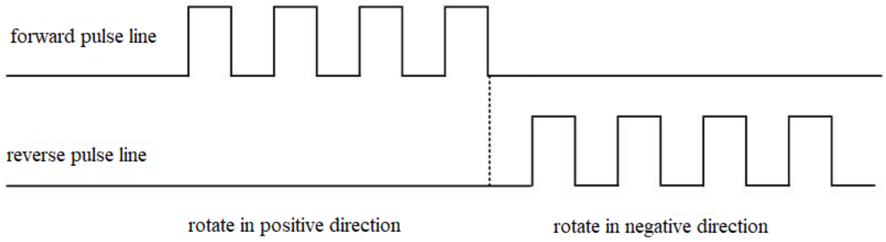
PUL+, PUL- output instruction pulse string, the number of pulse is relative to motion running distance, and the pulse frequency is relative to motion running speed.

DIR+, DIR- output direction signal, different levels of this signal are relative to different rotation direction. This mode occupied the most in drive.



2) CW/CCW: dual-pulse work mode

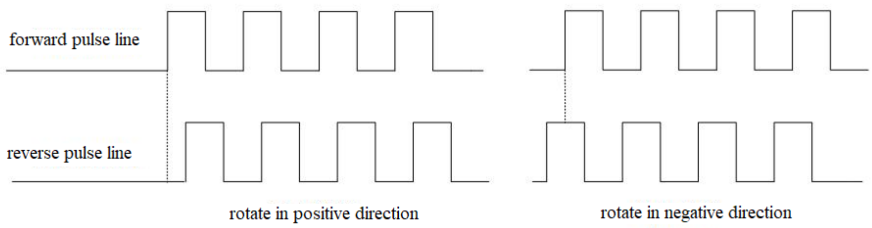
Two lines both output pulse signal, CW means output pulse signal in positive direction, CCW means output pulse signal in negative direction. Usually, they are differential output, the phase difference angle between the two signals is determined by the phase lead or lag.



3) AB Phase: orthogonal pulse work mode

It refers to two identical pulse signals (both are square waves) that are independent of each other. The positive direction pulse signal is generated before the negative direction pulse signal, and the phase difference between the two is 90 degrees. At this time, it is a positive rotation. The negative direction pulse signal is generated before the positive direction pulse signal, and the two are 90 degrees out of phase, which is negative rotation at this time.

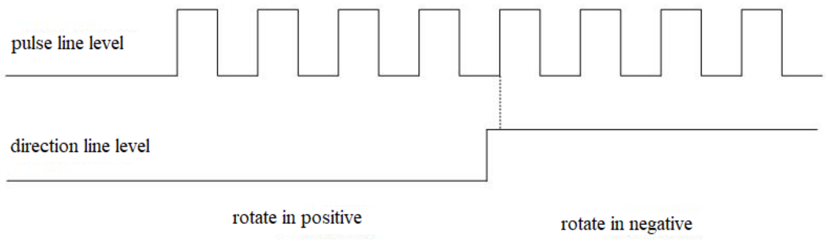
The function of counting or encoding is achieved by the phase difference between the two pulses.



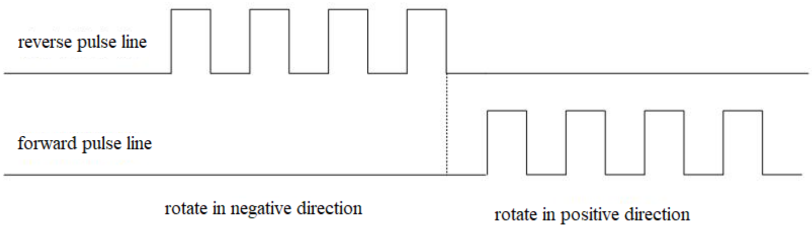
Polarity reversal

If the positive and negative of the pulse line are switched, that is, the original positive direction pulse signal becomes a negative direction pulse signal, and the negative direction pulse signal becomes a positive direction pulse signal, and the movement direction at this time will be opposite to the above situation.

1) pulse + direction mode



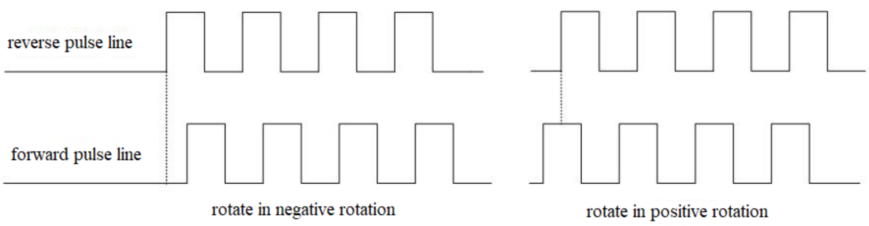
2) CW/CCW: dual-pulse work mode



3) AB Phase: orthogonal pulse work mode

The criterion for judging the rotation direction in this mode is to observe which direction sends out the pulse signal first, and the rotation direction is the negative direction.

The negative direction pulse signal is generated before the positive direction pulse signal, and the phase difference between the two is 90 degrees. At this time, it is a negative direction rotation. The positive direction pulse signal is generated before the negative direction pulse signal, and the phase difference between the two is 90 degrees. At this time, it is a positive direction rotation.



In the above modes, pulse + direction mode and dual-pulse two polarities are related to 8 different motion states. AB phase/BA phase mode is the customized mode of some controllers (ZMC4XX series or above).

3. ATYPE=3 or 6 encoder axis

When encoder separately occupies one axis number, axis type can be selected as 3 or 6 according to encoder type.

4. ATYPE=4 pulse axis and encoder axis share the axis number

When the current pulse axis with encoder feedback, the axis type is set as 4, and the signal output by pulse and the signal input by encoder both are on the same axis number.

5. ATYPE=8 CAN expand axis

When expanding axis through CAN bus, set the axis type of expanded pulse axis as 8, and set the axis type of connected encoder axis on expanded axis as 9.

6. ATYPE=21 galvanometer axis number

When galvanometer equipment is connected, the axis type of galvanometer should be set 21, and galvanometer axis is supported by some models.

7. ATYPE=50,51,52 RTEX bus axis number

When using the RTEX bus driver, the axis type can only be selected from the above three, among which ATYPE=50 is the position mode, the motion command is used to control the motor running. ATYPE=51 speed mode in the speed mode, the DAC command is used to set the running speed of the motor, and continue to run. ATYPE=52 torque mode uses DAC command to set the motor torque in torque mode, and continue to run, motion command cannot be used in speed and torque mode, so there is no need to set axis parameters, stop running with DAC=0.

To switch modes in speed and torque mode, in order to prevent accidents, first set the DAC to 0 and then use the ATYPE command to switch.

Note: Before modifying ATYPE to switch to torque mode, please set the first position of the drive parameter Pr6.47 to 0 and turn off the 2-DOF control mode. Then set the speed limit through parameter Pr3.17. When the set value of Pr3.17 (speed limit selection) is 0, set the speed limit through Pr3.21, and when the set value is 1, you can switch between Pr3.21 or Pr3.22 for the speed limit value during torque control through SL_SW.

8. ATYPE=65,66,67 EtherCAT bus axis number

When using the EtherCAT bus driver, the axis type can only be selected from the above three, among which ATYPE=65 is the position mode, the motion command is used to control the motor operation. ATYPE=66 speed mode is in the speed mode, the DAC command is used to set the running speed of the motor, and continue to run, there are two speed units, the number of pulses /S and R/MIN are determined by the drive. ATYPE=52 is torque mode, using the DAC command to set the torque of the motor in torque mode, and continue to run, the range of DAC value is 0-1000 in torque control mode, corresponding to 0-100%, such as DAC=10, the motor torque is 1% at this time, and motion commands cannot be used in speed and torque mode, so there is no need to set axis parameters, and DAC=0 to stop running.

To switch modes in speed and torque mode, in order to prevent accidents, first set the DAC to 0 and then use the ATYPE command to switch.

Chapter VII Motion Instructions

When the current motion command is being executed, the subsequently called motion commands will be automatically buffered. Each axis of the ZMotion motion controller can support up to 4096 levels of motion buffers (the number of buffers varies with different models of controllers). When all the buffers are occupied, the subsequent call of the motion instruction will block the current task, and the task will continue to run until there is a space in the buffer.

Each motion instruction has a MOVE_MARK parameter, and which motion buffer is currently running can be known through MOVE_CURMARK.

Single-axis motion commands such as MOVE use the axis parameters of the respective single-axis, such as SPEED of this axis.

The multi-axis interpolation motion commands such as MOVE use the SPEED and other axis parameters of the BASE spindle as the vector composite speed, but they have corresponding SP commands, which can specify various speed parameters for each movement, such as, FORCE_SPEED, STARTMOVE_SPEED, ENDMOVE_SPEED, see the corresponding *SP instruction.

The axis parameter MERGE is used to set whether to decelerate to zero in the middle of the single-axis positioning or multi-axis interpolation command of the axis group. When MERGE=OFF, it decelerates to 0. When MERGE=ON, it does not decelerate. At this time, the axis parameter CORNER_MODE of the BASE spindle will set more than one value. Whether to automatically decelerate to the necessary speed between axis interpolations.

ZMotion motion controller supports motion pause or resume of single-axis or axis group, refer to MOVE_PAUSE, MOVE_RESUME.

ZMorion motion controller supports motion superposition, refer to ADDAX.

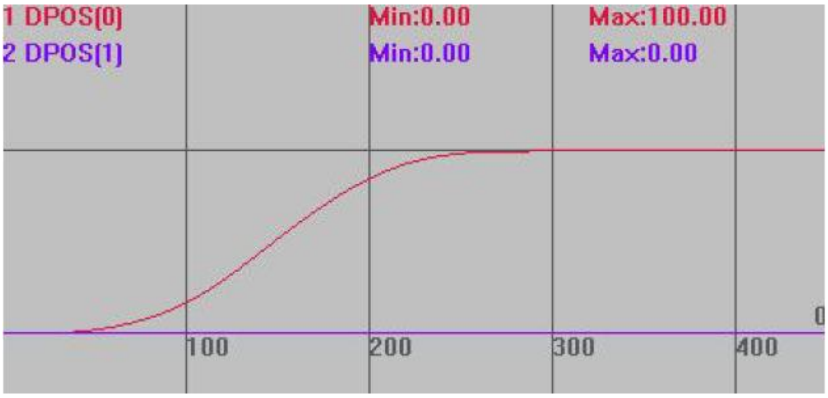
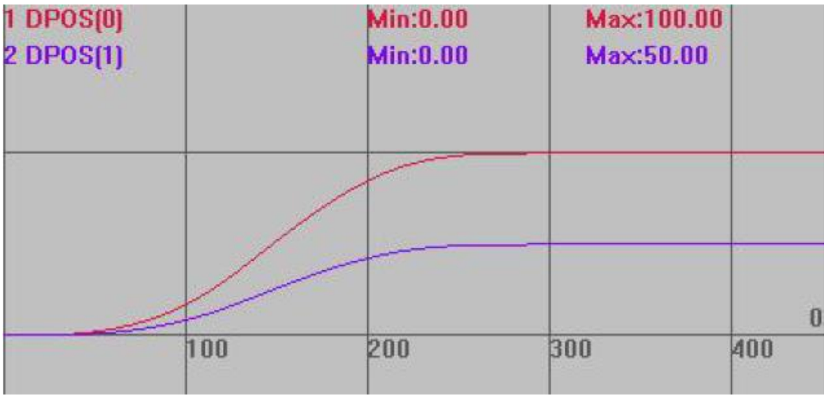
7.1 Single-axis Motion Instructions

ADDAX -- Motion Superposition

Type	Single Axis Motion Instruction

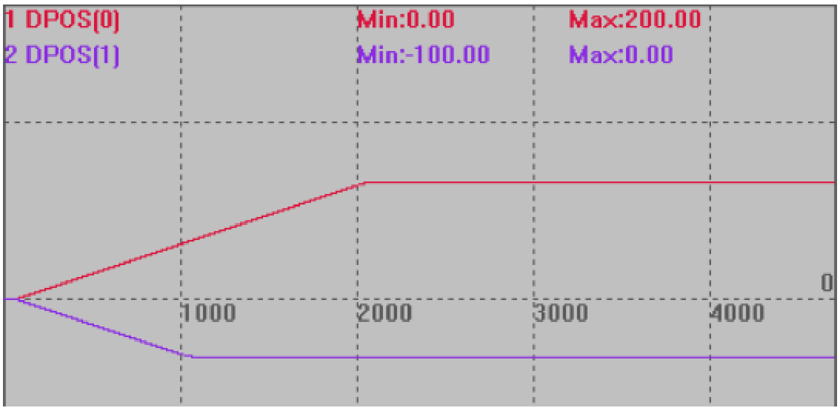
Description	**Motion superposition: add motion of one axis to another axis.** When using ADDAX to realize superposition. the added value is not units but pulse amount. **Conversion relationship:** Distance of superimposing axis *unites of superimposing axis /unites of superimposed axis = distance of superimposed axis For example: If UNITS of axis A equals to 100, and UNITS of Axis B equals to 50, and the superposition axis moves 100. Situation 1: add motion of axis A to axis B, now showing Axis A moves 100, then the axis B moves 100*100/50=200 Situation 2: add motion of axis B to axis A, now showing axis B moves 100, then axis A moves 100*50/100=50. Motion can not be added to each other simultaneously between 2 axes, when add motion of axis A to axis B, then add motion of axis B to axis A simultaneously is not allowed. Support series superposition, motion A superimposes to B, B is superimposed to C. Support parallel superposition, motion A is superimposed to B and C at the same time. When superimposing, the speed starts to change from the superimposed axis, and the acceleration and deceleration are determined according to the superimposed axis acceleration and deceleration and the ratio of the units of the two axes. ADDAX has no effect when the axis MTYPE is FRAME or REFRAME.
Grammar	Superposition: ADDAX (superposing axis No.) AXIS (superposed axis No.) Cancel superposition: ADDAX(-1) AXIS (superposed axis No.) This superposition is added in controllers above 4xx series with 20220708 firmware version or above. ADDAX(srcaxis ,[imode], [para]) destaxis: the superposed target axis number srcaxis: the superposed axis number of the source axis imode: superposition mode 0: default value, single-axis superposition, compatible with previous direct pulse number superposition 1: single-axis superposition, support scale adjustment. ADDAX(srcaxis, 1, ratio) ratio: ratio value, supports floating point numbers, target axis distance = source axis distance * ratio. 2: single-axis superimposition, supports gear ratio adjustment ADDAX(srcaxis, 2, ratioin, ratioout) ratioin: numerator, integer, supports negative numbers ratioout: denominator, positive integer.

	target axis distance = source axis distance * ratioin / ratioin 3: single axis superimposed to two axes, support angle adjustment BASE(destaxis1, destaxis2) ADDAX(srcaxis, 3, angle) destaxis: the superposed target axis 1, 2 angle: angle, radian value, target axis 1 distance = source axis distance * cos(angle). target axis2 distance = source axis distance * sin(angle). Note: If needs to cancel, cancel the two axes ADDAX(-1, 3, 0) or ADDAX(-1) AXIS (the superposed axis No.) respectively 4: SCAN linkage superposition, use SCAN axis to compensate the deviation of platform axis, and their directions and amounts must be consistent, if not, please adjust gear ratio or add ratio for SCAN correction. BASE(destaxis, destaxis2) ADDAX(srcaxis, 4, srcaxis2) Use srcaxis to compensate destaxis, use srcaxis2 to compensate destaxis2. Note: two axes should be cancelled together, ADDAX(-1, 4, -1) or ADDAX(-1) AXIS (superposed axis No.) 5: SCAN linkage superposition, platform axis is superposed at SCAN axis, their directions and amounts must be consistent, if not, please adjust gear ratio or add ratio for SCAN correction. BASE(destaxis, destaxis2) ADDAX(srcaxis, 5, srcaxis2) srcaxis is superposed at destaxis, srcaxis2 is superposed at destaxis2. Note: two axes should be cancelled together, ADDAX(-1, 5, -1) or ADDAX(-1) AXIS (superposed axis No.)
Controller	General
Example	**Example 1:** BASE(0,1) ATYPE=1,1 UNITS=100,200 'set UNITS of axis 0 as 100, and axis 1 as 200 SPEED=1000,1000 'set speed as 1000 ACCEL=10000,10000 'set acceleration as 10000 DECEL=10000,10000 'set deceleration as 10000 **ADDAX**(0) AXIS(1) 'add motion of axis 0 to axis, superpose according to the number of pulse DPOS=0,0 'set position as 0,0 TRIGGER 'trigger oscilloscope automatically MOVE(100) 'axis 0 moves 100, axis 1 moves 100*100/200=50 'the switch of UNITS two axes should be considered

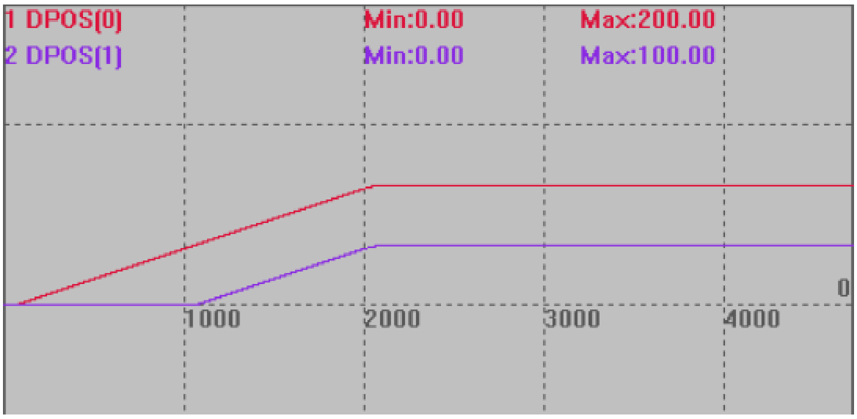
	WAIT IDLE 'wait until motion ends ADDAX(-1) AXIS(1) 'cancel the motion superposition The motion trajectory when the superposition command is not used. (there is no special description in below graphic, which means offset is not configured). DPOS(0) vertical scale 100 DPOS(1) vertical scale 100  The motion trajectory when superposition command is used: DPOS(0) vertical scale 100 DPOS(1) vertical scale 100  **Example 2:** RAPIDSTOP(2) WAIT IDLE BASE(0,1) DPOS=0,0 ATYPE=1,1 UNITS=100,100 ‘pulse proportion is 1:1 SPEED=100,100 ACCEL=1000,1000 DECEL=1000,1000 ADDAX(0) AXIS(1) ‘superpose axis 0 to axis 1 TRIGGER MOVE(200) AXIS(0)

MOVE(-100) AXIS(1)
WAIT IDLE   'wait running ends
ADDAX(-1)   AXIS(1)   'cancel superposition

Before superposition:

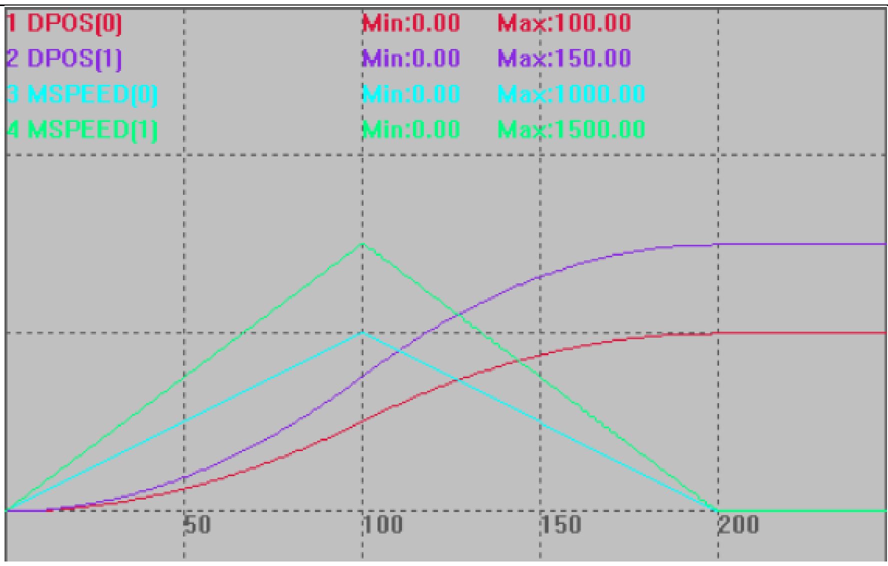


After superposition: following, send motion command for axis 0, axis 0 and axis 1 move together, and keep superposition. Cancel superposition until ADDAX(-1) AXIS(1).



Example 3: mode 1

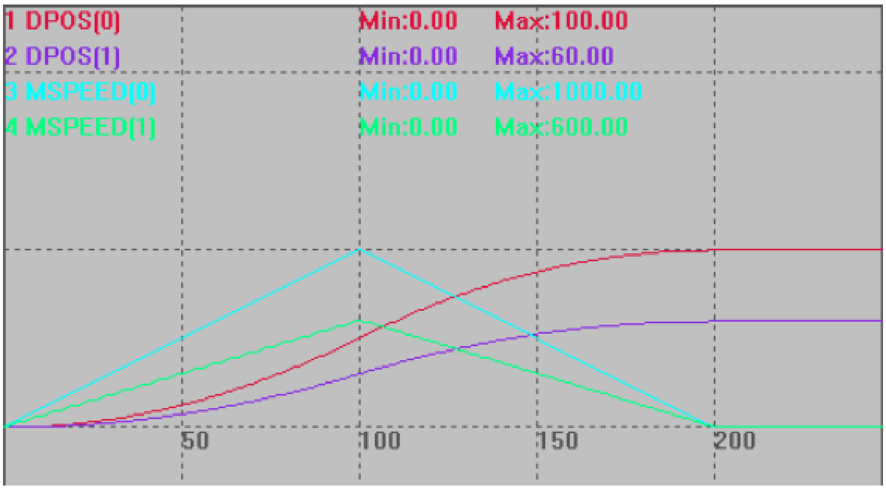
BASE(0,1)   'select axis No.
UNITS = 100,100
DPOS=0,0
TRIGGER
BASE(1)   'select superposed axis
ADDAX(0,1,1.5)AXIS(1)
            'mode 1 superposition, superpose axis 0 to axis 1, the ratio is 1.5
MOVE(100) AXIS(0)
WAIT UNTIL IDLE(0) AND IDLE(1)
?"axis 1 superposing axis No." ADDAX_AXIS(1)
ADDAX(-1) AXIS(1)   'cancel superposition
The pulse amount is the same, axis 1 motion distance is 1.5 times of axis 0.



Example 4: mode 2

BASE(0,1) ‘select axis No.
UNITS = 100,100
DPOS=0,0
TRIGGER
BASE(1) ‘select superposed axis
ADDAX(0,2,3.5) AXIS(1) ‘mode 2 superposition, superpose axis 0 to axis 1
MOVE(100) AXIS(0)
WAIT UNTIL IDLE(0) AND IDLE(1)
?”axis 1 superposing axis No.” ADDAX_AXIS(1)
ADDAX(-1) AXIS(1) ‘cancel superposition
END

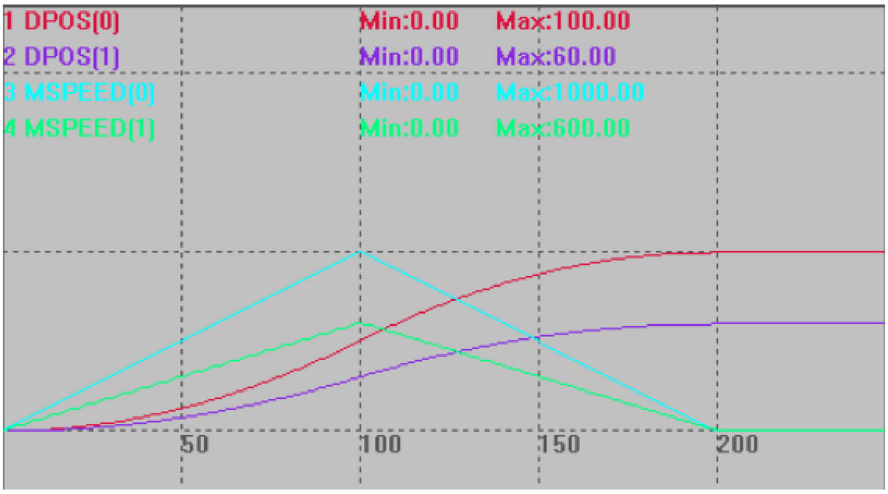
The pulse amount is the same, axis 1 motion distance is 3/5 times of axis 0.



Example 5: mode 3

BASE(0,1,2) ‘select axis No.
UNITS = 100,100,100
DPOS=0,0,0
BASE(1,2) ‘select target No., axis 1 and axis 2

```
TRIGGER
ADDAX(0,3,PI/3)   'mode 3 superposition, superpose axis 0 to axis 1, axis 2
MOVE(100) AXIS(0)
WAIT UNTIL IDLE(0) AND IDLE(1) AND IDLE(2)
DIM pos 1, pos2
pos 1=100*cos(PI/3)
pos 2=100*sin(PI/3)
?"axis 0 target position", 100, "actual position" ENDMOVE(0)
?"axis 1 target position", pos1, "actual position" ENDMOVE(1)
?"axis 2 target position", pos2, "actual position" ENDMOVE(2)
?"axis 1 superposing axis No." ADDAX_AXIS(1)
?"axis 2 superposing axis No." ADDAX_AXIS(2)
ADDAX(-1) AXIS(1)          'cancel superposition
ADDAX(-1) AXIS(2)
```



CANCEL -- Stop Single-Axis / Axis Group

Type	Single Axis Motion Instruction
Description	**Axis defined by "BASE" decelerate to stop, if the BASE axis is involved in interpolation movement, the interpolation movement also stops.** If the defined axis is in the list of BASE, whether CANECL master axis or any axis in BASE axis list, interpolations of axis group all stop. The deceleration of Mode 2 obeys the bigger value between FASTDEC and DECEL. Generally, FASTDEC is set as bigger than DECEL. If there is requirement of calling absolute position after using CANCEL, it needs to use "WAIT IDLE" to wait the movement to stop.

Grammar

CANCEL (mode)

Mode: mode selection

0(default)	Cancel the motion in process
1	Cancel the motion in buffer
2	Cancel motions in process and in buffer, stop speed refers to fast deceleration “FASTDEC”.
3	Stop pulse delivery immediately
4	Cancel motions in process and in buffer, stop speed refers to deceleration “DECEL”.

CANCEL (4) is valid in ZMC4XX series controllers whose firmware version is above 170708.

CANCEL (3) can’t be used for the slave axis that is in interpolation.

Controller

General

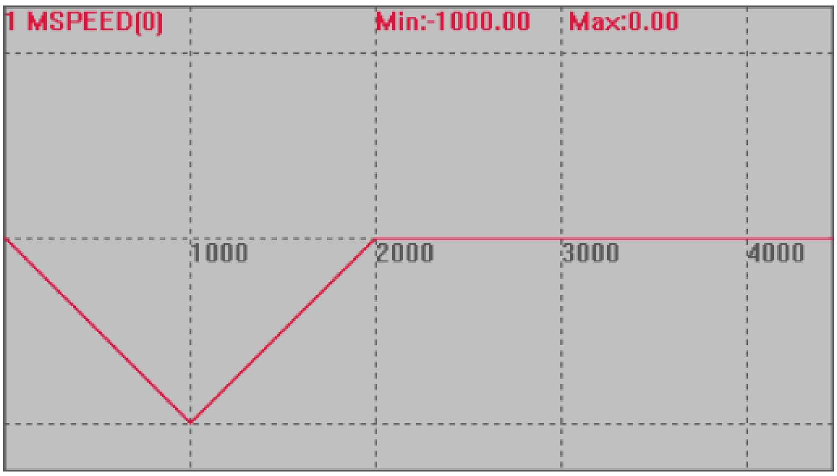
Example

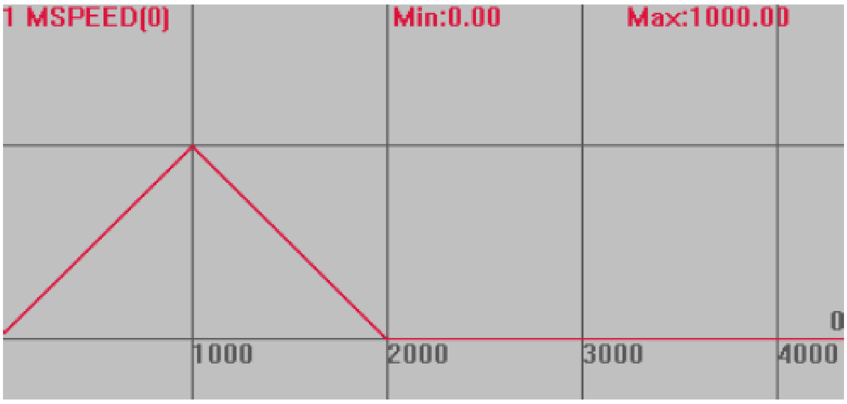
Example 1: mode = 0

```
BASE(0)
DPOS=0
SRAMP=0
ATYPE=1
UNITS=100
SPEED=1000
ACCEL=1000
DECEL=1000          'set deceleration as 1000
FASTDEC=10000       'set fast deceleration as 10000
TRIGGER             'trigger oscilloscope automatically
MOVE(1000)          'motion in process
MOVE(-1000)         'motion in buffer
CANCEL(0)           'axis will only execute MOVE(1000)
```

Motion trajectory:

MSPEED(0) Vertical scale 1000



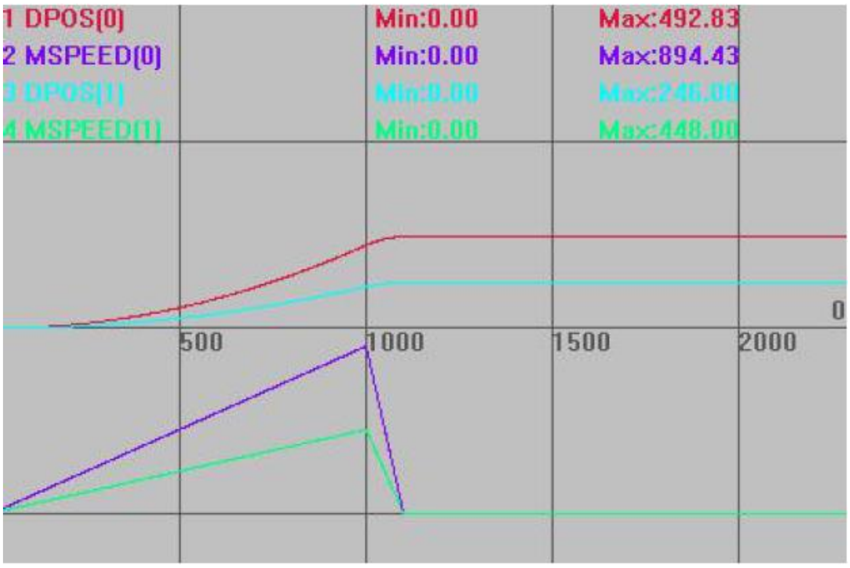
	Example 2: mode = 1 BASE(0) DPOS=0 SRAMP=0 ATYPE=1 SPEED=100 ACCEL=1000 DECEL=1000 'set deceleration as 1000 FASTDEC=10000 'set fast deceleration as 10000 TRIGGER 'trigger oscilloscope automatically MOVE(1000) ‘motion in process DELAY(-1000) ‘motion in buffer **CANCEL**(1) 'axis will only execute MOVE (1000) Motion trajectory: MSPEED(0) Vertical scale 1000  Current motion still runs at deceleration speed to stop because it only cancels the buffer motion. Example 3: mode = 2 BASE(0,1) DPOS=1,1 ATYPE=1,1 SPEED=1000,1000 ACCEL=1000 DECEL=1000 'set deceleration as 1000 FASTDEC=10000 'set fast deceleration as 10000 SRAMP=0,0 TRIGGER MOVE(1000,500) 'interpolation movement DELAY(1000) 'delay 1 second **CANCEL**(2) AXIS(1) 'axis 1 stops, axis 1 was involved in interpolation, the interpolation also stops, and deceleration is 10000. Motion trajectory and speed curve:

DPOS(0) Vertical scale 1000, no offset

MSPEED(0) Vertical scale 1000, offset -1000

DPOS(1) Vertical scale 1000, no offset

MSPEED(1) Vertical scale 1000, offset -1000



Example 4: mode = 3

BASE(0)

ATYPE=1

DPOS=0

SPEED=100

ACCEL=1000

DECEL=1000 ‘set deceleration as 1000

FASTDEC=10000 ‘set fast deceleration as 10000

TRIGGER ‘trigger oscilloscope automatically

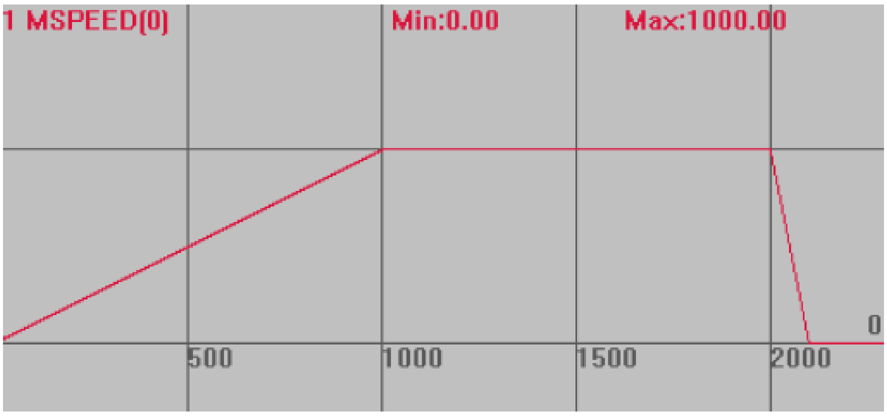
MOVE(10000) ‘the current motion 10000

DELAY(2000) ‘delay 2 seconds

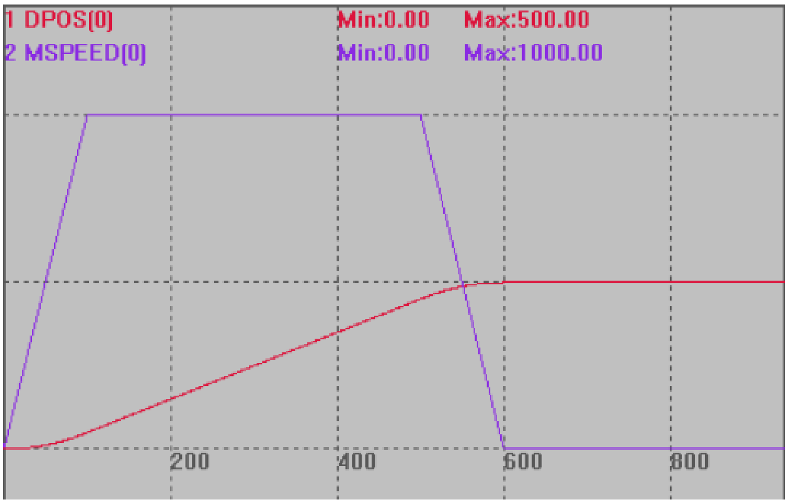
CANCEL(3) ‘now directly stop sending pulse, axis stops immediately

Motion Trajectory

MSPEED (0) vertical scale 1000



Example 5: mode = 4

	BASE(0) DPOS=0 ATYPE=1 UNITS=100 SPEED=1000 ACCEL=10000 DECEL=10000 ‘set deceleration as 10000 FASTDEC=10000 ‘set fast deceleration as 100000 TRIGGER ‘trigger oscilloscope automatically MOVE(1000) ‘the current motion DELAY(-2000) ‘the motion in buffer DELAY (500) **CANCEL(4)** ‘emergency stop, deceleration is 10000 Motion Trajectory DPOS (0) vertical scale 500, no offset MSPEED (0) vertical scale 500, no offset 
Instructions	RAPIDSTOP , DECEL, FASTDEC

DATUM – Homing

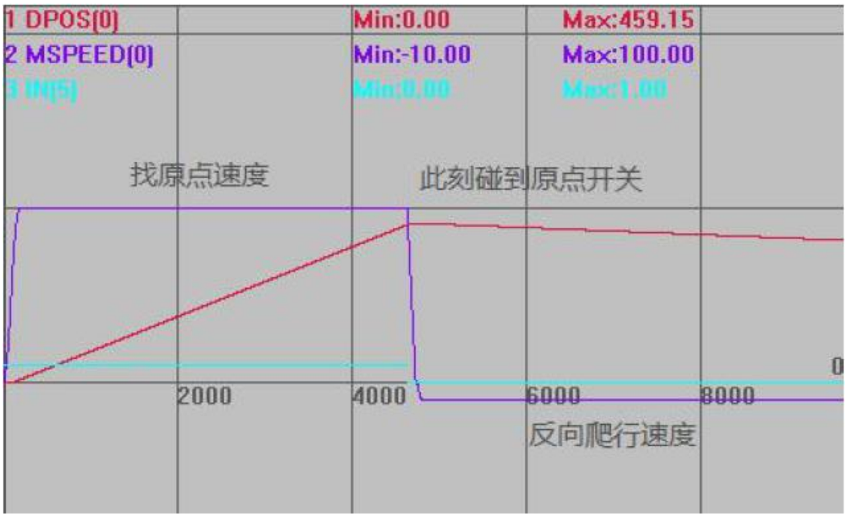
Type	Single Axis Motion Instruction
Description	**Origin (home position or zero position) finding movement of single axis.** Origin switch is set by DATUM_IN, plus-minus switches are set by FWD_IN and REV_IN respectively. Inputs of ZMC motion controller are effective when they are 0, when the input is OFF, it indicates the movement reaches origin or limit position. For common-opened signal, the signal electrical level is needed to be reversed by using **INVERT_IN.** Inputs of ECI motion controller are effective when they are 1, when the input is ON, it indicates the movement reaches origin or limit position. For

	common-closed signal, the signal electrical level is needed to be reversed by using **INVERT_IN.** When using Z signal to trigger origin position finding, **ATYPE**(ATYPE=4/7) should be configured to the mode which contains Z signal. When LSPEED is configured, it will stop emergency when found the origin, and the position that decelerated to LSPEED is the origin position. When multi-axis finds the origin position, every axis should use DATUM instruction. In terms of BUS (EtherCAT or RTEX) motion controller, after using DATUM to find origin position, the relevant MPOS should be cleared by manual.
Grammar	DATUM (mode), DATUM (21, mode2) Mode: zero position finding mode, when using "mode+10", it means the axis will move backward to find zero position after reaching the limit position, it will not stop, such as, if mode=13, 13=mode 3 + move backward 10, this is valid when the origin position is in the center. When ATYPE=4, homing mode plus 100 (mode 100+n and 110+n corresponds n and 10+n), indicating the relevant MPOS will be cleared automatically after linking the encoder (only ZMC4XX series controller support). DATUM (0) AXIS (Axis No.) to clear assigned axis' error state.

Value	Description
0	Clear error states of all axes.
1	Axis runs forward at the speed of CREEP until signal Z appeared, it will directly stop when meeting limit switch. DPOS value will be reset to 0, at the same time, correct MPOS.
2	Axis runs reverse at the speed of CREEP until signal Z appeared, it will directly stop when meeting limit switch. DPOS value will be reset to 0, at the same time, correct MPOS.
3	Axis runs forward at the speed of SPEED, until meeting origin switch, then axis runs reverse at the speed of CREEP until away from origin switch. When in the finding origin process, it will directly stop when meeting positive limit switch, when in the creeping process, it will directly stop when meeting negative position limit. DPOS value will be reset to 0, at the same time, correct MPOS.
4	Axis runs reverse at the speed of SPEED, until meeting origin switch, then axis runs forward at the speed of CREEP until away from origin switch. When in the finding origin process, it will directly stop when meeting negative limit switch, when in the creeping process, it will directly stop when meeting positive position limit. DPOS value will be reset to 0, at the same time, correct MPOS.
5	Axis runs forward at the speed of SPEED, until meeting origin

<table>
<tr><td></td><td>
<table>
<tr><td></td><td>switch, then axis runs reverse at the speed of CREEP until away from origin switch.
Then, keep moving at CREEP speed reversely until meeting signal Z. It stops immediately when met limit switch.
DPOS value will be reset to 0, at the same time, correct MPOS.</td></tr>
<tr><td>6</td><td>Axis runs reverse at the speed of SPEED, until meeting origin switch, then axis runs forward at the speed of CREEP until away from origin switch.
Then, keep moving at CREEP speed forward until meeting signal Z. It stops immediately when met limit switch.
DPOS value will be reset to 0, at the same time, correct MPOS.</td></tr>
<tr><td>8</td><td>Axis runs forward at speed of SPEED, until meeting origin switch, it will stop immediately when met limit switch.</td></tr>
<tr><td>9</td><td>Axis runs reverse at speed of CREEP, until meeting origin switch, it will stop immediately when met limit switch.</td></tr>
<tr><td>21</td><td>Use EtherCAT drive homing function, now mode 2 is valid.
Set drive homing mode (6098h), default 0 means using drive current homing mode. Using axis SPEED, CREEP, ACCEL and DECEL to multiple UNITS, then automatically set drive 6099h and 609Ah.
Action sequence: 6098 homing mode – 6099 speed – 609A acceleration – 6060 switch to current mode.</td></tr>
</table>
Mode2: it is valid when mode=21, default value is 0. When it is not 0, set it as drive homing mode, the value is set according to drive manual data dictionary 6098h.
</td></tr>
<tr><td>Controller</td><td>General</td></tr>
<tr><td>Example</td><td>Example 1　find origin directly.
BASE(0)
DPOS=0
ATYPE=1
SPEED = 100　　'speed when searching for original switch.
CREEP = 10　　'speed when moving backward.
DATUM_IN=5　　'input 5 as original switch signal input.
INVERT_IN(5,ON)　'reverse the electricity level signal of IN5, when signal is normally open.(ZMC series)
TRIGGER　　'trigger the oscilloscope automatically
DATUM(3)　　'axis 0 moves forward to find original switch at speed of 100units/s, then continue to move at speed of 10units/s after reaching the original switch until leave, DPOS reset as 0 at the same time.

Motion trajectory and speed curve:
DPOS(0)　Vertical scale 500
MAPEED(0)　Vertical scale 100
IN(5)　Vertical scale 10</td></tr>
</table>

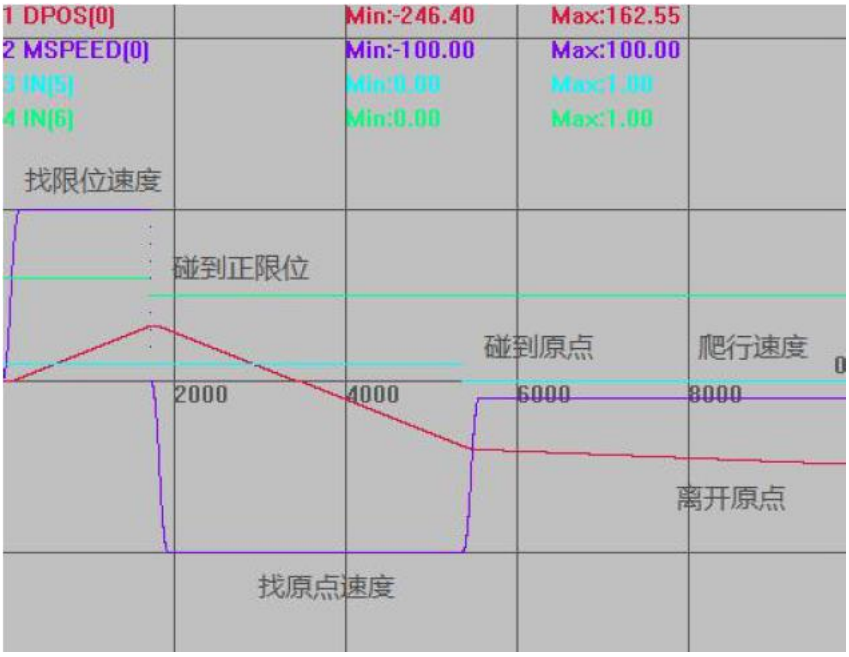


After reaching origin switch, creep backward until leave, at this time, DPOS is cleared as 0, homing movement finished. In order to make it clear, the creep process is longer here. It is very short in the actual applications.

Example 2 Searching for origin reversely after meeting position limit switch.

Base(0)
DPOS=0
ATYPE=1
SPEED = 100 'speed when searching for original switch.
CREEP = 10 'speed when moving backward.
DATUM_IN=5 'input5 as original switch signal input.
FWD_IN=6 'input6 as positive position switch signal input.
INVERT_IN(5,ON) 'Reverse the signal electricity level, often need the common closed signal.
INVERT_IN(6,ON)
Trigger 'Trigger oscilloscope automatically
DATUM(13) 'axis0 moves forward to find original switch at speed of 100units/s, move backward at speed of 10units/s after reaching the original switch until leave, DPOS reset as 0 at the same time.

Motion trajectory and speed curve:
DPOS(0) Vertical scale 500
MAPEED(0) Vertical scale 100
IN(5) Vertical scale 10
IN(6) Vertical scale 10



Example 3 EtherCAT Bus Homing (Panasonic A6N Servo)

Enable the motor according to EtherCAT initialization routine.

SPEED=100 'Homing speed*UNITS, transferred to 6099.

CREEP=10 'Creep speed*UNITS, transferred to 6099.

ACCEL=1000 'Acceleration speed*UNITS, transferred to 609A.

DECEL=1000

DATUM (21,0) 'Start homing as per the present homing mode of drive, now judge according to drive signal, not controller signal.

```
WHILE 1
    TABLE(0)=DRIVE_STATUS  'read the present homing status to judge.
    IF READ_BIT2(10,TABLE(0)) THEN  'depend on below figure to judge
        IF READ_BIT2(12,TABLE(0)) THEN
            ?"homing finished"
        ENDIF
      ENDIF
  WEND
END
```

There are different homing processes of different drive manufacturers, please see drive manual to determine.

Description of bit 13, bit 12, bit 10:

bit 13	bit 12	bit 10	Description
0	0	0	in the motion of homing
0	0	1	The homing motion doesn’t start or interrupts
0	1	0	homing finishes, not achieve target position
0	1	1	Homing finishes normally
1	0	0	Detect that homing is abnormal, it still moves
1	0	1	Detect that homing is abnormal, it stops

Example 4   Rtex Bus homing (Panasonic A6N Servo)

Enable the motor according to Rtex initialization routine.

SPEED=100   'the speed of finding the origin

ACCEL=1000   'acceleration and deceleration

DECEL=1000

DATUM (21, $11) 'Start homing as per the present homing mode of drive, now judge according to drive signal, not controller signal.

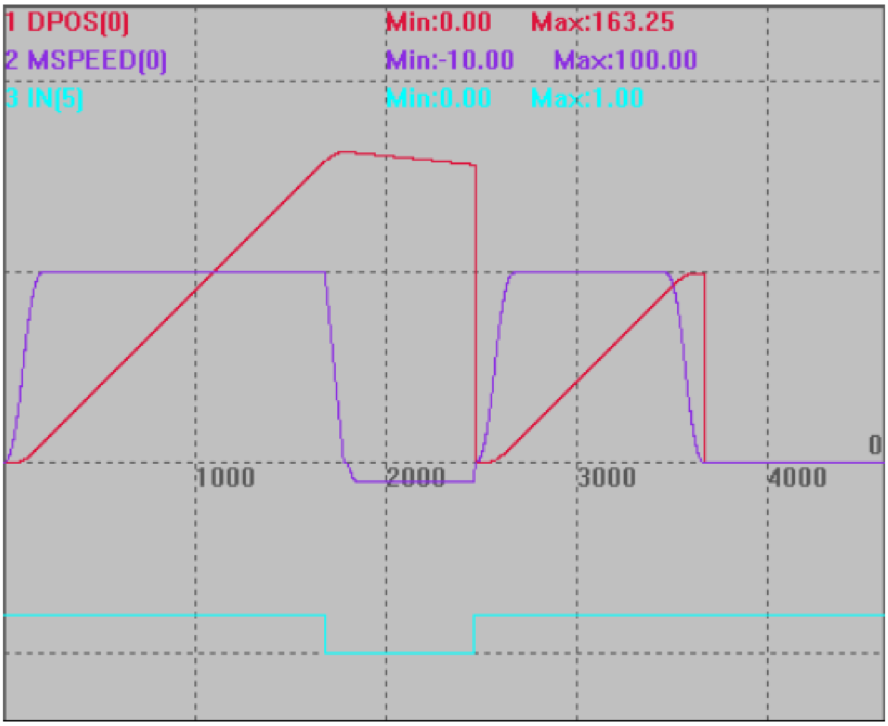
Determine the homing mode according to drive manual

Initialization mode	11h	Z Phase
	12h	HOME ↑ *2
	13h	HOME ↑ *3
	14h	POT ↑ *2
	15h	POT ↑ *3
	16h	NOT ↑ *2
	17h	NOT ↑ *3
	18h	EXIT1 ↑ *2
	19h	EXIT1 ↑ *3
	1Ah	EXIT2 ↑ *2
	1Bh	EXIT2 ↑ *3
	1Ch	EXIT3 ↑ *2
	1Dh	EXIT3 ↑ *3

Instructions	DATUM_IN, INVERT_IN

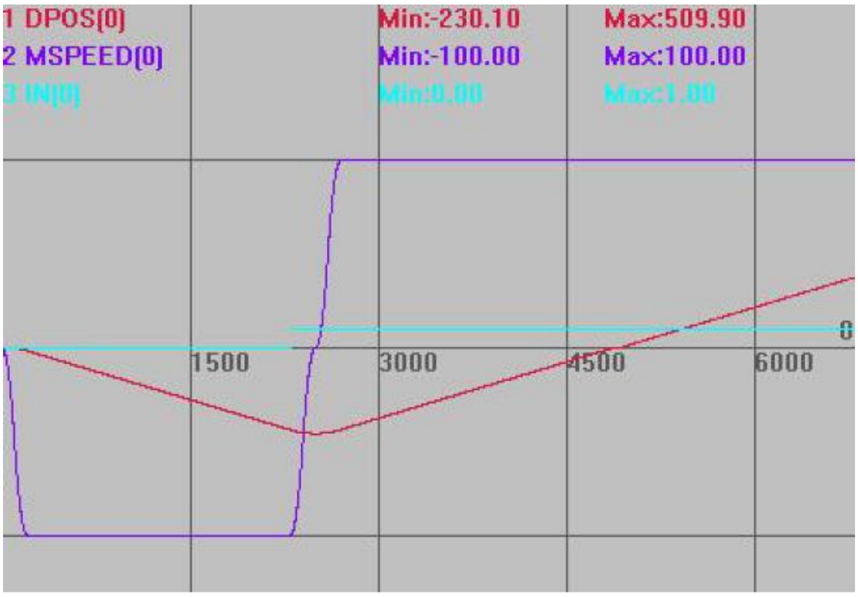
DATUM_OFFSET – Origin Position Offset

Type	Single Axis Motion Instruction
Description	**Set position offset of origin.** **When returned to the origin successfully, axis moves to offset position.**
Grammar	DATUM_OFFSET(axis)=distance distance: offset distance
Controller	Valid in 4xx series controllers and above.
Example	BASE(0) DPOS=0 ATYPE=1 SPEED=100 ‘the speed of finding origin CREEP=10 ‘reveres finding speed DATUM_IN=5 ‘input IN5 as origin switch INVERT_IN(5,ON) ‘reverse IN5 electric level signal, common-opened signal starts to reverse (ZMC controllers) TRIGGER ‘automatically trigger oscilloscope DATUM_OFFSET(0)=100 ‘axis 0 homing, then offset

	DATUM(3) 'axis 0 does homing at the speed of 100units/s firstly, then leaves origin at the speed of 10units/s after finding origin, and clear DPOS as 0 at the same time. Motion trajectory and speed curve: axis stops at the position DATUM_OFFSET finally. DPOS(0) Vertical scale 500 MAPEED(0) Vertical scale 100 IN(5) Vertical scale 5, offset -5 
Instruction	DATUM

VMOVE – Continuous Movement

Type	Single Axis Motion Instruction
Description	**Move in one direction continuously.** There is no need to use "CANCEL" to stop the "VMOVE" movement in advance, the new "VMOVE" movement will automatically replace the former "VMOVE" and modify the direction.
Grammar	VMOVE (dir1) dir1= -1: negative movement 1: positive movement
Controller	General
Example	BASE(0) DPOS=0 ATYPE=1 SPEED=100 ACCEL=1000

	DECEL=1000 'set deceleration as 1000 SRAMP=100 **VMOVE**(-1) 'continuous negative movement WAIT UNTIL IN(0)=ON 'wait until input 1 is on **VMOVE**(1) 'continuous positive movement DPOS(0) Vertical scale 500, no offset MAPEED(0) Vertical scale 100, no offset IN(0) Vertical scale 1000, no offset 
Instructions	FORWARD, REVERSE

FORWARD – positive movement

Type	Single Axis Motion Instruction
Description	**BASE selects axis to move forward.** REVERSE is switched after CANCEL.
Grammar	Forward [axis(axis number)]
Controller	General
Example	Example 1 Base(0) **FORWARD** 'axis 0 move forward continuously WAIT UNTIL IN(1)=ON 'wait until input 1 is on CANCEL(2) Example 2 **FORWARD** AXIS(1) 'axis 1 move forward WAIT UNTIL IN(1)=ON 'wait until input 1 is on CANCEL(2) AXIS(1)
Instructions	REVERSE, VMOVE

REVERSE – negative movement

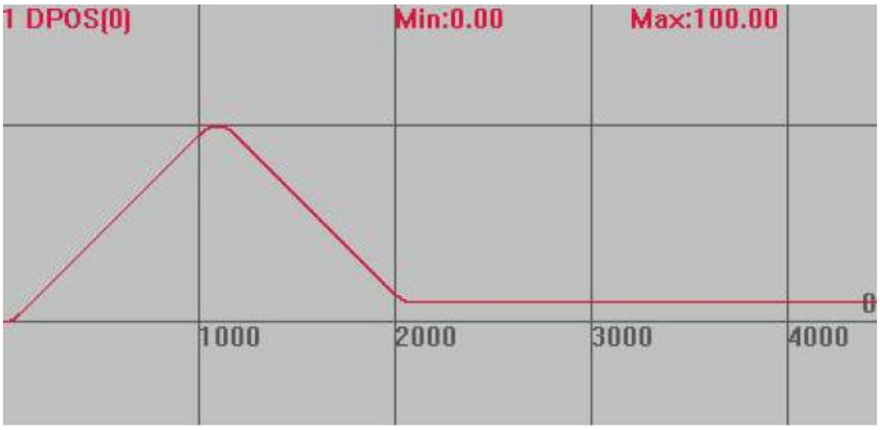
Type	Single Axis Motion Instruction
Description	**BASE selects axis to move reverse.** FORWARD is switched after CANCEL.
Grammar	reverse [axis(axis number)]
Controller	General
Example	Example 1 Base(0) **REVERSE** 'axis 0 move backwards continuously WAIT UNTIL IN(1)=ON 'wait until input 1 is on CANCEL(2) Example 2 **REVERSE** AXIS(1) 'axis 1 move backwards WAIT UNTIL IN(1)=ON 'wait until input 1 is on CANCEL(2) AXIS(1)
Instructions	FORWARD, VMOVE

MOVEMODIFY – Modify Motion Position

Type	Single Axis Motion Instruction
Description	**Change the last motion target position.** The effect is the same as **MOVEABS** when there is no motion before, but it will not enter the motion buffer, see Example 1 for reference. Need WAIT command, see example 2 for reference. If it is continuous interpolation, then use MOVEMODIFY will interrupt the continuity of motion speed. When MOVEMODIFY is used in multi-axis, the motion is not absolutely linear interpolation movement.
Grammar	MOVEMODIFY (distance) distance1: the motion distance of one single axis Only support single axis modification at present.
Controller	General
Example	**Example 1** BASE(0) UNITS=100 'set the pulse amount DPOS=0 SPEED=100 'speed setting ACCEL=1000 'acceleration setting DECEL=1000 TRIGGER 'trigger the oscilloscope automatically MOVEABS(100)

MOVEABS(10)   'axis will move to position 100, then move back to 10.

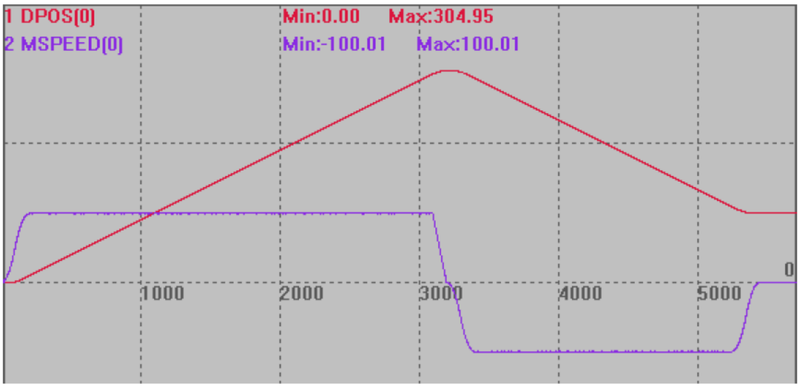
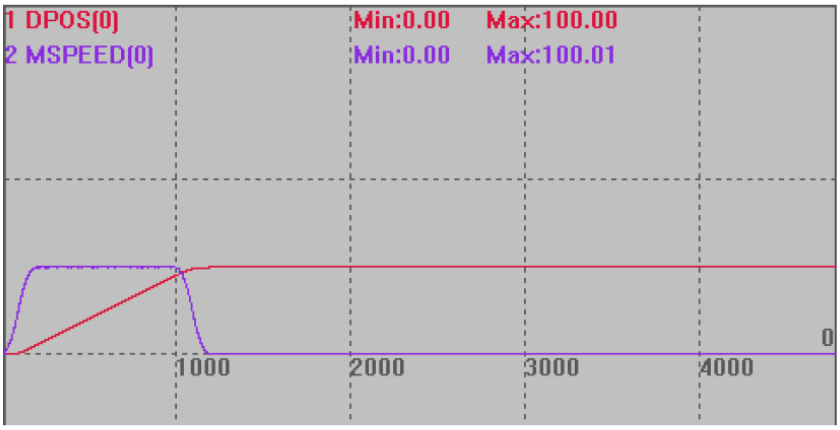
Motion trajectory:
DPOS(0)   Vertical scale 100



If：
MOVEMODIFY(100)
MOVEMODIFY(10)'axis will move to position 10 directly, MOVEMODIFY will not enter the motion buffer.
Motion trajectory:
DPOS(0)   Vertical scale 100

Example 2
BASE (0)
UNITS=100   'pulse equivalent setting
DEFPOS(0)
SPEED=100   'speed setting
ACCEL=1000   'acceleration setting
DECEL=1000
TRIGGER   'trigger the oscilloscope automatically
MOVEABS(500)
WAIT UNTIL DPOS >=300 'modify the target position until axis reaches 300.
MOVEMODIFY (100) 'Change the target position to 100, the axis will decelerate to stop, then move inversely.

Use WAIT, motion trajectory:
DPOS(0) vertical scale 200

	MSPEED(0) vertical scale 200  Not use WAIT, move to position 100 directly. DPOS(0) vertical scale 300 
Instructions	MOVEMODIFY2

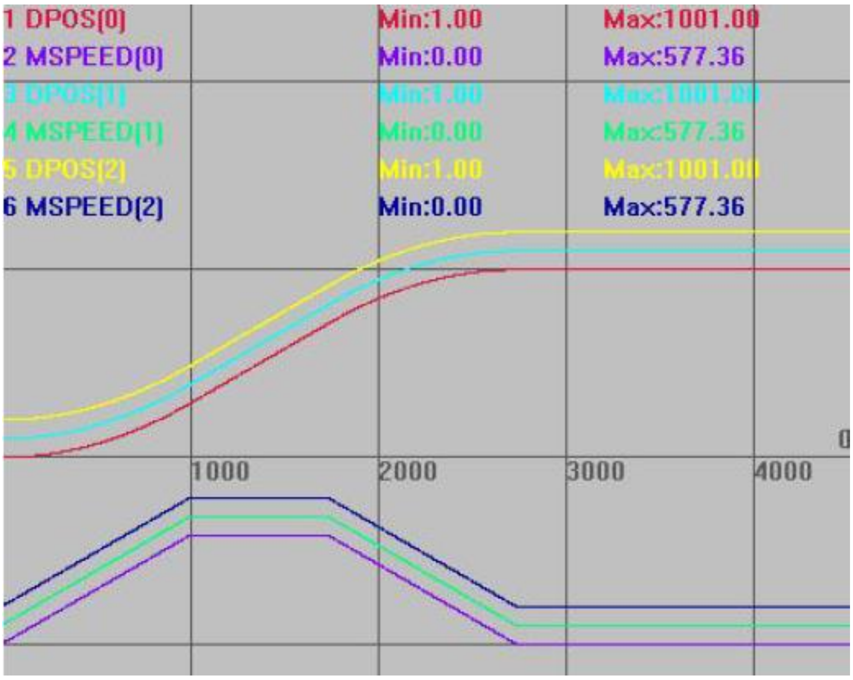
7.2 Multi-axis Motion Instruction

RAPIDSTOP – all axes stop

Type	Multi-Axis Motion Instruction
Description	**All axes stop immediately, if axes were involved in interpolation movement, the interpolation movement also stops.** In the mode 2, deceleration obeys the bigger value between FASTDEC and DECEL. Generally, FASTDEC is set to be bigger than DECEL. If there is a requirement of calling absolute position after using RAPIDSTOP, it needs to use “WAIT IDLE” to wait the movement to stop.

<table>
<tr><td>Grammar</td><td>RAPIDSTOP (mode)
Mode: mode selection
<table>
<tr><td>0(default)</td><td>Cancel motion in process</td></tr>
<tr><td>1</td><td>Cancel motion in buffer</td></tr>
<tr><td>2</td><td>Cancel motions in process and in buffer, stop speed refer to fast deceleration FASTDEC</td></tr>
<tr><td>3</td><td>Stop pulse delivery immediately</td></tr>
<tr><td>4</td><td>Cancel motions in process and in buffer, stop speed refer to deceleration DECEL</td></tr>
</table>
RAPIDSTOP (4) is valid in ZMC4XX series controllers with firmware version 170708 or above.</td></tr>
<tr><td>Controller</td><td>General</td></tr>
<tr><td>Example</td><td>Example 1
BASE(0,1,2)
DPOS=1,1,1
ATYPE=1,1,1
UNITS=100,100,100
SPEED=1000 'interpolated resultant speed is 100
ACCEL=1000
DECEL=1000 'set deceleration as 1000
FASTDEC=10000 'set fast deceleration as 10000
TRIGGER
MOVE(1000,1000,1000) 'motion in process
MOVE(-1000,-1000,-1000) 'motion in buffer
RAPIDSTOP(1) 'axis only executes the current motion

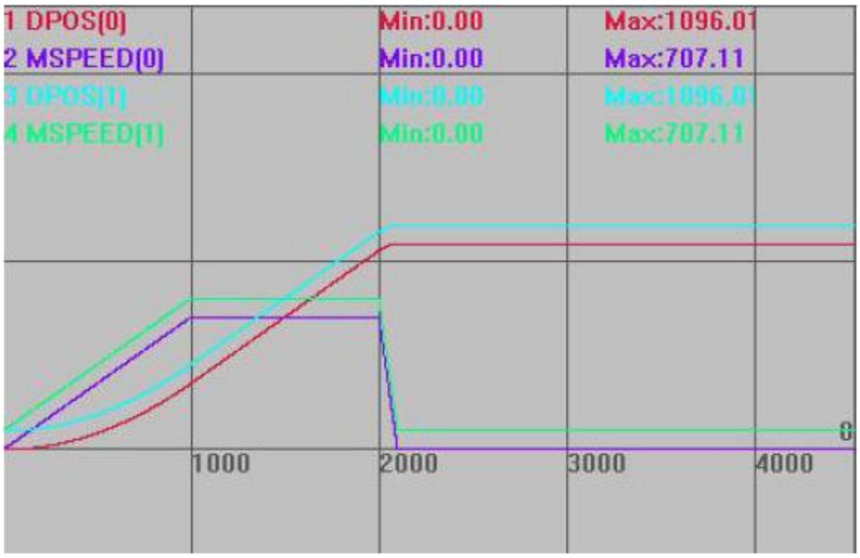
Motion trajectory and speed curve:
DPOS(0) vertical scale 1000, no offset
MSPEED(0) vertical scale 1000, offset -1000
DPOS(1) vertical scale 1000, offset 100
MSPEED(1) vertical scale 1000, offset -900
DPOS(2) vertical scale 1000, offset 200
MSPEED(2) vertical scale 1000, offset -800</td></tr>
</table>



Example 2

BASE(0,1)
DPOS=0,0
ATYPE=1,1
SPEED=1000
ACCEL=1000
DECEL=1000 'set deceleration as 1000
FASTDEC=10000 'set fast deceleration as 10000
TRIGGER
MOVE(10000,10000) 'interpolation movement
DELAY(2000) 'delay 2 seconds
RAPIDSTOP(2) 'axis stops immediately, deceleration is 10000

Motion trajectory and speed curve:
DPOS(0) vertical scale 1000, no offset
MSPEED(0) vertical scale 1000, no offset
DPOS(1) vertical scale 1000, offset 100
MSPEED(1) vertical scale 1000, offset 100

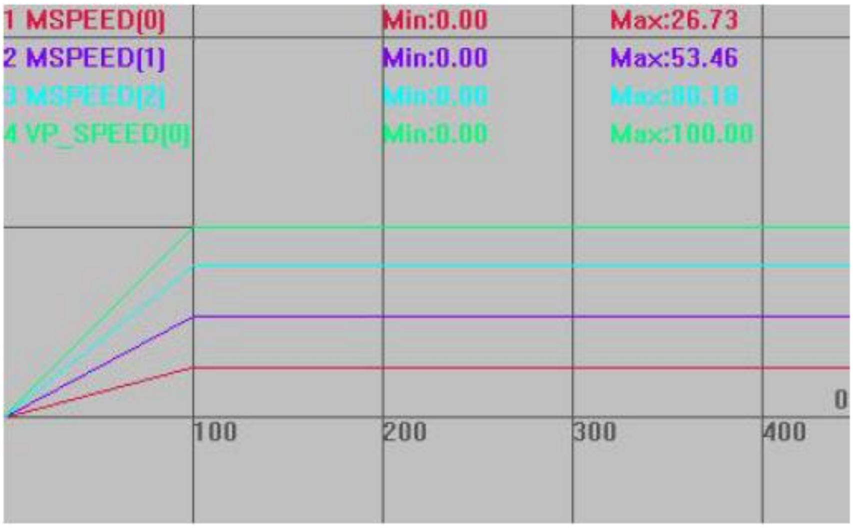
<table>
<tr><td></td><td>
</td></tr>
<tr><td>Instructions</td><td>CANCEL, DECEL, FASTDEC</td></tr>
</table>

MOVE – linear motion

Type	Multi-Axis Motion Instruction
Description	**linear interpolation motion, which is relative motion.** Only speed of main axis is valid in interpolation motion, main axis is the first axis in BASE list, motion will follow parameters of main axis. This instruction can be used in continuous interpolation movements by adding SP, see *SP for reference. Interpolation motion distance: X= $\sqrt{X_0^2 + X_1^2 + X_2^2 + ... + X_n^2}$ Motion time: T=X/speed of main axis.
Grammar	MOVE(distance1 [,distance2 [,distance3 [,distance4...]]]) Parameters: distance1 -move distance of the first axis distance2 -move distance of the next axis
Controller	General
Example	Example 1 Base(0,1,2,) 'axis 0 is the main axis ATYPE=1,1,1 'set as pulse type UNITS=100,100,100 'pulse equivalent configuration SPEED=100,10,1000 'only speed of main axis is valid, act as resultant speed ACCEL=1000,1000,1000 DECEL=1000,1000,1000 DPOS = 0,0,0 Trigger 'Trigger the oscilloscope automatically **MOVE**(500,1000,1500) 'axis 0,1,2 will do linear interpolation, relative

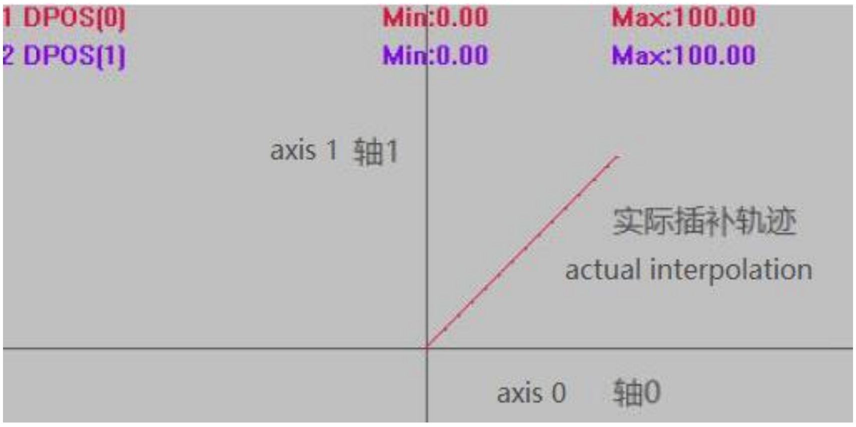
distance.
WAIT IDLE 'wait until the motion stops.
PRINT *DPOS 'Printed result:500,1000,1500

Speed of each axis in interpolation motion is the component speed of main axis.
MSPEED(0) vertical scale 100
MSPEED(1) vertical scale 100
MSPEED(2) vertical scale 100
VP_SPEED(0) vertical scale 100



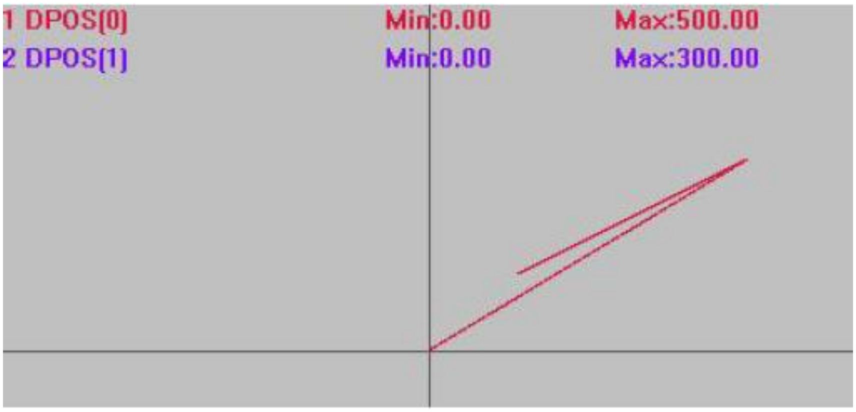
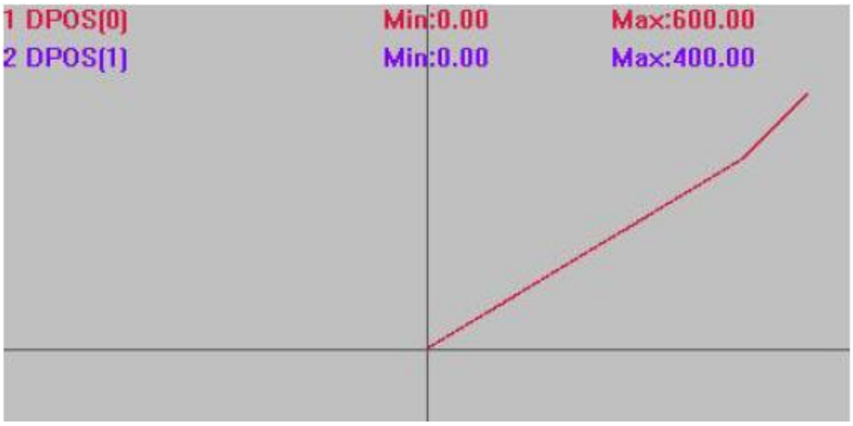
Example 2
BASE(0,1)
ATYPE=1,1
UNITS=100,100 'pulse equivalent configuration
SPEED=100,100
ACCEL=1000,1000
DECEL=1000,1000
DPOS=0,0
MPOS=0,0
Trigger 'Trigger the oscilloscope automatically
MOVE(100,100)

Interpolation trajectory
DPOS(0) horizontal scale 100
DPOS(1) vertical scale 100

	
Instructions	MOVEABS,*SP

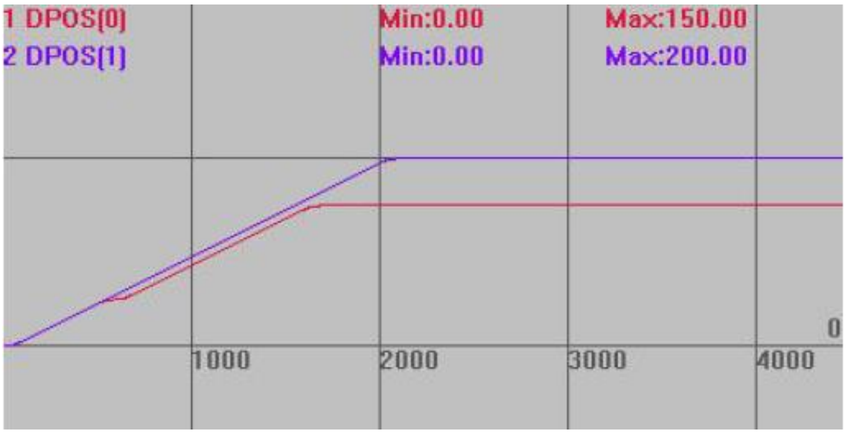
MOVEABS – Linear Motion-Absolutely

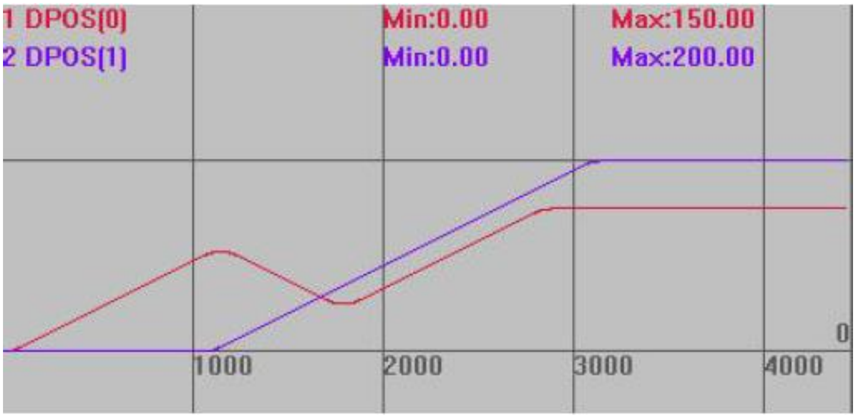
Type	Multi-Axis Motion Instruction
Description	**Linear Interpolation movement, it moves absolutely to defined coordinate.** This instruction can be used in continuous interpolation movements by adding SP, see ***SP** for reference.
Grammar	MOVEABS(position1[, position2[, position3[, position4...]]]) position1 -coordinate of first axis position2 -coordinate of next axis
Controller	General
Example	BASE(0,1) UNITS=100,100 DPOS=0,0 MPOS=0,0 SPEED=100,100 ACCEL=1000,1000 DECEL=1000,1000 TRIGGER 'Trigger the oscilloscope automatically. **MOVEABS**(500,300) 'axis 0 moves to 500, axis 1 moves to 300, interpolation motion **MOVEABS**(100,100) 'axis 0 moves back to 100, axis 1 moves back to 100. Interpolation trajectory; DPOS(0) horizontal scale 300 DPOS(1) vertical scale 300

	 When using MOVE relative motion, other conditions are the same, MOVEBAS instruction is turned into MOVE instruction. Interpolation trajectory: DPOS(0) horizontal scale 300 DPOS(1) vertical scale 300 
Instructions	MOVE, *SP

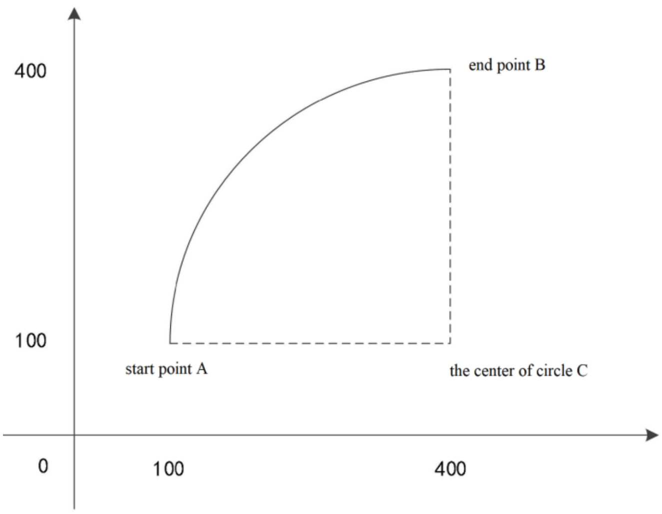
MOVEMODIFY2 – Move to new position

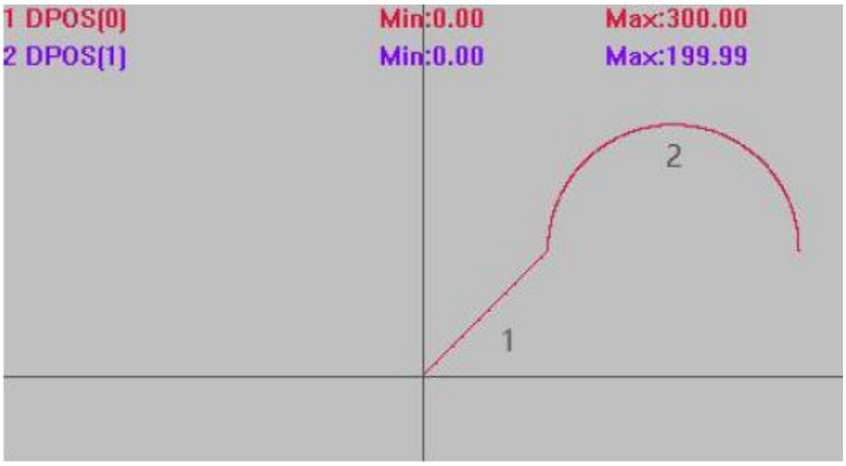
Type	Multi-axis motion instruction
Description	**Force the previous motion to stop, move to a new target position at former speed and acceleration.** If there isn't motion in the former, then the result caused by this instruction is the same as MOVEABS, but each axis' motion is independent and will not enter the motion buffer, see example 1 for reference. It must be used with WAIT instruction. See example 2 for reference. When there is continuous interpolation, MOVEMODIFY2 will interrupt the continuity of motion. When MOVEMODIFY2 is used in multi-axis situation, the motion is not absolutely linear interpolation movement.
Grammar	MOVEMODIFY2 (abspos1, abspos2,[...])

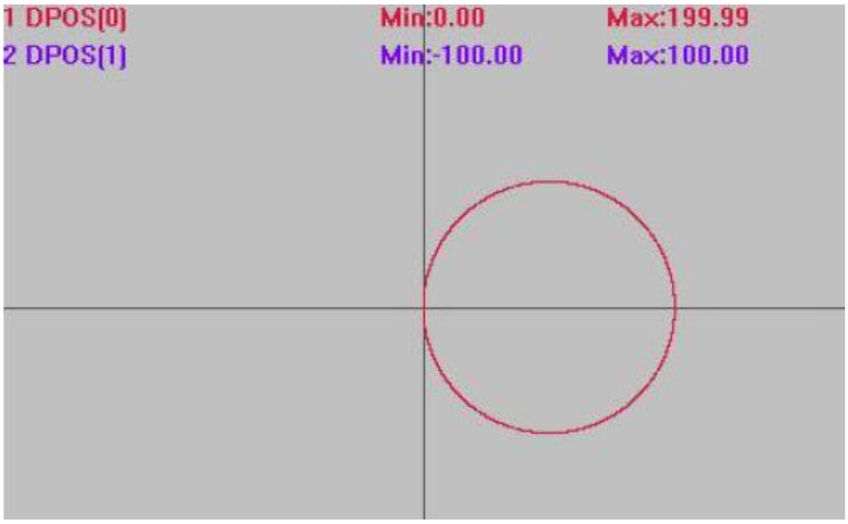
	abspos1 BASE -Target position of axis 1 abspos2 BASE -Target position of axis 2 ZMC3XX series with firmware version above 20161209. ZMC4XX series with firmware version above 20170509.
Controller	Special firmware
Example	**Example 1** BASE(0,1) 'set as pulse type ATYPE = 1,1 DPOS=0,0 SPEED = 100,100 ACCEL=1000 'acceleration configuration DECEL=1000 TRIGGER MOVE(200) AXIS(0) **MOVEMODIFY2(50,200)** 'cancel MOVE (200), force the axis to move to a new position (50,200). MOVE(100) AXIS(1) Motion trajectory : DPOS(0) vertical scale 200 DPOS(1) vertical scale 200  **Example 2** BASE(0,1) ATYPE=1,1 'set as pulse type DPOS=0,0 SPEED=100,100 ACCEL=1000,1000 'acceleration configuration DECEL=1000,1000 TRIGGER MOVE(200) AXIS(0) WAIT UNTIL DPOS(0)>=100 'wait until the axis 0 reaches position 100 **MOVEMODIFY2(50,200)** MOVE(100) AXIS(0)

	Motion trajectory: The vertical scale is the same as the above. 
Instructions	MOVEMODIFY

MOVECIRC –Arc at the Center

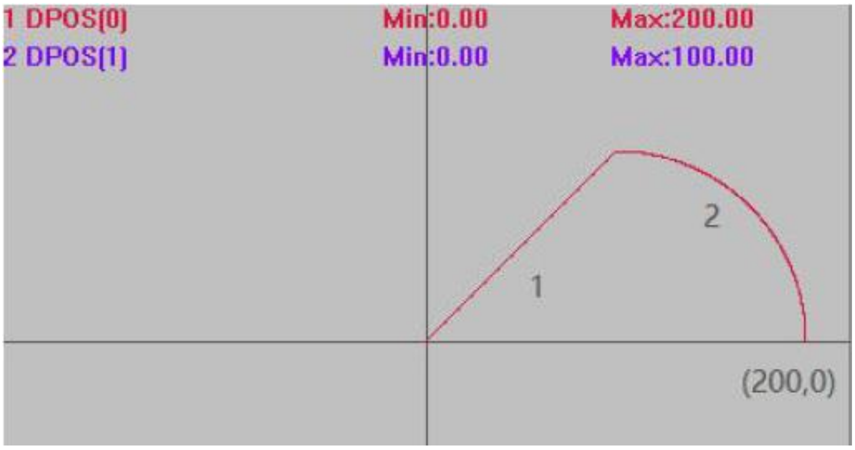
Type	Multi-Axis Motion Instruction
Description	**Circular interpolation between two axes, arc at the center, relative motion.** The first axis and second axis in BASE list will execute circular interpolation, and in relative motion mode. If the end distance is 0, then the motion will generate a full circle. This instruction can be used in continuous interpolation movements by adding SP, see *SP for reference. When using, it is necessary to obtain the coordinates of the center of the circle and the end point of the arc relative to the starting point. Ensure the coordinates are correct, or the actual motion path will be wrong.  Suppose start point A is (100,100), the center point C is (400,100), end point B is (400,400).

	Then the coordinate of point C that is related to starting point A is (300,0), for point B is (300,300).
Grammar	MOVECIRC (end1, end2, centre1, centre2, direction) end1: end point coordinate of the first axis, which is relative to starting point. end2: end point coordinate of the second axis, which is relative to starting point. center1: center point coordinate of the first axis, relative to starting point. center2: center point coordinate of the second axis, relative to starting point. direction: 0-anticlockwise 1-clockwise
Controller	General
Example	BASE(0,1) ATYPE=1,1 'set as pulse type UNITS=100,100 DPOS=0,0 SPEED=100,100 ACCEL=1000,1000 DECEL=1000,1000 TRIGGER 'trigger the oscilloscope automatically MOVE(100,100) 'move to position (100,100) **MOVECIRC**(200,0,100,0,1) 'draw the semicircle with 100 radius in clockwise, end point coordinate is (300,100). Interpolation trajectory: DPOS(0) vertical scale 150 DPOS(1) vertical scale 150  Other conditions are the same, the motion instruction is modified: **MOVECIRC**(0,0,100,0,0'radius is 100, center (100,0), draw in anticlockwise Interpolation trajectory: Same as the above.

	
Instructions	MOVECIRCABS, MOVECIRC2, *SP

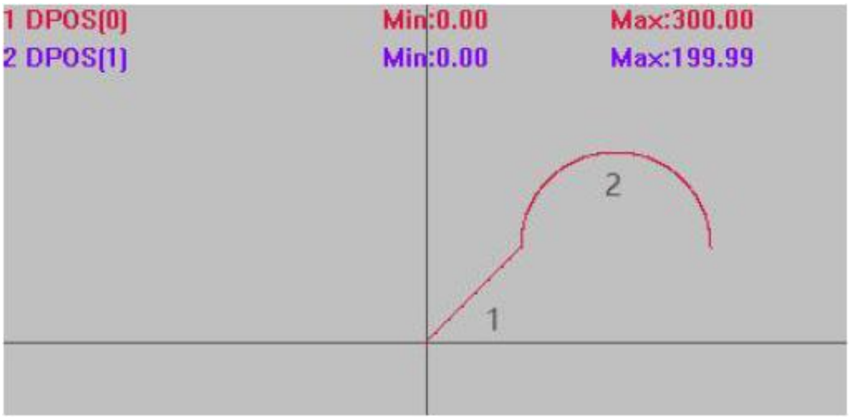
MOVECIRCABS - Center Based Arc - Absolute

Type	Multi-Axis Motion Instruction
Description	**Circular interpolation between two axes, draw the arc at the center, absolute motion.** The first and second axis in BASE list will execute circular interpolation, and in absolute motion mode. This instruction can be used in continuous interpolation movements by adding SP, see *SP for reference. MOVECIRCABS doesn't support moving a whole circle, but MOEVCIRC supports.
Grammar	MOVECIRCABS (end1, end2, centre1, centre2, direction) end1: end point coordinate of the first axis, the absolute position. end2: end point coordinate of the second axis, the absolute position. center1: center point coordinate of the first axis, the absolute position. center2: center point coordinate of the second axis, the absolute position. direction: 0-anticlockwise 1-clockwise Ensure the coordinate is correct, or the actual motion path will be wrong.
Controller	General
Example	BASE(0,1) ATYPE=1,1 'set as pulse type UNITS=100,100 DPOS=0,0 SPEED=100,100 ACCEL=1000,1000 DECEL=1000,1000 TRIGGER 'Trigger the oscilloscope automatically MOVE(100,100) 'move to position (100,100)

	MOVECIRCABS(200,0,100,0,1) 'draw quarter circle of radius 100 clockwise, end point is (200,0). Interpolation Path DPOS(0) vertical scale 100 DPOS(1) vertical scale 100 
Instructions	MOVECIRC, MOVECIRC2ABS, *SP

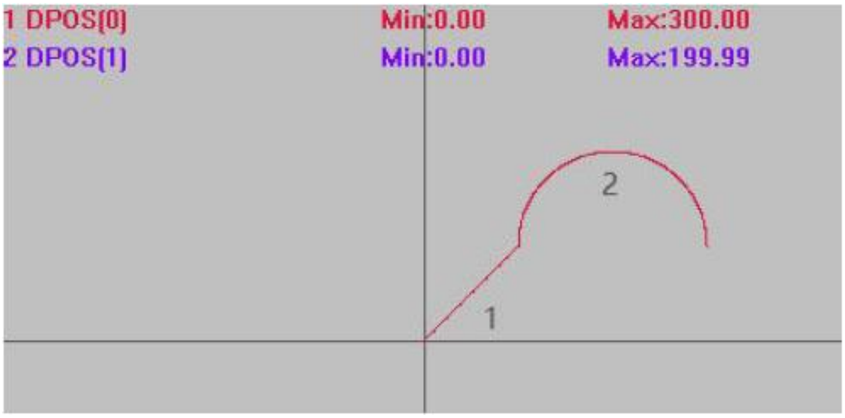
MOVECIRC2 - Three-Point Based Arc

Type	Multi-Axis Motion Instruction
Description	**Circular interpolation between two axes, three-point based arc, relative motion.** The first and second axis in BASE list will execute circular interpolation, and in relative motion mode, which is relative to start point. This instruction can be used in continuous interpolation movements by adding SP, see *SP for reference. Note: don’t use this instruction to do full circle interpolation. it is better to use MOVECIRC or use MOVECIRC2 two times.
Grammar	MOVECIRC2(mid1, mid2, end1, end2) mid1: middle point coordinate of the first axis, which is relative to start point. mid2: middle point coordinate of the second axis, it is relative to start point. end1: end point coordinate of the first axis, which is relative to start point. end2: end point coordinate of the second axis, which is relative to start point. Ensure the coordinate is correct, or the actual motion path will be wrong.
Controller	General
Example	BASE(0,1) ATYPE=1,1 'set as pulse type UNITS=100,100 DPOS=0,0 SPEED=100,100

	ACCEL=1000,1000 DECEL=1000,1000 TRIGGER 'Trigger the oscilloscope automatically MOVE(100,100) 'move to position (100,100) **MOVECIRC2**(100,100,200,0) 'draw semicircle through 3 points, relative coordinate Interpolation path: DPOS(0) vertical scale 200 DPOS(1) vertical scale 200 
Instructions	MOVECIRC2ABS, MOVECIRC, *SP

MOVECIRC2ABS --Three-Point Based Arc - Absolute

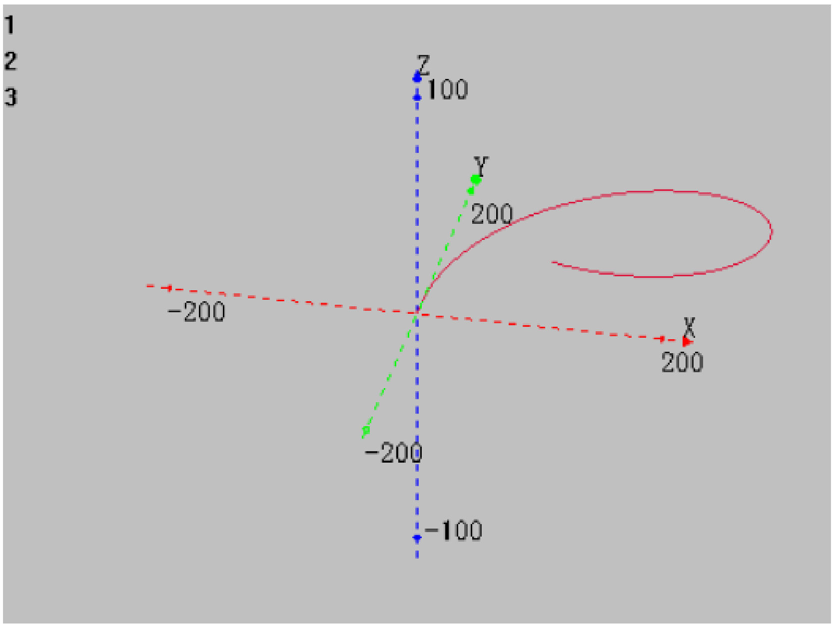
Type	Multi-Axis Motion Instruction
Description	**Circular interpolation, arc at the center, absolute motion.** The first and second axis in BASE list will execute circular interpolation, and in relative motion mode. This instruction can be used in continuous interpolation movements by adding SP, see *SP for reference. Note: don't use this instruction to do full circle interpolation. it is better to use MOVECIRC or use MOVECIRC2 two times.
Grammar	MOVECIRC2ABS(mid1, mid2, end1, end2) mid1: middle point coordinate of the first axis, which is relative to start point. mid2: middle point coordinate of the second axis, it is relative to start point. end1: end point coordinate of the first axis, which is relative to start point. end2: end point coordinate of the second axis, which is relative to start point. Ensure the coordinate is correct, or the actual motion path will be wrong.
Controller	General
Example	BASE(0,1) ATYPE=1,1 'set as pulse type UNITS=100,100

	DPOS=0,0 SPEED=100,100 ACCEL=10000,10000 DECEL=10000,10000 TRIGGER 'Trigger the oscilloscope automatically MOVE(100,100) 'move to position (100,100). **MOVECIRC2ABS**(200,200,300,100) 'draw semicircle through 3 points, absolute coordinate Interpolation path: DPOS(0) vertical scale 200 DPSO(1) vertical scale 200 
Instructions	MOVECIRC2, MOVECIRCABS, *SP

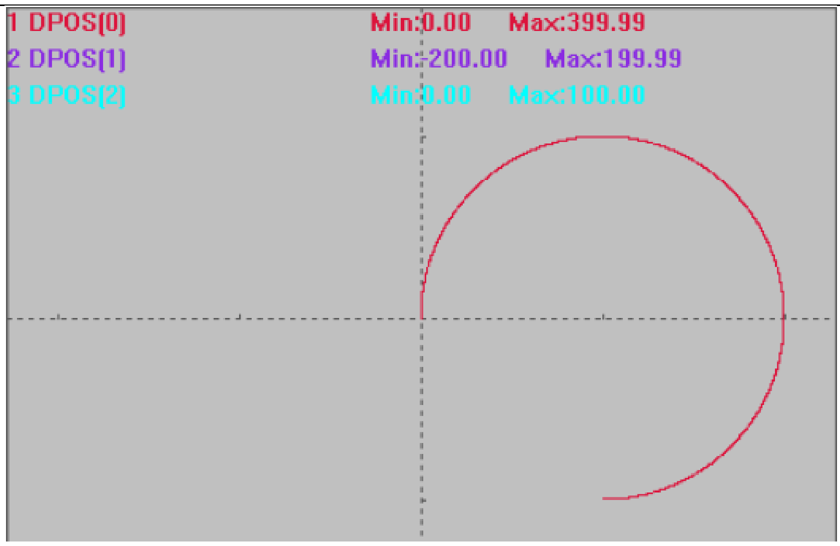
MHELICAL – Central Helical

Type	Multi-Axis Motion Instruction
Description	**Helical Interpolation, arc at the center, relative motion.** The first and second axis in BASE list will execute circular interpolation, the third axis will execute helical, and they are relative to start point. This instruction can be used in continuous interpolation movements by adding SP, see *SP for reference. It can execute a full circle in Z direction.
Grammar	MHELICAL(end1,end2,centre1,centre2,direction,distance3,[mode]) end1: end point coordinate of the first axis, which is relative to start point. end2: end point coordinate of the second axis, which is relative to start point. center1: center point coordinate of the first axis, relative to start point. center2: center point coordinate of the second axis, relative to start point. direction: 0-anticlockwise 1-clockwisemode distance3: motion distance of the third axis mode: speed calculation of the third axis Value | Description

<table>
<tr><td></td><td><table><tr><td>0(default)</td><td>Third axis participates interpolation speed calculation.</td></tr><tr><td>1</td><td>Third axis keep independent.</td></tr></table>Ensure the coordinate is correct, or the actual motion path will be wrong.</td></tr>
<tr><td>Controller</td><td>General</td></tr>
<tr><td>Example</td><td>BASE(0,1,2)
ATYPE=1,1,1 'set as pulse type
UNITS=100,100,100
DPOS=0,0,0
SPEED=100,100,100 'main axis speed
ACCEL=1000,1000,1000 'main axis acceleration
DECEL=1000,1000,1000
TRIGGER
MHELICAL(200,-200,200,0,1,100) 'original point as start point, center is (200,0), end point is (200,0), clockwise, Axis Z participates speed calculation, move 100.

Axis 0, axis 1 and axis 2 interpolation trajectory under XYZ mode:


Axis 0 and axis 1 interpolation under XY mode:</td></tr>
</table>



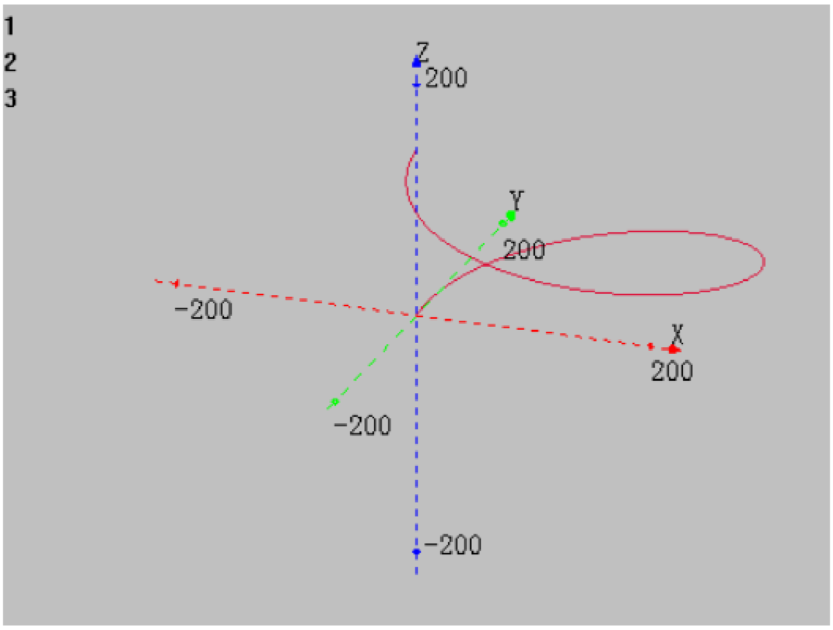
Instructions	MHELICAL2, MHELICALABS, *SP

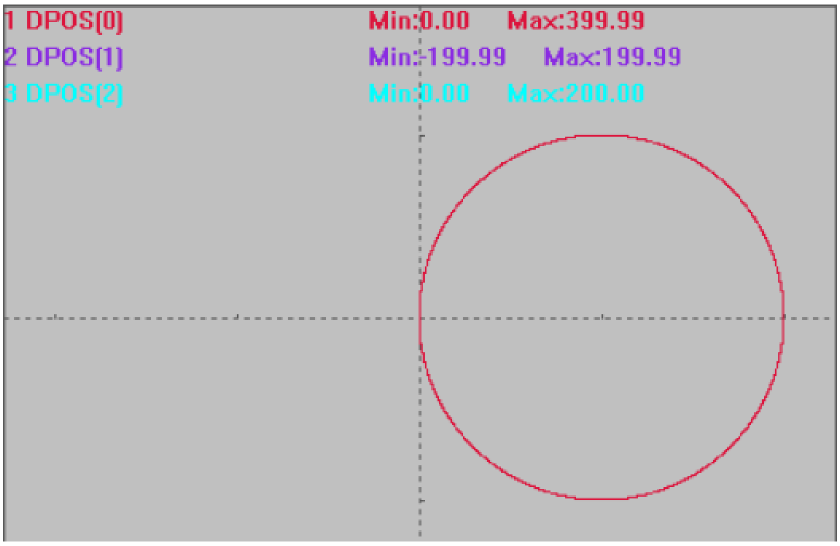
MHELICALABS – Central Helical - Absolute

<table>
<tr><td>Type</td><td>Multi-Axis Motion Instruction</td></tr>
<tr><td>Description</td><td>Helical Interpolation, arc at the center, absolute motion.

The first and second axis in BASE list will execute circular interpolation, the third axis will execute helical, in absolute motion way.
This instruction can be used in continuous interpolation movements by adding SP, see *SP for reference.
It can execute a full circle in Z direction.</td></tr>
<tr><td>Grammar</td><td>MHELICALABS(end1,end2,centre1,centre2,direction,distance3,[mode])
end1: motion coordinate of the first axis
end2: motion coordinate of the second axis
center1: motion center point of the first axis
center2: motion center point of the second axis
direction: 0-anticlockwise 1-clockwisemode
distance3: motion distance of the third axis
mode: speed calculation of the third axis
<table>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>0(default)</td><td>Third axis participates interpolation speed calculation.</td></tr>
<tr><td>1</td><td>Third axis keep independent.</td></tr>
</table>
Ensure the coordinate is correct, or the actual motion path will be wrong.</td></tr>
<tr><td>Controller</td><td>General</td></tr>
<tr><td>Example</td><td>BASE(0,1,2)
ATYPE=1,1,1 'set as pulse type
UNITS=100,100,100
DPOS=0,0,0
SPEED=100,100,100 'main axis speed
ACCEL=1000,1000,1000 'main axis acceleration</td></tr>
</table>

<table>
<tr><td></td><td>
DECEL=1000,1000,1000
TRIGGER
MHELICALABS(0,0,200,0,1,100) 'start from original point, center point (200,0), end point (0,0), clockwise, Axis Z participates speed calculation, move 200.

Axis 0, axis 1 and axis 2 interpolation trajectory under XYZ mode:


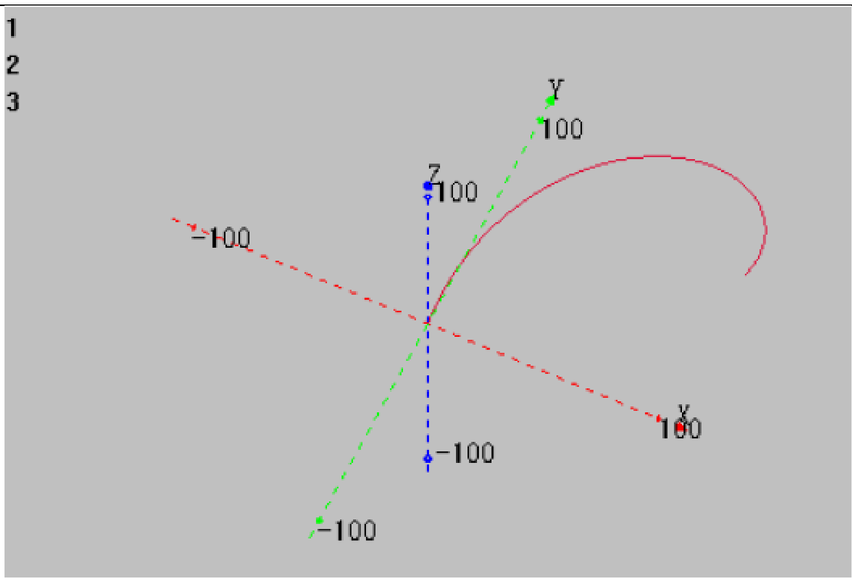
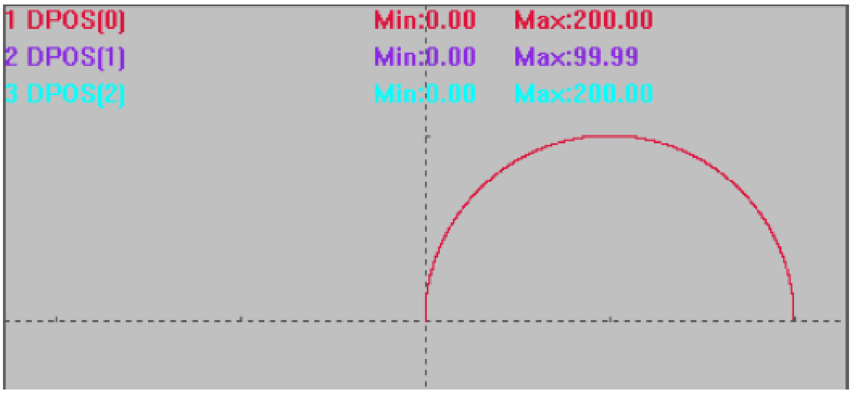
Axis 0 and axis 1 interpolation trajectory under XY mode:


</td></tr>
<tr><td>Instructions</td><td>MHELICAL, MHELICAL2ABS, *SP</td></tr>
</table>

MHELICAL2 – Three-Point Based Helical

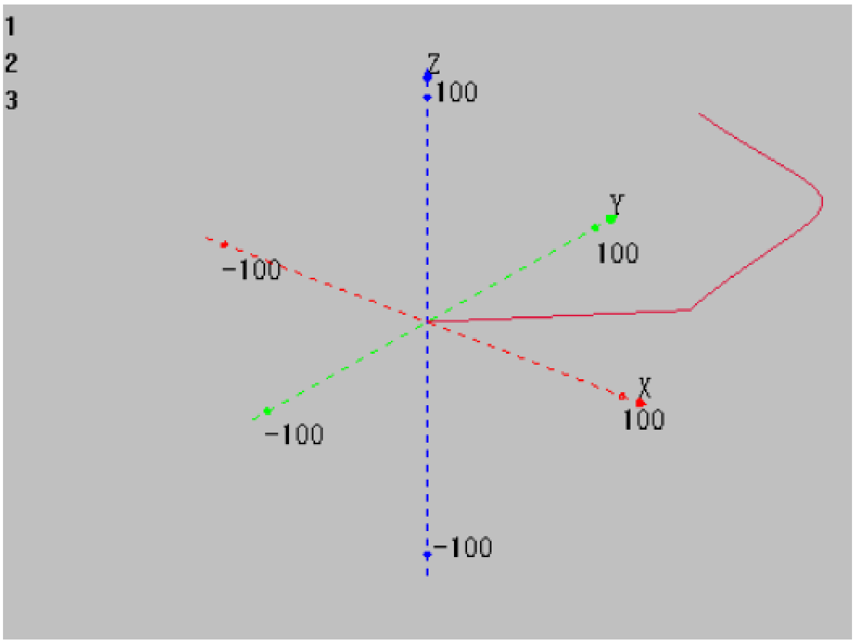
Type	Multi-Axis Motion Instruction
Description	**Helical Interpolation, arc at the center, absolute motion.**

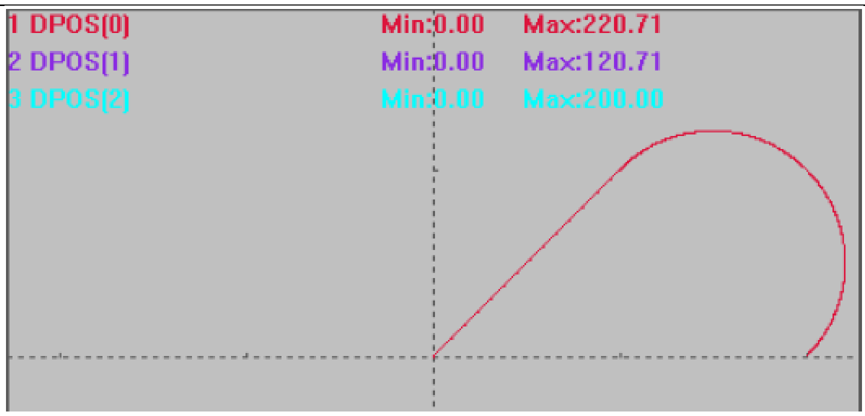
	The first and second axis in BASE list will execute circular interpolation, the third axis will execute helical, in relative motion way. This instruction can be used in continuous interpolation movements by adding SP, see *SP for reference. It can't generate a full circle in Z direction, please use MHELICAL or MHELICALABS.
Grammar	MHELICAL2(mid1, mid2, end1, end2, distance3,[mode]) mid1: middle point coordinate of the first axis, which is relative to start point. mid2: middle point coordinate of the second axis, it is relative to start point. end1: end point coordinate of the first axis, which is relative to start point. end2: end point coordinate of the second axis, which is relative to start point. distance3: motion distance of the third axis, which is relative to start point. mode: speed calculation of the third axis <table><tr><th>Value</th><th>Description</th></tr><tr><td>0(default)</td><td>Third axis participates interpolation speed calculation.</td></tr><tr><td>1</td><td>Third axis keep independent.</td></tr></table> Ensure the coordinate is correct, or the actual motion path will be wrong.
Controller	General
Example	BASE(0,1,2) ATYPE=1,1,1 'set as pulse type UNITS=100,100,100 DPOS=0,0,0 SPEED=100,100,100 'main axis speed ACCEL=1000,1000,1000 'main axis acceleration DECEL=1000,1000,1000 **MHELICAL2**(100,100,200,0,200) 'start from original point, center point (100,100), end point (200,0), Axis Z participates speed calculation, move 200. Axis 0, axis 1 and axis 2 interpolation trajectory under XYZ mode:

	 Axis 0 and axis 1 trajectory interpolation under XY mode: 
Instructions	MHELICAL2ABS, MHELICAL, *SP

MHELICAL2ABS-Three-Point Based Helical-Absolute

Type	Multi-Axis Motion Instruction
Description	**Helical Interpolation, arc at the center, absolute motion.** The first and second axis in BASE list will execute circular interpolation, the third axis will execute helical, in relative motion way. This instruction can be used in continuous interpolation movements by adding SP, see *SP for reference. It can't generate a full circle in Z direction, please use MHELICAL or MHELICALABS.
Grammar	MHELICAL2(mid1, mid2, end1, end2, distance3,[mode]) mid1: middle point coordinate of the first axis, which is relative to start point. mid2: middle point coordinate of the second axis, it is relative to start point. end1: end point coordinate of the first axis, which is relative to start point. end2: end point coordinate of the second axis, which is relative to start point. distance3: motion distance of the third axis, errata: there is a problem with this

	parameter in versions before 20150306, it is recommended to use the MHELICAL2 relative command mode: speed calculation of the third axis <table><tr><th>Value</th><th>Description</th></tr><tr><td>0(default)</td><td>Third axis participates interpolation speed calculation.</td></tr><tr><td>1</td><td>Third axis keep independent.</td></tr></table> Ensure the coordinate is correct, or the actual motion path will be wrong.
Controller	General
Example	BASE(0,1,2) ATYPE=1,1,1 'set type as pulse UNITS=100,100,100 DPOS=0,0,0 SPEED=100,100,100 'main axis speed ACCEL=1000,1000,1000 'main axis acceleration DECEL=1000,1000,1000 MOVE(100,100) 'move to position（100,100） TRIGGER **MHELICAL2ABS**(200,100,200,0,200) 'start from point (100,100), center point (200,100), end point (200,0). Axis Z participates speed calculation, move 200. Axis 0, axis 1 and axis 2 interpolation trajectory under XYZ mode:  Axis 0 and axis 1 interpolation trajectory under XY mode:

<table>
<tr><td></td><td>
</td></tr>
<tr><td>Instructions</td><td>MHELICAL2, MHELICALABS, *SP</td></tr>
</table>

MECLIPSE -- Ellipse

<table>
<tr><td>Type</td><td>Multi-Axis Motion Instruction</td></tr>
<tr><td>Description</td><td>Ellipse interpolation, arc at the center, relative motion, helical is optional.

Execute elliptical interpolation with first and second axis in BASE list, relative motion mode, the third axis is available for synchronized helical motion.
This instruction can be used in continuous interpolation movements by adding SP, see *SP for reference.
Valid for full ellipse drawing.
Valid for ellipse drawing whose major axis is parallel or perpendicular to X.</td></tr>
<tr><td>Grammar</td><td>MECLIPSE (end1, end2, centre1, centre2, direction, adis, bdis[, end3])
end1: end point coordinate of the first axis, which is relative to start point.
end2: end point coordinate of the second axis, which is relative to start point.
center1: center point coordinate of the first axis, relative to start point.
center2: center point coordinate of the second axis, relative to start point.
direction: 0-anticlockwise 1-clockwise
<table>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>0</td><td>Clockwise</td></tr>
<tr><td>1</td><td>Anticlockwise</td></tr>
</table>
adis: ellipse radius of the first axis, semi-major or semi-minor axis is optional.
bdis: ellipse radius of the second axis, semi-major or semi-minor axis is optional. when adis is equal to bdis, the path is arc or helical line.
end3: distance of the third axis, fill this value when helical is necessary.

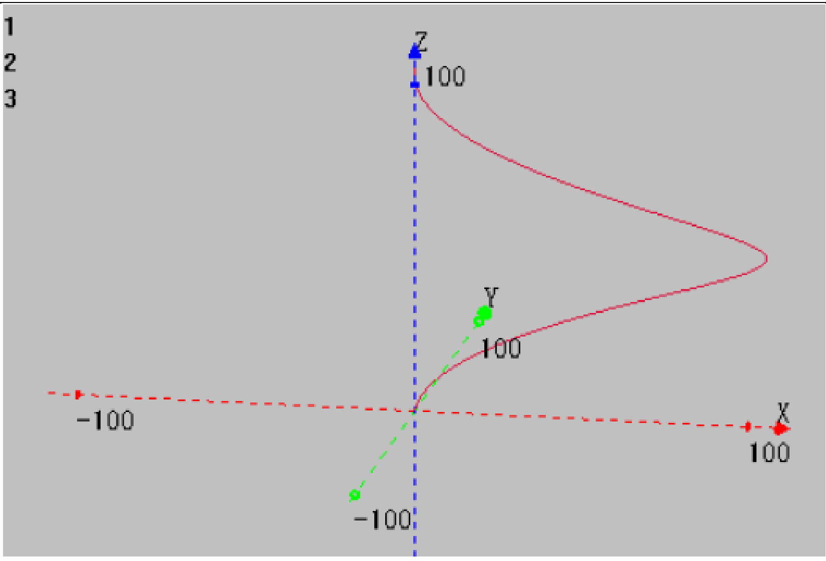
Ensure the coordinate is correct, or the actual motion path will be wrong.</td></tr>
<tr><td>Controller</td><td>General</td></tr>
<tr><td>Example</td><td>Example 1 No helical
RAPIDSTOP(2)
WAIT IDLE(0)
BASE(0,1,2)</td></tr>
</table>

ATYPE=1,1,1 'set type as pulse
UNITS=100,100,100
SPEED=100,100,100 'main axis speed
ACCEL=1000,1000,1000 'main axis acceleration
DECEL=1000,1000,1000
DPOS=0,0,0
TRIGGER 'Trigger the oscilloscope automatically
MECLIPSE(0,0,100,0,1,100,50) 'center point (100,0), end point (0,0), semi-minor axis 50, semi-major axis 100, full ellipse drawing clockwise, no helical.

Interpolation Path:
DPOS(0) vertical scale 100
DPOS(1) vertical scale 100

Example2 with helical.
RAPIDSTOP(2)
WAIT IDLE(0)
BASE(0,1,2)
ATYPE=1,1,1 'set type as pulse
UNITS=100,100,100
SPEED=100,100,100 'main axis speed
ACCEL=1000,1000,1000 'main axis acceleration
DECEL=1000,1000,1000
DPOS=0,0,0
TRIGGER 'Trigger the oscilloscope automatically
MECLIPSE(0,0,100,0,1,100,50,200) 'center point (100,0), end point (0,0), semi-minor axis 50, semi-major axis 100 full ellipse drawing clockward with helical motion. The motion distance of the third axis is 200.

Axis 0, axis 1 and axis 2 interpolation trajectory under XYZ mode, and it merges axis Z motion on the basis of example 1.

	
Instructions	MECLIPSEABS, *SP

MECLIPSEABS – Ellipse - Absolute

<table>
<tr><td>Type</td><td>Multi-Axis Motion Instruction</td></tr>
<tr><td>Description</td><td>Ellipse interpolation, arc at the center, absolute motion, helical is optional.

Execute elliptical interpolation with first and second axis in BASE list, absolute motion mode, the third axis is available for synchronized helical motion.
This instruction can be used in continuous interpolation movements by adding SP, see *SP for reference.
Valid for full ellipse drawing.</td></tr>
<tr><td>Grammar</td><td>MECLIPSEABS(end1, end2, centre1, centre2, direction, adis, bdis[, end3])
end1: end point coordinate of the first axis, which is relative to start point.
end2: end point coordinate of the second axis, which is relative to start point.
center1: center point coordinate of the first axis, relative to start point.
center2: center point coordinate of the second axis, relative to start point.
direction: 0-anticlockwise 1-clockwise
<table>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>0</td><td>Clockwise</td></tr>
<tr><td>1</td><td>Anticlockwise</td></tr>
</table>
adis: ellipse radius of the first axis, semi-major or semi-minor axis is optional.
bdis: ellipse radius of the second axis, semi-major or semi-minor axis is optional. when adis is equal to bdis, the path is arc or helical line.
end3: distance of the third axis, fill this value when helical is necessary.

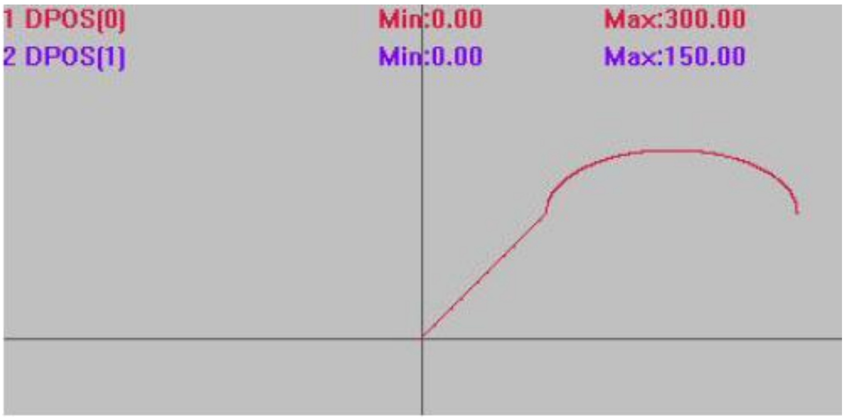
Ensure the coordinate is correct, or the actual motion path will be wrong.</td></tr>
<tr><td>Controller</td><td>General</td></tr>
</table>

Example

Example 1　no helical

```
BASE(0,1,2)
ATYPE=1,1,1                 'set type as pulse
UNITS=100,100,100
DPOS=0,0,0
SPEED=100,100,100           'main axis speed
ACCEL=1000,1000,1000        'main axis acceleration
DECEL=1000,1000,1000
TRIGGER                     'Trigger the oscilloscope automatically
MOVE(100,100)
MECLIPSEABS(300,100,200,100,1,100,50)
                            'center (200,100), end point (300,100), semi-minor
                            axis 50, semi-major axis 100, semi ellipse drawing,
                            no helical.
```

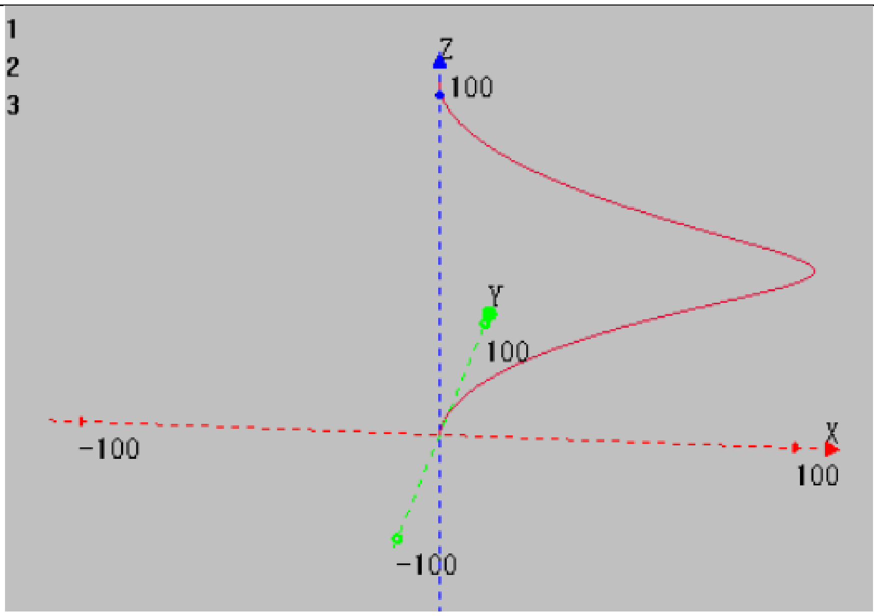
Interpolation Path:
DPOS(0) vertical scale 150
DPOS(1) vertical scale 150

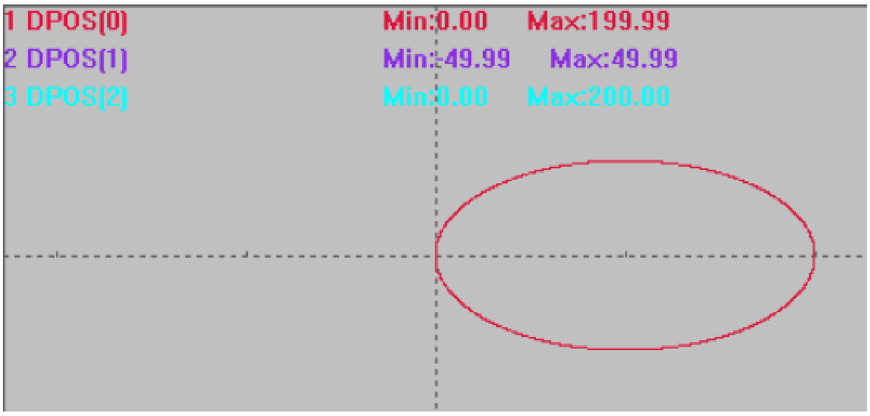


Example 2　with helical

```
BASE(0,1,2)
ATYPE=1,1,1                 'set type as pulse
UNITS=100,100,100
DPOS=0,0,0
SPEED=100,100,100           'main axis speed
ACCEL=1000,1000,1000        'main axis acceleration
DECEL=1000,1000,1000
MECLIPSEABS(0,0,100,0,1,100,50,200)
                            'center point (100,0), end point (0,0), semi-
                            minor axis 50, semi-major axis 100, full ellipse
                            drawing with helical.
```

Axis 0, axis 1 and axis 2 interpolation trajectory under XYZ mode:

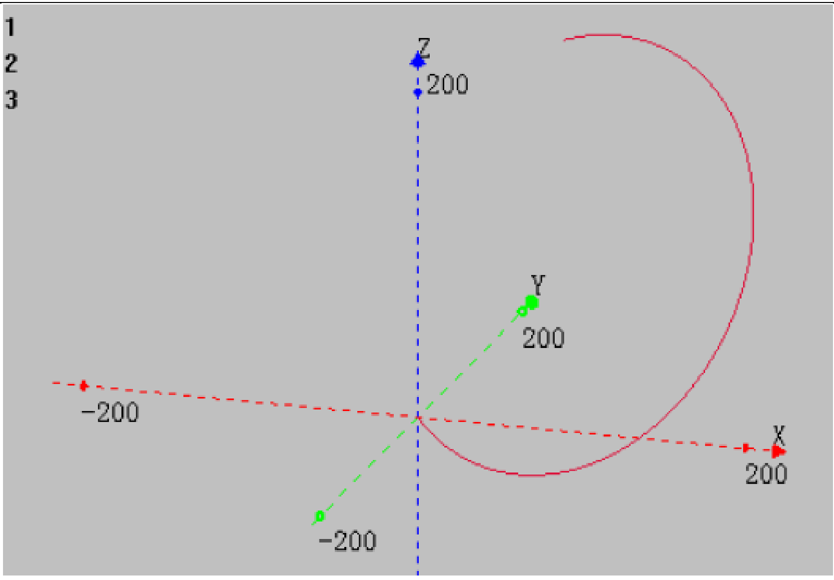
<table>
<tr><td></td><td>

Axis 0 and axis 1 interpolation trajectory:

</td></tr>
<tr><td>Instructions</td><td>MECLIPSE, *SP</td></tr>
</table>

MSPHERICAL – Space Arc

Type	Multi-Axis Motion Instruction
Description	**Spherical arc interpolation motion, relative motion mode, helical is optional.** This instruction can be used in continuous interpolation movements by adding SP, see *SP for reference.
Grammar	MSPHERICAL(end1,end2,end3,centre1,centre2,centre3,mode[,distance4][,distance5]) Parameters: end1 motion distance parameter1 of axis 1 end2 motion distance parameter1 of axis 2 end3 motion distance parameter1 of axis 3 centre1 motion distance parameter2 of axis 1 centre2 motion distance parameter2 of axis 2

	centre3 motion distance parameter2 of axis 3 mode specify the meaning of above parameters Value / Description: 0 — Generate arc by present point, middle point, and end point. parameter1: end point distance parameter2: middle point distance 1 — Generate arc by present point, central point, and end point. Move along the shortest arc. parameter1: end point distance parameter2: central point distance. 2 — Generate circle by present point, middle point, and end point. parameter1: end point distance parameter2: middle point distance. 3 — Generate circle by present point, central point, and end point. Move along the shortest arc first, then continue to finish the whole circle. parameter1: end point distance parameter2: central point distance. distane4: add the fourth axis as helical motion, appoint the relative motion distance of axis 4. This axis is not involved in speed calculation. distane5: add the fifth axis as helical motion, appoint the relative motion distance of axis 5. This axis is not involved in speed calculation. Ensure the coordinate is correct, otherwise, the actual motion path will be wrong.
Controller	General
Example	BASE(0,1,2) ATYPE=1,1,1 'set type as pulse UNITS=100,100,100 DPOS=0,0,0 SPEED=100,100,100 'main axis speed ACCEL=1000,1000,1000 'main axis acceleration DECEL=1000,1000,1000 TRIGGER Suppose central point is (120,160,150), radius is 250, 4 trajectories are generated below due to mode differences. mode 0: **MSPHERICAL**(120,160,400,240,320,300,0) 'end point: (120,160,400), middle point: (240,320,300), mode 0: three-point based arc. Axis 0, axis 1 and axis 2 interpolation trajectory under XYZ mode:

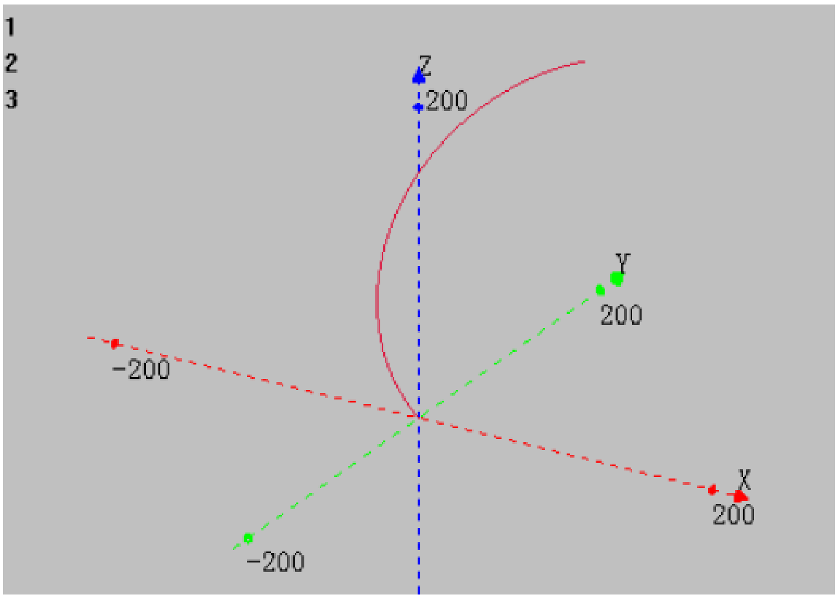


mode 1:

MSPHERICAL(120,160,400,120,160,150,1)

'relative position, end point (120,160,400), central point (120,160,150), mode 1: move along the shorter arc.

Axis 0, axis 1 and axis 2 interpolation trajectory under XYZ mode:

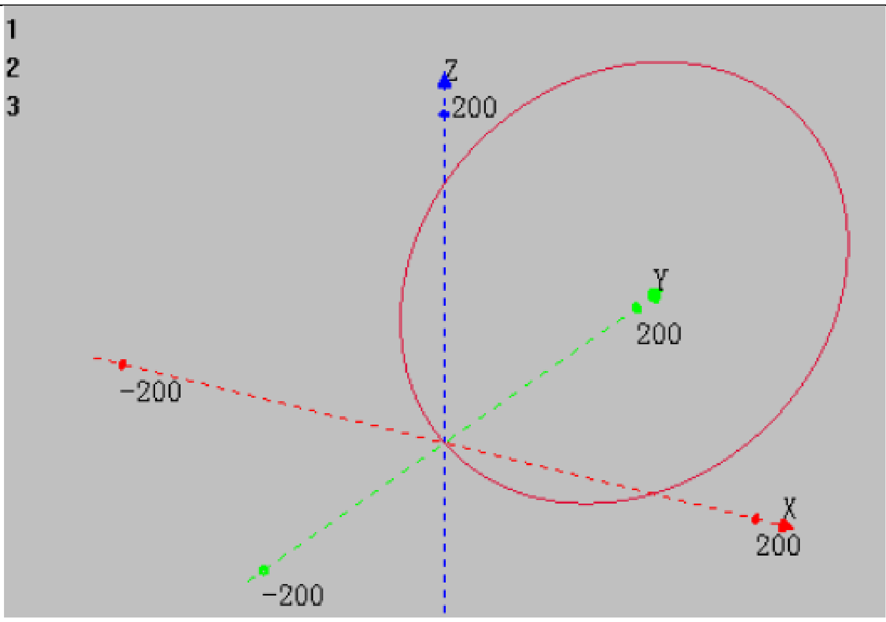


mode 2:

MSPHERICAL(120,160,400,240,320,300,2)

'end point: (120,160,400), middle point: (240,320,300), mode2: three points base circle.

Axis 0, axis 1 and axis 2 interpolation trajectory under XYZ mode:

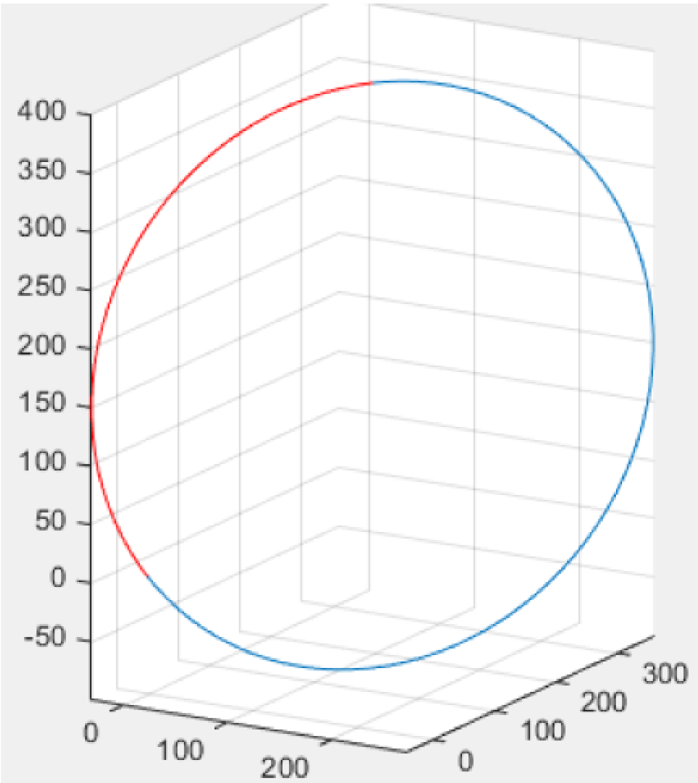


mode 3

MSPHERICAL(120,160,400,120,160,150,3)

'end point: (120,160,400), central point: (120,160,150), mode3: move along the shorter arc first (red part), then continue to finish the whole circle.

Axis 0, axis 1 and axis 2 interpolation trajectory under XYZ mode:



MSPHERICALABS – Space Arc – Absolute

<table>
<tr><td>Type</td><td>Multi-Axis Motion Instruction</td></tr>
<tr><td>Description</td><td>Space arc interpolation motion, absolute motion mode, helical is optional.
For continuous interpolation of custom speed, it can use command with SP suffix, please refer to *SP description.</td></tr>
<tr><td>Grammar</td><td>MSPHERICALABS(end1,end2,end3,centre1,centre2,centre3,mode[,distance4][,distance5])
Parameters:
end1 motion distance parameter1 of axis 1
end2 motion distance parameter1 of axis 2
end3 motion distance parameter1 of axis 3
centre1 motion distance parameter2 of axis 1
centre2 motion distance parameter2 of axis 2
centre3 motion distance parameter2 of axis 3
mode specify the meaning of above parameters
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Generate arc by present point, middle point, and end point.
parameter1: end point distance
parameter2: middle point distance</td></tr>
<tr><td>1</td><td>Generate arc by present point, central point, and end point.
Move along the shortest arc.
parameter1: end point distance
parameter2: central point distance.</td></tr>
<tr><td>2</td><td>Generate circle by present point, middle point, and end point.
parameter1: end point distance
parameter2: middle point distance.</td></tr>
<tr><td>3</td><td>Generate circle by present point, central point, and end point.
Move along the shortest arc first, then continue to finish the whole circle.
parameter1: end point distance
parameter2: central point distance.</td></tr>
</table>
distane4: add the fourth axis as helical motion, appoint the relative motion distance of axis 4. This axis is not involved in speed calculation.
distane5: add the fifth axis as helical motion, appoint the relative motion distance of axis 5. This axis is not involved in speed calculation.

Ensure the coordinate is correct, otherwise, the actual motion path will be wrong.</td></tr>
<tr><td>Controller</td><td>General</td></tr>
<tr><td>Example</td><td>BASE(0,1,2)
ATYPE=1,1,1 'set type as pulse
UNITS=100,100,100
DPOS=0,0,0</td></tr>
</table>

SPEED=100,100,100 'main axis speed
ACCEL=1000,1000,1000 'main axis acceleration
DECEL=1000,1000,1000
TRIGGER

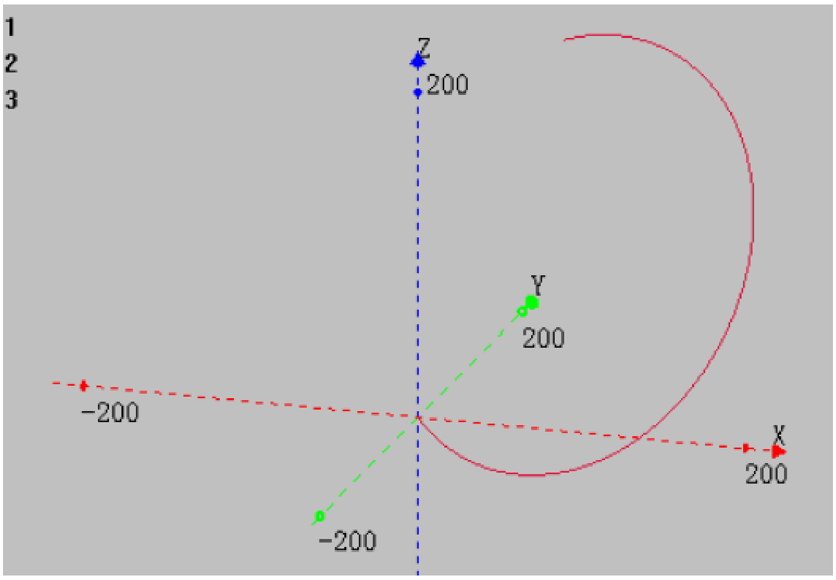
Suppose central point is (120,160,150), radius is 250, then below shows motion trajectories of 4 modes.

mode 0:

MSPHERICALABS(120,160,400,240,320,300,0)

'end point: (120,160,400), middle point: (240,320,300), mode 0, arc is made by three points.

Interpolation trajectory of axis 0, axis 1 and axis 2 under XYZ mode:

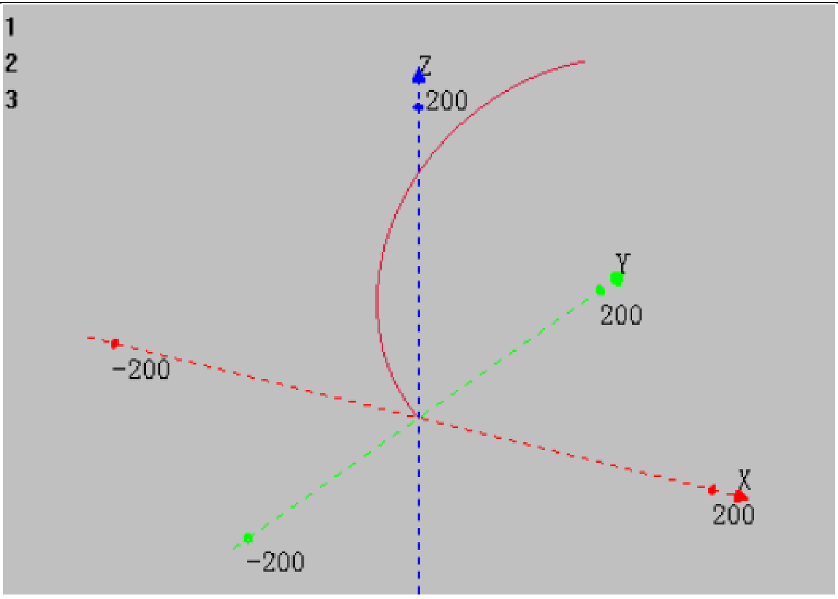


mode 1:

MSPHERICALABS(120,160,400,120,160,150,1)

'absolute position, end point (120,160,400), central point (120,160,150), mode 1: move along the shortest arc.

Interpolation trajectory of axis 0, axis 1 and axis 2 under XYZ mode:

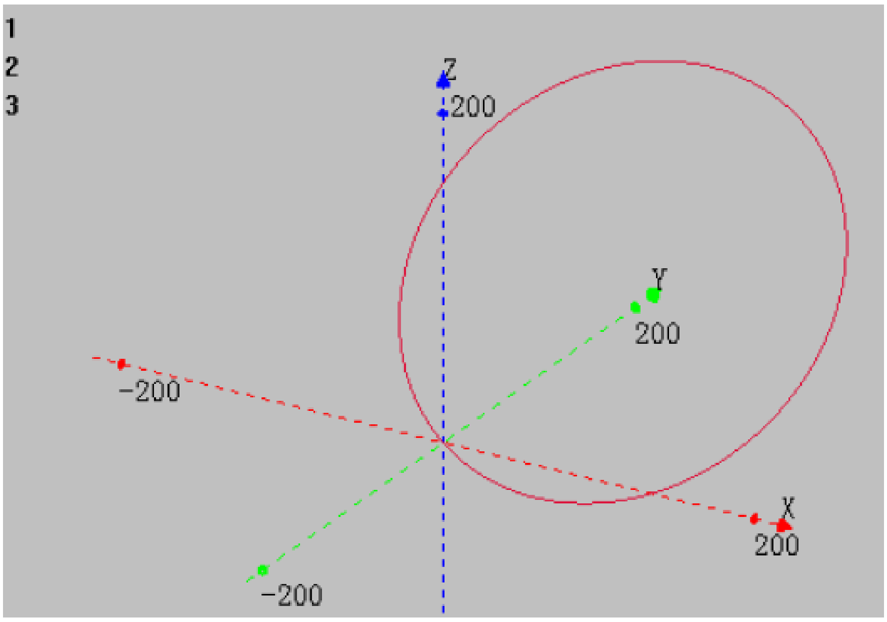


mode 2:

MSPHERICALABS(120,160,400,240,320,300,2)

‘end point: (120,160,400), middle point: (240,320,300), mode2: a full circle is made by three points

Interpolation trajectory of axis 0, axis 1 and axis 2 under XYZ mode:

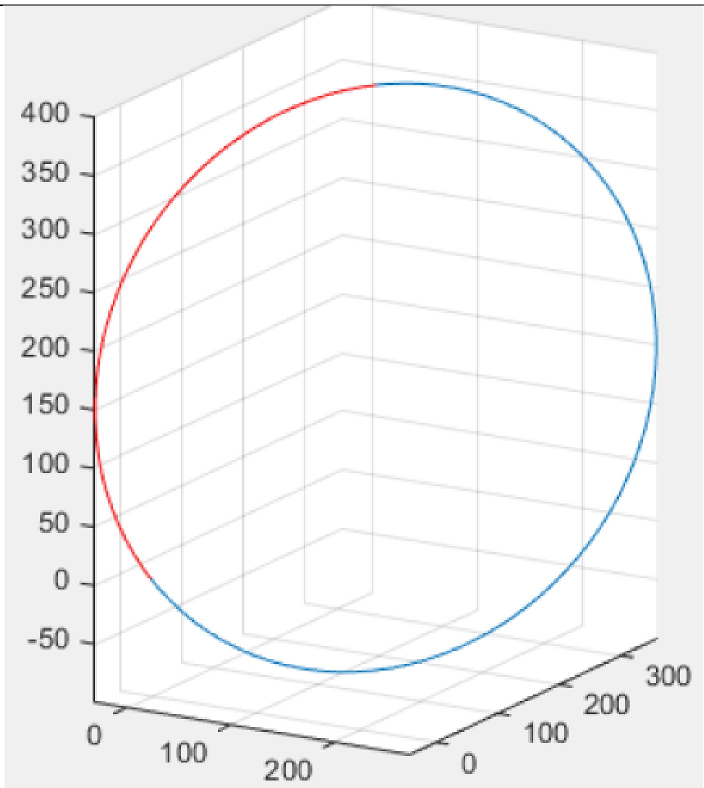


mode 3

MSPHERICALABS(120,160,400,120,160,150,3)

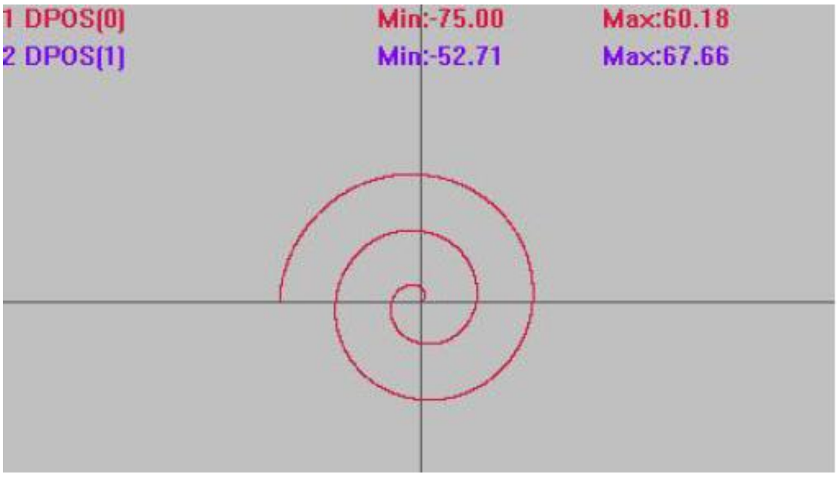
'end point: (120,160,400), central point: (120,160,150), mode3: move along the shortest arc first (red part), then continue to finish the whole circle.

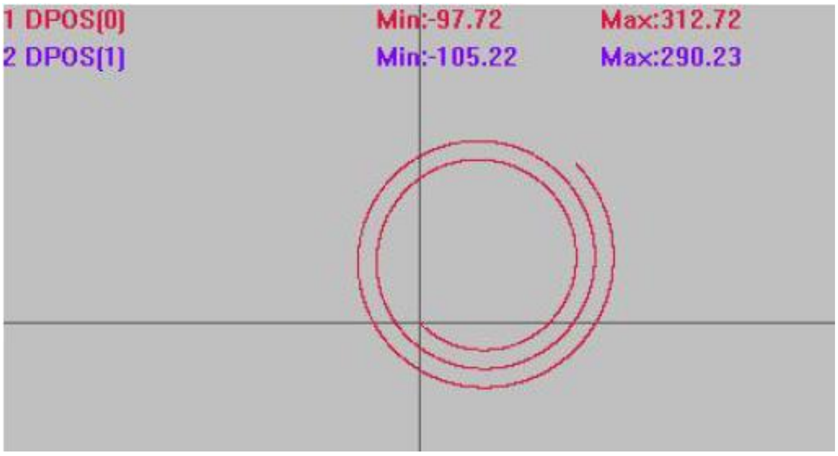
Interpolation trajectory of axis 0, axis 1 and axis 2 under XYZ mode:

	
Instructions	MSPHERICAL

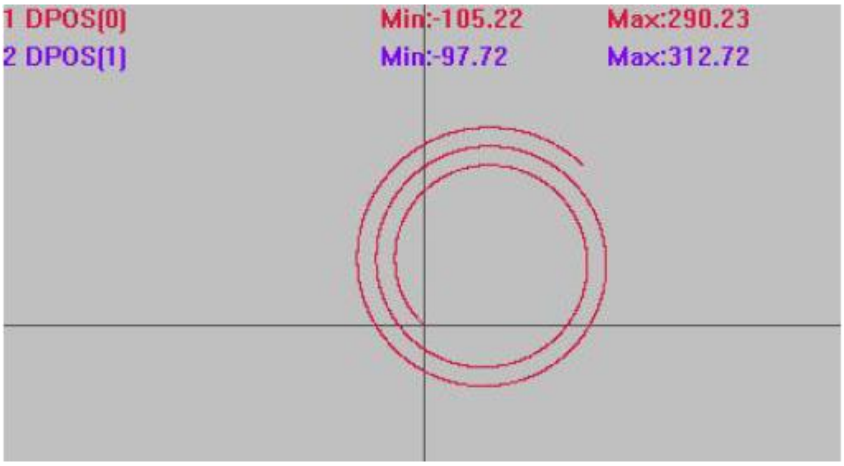
MOVESPIRAL – Involute Arc

Type	Multi-Axis Motion Instruction
Description	**Involute arc interpolation movement, relative motion mode, helical is optional.** Distance between present point and central point will determine the start radius, if the start radius is 0, then angel can’t be determined, it will start from angel 0 directly, see the example 1 for reference. This instruction can be used in continuous interpolation movements by adding SP, see *SP for reference.
Grammar	MOVESPIRAL(centre1,centre2,circles,pitch[,distance3][,distance4]) Parameters: centre1: central point coordinate-aixs1, relative. centre2: central point coordinate-aixs2, relative. circles: circles amount, integral or decimal. Minus value means clockwise, end point of each circle is the one point of the line between start point and central point. pitch: diffusion distance of each circle, which can be minus value. distane3: add the third axis as helical motion, appoint the relative motion distance of axis 3. This axis is not involved in speed calculation.

	distane4: add the fourth axis as helical motion, appoint the relative motion distance of axis 4. This axis is not involved in speed calculation.
Controller	General
Example	BASE(0,1,2) ATYPE=1,1,1 'set type as pulse UNITS=100,100,100 DPOS=0,0,0 SPEED=100,100,100 'main axis speed ACCEL=1000,1000,1000 'main axis acceleration DECEL=1000,1000,1000 TRIGGER 'Trigger the oscilloscope automatically **Example 1 diffusion starts from start point** **MOVESPIRAL**(0,0,2.5,30) 'set start point as central point, rotate 2.5 circles anticlockwise, diffusion distance of each circle is 30. Interpolation path DPOS(0) vertical scale 100 DPOS(1) vertical scale 100  **Example 2 no helical motion** **MOVESPIRAL**(100,100,2.5,30) 'start radius is 100, central point is (100,100), rotate 2.5 circles anticlockwise, diffusion distance of each circle is 30. Interpolation Path (if path circle amount is not full displayed, make captured gap proper bigger) DPOS(0) vertical scale 300 DPOS(1) vertical scale 300



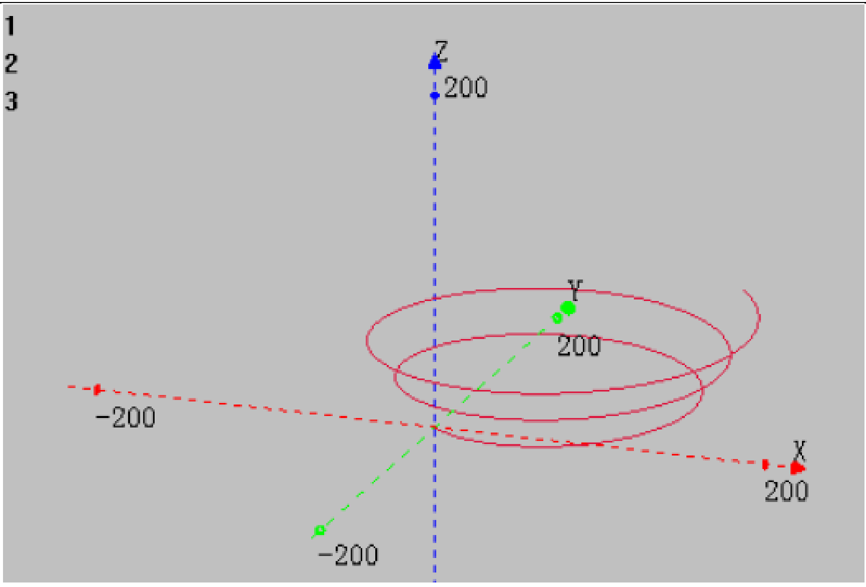
MOVESPIRAL (100,100,-2.5,3.)   'When the number of rotations is negative (-2.5), rotate clockwise



Example 3   with helical motion

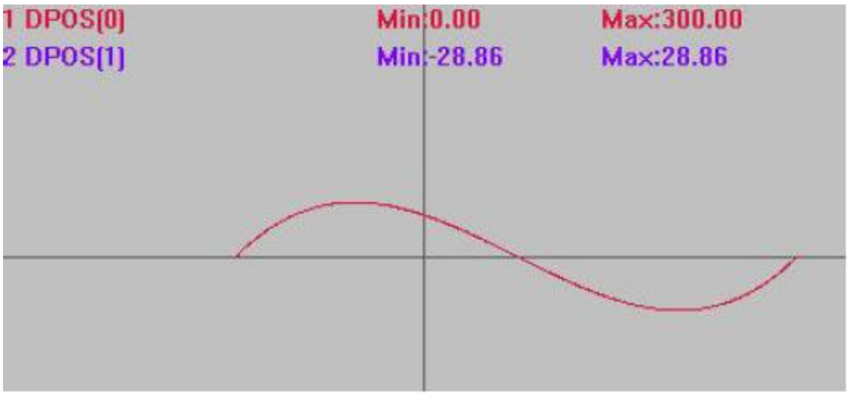
MOVESPIRAL(100,100,2.5,30,100)   'start radius is 100, central point is (100,100), rotate 2.5 circles anticlockwise, diffusion distance of each circle is 30, Axis Z moves upwards to 100.

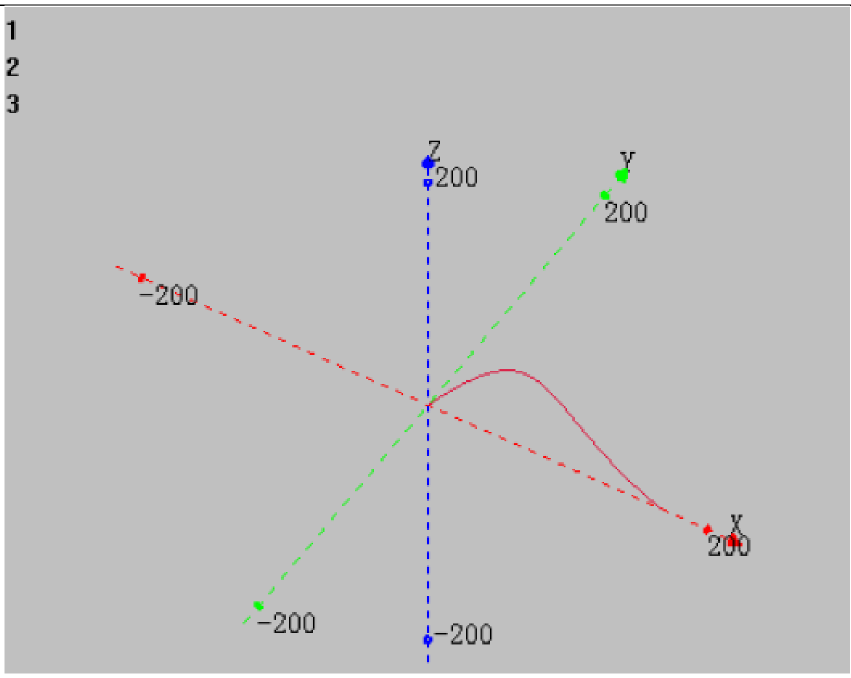
Axis 0, axis 1 and axis 2 interpolation trajectory under XYZ mode:



MOVESPLINE/MOVESPLINEABS -- Spline Interpolation

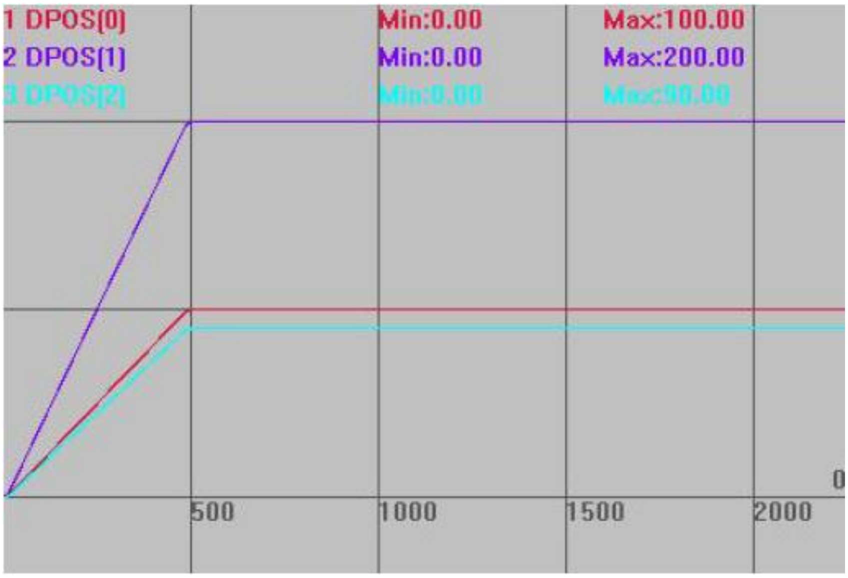
Type	Special Motion Instruction
Description	**Spline interpolation, relative or absolute motion.** Fill the spline points data into **TABLE** in advance. This instruction doesn't support SP function, continuous interpolation with self-defined speed can be set by instructions: BIT8 of CONNER_MODE.
Grammar	MOVESPLINE (axes,mode ,dtendcontrol4, dtcontrol2, dtcontrol3) axes: the number of interpolation axes mode: mode, 0 means 3 Layer Bezier Splines is used. dtendcontrol4: table index of fourth control point. for Bessel spline, it means the end point. dtcontrol2: table index of second control point. dtcontrol3: table index of third control point. For Bessel spline, present point is the first control point.
Controller	ZMC4XX series with firmware version above 170507. ZMC306X with firmware version above 161208.
Example	**Example 1:** BASE(0,1) DPOS=0,0 ATYPE=1,1, 'set type as pulse SPEED=100,100 'main axis speed ACCEL=1000,1000 'main axis acceleration DECEL=1000,1000 TRIGGER CORNER_MODE=2 + 256 'set SP motion, use FORCE_SPEED. FORCE_SPEED=100 TABLE(0,100,100) 'TABLE(0) and TABLE(1) will store the second

	control point, relative to start point. TABLE(10,200,-100) 'TABLE(10) and TABLE(11) will store the third control point, relative to start point. TABLE(20,300,0) 'TABLE(20) and TABLE(21) will store the fourth control point, distance of end point. **MOVESPLINE(2, 0, 20, 0, 10)** '2 axes relative spline interpolation. Interpolation path: DPOS(0) vertical scale 100, offset -100 DPOS(1) vertical scale 100, no offset  **Example 2:** BASE(0,1,2) ATYPE=1,1,1 ‘set type as pulse axis UNITS=100,100,100 DPOS=0,0,0 SPEED=100,100,100 ‘set main axis speed ACCEL=1000,1000,1000 ‘set main axis acceleration DECEL=1000,1000,1000 TRIGGER TABLE(0,100,100,100) 'TABLE(0) and TABLE(1) will store the second control point, relative to start point. TABLE(10,200,-100,200) 'TABLE(10) and TABLE(11) will store the third control point, relative to start point. TABLE(20,300,0,0) 'TABLE(20) and TABLE(21) will store the fourth control point, distance of end point. **MOVESPLINE(3, 0, 20, 0, 10)** '2 axes relative spline interpolation. Axis 0, axis 1 and axis 2 interpolation trajectory under XYZ mode:

	
Instructions	CORNER_MODE

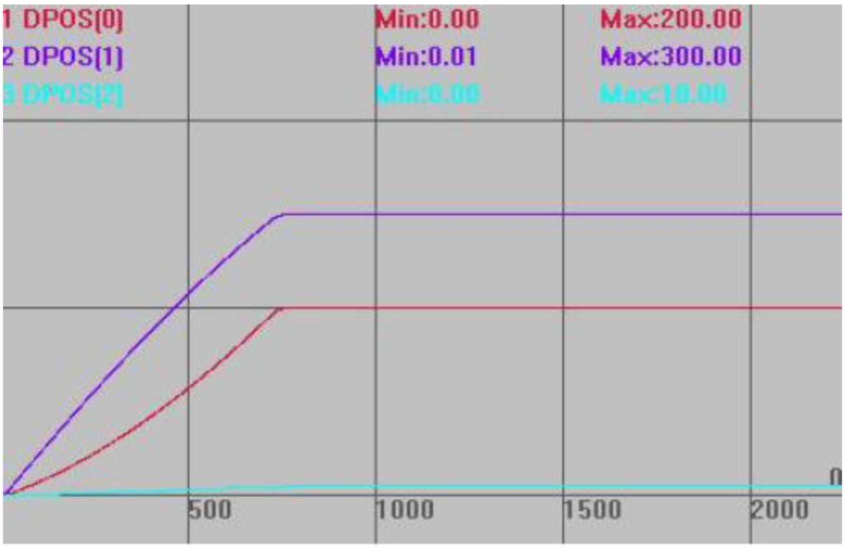
MOVE_TURNABS-Rotating Stage Interpolation

Type	Multi-Axis Motion Instruction
Description	**Rotating Stage Interpolation - ensure motion on stage is linear.** The rotation function means that the work platform rotates on a plane parallel to XY, and the positive direction of rotation should be consistent with the positive direction of XY (right-hand rule). The rotation parameters are stored in the TABLE, which stores the R axis number as order, the number of pulses per revolution of the R axis, the X axis number, the Y axis number, the X circle center, and the Y circle center. This instruction can be used in continuous interpolation movements by adding SP, see *SP for reference. It is recommended to use robotic algorithm directly, see *ZMOTION Robotic Instructions Reference* for reference, the related frame is frame11/17.
Grammar	MOVE_TURNABS(tablenum,position1[,position2[,position3[, position4...]]]) Parameters: tablinum: Table NO. which saves rotating parameters. position1: coordinate of the first axis position2: coordinate of next axis
Controller	General
Example	BASE(0,1,2) ATYPE=1,1,1 'set type as pulse UNITS=100,100,100 DPOS=0,0,0 SPEED=100,100,100 'main axis speed

	ACCEL=1000,1000,1000 'main axis acceleration DECEL=1000,1000,1000 'main axis deceleration TABLE(0, 3, 3600, 0,1, 0,0) 'set parameters of rotating stage. TRIGGER **MOVE_TURNABS**(0,100,200,90) 'move to target position by linear. WAIT IDLE 'wait until the motion stops. Interpolation Path: DPOS(0) vertical scale 100 DPOS(1) vertical scale 100 DPOS(2) vertical scale 100 
Instructions	MCIRC_TURNABS

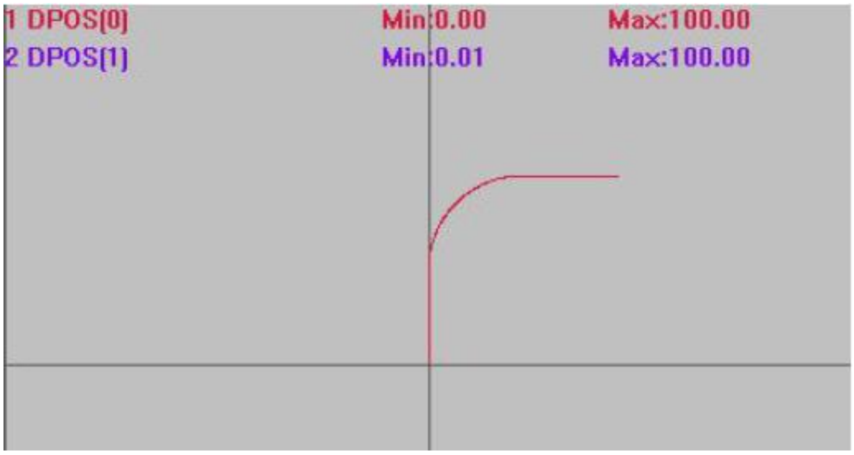
MCIRC_TURNABS-Rotating Stage Interpolation-Absolute

Type	Multi-Axis Motion Instruction
Description	**Rotating Interpolation-ensure motion on stage is circular.** The rotation function means that the work platform rotates on a plane parallel to XY, and the positive direction of rotation should be consistent with the positive direction of XY (right-hand rule). The rotation parameters are stored in the TABLE, which stores the R axis number as order, the number of pulses per revolution of the R axis, the X axis number, the Y axis number, the X circle center, and the Y circle center. This instruction can be used in continuous interpolation movements by adding SP, see *SP for reference.
Grammar	MCIRC_TURNABS(tablenum, refpos1, refpos2, mode, end1, end2 [, dis3, dis4, dis5]) tablinum: Table NO. which saves rotating parameters. refpos1: reference point of the first axis, absolute position

	refpos2: reference point of the second axis, absolute position mode: 1-the reference point is before the current point 2-the reference point is behind the end point 3-the reference point is in the middle It uses the method of three-point circle. end1: the end point of the first axis, absolute position end1: the end point of the second axis, absolute position dis3: the end position of rotating axis
Controller	General
Example	Base(0,1,2) ATYPE=1,1,1 'set type as pulse UNITS=100,100,100 DPOS=0,0,0 SPEED=100,100,100 'main axis speed ACCEL=1000,1000,1000 'main axis acceleration DECEL=1000,1000,1000 Table(0, 3, 3600, 0,1, 0,0) 'set parameters of rotating stage TRIGGER TURN_POSMAKE(0,100,200,5,10) **MCIRC_TURNABS**(0,table(10),table(11),3,200,300,10) '3 axes also rotates when circular is in process WAIT IDLE Interpolation Path: DPOS(0) vertical scale 200 DPOS(1) vertical scale 200 DPOS(2) vertical scale 200 
Instructions	MOVE_TURNABS, TURN_POSMAKE

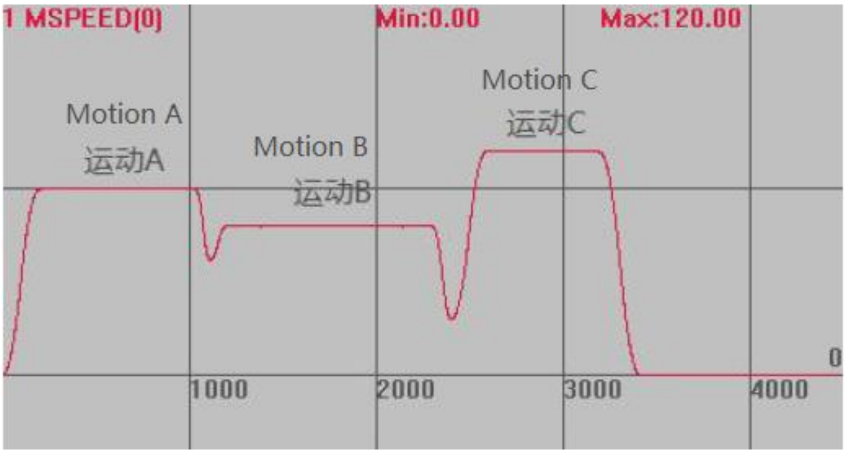
MOVESMOOTH-Fillet

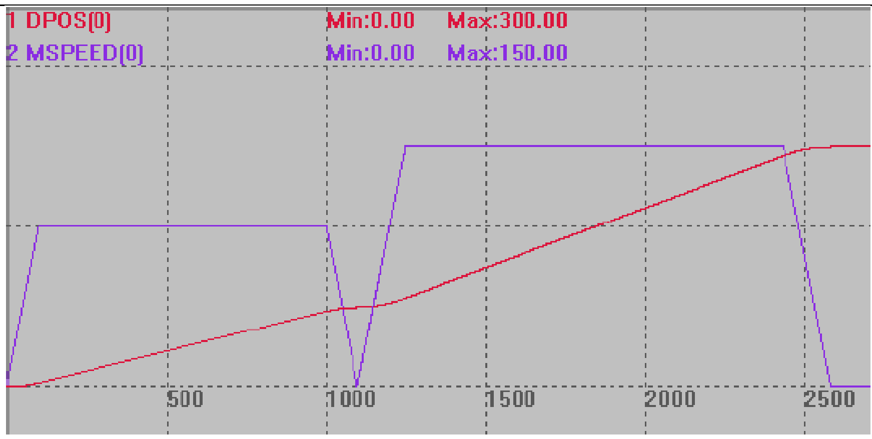
Type	Multi-Axis Motion Instruction
Description	**Space Linear Fillet Motion.** Insert arc at the turning angle depends on absolute coordinate of next linear motion, once arc was inserted, the final end point of the motion will be different from end point of the linear. If the turning angle is too big, arc will not be inserted, radius will be reduced automatically when distance is not enough. This instruction can be used in continuous interpolation movements by adding SP, see *SP for reference. This is an instruction developed early, so there is limit for axes, it is recommended to use CORNER_MODE because its function is more.
Grammar	MOVESMOOTH (end1, end2, end3, next1, next2, next3, radius) Parameters: end1 absolute coordinate of axis1; end2 absolute coordinate of axis2; end3 absolute coordinate of axis3; next1 absolute coordinate of next straight line, axis1; next2 absolute coordinate of next straight line, axis2; next3 absolute coordinate of next straight line, axis3; radius the radius of the inserted arc, it will minish if too big.
Controller	General
Example	BASE(0,1,2) ATYPE=1,1,1 'set type as pulse UNITS=100,100,100 DPOS=0,0,0 SPEED=100,100,100 'main axis speed ACCEL=1000,1000,1000 'main axis acceleration DECEL=1000,1000,1000 'main axis deceleration TRIGGER 'Trigger the oscilloscope automatically **MOVESMOOTH** (0,100,0,100,100,0,50) 'after the arc was inserted, the actual movement reaches (50,100,0) MOVEABS(100,100,0) Interpolation path: DPOS(0) vertical scale 100 DPOS(1) vertical scale 100

*SP-Motion Independent Speed

Type	Multi-Axis Motion Instruction
Description	**It is used to set starting speed, running speed and end speed of every stage of motion.** Multi-Axis motion instructions have related SP instructions. Now it can use FORCE_SPEED, ENDMOVE_SPEED and STRATMOVE_SPEED to set motion speed, end speed and start speed. If there is no need to set speed of every motion, then no need to use SP instruction.
Grammar	SP based instructions: MOVESP, MOVEABSSP, MOVECIRCSP, MOVECIRCABSSP, MHELICALSP, MHELICALABSSP, MECLIPSESP, MECLIPSEABSSP, MSPHERICALSP. FORCE_SPEED, ENDMOVE_SPEED and STRATMOVE_SPEED will enter motion buffer.
Controller	General
Example	**Example 1** BASE(0) DPOS=0 ATYPE=1 UNITS=100 ACCEL=1000 DECEL=1000 SRAMP=100 MERGE=ON 'open continuous interpolation. SPEED=100 'motion speed is 100 FORCE_SPEED=80 'limit speed is 80 STARTMOVE_SPEED=60 'start speed is 60 ENDMOVE_SPEED=30 'end speed is30 TRIGGER 'Trigger the oscilloscope automatically MOVE(100) 'motion A, no SP limit.

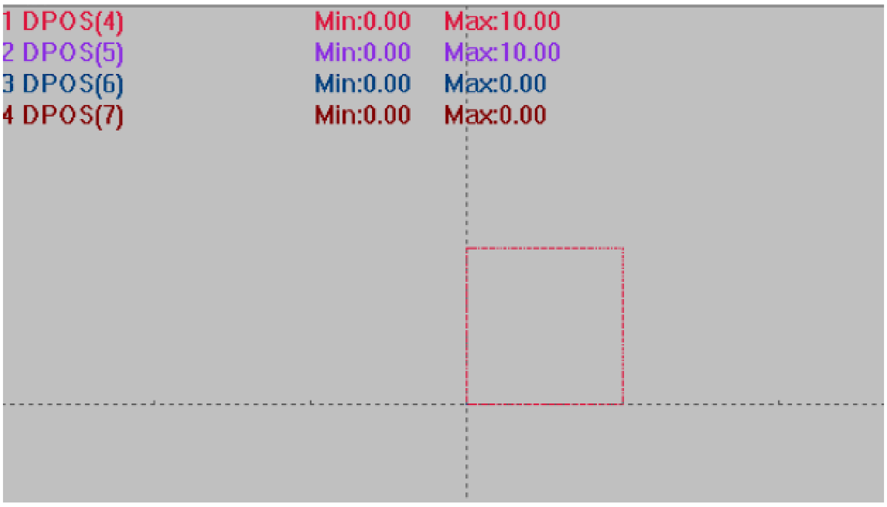
	MOVESP(100) 'motion B, use SP limit. FORCE_SPEED=120 'speed limit is 120 ENDMOVE_SPEED=30 'end speed limit is 30 **MOVESP**(100) 'motion C, use SP limit. Speed Path MSPEED(0) vertical scale 100  The motion speed is SPEED when there is no SP limit, the motion speed is FORCESPEED when there is SP limit. When Both STARTMOVE_SPEED and ENDMOVE_SPEED are set, STARTMOVE_SPEED will take effect in priority. **Example 2** BASE(0) DPOS=0 ATYPE=1 UNITS=100 ACCEL=1000 DECEL=1000 SPEED=100 ‘running speed is 100 SRAMP=100 ‘S curve FORCE_SPEED=150 ‘speed limit is 120 TRIGGER MOVE(100) ‘motion speed is SPEED MOVESP(200) ‘motion speed is FORCE_SPEED Motion trajectory: DPOS (0) = 200 (vertical scale) MSPEED (0) =100 (vertical scale)

	
Instructions	FORCE_SPEED, ENDMOVE_SPEED, STRATMOVE_SPEED

MOVESCAN – Galvanometer (SCAN) Motion

Type	Motion Instruction
Description	**The Motion command is without acceleration and deceleration, and it supports time control at the us level.** The running time is directly calculated through FORCE_SPEED and vector distance. For example, the SCAN vector distance is 1, FORCE_SPEED=10000, then the motion time is 1/10000, the unit is s, namely 100us. Valid in galvanometer controllers with firmware version above 20180714. Under this motion, corner delay means the maximum corner delay, and ZSMOOTH indicates the actual delay time is linearly distributed between DECEL_ANGLE and STOP_ANGLE. Bit1 of CORNER_MODE sets whether the corner delay is used or not, if it sets, ZSMOOTH sets max delay time, the unit is us, then this motion meets corner condition dely. Time control at the us level can be achieved together with MOVE_WAIT and MOVE_OP. Non-SCAN axis also can be used, but it needs to control the speed in sections to do acceleration and deceleration.
Grammar	MOVESCAN(pos1[,pos2][,pos]…) pos1: motion distance of the first axis pos2: motion distance of the next axis
Controller	Galvanometer controller
Example	**Example 1** BASE(4,5) AXIS_ZEST=2 'open precision output TRIGGER CORNER_MODE=0 'no corner delay MOVE_PAUSE(3) 'force to stop MOVE_OP(0,1)

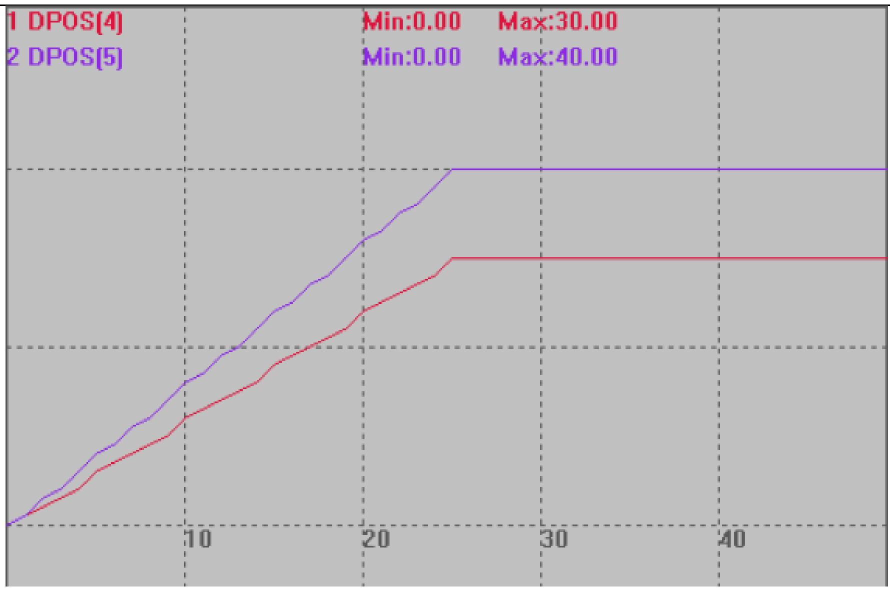
<table>
<tr><td></td><td>FORCE_SPEED=10000
MOVESCANABS(0,0)
MOVESCANABS(10,0)   'galvanometer motion, time: 10/10000=1000us
MOVESCANABS(10,10)   'galvanometer motion
MOVESCANABS(0,10)   'galvanometer motion
MOVESCANABS(0,0)   'galvanometer motion
MOVE_DELAY(0.25)   'delay 250us
MOVE_OP(0,0)   'output
MOVE_RESUME
END

Resultant trajectory under galvanometer axis XY mode:
DPOS(4), vertical scale (Y scale): 10
DPOS(5), vertical scale (Y scale): 10


Example 2
BASE(4,5)
AXIS_ZSET=2
CORNER_MODE=2   'corner delay
ZSMOOTH=100   'maximum corner delay 100us
DECEL_ANGLE = 25 * (PI/180) 'set the start deceleration corner, in radians
STOP_ANGLE = 90 * (PI/180)   'set the end deceleration corner, in radians
MOVE_PAUSE(3)
MOVE_OP(0,1)
FORCE_SPEED=10000
MOVESCAN(1,0)   'time of motion 100us
MOVESCAN(0,1)   'add 100us corner delay time, then move 100us
MOVE_DELAY(0.25)
MOVE_OP(0,0)   'after 550us, it outputs
MOVE_RESUME</td></tr>
<tr><td>**Instructions**</td><td>MOVE</td></tr>
</table>

MPULSCAN – Galvanometer Motion 2

Type	Motion Instruction
Description	**Motion commands are without acceleration and deceleration, the unit is the number of pulses.** The running time is directly calculated through FORCE_SPEED and vector distance. For example, galvanometer vector distance is 1, FORCE_SPEED = 10000, the running time is 1/10000, the unit is s, that is, 100us. Support MOVESCANABS absolute motion. The time control at us level can be achieved when it is used together with MOVE_WAIT and MOVE_OP. Non-galvo axis can also be used, but it needs to control the speed in sections to do acceleration and deceleration. This command doesn't have corner deceleration, MOVE_DELAY must be added to achieve delay deceleration. Valid in firmware version above 20220225.
Grammar	MPULSCAN vectpul, pul1[,pul 2] [,pul 3]… vectpul: vector pulse length, calculate externally to reduce controller execution time. pul1,2,3: pulse distance or length of each axis, directly use pulse unit, no need to do UNITS conversion.
Controller	Galvanometer controller
Example	BASE(4,5) ATYPE=21,21 FORCE_SPEED=1000,1000 'galvanometer motion speed DPOS=0,0 AXIS_ZSET=2 'open precision output TRIGGER MOVE_OP(0,1) MPULSCANABS 50,30,40 'galvanometer motion, vector length is 50 pulses, axis 4 moves 30, axis 5 moves 40 MOVE_DELAY(0.2) 'delay 200us MOVE_OP(0,0) 'output END Galvanometer axis motion trajectory: DPOS(4) vertical scale 20 DPOS(5) vertical scale 20

<table>
<tr><td></td><td>
</td></tr>
<tr><td>Instructions</td><td>MOVESCAN</td></tr>
</table>

7.3 Special Motion Instruction

MOVE_PAUSE – Motion Pause

<table>
<tr><td>Type</td><td>Special Motion Instruction</td></tr>
<tr><td>Description</td><td>BASE axis motion pause.
It is valid when single axis or multi axes interpolation movement, axes will pause simultaneously while multi axes coordination.
Use AXISSTATUS to check if any motion is paused.
If axes already paused or stopped, there is alarm output after calling this instruction, but will not affect procedure process. Some motions don’t support pause, such as, VMOVE, synchronization motion instructions, etc.</td></tr>
<tr><td>Grammar</td><td>MOVE_PAUSE (mode)

<table>
<tr><td>0 （default）</td><td>Pause the present motion.</td></tr>
<tr><td>1</td><td>Pause when the present motion finished completely.</td></tr>
<tr><td>2</td><td>Pause when present motion is finished completely and MARK of present motion instruction is different from the following motion instruction.
This mode can be used to suspend one motion which consist of multiple instructions when it is finished.</td></tr>
<tr><td>3</td><td>Pause mandatorily, even pause while IDLE mode is in process.
This mode is only supported in controller with firmware version above 20170513.</td></tr>
</table></td></tr>
<tr><td>Controller</td><td>General</td></tr>
</table>

Example	BASE(0) DPOS=0 SPEED=100 Example1 mode 0 MOVE(1000) 'motion in process MOVEABS(-100) 'motion in buffer **MOVE_PAUSE**(0) 'mode 0, pause motion in process ?DPOS(0) 'print result,0 'the present motion only executes for a short time. Then pause when MOVE_PAUSE is detected during the scanning. Example 2 mode 1 MOVE(1000) 'motion in process MOVEABS(-100) 'motion in buffer MOVE_PAUSE(1) 'mode 1, pause after the present motion finished. ?DPOS(0) 'print result, 1000 'the present motion is finished before pausing. DPOS is 1000 Example 3 mode 2 MOVE_MARK=1 'define mark NO. as 1 manually. MOVE(200) 'motion in process MOVE_MARK=1 'define mark NO. the same as last motion. MOVEABS(-100) 'motion in buffer MOVEABS(100) 'mark NO. is not defined manually, plus 1 automatically. **MOVE_PAUSE**(2) 'mode 2, finish present motion first, then pause until mark of next motion differs from the present motion. DELAY(3000) 'wait until motion pause. ?DPOS(0) 'print result, -100 (present motion will not pause if the speed is too slow, the print result will over -100) 'Finish motion with same mark, pause until meet the last motion which has a different mark NO.3.
Instructions	MOVE_MARK, MOVE_RESUME, AXISSTATUS

MOVE_RESUME – Motion Resume

Type	Special Motion Instruction
Description	**Resume the motion of axes assigned by BASE from where it paused.** Use AXISSTATUS to check if any motion is paused.
Grammar	MOVE_RESUME
Controller	General
Example	BASE(0)

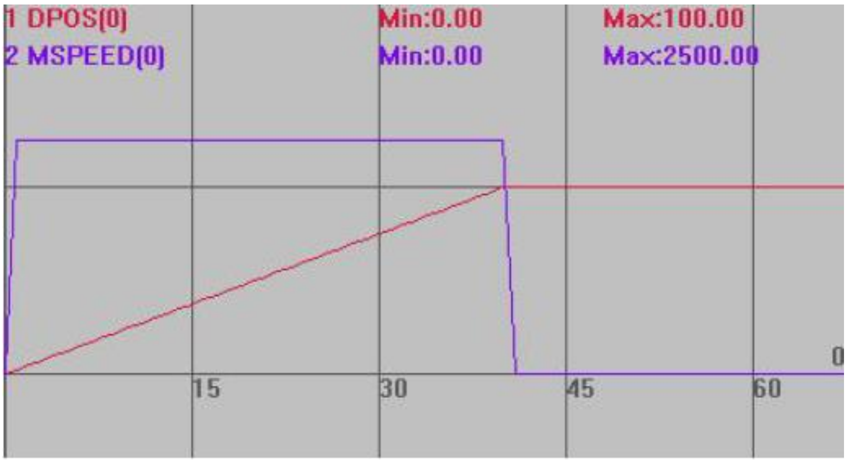
	UNITS=100 DPOS=0 SPEED=100 ACCEL=1000 DECEL=1000 MOVE(100) 'motion in process MOVE(100) 'motion in buffer MOVE_PAUSE(1) 'pause after motion in process finished. WA 2000 'wait until motion in process finished. ?DPOS(0) 'print result,100 DELAY(1000) **MOVE_RESUME** 'continue to motion. WAIT IDLE ?DPOS(0) 'print result,200
Instructions	MOVE_PAUSE, AXISSTATUS

MOVE_PT -Distance in Unit Time

Type	Special Motion Instruction
Description	**Set the distance of motor motion in a certain time.** Usually, PC will calculate relative coordinate in every period, then transfer it to controller. BASE assigned axis can be used. Motion speed=(DIS/TICKS)*1000units/s Don't let the motor run a long distance in a very short time, then the pulse frequency will be high, which will result to motor stalling. It is better to divide long distance into pieces, then send repeatedly. "multi-period speed auto-even" function is added by MOVE_PT.
Grammar	MOVE_PT (TICKS, DIS1,DIS2…) ticks: servo period numbers of time, time=system period*ticks dis1: motion distance controller SERVO_PERIOD is 1ms, TICKS = 1ms (for different SERVO_PERIOD, TICKS are different).
Controller	General
Example	**Example 1:** BASE(0) UNITS=100 DPOS= 0 SPEED=100 ACCEL=1000 DECEL=1000 TRIGGER 'trigger the oscilloscope automatically For i=0 to 9

MOVE_PT (4, 10)   'move 10 units in 4 TICKS, speed=2500 units/s.
NEXT
WAIT IDLE
PRINT*DPOS   'print result, 100

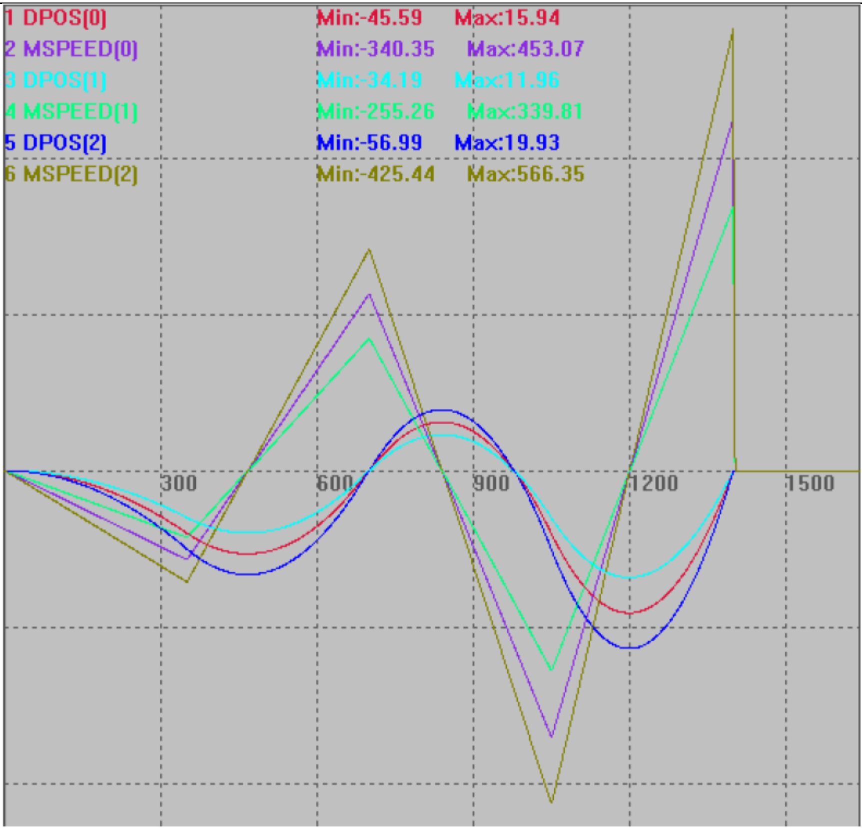
Interpolation Speed:
DPOS(0) vertical scale 100
MSPEED(0) vertical scale 2000



Example 2:
BASE(0,1,2)
UNITS=1000,1000,1000
DPOS= 0,0,0,
SPEED=10,10,10,10
ACCEL=1000,1000,1000,1000
DECEL=1000,1000,1000,1000
MERGE = ON
TRIGGER

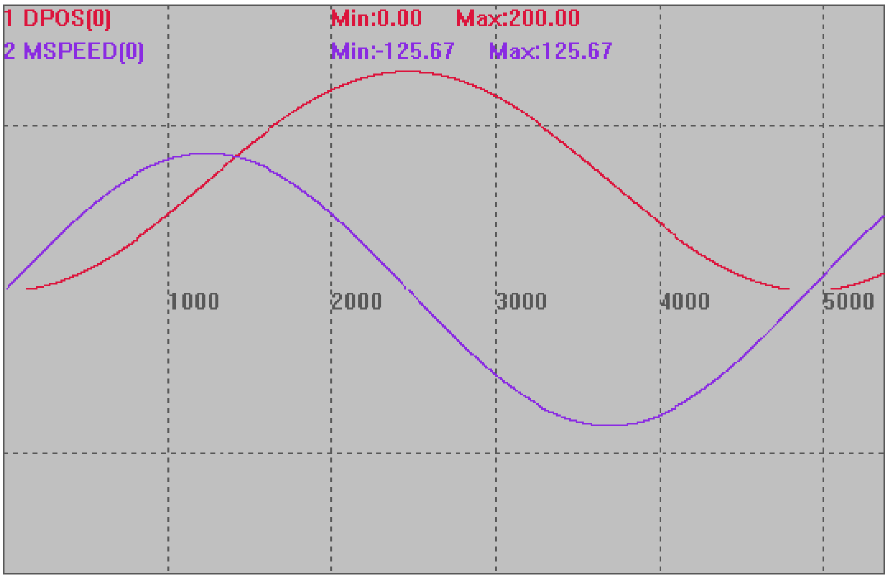
MOVE_PT(350,-20,-15,-25) 'in 350 ticks, axis 0, axis 1, axis 2 run -20, -15, -25
MOVE_PT(350,20,15,25) 'in 350 ticks, run 20,15,25
MOVE_PT(350.-20,-15,-25)
MOVE_PT(350.20,15,25)
WAIT IDLE

Speed Curve:
DPOS(0) = 50 (vertical scale)
MSPEED(0) = 200 (vertical scale)
DPOS(1) = 50 (vertical scale)
MSPEED(1) = 200 (vertical scale)
DPOS(2) = 50 (vertical scale)
MSPEED(2) = 200 (vertical scale)



Example 3:

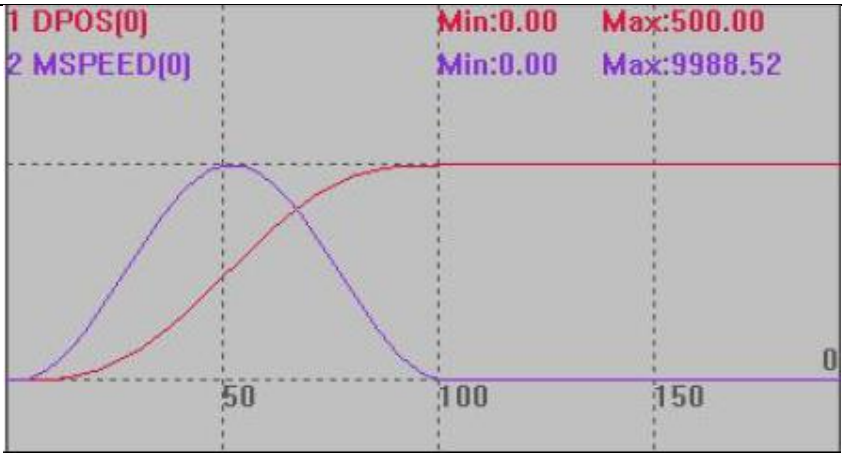
```
BASE(0)
UNITS=100
DPOS=0
SPEED=10
ACCEL=100
DECEL=100

DIM timeadd
DIM val1
DIM SPEEDval
timeadd=0

DPOS=100=COS(2*PI*0.2*timeadd+pi)+100

DELAY(1000)
TRIGGER

WHILE TRUE

        'period=2π/|ω|
        'x=A*COS(ωx+ψ)+C

        'take the derivative to get the slope, which is the speed:
```

	'v=A*ω*SIN(ωx+ψ) val1=100*COS(2*PI*0.2*timeadd+pi)+100 SPEEDval=100*2*PI*0.2*SIN(2*PI*0.2*timeadd+pi) ?"val1="val1 ?"SPEEDval="SPEEDval MOVE_PVTABS(10, val1, SPEEDval) 'MOVE_PTABS(10,val1) timeadd=timeadd+0.01 if timeadd>(2*PI/ABS(2*PI*0.2)*0.2)THEN EXIT WHILE ENDIF WEND DPOS(0), MSPEED(0), vertical scale (Y scale) is 150 
Instructions	MOVE_PTABS

MOVE_PTABS – Absolute motion distance in unit time.

Type	Special Motion Instruction
Description	**Drive the motor to reach one certain position in a period time.** Usually, PC will calculate relative coordinate to reach in every period, then transfer it to controller. Motion speed=(DIS/TICKS)*1000units/s Don't let the motor run a long distance in a very short time, then the pulse

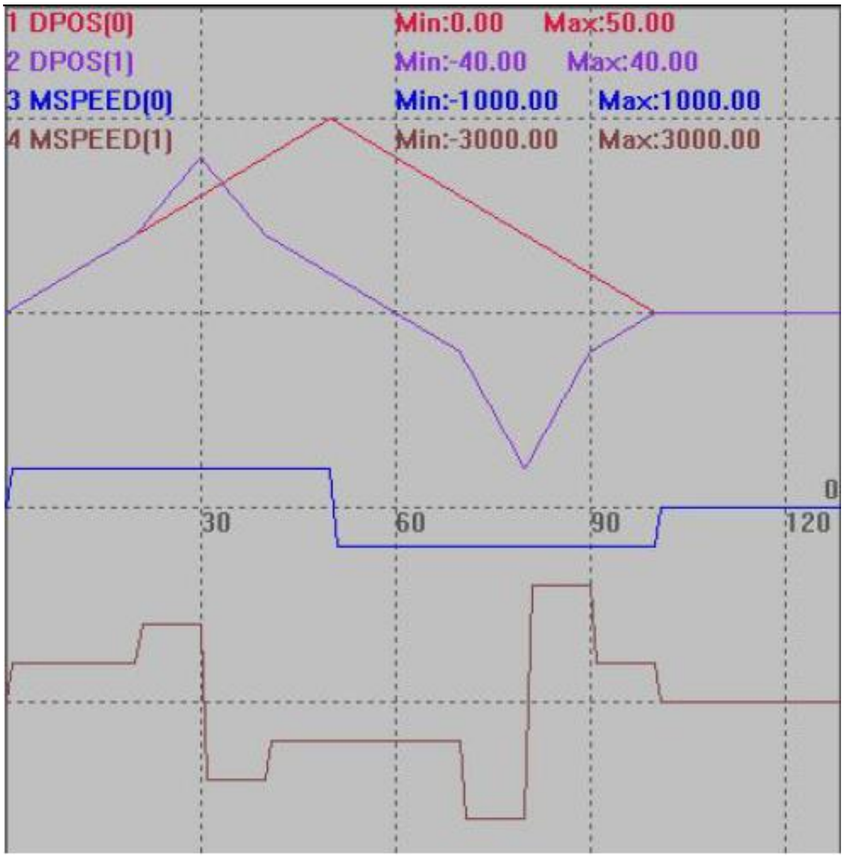
	frequency will high, which will result to motor stalling. It is better to divide long distance into pieces, then send repeatedly.
Grammar	MOVE_PTABS (TICKS, DIS1,DIS2…) ticks: servo period numbers of time dis1: motion distance
Controller	General
Example	**Example 1** Base(0,1) DPOS=0,0 **MOVE_PTABS** (3, 20,20) 'move to (20,20) in 3 ticks. WAIT IDLE PRINT*DPOS 'print result, 20,20 **Example 2** RAPIDSTOP(2) WAIT IDLE(0) BASE(0) ATYPE=1 UNITS=100 SPEED=100 ACCEL=1000 DECEL=1000 DPOS = 0 SetSine 'call function, then produce SINE curve TRIGGER 'Trigger the oscilloscope automatically FOR i=0 TO 100 MOVE_PTABS (1, TABLE(i)) 'move TABLE distance in 1TICK NEXT WAIT IDLE(0) PRINT DPOS(0) 'print result, 500 END GLOBAL SUB SetSine() 'calculate the displacement of small segment LOCAL num_p,scale 'variable definition num_p=100 scale=500 FOR p=0 TO num_p TABLE(p,((-SIN(PI*2*p/num_p)/(PI*2))+p/num_p)*scale) 'save parameters NEXT END SUB DPOS(0) vertical scale 500 MSPEED(0) vertical scale 10000



Example 3

RAPIDSTOP(2)
WAIT IDLE(0)
WAIT IDLE(1)
BASE(0,1)
ATYPE=1,1
UNITS=100,100
DPOS=0,0
TRIGGER
MOVE_PTABS (10,10,10)   'reach absolute position (10,10) in 10ticks
MOVE_PTABS (10,20,20)
MOVE_PTABS (10,30,40)
MOVE_PTABS (10,40,20)
MOVE_PTABS (10,50,10)
MOVE_PTABS (10,40,0)
MOVE_PTABS (10,30,-10)
MOVE_PTABS (10,20,-40)
MOVE_PTABS (10,10,-10)
MOVE_PTABS (10,0,0)
END

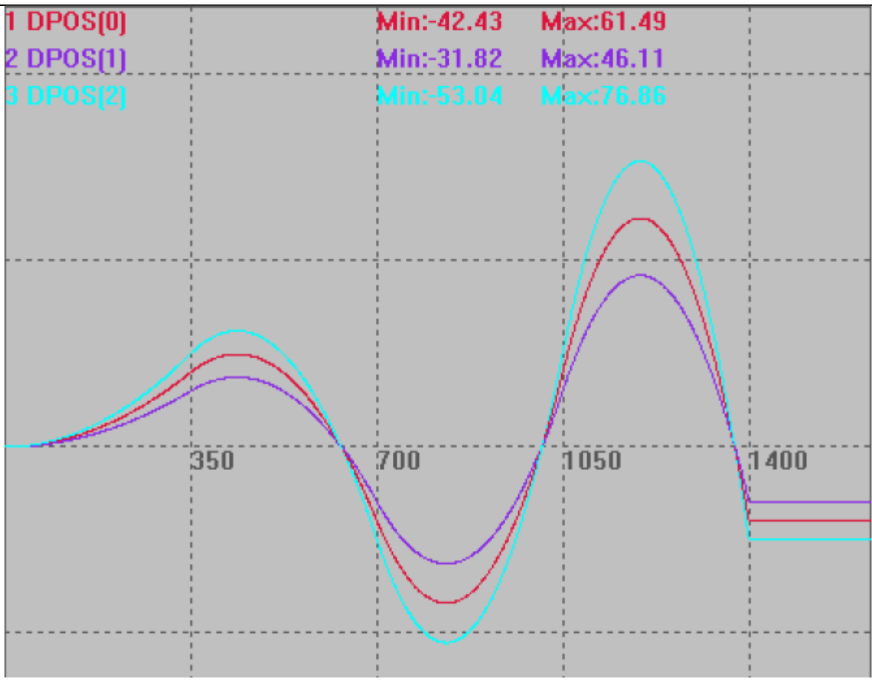
DPOS(0), DPOS(1) vertical scale 50
MAPSEED(0), MSPEED(0) vertical scale 5000



Example 4

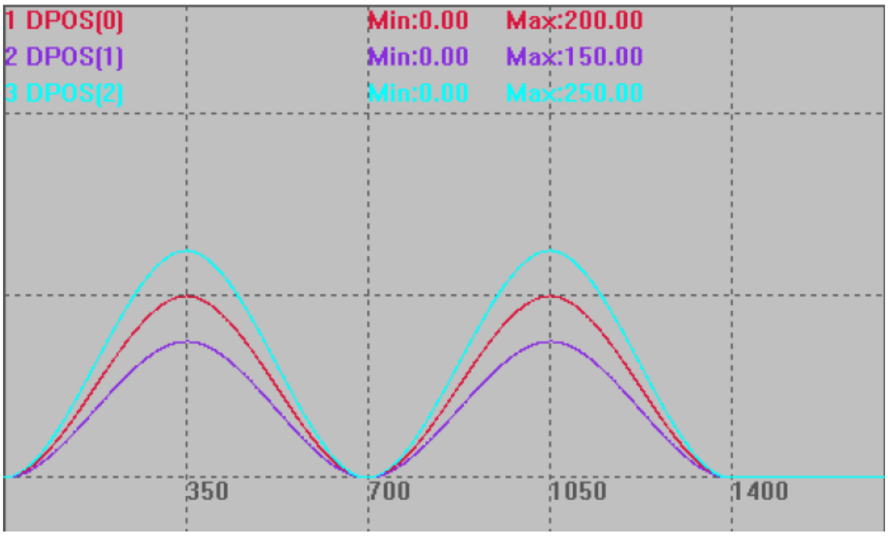
```
BASE(0)
ATYPE=1000,1000,1000
DPOS=0,0,0,0
SPEED=10,10,10,10
ACCEL=1000,1000,1000,1000
DECEL=1000,1000,1000,1000
MERGE = ON
TRIGGER
MOVE_PTABS(350,20,15,25)  'in one period, axis 0 runs to 20, axis 1 runs to
                           15, axis 2 runs to -25
MOVE_PTABS (350,-20,-15,-25)  'in one period, axis 0, axis 1 and axis 2 run
                               to position -20, -15, -25 separately
MOVE_PTABS(350, 20, 15,  25)
MOVE_PTABS(350,-20,-15,-25)
WAIT IDLE
```

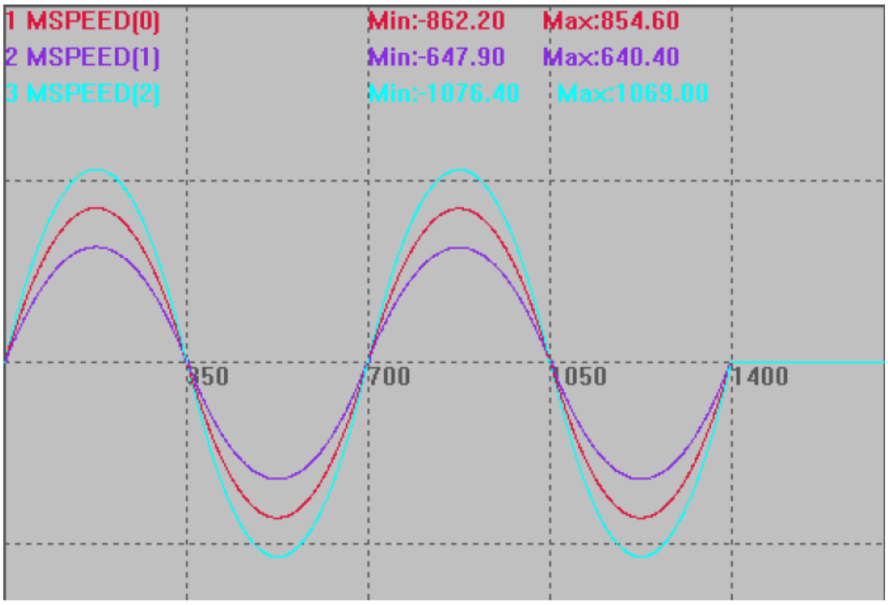
DPOS(0), DPOS(1) and DPOS(2) are 50 (vertical scale)

	
Instructions	MOVE_PT

MOVE_PVT – Unit Distance (with speed planning)

Type	Special Motion Instruction
Description	**Set the distance of motor motion in a certain time, and it is with speed planning and can assign end speed. Speed in small distance will plan automatically according to former speed and end speed, as consecutive as possible.** Usually, PC will calculate relative coordinate in every period, then transfer it to controller. BASE assigned axis can be used. Motion speed=(DIS/TICKS)*1000units/s Don't let the motor run a long distance in a very short time, then the pulse frequency will be high, which will cause motor block. It is better to divide long distance into pieces, then send repeatedly.
Grammar	MOVE_PVT (ticks, dis1, sp1, dis2, sp2…) ticks: servo period numbers of time dis1: motion distance of the first axis sp1: the end speed when first axis moved dis2: motion distance of the second axis sp2: end speed when second axis moved SERVO_PERIOD of controller is 1000us, 1 TICKS is equal to 1 ms. (ticks differ from different SERVO_PERIOD)
Controller	General
Example	**Example 1:**

	BASE(0,1,2) UNITS=10,10,10,10 DPOS= 0,0,0,0 SPEED=10,10,10,10 ACCEL=1000,1000,1000,1000 DECEL=1000,1000,1000,1000 MERGE = ON TRIGGER MOVE_PVT(350,200,10,150,10,250,10) 'in one period, axis 0, axis 1 and axis 2 run 200, 150, 250 separately, end speed is 10 MOVE_PVT(350,-200,10,-150,10,-250,10) 'in one period, axis 0, axis 1 and axis 2 run -200, -150, -250 separately, end speed is 10 MOVE_PVT(350,200,10,150,10,250,10) MOVE_PVT(350,-200,10,-150,10,-250,10) DPOS(0), DPOS(1), DPOS(2) are 200 (vertical scale)  MSPEED(0), MSPEED(1) and MSPEED(2) are 1000 (vertical scale)

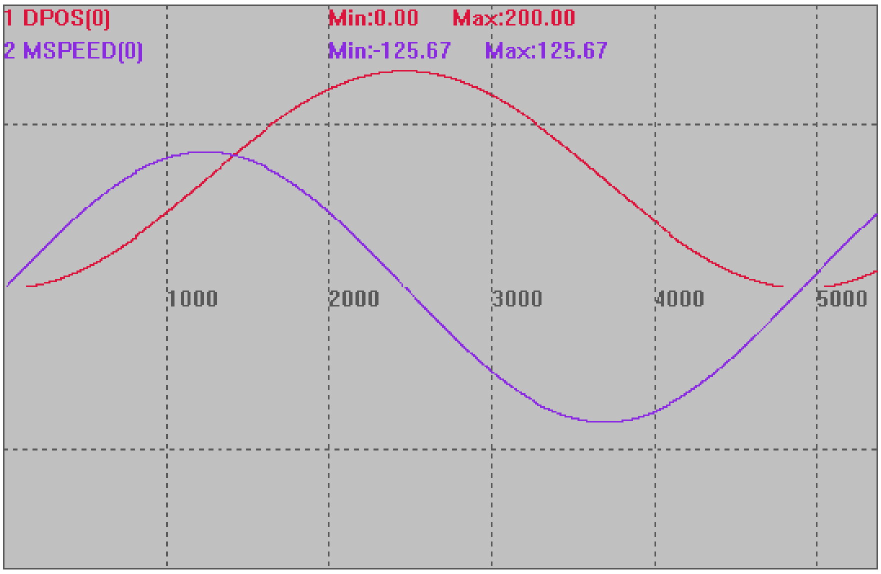


Example 2:

```
BASE(0)
UNITS=100
DPOS=0
SPEED=10
ACCEL=100
DECEL=100

DIM timeadd
DIM val1
DIM SPEEDval
timeadd=0

DPOS=100=COS(2*PI*0.2*timeadd+pi)+100

DELAY(1000)
TRIGGER

WHILE TRUE

        'period=2π/|ω|
        'x=A*COS(ωx+ψ)+C

        'take the derivative to get the slope, which is the speed:
        'v=A*ω*SIN(ωx+ψ)

        val1=100*COS(2*PI*0.2*timeadd+pi)+100
```

```
            SPEEDval=100*2*PI*0.2*SIN(2*PI*0.2*timeadd+pi)

            ?"val1="val1
            ?"SPEEDval="SPEEDval

            MOVE_PVTABS(10, val1, SPEEDval)

            'MOVE_PTABS(10,val1)

            timeadd=timeadd+0.01

            if timeadd>(2*PI/ABS(2*PI*0.2)*0.2)THEN
                        EXIT WHILE
            ENDIF
WEND
```

DPOS(0), MSPEED(0), vertical scale (Y scale) is 150

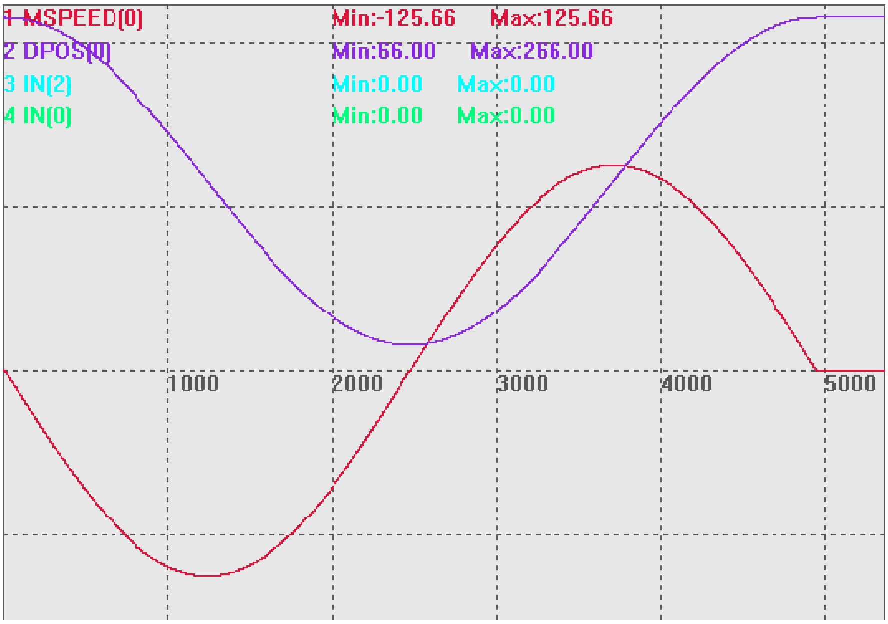


Instructions	MOVE_PT

MOVE_PVTABS – Unit Absolute Distance (with speed planning)

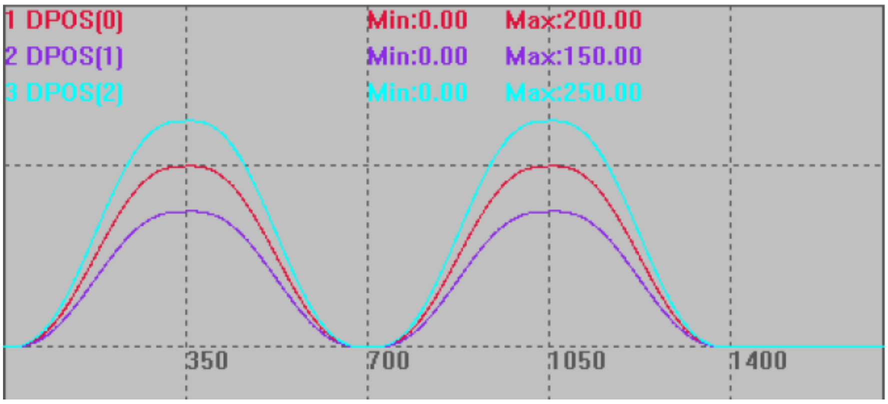
Type	Special Motion Instruction
Description	**Set the distance of motor motion in a certain time, and it is with speed planning and can assign end speed. Speed in small distance will plan automatically according to former speed and end speed, as consecutive as possible.** Usually, PC will calculate relative coordinate in every period, then transfer it to controller. BASE assigned axis can be used.

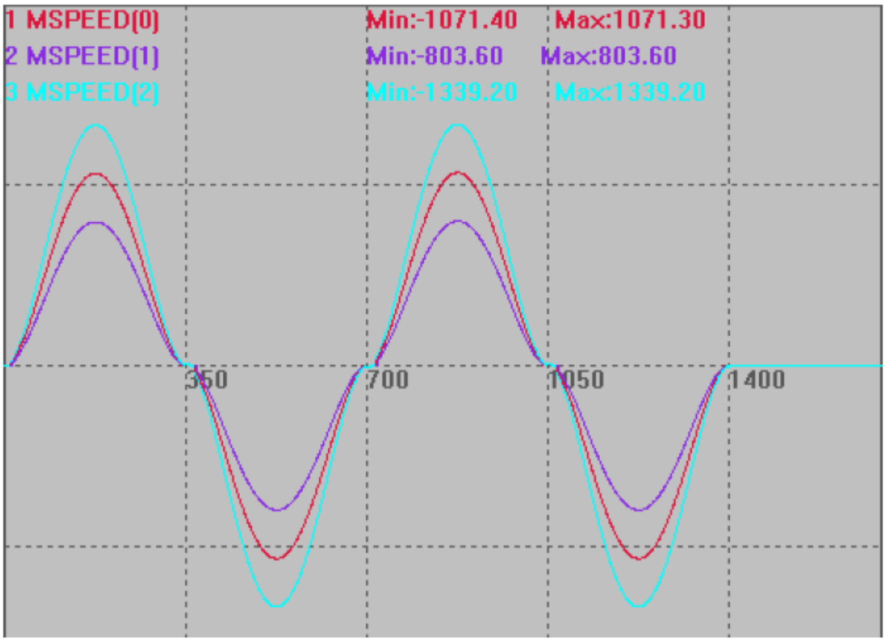
	Motion speed=(DIS/TICKS)*1000units/s Don't let the motor run a long distance in a very short time, then the pulse frequency will high, which will result to motor stalling. It is better to divide long distance into pieces, then send repeatedly.
Grammar	MOVE_PVTABS (ticks, dis1, sp1, dis2, sp2…) ticks: servo period numbers of time dis1: absolute motion distance of first axis sp1: the end speed after first axis motion dis2: absolute motion distance of the second axis sp2: end speed after second axis motion SERVO_PERIOD of controller is 1000us, then 1 TICKS equals to 1 ms. (TICKS differ from different SERVO_PERIOD)
Controller	General
Example	**Example 1:** BASE(0) UNITS=13107.2 DPOS=0 SPEED=10 ACCEL=100 DECEL=100 MAX_SPEED=8000000 DIM timeadd DIM val1 DIM SPEEDval timeadd=0 DPOS=100=COS(2*PI*timeadd*0.2)+166 DELAY(1000) TRIGGER WHILE TRUE 'period=2π/|ω| 'x=A*COS(ωx+ψ)+C 'take the derivative to get the slope, which is the speed: 'v=A*ω*SIN(ωx+ψ) val1=100*COS(2*PI*timeadd*0.2)+166 SPEEDval=100*2*PI*0.2*SIN(2*PI*timeadd*0.2) ?"val1="val1 ?"SPEEDval="SPEEDval MOVE_PVTABS(10, val1, SPEEDval)

	'MOVE_PTABS(10,val1) timeadd=timeadd+0.01 if timeadd>(2*PI/ABS(2*PI*0.2))THEN EXIT WHILE ENDIF WEND MSPEED(0), no offset, vertical scale (Y scale) is 100 DPOS(0), offset -50, vertical scale (Y scale) is 100 
Instructions	MOVE_PTABS

MOVE_PVTPP – Distance of unit time

Type	Special Motion Instruction
Description	**Same as MOVE_PVT, set the distance of motor motion in a certain time, and it is with speed planning. Accelerations of start and end moments can be sure as 0 through planning speed, and it can be used for point motions of assigned time. MOVE_PVT only can be used to send continuous and small distance motions, for long distance motions, please use MOVE_PVTPP.** Usually, PC will calculate relative coordinate in every period, then transfer it to controller. BASE assigned axis can be used. Motion speed=(DIS/TICKS)*1000units/s Don't let the motor run a long distance in a very short time, then the pulse frequency will high, which will result to motor stalling. It is better to divide long distance into pieces, then send repeatedly.
Grammar	MOVE_PVTPP (ticks, dis1, sp1, dis2, sp2…) ticks: servo period numbers of time

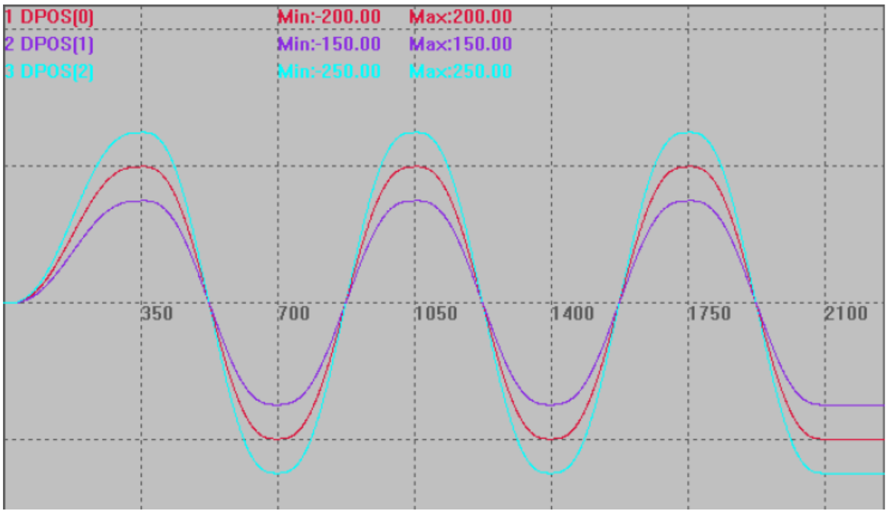
	dis1: motion distance of first axis sp1: the end speed when first axis moved dis2: motion distance of the second axis sp2: end speed when second axis moved SERVO_PERIOD of controller is 1000us, 1 TICKS is equal to 1 ms. (ticks differ from different SERVO_PERIOD)
Controller	General
Example	**Example 1:** BASE(0,1,2) UNITS=10,10,10,10 DPOS= 0,0,0,0 SPEED=10,10,10,10 ACCEL=1000,1000,1000,1000 DECEL=1000,1000,1000,1000 MERGE = ON TRIGGER MOVE_PVTPP(350,200,10,150,10,250,10) 'in one period, axis 0, axis 1 and axis 2 run 200, 150, 250 separately, end speed is 10 MOVE_PVTPP(350,-200,-10,-150,10,-250,-10) 'in one period, axis 0, axis 1 and axis 2 run -200, -150, -250 separately, end speed is 10 MOVE_PVTPP(350,200,10,150,10,250,10) MOVE_PVTPP(350,-200,-10,-150,-10,-250,-10) WAIT IDLE DPOS(0), DPOS(1), DPOS(2) are 200 (vertical scale)  MSPEED(0), MSPEED(1) and MSPEED(2) are 1000 (vertical scale)

	
Instructions	MOVE_PT, MOVE_PVT

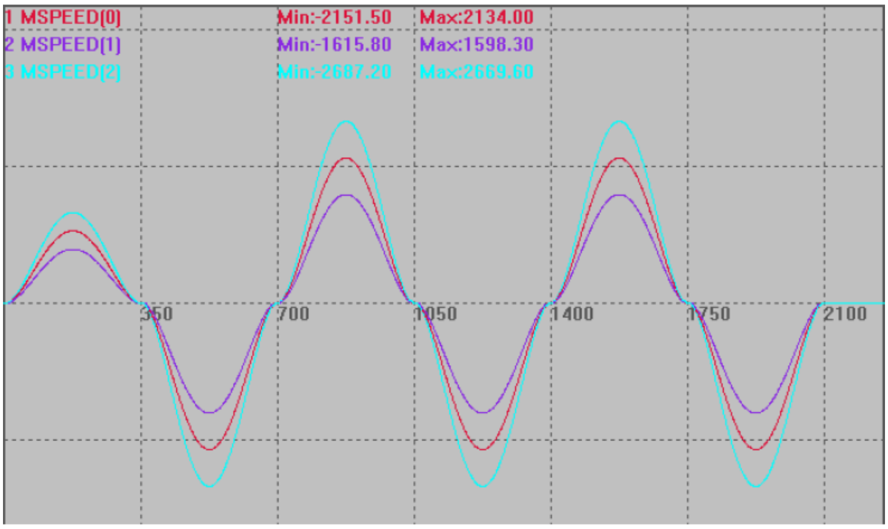
MOVE_PVTPPABS – Distance of unit time

Type	Special Motion Instruction
Description	**Same as MOVE_PVT, set the distance of motor motion in a certain time, and it is with speed planning. Accelerations of start and end moments can be sure as 0 through planning speed, and it can be used for point motions of assigned time. MOVE_PVT only can be used to send continuous and small distance motions, for long distance motions, please use MOVE_PVTPP.** Usually, PC will calculate relative coordinate in every period, then transfer it to controller. BASE assigned axis can be used. Motion speed=(DIS/TICKS)*1000units/s Don't let the motor run a long distance in a very short time, then the pulse frequency will high, which will result to motor stalling. It is better to divide long distance into pieces, then send repeatedly.
Grammar	MOVE_PVTPPABS (ticks, dis1, sp1, dis2, sp2…) ticks: servo period numbers of time dis1: motion distance of first axis sp1: the end speed when first axis moved dis2: motion distance of the second axis sp2: end speed when second axis moved SERVO_PERIOD of controller is 1000us, 1 TICKS is equal to 1 ms. (ticks differ from different SERVO_PERIOD)

<table>
<tr><td>Controller</td><td>General</td></tr>
<tr><td>Example</td><td>Example 1:
BASE(0,1,2)
UNITS=10,10,10,10
DPOS= 0,0,0,0
SPEED=10,10,10,10
ACCEL=1000,1000,1000,1000
DECEL=1000,1000,1000,1000
MERGE = ON
TRIGGER
MOVE_PVTPPABS(350,200,10,150,10,250,10)
'in one period, axis 0, axis 1 and axis 2 run 200, 150, 250 separately, end speed is 10, acceleration decrease to be 0
MOVE_PVTPPABS(350,-200,-10,-150,10,-250,-10,10)
'in one period, axis 0, axis 1 and axis 2 run -200, -150, -250 separately, end speed is 10, acceleration decrease to be 0
MOVE_PVTPPABS(350,200,10,150,10,250,10)
MOVE_PVTPPABS(350,-200,10,-150,10,-250,10)
WAIT IDLE

DPOS(0), DPOS(1), DPOS(2) are 200 (vertical scale)


MSPEED(0), MSPEED(1) and MSPEED(2) are 2000 (vertical scale)</td></tr>
</table>

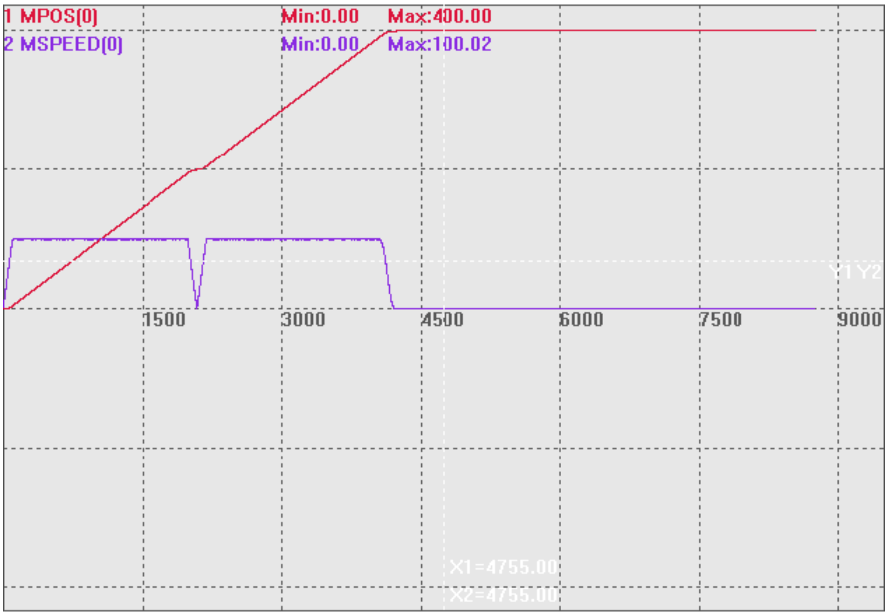
	
Instructions	MOVE_PTABS, MOVE_PVT

MOVE_PTP – Point to Point

Type	Special Motion Instruction
Description	**This is the linear interpolation motion command used for point-to-point motion.** This command doesn't support speed ahead in continuous motion, and it uses each axis' SPEED, then speed parameter will enter motion buffer automatically. This command doesn't support modifying online commands dynamically, but it can use SPEED_RATIO to adjust speed, and VP_MODE configuration is valid.
Grammar	MOVE_PTP (mode, dis1, dis2……) mode: BIT0: 1 -- speed and acceleration are calculated by each axis' speed and acceleration limits. BIT2: 1 – reserved dis1: motion distance of first motion, unit: units, support decimal dis2: motion distance of second motion, unit: units, support decimal
Controller	General
Example	**Example 1:** 'example of mode 1 single-axis and VP_MODE usage rapidstop(2) trigger speed_ratio = 1 base(0) mpos = 0 dpos = 0 atype = 1 speed = 100

```
units = 100
accel = 1000
decel = 1000
vp_mode = 0
move_ptp(1,200)
wait idle
vp_mode = 6
move_ptp(1,200)
```

MPOS(0), MSPEED(0), vertical scale (Y scale): 200



Example 2:

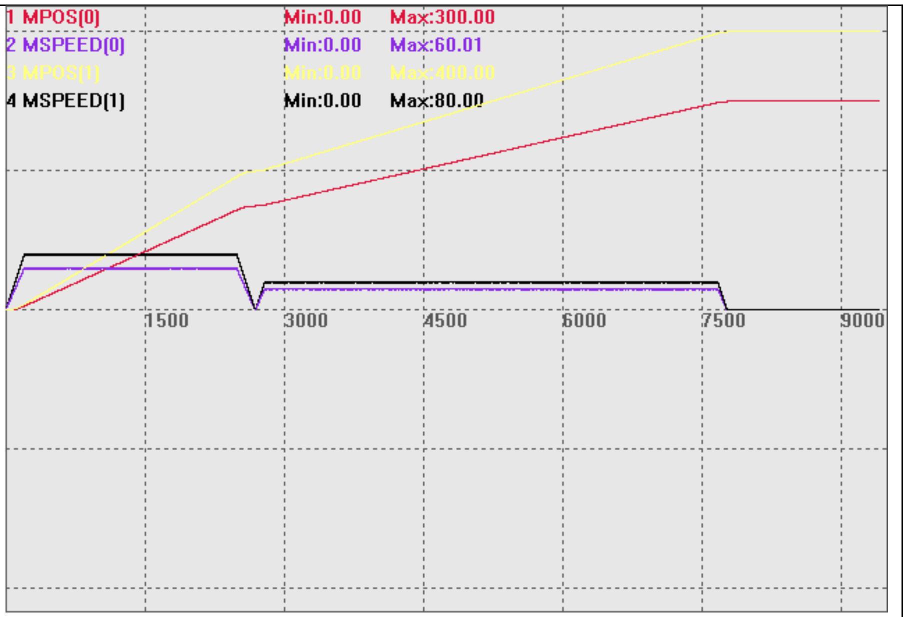
```
'example of mode 1 single-axis and speed_ratio usage
rapidstop(2)
trigger
speed_ratio = 1
vp_mode = 0,0
base(0,1)
mpos = 0,0
dpos = 0,0
atype = 1,1
speed = 100,200
units = 500,500
accel = 500,500
decel = 500,500
move_ptp(1,150,200)
wait idle
speed_ratio = 0.5
move_ptp(1,150,200)
```

MPOS(0), MSPEED(0), MPOS(1), MSPEED(1), vertical scale (Y scale): 200

Example 3:

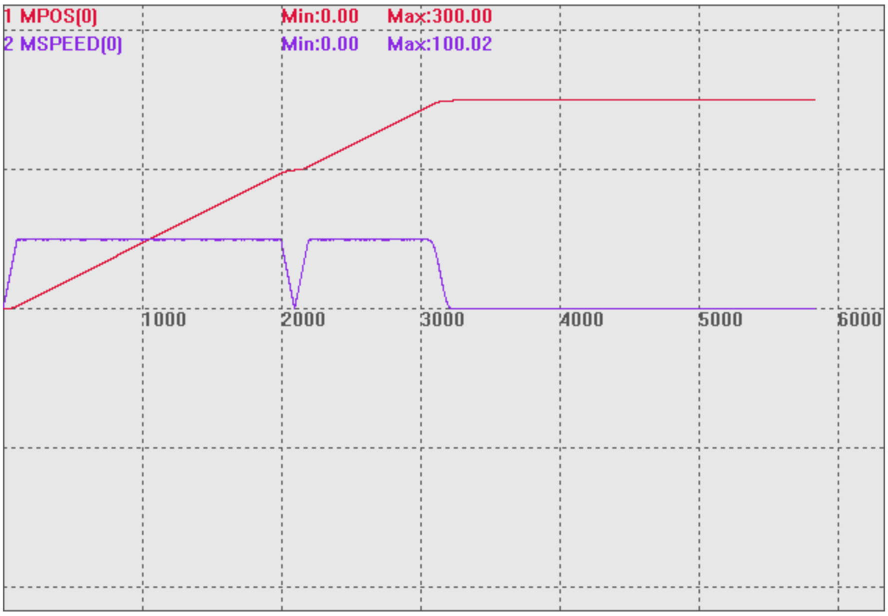
```
'example of mode 0 multi-axis and speed_ratio usage
rapidstop(2)
trigger
speed_ratio = 1
vp_mode = 0,0
base(0,1)
mpos = 0,0
dpos = 0,0
atype = 1,1
speed = 100,200
units = 500,500
accel = 500,500
decel = 500,500
move_ptp(0,150,200)    'under mode 0, same effect as "MOVE" command
wait idle
speed_ratio = 0.5
move_ptp(0,150,200)
```

MPOS(0), MSPEED(0), MPOS(1), MSPEED(1), vertical scale (Y scale): 200

	
Instructions	MOVE_PTPABS

MOVE_PTPABS – Point-to-Point | Absolute

Type	Special Motion Instruction
Description	**This is the linear interpolation motion command used for point-to-point motion.** This command doesn't support speed ahead in continuous motion, and it uses each axis' SPEED, then speed parameter will enter motion buffer automatically. This command doesn't support modifying online commands dynamically, but it can use SPEED_RATIO to adjust speed, and VP_MODE configuration is valid.
Grammar	MOVE_PTPABS (mode, dis1, dis2……) mode: BIT0: 1 -- speed and acceleration are calculated by each axis' speed and acceleration limits. BIT2: 1 – reserved dis1: absolute motion distance of first motion, unit: units, support decimal dis2: absolute motion distance of second motion, unit: units, support decimal
Controller	Valid in ZMC4XX series controllers, and the version above "version_buid 230510".
Example	**Example 1:** 'example of mode 1 single-axis and VP_MODE usage rapidstop(2) trigger speed_ratio = 1 base(0)

	mpos = 0 dpos = 0 atype = 1 speed = 100 units = 100 accel = 1000 decel = 1000 vp_mode = 0 move_ptp(1,200) wait idle vp_mode = 6 move_ptpabs(1,300) MPOS(0), MSPEED(0), vertical scale (Y scale): 200  **Example 2:** 'example of mode 1 single-axis and speed_ratio usage rapidstop(2) trigger speed_ratio = 1 vp_mode = 0,0 base(0,1) mpos = 0,0 dpos = 0,0 atype = 1,1 speed = 100,200 units = 500,500 accel = 500,500 decel = 500,500 move_ptpabs(1,150,200)

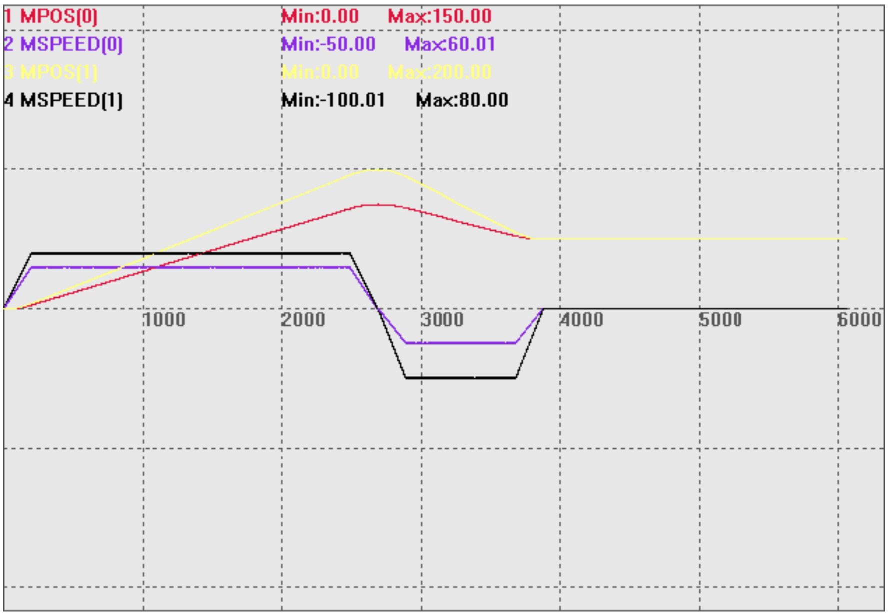
wait idle
speed_ratio = 0.5
move_ptp(1,100,100)

MPOS(0), MSPEED(0), MPOS(1), MSPEED(1), vertical scale (Y scale): 200

Example 3:

'example of mode 0 multi-axis and speed_ratio usage
rapidstop(2)
trigger
speed_ratio = 1
vp_mode = 0,0
base(0,1)
mpos = 0,0
dpos = 0,0
atype = 1,1
speed = 100,200
units = 500,500
accel = 500,500
decel = 500,500
move_ptp(0,150,200) 'under mode 0, same effect as "MOVEABS" command
wait idle
speed_ratio = 0.5
move_ptpabs(0,100,100)

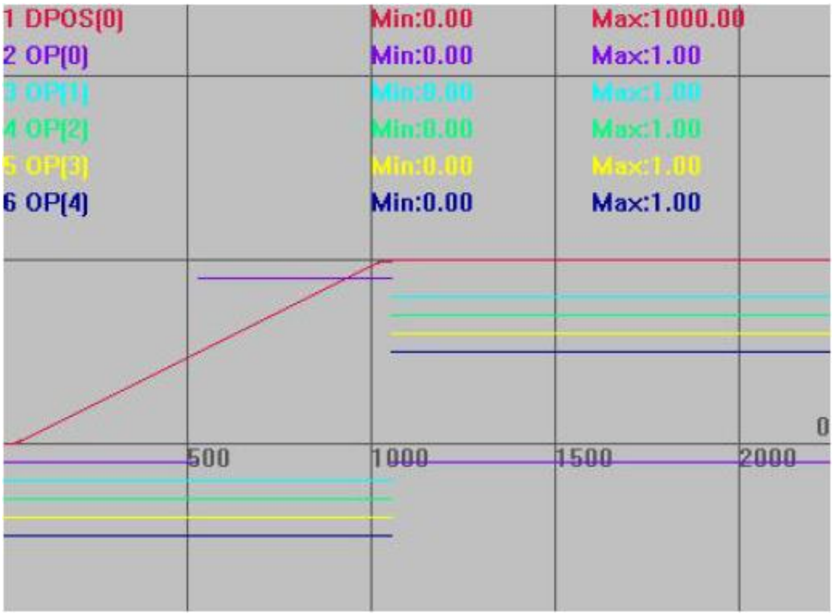
MPOS(0), MSPEED(0), MPOS(1), MSPEED(1), vertical scale (Y scale): 200

<table>
<tr><td></td><td>
</td></tr>
<tr><td>Instructions</td><td>MOVE_PTP</td></tr>
</table>

MOVE_OP--Output in Buffer

Type	Special Motion Instruction
Description	**Add one output to the motion buffer of BASE axis.** When LOAD is executed in the buffer, only operate the outputs, using the same grammar as **OP**. Normal Mode: error is one scan period, this mode is valid in all controllers. High Precision Output Mode: error is within 1 microsecond. ZMC4XX series or controller with firmware version above 20170421 supports. 1. Only valid in OP that supports hardware comparison output. 2. It is necessary to span one period between each effected precision output MOVE_OP, then it can take effect continuously, and in this gap, new MOVE_OP will use normal mode automatically. If exceeds the span, new MOVE_OP also can take effect, then it is continuous MOVE_OP, but only the first one is valid because of no span time (some controllers can trigger several precision outputs at the same time, see *Controller Hardware Manual* for details. For example, first 8 outputs of ZMC420SCAN support HPO, and every output uses HPO synchronously) 3. Even if the OP port is independent, when there are different OP ports of multi-axis, MOVE_OP also can be output highly precision. When HPO function is not independent, using HPO simultaneously will cause conflicts. 4. MOVE_OP precision function is on the basis of BASE master axis, when there is multiple axes interpolation, precision output of slave axis whose ATYPE type is different from BASE master axis can't be ensured. 5. Different precision parameters can be set through different MOVE_OP

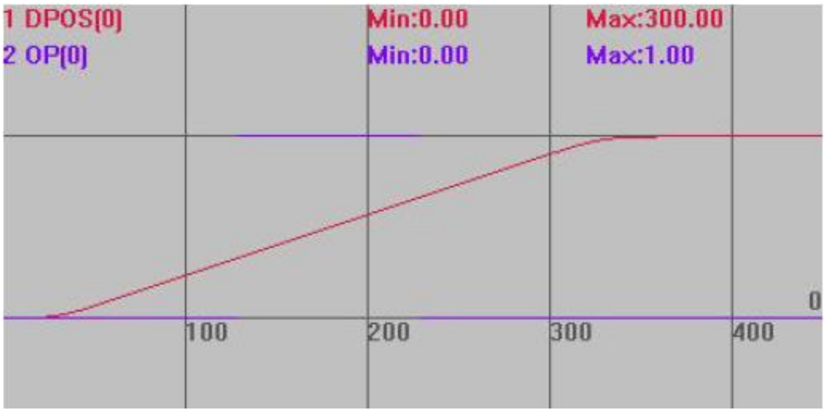
	instructions, then some parameters can be set well before calling MOVE_OP, such as, us level control for laser power, precision output, output delay, etc.
Grammar	**Grammar1: MOVE_OP ([ionum],value)** ionum: output No., which starts from 0 value: output status, indicating several ports' statuses by bits when multi outputs are operated. **Grammar2: MOVE_OP (ionum1, ionum2,value[,mask])** ionum1: the first output channel to operate ionum2: the last output channel to operate value: output status indicates status by bits when operating multi outputs. mask: set value according to bits status, and set which IOs to be operated, if it is blank, all channels (from the first to the last) are to be operated
Controller	General
Example	**Example 1: Normal Mode** BASE(0) UNITS=100 DPOS=0 SPEED=200 ACCEL=1000 DECEL=1000 TRIGGER 'Trigger the oscilloscope automatically MOVE(500) **MOVE_OP** (0,ON) 'wait until last instruction finished, OUT0 outputs signal MOVE(500) **MOVE_OP** (0,OFF) 'wait until last instruction finished, OUT0 closes signal **MOVE_OP**(1,4,15) 'OUT1-4 output signal, 15 is value of binary status:1111 Some offset in vertical direction was done in order to get better view of the trajectory curve. DPOS(0) vertical 1000 OP(0) vertical 1, offset -0.1 OP(1) vertical 1, offset -0.2 OP(2) vertical 1, offset -0.3 OP(3) vertical 1, offset -0.4 OP(4) vertical 1, offset -0.5



Example 2: High Precision Output Mode

BASE(0)
UNITS=100
DPOS=0
SPEED=200
ACCEL=1000
DECEL=1000
TRIGGER 'trigger oscilloscope automatically
ATYPE=1
MERGE=1
AXIS_ZSET(0) = 2 'open MOVE_OP precision output function
MOVE(100)
MOVE_OP(0,1) 'precision takes effect.
MOVE(100) 'since motion exceeds 2 ms, the next MOVE_OP will take effect in HPO mode.
MOVE_OP(0,0) 'precision takes effect .
MOVE(100)

Path Curve:
DPOS(0) vertical scale 300
OP(0) vertical scale 1



Example 3: Encoder High Precision Output Mode

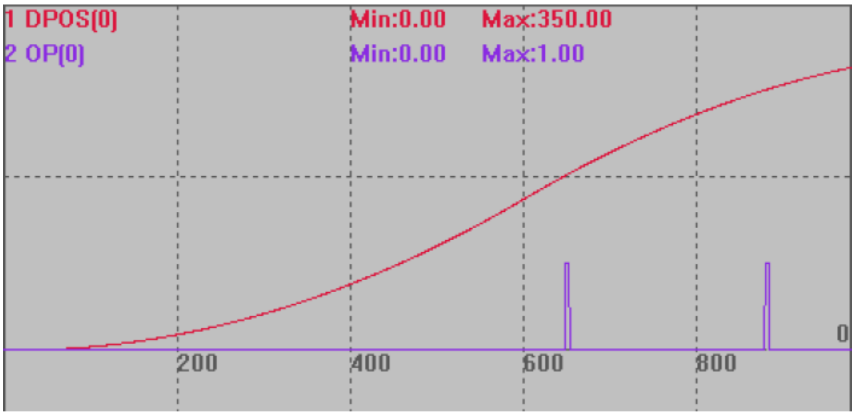
Valid in controller with firmware version above 20170505.

```
DIM opnum
AXIS_ZEST=3+16              'BIT4 supports precision output of encoder.
BASE(0)
ATYPE= 4    'set axis as pulse + encoder mode, encoder wiring is necessary.
DPOS=0
MPOS=0
units=1000
SPEED= 1000
ACCEL = 1000
MERGE=1
TRIGGER
opnum = 0
MOVEOP_DELAY =2          'actual output time delays 2 ms.
HW_TIMER(0,10000,5000,1,0,opnum)

OP(opnum,0)                         'initialize OP.
HW_TIMER(2, 10000, 5000, 1, 0, opnum)  'switch output to off after 5000 us.
MOVE(200)
MOVE_OP(opnum,1)         'use HW_Timer to close output, close after 5ms.
MOVE(100)
MOVE_OP(opnum,1)
MOVE(50)
END
```

Path Curve:

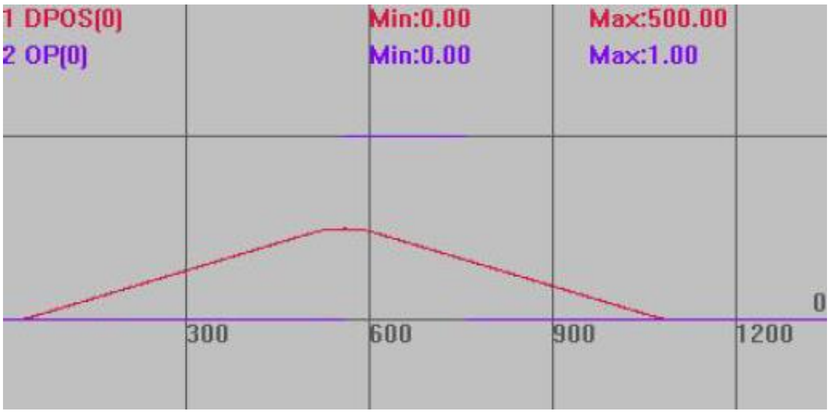
DPOS(0) vertical scale 200

OP(0) vertical scale 2

	
Instructions	OP, MOVE_OP2 . Precision Output Mode SYSTEM_ZSET, AXIS_ZSET

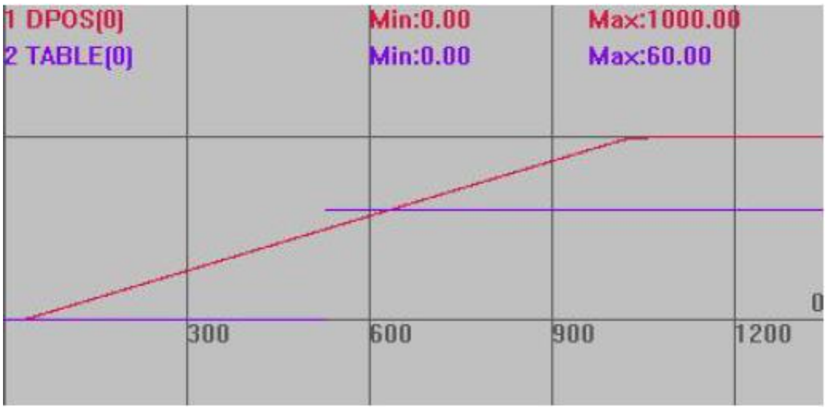
MOVE_OP2-Output2 in buffer

Type	Special Motion Instruction
Description	**Add output to the motion buffer, invert the output status after determined period.** When LOAD is executed in the buffer, only operate the outputs. One axis only supports one MOVE_OP2 instruction, if the second MOVE_OP2 is executed, then the former MOVE_OP2 will be closed automatically.
Grammar	MOVE_OP2(ionum,state,offtimems) ionum: output port NO., default value:0-31. state: output state. offtimems: the period after which signal inverts. (ms)
Controller	General
Example	BASE(0) UNITS=100 DPOS=0 SPEED=200 ACCEL=1000 DECEL=1000 OP(0,OFF) 'shut the OUT0 port. TRIGGER 'Trigger the oscilloscope automatically MOVE(500) **MOVE_OP2** (0,ON,1000) 'output 0 outputs 1s pulse after the former instruction is finished. This instruction will not obstruct the execution of next instruction. MOVE(-500) Motion Path: DPOS(0) vertical scale 1000

	OP(0) vertical scale 1 
Instructions	MOVE_OP, OP

MOVE_TABLE – Table in Buffer

Type	Special Motion Instruction
Description	**Add one TABLE to motion buffer based on BASE axis motion.** When LOAD is executed in the buffer, only modifies TABLE. MTYPE and MOVE_OP are the same.
Grammar	MOVE_TABLE(tablenum, value) tablinum: TABLE NO. value: the value to be modified.
Controller	General
Example	BASE(0) UNITS=100 DPOS=0 SPEED=200 ACCEL=1000 DECEL=1000 TABLE(0)=0 'assign Table(0) as 0 ?TABLE(0) 'print the value of Table(0), result:0 TRIGGER 'trigger the oscilloscope automatically MOVE(500) **MOVE_TABLE**(0, 60) 'after motion finished, assign Table(0) as 60. MOVE(500) WAIT IDLE ?TABLE(0) 'print the changed value of Table(0), result:60 Path Curve: DPOS(0) vertical scale 1000 TABLE(0) vertical scale 100

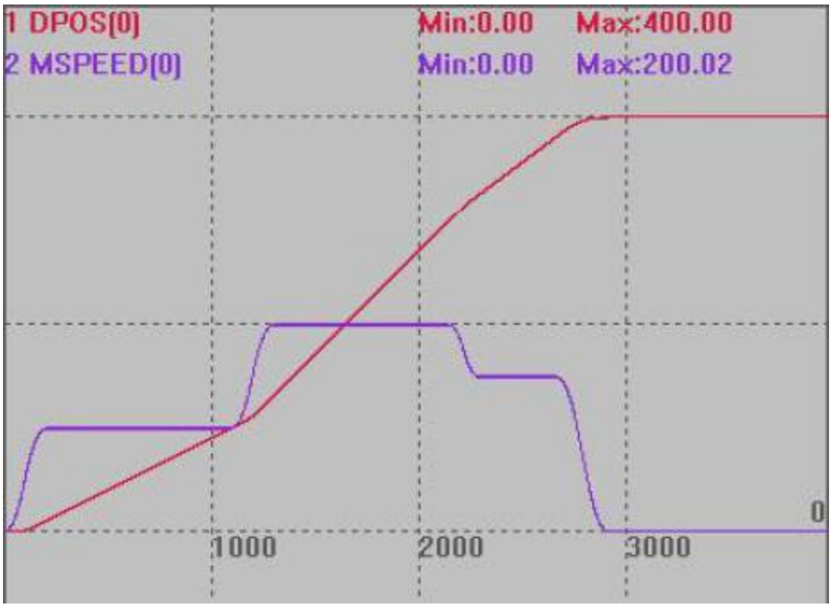
	
Instructions	TABLE

MOVE_PARA-Parameters in buffer

Type	Special Motion Instruction
Description	**Modify parameters in motion buffer based on BASE axis motion.** When LOAD is executed in the buffer, only modifies parameters. MTYPE value of this instruction is same as MTYPE value of MOVE_OP.
Grammar	MOVE_PARA(PARANAME,INDEX,VALUE) paraname: parameter's name, must be non-read only parameters in ?*set index: parameters NO. value: Parameters value
Controller	With firmware version above 20170503
EXAMPLE	**Example 1: Modify SPEED** BASE(0) 'select axis 0 ATYPE=1 SPEED=100 PRINT SPEED 'print result: 100 MOVE_PARA(speed,0,200) 'change SPEED value as 200 of axis 0 DELAY(1000) PRINT SPEED 'print result: 200 **Example 2: Single-axis speed changing** BASE(0) 'select axis 0 UNITS=1000 ATYPE=1 SPEED=100 ACCEL=1000 DECEL=1000 SRAMP=100 DPOS=0 MERGE=ON TRIGGER

```
MOVE(100)
MOVE_PARA(SPEED,0,200)
MOVE(200)
MOVE_PARA(SPEED,0,150)
MOVE(100)
END
```

Vertical scale 200

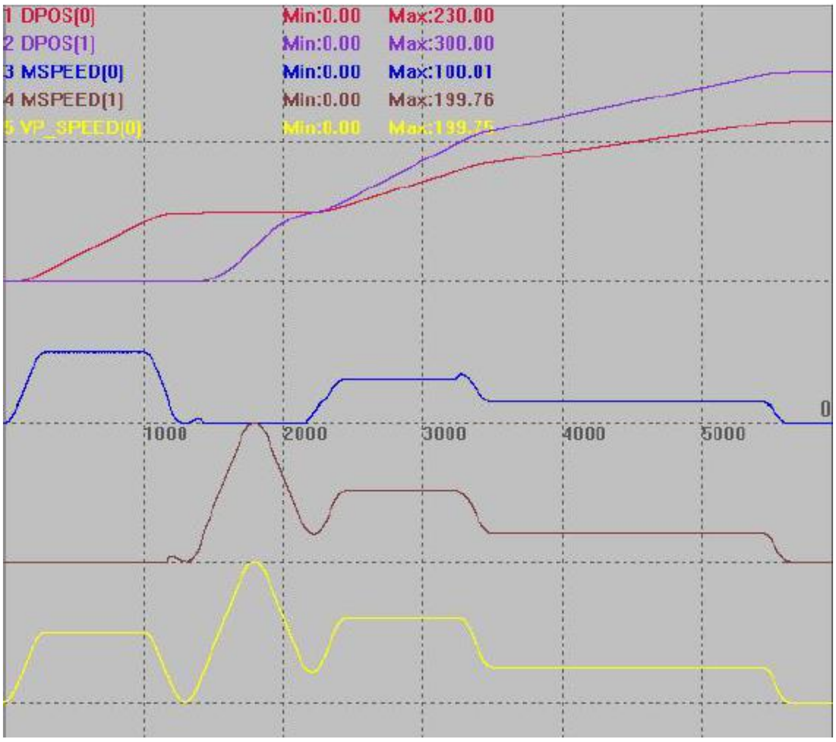


Example 3: Multi-axis speed changing

```
RAPIDSTOP(2)
WAIT IDLE(0)
WAIT IDLE(0)

BASE(0,1)
DPOS=0,0
UNITS=100,100
SPEED=100,100               'set speed
ACCEL=500,500               'set acceleration
DECEL=500,500               'set deceleration
SRAMP=100,100               'S curve

MERGE=ON                    'open continuous interpolation
CORNER_MODE=2+8+32          'set corner deceleration
DECEL_ANGLE = 15 * (PI/180) 'set the angle of start deceleration
STOP_ANGLE = 45 * (PI/180) 'set the angle of end deceleration
ZSMOOTH=2
FORCE_SPEED=100             'it works when proportional deceleration
TRIGGER                     'trigger oscilloscope automatically

MOVE(100,0)

MOVE_PARA(SPEED,0,200)
```

```
MOVE(0,100)                 'motion angle is over 45, total deceleration

MOVE_PARA(SPEED,0,120)
MOVE(60,100)                'motion angle is 30.96, which is within 15~45,
                             proportional deceleration

MOVE_PARA(SPEED,0,50)
MOVE(70,100)                'motion angle is 4.03, which is less 15, not to
                             decelerate
END
```

Vertical scale 200



MOVE_PWM-PWM in Buffer

Type	Special Motion Instruction
Description	**Operate PWM in motion buffer based on BASE axis motion.** When LOAD is executed, only operate the PWM. MTYPE value of this instruction is same as MTYPE value of MOVE_OP. When the duty ratio is 0, PWM can be closed, don’t set PWM frequency as 0 for closing it, PWM frequency should be modified before PWM switch setting.
Grammar	MOVE_PARA(PWMINDEX,duty[,freq]) pwmindex: PWM NO. duty: duty ratio, the ratio of valid signal electrical level to entire period, and the range is 0-1, when it is 0, close pwm. In one period, output valid electrical level first, invalid will be output next.

	freq: frequency, the default value is 1KHz, for hardware up to 1MHz, for software up to 2KHz.
Controller	With firmware version above 20170503

Routine	RAPIDSTOP(2)
	WAIT IDLE
	TRIGGER
	TICKS=0
	BASE(0)
	SPEED = 1000
	move(10)
	MOVE_PWM(0, 0.111, 2000) 'when axis0 reaches 10, PWM0 activates.
	MOVE_DELAY(111)
	MOVE_PWM(0, 0.333)
	MOVE_DELAY(111)
	MOVE_PWM(0, 0.555, 3000)
	move(100)
	WHILE NOT IDLE
	    MOVE_PWM(0,0,1000) 'close PWM
	    ? -TICKS, PWM_FREQ(0), PWM_DUTY(0)
	    WA 10
	WEND

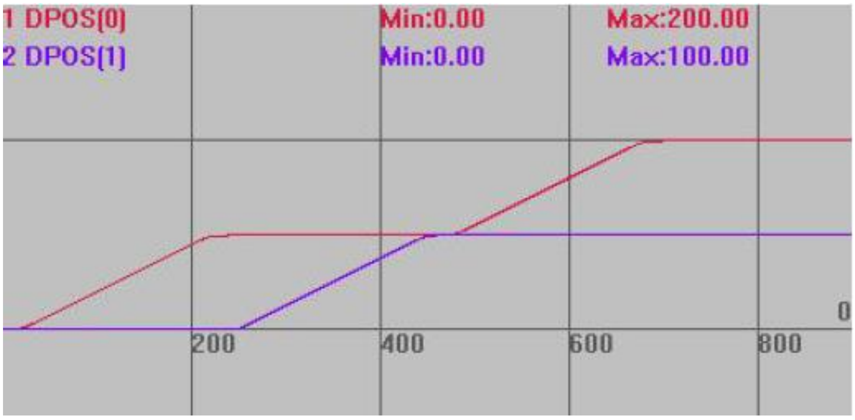
Instructions	PWM_DUTY, PWM_FREQ

MOVE_SYNMOVE-Synchronous Axis in Buffer

Type	Special Motion Instruction
Description	**Tigger other axis motions in motion buffer based on BASE, the present axis waits.** MTYPE value of this instruction is same as MYTPE value of MOVE_Delay.

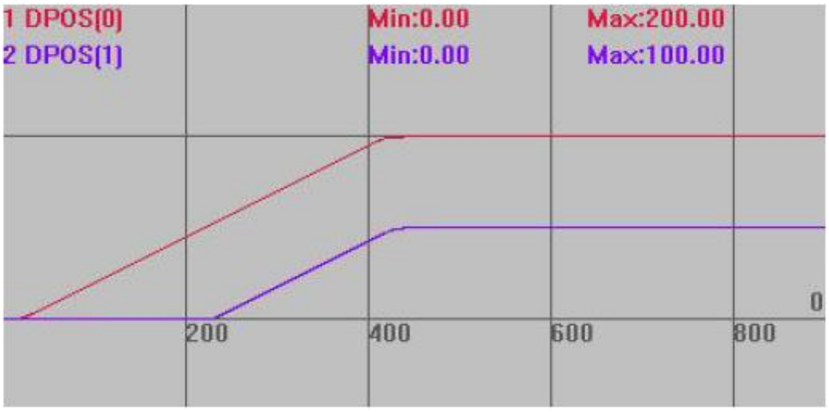
Grammar	MOVE_SYNMOVE(AXISNUM,DIS[,IFSP])
	    axisnum: the synchronous axis NO.
	    dis: relative motion distance
	    ifsp: use SP function or not, not to use by default.

Controller	With firmware version above 20170503
Routine	RAPIDSTOP(2) WAIT IDLE TRIGGER TICKS=0 BASE(0,1) DPOS=0,0 UNITS=100,100 SPEED=100,100 ACCEL=1000,1000 DECEL=1000,1000

	MOVE(100) **MOVE_SYNMOVE(1,100,0)** 'axis1 starts to move when axis0 reaches 100. MOVE(100) 'wait until axis synchronization finished, axis 0 continues to move Motion Path: DPOS(0) vertical scale 200 DPOS(1) vertical scale 200 
Instructions	MOVE_ASYNMOVE

MOVE_SYNMOVE-Synchronous Axis in Buffer 2

Type	Special Motion Instruction
Description	**Tigger other axis motions in motion buffer based on BASE, the present axis doesn't wait.** MTYPE value of this instruction is same as MYTPE value of MOVE_OP.
Grammar	MOVE_SYNMOVE(axisnum,dis[,ifsp]) axisnum: the synchronous axis NO. dis: relative motion distance ifsp: use SP function or not, not to use by default.
Controller	With firmware version above 20170503
Routine	RAPIDSTOP(2) WAIT IDLE TRIGGER TICKS=0 BASE(0,1) DPOS=0,0 UNITS=100,100 SPEED=100,100 ACCEL=1000,1000 DECEL=1000,1000 MOVE(100) **MOVE_SYNMOVE(1,100,0)** 'axis1 starts to move when axis0 reaches 100.

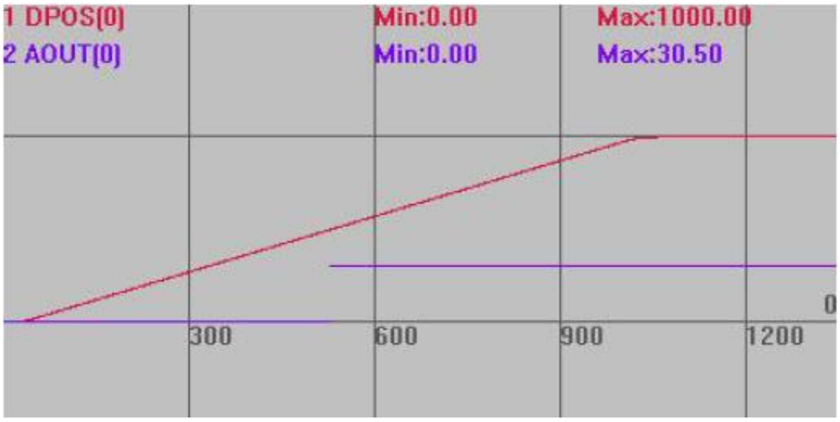
	MOVE(100) 'axis 0 continues to move Motion Path: DPOS(0) vertical scale 200 DPOS(1) vertical scale 200 
Instructions	MOVE_ASYNMOVE

MOVE_TASK-Start Task in Buffer

Type	Special Motion Instruction
Description	**Add TASK to motion buffer.** When LOAD is executed, only operate TASK. MTYPE value of this instruction is same as MTYPE value of MOVE_OP. When task is started, error will occur, but no influence on procedure execution.
Grammar	MOVE_TASK(tasknum, label) tasknum: task NO. label: function name or Label(:).
Controller	General
Example	BASE(0) DPOS=0 UNITS=100 ACCEL=1000 DECEL=1000 SPEED=100 ACCEL=1000 DECEL=1000 MOVE(100) **MOVE_TASK**(1,task_move) 'start task_move as task 1 after former motion is finished. MOVE(100) END TASK_MOVE:

	PRINT "TASK_MOVE" END
Instructions	RUNTASK, RUN

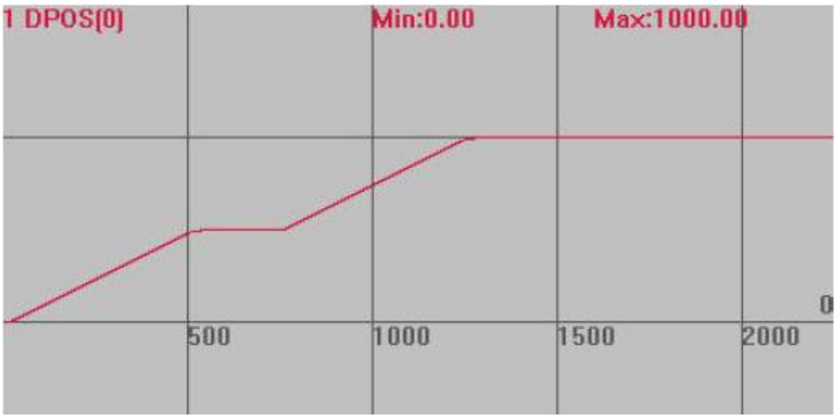
MOVE_AOUT-Analog Signal in Buffer

Type	Special Motion Instruction
Description	**Add one AOUT instruction into BASE axis motion buffer.** When LOAD is executed, only change value of AOUT, MTYPE value of this instruction is same as MYTPE value of MOVE_OP. When laser energy output is set, precision output is valid.
Grammar	MOVE_AOUT(danum, value) danum: DA NO. value: value to be modified.
Controller	General, controller with DA channels.
Example	BASE(0) UNITS=100 DPOS=0 SPEED=200 ACCEL=1000 DECEL=1000 AOUT(0)=0 'assign value to DA0. ?AOUT(0) 'print TRIGGER 'trigger the oscilloscope automatically MOVE(500) **MOVE_AOUT**(0, 30.5) 'assign 30.5 to DA0 after first motion is finished. MOVE(500) WAIT IDLE ?AOUT(0) 'print DA0,it is 30.5. Motion Path: DPOS(0) vertical scale 1000 AOUT(0) vertical scale 100 

Instructions	AOUT

MOVE_DELAY-Delay in buffer

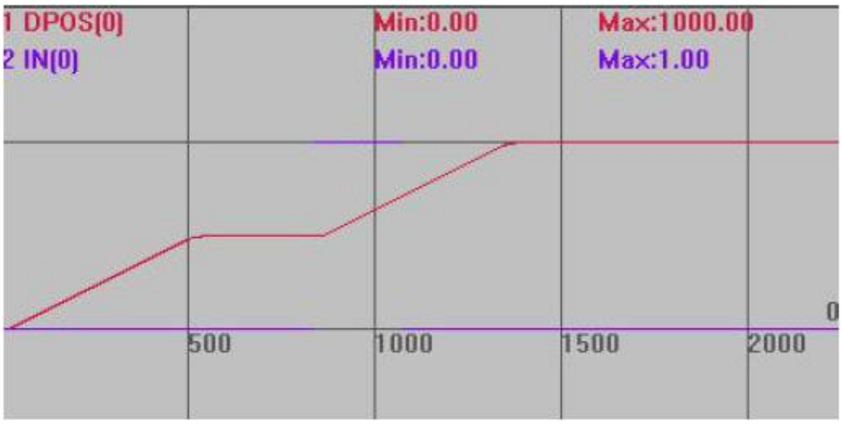
<table>
<tr><td>Type</td><td>Special Motion Instruction</td></tr>
<tr><td>Description</td><td>Add one delay to motion buffer of BASE axis.
When LOAD is executed, only assign the time value of delay.
Speed will decrease to 0 after the former instruction ends.</td></tr>
<tr><td>Grammar</td><td>MOVE_DELAY(timems)
Other Name：
move_wa(timems)
timems: delay time, ms as units.</td></tr>
<tr><td>Controller</td><td>General</td></tr>
<tr><td>Example</td><td>BASE(0)
UNITS=100
DPOS=0
SPEED=200
ACCEL=2000
DECEL=2000
TRIGGER 'trigger the oscilloscope automatically
MOVE(500)
MOVE_DELAY(1000) 'wait 1 second between 2 MOVE instructions.
MOVE(500)

Motion Path:
DPOS(0) vertical scale 1000


</td></tr>
<tr><td>Instructions</td><td>DELAY</td></tr>
</table>

MOVE_WAIT - Wait in Buffer

Type	Special Motion Instruction
Description	**Add one condition judge to BASE axis motion buffer.** When LAOD is executed, don’t do any motions. It only waits the specified

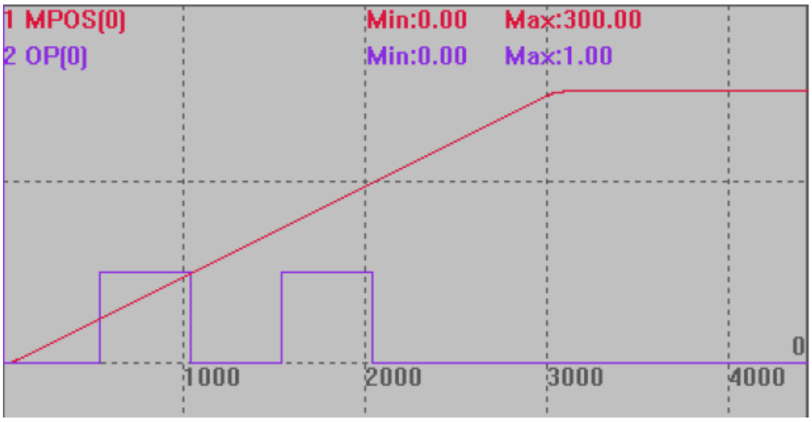
	condition to be met. And the speed will decrease to 0 automatically when former motion commands end.
Grammar	MOVE_WAIT(paraname, paranum, eq, value) paraname: choose parameter's name (it can be DPOS, MPOS, IN, AIN, VSPEED, MSPEED, MODBUS_REG, MODBUS_IEEE, MODBUS_BIT, TABLE, VECTOR_BUFFERED, REMAIN.) paranum: parameter No. / axis No. eq: 1 ≥ -1 ≤ Invalid for IN port or other BIT based parameters. 0 Not recommend value: comparison value.
Controller	With firmware version above 150802, or above XPLC160405
Example	BASE(0) DPOS=0 ATYPE=1 UNITS=100 SPEED=200 ACCEL=2000 DECEL=2000 TRIGGER 'trigger the oscilloscope automatically MOVE(500) **MOVE_WAIT**(IN, 0, 0, 1) 'execute the next motion buffer until IN(0) appear signal MOVE(500) Motion Path: DPOS(0) vertical scale 1000 IN(0) vertical scale 1 
Instructions	WAIT_UNTIL

MOVE_CANCEL—Stop Buffer

Type	Special Motion Instruction
Description	**Add CANCEL in motion buffer**
Grammar	MOVE_CANCEL(iaxis, imode) iaxis: axis to operate imode: select CANCEL mode, same as CANCEL instruction
Controller	General
Example	MOVE_CANCEL(1,0) 'the instruction of stop axis1 is written in axis0 buffer
Instruction	CANCEL

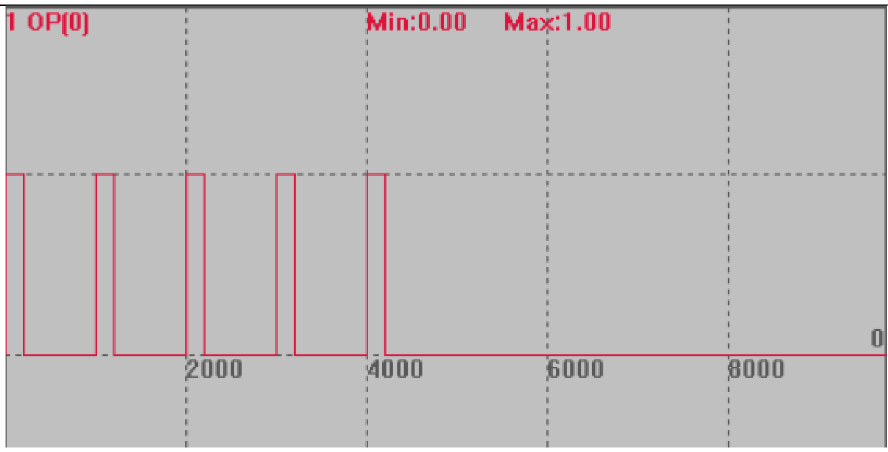
MOVE_HWPSWITCH2 — Buffer hardware comparison output

Type	Special Motion Instruction
Description	**Enter HW_PSWITCH2 command into buffer, hardware comparison output will be executed in motion buffer.** Same as HW_PSWITCH2, HW_TIMER parameter can be modified dynamically, but controllers must support HW function.
Grammar	MOVE_HWPSWITCH2(mode,[…]) mode: open different comparers, parameters to be filled also different. Refer to HW_PSWITCH2 command.
Controller	General
Example	BASE(0) UNITS=100 SPEED=100 ACCEL=1000 DECEL=1000 DPOS=0 MPOS=0 OP(0,OFF) TABLE(0,50,100,150,200) 'set compare points' coordinates MOVE_HWPSWITCH2(2) 'stop and delete uncompleted compare points MOVE_HWPSWITCH2(1,0,1,0,3,1) 'compare 4 points, mode 1, operate OUT0 TRIGGER 'trigger oscilloscope MOVE(300) END Compare output: Same output effect as HW_PSWITCH2, but this example enters 3 commands into buffer.

	MPOS(200) vertical scale OP(0) vertical scale 
Instruction	HW_PSWITCH2

MOVE_HWTIMER – Buffer Hardware Timer

Type	Special Motion Instruction
Description	**Enter HW_TIMER command into motion buffer, and execute hardware timer in motion buffer.** Parameters are the same as HW_TIMER, and HW_TIMER parameters can be changed dynamically, but controllers must support HW function. Must use MOVE_OP to trigger HW_TIMER, OP can’t trigger. When HW_TIMER is not used, calling mode 0 is off, otherwise, following output will be effected.
Grammar	MOVE_HWTIMER (mode, […]) mode: mode 0 / 1 / 2 / 3 / 4, the parameters that need to be filled in are also different, see HW_TIMER syntax description and routine.
Controller	General
Example	BASE(0) ATYPE=1 UNITS=100 SPEED=100 ACCEL=500 DPOS=0 TRIGGER MOVE_OP(0,OFF) MOVE_HWTIMER(2, 1000000, 200000, 5, OFF, 0) ‘when output 0 becomes ON, hardware timer triggers to do timing, turn to off after 500ms, the period is 5 times. MOVE_OP(0,ON) END

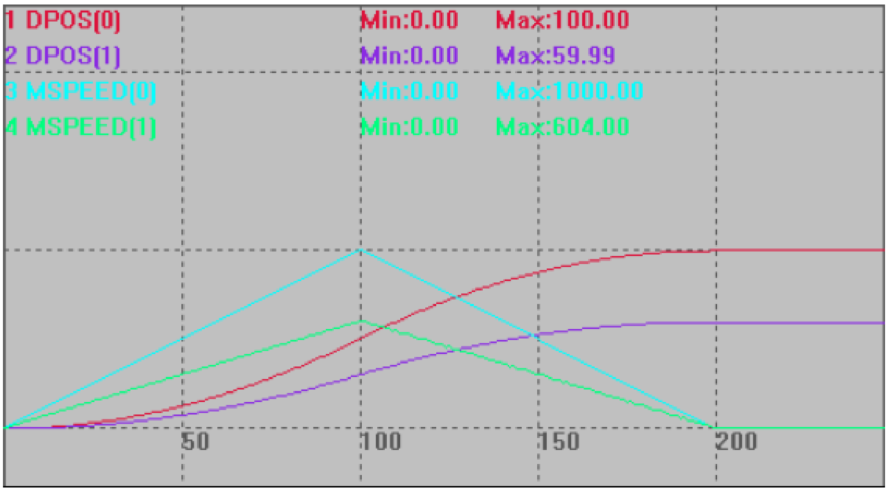


Instruction	HW_TIMER, HW_PSWTICH2

MOVE_ADDAX – Motion Superposition

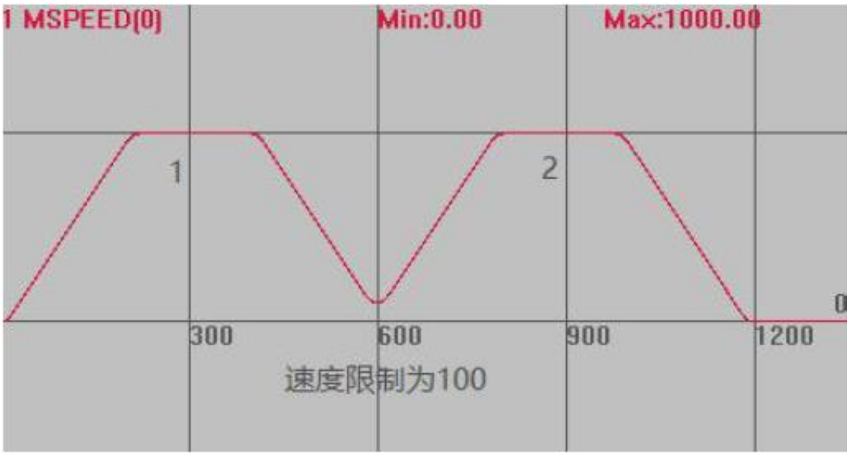
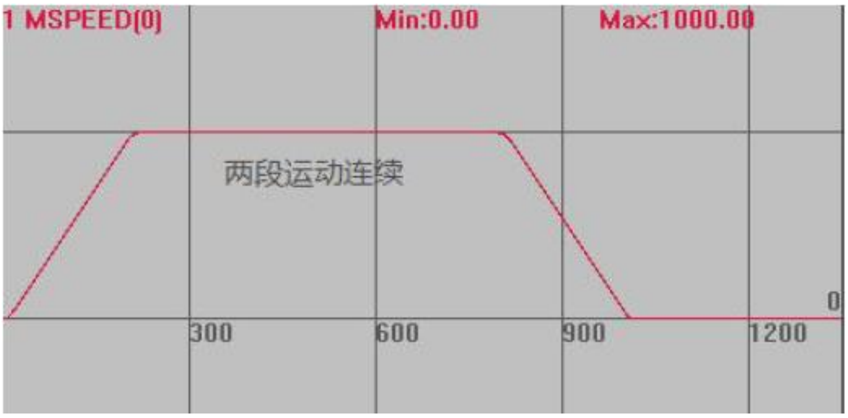
Type	Single Axis Motion Intructions
Description	**Motion superposition adds to buffer, superimposes the motion of one axis to another axis, supports BASE_MOVE, so that destaxis can be adjusted at will.** **The ADDAX command superimposes the number of pulses, not the set units.** Conversion relationship: superimposing axis movement distance * superimposing axis UNITS/superimposed axis UNITS=superimposed axis movement distance. Suppose the UNITS of axis A is 100, the UNITS of axis B is 50, and the superimposing axis movement is 100 Superimpose the movement of axis A to axis B. At this time, axis A shows a movement of 100, and axis B moves 100*100/50=200. The movement of axis B is superimposed on axis A. At this time, axis B shows a movement of 100, and axis A moves 100*50/100=50. The axes cannot be superimposed on each other at the same time. After A is superimposed on B, B can no longer be superimposed on A. Support series superposition, A superimposed to B, then B superimposed to C. Support parallel superposition, A motion is superimposed on B and C at the same time. When superimposing, the speed changes from the superimposed axis, and the acceleration and deceleration are determined according to the acceleration and deceleration of the superimposing axis and the ratio of the units of the two axes. ADDAX has no effect when the axis MTYPE is FRAME or REFRAME.

Grammar	ZMC4XX series and above controllers with 20220728 firmware adds superposition. MOVE_ADDAX(srcaxis ,[imode], [para]) destaxis: the superposed target axis number srcaxis: the superposed axis number of the source axis imode: superposition mode 0: default value, single-axis superposition, compatible with previous direct pulse number superposition 1: single-axis superposition, support scale adjustment. MOVE_ADDAX(srcaxis, 1, ratio) ratio: ratio value, supports floating point numbers, target axis distance = source axis distance * ratio. 2: single-axis superimposition, supports gear ratio adjustment MOVE_ADDAX(srcaxis, 2, ratioin, ratioout) ratioin: numerator, integer, supports negative numbers ratioout: denominator, positive integer. target axis distance = source axis distance * ratioin / ratioin 3: single axis superimposed to two axes, support angle adjustment BASE(destaxis1, destaxis2) MOVE_ADDAX(srcaxis, 3, angle) destaxis: the superposed target axis 1, 2 angle: angle, radian value, target axis 1 distance = source axis distance * cos(angle). target axis2 distance = source axis distance * sin(angle). Note: If needs to cancel, cancel the two axes MOVE_ADDAX(-1, 3, 0) or MOVE_ADDAX(-1) AXIS (the superposed axis No.) respectively 4: SCAN linkage superposition, use SCAN axis to compensate the deviation of platform axis, and their directions and amounts must be consistent, if not, please adjust gear ratio or add ratio for SCAN correction. BASE(destaxis, destaxis2) ADDAX(srcaxis, 4, srcaxis2) Use srcaxis to compensate destaxis, use srcaxis2 to compensate destaxis2. Note: two axes should be cancelled together, ADDAX(-1, 4, -1) or ADDAX(-1) AXIS (superposed axis No.) 5: SCAN linkage superposition, platform axis is superposed at SCAN axis, their directions and amounts must be consistent, if not, please adjust gear ratio or add ratio for SCAN correction. BASE(destaxis, destaxis2)

	ADDAX(srcaxis, 5, srcaxis2) srcaxis is superposed at destaxis, srcaxis2 is superposed at destaxis2. Note: two axes should be cancelled together, ADDAX(-1, 5, -1) or ADDAX(-1) AXIS (superposed axis No.) Add BASE_MOVE support: BASE_MOVE=moveaxis BASE(destaxis) MOVE_ADDAX(srcaxis ,[imode], [para]) BASE_MOVE=-1
Controller	General
Example	**Example 1: For more mode description, see the ADDAX command** BASE(0,1) 'select axis number UNITS = 100,100 DPOS=0,0 TRIGGER BASE(1) 'select the axis to be superimposed ADDAX(0,2,3,5)AXIS(1) 'mode 2 superpose, axis 0 superposes on axis 1 MOVE(100) AXIS(0) WAIT UNTIL IDLE(0) AND IDLE(1) ?"axis 1 superposing axis number" ADDAX_AXIS(1) ADDAX(-1) AXIS(1) 'cancel superposition END The pulse equivalent is the same, and the movement distance of axis 1 is 3/5 times that of axis 0. 
Instruction	[ADDAX](#)

MOVELIMIT – Speed Limit

Type	Special Motion Instruction

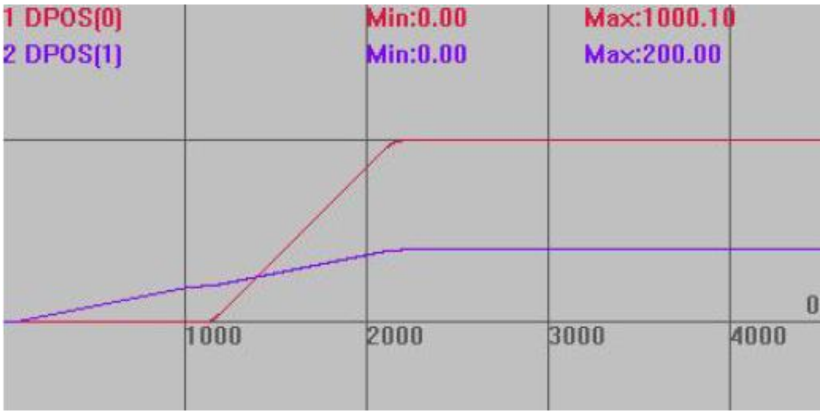
Description	**Add speed limit to the end of present motion, in order to force the axis to decelerate at the turning corner.**
Grammar	MOVELIMIT(limitspeed) limitspeed: the limit speed value.
Controller	General
Example	BASE(0) UNITS=100 DPOS=0 SPEED=1000 'axis speed ACCEL=1000 'axis acceleration. DECEL=1000 SRAMP=100 MERGE=ON ‘open continuous interpolation mode, the speed will be continuous between multi movements. TRIGGER ‘trigger the oscilloscope automatically MOVE(2000) **MOVELIMIT**(100) ‘speed between two motions will decrease to 100. MOVE(2000) Interpolation Speed (with MOVELIMIT) MSPEED(0) vertical scale 1000  Interpolation Speed (without MOVELIMIT): MSPEED(0) vertical scale 1000 

Instructions	CORNER_MODE

7.4 Synchronization Motion Instruction

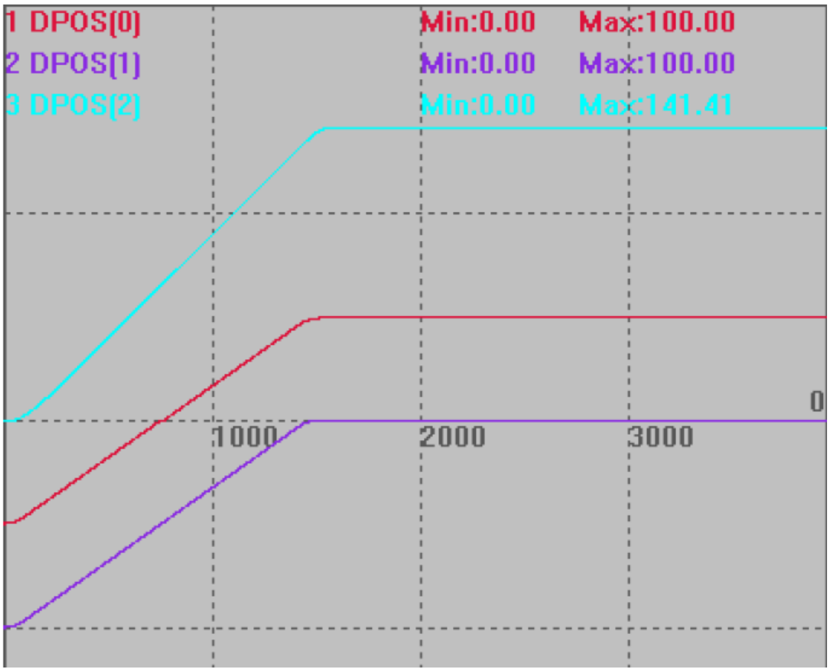
CONNECT-Synchronization Motion

Type	Synchronization Motion Instruction
Description	**Target position of present axis and measured position of driving_axis will be linked by electronic gearbox.** The link relationship is calculated in pulse amount, so do take UNITS of axes into consideration. Use CANCEL to cancel connection. Suppose UNITS of link axis0 is 100, UNITS of link axis1 is 10. in this situation, when using CONNECT to link these two axes, ratio=1, if axis0 moves s1=100, then axis1 will move 1000, s1*UNITS(0)*ratio/UNITS(1). Ratio can be changed dynamically by calling instruction repeatedly. Usually used to link Handwheel.
Grammar	CONNECT（ratio, driving_axis） ratio: the ratio can be negative or positive, it is ratio of pulse amount. driving_axis: axis NO. of link axis, it is encoder axis when handwheel
Controller	General
Example	RAPIDSTOP(2) WAIT IDLE(0) WAIT IDLE(1) BASE(0,1) ATYPE=1,1 UNITS=10,100 DPOS=0,0 SPEED=100,100 ACCEL=1000,1000 DECEL=1000,1000 TRIGGER 'Trigger the oscilloscope automatically MOVE(100) AXIS(1) 'axis1 moves 100, axis 0 moves 0. WAIT IDLE(1) **CONNECT**(1,1) AXIS(0) 'axis0 is linked to axis1, ratio is 1. MOVE(100) AXIS(1) 'axis1 moves 100, then axis0 moves 1000. Motion Path DPOS(0) vertical scale 1000 DPOS(1) vertical scale 500

	
Instructions	CLUTCH_RATE，CONNPATH

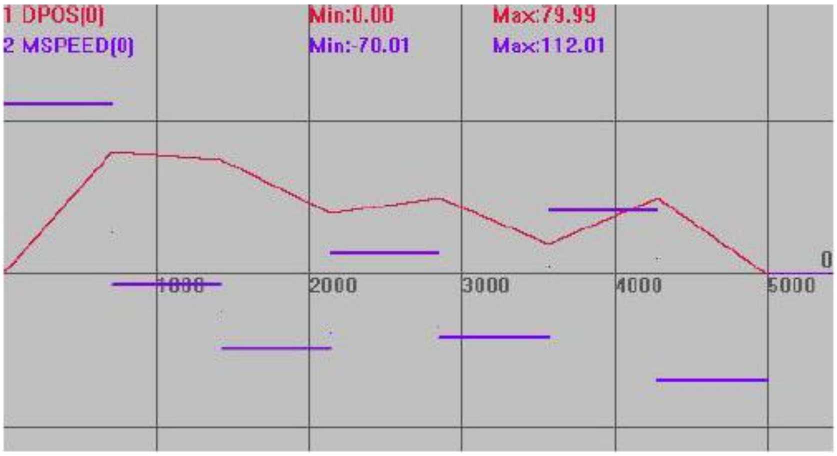
CONNPATH-Synchronization Motion 2

Type	Synchronization Motion Instruction
Description	**Target position of present axis and interpolation vector length of driving_axis will be linked by electronic gear.** It needs to be connected to the master axis of the interpolation motion to establish a connection with the length of the interpolation vector, and the effect of connecting to the slave axis is the same as CONNECT. The link relationship is calculated in pulse amount, so do take UNITS of axes into consideration. Use CANCEL to cancel connection. Ratio can be changed dynamically by calling instruction repeatedly.
Grammar	CONNECT（ratio, driving_axis） ratio: the ratio can be negative or positive, it is ratio of pulse amount. driving_axis: axis NO. of link axis, it is encoder axis when handwheel
Controller	General
Example	RAPIDSTOP(2) WAIT IDLE(0) WAIT IDLE(1) BASE(0,1,2) ATYPE=1,1,1 UNITS=100,100,100 SPEED=100,100,100 ACCEL=1000,1000,1000 DECEL=1000,1000,1000 TRIGGER 'Trigger the oscilloscope automatically CONNPATH(1,0) AXIS(2) 'axis 2 is linked to axis 0 (master axis of interpolation), ratio is 1. MOVE(100,100) 'interpolation motion. Motion Path DPOS(0) vertical scale 100, offset -50

	DPOS(1) vertical scale 100, offset -50 DPOS(2) vertical scale 100, no offset 
Instructions	CLUTCH_RATE, CONNECT

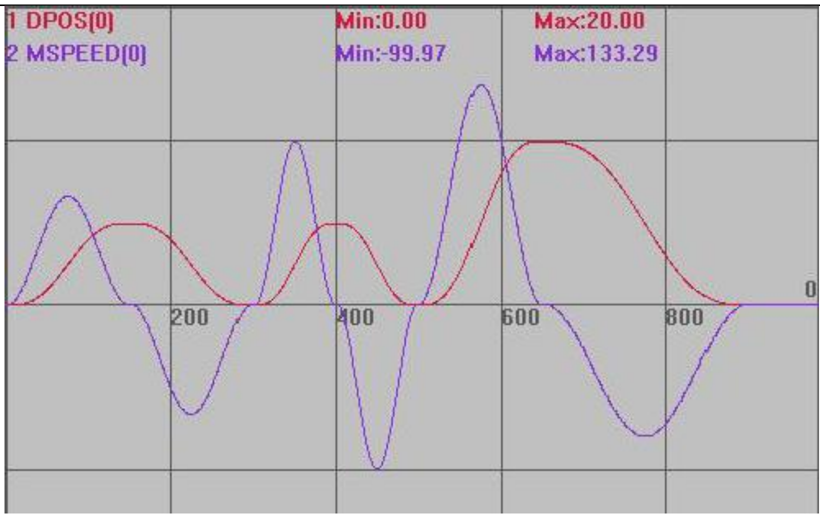
CAM-Cam Based Motion

Type	Synchronization Motion Instruction
Description	**CAM will determine motion of axis according to data saved in TABLE, data in TABLE is related to position where the motion should reach, it is absolute position relative to start position.** Note: two or more CAM instructions can use data in the same TABLE to generate its path. Total motion time is determined by set speed and the fourth parameter, actual speed of motion is determined automatically by motion path based on TABLE data and total motion time. Data in TABLE should be filled by manual, first data is guide point, 0 is recommended to be as this guide point. Table data*table multiplier=pulse amount to deliver. Ensure the parameter (distance) delivered by instruction is integer value of pulse, or it will emerge floats, then motion has minor errors.
Grammar	CAM(start point, end point, table multiplier, distance) start point: TABLE No. of start point, save position of first point end point: TABLE NO. of end point. table multiplier: position multiply this value, usually this value is set the same as UNITS. distance: refer to motion distance.

	total motion time=distance/Axis SPEED.
Controller	General
Example	**EXAMPLE 1:** RAPIDSTOP(2) WAIT IDLE(0) WAIT IDLE(1) BASE(0) UNITS=100 DPOS=0 ACCEL=1000 DECEL=1000 SPEED=100 TABLE(10,0,80,75,40,50,20,50,0) 'table starts to save data from 10. TRIGGER 'trigger the oscilloscope automatically **CAM**(10,17,100,500) 'motion path is from table(10) to table(17), total motion time is 500/100=5. Path and Speed: DPOS(0) vertical scale 100 MSPEED(0) vertical scale 100  **EXAPMLE 2: application of CAM in high speed, high precision motion.** DIM num_p,scale,m,t 'defined variables num_p=100 scale=500 FOR p=0 TO num_p TABLE(p,((-SIN(PI*2*p/num_p)/9PI*2))+p/num_p)*scale 'table save cam motion parameters NEXT RAPIDSTOP(2) WAIT IDLE(0) WAIT IDLE(1) BASE(0) 'select axis 0 DEFPOS(0) SERVO=ON

```
UNITS=500
SPEED=1000
ACCEL=1000000
DECEL=1000000
TRIGGER

m=10                          'it means the multiple of distance
t=0.3                         'operation time
SPEED=1000
CAM(0,100,m,SPPEED*t)
WAIT IDLE

m=10
t=0.3
SPEED=1000
CAM(0,100,-m,SPEED*t)
WAIT IDLE

m=10
t=0.2
SPEED=500
CAM(0,100,m,SPEED*t)
WAIT IDLE

m=10
t=0.2
SPEED=500
CAM(0,100,-m,SPEED*t)
WAIT IDLE

m=20
t=0.3
SPEED=1000
CAM(0,100,m,SPEED*t)
WAIT IDLE

m=20
t=0.5
SPEED=500
CAM(0,100,-m,SPEED*t)
WAIT IDLE
```

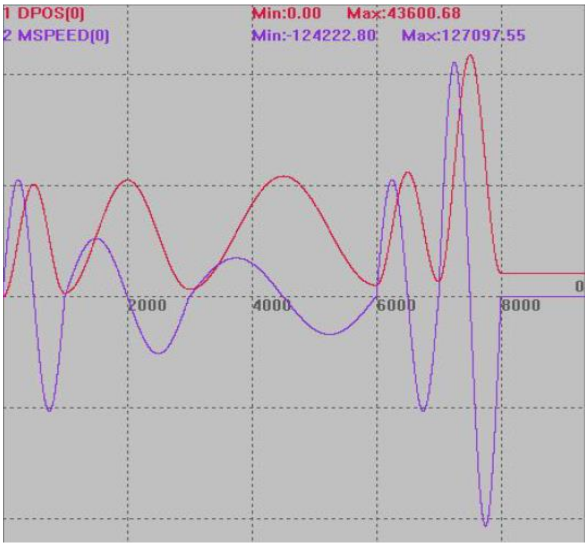
Interpolation Path:
DPOS(0) vertical scale 20
MSPEED(0) vertical scale 100



Example 3: Continuous Cam

```
RAPIDSTOP(2)
WAIT IDLE(0)
BASE(0)
UNITS=100
DPOS=0
ACCEL=1000
DECEL=1000
SPEED=100

DIM rad,x,deg
FOR deg= 0 TO 360 STEP 1      'build 0-360º cam data
        rad = deg * 2 * PI / 360
        x = deg * 2 + 10000 * ( 1 - COS (rad))
        TABLE (deg ,x)
        print deg,x
NEXT deg
TRIGGER                       'trigger oscilloscope automatically

CAM(0,360,100,100)            'motion path is from table 0 to 360, motion total
                               time is 100/100=1s
CAM(0,360,100,200)            'motion path is from table 0 to 360, motion total
                               time is 200/100=2s
CAM(0,360,100,300)            'motion path is from table 0 to 360, motion total
                               time is 300/100=3s
WAIT UNTIL REMAIN_BUFFER(1) > 0
              'save motion instruction until the buffer space is blank.
CAM(0,360,100,100)            'motion path is from table 0 to 360, motion
                               distance is 100*table data/units(0)
CAM(0,360,200,100)            'motion path is from table 0 to 360, motion
                               distance is 200*table data/units(0)
```

DPOS(0) vertical scale 20000
MSPEED(0) vertical scale 60000

	
Instructions	CAMBOX

CAMBOX- Following Motion of CAMBOX

<table>
<tr><td>Type</td><td>Synchronization Motion Instruction</td></tr>
<tr><td>Description</td><td>CAMBOX will determine motion of axis according to data saved in TABLE, data in TABLE is related to position where the motion should reach, it is position relative to start position. Motion of slave axis is determined by reference axis.
Note: two or more CAMBOX instructions can use data in the same TABLE to generate its path.

Total motion time is determined by motion distance and axis speed of reference axis, and the speed is matched automatically.
Data in TABLE should be filled by manual, first data is guide point, 0 is recommended to be as this guide point.
Table data*table multiplier=Pulse amount to deliver.
Ensure the parameter(distance) delivered by instruction is integer value of pulse, or it will emerge floats, then motion has minor errors.</td></tr>
<tr><td>Grammar</td><td>CAMBOX(start_point, end_point, table_multiplier, link_distance, link_axis[, link_options][, link_pos][, link_offpos])
start point: TABLE No. of start point, save position of first point.
end point: TABLE NO. of end point.
table multiplier: position multiply this value, usually this value is set the same as UNITS
link_distance: motion distance of reference axis.
link_axis: axis number of reference axis.
link_options: link mode with reference axis, different binary bits stand for different meaning.
<table>
<tr><td>Bit</td><td>Meaning</td></tr>
<tr><td>bit 0</td><td>Present axis starts to link with reference axis when MARK</td></tr>
</table>
</td></tr>
</table>

	signal of reference axis is triggered.
bit 1	Present axis starts to link with reference axis when reference axis reaches set absolute position.
bit 2	Repeat continuous double-direction motion automatically. (cancel the repeat by setting REP_OPTION=1.)
bit 4	Start from middle position, then use power failure interruption to realize CAMBOX recovering.
bit 5	Present axis links with reference axis when reference axis is moving in positive direction.
bit 8	Present axis starts to link with reference axis when MARKB signal of reference axis is triggered, and latch axis is the reference axis. Need latest firmware to support.

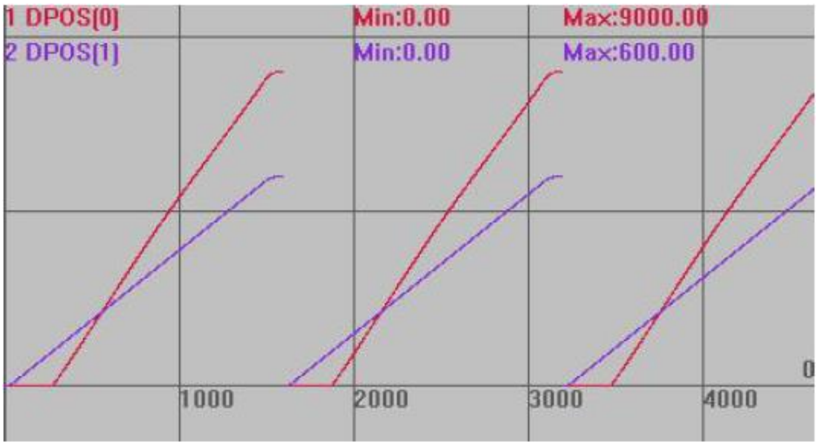
link_pos: when link_options is set as 2, then it means the absolute position where link between reference axis and slave axis starts.

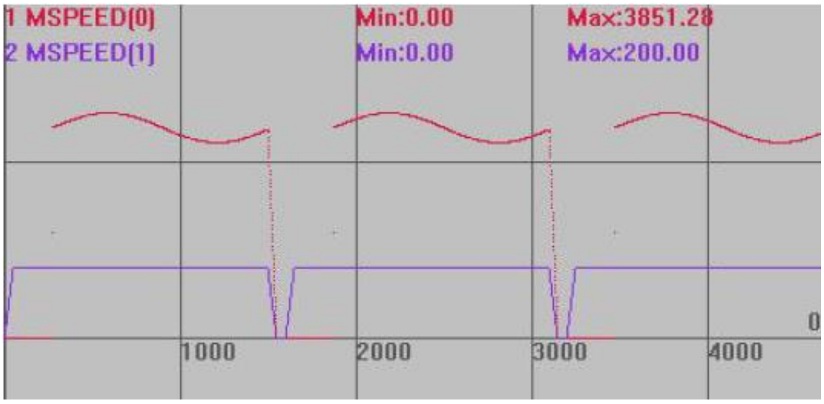
Link offpos: when Bit4 of link_options is 1, it means the relative distance that main axis has finished.

Controller	General

Example

```
ERRSWITCH = 3
RAPIDSTOP(2)
WAIT IDLE(0)
WAIT IDLE(1)
BASE(0,1)                       'select axis 0
ATYPE=1,1                       'pulse based stepper or servo
DPOS = 0,0
UNITS = 100,100                 'pulse equivalent
SPEED = 200,200
ACCEL= 2000,2000
DECEL= 2000,2000
'calculate data in TABLE
DIM deg, rad, x, stepdeg
stepdeg = 2                'change sections of data to generate, the more
                           sections it generates, the smoother speed will be.
FOR deg=0 TO 360 STEP stepdeg
    rad = deg * 2 * PI/360      'transfer to rad.
    x = deg * 25 + 10000 * (1-COS(rad))/100
    TABLE(deg/stepdeg,x)    'save into TABLE
    trace deg/stepdeg,x
NEXT deg
TRIGGER                        'trigger the oscilloscope automatically
WHILE 1                        'cycle motion

    IF IN (0) = on then        'trigger when in(0) is on.
        DPOS= 0,0
        CAMbox(0,360/stepdeg, 100, 500, 1,2,100)
                               'start to link when axis1 reaches 100.
        MOVE(600) AXIS(1)
```

<table>
<tr><td></td><td>
<pre>
        WAIT UNTIL IDLE AND IDLE   'wait until motion finishes.
        DELAY(100)   'time delay
    ENDIF
WEND
END   'stop present task.
</pre>
Motion Path:
DPOS(0) vertical scale 5000
DPOS(1) vertical scale 500


Speed Curve:
MSPEED(0) vertical scale 3000
MSPEED(1) vertical scale 500


</td></tr>
<tr><td>Instructions</td><td>CAM</td></tr>
</table>

MOVELINK-Auto Cam

Type	Synchronization Motion Instruction
Description	**Self-defined cam motion with adjustable acceleration and deceleration stages.** The connection axis is slave axis, the axis to be linked is reference axis. Distance of slave axis is divided into 3 parts to match motion of reference

axis, they are acceleration, uniform and deceleration.

During the acceleration or deceleration stage, in order to match the speed, link distance (distance of reference axis) must be two times of distance (distance of slave axis).

Ensure the parameter(distance) delivered by instruction is integer value of pulse, or it will emerge floats, then motion has minor errors.

Grammar

MOVELINK (distance, link dist, link acc, link dec, link axis[, link options] [, link pos][, link offpos])

distance: distance of slave axis during the link, this parameter can be negative or positive. Units as unit. When it is positive, it will move in forward direction. When it is minus, it will move in inverse direction.

link dist: absolute distance of reference axis during the link, units as unit.

link acc: absolute distance of reference axis during acceleration stage of salve axis, units as unit.

link dec: absolute distance of reference axis during deceleration stage of salve axis, units as unit.

Note: if sum of link dec and link acc is bigger than link dist, then they will be minished as per the scale until the sum is equal to link dist.

link axis: axis number of reference axis.

link options: link mode, different binary bits stand for different meaning.

Mode	Bit	Description
1	Bit 0	link starts when MARK signal of reference axis is triggered.
2	Bit 1	link starts when reference axis reaches a determined absolute position. (see link pos parameter description)
4	Bit 2	MOVELINK will execute repeatedly, and it can be inversed. (this mode can be cancelled by setting the bit1 of REP_OPTION as 1.)
8	Bit 3	curve acceleration or deceleration S mode, firmware version above 20170502 supports.
16	Bit 4	start from a position in the middle, then use power failure interruption to realize link recovering.
32	Bit 5	link happens only when reference axis is moving in positive direction
256	Bit 8	link starts when MARKB signal of reference axis is triggered, need latest firmware version to support.

link pos: when link options is set as 2, which means absolute position of reference axis where link starts.

link offpos: when bit4 of link_options is set as 1, which means the relative distance that master axis has finished, firmware with version above 20170428 supports.

Controller

General

Example

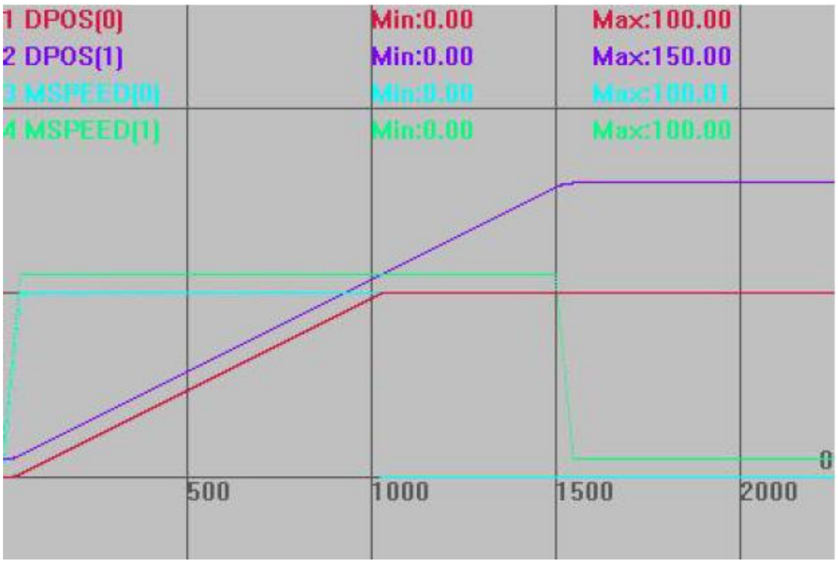
Example1:

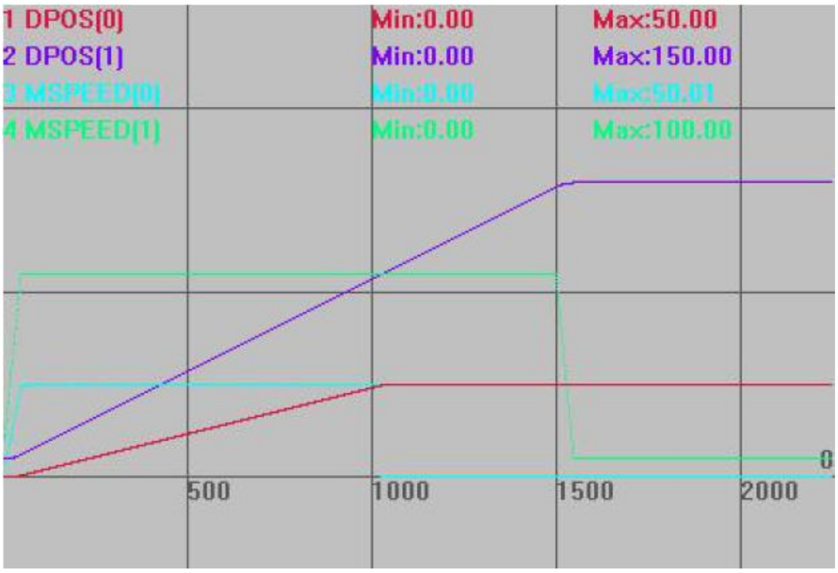
RAPIDSTOP(2)

WAIT IDLE(0)

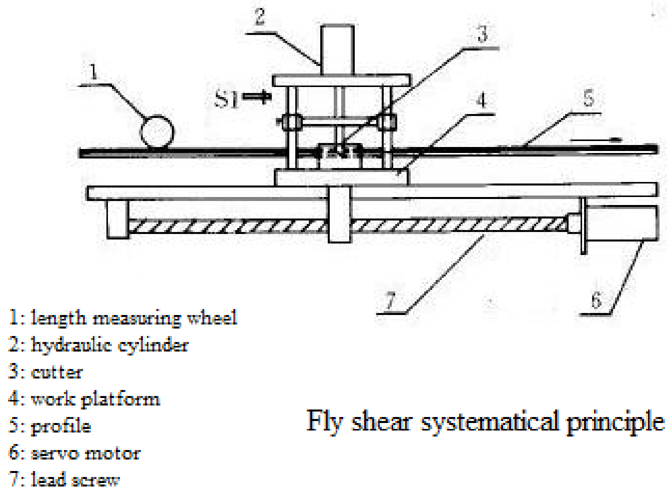
WAIT IDLE(1)

BASE(0,1) 'set axis0 as slave axis, set axis1 as reference axis.

	UNITS=100,100 ATYPE=1,1 DPOS=0,0 SPEED=100,100 ACCEL=2000,2000 DECEL=2000,2000 TRIGGER 'trigger the oscilloscope automatically **MOVELINK**(100,100,0,0,1) AXIS(0) 'the effect is the same as CONNET when Acceleration or Deceleration is not set, and no concern of difference of UNITS, no error accumulation will happen, motion ratio is 1:1 in this situation. MOVE(150)AXIS(1) 'axis1 moves 150, and axis 0 will moves 100. Interpolation path and speed curve: DPOS(0) vertical scale 100, no offset DPOS(1) vertical scale 100, offset 10 MSPEED(0) vertical scale 100, no offset MSPEED(1) vertical scale 100, offset 10  MOVELINK(50,100,0,0,1), vertical scales are the same:



Example2: Fly Shear Application:



Fly shear systematical principle

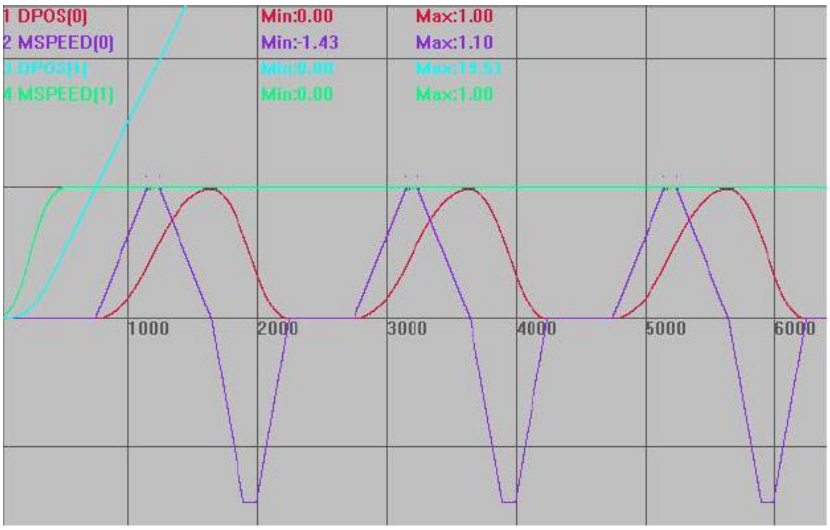
The sectional material keeps moving, and work station keeps stand first. When material moves a determined distance, then the station starts to accelerate until the speed is same as material feeding, the tool S1 will fall down to cut the material, return after cutting is finished, then station then starts to decelerate, move back to its starting position. The process cycle will repeat continuously to get material parts with determined length.

Suppose required length of material is 4m, the motion distance of work station is 1m, axis1 is defined as reference axis (for material feeding), axis 0 is defined as slave axis (fly shear work station), OUT0 is defined as tool cutting control point, then the code is as follow:

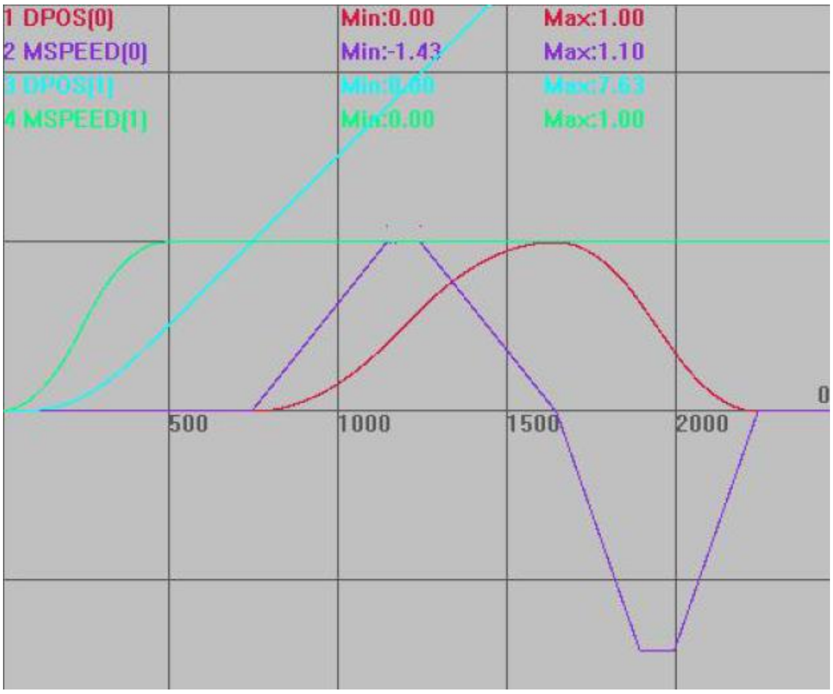
```
RAPIDSTOP(2)
WAIT IDLE(0)
WAIT IDLE(1)
BASE(0,1)
UNITS=10000,10000
```

```
ATYPE=1,1
DPOS=0,0
SPEED=1,1                         'running speed of profile is 1m/s, 60m/min
ACCEL=2,2
DECEL=2,2

VMOVE(1)AXIS(1)                   'the material keep feeding.
TRIGGER                           'trigger the oscilloscope automatically

WHILE 1
    BASE(0)
    MOVELINK(0,1,0,0,1)AXIS(0)    'before the material feeds 1m, station
                                   keeps stand.
    MOVELINK(0.4,0.8,0.8,0,1)AXIS(0)    'the station starts to accelerate.
    MOVELINK(0.2,0.2,0,0,1)AXIS(0)      'follow 0.2m with same speed.
    MOVE_OP2(0,on,1000)           'cutting tool falls down, return after 1s.
                                   (note: time should be calculated)
    MOVELINK(0.4,0.8,0,0.8,1)AXIS(0)
                                  'deceleration stage of work station.
    MOVELINK(-1,1.2,0.5,0.5,1)AXIS(0)
                                  'work station returns to starting position.
WEND
```

Motion Path and Speed Curve:
DPOS(0) vertical scale 1, no offset
MSPEED(0) vertical scale 1, no offset
DPOS(1) vertical scale 1, no offset
MSPEED(1) vertical scale 1, no offset



The operation curve in one period:



Station (slave axis) distance:0.4 (acceleration stage) + 0.2 (synchronous follow) + 0.4 (deceleration stage) = 1m (units), then move back 1 unit.

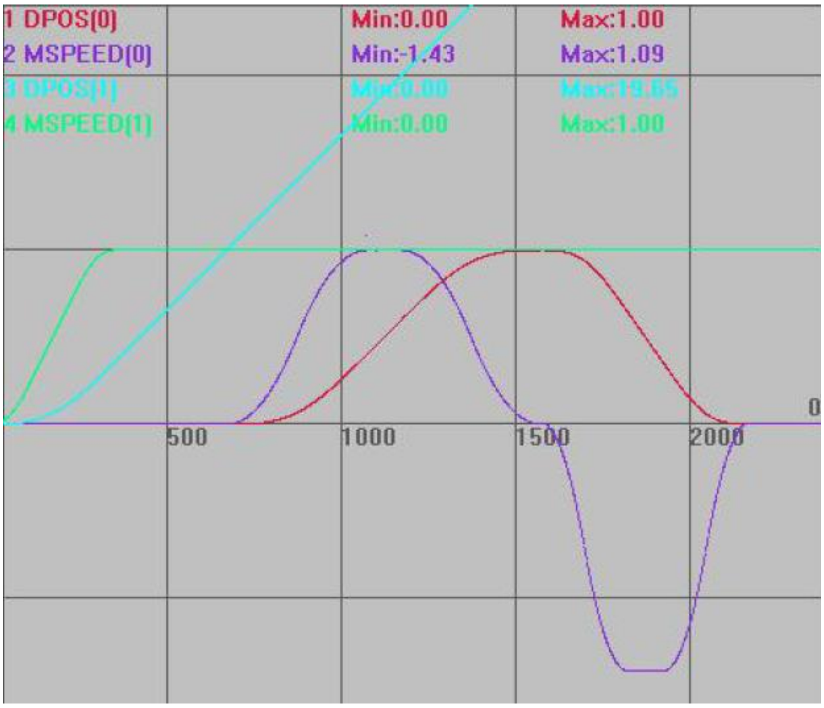
Material feeding (reference axis):1+0.8+0.2+0.8+1.2=4m, and total process is in constant speed.

Example 3: when link options bits3=1, slave axis accelerates and decelerates in S curve.

RAPIDSTOP(2)
WAIT IDLE(0)
WAIT IDLE(1)
DATUM(0)
BASE(0,1)
UNITS=10000,10000
ATYPE=1,1
DPOS=0,0
SPEED=1,1 'material operation speed is 1m/s, 60m/min
ACCEL=2,2
DECEL=2,2
SRAMP=200,200

VMOVE(1)AXIS(1) 'the material keep feeding.
TRIGGER 'trigger the oscilloscope automatically

WHILE 1
BASE(0)
MOVELINK(0,1,0,0,1,8)AXIS(0) 'before the material feeds 1m, station keeps stand.
MOVELINK(0.4,0.8,0.8,0,1,8)AXIS(0) 'the station starts to accelerate.

	MOVELINK(0.2,0.2,0,0,1,8)AXIS(0) 'follow 0.2m with same speed. **MOVE_OP2**(0,on,1000) 'cutting tool falls down, return after 1s. (note: time should be calculated) **MOVELINK**(0.4,0.8,0,0.8,1,8)AXIS(0) 'deceleration stage of work station. **MOVELINK**(-1,1.2,0.5,0.5,1,8)AXIS(0) 'work station returns to starting position. WEND Motion Path and Speed Curve: DPOS(0) vertical scale 1, no offset MSPEED(0) vertical scale 1, no offset DPOS(1) vertical scale 1, no offset MSPEED(1) vertical scale 1, no offset 
Instructions	MOVELINK_MODIFY, MOVESLINK

MOVESLINK-Auto Cam 2

Type	Synchronization Motion Instruction
Description	**This instruction is used for self-defined cam motion, it plans the middle curve automatically, no need of calculating cam table.** The connection axis is following axis, the axis to be linked is reference axis. During the acceleration or deceleration stage, in order to match the speed, start sp of the next MOVESLINK must be same as end sp of current MOVESLINK. Please ensure the parameter(distance)*UNITS passed by instruction is an

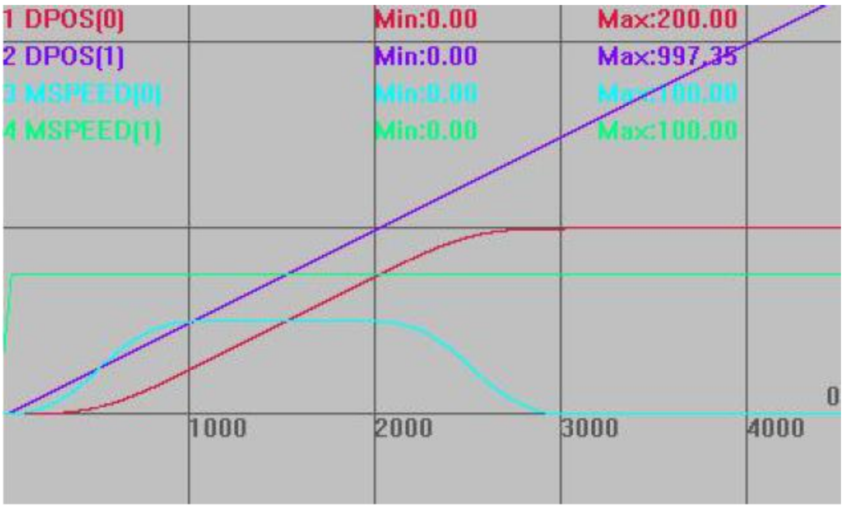
	integer value of pulse, otherwise there will be small errors caused by floats.
Grammar	MOVELINK (distance, link dist, start sp, end sp, link axis[, link options] [, link pos][, link offpos]) behind three are optional parameters, when they are not set, comma must be added, because controller judges them according to their position. distance: distance of slave axis during the link, this parameter can be negative or positive. Units as unit. When it is positive, it will move in forward direction. When it is negative, it will move in inverse direction. link dist: absolute distance of reference axis during the link, units as unit. start sp: speed ratio of reference axis and slave axis when starting, units/units as unit. Negative value means the slave axis moves in inverse direction. end sp: speed ratio of reference axis and slave axis when ending, units/units as unit. Negative value means the slave axis moves in inverse direction. Note: when start sp = end sp = distance/link dist, it moves at constant speed. link axis: axis No. of reference axis. link options: link mode, different binary bits indicate different meanings.

Mode	Bit	Description
1	Bit 0	The connection starts exactly at the moment the MARK event is triggered on the reference axis.
2	Bit 1	The connection starts when reference axis arrives at one absolute position (refer to “link pos”).
4	Bit 2	When Bit2 is set, MOVELINK will automatically execute in cycle, and it can run inversely (this mode can be OFF through setting Bit1 of axis parameter REP_OPTION as 1).
16	Bit 4	Use link offpos to start from a position in the middle, then recover through power failure interruption. Valid in firmware version 20170428 or above.
32	Bit 5	It connects only when the reference axis runs forward.
256	Bit 8	The connection starts exactly at the moment the MARK event is triggered on the reference axis, but it needs the latest firmware.

	link pos: when link options is set as 2, which means the connection starts when the reference axis is at the absolute position value. link offpos: when bit4 of link_options is set as 1, this parameter is the relative distance that master axis has finished. Valid in firmware version 20170428 or above.
Controller	General
Example	Functions are same as MOVELINK, the difference is only the parameter configuration. **Example1:** RAPIDSTOP(2) WAIT IDLE(0) WAIT IDLE(1) DATUM(0) BASE(0,1)

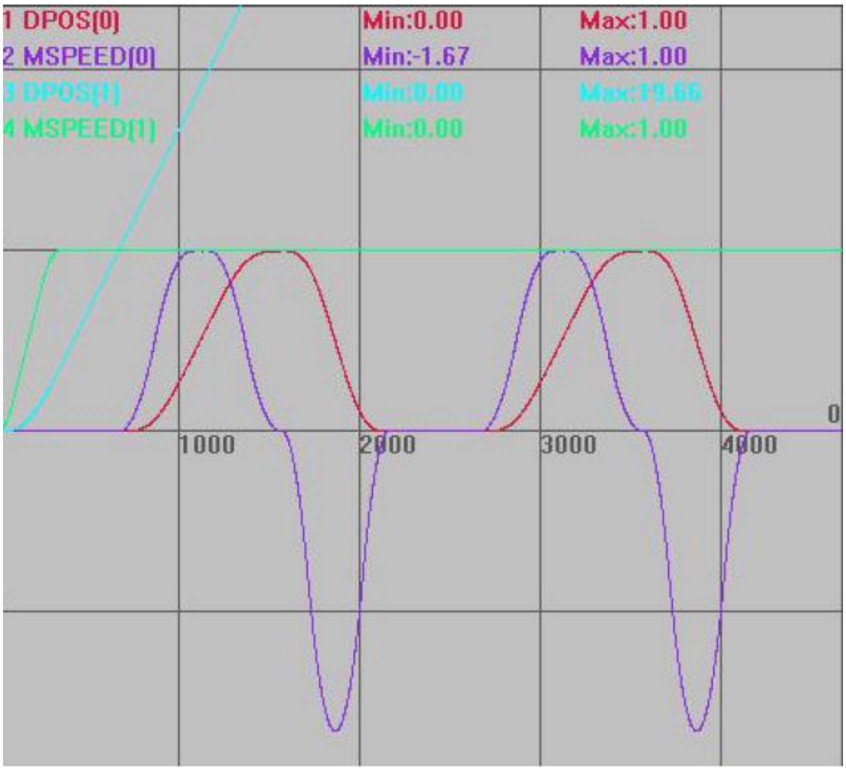
UNITS=100,100
ATYPE=1,1
DPOS=0,0
SPEED=100,100
ACCEL=2000,2000
DECEL=2000,2000
TRIGGER 'trigger the oscilloscope automatically
MOVELINK(50,100,0,1,1) AXIS(0)
'axis 0 follows axis 1, speed is from 0 to the same
MOVELINK(100,100,1,1,1) AXIS(0)
'axis 0 follows axis 1, the constant speed 100units
MOVELINK(50,100,1,0,1) AXIS(0)
'axis 0 follows axis 1, speed is decreased to 0
VMOVE(1) AXIS(1)

Interpolation path and speed curve:
DPOS(0) vertical scale 200, no offset
DPOS(1) vertical scale 200, offset 10
MSPEED(0) vertical scale 200, no offset
MSPEED(1) vertical scale 200, offset 50



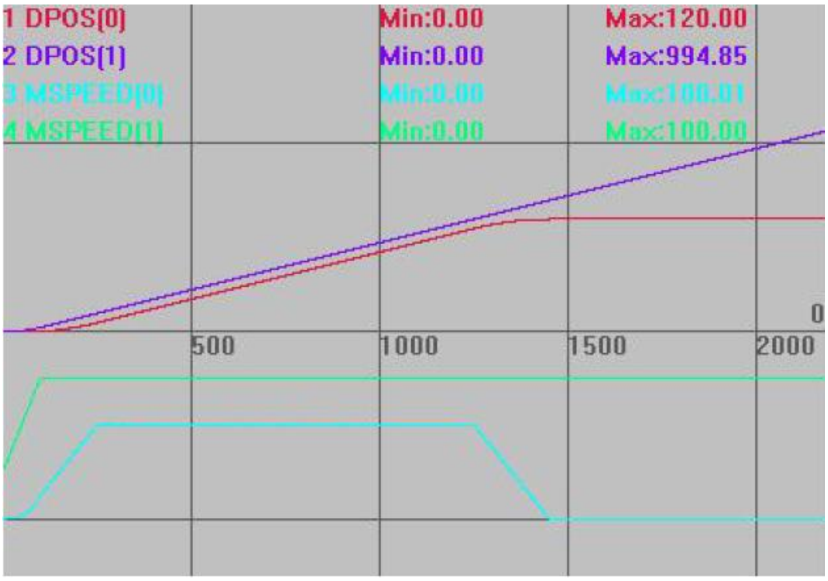
Example2: Fly Shear
RAPIDSTOP(2)
WAIT IDLE(0)
WAIT IDLE(1)
DATUM(0)

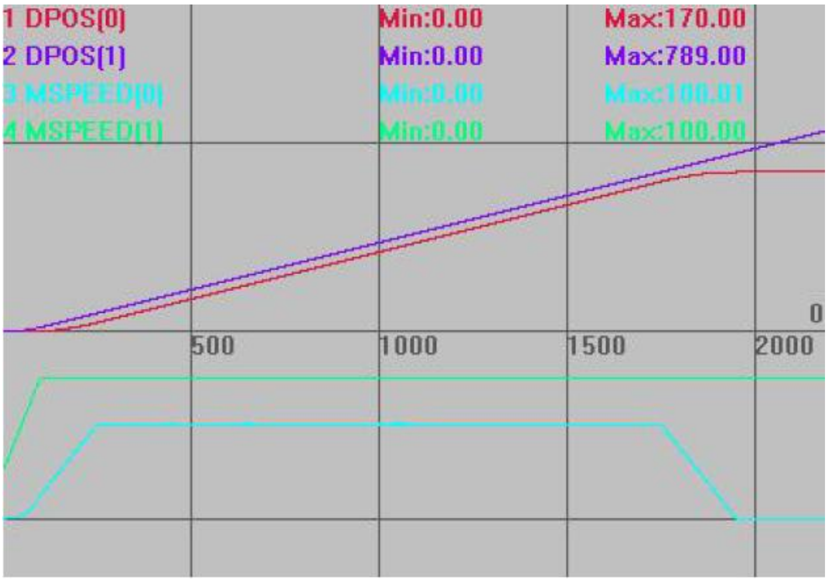
BASE(0,1)
UNITS=10000,10000
ATYPE=1,1
DPOS=0,0
SPEED=1,1
ACCEL=2,2
DECEL=2,2

	SRAMP=200,200 TRIGGER 'trigger the oscilloscope automatically VMOVE(1) AXIS(1) WHILE 1 **MOVESLINK**(0,1,0,0,1)AXIS(0) '1 unit before profile motion, workbench keeps still **MOVESLINK**(0.4,0.8,0,1,1)AXIS(0) 'workbench starts to accelerate. **MOVESLINK**(0.2,0.2,1,1,1)AXIS(0) 'speed following **MOVESLINK**(0.4,0.8,1,0,1)AXIS(0) 'workbench starts to decelerate. **MOVESLINK**(-1,1.2,0,0,1)AXIS(0) 'workbench returns to starting position. WEND Motion Path and Speed Curve: DPOS(0) vertical scale 1, no offset DPOS(1) vertical scale 1, no offset MSPEED(0) vertical scale 1, no offset MSPEED(1) vertical scale 1, no offset 
Instructions	MOVESLINK

MOVELINK_MODIFY-Link Distance Modification

Type	Axis Parameters
Description	**Relatively modify the synchronous length of MOVELINK.** When bringing into motion buffer, it only takes effect after the synchronous segment.

Grammar	VAR1 = MOVELINK_MODIFY, MOVELINK_MODIFY = expression
Controller	With firmware version above 2160926
Example	**Example 1:** RAPIDSTOP(2) WAIT IDLE(0) WAIT IDLE(1) BASE(0,1) UNITS=100,100 ATYPE=1,1 DPOS=0,0 SPEED=100,100 ACCEL=1000,1000 DECEL=1000,1000 TRIGGER 'trigger the oscilloscope automatically Without modify the link distance MOVELINK(10,20,20,0,1) 'acceleration stage of station MOVELINK(100,100,0,0,1) 'link distance of slave axis is 100 MOVELINK(10,20,0,20,1) 'deceleration stage VMOVE(1) AXIS(1) Motion Path and Speed Curve: DPOS(0) vertical scale 200, no offset DPOS(1) vertical scale 200, no offset MSPEED(0) vertical scale 200, offset -200 MSPEED(1) vertical scale 200, offset -150  Add link distance, others same as former. MOVELINK(10,20,20,0,1) 'acceleration stage of station MOVELINK(100,100,0,0,1) 'link distance of slave axis is 100 **MOVELINK_MODIFY=50** 'modify the link distance as 100+50 MOVELINK(10,20,0,20,1) 'deceleration stage



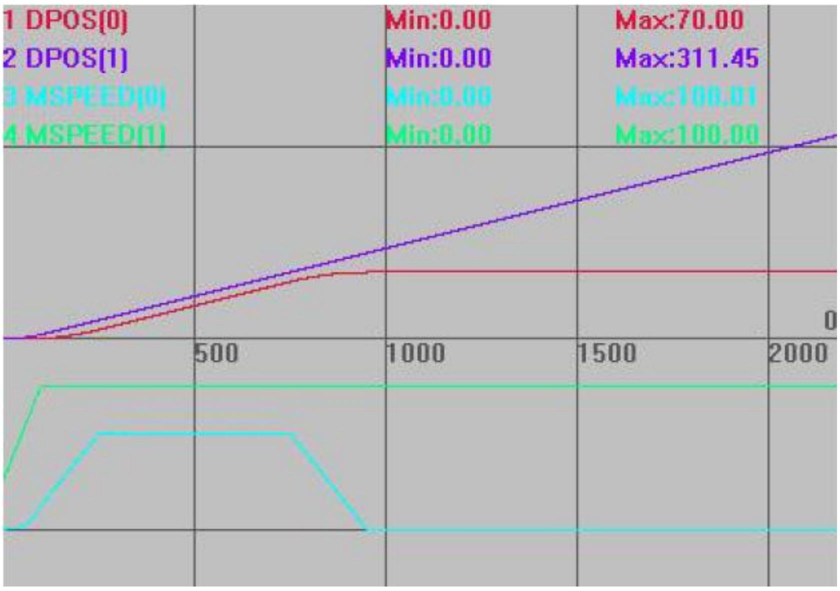
Decrease link distance, others same as former.

MOVELINK(10,20,20,0,1) 'acceleration stage of station

MOVELINK(100,100,0,0,1) 'link distance of slave axis is 100

MOVELINK_MODIDY=-50 'modify the link distance as 100-50

MOVELINK910,20,0,20,1) 'deceleration stage



Note: this instruction only can be used until link distance finished, if in the acceleration or deceleration stage, there will be wrong, and can not modify.

MOVELINK(10,20,20,0,1) 'acceleration stage of station

MOVELINK_MODIFY=50

MOVELINK(100,100,0,0,1) 'link distance of slave axis is 100

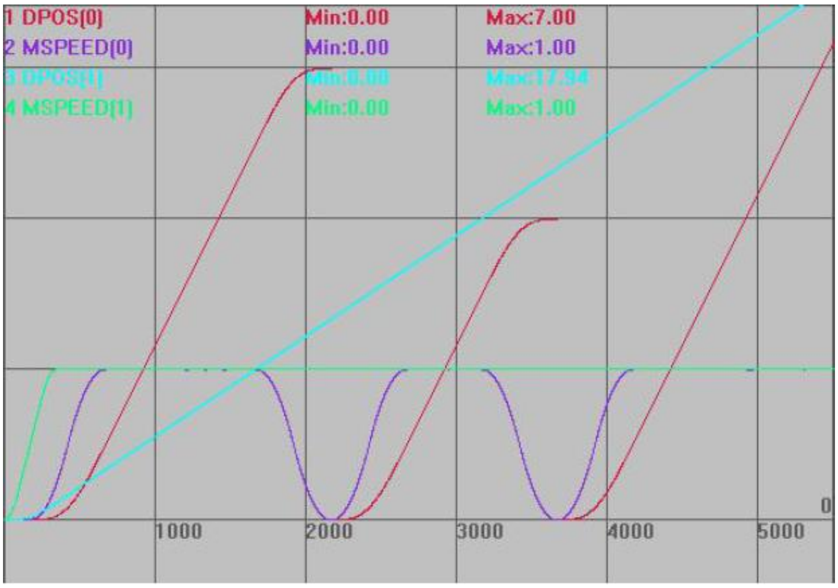
Axis:0 MOVELINK_MODIFY:50.000 failed.

Example 2: (slave axis) fly shear axis accelerates and decelerates in S curve

```
RAPIDSTOP(2)
WAIT IDLE(0)
WAIT IDLE(1)
DATUM(0)

BASE(0,1)
UNITS=10000,10000
ATYPE=0,0
DPOS=0,0
SPEED=1,1                       'material operation speed 1 m/s, 60m/min
ACCEL=2,2
DECEL=2,2
SRAMP=200,200

STOPTASL1
RUNTASK1, Task_FlyShear
DELAY(200)

VMOVE(1)   AXIS(1)              'material keeping motion
TRIGGER                         'trigger oscilloscope automatically
END

Task_FlyShear:
WHILE1
    BASE(0)
    'MOVELINK_MODIFY=0          'clear first
    MOVELINK(3,4,1,1,1,8)   AXIS(0)
    WAIT   UNTIL MPOS(0)>1      'wait until slave axis distance > 2
    MOVELINK_MODIFY=-1          'decrease 1 for slave axis distance
    WAIT IDLE(0)
    WAIT UNTIL MOVELINK_MODIFY=0
                                'wait until synchronic offset finished
    WAIT IDLE(0)

    BASE(0)
    DPOS=0
    'MOVELINK_MODIFY=0    'clear first
    MOVELINK(3,4,1,1,1,8)   AXIS(0)
    WAIT UNTIL MPOS(0)>1    'wait until link distance of slave axis>2
    MOVELINK_MODIFY=1       'add 1 for slave axis distance
    WAIT UNTIL MOVELINK_MODIFY=0
                                'wait until synchronic offset finished
WEND
```

Motion Path and Speed Curve:
DPOS(0) vertical scale 1, no offset
DPOS(1) vertical scale 3, no offset
MSPEED(0) vertical scale 1, no offset

<table>
<tr><td></td><td>MSPEED(1) vertical scale 1, no offset



</td></tr>
<tr><td>Instructions</td><td>MOVELINK</td></tr>
</table>

MOVESYNC – Sychronous Motion

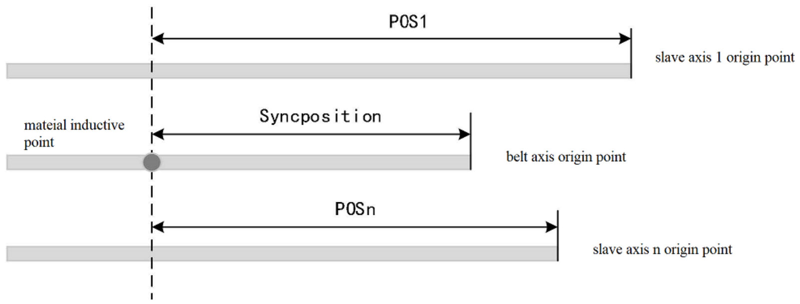
<table>
<tr><td>Type</td><td>Motion Setting Instruction</td></tr>
<tr><td>Description</td><td>Motion synchronization, Belt objects follow to move. This isn't interpolation motion, so it can't ensure linear path.

The belt axis length unit is required the same as slave axis of BASE.
When BASE axis finished follow motion, this instruction ends. In this situation, if corresponding inductive position of belt objects has moved a certain distance, then BASE axis is not in the absolute position, and it is running in follow speed.

MOVESYNC supports continuous using, it won't interrupt speed continuity, and can add MOVE_OP in the middle. In case the high-speed follow motion stop directly when motion finished, the final instruction, MOVESYNC, please use Mode -1.
This instruction belongs to CAM instruction, doesn't support motion pause.</td></tr>
<tr><td>Grammar</td><td>MOVESYNC(mode,synctime,syncposition,syncaxis,pos1[,pos2, pos3…])

<table>
<tr><td>Mode</td><td>Description</td></tr>
<tr><td>-1</td><td>synchronization mode ends, motion has reached defined absolute position. If there are other MOVESYNC instructions next, it will be covered, syncaxis is invalid under the mode.</td></tr>
<tr><td>-2</td><td>force it to end. When -2 is called, original MOVESYNC instruction will stop, and move to defined ending position. If there are other MOVESYNC instructions next, it will be covered, syncaxis is invalid under the mode.</td></tr>
</table>
</td></tr>
</table>

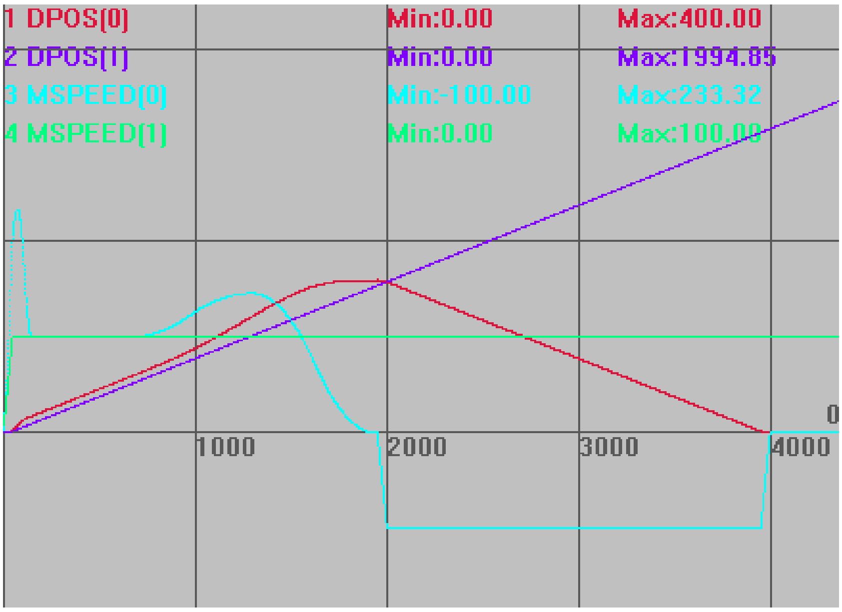
<table>
<tr><td></td><td>
<table>
<tr><td>0</td><td>the first axis(x) of BASE follows Belt axis objects.</td></tr>
<tr><td>10</td><td>the second axis(y) of BASE follows Belt axis objects.</td></tr>
<tr><td>20</td><td>the third axis of BASE follows Belt axis objects.</td></tr>
</table>
mode = 0+ angle, angle: belt rotation angle, angle = forward rotating angle between belt and the first/second axis of BASE axis. Such as, Mode=PI/4, belt is at 45 degrees. Mode=PI/2, belt is at Y direction. Mode=PI, belt is at x negative direction. Mode=(PI*1.75), belt is at -45 degrees.
synctime: synchronization time, ms as unit, and the motion will finish in defined time, when it finished, BASE axis follows belt and their speed are the same. 0 means the synchronization time can be estimated according to motion axis' speed, acceleration, but sometimes not accurate.
syncposition: belt position when belt objects are reacted, it supports belt axis coordinate cycle, but if it is called, ensure coordinate is not modified or operated cycle between the parameter position and belt axis position. Therefore, don't use the instruction near the cycle point.
syncaxis: belt axis NO., -1 means no belt axis, moving to pos1 directly.
pos1: the first axis(BASE) absolute position when belt object is reacted.
pos2: the n axis(BASE) absolute position when belt object is reacted.



</td></tr>
<tr><td>Controller</td><td>With firmware version above 170601.</td></tr>
<tr><td>Example</td><td>
Example 1: belt takes the material
RAPIDSTOP(2)
WAIT IDLE(0)
WAIT IDLE(1)
BASE(0,1)
DPOS=0,0
UNITS=100,100
ATYPE=1,1
SPEED=100,100
ACCEL=1000,1000
DECEL=1000,1000
TRIGGER
MOVESYNC(0, 0, 100, 1, 120) 'move to belt objects synchronically
MOVE_OP(1,1) 'decrease, if axis Z decreases, also can be used by MOVESYNC
MOVE_OP(0,1) 'open nozzle
MOVESYNC(0,1000,100,1,120) 'continue to follow 1s
MOVE_OP(1,0) 'increase
</td></tr>
</table>

```
MOVESYNC(-1, 0, 0, -1, 400)      'move to position where put materials400
MOVE_OP(1,1)                     'decrease
MOVE_OP(0,0)                     'close nozzle
MOVE_DELAY(2)                    'delay 2ms, can't insert these sentences in
                                 MOVESYNC continuous motion
MOVE_OP(1,0)                     'increase
MOVEABS(0)                    'back to origin
VMOVE(1) AXIS(1)              'belt axis motion
```

Motion path and speed curve:
DPOS(0) vertical scale 500, no offset
DPOS(1) vertical scale 500, no offset
MSPEED(0) vertical scale 200, no offset
MSPEED(1) vertical scale 200, no offset

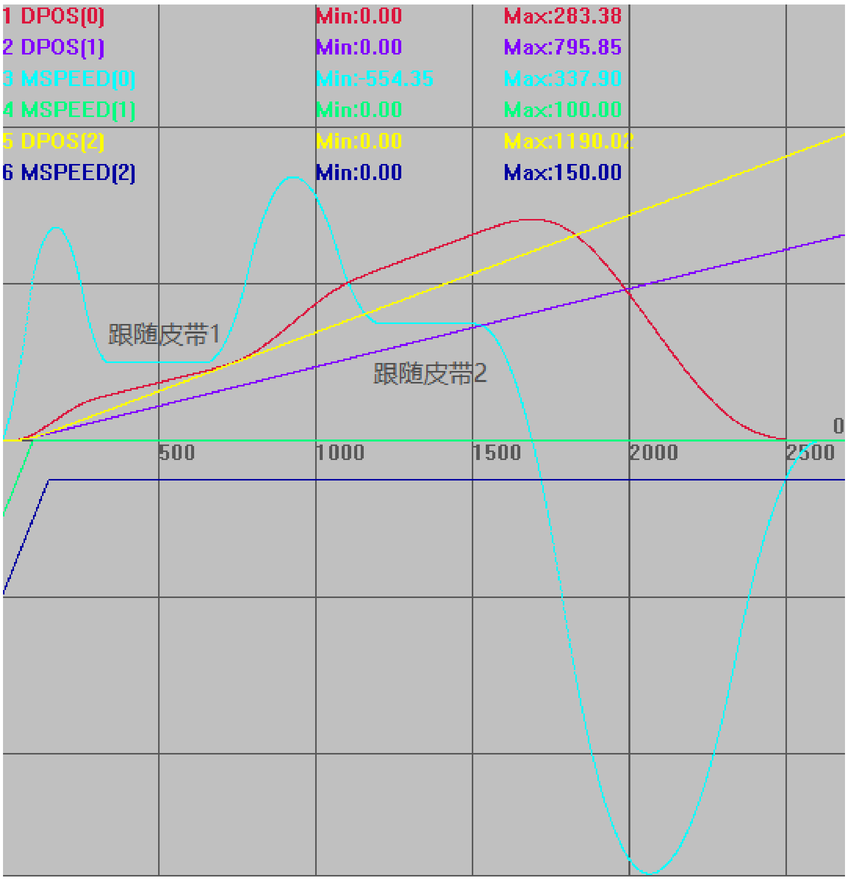


Example 2: Take the material from the belt to another belt.

```
RAPIDSTOP(2)
WAIT  IDLE(0)
WAIT  IDLE(1)
BASE(0,1,2)
DPOS=0,0,0
UNITS=100,100,100
ATYPE=1,1,1
SPEED=1000,100,150               'set different speeds
ACCEL=1000,1000,1000
DECEL=1000,1000,1000
TRIGGER
MOVESYNC(0, 0, 50, 1, 80 )       'move to belt object synchronically
MOVE_OP(0,1)                     'open the nozzle
MOVE_OP(1,1)         'decrease, if axis Z decreases, can use MOVESYNC
MOVESYNC(0, 300, 50, 1, 80 )     'continue to synchronize 2ms
```

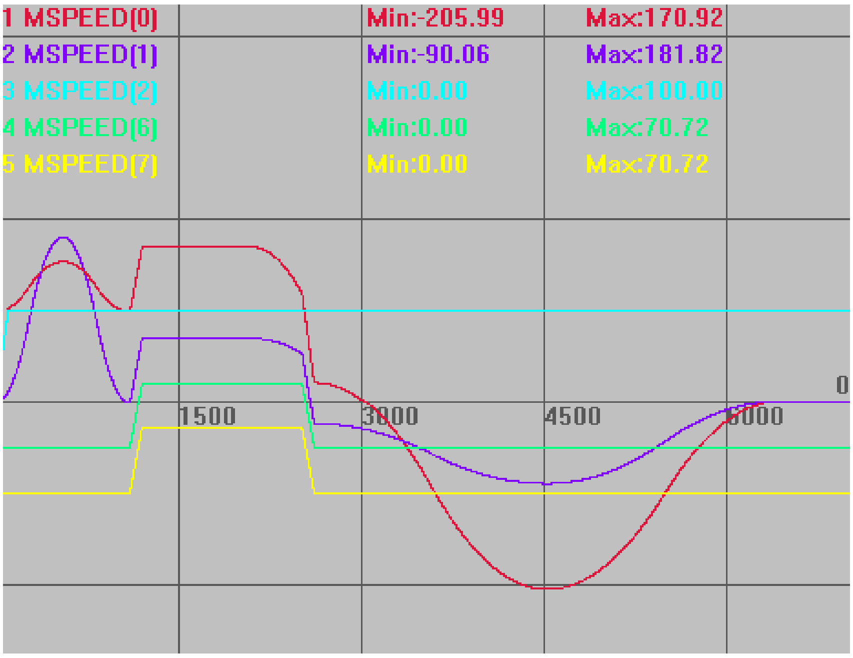
MOVE_OP(1,0) 'increase
MOVESYNC(0, 0, 100, 2, 150) 'move to the second corresponding belt
MOVE_OP(1,1) 'decrease
MOVE_OP(0,0) 'close the nozzle
MOVESYNC(0,300, 100, 2, 150) 'synchronize 2ms
MOVE_OP(1,0) 'increase
MOVESYNC(-1, 0, 0, -1, 0) 'move to stop position
VMOVE(1) AXIS(1) 'motion of belt axis 1
VMOVE(1) AXIS(2) 'motion of belt axis 2

Motion path and speed curve:
DPOS(0) vertical scale 200, no offset
DPOS(1) vertical scale 200, no offset
MSPEED(0) vertical scale 200, no offset
MSPEED(1) vertical scale 200, offset -200
DPOS(2) vertical scale 200, no offset
MSPEED(2) vertical scale 200, offset -200

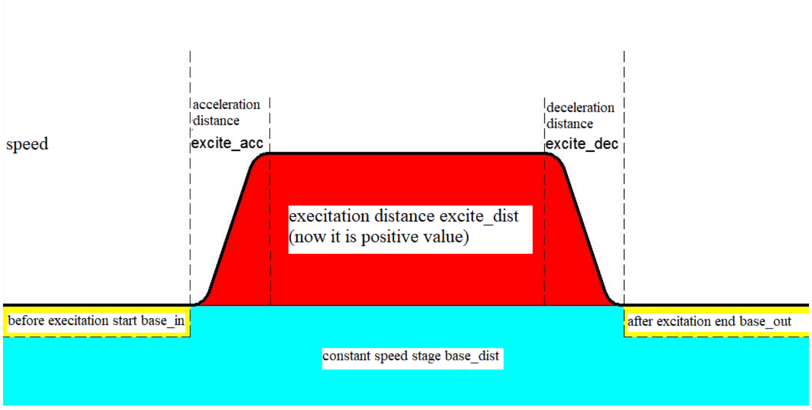


Example 3: Carved on the belt object
RAPIDSTOP(2)
WAIT IDLE(0)
WAIT IDLE(1)
BASE(0,1,2,6,7)
UNITS=100,100,100,100,100

	DPOS=0,0,0,0,0 SPEED=100,100,100,100,100 ACCEL=1000,1000,1000,1000,1000 DECEL=1000,1000,1000,1000,1000 TRIGGER ADDAX(6) AXIS(0) 'carve on the virtual axis, then superpose to actual axis ADDAX(7) AXIS(1) BASE(0, 1) **MOVESYNC(0, 0, 50, 2, 80,100) 'move synchronically to belt object** MOVE_TASK(1, task1) 'trigger superposition axis carving **MOVESYNC(0, 1000, 50, 2, 80, 100)'longer carving motion time** **MOVESYNC(-1, 0, 0, -1, 0 ,0)** 'move to stop position VMOVE(1) AXIS(2) 'belt axis motion END TASK1: DELAY(2) 'when superposition carving in process, absolute motion instruction position will be wrong, delay for avoiding calling instructions. BASE(6, 7) MOVE(100,100) 'carve with lines in two sides WAIT IDLE 'wait until carving ends BASE(0, 1) **MOVESYNC(-2, 0, 0, -1, 0 ,0)** 'when carving finished, force to move to stop position Motion path and speed curve MSPEED(0) vertical scale 200, no offset MSPEED(1) vertical scale 200, no offset MSPEED(2) vertical scale 200, no offset MSPEED(6) vertical scale 200, offset -50 MSPEED(7) vertical scale 200, offset -100

FLEXLINK--Excitation Motion

Type	Synchronization Motion Instruction
Description	**This instruction is used to realize excitation motion of axis. It consists of uniform motion and excitation motion.**  Please ensure the distance (pulse amount, parameters*units) is integer,or will cause slight motion precision error if it is a float value.
Grammar	FLEXLINK(base_dist, excite_dist, link_dist, base_in, base_out, excite_acc, excite_dec, link_axis, link_options, [start_pos], [link_offpos]) Parameters： base_dist: uniform motion distance of slave axis. excite_ist: excitation motion distance of slave axis, +: increase, -: decrease. Total distance of slave aixs= base_dist + excite_dist. link_dist: distance of main axis during the whole link.

base_in: percentage of base_dist that distance of slave axis will possess before the excitation starts.

base_out: percentage of base_dist that remaining distance of slave axis will possess after excitation motion. (base_in + base_out should not exceed 100%, or excite_dist will be invalid)

excite_acc: percentage of excite_dist that slave axis' acceleration distance will possess during the excitation motion, when excite_dist is minus, indicating deceleration stage.

excite_dec: percentage of excite_dist that slave axis' deceleration distance will possess during the excitation motion, when excite_dist is minus, indicating acceleration stage.

(base_in, base_out, excite_acc and excite_dec will be valid only when excite_dist is not 0.)

link_axis: main aixs NO.

link_options: link mode with reference axis (main axis), different binary bit value has different meanings.

Bit	Description
bit0	link starts when Mark(latch is triggered) of reference axis is on.
bit1	link starts when reference axis reaches set absolute position.
bit2	repeat double direction motion continuously. (cancel the repeat by setting REP_OPTION=1).
bit4	CAM starts in the middle.
bit5	link only happens when the reference axis moves in positive direction.
bit8	link starts when MARKB is on.

start_pos: absolute position trigger

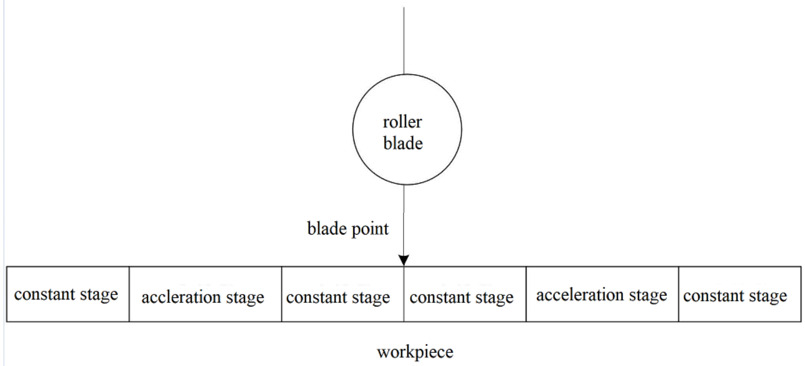
link_offpos: middle position where CAM starts

Controller

ZMC4XX series with firmware version above 20170518.
ZMC3XX series with firmware version above 20161212.

Example

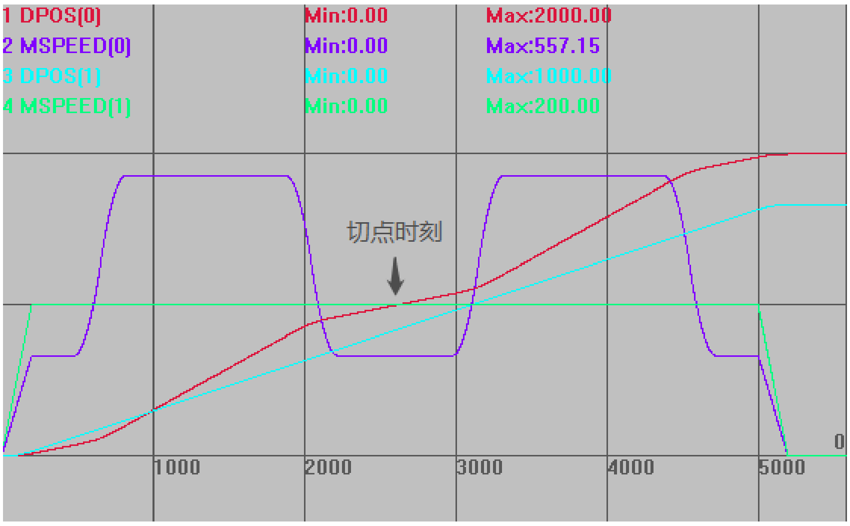
Example 1: Round Cutting




```
RAPIDSTOP(2)
WAIT  IDLE(0)
WAIT  IDLE(1)
BASE(0,1)
DPOS=0,0
```

UNITS=100,100
SPEED = 200,200
ACCEL=1000,1000
DECEL=1000,1000
TRIGGER
FLEXLINK(500,500,500,15,15,20,20,1)
    'cutting tool will move 500 units, material will move 1000 units, the speed where cutting begin will be the same, it is just the position of one cycle.
FLEXLINK(500,500,500,15,15,20,20,1)   AXIS(0)
MOVE(1000)   AXIS(1)

Motion path and speed curve:
DPOS(0) vertical scale 1000, no offset
MSPEED(0) vertical scale 300, no offset
DPOS(1) vertical scale 600, no offset
MSPEED(1) vertical scale 200, no offset

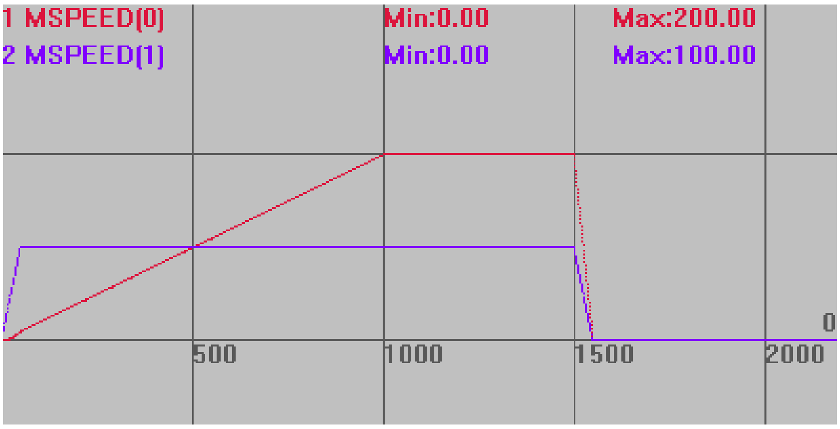


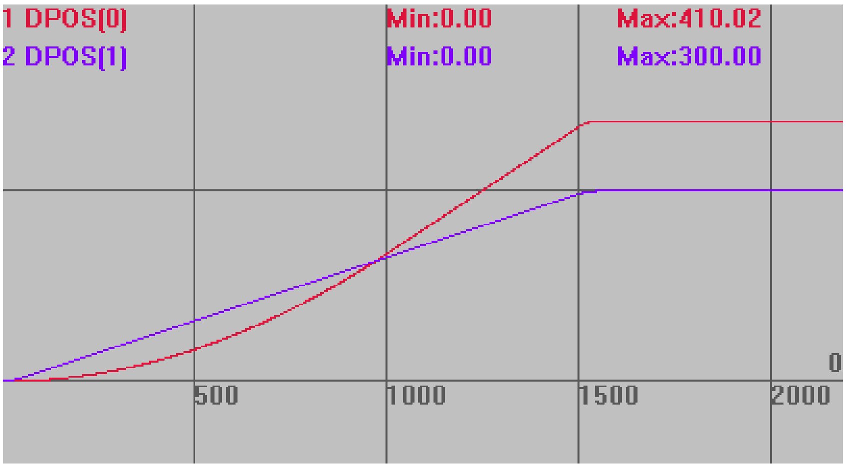
7.5 Motion Setting Instructions

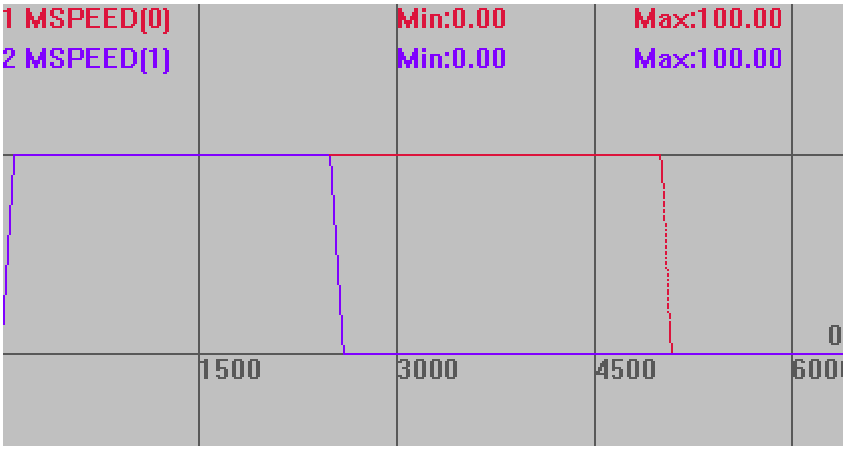
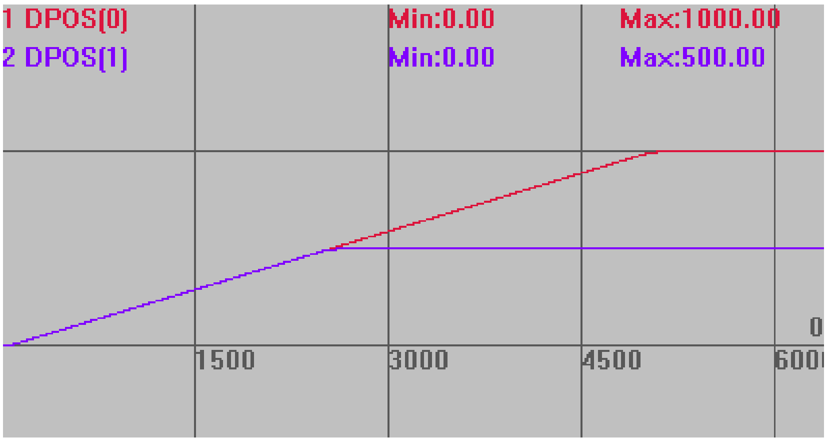
CLUTCH_RATE--Link Speed

Type	Axis Parameters
Description	**link speed of instruction CONNECT, default value is 1000000.** It is used to define changing time of connection ration from 0 to ratio configuration, the unit is ratio/s, please see example 1. If the value is not set far bigger than link ratio of CONNECT, then actual ratio will be smaller. Please see offset curve graph of example 1. When it is set as 0, the link will change as per the value of followed axis speed/acceleration, it is suitable in handwheel link. (When speed is too slow, link will end after motion continue to move some distance.)

<table>
<tr><td>Grammar</td><td>CLUTCH_RATE= value</td></tr>
<tr><td>Controller</td><td>General</td></tr>
<tr><td>Example</td><td>Example 1:
BASE(0,1)
ATYPE=1,1
DPOS=0,0
UNITS=100,100
SPEED=100,100
ACCEL=1000,1000
DECEL=1000,1000
CLUTCH_RATE=1 'set link ratio as 1 ratio/s
TRIGGER 'trigger oscilloscope automatically
CONNECT(2,1) AXIS(0) 'link ratio is 2, need 2 seconds to build link.
MOVE(300) AXIS(1) 'axis 1 moves, axis 0 follows.

Speed curve, link time is based on link ratio and clutch_rate
MSPEED(0) vertical scale 200
MSPEED(1) vertical scale 200


Offset curve, clutch_rate is too small, actual link ratio will be less than 2:1.
MSPEED(0) vertical scale 300
MSPEED(1) vertical scale 300

</td></tr>
</table>

	Example 2: BASE(0,1) DPOS=0,0 ATYPE=1,1 UNITS=100,100 SPEED=100,100 ACCEL=500,500 DECEL=500,500 **CLUTCH_RATE**=0 'link as per value of followed axis speed/acceleration, maybe not synchronized TRIGGER 'trigger oscilloscope automatically CONNECT(2,1) AXIS(0) 'link speed is 0.2s, link time is determined by slave axis speed / acceleration VMOVE(500)AXIS(1) Speed curve MSPEED(0) vertical scale 100 MSPEED(1) vertical scale 100  Offset curve: MSPEED(0) vertical scale 1000 MSPEED(1) vertical scale 1000 
Instructions	CONNECT

ENCODER_RATIO-Gear Ratio of Encoder

Type	Motion Setting Instruction
Description	**Input Gear Ratio of Encoder, default value is (1,1). The direction can be changed by setting as minus value.**
Grammar	ENCODER_RATIO(mpos_count, input_count,[, mode]) mpos_count: numerator, maximum is 65535 input_count: denominator. maximum is 65535 Mode / Description: 1 — AB 1X Mode 2 — AB 2X Mode 3 — AB 4X Mode Please set ATYPE as encoder type, then call mode to set. Valid in firmware ZMC406 20170502 above.
Controller	General
Example	ENCODER_RATIO(4,1) 'encoder 4 times input, which equals to ENCODER_RATIO (1,1,4) ENCODER_RATIO(1,-1) 'encoder input to switch the direction, which equals to ENCODER_RATIO (-1,1)
Instructions	PP_STEP, ENCODER

Mode	Description
1	AB 1X Mode
2	AB 2X Mode
3	AB 4X Mode

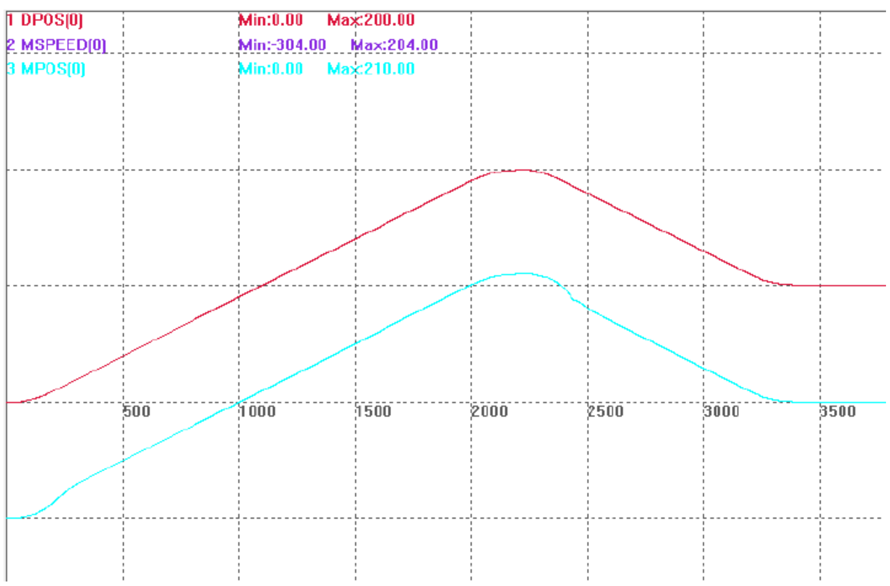
STEP_RATIO- Gear Ratio of Motor

Type	Motion Setting Instruction
Description	**Set output gear ratio of stepper, default value is (1,1). Range:1-65535.** The motor direction can be changed by setting minus value, but it is not recommend. Pulse motor uses INVERT_STEP, bus motor modifies in the actuator. Don’t modify the value frequently, it is better to change the pulse amount to realize the same effect.
Grammar	STEP_RATIO(output_count, input_count) output_count: numerator, maximum is 65535 input_count: denominator, maximum is 65535
Controller	General
Example	**STEP_RATIO** (16,1) 'pulse output 16 times of the set pulse value. Also it can be achieved through pulse equivalent multiples 16.

BACKLASH- Reverse Clearance Compensation

Type	Motion Setting Instruction

Description	**To set reverse compensation of axis, not valid in extended axis.**
Grammar	BACKLASH (enable [,dist[, speed, acceleration]]) enable enable or not. dist distance, UNITS as unit. speed speed of reverse compensation. acceleration acceleration of reverse compensation.
Controller	General
Example	**Example 1:** **BACKLASH**(0) 'shut reverse compensation function. **BACKLASH**(1, 0.1) 'set reverse compensation as 0.1mm. **Example 2:** RAPIDSTOP(2) WAIT IDLE(0) BASE(0) ATYPE = 5 'with encoder feedback SPEED =1000 ACCEL = SPEED * 10 DECEL = SPEED * 10 SRAMP = 0 DPOS = 0 MPOS = 0 **BACKLASH**(0) 'close reverse gap TRIGGER 'apply reverse clearance parameters **BACKLASH**(1, 10, 50, 100) '10mm compensation MOVE(200) MOVE(-100) 'start to compensate when reverse END

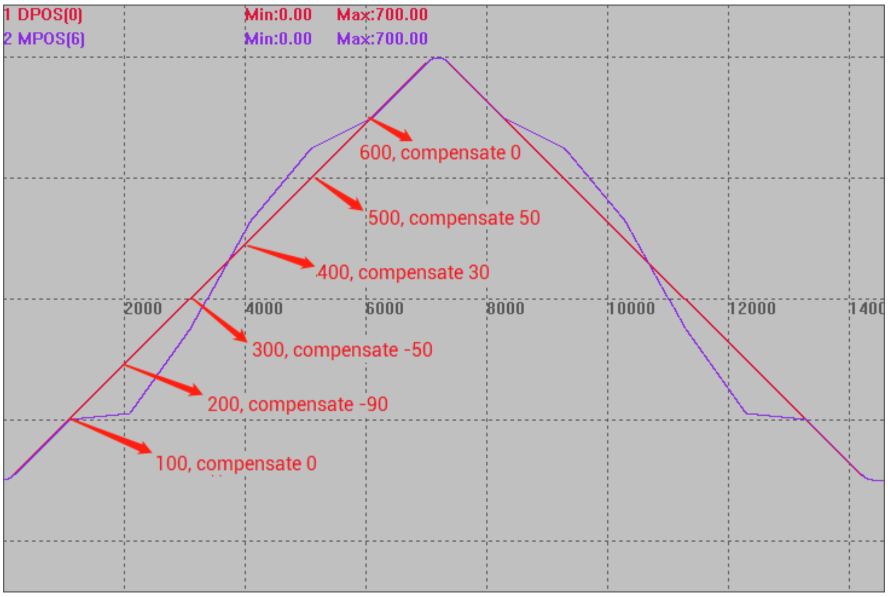
	

PITCHSET -- Screw Pitch Compensation

Type	Motion Setting Instruction
Description	**To set axis screw pitch compensation, not valid in extended axis.** The number of compensation pulses of each point are saved into TABLE.
Grammar	PITCHSET(enable [, startpos, disone, maxpoint, tablenum]) enable enable or not. startpos the MPOS position where compensation starts, UNITS as unit. Note: the point that corresponds to startpos is not saved. disone distance between points, UNITS as unit. maxpoint total points need to be compensated tablenum TABLE position where saved the compensation point, it saves starting from next point of "startpos", the unit is pulse **Support dynamically modifying compensation parameters.** **When the compensation is ON or OFF, adjust dpos and mpos, but don't make them correct by the motion.**
Controller	General
Example	**Singe-Axis Pitch Compensation:** **Example 1:** ATYPE(6)=6 UNITS(6)=100 DPOS(6)=0 BASE(0) ATYPE=1 UNITS=100

SPEED=100
ACCEL=500
DECEL=500
TABLE(0, 0*UNITS(0), -90*UNITS(0), -50*UNITS(0), 30*UNITS(0), 50*UNITS(0),0)
'TABLE saves pitch compensation value, the value is the number of pulses, not the pitch compensation value.
DPOS=0
MPOS=0
'//starting compensation position (MPOS)   compensation value (the number of pulses at a distance of 100)
'//   100   0
'//   200   -90
'//   300   -50
'//   400   30
'//   500   50
'//   600   0
PITCHSET(1,0,100,6,0)   'when MPOS=0, it starts to compensate 6 points, a space of 100
TRIGGER
MOVE(70)
MOVE(-700)
WAIT IDLE
PITCHSET(0,100,100,6,0)

In this waveform, in order to show compensation effect, connect axis 0 pulse OUT to axis 6 encoder IN, and do some offsets for obvious compensation effects.
DPOS(0) – offset -300, vertical scale (Y scale): 200
DPOS(6) – offset -300, vertical scale (Y scale): 200



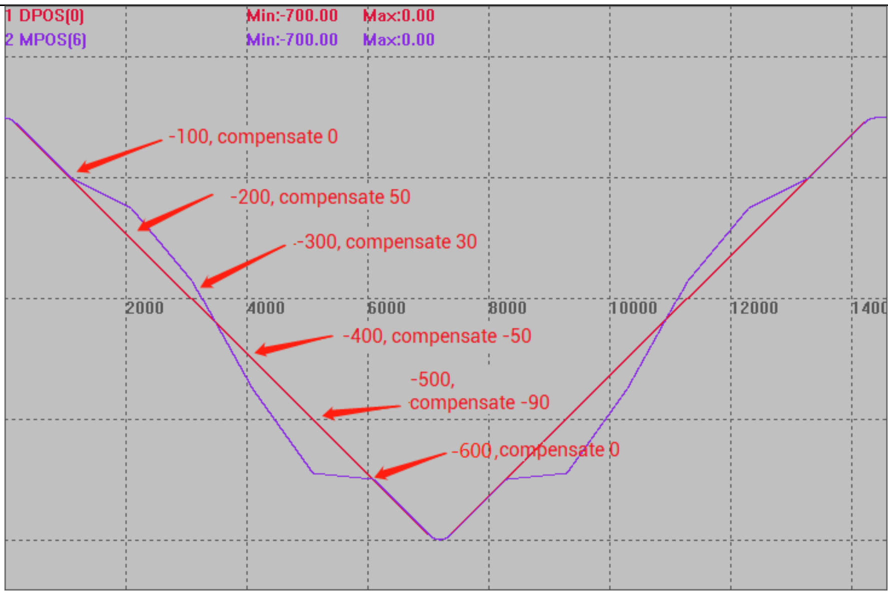
Example 2:

```
ATYPE(6)=6
UNITS(6)=100
DPOS(6)=0

BASE(0)
ATYPE=1
UNITS=100
SPEED=100
ACCEL=500
DECEL=500
TABLE(0, 0*UNITS(0), -90*UNITS(0), -50*UNITS(0), 30*UNITS(0), 50*UNITS(0),0)
'TABLE saves pitch compensation value, the value is the number of pulses, not the pitch compensation value.
DPOS=0
MPOS=0
'//starting compensation position (MPOS)    compensation value (the number
                                            of pulses at a distance of 100)
'//    100                                  0
'//    200                                  -90
'//    300                                  -50
'//    400                                  30
'//    500                                  50
'//    600                                  0
PITCHSET(1,-700,100,6,0)
'when MPOS=700, it starts to compensate 6 points, a space of 100
TRIGGER
MOVE(-700)
MOVE(700)
WAIT IDLE
PITCHSET(0,100,100,6,0)
```

In this waveform, in order to show compensation effect, connect axis 0 pulse OUT to axis 6 encoder IN, and do some offsets for obvious compensation effects.

DPOS(0) – offset 300, vertical scale (Y scale): 200

DPOS(6) – offset 300, vertical scale (Y scale): 200



Example 3: Multi-Axis Pitch Compensation

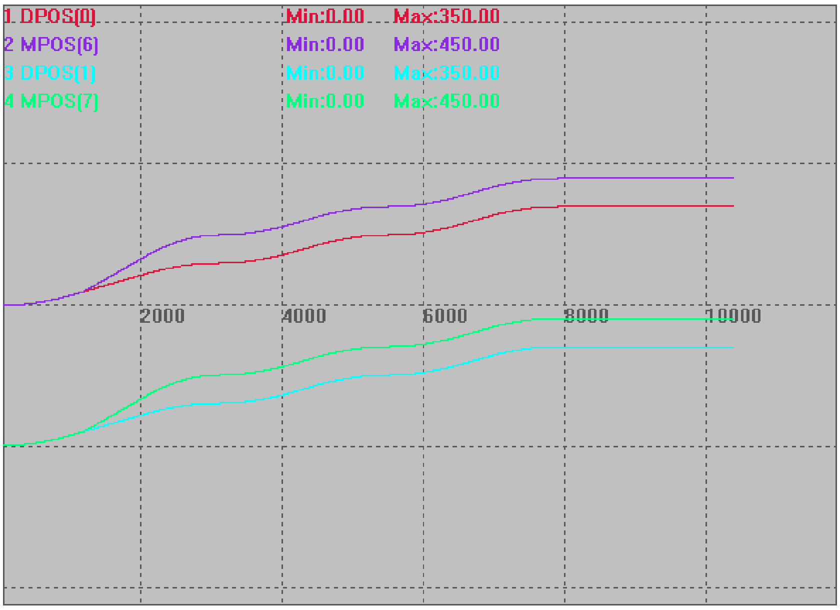
BASE(6,7)
UNITS=100,100
SPEED=100,100
ACCEL=100,100
DPOS=0,0
MPOS=0,0

BASE(0,1)
ATYPE=5,5
UNITS=100,100
SPEED=100,100
ACCEL=100,100
DPOS=0,0
MPOS=0,0

TABLE(0, 100*UNITS(0), 100*UNITS(0), 100*UNITS(0))
'TABLE saves pitch compensation value, the value is the number of pulses, not the pitch compensation value.

BASE(0)
PITCHSET(0)
PITCHSET(1,50,100,3,0)
‘when MPOS=50, start to compensate 3 points, a space of 100
BASE(1)
PITCHSET(0)
PITCHSET(1,50,100,3,0)
‘when MPOS=50, start to compensate 3 points, a space of 100

TRIGGER

	BASE(0,1) MOVE(150,150) wait IDLE ?PITCH_DIST(0) ?PITCH_DIST(1) MOVE(100,100) wait IDLE ?PITCH_DIST(0) ?PITCH_DIST(1) MOVE(100,100) wait IDLE ?PITCH_DIST(0) ?PITCH_DIST(1) In this waveform, in order to show compensation effect, connect axis 0, axis 1 pulse OUT to axis 6, axis 7 encoder IN. and do some offsets for obvious compensation effects. DPOS(0) – offset 0, vertical scale (Y scale): 500 DPOS(6) – offset -0, vertical scale (Y scale): 500 DPOS(1) – offset -500, vertical scale (Y scale): 500 DPOS(7) – offset -500, vertical scale (Y scale): 500 
Instruction	PITCH_DIST

PITCH_DIST -- Pitch Compensation Distance

Type	Axis State
Description	**Read distance value of current axis pitch compensation, the real MPOS returned value will minus the value.**

Grammar	VAL=PITCH_DIST (axisnum) axisnum: axis No.
Controller	General
Instruction	PITCHSET

7.6 Robot Instructions

CONNFRAME – Inverse Solution of Robotic Arm

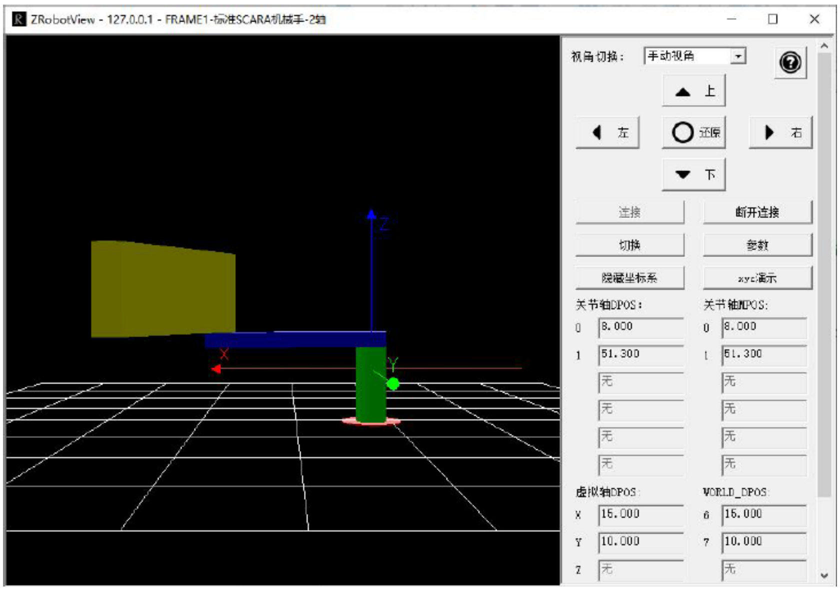
Type	Synchronization Motion Instruction
Description	**Target position of current joint coordinate correlates with virtual coordinate.** When CLUTCH_RATE=0, motion speed and acceleration of joint coordinate are limited by SPEED and other parameters. ⚠When there is warning, motion will be canceled by CANCEL. ⚠Don't CANCEL the motion when virtual axis is running at high speed, axis will stop. ⚠Virtual axis coordinate will be modified automatically under LOAD, making it same as joint axis, so need to use WAIT LOADED for starting moving. ⚠Do not modify the virtual axis coordinate during link process, or do not call DATUM and other instructions that might modify coordinate, it will cause joint axis move to a new virtual position rapidly. ⚠When CONFRAME is taken effect, MTYPE=33, now joint axis can't move directly, it needs virtual axis to move joint axis. When wants to move joint axis directly, call the CANCEL instruction to cancel CONNFRAME at first, then move joint axis. ⚠When virtual axis and actual axis are the rotating axis, their pulse amount should be the same, for example, terminal axis of rotation.
Grammar	CONNFRAME(frame, tablenum, viraxis0, viraxis1,[…]) frame: coordinate type, 1- scara (if needs special defined robotic arm type, please contact with manufacturers) tablinum: TABLE position for saving conversion parameters. When frame=1, save one by one: the first joint axis length, the second joint axis length, the first joint axis pulse amount as per round, the second joint axis pulse amount as per round. viraxis0: the first axis of virtual coordinate viraxis1: the second axis of virtual coordinate

	[…]: the N axis of virtual coordinate, it can be actual axis, exact axis is determined by robotic arm type. **FRAME List of mechanical structures** Please see *Zmotion robotic arm instructions* for details. Please contact with manufacturers if needs other special robotic structures.

frame	Structure Type
1	Standard SCARA robotic arm
101	SCARA + swing, 4 defined virtual axes
105	SCARA + swing, 5 defined virtual axes
106	Special SCARA
107	Special SCARA
108	Special 5-axis SCARA
11	Rotary table
17	Double- rotary table
18	Offset rotary table
19	Offset double-rotary table
3	Palletizing robotic arm
103	Palletizing deformation, spraying robotic arm
5	Rotary scalable robotic arm
15	XY sliding table
102	2-axis delta
2	3-axis delta, R type controllers support
12	4-axis delta, R type controllers support
13	3-axis vertical spider hand, R type controllers support
25	5-joint robotic arm
6	Robotic arm with 6 DOF (degree of freedom), R type controllers support
26	Special 6 DOF
36	Special 6 DOF
100	XYZ+2-axis wrist, defined 3 virtual axes
104	XYZ+2-axis wrist, defined 5 virtual axes

Controller	General
Example	See below

```
DIM L1,L2
L1=10                        'the first joint axis length
L2=10                        'the second joint axis length
BASE(0,1)                    'joint axis number is 0,1
ATYPE=1,1
UNITS=10,10                  'pulse amount, degree as unit
DPOS=0,0                     'set joint axis position, and modify it according
                              to actual situation
BASE(6,7)                    'virtual axis number is 6,7
ATYPE=0,0                    'set as virtual axis
TABLE(0,L1,L2,3600,3600) 'parameters are saved starting from TABLE(0),
                              a round of motor, there are 3600 pulses.
```

	UNITS=1000,1000 'pulse equivalent should be set in advance, and it can't be changed in the process. BASE(0,1) CONNFRAME(1,0,6,7) 'set reversed solution, coordinate of axis 0/1 calculate motion joint axis according to axis 6/7. WAIT LOADED BASE(6,7) MOVEABS(15,10) 'virtual axis sends motion instructions END Connect the simulation tool of robotic arm to view the running effect as shown below: 
Instructions	CONNREFRAME

CONNREFRAME –Forward Solution of Robotic Arm

Type	Synchronization Motion Instruction
Description	**Virtual axis coordinate correlates with joint axis coordinate, when joint axis moves, virtual axis will move to corresponding position.** This is the inversed motion instruction of CONNFRAME. ⚠ When virtual axis CONNREFRAME moved LOAD, joint axis CONNFRAME will be cancelled automatically by CANCEL. ⚠ When joint axis CONNFRAME moved LOAD, virtual axis CONNREFRAME will be cancelled automatically by CANCEL.
Grammar	CONNREFRAME(frame, tablenum, axis0, axis1,[…]) frame: coordinate type, 1-scara (if needs special defined robotic arm

	type, please contact with manufacturers) tablinum: TABLE position for saving conversion parameters. When frame=1, save one by one: the first joint axis length, the second joint axis length, the first joint axis pulse amount as per round, the second joint axis pulse amount as per round. viraxis0: the first axis of joint coordinate viraxis1: the second axis of joint coordinate […]: the N axis of joint coordinate The position of BASE axis is opposite to parameter axis.
Controller	General
Example	DIM L1,L2 L1=10 'the first joint axis length L2=10 'the second joint axis length BASE(0,1) 'suppose joint axis number is 0/1 UNITS=10,10 'pulse amount is 10 DPOS=0,0 'set joint axis position, modify it according to actual situation BASE(6,7) ATYPE=0,0 'set as virtual axis TABLE(0,L1,L2,3600,3600) 'parameters are saved starting from TABLE(0), a round of motor, there are 3600 pulses. UNITS=1000,1000 'pulse amount should be set in advance, and it can't be changed in the process. CONNREFRAME(1,0,0,1) 'coordinate of axis 6/7 calculate motion joint axis according to axis 0/1. BASE(0,1) MOVEABS(90,0) 'virtual coordinate is changed to 0,20.
Instructions	CONNFRAME

FRAME--Robotic Arm Type

Type	Robotic Arm Instruction
Description	**Choose robotic Type**, see *Robotic Arm Manual* for reference.
Instructions	CONNFRAME

FRAME_STATUS-Axis Status of Robot

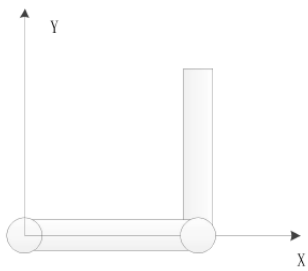
Type	Robotic Arm Instruction
Description	**Indicate current robotic arm attitude.** When the status is not robotic arm, it returns -1, FRAME_TRANS2 instruction will use this attitude. Several attitudes are only for SCARA, kind of SCARA and 6 DOF. SCARA left-hand status value is 0, right-hand status value is 1.

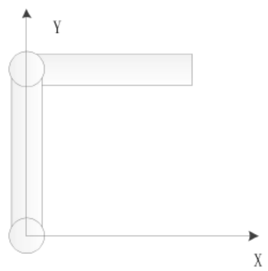
Grammar	VAR1=FRAME_STATUS（AXIS）
Controller	General
Example	Input online instruction ?FRAME_STATUS, and print the current status. >>?**FRAME_STATUS**
Instructions	BASE

FRAME_TRANS2-Coordinate Conversion of Forward and Inverse Solutions

<table>
<tr><td>Type</td><td>Robotic Calculation Instruction</td></tr>
<tr><td>Description</td><td>Coordinate transformation function.
It must be used when the forward or inverse is built.
The axis No. of BASE must be correct when using the instruction. For inverse solution, base as virtual axis, for forward solution, base as joint axis.
According to correct sequence, fill corresponding data. And data and number filled in should be the same as ?*frame.</td></tr>
<tr><td>Grammar</td><td>FRAME_TRANS2(tablein, tableout, dir)
tablein: index No. of table array, from this index, start to continuously save data. When in forward solution, input joint coordinate, when inverse solution, input virtual axis coordinate, at last, plus the status.
tableout: table, this index No. starts to save data. When in forward solution, output virtual coordinate and then plus status, when in inverse solution, output joint coordinate list.
Dir: mode selection
<table>
<tr><th>Mode</th><th>Type</th><th>Description</th></tr>
<tr><td>0</td><td>Inverse</td><td>From virtual axis to joint axis, no status, use current status automatically.</td></tr>
<tr><td>1</td><td>Forward</td><td>From joint axis to virtual axis, no status output.</td></tr>
<tr><td>2</td><td>Inverse</td><td>Input virtual axis coordinate, at last plus status.</td></tr>
<tr><td>3</td><td>Forward</td><td>Output virtual axis coordinate, when output the final position, fill in status.</td></tr>
</table>
</td></tr>
<tr><td>Controller</td><td>General</td></tr>
<tr><td>Example</td><td>Take scara structure as example, the first joint axis L1=10, the second joint axis L2=10. Table (100) as saved position of input coordinate, table (200) as saved position of output coordinate.
After linking, origin coordinate of joint axis is (0,0), and the virtual axis coordinate is (20,0), as below:</td></tr>
</table>

When virtual axis coordinate is (10,10), there are two statuses.
Joint axis (0,90) and joint axis (90,-90).



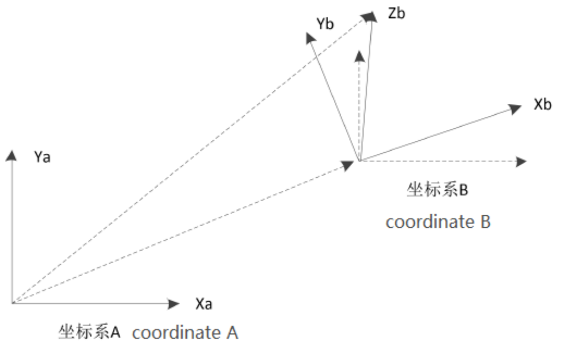


Coordinate transformation form:

Status	BASE axis	Input X,Y, status	Instruction	Output joint coordinate
Inverse	Virtual axis	table(100,20,0,0)	frame_trans 2 (100,200,0)	table(200,0,0)
		table(100,10,10,1)		table(200,0,90)
		table(100,10,10,1)	frame_trans 2 (100,200,2)	table(200,90,-90)
Status	**BASE axis**	**Input joint coordinate**	**Instruction**	**Output X,Y, status**
Forward	Joint axis	table(100,0,0)	frame_trans 2 (100,200,1)	table(200,20,0,0)
		table(100,0,90)		table(200,10,10,0)
		table(100,90,-90)		table(200,10,10,0)
		table(100,90,-90)	frame_trans 2 (100,200,3)	table(200,10,10,1)

FRAME_ROTATE-Workpiece Coordinate Conversion

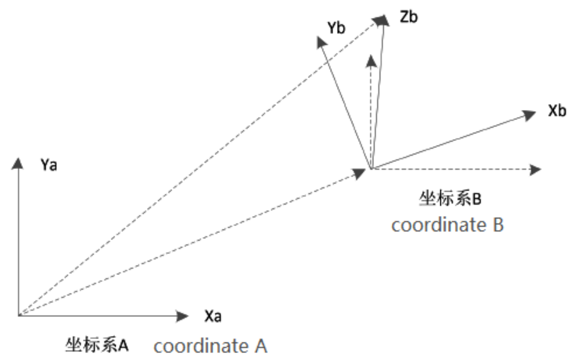
Type	Robot Calculation Instruction
Description	**Used to translate and rotate workpiece coordinate system.** At present, it only can rotate FRAME6 status at the same time, other virtual axes that has XYZ support 3-axis rotation, but status axis can’t rotate. After rotation, virtual axis WORLD_DPOS means world coordinate won’t

	change, virtual axis DPOS means workpiece axis will change. When using, there needs mechanical link of the controller. The axis of BASE can be either virtual axis or joint axis. If BASE axis has no robotic arm link, an error 1025 will appear. When there are several robotic arm superpositions, identify which is the robotic arm mode according to BASE axis. If in BASE (axis_1, axis_2), axis 1 is the robotic arm axis of mode1, axis 2 is the robotic arm axis of mode2, so calculating coordinate with robotic arm mode1, which means in BASE axis sequence.
Grammar	FRAME_ROTATE(X,Y,Z,RX,RY,RZ) X: translation distance of coordinate B towards X^ Y: translation distance of coordinate B towards Y^ Z: translation distance of coordinate B towards Z^ RX: rotation angle of coordinate B towards X^ RY: rotation angle of coordinate B towards Y^ RZ: rotation angle of coordinate B towards Z^  Rotation of coordinate system: The method: x_y_z coordinate system with fixed angle. At first, superpose coordinate {B} and coordinate {A}(known reference). {A} rotates RX angle about Xa, then rotates RY angle about Ya, at last, rotates RZ angle about Za.
Controller	General
Routine	**Example 1 FRAME=2, DELTA, rotates X axis 90 degrees.** BASE(0,1,2) RAPIDSTOP ATYPE = 1,1,1 UNITS=3600/360,3600/360,3600/360 DPOS=0,0,0 BASE(6,7,8) ATYPE = 0,0,0 TABLE(0,40,10,32,85,3600,3600,3600, 0, 0, 0) UNITS = 100,100,100 BASE(0,1,2) CONNFRAME(2,0,6,7,8) WAIT LOADED

	BASE(6,7,8) **FRAME_ROTATE**(0,0,0,PI/2,0,0) ?"DPOS(7)-WORLD_DPOS(7)=",DPOS(7)-WORLD_DPOS(7) ?"DPOS(8)-WORLD_DPOS(8)=",DPOS(8)-WORLD_DPOS(8) Output results: DPOS(7)-WORLD_DPOS(7)=-58.1400 DPOS(8)-WORLD_DPOS(8)=58.1400 **Example 2: FRAME=1, SCARA, rotates Z axis 90 degrees.** BASE(0,1,2,3) RAPIDSTOP ATYPE = 1,1,1,1 UNITS=3600/360,3600/360,3600/360,1000 DPOS=0,0,0,0 BASE(6,7,8,9) ATYPE = 0,0,0,0 TABLE(0,100,100,3600,3600,3600) UNITS = 100,100,3600/360,1000 BASE(0,1,2,3) CONNFRAME(1,0,6,7,8,9) WAIT LOADED BASE(6,7,8,9) **FRAME_ROTATE**(0,0,0,0,0,PI/2) ?"DPOS(7)-WORLD_DPOS(7)=",DPOS(7)-WORLD_DPOS(7) ?"DPOS(8)-WORLD_DPOS(8)=",DPOS(8)-WORLD_DPOS(8) Output result: DPOS(7)-WORLD_DPOS(7)=-200 DPOS(8)-WORLD_DPOS(8)=0
Instructions	FRAME_ROTATE2

FRAME_ROTATE2-Coordinate Conversion Calculation

Type	Robot Calculation Instruction
Description	**Calculate coordinate value after rotation by manually.** When using, there needs mechanical link of the controller. The axis of BASE can be either virtual axis or joint axis. If BASE axis has no robotic arm link, an error 1025 will appear. When there are several robotic arm superpositions, identify which is the robotic arm mode according to BASE axis. If in BASE (axis_1, axis_2), axis_1 is the robotic arm axis of mode1, axis_2 is the robotic arm axis of mode2, so calculating coordinate with robotic arm mode1, which means in BASE axis sequence.
Grammar	FRAME_ROTATE2(tablein, tableout, dir[, x,y,z[, rx,ry,rz]])

<table>
<tr><td></td><td>ret = FRAME_ROTATE2(tablein, tableout, dir[, x,y,z[, rx,ry,rz]])
tablein: before conversion, filled coordinate is saved in TABLE.
tableout: after conversion, output coordinate is saved in TABLE.
dir: direction selection
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>From DPOS to WORLD_DPOS</td></tr>
<tr><td>1</td><td>From WORLD_DPOS to DPOS</td></tr>
</table>
X: translation distance of coordinate B towards X^
Y: translation distance of coordinate B towards Y^
Z: translation distance of coordinate B towards Z^
RX: rotation angle of coordinate B towards X^
RY: rotation angle of coordinate B towards Y^
RZ: rotation angle of coordinate B towards Z^
[, x,y,z[, rx,ry,rz]]: use present “rotate” default value when it’s blank.
Ret: return successfully or not. -1: successfully   0: fail


</td></tr>
<tr><td>Controller</td><td>General</td></tr>
<tr><td>Example</td><td>Example1: FRAME=2, DETLA, rotates X axis at 90 degrees.
BASE(0,1,2)
RAPIDSTOP
ATYPE = 1,1,1
UNITS=3600/360,3600/360,3600/360
DPOS=0,0,0
BASE(6,7,8)
ATYPE = 0,0,0
TABLE(0,40,10,32,85,3600,3600,3600, 0, 0, 0)
UNITS = 100,100,100
BASE(0,1,2)
CONNFRAME(2,0,6,7,8)
WAIT LOADED
FOR i=0 TO 2
TABLE(100+i)=DPOS(i+6)
NEXT
BASE(6,7,8)
FRAME_ROTATE(0,0,0,PI/2,0,0)
BASE(6,7,8)
FRAME_ROTATE(0,0,0,PI/2,0,0)</td></tr>
</table>

	WAIT LOADED
	ret=**FRAME_ROTATE2**(100,200,1,0,0,0,PI/2,0,0)
	IF ret=-1 THEN
	?"calculate value "
	?"DPOS(6)=",TABLE(200)
	?"DPOS(7)=",TABLE(201)
	?"DPOS(8)=",TABLE(202)
	?"compare value "
	?"DPOS(6)compare",TABLE(200)-DPOS(6)
	?"DPOS(7)compare",TABLE(201)-DPOS(7)
	?"DPOS(8)compare",TABLE(202)-DPOS(8)
	ENDIF
	
	Output result：
	Calculate value
	DPOS(6)=0
	DPOS(7)=-58.1400
	DPOS(8)=0.0000
	Compare value
	DPOS(6)compare 0
	DPOS(7)compare 0
	DPOS(8)compare 0.0000
	
	Example 2: FRAME=1, SCARA, rotates Z axis at 90 degrees.
	BASE(0,1,2,3)
	RAPIDSTOP
	ATYPE = 1,1,1,1
	UNITS=3600/360,3600/360,3600/360,1000
	DPOS=0,0,0,0
	BASE(6,7,8,9)
	ATYPE = 0,0,0,0
	TABLE(0,100,100,3600,3600,3600)
	UNITS = 100,100,3600/360,1000
	BASE(0,1,2,3)
	CONNFRAME(1,0,6,7,8,9)
	WAIT LOADED
	FOR i=0 TO 3
	TABLE(100+i)=DPOS(i+6)
	NEXT
	BASE(6,7,8,9)
	FRAME_ROTATE(0,0,0,0,0,PI/2)
	WAIT LOADED
	RET=**FRAME_ROTATE2**(100,200,1,0,0,0,0,0,PI/2)
	IF RET=-1 THEN
	?"calculate value"
	?"DPOS(6)=",TABLE(200)

	?"DPOS(7)=",TABLE(201) ?"DPOS(8)=",TABLE(202) ?"DPOS(9)=",TABLE(203) ?"compare value" ?"DPOS(6)compare",TABLE(200)-DPOS(6) ?"DPOS(7)compare ",TABLE(201)-DPOS(7) ?"DPOS(8)compare",TABLE(202)-DPOS(8) ?"DPOS(9)compare",TABLE(203)-DPOS(9) ENDIF Output result： Calculate value DPOS(6)=-0.0000 DPOS(7)=-200 DPOS(8)=0 DPOS(9)=0 Compare value DPOS(6)compare -0.0000 DPOS(7)compare 0 DPOS(8)compare 0
Instructions	FRAME_ROTATE

WORLD_DPOS-World coordinate system

Type	Axis Parameter
Description	**Virtual axis coordinate value refers to world coordinate system, when there is no rotation, same as DPOS.**
Grammar	var1=WORLD_DPOS(axis)
Controller	General
Instructions	Online instruction, print. >>?*WORLD_DPOS

MOVER_L/MOVER_LABS-Joint Axis Linear Interpolation

Type	Motion Instruction
Description	**Joint axis linear interpolation.** Robot joint interpolation motion, the terminal of robotic arm moves to defined coordinate in linear direction. This is used under the forward solution mode, it may change the status if operating joint axis directly, so please ensure the attitude of starting point and ending point are the same, or will appear errors.
Grammar	MOVER_L(distance1 [,distance2 [,distance3 [,distance4...]]]) distance1: the first axis motion distance

	distance2: next axis motion distance
Controller	Valid in ZMC4XX series with firmware version above 20170511.
Routine	BASE(0,1) DPOS=0,0 BASE(6,7) ATYPE = 0,0 'set as virtual axis UNITS=1000,1000 TABLE(0,L1,L2, 100*360, 100*360, 360) CONNREFRAME(1,0,0,1) 'the 6/7 axis as virtual XY axis, open connect. WAIT LOADED 'joint motion BASE(0,1) SPEED=400 SRAMP=100 ACCEL=1000 DECEL=1000 MERGE = 1 CORNER_MODE=32 'start chamfering ZSMOOTH=2 MOVEABS(45,90) 'joint motion, which means motion joint angle **MOVER_LABS**(90, 0) 'terminal linear motion WAIT IDLE 'wait until motion stop PRINT *DPOS
Instructions	MOVER C, MOVER C3

MOVER_C/MOVER_CABS-Plane Circular of Joint Axis

<table>
<tr><td>Type</td><td>Motion Instruction</td></tr>
<tr><td>Description</td><td>Joint axis moves circular interpolation directly.
It is used under forward solution mode.
BASE axis should be virtual XYZ axes, or XYZ can’t be determined. And the parameters are distance of virtual axis.</td></tr>
<tr><td>Grammar</td><td>MOVER_C/MOVER_CABS
（end1,end2,centre1,centre2,mode,[dis1,…,disn]）
end1: motion distance parameter 1 of the first axis
end2: motion distance parameter 1 of the second axis
centre1: motion distance parameter 2 of the first axis
centre2: motion distance parameter 2 of the second axis
mode:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>The present point, the middle point and the end point, three points set the circular arc.
Distance parameter 1 is the end point distance, distance parameter 2 is the middle point distance.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td><table>
<tr><td>1</td><td>The present point, the center of circle and the end point set the circular arc.
Moves the shortest arc distance.
Distance parameter 1 is the end point distance, distance parameter 2 is the center of the circle distance.</td></tr>
<tr><td>2</td><td>The present point, the middle point and the end point, three points set the circle.
Distance parameter 1 is the end point distance, distance parameter 2 is the middle point distance.</td></tr>
<tr><td>3</td><td>The present point, the center of circle and the end point set the circle.
Moves the shortest arc distance at first, then continues to finish the full circle.
Distance parameter 1 is the end point distance, distance parameter 2 is the center of the circle distance.</td></tr>
</table>
dis1-disn: the distance of spiral axis</td></tr>
<tr><td>Controller</td><td>ZMC4XX series with firmware version 20170511 or above support.</td></tr>
<tr><td>Routine</td><td>L1 = 500
L2 = 500
TABLE(0,L1,L2,100*360,100*360,360) 'parameters are saved starting from TABLE0, a round of motor means 360 pulses
BASE(6,7)
CONNREFRAME(1,0,0,1)'the 6/7 axis as virtual XY axis, start to connect
WAIT LOADED 'wait for motion loading
BASE(6,7) 'REFRAME moves to virtual axis directly(MOVER), it will converse into joint axis automatically.
MOVER_LABS(500)
MOVER_C(500,0, 250,250, 0)</td></tr>
<tr><td>Instructions</td><td>MOVER_L, MOVER_C3</td></tr>
</table>

MOVER_C3/MOVER_C3ABS-Space Circular of Joint Axis

<table>
<tr><td>Type</td><td>Motion Instruction</td></tr>
<tr><td>Description</td><td>Joint axis moves space circular interpolation directly.
It is used under forward solution mode.
BASE axes should be virtual axes, or XYZ can’t be determined. And now the parameters are distance of virtual axes.</td></tr>
<tr><td>Grammar</td><td>MOVER_C3 (endx,endy,endz,midx, midy, midz, mode[, dis1][,dis2][,dis3])
end1: motion distance parameter 1 of the first axis
end2: motion distance parameter 1 of the second axis
end3: motion distance parameter 1 of the third axis
centre1: motion distance parameter 2 of the first axis
centre2: motion distance parameter 2 of the second axis</td></tr>
</table>

<table>
<tr><td></td><td>centre3: motion distance parameter 2 of the third axis
mode:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>The present point, the middle point and the end point, three points set the circular arc.
Distance parameter 1 is the end point distance, distance parameter 2 is the middle point distance.</td></tr>
<tr><td>1</td><td>The present point, the center of circle and the end point set the circular arc.
Moves the shortest arc distance.
Distance parameter 1 is the end point distance, distance parameter 2 is the center of the circle distance.</td></tr>
<tr><td>2</td><td>The present point, the middle point and the end point, three points set the circle.
Distance parameter 1 is the end point distance, distance parameter 2 is the middle point distance.</td></tr>
<tr><td>3</td><td>The present point, the center of circle and the end point set the circle.
Moves the shortest arc distance at first, then continues to finish the full circle.
Distance parameter 1 is the end point distance, distance parameter 2 is the center of the circle distance.</td></tr>
</table>
dis1- disn: distance of spiral axis</td></tr>
<tr><td>Controller</td><td>Valid in ZMC4XX series with firmware version 20170511 or above.</td></tr>
<tr><td>Routine</td><td>L1 = 500
L2 = 500
TABLE(0,L1,L2, 100*360, 100*360, 360)
'parameters are saved starting from TABLE0, a round of motor means 360 pulse amounts.
BASE(6,7)
CONNREFRAME(1,0,0,1) 'the 6/7 axis as virtual XY axis, start to connect
WAIT LOADED 'wait for motion loading
BASE(6,7,8) 'CONNREFRAM moves to virtual axis directly(MOVER), it will converse into joint axis automatically.
MOVER_LABS(400)
MOVER_C3ABS(200,0,0,600,400,0, 0)</td></tr>
<tr><td>Instructions</td><td>MOVER_L, MOVER_C</td></tr>
</table>

FRAME_CAL-Parameter Correction

Type	Robot Calculation Instruction
Description	**It corrects the present robotic arm parameter automatically according to coordinate and features of robotic arm teaching.**

	Captured robotic arm joint coordinates are saved in Tablein, when present origin point position has a robotic arm linking relation with arm parameters, the terminal point of control robotic arm moves to correction point, then get the joint axis coordinate of correction point. Correct deviation between present origin position and theorical origin position, then calculate joint axis coordinate of theorical origin point. Correct robotic arm parameter values (correct some parameters), then calculate theorical robotic parameter values. FRAME_CAL is only for calculation, if return value is -1, which means succeeds in calculating, if it's 0, means it fails. BASE axis of FRAME_CAL must be the axis under FRAME.
Grammar	FRAME_CAL(tablein,space,groups,tableaux, zeroout, [tableout2]) tablein: saved table starting number of joint coordinate, and each coordinate is saved in sequence, multiple points are separated by space. space: table element between every two points. groups: the number of point tableaux: table number of assistant parameters, some frame need. zeroout: calculated table number of joint axis absolute coordinate when in the theorical origin. tableout2: calculated store position of robotic arm parameter, it saves in the origin parameter position when it's blank.
Controller	General
Routine	See robotic arm instruction description manual chapter for details

Chapter VIII Program Structure and Process Instruction

8.1 Procedure Symbol

' --Add Comments

Type	Special Character
Description	**Followed contents are all explanation until next line.**
Controller	General

_--Change Line

Type	Special Character
Description	**Continue in next line.** Don’t use this instruction in condition judgement, storage, print output.
Controller	General

:--Label

Type	Grammar Structure
Description	**Make the label for user process, which can be used as SUB process without parameters.**
Grammar	Label: label name, but it can’t be same as existing words.
Controller	General
Example	GOTO label1 END 'main process ends. label1: 'add: define label END

8.2 Data Definition Instruction

CONST--Define Constant

Type	Grammar Instructions
Description	**Define a symbol to indicate constant value, avoid using value directly.**

Grammar	CONST CVARNAME = value CVARNAME: constant name value: constant value
Controller	General
Example	Example One **CONST** MAX_VALUE = 100000 'define constant TABLE(0)=MAX_VALUE 'assign 10000 to table(0) Example two GLOBAL **CONST** MAX_AXIS=6 'define total axes number
Instructions	DIM

DIM—Define Variables

Type	Grammar Instructions
Description	**Define file module variables, arrays.** If variables are not defined, then assign directly, file module variables will be defined automatically. File module variables only can be used inside this program file. Array can be used as the character string, one element means one byte.
Grammar	DIM varname, arrayname (space) varname: variables name arrayname: array name space: array space Valid in ZMC5XX series controllers with firmware after February 2022. 1. Variables definition initialization: DIM varname = 1 2. Array definition initialization: DIM arrayname(size) = {1, 2, 3} DIM arrayname(size) = “string” 3. Structure definition initialization: DIM strname(size) as structname = {.item = 1, .item = {1, 2, 3} } Initialized assignment value also can be used in other assignment commands, for example, GLOBAL.
Controller	General
Example	DIM ARRAY1(100) ‘define array ARRAY1 DIM VAR1 ‘define variable VAR1 VAR2 = 100 ‘assigned command will be defined as file module variables automatically. ARRAY1 = “asdf” ARRAY1(0, 100, 200, 300) ‘assign consecutively for array ‘ARRAY1(0) =100, ARRAY1(1) = 200, ARRAY1(2) =300

Instructions	CONST, LOCAL, GLOBAL

LOCAL—Define Local

Type	Grammar Instructions
Description	**Define local variables.** Local variables are usually used in SUB process. There is limit of local variables in one SUB process, parameters of SUB process will be converted to local variables automatically. When different tasks call the same SUB process, then it will generate different local variables in different tasks, when SUB recursive process of the same task is called, it will also generate different local variables.
Grammar	SUB subA() LOCAL localname 'localname local variable name …… ENDSUB
Controller	General
Example	SUB aaa() **LOCAL** v1 'define local variable V1 v1=100 END SUB
Instructions	DIM,GLOBAL

GLOBAL—Define Global

Type	Grammar Instructions
Description	**Define global variables, array. Define global SUB process.** Global variables can be used in any process file of the whole project.
Grammar	Grammar1: GLOBAL VAR1 Grammar2: GLOBAL SUB SUB1() Grammar3: GLOBAL CONST CVARNAME = value Parameters: VAR1 variable name SUB1 process name CVARNAME constant name value constant value Valid in ZMC5XX series controllers with firmware after February 2022. 4. Variables definition initialization: GLOBAL varname = 1 5. Array definition initialization: GLOBAL arrayname(size) = {1, 2, 3}

	GLOBAL arrayname(size) = “string” 6. Structure definition initialization: GLOBAL strname(size) as structname = {.item = 1, .item = {1, 2, 3} } Initialized assignment value also can be used in other assignment commands, for example, DIM.
Controller	General
Example	**GLOBAL** SUB g_sub2() 'define global process g_sub2, which can be used in any file. **GLOBAL** CONST g_convar = 100 'define global constant. **GLOBAL** g_var2 'define global variable g_var2
Instructions	RTC_DATE, LOCAL

8.3 Array Operation Instruction

DMINS--Insert Array Link List

Type	Grammar Instructions
Description	**Operation of array link list, when insert one element into one array, then present element and all later elements will move backward one space.** Be careful when operating long size array, especially TABLE.
Grammar	DMINS ArrayName(pos, size) arrayname: array name pos: array index size: amounts to be modified. Attention: pos+size<array
Controller	General
Example	DIM aa(6) 'define array aa FOR i=0 TO 4 'assign value:0,1,2,3,4 aa(i)=i NEXT ?*aa 'print all elements of array. **DMINS** aa(0) 'insert element 0, all behind elements will move backward one space aa(0) = 10 'assign value to aa(0) ?*aa 'print all array elements after insert operation.
Instructions	DMDEL , DMCPY

DMADD –Arrays Volume Increase

Type	Grammar Instructions
Description	**Add array elements value in batch.**

	Don’t modify over 500 elements once.
Grammar	DMADD ArrayName (pos, size, data) arrayname: array name pos: start index size: element number to be modified. Don’t exceed array size when adding pos. data: value to be added
Controller	General
Example	dim aaa(20) 'define a array with 20 elements. ?*aaa 'print, all is 0. DMADD aaa(10,5,2) 'starts from element 10, the value adds 2 when modifying 5 elements ?*aaa 'print, table(10) to table(14) is 2, the other is 0. DMADD aaa(10,5,2) 'starts from element 10, the value adds 2 when modifying 5 elements ?*aaa 'print, table(10) to table(14) is 4, the other is 0.
Instructions	DMINS, DMCPY

DMDEL--Delete Array Link List

Type	Grammar Instructions
Description	**Operation of array link list, when delete one element from one array, then present element and all behind parameters will move forward one space.** Be careful when operating long size array, especially TABLE.
Grammar	DMDEL ArrayName(pos) arrayname array name pos array index
Controller	General
Example	DIM aa(6) 'define array aa FOR i=0 TO 4 'assign value 0,1,2,3,4 aa(i)=i NEXT ?*aa 'print all array elements **DMDEL** aa(0) 'delete the first element of array. ?*aa 'print all array elements after delete operation.
Instructions	DMINS, DMCPY

DMCPY--Array Copy

Type	Grammar Instructions

Description	**Copy array, starting from array Src to array Des.** Be careful when operating long size array, especially TABLE.
Grammar	DMCPY ArrayDes(startpos) , ArraySrc(startpos)[, size] arrayname array name startpos array start index size elements number to copy, it will reduce automatically if exceeds maximum value.
Controller	General
Example	GLOBAL aa(6),bb(6) 'define array aa, bb FOR i=0 TO 4 'assign value 0, 1, 2, 3, 4 aa(i)=i NEXT ?*aa 'print all elements in array ?*bb **DMCPY** aa(0), bb(0),6 'assign value of bb to aa ?*aa 'print all array elements after copy operation ?*bb
Instructions	DMINS, DMDEL

DMSET- Array Assign

Type	Grammar Instructions
Description	**Assign array.** Be careful when operating long size array, especially TABLE.
Grammar	DMSET arrayname(pos, size, data) pos： start index size： length data: array to be set
Controller	General
Example	**DMSET** TABLE(0,10,2) 'assign value in the array part FOR i=0 TO 9 PRINT "TABLE",i, TABLE(i) 'print array NEXT **DMSET** TABLE(0,10,3) 'assign value in the array part FOR i=0 TO 9 PRINT "TABLE",i, TABLE(i) 'print array NEXT
Instructions	DMINS, DMDEL

DMCMP- Array Comparison

Type	Grammar Instructions
Description	**Array comparison, compare values of elements in array one by one,**

	then return results. Please cautious to oversize arrays operation, especially array TABLE.

Grammar	value = DMCMP(arr1, arr2, size)
	        arr1: array to be compared
	        arr2: array to be compared
	        size: the number of elements to compare, which can’t exceed the length of arr1 and arr2.
	
	Gained return values:
	        arra1 > arr2 value = 1
	        arra1 = arr2 value = 0, in comparison range, element values equal
	        arra1 < arr2 value = -1

Controller	General

Example	DIM value,i
	DIM arr3(5), arr5(6)
	
	FOR i = 0 TO 4
	        arr3(i) = i*10
	NEXT
	FOR i = 0 TO 5
	        arr5(i) = i*100+1
	NEXT
	
	value = **DMCMP**(arr3,arr5,5)
	?value
	IF value = -1 THEN
	        ?"less than"
	ELSEIF value = 1 Then
	        ?"more than"
	ELSE
	        ?"equal"
	END IF

Instructions	DMINS, DMDEL

DMCMP- Array Search

Type	Grammar Instructions
Description	**According to element value, search the position of this element in array, then return the value that indicates the first searched array index, if it can not be searched, it will return -1.** Please cautious to oversize arrays operation, especially array TABLE.

Grammar	Pos = DMSearch (array, startpos, offset, maxtimes, value)
	        array: array name
	        startpos: starting position of searching
	        offset: span that jumped in each search
	        maxtimes: max judged times

	value: searched value Return: Pos: index of array, -1 means no found.
Controller	General

Example	DIM ruturn, value
	DIM arr1(10)
	FOR i = 0 TO 9
	Arr1(i) = i
	NEXT
	value = **DMsearch**(arr1,0,1,10,3)
	ruturn = **DMsearch**(arr1,0,1,10,20)
	IF value = 3 AND ruturn = -1 THEN
	?"success"
	ELSE
	?"fail"
	END IF

Instructions	DMINS, DMDEL

SIZEOFARRAY – Get Array Space

Type	Grammar Instructions
Description	**Get occupied space.**
Grammar	VAR = SIZEOFARRAY (array name) return the number of arrays, variables are not supported. VAR = SIZEOFARRAY (structural name) return space occupied by structure. VAR = SIZEOFARRAY (structural variables name) return structural variables / arrays occupied space Valid in 5xx series controllers with firmware version above 20180327 Valid in 4xx series controllers of fast version with firmware version above 20190107.
Controller	General
Example	Example 1: the number of returned arrays GLOBAL aa(12),bb 'define array aa, bb FOR i=0 TO 4 'assign aa as 0,1,2,3,4 aa(i)=i NEXT ?*aa 'print all elements of array ?SIZEOFARRAY(aa) 'print result : 12 Example 2: the number of returned structural variables / arrays 'statement structure AA GLOBAL Structure ClassAA DIM AA_val1 'member variables

	DIM AA_array(10) 'member arrays END Structure 'build structure variables GLOBAL Class1 AS ClassAA Class1.AA_val1=123 ?Class1.AA_val1 'print result: 123 class1.AA_array="abc" ?class1.AA_array 'print result: abc ?SIZEOFARRAY(class1) 'print result:11 ?SIZEOFARRAY(classAA) 'print result: 11 ?SIZEOFARRAY(class1.AA_array) 'print result: 10 ?SIZEOFARRAY(Class1.AA_val1) 'print result: 1
Instructions	DMINS, DMDEL

8.4 Self-defined Sub Function Instruction

SUB--Self-defined Subfunction SUB

Type	Grammar Instructions
Description	**Users custom SUB process, GLOBAL description can be added before to define SUB process for global use.**
Grammar	SUB label([para1] [,para2]…) … END SUB Parameters label: process name, it can't be same as current key words. para1: transferred parameters when calling SUB, and it is changed into local variables automatically. para2: transferred parameters when calling SUB, and it is changed into local variables automatically. Valid in 5xx series controllers, and the firmware version after February 2022 added this function. SUB subname(BYREF paraname[(dimsize)] [AS structname]) subname: sub name dimsize: the length of the array, must be defined as a constant structname: the name of the structure type, supporting BYREF to transfer ZVOBJ BYREF represents a quote, at this time, for calling method, please fill in variables, arrays or others of corresponding types. The default BYVAL means transfer by copy, and BYVAL does not support

	arrays, structures, etc. temporarily. The array defined by BYREF cannot use {} to assign multiple elements, and does not support the original array multiple element assignment method. The data passed by BYREF can no longer use the ZINDEX index function. In principle, it is not recommended to use ZINDEX for LOCAL data
Controller	General

Example	**Example 1:**
	SUB sub1() 'define process SUB1, which is only used in present file.
	    ?1
	    ...
	END SUB
	
	GLOBAL SUB g_sub2() 'define global SUB g_sub2, it is used in any file.
	    ?2
	    ...
	END SUB
	
	GLOBAL SUB g_sub3(para1,para2) 'define global SUB g_sub3, and transfer 2 parameters.
	    ?Para1,para2
	    ...
	  RETURN para1+para2 'function returns, parameters are combined
	END SUB
	Example 2: valid in 5xx series controllers or above
	STRUCTURE POS
	    DIM a
	    DIM b(11)
	END STRUCTURE
	
	DIM var1
	DIM arr2(11)
	DIM arr3(300)
	DIM str3(2) as pos
	SUB2(var1, arr2, str3) 'mode 1
	SUB2(var1, arr2(100, 200), str3) 'mode 2: get the middle part of the array through arr2(100, 200)
	
	SUB SUB2(byref var1, byref arr2(100), byref arr3(2) as pos)
	    ?SUB_IFPARA(0)
	    ?var1
	    ?arr2(1)
	    ?arr3(1).a
	END SUB
	
	SUB SUB3(byref var1, byref obj1 as ZVOBJ)
	    'support BYREF to pass ZVOBJ
	    ?SUB_IFPARA(0)
	    ?var1

	?arr2(1) ?arr3(1).a END SUB
Instructions	SUB_PARA, SUB_IFPARA

SUB_PARA—SUB Transfers Parameters

Type	Grammar Instructions
Description	**Choose input parameters of SUB.**
Grammar	SUB_PARA(address) address: NO. of input parameters, starts from 0.
Controller	General
Example	SUB AAA(NUM1,NUM2,NUM3) ?**SUB_PARA(0)** 'print the first parameter num1 when calls AAA ?**SUB_PARA(1)** 'print the second parameter num2 ?**SUB_PARA(2)** 'print the third parameter num3 END SUB
Instructions	SUB,SUB_IFPARA

SUB_IFPARA --Judgement of SUB Input Parameters

Type	Grammar Instructions
Description	**Judge if SUB parameters were input.**
Grammar	SUB_IFPARA(address) -1: already input, 0: -not input address: NO. of input parameters, starts from 0.
Controller	General

Example	AAA(0,100) 'input num1,num2
	AAA(,100) 'only input num2
	END
	SUB AAA(NUM1,NUM2)
	IF **SUB_IFPARA(0)** THEN 'check if num1 was input when calls AAA
	?1 'input and print 1
	ELSE
	?0
	ENDIF
	END SUB

Instructions	SUB, SUB_PARA

GOSUB/CALL—SUB Calling

Type	Procedure Structure
Description	**Call SUB process, which is only valid for SUB process in present file or SUB process defined as global.** When call SUB process directly, GOSUB can be omitted. If there are no parameters in SUB process, “()”in SUB can be omitted After using GOSUB, the present content will be pushed onto stack, which means the present local variables can not be accessed in called SUB process. Contents will pop from stack when RETURN.
Grammar	GOSUB/CALL label label: SUB name
Controller	General
Example	'Main process main: GOSUB sub1() sub2(1,2) 'transfer 1 to para1, transfer 2 to para2. call sub3 END 'defined SUB SUB sub1() a=100 PRINT "sub1" RETURN SUB sub2(para1,para2) a=200 PRINT "sub2",para1,para2 RETURN GLOBAL SUB sub3() 'It can be called in another procedure file a=300 PRINT"sub3" RETURN

GSUB--Self-defined Subfunction-G Code

Type	Grammar Instructions
Description	**Users customize GSUB process.** GLOBAL description can be added before to define global use GUSB process. When call GSUB, it will follow G code grammar, no need to add ().
Grammar	GSUB label([char1] [,char2]…) …

	END SUB Parameters label: process name, which can not be same as some key words. char1: input parameters when call SUB, which is changed into local variables automatically. char2: input parameters when call SUB, which is changed into local variables automatically. Alphabet Parameters can only be as single character
Controller	General
Example	G01 X100 Y100 Z100 U100 'call G01 END 'main process ends. GLOBAL GSUB G01(X, Y, Z, U) 'define GSUB process G01 ... END SUB
Instructions	GSUB_PARA, GSUB_IFPARA

GSUB_PARA--Input Parameters of GSUB

Type	Grammar Instructions
Description	**Choose input parameters of GSUB.**
Grammar	GSUB_PARA(char) char: input alphabet parameter when define GSUB
Controller	General
Example	GSUB AAA(X,Y,Z) ?**GSUB_PARA(X)** 'print the first parameter X when calls AAA ?**GSUB_PARA(Y)** 'print the second parameter Y ?**GSUB_PARA(Z)** 'print the third parameter Z END SUB
Instructions	GSUB, GSUB_IFPARA

GSUB_IFPARA-- Judgement of GSUB Input Parameters

Type	Grammar Instructions
Description	**Judge if GSUB parameters were input.**
Grammar	GSUB_IFPARA(char) -1-already input 0-not input char: input alphabet parameter when define GSUB
Controller	General
Example	AAA X0 Y100 'input X,Y AAA X0 'only input X

	END
	
	GSUB AAA(X,Y)
	    IF **GSUB_IFPARA(Y)** THEN 'check if Y was input when calls AAA
	        ?1 'if Y was input, print 1.
	    ELSE
	        ?0
	    ENDIF
	END SUB

Instructions	GSUB, GSUB_PARA

END SUB--End of Self-defined Function

Type	Procedure Structure
Description	**Customized SUB process ends, see SUB for reference.**
Controller	General

RETURN--Function Value Return

Type	Procedure Structure
Description	**It is used for users' SUB process return or return value.** Default returned value is 0. Externally, read returned value in former SUB process through RETURN . Different tasks will return different values.
Grammar	RETURN
Controller	General
Example	CALL sub1 ?**RETUEN** 'result is 111 END 'main procedure ends SUB sub1() RETURN 111 'return 111 END SUB

XSUB – Custom XSUB Sub-Function

Type	Procedure Structure
Description	**XSUB is the process customized by users to transfer parameters into subfunction.** There is one difference between RSUB and XSUB, XSUB needs to add the brackets when calling.
Grammar	Grammar is the same as SUB. When calling, it can use the grammar of paraname = value.

Controller	General

Example	**Example 1:**
	subX(ARR2=a2, VAR1 =a1, ARR3=pos4)
	XSUB subX(byref var1, byref arr2(100), byref arr3(2) as pos)
	    ?SUB_IFPARA(0)
	    ?var1
	    ?arr2(1)
	    ?arr3(1).a
	END SUB
	
	Example 2:
	TR VAR1 = 5,VAR2 = 1
	dim a
	a = 10
	TX(var1 = a ,arr = "Zmotion")
	
	RSUB TR(VAR1,VAR2)
	    ?SUB_IFPARA(0)
	    ?var1
	    ?var2
	END SUB
	
	XSUB TX(byref var1,byref arr(100))
	    ?var1
	    ?arr
	end sub

Instructions	SUB, RSUB

RSUB – Custom RSUB Sub-Function

Type	Procedure Structure
Description	**RSUB is the process customized by users to transfer parameters into subfunction.** There is one difference between RSUB and XSUB, RSUB doesn't need to add the brackets when calling.
Grammar	Grammar is the same as SUB. When calling, it can use the grammar of paraname = value.
Controller	General

Example	subR ARR2=a2, VAR1 =a1, ARR3=pos4
	RSUB subR(byref var1, byref arr2(100), byref arr3(2) as pos)
	    ?SUB_IFPARA(0)
	    ?var1
	    ?arr2(1)
	    ?arr3(1).a
	END SUB

Instructions	SUB, XSUB

8.5 Structural Definition Instruction

STRUCTURE-Definition of Structural Body

Type	Grammar instruction
Description	**Definition of structural body.** With firmware version above 5 xxx serials of 20180327 support. With firmware fast version above 4 xxx serials of 20190107 support.

Grammar	Structure name of structure
	    Dim: member 1 name [As data type1]
	    ... ...
	    Dim: member n name [(array length)][As data type 1]
	End Structure
	
	Data type only supports structural body. Every element has the same array element and occupies one array element space.
	Structure is not recursive.
	
	Structure variables definition:
	DIM: variables name AS structure name
	DIM: structure array name [(array length)] AS structure name
	
	GLOBAL: variables name AS structure name
	GLOBAL: structure array name [(array length)] AS structure name
	
	The reserved function:
	LOCAL variables name AS structure name
	
	Support use FLASH_WRITE，FLASH_READ to read and write variables and arrays of structure definition.
	FLASH_WRITE id, structure variables
	FLASH_WRITE id, structure array
	FLASH_WRITE id, structure array(index)
	FLASH_WRITE id, structure array(index).item
	FLASH_WRITE id, structure array(index).item array(index)
	FLASH_READ same as former.
	
	Support use array operation instructions to operate arrays of structural body.
	DMINS structure array(index) [,numes]
	DMINS structure array(index).item array(index) [,numes]
	DMDEL same as former.
	
	DMCPY structure array 1(index1), structure array 2 (index2) [,size]
	
	DMSET only supports operate the last level arrays, it can’t assign structural

	array. DMSET structure variables. item array(index, size, data) DMADD: same as former.
Controller	General
Example	'declaration structural body AA GLOBAL Structure ClassAA DIM AA_val1 'member variables DIM AA_array(10) 'member array END Structure 'declaration structural body BB GLOBAL Structure ClassBB DIM BB_val1 AS ClassAA 'member variables are structural body END Structure 'build structural body variables GLOBAL Class1 AS ClassAA GLOBAL Class2 AS ClassBB Class1.AA_val1=123 ?Class1.AA_val1 class1.AA_array="abc" ?class1.AA_array Class2.BB_val1.AA_val1=567 ?Class2.BB_val1.AA_val1 Class2.BB_val1.AA_array="zxc" ?Class2.BB_val1.AA_array AA_val1=8 FLASH_WRITE 0,AA_val1 AA_val1=123 FLASH_READ 0,AA_val1 ?AA_val1 END
Instructions	DIM, GLOBAL, UNION

UNION-Definition of Community

Type	Grammar instruction
Description	**Definition of community.** With firmware version above 5 xxx serials of 20180327 support. With firmware fast version above 4 xxx serials of 20190107 support.
Grammar	UNION structure name Dim: member 1 name[As data type1]

	… … Dim: member n name[(array length)][As data type1] End UNION Structural variables definition： DIM: variables name AS structure name DIM: structure array name[(array length)] AS structure name GLOBAL: variables name AS structure name GLOBAL: structure array name[(array length)] AS structure name The reserved function： LOCAL: variables name AS structure name Support use FLASH_WRITE，FLASH_READ to read and write variables and arrays of structure definition. FLASH_WRITE id, structure variables FLASH_WRITE id, structure array FLASH_WRITE id, structure array(index) FLASH_WRITE id, structure array(index).item FLASH_WRITE id, structure array(index).item array(index) FLASH_READ same as former. Support use array operation instructions to operate arrays of structural body. DMINS structure array(index) [,numes] DMINS structure array(index).item array(index) [,numes] DMDEL: same as former. DMCPY structure array 1(index1), structure array 2 (index2) [,size] DMSET only supports operate the last level arrays, can't assign structural array. DMSET structure variables. item array(index, size, data) DMADD: same as former.
Controller	General
Example	Please refer to STRUCTURE for examples.
Instructions	DIM, GLOBAL, STRUCTURE

8.6 Jump Instruction

GOTO--Forced Jump

Type	Procedure Structure
Description	**Force to jump, the difference from GOSUB is that process called by GOTO will not be pushed onto stack.**
Grammar	GOTO label

Controller	General
Example	a=100 **GOTO** label1 'force to jump to label1 a=1000 END 'main procedure ends label1: PRINT a 'result is a=100 END 'label1 ends

ON GOSUB--Condition Jump

Type	Procedure Structure
Description	**When expression is true, then call label process.**
Grammar	ON expression GOSUB label expression: judgement condition label: jump to sub or label
Controller	General
Example	a=100 **ON** a>10 **GOSUB** label1 'when a>10, call label process. a=1000 PRINT a END 'main procedure ends label1: PRINT a RETURN 'process will return

ON GOTO-- Condition Jump 2

Type	Procedure Structure
Description	**Condition jump, procedure jump when expression is true, called process will not be pushed onto stack.**
Grammar	ON expression GOTO label expression: judgement condition label: jump to sub or label
Controller	General
Example	a=100 **on** a>10 **goto** label1 a=1000 END 'main procedure ends label1: PRINT a END 'can not return when use goto jump.

8.7 Condition Judgement Instruction

IF--Condition Judgement Structure

Type	Procedure Structure
Description	**Condition Judgement, its structure same as standard BASIC grammar.**

Grammar	IF <condition1> THEN
	commands
	ELSEIF <condition2> THEN
	commands
	ELSE
	commands
	ENDIF
	Parameters
	condition1 condition
	condition2 condition

Controller	General
Example	Example one： DIM a 'define variable a=12 'assign value **IF** a>11 then 'judgement condition TRACE "the val a is bigger then 11" ELSELF a<11 then TRACE "the val a is less then 11" ENDIF Example two： IF IN (0) THEN OUT(0,ON) 'if there is only one line, no need of endif.
Instructions	THEN， ENDIF

THEN--Condition Judgement Structure

Type	Procedure Structure
Description	**See：IF**
Controller	General

ENDIF--Condition Judgement Structure

Type	Procedure Structure
Description	**See：IF**

Controller	General

ELSEIF--Condition Judgement Structure

Type	Procedure Structure
Description	**See：IF**
Controller	General

8.8 Cycle Instruction

FOR – “for” Cycle

Type	Procedure Structure
Description	**“Loop”, it uses standard BASIC grammar.**

Grammar	FOR variable=start TO end [STEP increment]
	    commands
	NEXT variable
	Parameters：
	    variable: variable name
	    start: starting cycle value
	    end: end cycle value
	    increment: incremental value of cycle, it is selectable.
	Please don’t use same “variable” (when it is not local) in multi-task, otherwise, they will bother each other.

Controller	General
Example	**Example 1:** LOCAL a **FOR** a=1 to 100 'cycle from 1 to 100 PRINT a 'print a NEXT **Example 2:** DIM i FOR i = 0 TO 50 STEP 2 ‘cycle from 1 to 50, the space is 2 TABLE(i) = i ?TABLE(i) NEXT
Instructions	TO,STEP,NEXT

TO—for Cycle Structure

Type	Procedure Structure
Description	**See：FOR**
Controller	General

STEP--For Cycle Structure

Type	Procedure Structure
Description	**See：FOR**
Controller	General

NEXT--For Cycle Structure

Type	Procedure Structure
Description	**See：FOR**
Controller	General

WHILE--while Cycle Structure

Type	Procedure Structure
Description	**Execute cycle when condition is met.**
Grammar	WHILE condition … WEND
Controller	General

Example	a=0
	WHILE IN(4)=OFF 'exit cycle until input 4 is ON.
	    a=a+1
	    PRINT a
	    DELAY(1000)
	WEND

WEND--While Cycle

Type	Procedure Structure
Description	**see：WHILE**
Controller	General

EXIT--Exit Cycle

Type	Procedure Structure
Description	**Exit cycle sentence.**
Grammar	EXIT FOR, EXIT WHILE
Controller	General
Example	LOCAL a FOR a=1 TO 100 'cycle from 1 to 100 PRINT a IF a> 20 THEN **EXIT** FOR 'must use the method, or IF doesn't match with ENDIF NEXT

REPEAT--Condition Cycle

Type	Procedure Structure
Description	**Cycle sentence.** Execute commands by cycle, exit cycle when condition is true.
Grammar	REPEAT commands UNTIL condition
Controller	General
Example	a=0 **REPEAT** 'execute followed sentences by cycle PRINT a a=a+1 DELAY(1000) UNTIL IN(4)=ON 'valid until input 4 is on

UNTIL--Condition Structure

Type	Procedure Structure
Description	**See: REPEAT, WAIT**
Controller	General

8.9 Wait Execution Instruction

DELAY--Time Delay

Type	Grammar Instructions
Description	**delay delay time, unit is ms.** Other name: wa

Grammar	DELAY (delay time) Delay time: the number of ms
Controller	General
Example	**DELAY**(100) 'delay 100ms

WAIT UNTIL--Wait for Meeting Condition

Type	Procedure Structure
Description	**Wait until condition is met.**
Grammar	WAIT UNTIL condition1 [and codition2 or codition3 ...] Use logic operation to operate multi conditions.
Controller	General
Example	Example 1 WAIT UNTIL DPOS(0) > 0 'wait until position of axis 0 exceeds 0. Example 2 used with TICKS TICKS=2000 'set ticks as 2000 **WAIT UNTIL** TICKS<0 'wait 2 seconds ?"execute next step" Example 3 used with logic conditions **WAIT UNTIL** IDLE(0)=-1 AND IDLE(1)=-1 AND IDLE(2)=-1 'wait until axis 0,1,2 stop.

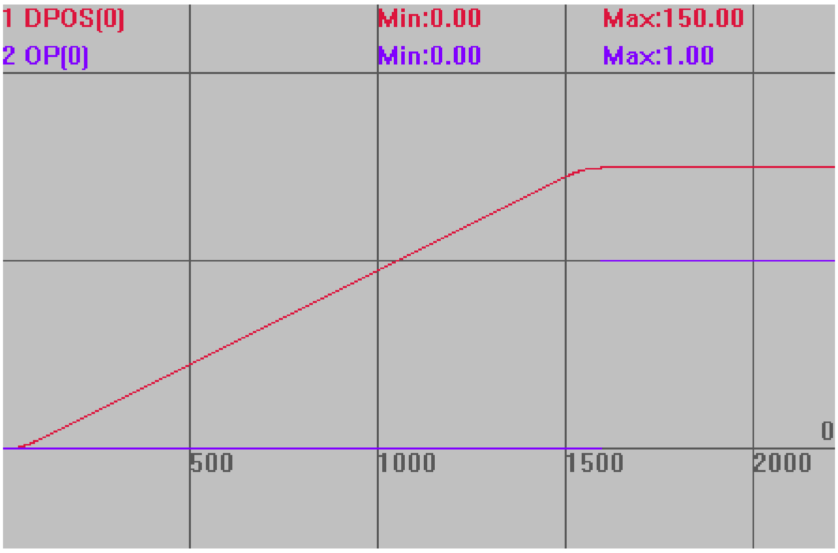
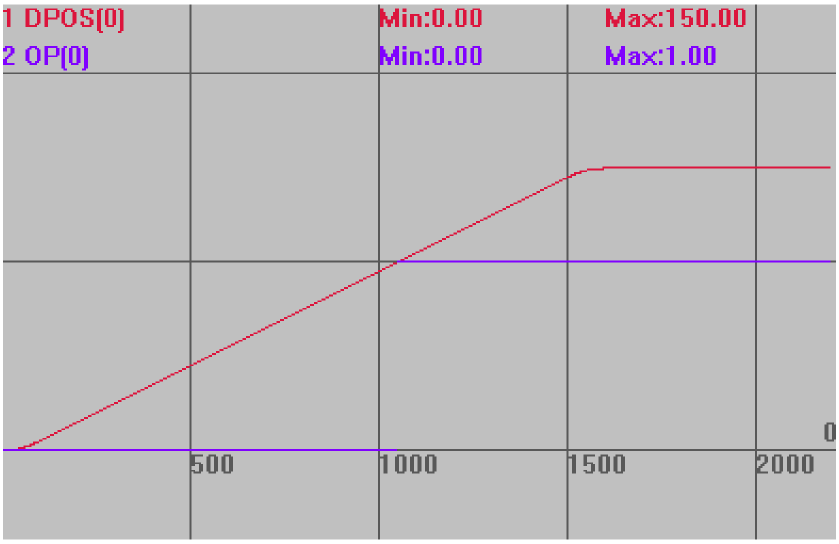
WAIT IDLE--Wait Until Axes Stop

Type	Grammar Instructions
Description	**Wait axis or axes of BASE to stop, when BASE axis / axes don't finish, following program will not be executed.** Same as WAIT UNTIL IDLE. IDLE is axis parameter, it supports grammar of axis parameters. Note: controller succeeds in sending motion that doesn't represent servo executed.
Grammar	WAIT IDLE
Controller	General
Example	**Example 1:** BASE(0,1) MOVE(100,100) **WAIT IDLE** 'wait until present interpolation motion end **Example 2:** BASE(0,1) MOVE(100,100) BASE(2,3)

	MOVE(200,200) **WAIT UNTIL** **IDLE**(0) AND **IDLE**(1) AND **IDLE** (2) AND **IDLE**(3) 'wait until motion axis 0,1,2,3 end. ?"motion finished"

WAIT LOADED--Wait Until Axes Buffer Clears

Type	Grammar Instructions
Description	**Wait until axes buffer clears, this instruction will block and won't execute followed procedures.** The last motion in buffer can be executed correctly, followed procedures continue to execute at the same time. Same as WAIT UNTIL LOADED, LOADED is axis parameter, it supports grammar of axis parameters.
Grammar	WAIT LOADED
Controller	General
Example	Difference from WAIT IDLE BASE(0) ATYPE=1 UNITS=100 DPOS=0 SPEED=100 ACCEL=1000 MERGE=1 TRIGGER MOVE(100) 'motion in process MOVE(50) 'motion in buffer, there is only one motion in buffer 'when the motion was executed, buffer has been cleared. WAIT IDLE 'when use wait idle, followed procedures will be executed until all motions are finished. OP(0,ON) 'open op0 Motion Path: DPOS(0) vertical scale 100 OP(0) vertical scale 1

	 **WAIT LOADED** 'when use wait loaded, followed procedures will execute once buffer clears. OP(0,ON) 

8.10. ZINDEX Pointer Instructions

ZINDEX_LABEL – Build Pointer Index

Type	Grammar Instructions.
Description	**Build pointer index, then it is convenient for behind to call pointer.**
Grammar	Pointer = zindex_label (subname) subname: array or SUB name
Controller	General
Example	DIM arr1(100) ‘define array

	Arr1(0,1) 'assign 1 to array address 0 Pointer = ZINDEX_LABEL (arr1) 'build index pointer PRINT ZINDEX_ARRAY (Pointer) (0) 'access array, print the first bit data of array, the result is 1
Instructions	ZINDEX_CALL, ZINDEX_ARRAY, ZINDEX_VAR

ZINDEX_CALL – Access SUB Function

Type	Grammar Instructions.
Description	**Call SUB function through index pointer.**
Grammar	ZINDEX_CALL (zidnex) (subpara, …) zidnex: index pointer generated from ZINDEX_LABEL subpara: sub parameters calling
Controller	General
Example	Pointer = ZINDEX_LABEL (sub1) 'build index pointer ZINDEX_CALL (Pointer) (2) 'call function SUB sub1 (a) PRINT a END SUB
Instructions	ZINDEX_LABEL

ZINDEX_ARRAY – Access Array

Type	Grammar Instructions.
Description	**Access array through index pointer.**
Grammar	var = ZINDEX_ARRAY (Pointer) (index) pointer: pointer index generated from ZINDEX_LABEL index: array index
Controller	General
Example	DIM arr1(100) 'define array Arr1(0,1) 'assign 1 to array address 0 Pointer = ZINDEX_LABEL (arr1) 'build index pointer PRINT ZINDEX_ARRAY (Pointer) (0) 'access array, print the first bit data of array, the result is 1
Instructions	ZINDEX_LABEL

ZINDEX_VAR – Access Variables

Type	Grammar Instructions.
Description	**Access variables through index pointer.**
Grammar	ZINDEX_VAR (zindex)

	zidnex: index pointer generated from ZINDEX_LABEL zindex= ZINDEX_LABEL(varname) ZINDEX_VAR(zindex)=value VAR2 = ZINDEX_VAR(zindex)
Controller	General
Example	global gTestVar global VarAdd1 VarAdd1=ZINDEX_LABEL(gTestVar) ZINDEX_VAR(VarAdd1)=10 ?ZINDEX_VAR(VarAdd1)
Instructions	ZINDEX_LABEL

ZINDEX_STRUCT – Access Structure

Type	Grammar Instructions.
Description	**Access structural variables or arrays through pointer after getting the pointer of structural variables.**
Grammar	GLOBAL structarrname(num) As structname zindex = ZINDEX_LABEL(structarrname) ZINDEX_STRUCT(structname,zindex)(arrindex).item = var var = ZINDEX_STRUCT(structname,zindex)(arrindex).item structarrname: generated structural arrays, variables. num: the number of generated structural arrays and variables elements. zidnex: generated index pointer through ZINDEX_LABEL arrindex: structural array subscript structname: structure name item: structure memeber
Controller	The structural pointer function is only valid in controllers with special firmware version. Valid in ZMC5XX series controllers with firmware above 20180327. Valid in ZMC4XX series controllers with fast version and firmware version above 20190107.
Example	**Example 1:** GLOBAL Structure ClassAA 'structure statement DIM AA_val1 'member variables DIM AA_array(10) 'member arrays END Structure GLOBAL Class1 AS ClassAA 'structure global variables definition GLOBAL gStructureAdd Class1.AA_array(0,1,2,3) 'assign structure arrays ?Class1.AA_array(0) 'result: 1 gStructureAdd = ZINDEX_LABEL(Class1) 'build structure index pointer

	?ZINDEX_STRUCT(ClassAA,gStructureAdd).AA_array(0) 'result: 1 ZINDEX_STRUCT(ClassAA,gStructureAdd).AA_array(0)= 10 ?ZINDEX_STRUCT(ClassAA,gStructureAdd).AA_array(0) 'result: 10 END **Example 2:** GLOBAL STRUCTURE stru_node 'define structure DIM m_data DIM m_Left DIM m_right DIM m_Temp END STRUCTURE DIM root GLOBAL g_node(100) AS stru_node root = ZINDEX_LABEL(g_node) 'build structure array pointer ZINDEX_STRUCT(stru_node,root)(99).m_data = 11 ?ZINDEX_STRUCT(stru_node,root)(99).m_data 'result: 11 END
Instructions	ZINDEX_LABEL

Chapter IX Instructions Related to Task

ZBASIC supports real-time multi tasks run, one file can run multi tasks at the same time. It can start to run task from the first line through RUN, and can assign any SUB process start to run through RUNTASK.

9.1 Task Start and Stop Instruction

RUN--Start File Task

Type	Task Instructions
Description	**Start a new task to execute a file on controller.** Restart the same task that will report error. When use RUN instruction without task number parameters frequently, one file will be matched with multi tasks. It is recommended to use RUNTASK instruction to start task. Multi-task running instructions： END: Present task ends normally. STOP: Stop assigned files. STOPTASK Stop assigned tasks HALT: Stop all tasks. RUN Start file as new task. RUNTASK Start task that executes on one SUB
Grammar	RUN "filename"[, tasknum] filename: procedure file name, no need to add extension name on bas file tasknum: Task NO., find first valid task NO. in default mode.
Controller	General
Example	**RUN** "aaa", 1 'start task 1 to run aaa.bas file
Instructions	RUNTASK

RUNTASK--Start SUB TASK

Type	Task Instructions
Description	**Make a sub process or a label process as one new task to execute** Restart the same task that will report error.
Grammar	RUNTASK tasknum, label tasknum: Task No. label: self-defined SUB process (it can attch parameters) or label

Controller	General
Example	**RUNTASK** 1, taska 'open task 1 to trace and print position. MOVE(1000,100) MOVE(1000,100) END taska: 'print position in cycle WHILE 1 PRINT*mpos DELAY(1000) WEND END
Instructions	RUN

END--End Task

Type	Task Instructions
Description	**End Present Task.** If there are main process and SUB process in one file, do add END at the end of main process, or the procedure will continue to execute followed SUB process after main process is finished, it will end until meeting subprogram END SUB.
Controller	General
Instructions	RUN, RUNTASK

STOP--Stop File Task

Type	Task Instructions
Description	**Force to stop program, and operate file.** Do stop the tasks before restart tasks. When use STOP instruction without task number, it only stops one task in a time, not all tasks of the file. When there are multiple tasks in one file, it is recommended to use STOPTASK to stop tasks.
Grammar	STOP program name, [tasknum] program name: procedure file name, no need to add extension name for bas file. tasknum: Task NO., when procedure file starts multi tasks, the default task number is the minimal task.
Controller	General
Example	RUN aaa, 1 'execute aaa.bas **STOP** aaa, 1 'stop task 1
Instructions	STOPTASK, HALT

STOPTASK--Stop SUB Task

Type	Task Instructions
Description	**Force to stop task. Operate SUB and Label.** Do stop the tasks before restart tasks.
Grammar	STOPTASK [tasknum] tasknum: task NO., default value is present task NO.
Controller	General
Example	**STOPTASK** 2 'stop task 2
Instructions	STOP, HALT

HALT--Stop All Tasks

Type	Task Instructions
Description	**Stop all tasks.** This instruction is only used to PC software calling. There will cause whole process stop if uses it in BASIC, the controller can't work.
Grammar	HALT
Controller	General
Example	**HALT** 'stop all tasks
Instructions	STOP, STOPTASK

PAUSE--Pause All Tasks

Type	Task Instructions
Description	**Pause all tasks.** It is usually used in PC, if breakpoint is built successfully, tasks will also enter pause status. This instruction is only used to PC software callings, there will cause whole process stop, the controller can't work, if uses it in BASIC. Task will continue when it resumes after pause.
Grammar	PAUSE
Controller	General
Example	**PAUSE** 'pause all tasks.
Instructions	PAUSETASK

PAUSETASK--Pause Assigned Tasks

Type	Task Instructions
Description	**Pause one specific task.**

	Task will continue when it resumes after pause.
Grammar	PAUSETASK tasknum tasknum: task NO., default value is present task NO.
Controller	General
Example	**PAUSETASK** 1 'Pause task 1.
Instructions	RESUMETASK

RESUMETASK--Resume Assigned Tasks

Type	Task Instructions
Description	**Resume a specific task.** Task will continue when it resumes after pause.
Grammar	RESUMETASK tasknum tasknum: task NO., default value is present task NO.
Controller	General
Example	PAUSETASK 1 'pause task 1. **RESUMETASK** 1 'continue to run task 1.
Instructions	PAUSETASK

9.2 Three-file Task Instruction

FILE3_RUN--Execute FILE3 Task

Type	Task Instructions
Description	**Start Three-file procedure file.** Three-file procedure file is a kind of oversize file that can be uploaded dynamically, it is used in Zbasic grammar. Condition judgement, procedure jump and other operations are not supported. Three-file procedure can be downloaded into controller by instructions or by tool: zfile3view to scan, upload and download.
Grammar	FILE3_RUN "filename", tasknum filename: the file name of File3, it must be downloaded into controller first. tasknum: task No., find the first valid task by default.
Controller	Controllers with large storage size and firmware version above 2015 support.
Example	**FILE3_RUN** "aaa.z3p", 1 'run FILE3 procedure aaa.z3p in task 1.
Instructions	FILE3_ONRUN

FILE3_ONRUN--FILE3 Callback Function

Type	Callback Function
Description	**It will be triggered automatically when File3 starts.**
Grammar	GLOBAL FILE3_ONRUN: Label NO. GLOBAL SUB FILE3_ONRUN() self-defined SUB process (no attached parameters) Callback functions belong to File3 task.
Controller	General

Example	FILE3_RUN "aaa.z3p", 1 'run aaa.z3p in task 1.
	END
	
	GLOBAL SUB **FILE3_ONRUN()** 'start automatically when file3 starts
	IF 1= PROCNUMBER THEN
	BASE(0,1,2) 'choose axes list for three-time file
	SPEED=1000
	ACCEL=10000
	ELSE
	BASE(4,5,6)
	ENDIF
	END SUB

Instructions	FILE3_RUN

FILE3_GOTO--FILE3 Process Forces to Jump

Type	Task Functions
Description	**Valid in File3 task, it forces to jump into defined line number to run.**
Grammar	FILE3_GOTO(linenum) linenum: line NO. to jump to, starting from 1.
Controller	General
Instructions	FILE3_LINE, FILE3_RUN

FILE3_LINE -- FILE3 line NO.

Type	Task Functions
Description	**Return present running line NO. of File3, no matter the three-time file enters BASIC file due to SUB process calling, it will always return running line NO. of File3.**
Grammar	Value=FILE3_LINE([taskid]) taskid: task No. of file 3. When it is not filled, it will return function calling for the present task.
Controller	General

Instructions	FILE3_RUN, FILE3_GOTO

9.3 Task Parameter Instruction

BASE_MOVE--Assign Main Axis

Type	Task Parameters
Description	**Force to assign the main axis of interpolation motion fucntion, this instruction does not change actual motion.** Default value is -1, which is not valid at this time. Valid in firmware above 20160326. Each task has its unique BASE_MOVE parameter. Valid in interpolation instructions after BASE_MOVE setting: MOVE MOVEABS, MOVECIRC, MOVE_OP, MOVE_TASK etc. are not valid in single axis functions: cam, point-to-point etc.
Grammar	VAR1 = BASE_MOVE, BASE_MOVE = value
Controller	General
Example	**BASE_MOVE=2** 'force axis 2 as main axis, followed interpolation motion will execute by using axis 2 as main axis. and speed related parameters will also obey axis 2. MERGE(2)=ON 'defined as continuous interpolation SPEED(2)=100 ACCEL(2)=1000 BASE(0,1) MOVE(100,100) 'interpolation of axis 0 and axis 1, and axis 2 join this interpolation as main axis, but its move distance is 0. MOVE_OP(1,1) BASE(1) MOVE(100) 'axis 1 moves 100, and axis 2 as main axis. **BASE_MOVE=-1** 'cancel forced axis of present task.

PROC_STATUS--Task Status

Type	Task Status
Description	**Present Task Status** 0 task stops 1 task is running 3 task pauses
Grammar	VAR1 = PROC_STATUS(tasknum) tasknum Task NO.
Controller	General

Example	PRINT **PROC_STATUS(0)** 'Print status of task 0 Input remote instructions >> PRINT **PROC_STATUS(0)** Output:1
Instructions	PROC

PROC--Task Serial Number

Type	Task amendment subsidiary instructions
Description	**Appoint specific tasks when access to task parameters and task status.**
Grammar	PROC(tasknum) tasknum task NO. see instruction AXIS for reference to omit it.
Controller	General
Example	Example One: full format Print **PROC**_STATUS **PROC**(1) 'print running status of task 1. Exmple Two: brief fomat Print **PROC**_STATUS(1) 'print running status of task 1.
Instructions	PROCNUMBER

PROCNUMBER--Present Task NO.

Type	Task Specific Status, System Status
Description	**Task NO. of present running task.** Get task NO. through this instruction, task can not be modified by PROC in this situation.
Grammar	VAR1 = PROCNUMBER
Controller	General
Example	Print **PROCNUMBER** 'Print present task NO.
Instructions	PROC

PROC_LINE--Task Line

Type	Task Status
Description	**Present line NO. of task, which is only valid in other tasks.**
Grammar	VAR1 = PROC_LINE(tasknum) tasknum task NO.
Controller	General
Example	Print **PROC_LINE(0)** 'Print the code line of task 0.

	Input remote instructions >>print **PROC_LINE** Output:100
Instructions	PROC

PROC_PROGRESS-Progress of task instruction

Type	Task Status
Description	**The progress of task instruction, used by FILE, from 0-100.** Every task has the progress instruction. When LOAD_ZAR in FILE, can see the progress situation in HMII. Attention: FILE executes, only can be scanned synchronically in HMI task, do not drive FILE instruction directly through HMI task.
Grammar	VAR1 = PROC_LINE (tasknum) tasknum： task NO.
Controller	General
Instructions	PROC

PROC_PRIORITY-Task priority

Type	Task Status
Description	**Task priority, from 1-10, the highest is 10, the default value is 1, it is recommended only to modify a task.** Every task has the task priority. If needs to use the firmware that supports this function, it is recommended to update the firmware when the configuration doesn't take effect.
Grammar	Command Grammar: PROC_PRIORITY(tasknum)=value Read Grammar: VAR1=PROC_PRIORITY(tasknum) tasknum： task NO.
Controller	General
Example	PROC_PRIORITY(5)=3 'the priority of task 5 is 3 ?PROC_PRIORITY(5) 'check task 5 task priority
Instructions	PROC

ERROR_LINE--Error Line

Type	Task Status
Description	**Error Line NO. of present task.** Usually used through remote command after error happens.
Grammar	VAR1 = ERROR_LINE(tasknum) tasknum: Task NO.
Controller	General

Example	Input remote instruction >>**ERROR_LINE**(1) 'print error lines of task 1
Instructions	PROC, ERROR_SET

RUN_ERROR--Task Error Code

Type	Task Status
Description	**First error serial number in task.**
Grammar	VAR1 = RUN_ERROR(tasknum) tasknum: Task NO.
Controller	General
Example	?* **RUN_ERROR**(0) 'Print error NO. of task 0. 2043
Instructions	ERROR_LINE

TICKS--Task Count Period

Type	Task Parameters
Description	**Present task count period, minus 1 after every period. The unit is ms.** Each task has its unique TICKS parameters, period is 1ms in ZMC00X series and ZMC1XX series. No influence on TICKS count after system refresh period modified.
Grammar	VAR1 = TICKS, TICKS = value
Controller	General
Example	**TICKS** = 1000 WAIT UNTIL **TICKS** < 0 'wait until ticks<0. MOVE(100)
Instructions	TIME_TICKUS

TIME_TICKUS-Task Count Period

Type	Task Parameters
Description	**Present task count period, us 1 after every period.** Each task has its unique TIME_TICKUS parameters, 32 bits integer. No influence on TIME_TICKUS count after system refresh period modified.
Grammar	VAR1 = TIME_TICKUS, TIME_TICKUS = value
Controller	General
Example	**TIME_TICKUS**=0 DELAY(1) 'delay 1 ms

	?TIME_TICKUS 'print result: 1000, unit is us
Instructions	TICKS

Chapter X Operator and Mathematical Function Instructions

ZBASIC supports all operational characters in standard BASIC grammar, and it can also obey standard BASIC priority.

Priority: arithmetic operation> comparison operation >logic operation. If priority is the same, then operation will start from left to right in order.

Arithmetic Character		Comparison Character		Logic Character	
Description	**Character**	**Description**	**Character**	**Description**	**Character**
exponentiation	^	Equal	=	Logic negation	Not
Minus	-	Not equal	<>	Logic and	And
multiply	*	Less than	<	Logic or	Or or |
divide	/	More than	>	Logic XOR	Xor
exact divide	\	Less than or equal to	<=	Logic equivalence	Eqv
rermainder	Mod or %	More than or equal to	>=		
plus	+				
subtract	-				
shift left	<<				
shift right	>>				

10.1 Arithmetic Operation Instructions

+--Plus Operation

Type	Operational Character
Description	**Plus two expressions.**
Grammar	expression1+expression2 expression1: Any valid expressions expression2: Any valid expressions

Controller	General
Example	Online command input >>PRINT 1+2 Output： 3

---Minus Operation

Type	Operational Character
Description	**Minus two expressions.**
Grammar	expression1-expression2 expression1: Any valid expressions expression2: Any valid expressions
Controller	General
Example	Online command input >>PRINT 2-(2-1) Output： 1

* --Multiply Operation

Type	Operational Character
Description	**Multiply expression 1 and expression 2.**
Grammar	expression1 * expression2 expression1: Any valid expressions expression2: Any valid expressions
Controller	General
Example	Online command input >>PRINT 10*(1+2) Output： 30

/ --Divide Operation

Type	Operational Character
Description	**Use expression 1 to divide expression 2.**
Grammar	expression1 / expression2 expression1: Any valid expressions expression2: Any valid expressions
Controller	General
Example	Online command input >>PRINT 10/3 Output： 3.3333

\ --Exact Divide

Type	Operational Character
Description	**Exact Divide.**
Grammar	expression1 \ expression2 expression1: Any valid expressions expression2: Any valid expressions
Controller	General
Example	Online command input >>PRINT 10 \ (1+2) Output： 3

<< --Shift Left

Type	Operational Character
Description	**Shift left**
Grammar	expression1 << expression2 expression1: Any valid expressions expression2: Any valid expressions Priority is lower than other operational characters, so use () when they are commonly used. See example three for reference.
Controller	General
Example	**Example one: operate number directly.** Online command input >>PRINT 8<<1 'relevant binary shift left one bit Output: 16 Online command input >>PRINT 8<<2 'relevant binary shift left two bits Output: 32 **Example two: operate variable and registers** DIM bb bb=8 MODBUS_REG(0)=8 PRINT bb<<1,bb<<2 PRINT MODBUS_REG(0)<<1,MODBUS_REG(0)<<2 **Example Three: priority comparison** ?PRINT 8<<1+1 'relevant binary shift left two bits Output: 32 ?PEINT (8<<1)+1 'relevant binary shift left one bit Output: 17

>>--Shift Right

Type	Operational Character
Description	**Shift Right**
Grammar	expression1 >> expression2 expression1: Any valid expressions expression2: Any valid expressions Priority is lower than other operational characters, so use () when they are commonly used. See example three for reference.
Controller	General
Example	**Example one: operate number directly.** Online command input >>PRINT 8>>1 'relevant binary shift right one bit Output: 4 Online command input >>PRINT 8>>2 'relevant binary shift right two bits Output: 2 Example two: Operate variable and registers DIM bb bb=8 MDOBUS_REG(0)=8 PRINT bb>>1,bb>>2 PRINT MODBUS_REG(0)>>1, MODBUS_REG(0)>>2 Example Three: priority comparison ?PRINT 8>>1+1 'relevant binary shift right two bits Output: 2 ?PRINT (8>>1)+1 'relevant binary shift right one bit Output: 5

MOD--Remainder Operation

Type	Operational Character
Description	**Remainder Operation**
Grammar	expression1 MOD expression2 expression1: Any valid expressions, get integer part. expression2: Any valid expressions, get integer part
Controller	General
Example	Online command input >>PRINT 10 **MOD** (1+2) Output: 1

ABS--Absolute Operation

Type	Mathematical Function
Description	**Evaluate absolute value.**
Grammar	ABS(expression) expression Any valid expressions
Controller	General
Example	PRINT **ABS**(-11) 'result is 11

10.2 Comparison Operation Instructions

= --Comparison or Assign Operation

Type	Operational Character
Description	**Comparison Operational Character**: if expression 1 is equal to expression 2, then return TRUE, or it will return False. **Assign Operational Character**: assign value of expression 2 to the former variables or parameters.
Grammar	expression1 = expression2 expression1: Any valid expressions expression2: Any valid expressions
Controller	General
Example	**Example One:** ON IN(0)=ON GOTO label1 'if input channel 0 is ON, then jump to execute label1. Start to execute from the first line. label1: PRINT12 'print 12 **Example Two:** DIM aaa aaa=100 'assign variable aaa as 100 PRINT aaa

<>--Not Equal

Type	Operational Character
Description	**If expression 1 is not equal to expression 2, then return TRUE, or return FALSE.**
Grammar	expression1 <> expression2

	expression1: Any valid expressions expression2: Any valid expressions
Controller	General
Example	ON MODBUS_BIT(0)<>0 GOTO label1 'if MODBUS 0 is not 0, then go to execute label1. label1: PRINT11 'print 11

>--More Than

Type	Operational Character
Description	**If expression 1 is more than expression 2, then return TRUE, or return FALSE.**
Grammar	expression1 > expression2 expression1: Any valid expressions expression2: Any valid expressions
Controller	General
Example	WAIT UNTIL MPOS>100 'Wait until position feedback is more than 100. **Example One:** Dim q 'define variable q= 2>1 '2 is more than 1, return TRUE. PRINT q 'print return value **Example two:** DIM a 'define variable a=0 'assign variable REPEAT 'execute in cycle a=a+1 'plus 1 ?a 'print DELAY(200) 'delay UNTIL a>10 'when a is more than 10, cycle ends.

>= --More Than or Equal To

Type	Operational Character
Description	**If expression 1 is more than or equal to expression 2, then return TRUE, or return FALSE.**
Grammar	expression1 >= expression2 expression1: Any valid expressions expression2: Any valid expressions
Controller	General

Example	DIM a a= 1>=3 PRINT a	'define variables '1<3, so return FALSE 'print the return value

< --Less Than

Type	Operational Character
Description	**If expression 1 is less than expression 2, then return TRUE, or return FALSE.**
Grammar	expression1 < expression2 expression1: Any valid expressions expression2: Any valid expressions
Controller	General
Example	VAR1=1<0 Since 1<0, so var1=false.

<= --Less Than or Equal To

Type	Operational Character
Description	**If expression 1 is less than or equal to expression 2, then return TRUE, or return FALSE.**
Grammar	expression1 <= expression2 expression1: Any valid expressions expression2: Any valid expressions
Controller	General
Example	VAR1=1<=1 Since 1=1, so var1=true (-1).

10.3 Logical Operation Instruction

AND--Bit Operation: AND

<table>
<tr><td>Type</td><td>Operational Character</td></tr>
<tr><td>Description</td><td>Operate data bit: AND, only operate integer part.
<table>
<tr><th colspan="2">AND</th><th>Result</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td></tr>
</table>

Grammar	expression1 AND expression2 expression1: Any valid expressions expression2: Any valid expressions Result is AND bits operation of expression 1 and expression 2.
Controller	General
Example	Online command input >>PRINT 1 **AND** 2 Output: 0 Process: 1 bit format 01 2 bit format 10 after AND operation, bit is 00, which is 0 in decimal format.

OR--Bit Operation: OR

Type	Operational Character
Description	**Operate data bit: OR, only operate integer part.**

OR		Result
0	0	0
0	1	1
1	0	1
1	1	1

Grammar	expression1 OR expression2 expression1: Any valid expressions expression2: Any valid expressions Result is OR bits operation of expression 1 and expression 2.
Controller	General
Example	Online command input >>PRINT 1 **OR** 2 Output: 3 Process: 1 bit format 01 2 bit format 10 after AND operation, bit is 11, which is 3 in decimal format.

NOT--Bit Operation: NOT

Type	Operational Character

<table>
<tr><td>Description</td><td>Operate data bit: NOT, only operate integer part. Be careful to operate ON.
<table>
<tr><th>NOT</th><th>Result</th></tr>
<tr><td>0</td><td>-1</td></tr>
<tr><td>1</td><td>-2</td></tr>
</table></td></tr>
<tr><td>Grammar</td><td>NOT expression1
expression1: Any valid expressions</td></tr>
<tr><td>Controller</td><td>General</td></tr>
<tr><td>Example</td><td>Online command input
>>print NOT 1
Output: -2

Process
1 bit format …0000 0001
after NOT operation, bit is …1111 1110, which is -2 in decimal format.</td></tr>
</table>

XOR--Bit Operation:XOR

<table>
<tr><td>Type</td><td>Operational Character</td></tr>
<tr><td>Description</td><td>Operate data bit: XOR, only operate integer part.
<table>
<tr><th colspan="2">XOR</th><th>Result</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table></td></tr>
<tr><td>Grammar</td><td>expression1 XOR expression2
expression1: Any valid expressions
expression2: Any valid expressions</td></tr>
<tr><td>Controller</td><td>General</td></tr>
<tr><td>Example</td><td>Online command input
>>PRINT 1 XOR 1
Output: 0

Process:
1 bit format 01
after XOR operation, bit is 00, which is 0 in decimal format.</td></tr>
</table>

EQV--Bit Operation:EQV

Type	Operational Character

<table>
<tr><td>Description</td><td>Operate data bit: EQV, only operate integer part.
<table>
<tr><th colspan="2">EQV</th><th>Result</th></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table></td></tr>
<tr><td>Grammar</td><td>expression1 EQV expression2
expression1: Any valid expressions
expression2: Any valid expressions</td></tr>
<tr><td>Controller</td><td>General</td></tr>
<tr><td>Example</td><td>Online command input
>>print 2 EQV 1
Output: -4

Process:
2 bit format …0000 0010
1 bit format …0000 0001
after EQV operation, bit is …1111 1100, which is -4 in decimal format.</td></tr>
</table>

10.4 Trigonometry Instructions

SIN-- Trigonometric Function: SINE

Type	Mathematical Function
Description	**Evaluate sine, input parameter should be arc unit.**
Grammar	SIN (expression) expression Any valid expressions
Controller	General
Example	PRINT **SIN**(PI/6) 'result is 0.5000

ASIN--Trigonometric Function: Anti-SINE

Type	Mathematical Function
Description	**Evaluate anti-sine, returned value is arc unit.**
Grammar	ASIN (expression) expression Any valid expressions
Controller	General
Example	PRINT **ASIN**(0.5) 'result is 0.52360

COS--Trigonometric Function: Cosine

Type	Mathematical Function
Description	**Evaluate Cosine, input parameter should be arc unit.**
Grammar	COS(expression) expression Any valid expressions
Controller	General
Example	PRINT **COS**(PI/3) 'result is 0.5000

ACOS -- Trigonometric Function: Anticosine

Type	Mathematical Function
Description	**Evaluate anticosine, returned value is arc unit.**
Grammar	ACOS (expression) expression Any valid expressions
Controller	General
Example	PRINT **ACOS** (0.5) 'result is 1.04720=PI/3

TAN--Trigonometric Function: Tangent

Type	Mathematical Function
Description	**Evaluate Tangent, input parameter should be arc unit.**
Grammar	TAN (expression) expression Any valid expressions
Controller	General
Example	PRINT TAN(pi/3) 'result is 1.732

ATAN--Trigonometric Function: Antitangent

Type	Mathematical Function
Description	**Evaluate antitangent, returned value is arc unit.**
Grammar	ATAN(expression) expression Any valid expressions
Controller	General
Example	PRINT **ATAN**(1) 'result is 0.7854 = (45/180)*PI

ATAN2--Trigonometric Function: Antitangent 2

Type	Mathematical Function
Description	**Evaluate antitangent, returned value is arc unit.**
Grammar	ATAN2(y, x) y: y coordinate x: x coordinate
Controller	General
Example	PRINT **ATAN**2(1,0) 'result is 1.5708

10.5 Exponentiation Instructions

EXP--Exponent

Type	Mathematical Function
Description	**Exponent function**
Grammar	exp([base,] expvalue) base base number, default value is e expvalue exponent
Controller	General
Example	Example One： PRINT **EXP**(2,4) 'result is 16 (2*2*2*2) Example Two： PRINT **EXP**(1) 'result is 2.7183

SQR-- Square Root

Type	Mathematical Function
Description	**Square root function**
Grammar	SQR(expression) expression Any valid expressions
Controller	General
Example	a=**SQR**(4) PRINT a 'result is 2

LN-- Natural Logarithm

Type	Mathematical Function

Description	**Natural logarithm function**
Grammar	LN(expression) expression Any valid expressions
Controller	General
Example	a= **LN**(1) PRINT a 'result is 0

LOG--Logarithm of 10

Type	Mathematical Function
Description	**Logarithmo, which base number is 10.**
Grammar	LOG(expression) expression Any valid expressions
Controller	General
Example	a= **LOG**(100) PRINT a 'result is 2

10.6 Data Operate Instruction

SET_BIT--Set Bit

Type	Mathematics Instructions or Functions
Description	**Bit operation, only for integer, set bit as 1.** There are command grammar and function grammar. For VR register, it can be set as 0-24.
Grammar	**Command Grammar: SET_BIT(bit#,vr#)** Operate VR directly bit#: bit NO.:0-24 vr#: VR variable NO. to operate, integer part. There is no returned value when use command grammar, only modify value of object directly. **Function Grammar: ret=SET_BIT(bit#,int)** ret operation result bit# bit NO.:0-24 int expression to operate, only get the integer part. There is returned value after using function grammar, but value of object did not change.
Controller	General
Example	**Example One: Command Grammar** VR(23)=0.333 **SET_BIT**(0,23) 'set bit 0 of VR(23) as 1, and clear decimal part.

	?VR(23) 'result is 1 **Example Two: Function Grammar** Dim a,b a=0.333 b=0 b=**SET_BIT** (0,a) 'set bit 0 of a as 1, and assign value to b, clear decimal PRINT a,b 'print result:0.333,1, a didn't change, b=1.
Instruction	CLEAR_BIT, READ_BIT2, READ_BIT

CLEAR_BIT--Operate Bit 0

Type	Mathematics Instructions or Functions
Description	**Bit operation, only for integer, modify bit 0.** There are command grammar and function grammar. For VR register, it can be set as 0-24.
Grammar	Command Grammar: CLEAR_BIT(bit#,vr#) Operate VR directly bit#, bit NO.:0-24 vr# VR variable NO., integer part. There is no returned value when use command grammar, only modify value of object directly. Function Grammar: ret = CLEAR_BIT(bit#,int) ret operation result bit# bit NO.:0-24 int expression to be operated, only operate integer part. There is returned value after using function grammar, but value of object did not change.
Controller	General
Example	**Example One: Command Grammar** VR(23)=3.333 **CLEAR_BIT** (0,23) 'set bit 0 of VR(23) as 0 ?VR(23) 'print result: 2 **Example Two: Function Grammar** Dim a,b a=3.333 b=0 b= CLEAR_BIT(0,a) 'return value to b after clear bit 0 of a and get integer part. PRINT a,b 'result is: 3.333,2 a didn't change, b=2.
Instruction	SET_BIT, CLEAR_BIT, READ_BIT2

READ_BIT--Read Bit

Type	Mathematical Function
Description	**Bit operation, which is only for integer, read bit status.** Only operate VR, see READ_BIT2 if not VR. For VR register, it can be set as 0-24.
Grammar	ret = READ_BIT(bit#, vr#) ret: result:1 or 0 bit#: bit NO.:0-24 vr#: VR variable No. to operate
Controller	General
Example	VR(23)=3.333 PRINT **READ_BIT**(0,23) 'read bit 0 of VR(23), result is 1.
Instruction	SET_BIT, CLEAR_BIT, READ_BIT2

READ_BIT2--Read Bit 2

Type	Mathematical Function
Description	**Bit operation, only for integer, read bit status.**
Grammar	ret = READ_BIT2(bit#, int) ret: result:1 or 0 bit#: bit NO.:0-31 int: expression, use integer part.
Controller	General, valid in firmware version above 20130813.
Example	DIM a,b b=1.64 a=**READ_BIT2**(0,b) 'read bit 0 of b, assign value to b. PRINT a 'output a, result is 1.
Instruction	SET_BIT, READ_BIT

FRAC--Return Decimal

Type	Mathematical Function
Description	**Return decimal part, which is always over 0**
Grammar	FRAC(expression) expression: number to operate
Controller	General
Example	a=**FRAC**(1.235) **PRINT** a 'result is 0.235

INT--Return Integer

Type	Mathematical Function
Description	**Return integer part.**
Grammar	INT(expression) expression: Any valid expressions
Controller	General
Example	a=INT(1.235) PRINT a 'result is 1 ?**INT**(-1.1) 'print result: -2, since decimal part is aways converted to integer.

SGN--Return Sign

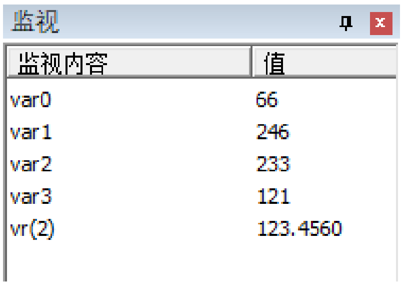
Type	Mathematical Function
Description	**Return sign.** 1: more than 0 0: equal to 0 -1: less than 0
Grammar	SGN(expression) expression: Any valid expressions
Controller	General
Example	a=**SGN**(-1.235) PRINT a 'result is -1

IEEE_IN--Combine Float Number

Type	Mathematical Function
Description	**Combine 4 bytes into a single-precision float point number**
Grammar	IEEE_IN(byte0,byte1,byte2,byte3) byte0 – byte3 4 bytes
Controller	General
Example	VAR = **IEEE_IN**(VR(10),VR(11),VR(12),VR(13)) Make these 4 data into one single-precision float point number.

IEEE_OUT--Select Single Byte

Type	Mathematical Function
Description	**Select one byte from a single-precision float point number**
Grammar	byte_n = IEEE_OUT(VAR, n) var: single-precision float point number N: 0-3, byte to be selected.

Controller	General
Example	Example 1 VAR = **IEEE_OUT**(VR(1),2) 'select the second byte of VR(1) Example 2 GLOBAL VAR0,VAR1,VAR2,VAR3 VAR0=0 VAR1=0 VAR2=0 VAR3=0 VR(1)=123.456 VAR0 = IEEE_OUT(VR(1),0) VAR1 = IEEE_OUT(VR(1),1) VAR2 = IEEE_OUT(VR(1),2) VAR3 = IEEE_OUT(VR(1),3) VR(2)=0 VR(2)=IEEE_IN(VAR0,VAR1,VAR2,VAR3) The result: 

$--Hexadecimal

Type	Special Character
Description	**Indicate the followed data is hexadecimal format.**
Grammar	$hexnum
Controller	General
Example	Online command input >>PRINT $F Output: 15

10.7 Character String Operation Instruction

CHR--ASCII Code Print

Type	String Functions
Description	**Return ASCII, which is only used in PRINT.**

Grammar	CHR(expression) expression: Any valid expressions
Controller	General
Example	Online command input >>PRINT **CHR**(66) Output: B

HEX--Print Hexadecimal

Type	String Functions
Description	**Return hexadecimal format, which is only used for PRINT.**
Grammar	HEX(expression) expression: Any valid expressions, only select integer part.
Controller	General
Example	Online command input >>PRINT **HEX**(15); Output: f 'hexadecimal

STRLEN-Return String Length

Type	String Functions
Description	**Return string length**
Grammar	len=STRLEN(str) str： string
Controller	General
Example	DIM str_a(20) str_a="len123" ?STRLEN(str_a) Print result: 6

TOSTR—Format Output

Type	String Functions
Description	**Format output function. Convert variable to string.**
Grammar	TOSTR(VAR1, [N],[DOT]) VAR1: Any valid expressions N: total output digits, including decimal digit and sign digit. when N is set as minus value, which means right alignment. DOT: decimal number to output, when N is too small, there is no decimal digit, then will not output decimal part. All Output is string type. Only the first parameter is printed to four decimal

	places by default.
Controller	General
Example	Example One Online command input >> PRINT **TOSTR**(2-100,6,2) Output： -98.00 Example two Dim aa(20) aa="asd13"+**TOSTR**(354) ?aa 'print result:asd13354.000

STRCOMP--String Comparison

Type	String Functions
Description	**String comparison function, return logic result: >0 or =0 or <0 after comparison of two strings.** The comparison length should not exceed 500 bytes, or the returned value will appear error.
Grammar	STRCOMP(str1, str2) str1: string1 str2: string2
Controller	General
Example	DIM AAA(10) AAA = "abc" Online command input >>PRINT STRCOMP(AAA, "abc") Output: 0

STRFIND—String Search

Type	String Functions
Description	**String searching function.**
Grammar	STRFIND(str1, str2 [, firstindex]) str1: string to search str2: search sample string firstindex: search from that position, default value is 0. Return: >= 0, return aimed index after searching. < 0, no string is found.
Controller	General

Example	DIM AAA(10),BBB(3) AAA="AD23GF41" BBB="23G" ?**STRFIND**(AAA,BBB) 'print aimed index after searching, 2

STRCONV—Encoder Conversion

Type	Character string function.
Description	**Character string conversion of different codes.** Only support encoder without 0, UTF16 is invalid. Encoders can be supported: CP936, UTF-7, UTF-8, GB2312, etc.
Grammar	string2=STRCONV(“srccodename”, “descodename”, “string1”)
Controller	Controllers with Linux and 7XX series.
Example	DIM arrstring(100) arrstring = STRCONV("gb2312","utf8", "folder") ?STRCONV("utf8", "gb2312", arrstring) 'output result: folder

VAL--Convert String to Number

Type	String Functions
Description	**Convert String to number.** Only convert string to number, when meets alphabet or sign, it will stop.
Grammar	VAL(str1) str1 string
Controller	General
Example	Example One VAR1 = **VAL**("123") ?VAR1 'print result,123 Example two VAR2 = **VAL**("123QWE23") ?VAR2 'print result,123

10.8 Constant Instruction

PI--Circular Constant

Type	Constant
Value	3.14159
Controller	General

TRUE--True Value

Type	Constant
Value	-1
Controller	General

FALSE--False Value

Type	Constant
Value	0
Controller	General

ON--Open

Type	Constant
Value	1
Controller	General

OFF--Close

Type	Constant
Value	0
Controller	General

10.9 Advanced Operational Instruction

CRC16 --CRC Verification Calculation

Type	Mathematical Function
Description	**CRC16 CCITT calculation.**
Grammar	CRC16(arrayname, index, size[, inital] [, poly]) arrayname: array where data are saved, one byte occupy one position index: array index size: array size. initial: default value of CRC calculation, default is $FFFF. poly: polynomial, only supports $A001 of MODBUS and $1021 of CCITT, default value is $A001
Controller	General

Example	TABLE(0, $FE, $48 , $06 , $00 , $6D , $00 , $00 , $00) 'store 8 data in TABLE. CRCVALUE = CRC16(TABLE, 0, 8) 'CRC calculation, result is $1A0D TABLE(8)= CRCVALUE \ 256 'add CRC to end of data, big end mode. TABLE(9)= CRCVALUE and $FF

DTSMOOTH--Table Smooth

Type	Mathematical Function
Description	**Smooth coordinate in TABLE**
Grammar	DTSMOOTH (axes, dtfirst, space, points, imode, referradius) axis: axis number dtfirst: TABLE index of first coordinate space: index interval of two points. or space one point consumes. Points: total points. Imode: 0-absolute mode, adjust when curvature radius is under reference value. referradius: reference curvature radius, equal to speed^2/corner speed.
Controller	ZMC3X series with firmware above 20161206. ZMC4 series with firmware above 20170508.
Example	TABLE(0, 0,0) TABLE(5, 99,0) TABLE(10, 100,0) TABLE(15, 100, 1) TABLE(20, 101, 1) TABLE(25, 200, 1) **DTSMOOTH(2, 0, 5, 6, 0, 5)** ?*TABLE(0, 2) ?*TABLE(5, 2) ?*TABLE(10, 2) ?*TABLE(15, 2) ?*TABLE(20, 2) ?*TABLE(25, 2)

B_SPLINE--B-Spline Smooth

Type	Mathematical Function
Description	**Use data in table to do B-spline smooth.**
Grammar	B_SPLINE(type, data_start, points, data_out, ratio) type: Type, valid in 1-B-spline. data_start: graphics data starting position in TABLE points: the number of graphics data. data_out: graphics data starting position in TABLE after smooth.

	ratio: smooth ratio of B_SPLINE function, the number after smooth is points * ratio. Add the automatic calculation function of spline control point. It is used together with MOVESPLINE spine curve motion. Products above 4 series support, 4 series with firmware above 20170621. B_SPLINE(type, axises, dtstartpos, dtendpos, dtlastpos, dtnexpos, dtoutcontrol1, dtoutcontrol2) type： 1: compatible with original functions 11: calculate spined control points for the first segment of continuous line segment. 12: calculate spined control points for the middle segment of continuous line segment. 13: calculate spined control points for the last segment of continuous line segment. axises: axes to do spline interpolation dtstartpos: table array index of segment starting point coordinate, multi-axis saved in different table continuously. Same as follow. dtendpos: table array index of segment ending point coordinate dtlastpos: front point's coordinate index of segment starting point is used to calculate for reference, the first segment parameter is invalid. dtnexpos: back point's coordinate index of segment starting point is used to calculate for reference, the last segment parameter is invalid. dtoutcontrol1: output control point data of spline, the first control point of Bessel. (except starting point as control point) dtoutcontrol2: output control point data of spline, the second control point of Bessel. There are four control points of Bessel, including starting point, dtoutcontrol1, dtoutcontrol2 and ending point.
Controller	General
Example	**Example 1 type1** **B_SPLINE**(1,0,10,100,10) 'smooth 10 picture data, saved in table(0) to table(9), after smooth, become 100 data, save in table(100) to table(199). **Example 2 New Mode** TABLE(0,0,0,0,100,100,100,200,100) 'coordinate data of 4 consecutive points on XY axis. **B_SPLINE**(11, 2, 0, 2, -1, 4, 100,200) 'the first segment ?TABLE(100),TABLE(101),TABLE(200),TABLE(201) **B_SPLINE**(12, 2, 2, 4, 0, 6, 100,200) 'the second segment

	?TABLE(100),TABLE(101),TABLE(200),TABLE(201) **B_SPLINE**(13, 2, 4, 6, 2, 6, 100,200) 'the third segment ?TABLE(100),TABLE(101),TABLE(200),TABLE(201)

TURN_POSMAKE--Rotating Center Calculation

Type	Mathematical Function
Description	**Center point calculation of rotating, positive direction of rotating should be same as positive direction of XY.(right hand rule)**
Grammar	TURN_POSMAKE(tablenum, posx, posy, disR, tableout) tablenum table NO. where save rotating parameters. posx X coordinate. posy Y coordinate disR relative offset of rotating axis. tableout save coordinate after calculation..
Controller	General
Instructions	MCIRC_TURNABS

ZCUSTOM--Motion Parameters Calculation

<table>
<tr><td>Type</td><td>Mathematical Function</td></tr>
<tr><td>Description</td><td>Calculate parameters in all kinds of motion commands, for detailed grammar function, please refer to following.</td></tr>
<tr><td>Grammar</td><td>Function 2: calculate the position of a point at a certain distance on a space arc or straight line.
In Table parameters, according to 3-point circle making mode, fill in other two parameters.
Fill in relative coordinates, then the returned coordinates are also relative.
Grammar: ZCUSTOM (2,tableend, tablemid, tableout, mode, vectdis)
tableend: table index that saves end point of circular arc.
tablemid: table index that saves middle point of arc, which together with the current point constitutes the three points of the arc.
tableout: table index that outputs calculation data
mode: value meanings as follow:
<table>
<tr><td>Mode</td><td>Description</td></tr>
<tr><td>1</td><td>Space circular arc length relates to starting point.</td></tr>
<tr><td>2</td><td>Space circular arc length relates to end point.</td></tr>
<tr><td>3</td><td>Linear distance, relate to starting point, now, tablemid is invalid.</td></tr>
<tr><td>4</td><td>Linear distance, relate to end point, now, tablemid is invalid.</td></tr>
<tr><td>5</td><td>Relatively calculate circle center of space arc, now, vectdis is invalid, and tableout only outputs circle center xyz,</td></tr>
</table>
</td></tr>
</table>

	radian range, and arc length in sequence.

vectdis: the distance of point (to be calculated) that relates to "mode", minus value means forward. In arc mode, positive value means clockwise, minus value means anticlockwise.

Function 6: calculate tangent angle direction of space arc's starting point and end point, the unit is radian.

Fill in relative coordinates, and returned coordinates are also relative.

Grammar: ZCUTOM (6, tableend, tablemid, tableout)

tableend: table index that saves end point of arc.

tablemid: table index that saves middle point of arc, consist 3 points of arc together with present point.

tableout: table index that saves result data. It will output starting point XY, starting point Z, end point XY, end point Z.

Function 7: input speed ratio, then calculate salve and main axis position of MOVESLINK.

Grammar: ZCUSTOM(7, distance, link_dist, start_sp, end_sp, speed ratio, tableout)

distance: distance that the slave axis moves during the link, unit is units.

link_dist: absolute distance that reference axis moves during the link, units is units.

star_sp: the speed ratio of slave axis to reference axis when starts, unit is units/units, minus value means slave axis moves in negative direction.

end_sp: the speed ratio of slave axis to reference axis when ends, unit is units/units, minus value means slave axis moves in negative direction.

speed ratio: speed ratio of points that needs calculating

tableout: forward salve axis distance, forward main axis distance, slave axis distance from reverse, and main axis distance from the reverse, they occupy 4 TABLE. (for one curve, there are many several solutions of speed ratio.)

Function 8: MOVESLINK inputs slave axis position, then calculates master axis position.

Grammar: ZCUSTOM(8, distance, link_dist, start_sp, end_sp, distancemoved, tableout)

distance: distance that slave axis moves during links.

link_dist: absolute distance that master axis moves during links.

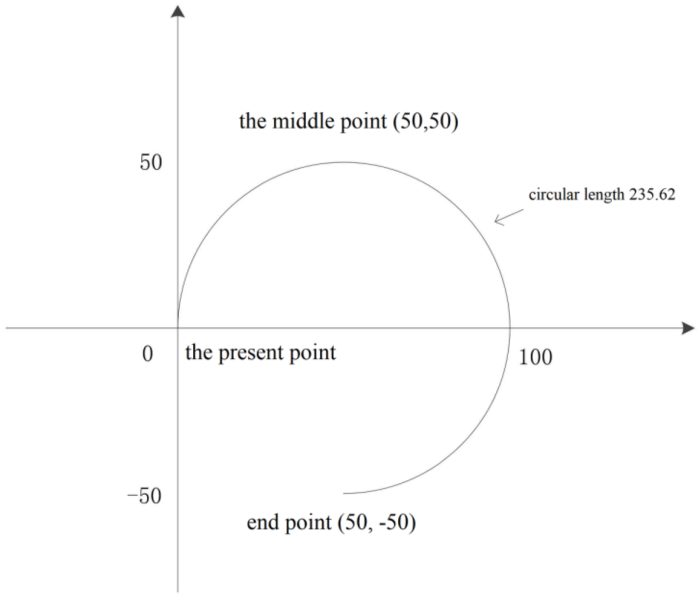
start_sp: speed ratio of starting pulse

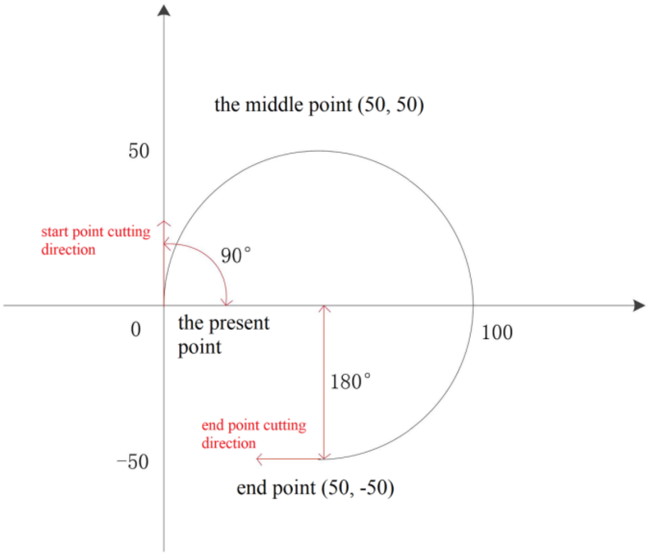
end_sp: speed ratio of end pulse

distancemoved: motion distance that slave axis already moved

tableout: table index that saved position of master axis, if there are multi results, return the first one.

	Function 9: FLEXLINK inputs slave axis position, then gets main axis position. Grammar: ZCUSTOM (9, base_dist, excite_dist, link_dist, base_in, base_out, excite_acc, excite_dec, distancemoved, tableout) base_dist: uniform motion distance of slave axis. excite_dist: excite motion distance of slave axis, when the value is opposite to base_dist, it can't calculate the main axis' position. link_dsit: whole process, after slave axis motion finished, motion distance of main axis. base_in: before excite motion, the percent of salve axis motion distance to base_dist. base_out: after excite motion, the percent of salve axis remain distance to base_dist. Don't add them more than 100%. excite_acc: in the process of excite motion, the precent of slave axis acceleration distance to excit_dist, when excite_dist is minus value, it's the deceleration stage. excite_dec: in the process of excite motion, the precent of slave axis deceleration distance to excit_dist, when excite_dist is minus value, it's the acceleration stage. distancemoved: pulse amounts of slave axis that had moved tableout: output Table index, output relative main axis position, if there are several solutions, return to the first one. **Function 10: calculate FRAME_ROTATE parameters in original coordinate according to workpiece coordinate three points. Every point needs to store xyz coordinates.** Grammar: ZCUSTOM (10, dtzero, dtx, dty, dtout) dtzero: workpiece origin point position in original coordinate. dtx: point (on the workpiece coordinate X) position at original coordinate. dty: point (on the workpiece coordinate Y) position at original coordinate. dtout: output TABLE index, and store respectively: X,Y,Z,RX,RY,RZ **Function 12: calculate circle center according to arc' end point, radius.** Grammar: ZCUSTOM (12, xpos, ypos, radius, anticlock, dtout) xpos: relative coordinate in X ypos: relative coordinate in Y radius: radius, negative value means the arc sector angle > 180°, positive value means < 180°. anticlock: 0: clockwise, 1: anticlockwise dtout: output circle center relative distance xy, it needs two positions. **Note:** dtzero, dtx, dty, and dtout are table index, it only needs to write index, table saves x, y, z of three directions, for robotic arm algorithm, it fills in

	virtual axis’ coordinates.
Controller	General
Example	**Example one Use function 2**  TABLE(0,50,-50,0) 'set end coordinate, relative position. TABLE(3,50,50,0) 'set middle coordinate, relative position. **'mode1， relative to start point** **ZCUSTOM(2,0,3,10,1,78.54)** 'relative to start point, clockwise, arc length is 78.54 ?TABLE(10),table(11),table(12)'output relative coordinate:0,0,0 **ZCUSTOM(2,0,3,10,1,-78.54)** 'relative to start point, anticlockwise, arc length is 78.54 ?TABLE(10),TABLE(11),TABLE(12) 'output relative coordinate:50,-50,0 **'mode2, relative to end point** **ZCUSTOM(2,0,3,10,2,78.54)** 'relative to end point, clockwise, arc length is 78.54. ?TABLE (10),TABLE (11),TABLE (12) 'output relative coordinate: 50,-50,0 **ZCUSTOM(2,0,3,10,2,-78.54)** 'relative to end point, anticlockwise, arc length is 78.54 ?TABLE (10),TABLE (11),TABLE (12) 'output relative coordinate:100,0,0 **'mode5, calculate center, radian, arc length.** **ZXUTOM(2,0,3,10,5,0)** 'distance parameters are not valid in this situation. ?TABLE (10),TABLE (11),TABLE (12) 'output relative coordinate: 50,-50,0 ?TABLE(13),TABLE(14) 'output arc angle: 4.712, arc length: 235.62 **Example two function 6, calculate tangent radian of start point and end point**

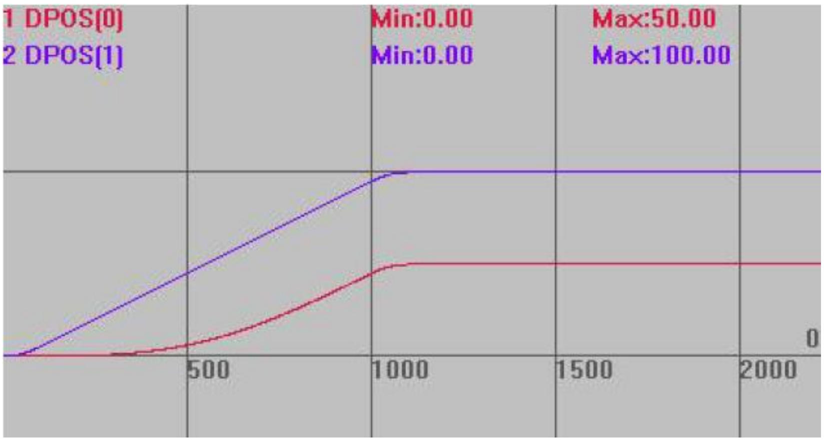


TBALE(0,50,-50,0)   'set end point, relative position.
TABLE(3,50,50,0)   'set middle point, relative position
ZCUSTOM(6,0,3,10) 'calculate tangent radian of present and end point.
?TABLE(10),TABLE(11) 'output present point:1.571,0 (1.571=90*PI/180)
?TABLE(12),TABLE(13) 'output end point:-3.142,0 (-3.142=-180*PI/180)

Example Three Function 8, calculate master axis position of MOVESLINK

BASE(0,1)
DPOS=0,0
UNITS=100,100
SPEED=100,100
ACCEL=1000,1000
TRIGGER
MOVESLINK(50,100,0,1,1)   'build MOVESLINK connection
MOVEABS(100)   AXIS(1)
ZCUSTOM(8,50,100,0,1,25,10) 'calculate master axis position when slave axis moves 25.
?TABLE(10)   'output master axis position:73.801

Motion Path:
DPOS(0) vertical scale 100
DPOS(1) vertical scale 100



ZMATH64-64 Bits Calculation

Type	64 bits calculation instruction
Description	**Calculate 64 bits stored in D Register. (MODBUS Register).** A 64 bits integer with symbol occupies 4 registers (the small end mode). Only operate MODBUS register, not to VR mapping, etc. 4xxx series controller with firmware version above 20170629.
Grammar	ZMATH64(opmode, dindex1, dindex2) opmode: operation NO. dindex1、dindex2：MODBUS register NO.

operation No.	Execute operation	Description
1	64-bit integer addition	D64(dindex1)+=D64(dindex2)
2	64-bit integer subtraction	D64(dindex1)-=D64(dindex2)
3	64-bit integer multiplication	D64(dindex1)*=D64(dindex2)
4	64-bit integer division	D64(dindex1)/=D64(dindex2)
5	64-bit integer is redundant	D64(dindex1)%=D64(dindex2)
11	64-bit integer read	D64(dindex1)=D64(dindex2)
12	64-bit integer conversion	D32(dindex1)=D64(dindex2)
13	64-bit integer read	D64(dindex1)=D32IEEE(dindex2)
14	64-bit integer conversion	D32EEE(dindex1)=D64(dindex2)

	15	assign	D64(dindex1)=double64(dindex2)
	16	assign	double64(dindex1)=D64 (dindex2)
	17	assign	double64(dindex1)=D32EEE(dindex2)
	18	assign	D32IEEE (dindex1) = double64 (dindex2)
	21	double addition	double64 (dindex1) += double64 (dindex2)
	22	double subtraction	double64 (dindex1) -= double64 (dindex2)
	23	double multiplication	double64 (dindex1) *= double64 (dindex2)
	24	double division	double64 (dindex1) /= double64 (dindex2)
	25	double remainder, decimals	double64 (dindex1) %= double64 (dindex2)
	D321IEEE means float points storage, same as MODBUS_IEEE. D64 means 64-bit integer with symbol storage, which can read 32-high-bit and 32-low-bit through two MODBUS_LONG.		
Controller	General		
Example	MODBUS_LONG(0)=100 MODBUS_LONG(8)=20 **ZMATH64**(1,8,0) '64-bit integer addition, then stores in the start address of MODBUS_LONG(8) ?MODBUS_LONG(0) 'print result, 100 ?MODBUS_LONG(8) 'print result, 120		
Instructions	MODBUS_IEEE, MODBUS_LONG, MODBUS_REG		

MODBUS_DOUBLE- Read MODBUS

Type	64-bit instruction
Description	**Read double data from MODBUS, and it can assign to other variable arrays.** 3 series and below arrays with float don’t support the instruction.
Grammar	MODBUS_DOUBLE(index) Index: modbus register NO.
Controller	General
Example	MODBUS_LONG(0)=100 MODBUS_LONG(8)=200 ZMATH64(16, 0, 8) 'assign 64-bit ?**MODBUS_DOUBLE**(0) 'print result, 200

	?MODBUS_LONG(0) 'print result, 0 ?MODBUS_LONG(8) 'print result, 200
Instruction	ZMATH64

Chapter XI Axis Parameter and Axis Status Instruction

Axis parameters modification grammar: SPEED=value, here is the speed of default axis: BASE axis. If needs to modify appointed axis parameters through SPEED AXIS(axisnum)=value, then grammar is: SPEED(axisnum)=value, and “axis” can be omitted.

Multiple BASE axis parameters can be set at the same time through SPEED=value1, value2……

Axis parameters are able to be read or written, such as, VAR1=SPEED (axisnum).

Axis status is only able to be read, value will change as per inside variation, while some unique axis status is also able to be written, such as MPOS, DPOS etc.

when in interpolation movement, parameters of main axis will be as interpolation parameters. When BASE several axes, the first axis is the main axis.

11.1 Axis Selection

BASE-Axis Selection/Axis Group Selection

Type	Axis parameter
Description	**Select axes to set parameters and to join in motion.** Default values:0, 1, 2… Before the next BASE instruction is executed, select axis based on the former BASE instruction. Every task has its own axis list, axis or the axis group selected by BASE in the task will be used to control different machines. In the interpolation motion, the first axis motion parameter is interpolation parameter. See example 1 for details. If there is no axis list in BASE, BASE will place the remaining axes in sequence. See example 2 for details.
Grammar	BASE(axis<,second axis><,third axis>...) axis: the first axis second axis: the next axis … Parameter is most as the axis amount supported by controller, please refer to relative controller hardware manuals.
Controller	General
Example	Example 1 **BASE**(0,1,2,3) 'axis list selection: 0,1,2,3

	SPEED=100,10,20,30 'axis 0,1,2,3 sets corresponding speed, but in interpolation, only speed 100 of main axis (axis 0) is valid. MOVE(100,100,100,100) 'axis 0,1,2,3 combined interpolation motion, the resultant motion speed is 100, the speed of each axis is partial speed Example 2 **BASE**(1) 'axis list selection: 1 MOVE(100,100,100) 'axis 1,2,3 do interpolation motion Example 3 **BASE**(0,2,5) 'axis list selection: 0,2,5 MOVE(100,100,100) 'axis 0,2,5 do interpolation motion

AXIS-Temporary Axis

Type	Assistant instruction
Description	**Modify a motion instruction or axis parameter temporarily to execute a defined axis.** For axis parameters, AXIS can be omitted.
Grammar	AXIS(expression) expression: new temporary modified axis, axis selection is still based on BASE instruction after finish execution.
Controller	General
Example	Example 1 BASE(0) MOVE(1000) **AXIS**(1) 'force axis 1 to move 1000units MOVE(100) 'axis 0 moves 100 units Example 2 BASE(1) UNITS **AXIS**(0)=100 'force defined axis 0 to set UNITS as 100, UNITS(0)=100 UNITS(2)=200 'force defined axis 2 to set UNITS as 200 UNITS=10 'set axis 1 UNITS as 10
Instruction	BASE

11.2 Basic Parameter Instruction

UNITS--Pulse Amount

Type	Axis Parameters

Description	**Pulse Amount**, assign pulse amount to send per unit, maximum precision is 5 decimal bits. Controller takes UNITS as basic unit, the coordinate will change with UNITS after it is modified. For Example: UNITS=10 Relative DPOS=3000, MPOS=3000 Modification: UNITS=100 Relative DPOS=300, MPOS=300
Grammar	For read: VAR1=UNITS / VR1=UNITS(axis number) For written: UNITS=expression / UNITS (axis number) = expression
Controller	General
Example	**How to set** Suppose Motor U need 3600 pulse to run one circle, and the screw pitch of guide screw p is 2mm(motor runs 1 circle, it moves 2 mm) Set relative UNITS value of 1° rotation, as below: **UNITS**=U/360=3600/360=10, now MOVE(1), 'motor runs 1° Set UNITS value of 1 mm movement, as below: **UNITS**=U/P=3600/2=1800, now MOVE(1), 'guide screw runs 1° Usually there is reduction ratio between motor and machine, suppose it is 2:1(i=2:1), then UNITS vale of 1 mm movement is: **UNITS**=U*i/P=3600*2/2=3600 **How to Program** BASE(0,1,2) 'choose axis 0,1,2. **UNITS**=10,100,1000,30 'UNITS of axis 0,1,2,3 is 10,100,1000,30 When UNITS setting axes exceed BASE list, additional UNITS value will be mapped to followed axes automatically, however, if no more than BASE list, then only set relevant axes. **UNTIS**(2)=100 'set UNITS of axis 2 directly, no influence from BASE list.

ATYPE--Axis Type

Type	Axis Parameters
Description	**Axis functions types configuration, only can set as axis types available.** **(**Find axis ATYPE in hardware manual or check in ZDevelop / RTSys.) It is better to set ATYPE before initialization. ZCAN extended axis should set AXIS_ADDRES first, and set delay 2 ticks, then call motion instructions. Due to limit of field bus bandwidth, extended axes through ZCAN should not exceed 2.

	For some products which have independent encoder inputs, then we can appoint virtual axes as encoder. For example, In ZMC206, motor axes are 0-5, then encoder axes can be 6-11, see details in ZDevelop / RTSys.
Grammar	VAR1 = ATYPE, ATYPE = expression

ATYPE	Description of Axis Type
0	Virtual-Axis
1	Stepper / Servo of Pulse Direction
2	Servo by Analog Signal Control
3	Quadrature Encoder
4	Pulse Dir OUT+ Quadrature Encoder IN
5	Pulse Dir OUT + Pulse Dir Encoder IN
6	Encoder of Pulse Dir
7	ATYPE 1 + EZ Signal IN
8	ATYPE 1 Expansion by ZCAN
9	ATYPE 3 Expansion by ZCAN
10	ATYPE 6 Expansion by ZCAN
20	SCAN-Axis with galvanometer State. Bit2 of AXISSTATUS will reset when SCAN can’t be connected, then ENCODER returns to original sending position, the unit is Pulse. Only for ZMC408SCAN.
21	SCAN-Axis, used by real controller. Default System Period: 250us, SCAN Refresh Period: 50us (Dpend on firmware). All motion control commands of ordinary axes are valid, including axis hybrid interpolation.
22	SCAN-Axis with galvanometer position feedback. Bit2 of AXISSTATUS will reset when SCAN can’t be connected, and bit3 resets when SCAN alarms. MPOS returns to measurement position, anti-correction is done. ENCODER returns to original feedback position, the unit is Pulse. Only for ZMC408SCAN.
24	Remote Encoder Axis Used in ZHD500X handwheel, some controllers support.
25	Define one encoder axis, coordinates are read from MODBUS / NODE_PDOBUFF. Valid in version after Version_build 230810. Example: **BASE(axisnum)** **AXIS_ADDRESS = (slot <<16) + nodenum** **ENCODER_ID=index<<16+subindex<<8+bites** **ATYPE=25** Slot: -1: read encoder position from MODBUS_LONG(nodenum) 0-n: read encoder position from NODE_PDOBUFF (slot, nodenum, index, subindex, type) ENCODER_ID: No. that saves data dictionary
26	Custom encoder: use C language to update encoder position, support closed-loop, valid in version after version_build 240702.

	48	SSI Absolute Encoder
	49	BISS Absolute Encoder
	50	RTEX Period Position Mode, only for RTEX controller.
	51	RTEX Period Speed Mode, only for RTEX controller.
	52	RTEX Period Torque Mode, only for RTEX controller. Please off driver 2-DOF mode, and set speed limit.
	65	EtherCAT Period Position Mode, only for EtherCAT controller.
	66	EtherCAT Period Speed Mode, only for EtherCAT controller. Note: PROFILE ≥ 20.
	67	EtherCAT Period Torque Mode, only for EtherCAT controller. Note: PROFILE ≥ 30.
	70	EtherCAT Custom, read encoder only, and only for EtherCAT controller.
	For motion mode "INVERT_STEP" instruction configuration, it is pulse direction by default.	
Controller	General	
Example	**Example 1: Pulse type** BASE(0,1) **ATYPE = 1,1** 'set axis 0,1 as pule type UNITS=100,100 'set pulse amount as 100 SPEED=100,100 'set speed as 100 units/s ACCEL=1000,1000 'set acceleration as 1000 units/s/s DECEL=1000 'set deceleration as 1000 units/s/s MOVE(100,100) 'linear interpolation **Example 2: EtherCAT Field bus control** SLOT_SCAN(0) 'scan field bus BASE(0) AXIS_ADDRESS(0)=1 'map first drive to axis 0. **ATYPE(0)=65** 'axis type is 65, position control. SLOT_START(0) 'start field bus AXIS_ENABLE=1 'axis enable WDOG=1 'enable all axes UNITS=100 'pulse amount is 100 SPEED=100 'speed 100units/s ACCEL=1000 'acceleration 1000units/s/s DECEL=1000 'deceleration 1000units/s/s MOVE(5000) **Example 3: Rtex torque mode** SLOT_SCAN(0) 'scan field bus BASE(0) AXIS_ADDRESS(0)=1 'map first drive to axis 0. **ATYPE(0)=52** 'axis type is 52, Rtex torque mode. DRIVE_WRITE(6*256+47,0) 'close 2 DOF control. DRIVE_WRITE(3*256+17,0) 'choose parameter 3.21 as speed limit.	

	DRIVE_WRITE(3*256+21,2000) 'maximum speed limit is 2000r/min
	SLOT_START(0) 'start Field bus
	AXIS_ENABLE=1 'axis enable
	WDOG=1 'enable all axes
	DAC=100 'send control value by DAC, see DAC for details.
	
	Example 4: galvanometer axis
	BASE(4,5)
	UNTIS=1,1
	ATYPE=21,21 'set as galvanometer axis
	
	Example 5: remote encoder axis
	BASE(axisnum)
	AXIS_ADDRESS = lcd NO
	ATYPE=24
	
	Example 6: define one encoder axis
	BASE(0)
	AXIS_ADDRESS = (0<<16) + 0
	ENCODER_ID = $60640020
	ATYPE = 25
	
	Example 7: custom encoder
	Added C function interface:
	// read DAC_OUT, rea axis DAC output value
	double motionrt_getaxisdacout(uint32 iaxis);
	
	// virtual, custom encoder update “encoder”
	uint32 motionrt_updatecoder(uint32 iaxis, int32 icoder);
	
	// virtual, custom encoder update “encoderdot”, float -1 to 1
	// usually don’t call
	uint32 motionrt_updatecoderdot(uint32 iaxis, float fdot);
	
	BASE(0)
	ENCODER_SERVO = 2 ‘configure closed-loop, AOUT output is not used, read output by motionrt_getaxisdaout
	ATYPE=26
	
	‘closed-loop processing, please refer to <<C Language Support>>
	datum(0)
	FE_LIMIT = 10000
	FE_RANGE = 10000
	axis_enable = 1
	servo = 1

Instructions	AXIS_ADDRESS, INVERT_STEP

AXIS_ADDRESS--Axis Address Configuration

<table>
<tr><td>Type</td><td>Axis Parameters</td></tr>
<tr><td>Description</td><td>
Axis address configuration of extended axes.
1. When the axis extended by ZCAN, there is one 8-code DIP switch (hardware version should be above V1.3)
Due to limit of ZCAN bandwidth, extended axes should not exceed 2.
Do set AXIS_ADDRESS first, then set ATYPE of extended axes. After modification, ATYPE must be reset.
see example one for reference.
<table>
<tr><td>Bit 1-4</td><td>CAN address DIP code, combination value is 0-15</td></tr>
<tr><td>Bit 5-6</td><td>CAN speed DIP code, different values have different speed.</td></tr>
<tr><td>Bit 7</td><td>Special function: Reserved</td></tr>
<tr><td>Bit 8</td><td>120ohm resistor DIP code, be connected when ON.</td></tr>
</table>
Rule：
AXIS_ADDRESS(axis NO.)=(32*0)+CAN ID
‘local axis0 of expansion module.
AXIS_ADDRESS(axis NO.)=(32*1)+CAN ID
‘local axis1 of expansion module.
2. Bus driver axis No. mapping, map connected drives correspondingly according to No. sequence.
Drive No. is sorted by connecting sequence, it ranges from 0 to EtherCAT drive number - 1.
Drive No. is different from device No., device No. includes all connected devices, but drive No. only includes drives.
Do set AXIS_ADDRESS first, then set ATYPE. After modification, ATYPE must be reset.
See Example two for reference
<table>
<tr><td>Bit 0-15</td><td>Drive No.+1, 0-Auto Assign</td></tr>
<tr><td>Bit 16-31</td><td>Slot No. (when there are multiple slots)</td></tr>
</table>
Rule:
AXIS_ADDRESS(Axis No.)=(Slot No.<<16)+Drive No.+1
3. Local pulse axis No. remapping, 4 series motion controllers support local pulse axis or encoder axis No. remapping, please note the firmware should be above 160608.
While remapping, set ARTPE of original local pulse axis as virtual axis. After modification, ATYPE must be reset.
see example three as reference.
<table>
<tr><td>Bit 0-15</td><td>Mapped local pulse axis No.</td></tr>
<tr><td>Bit 16-31</td><td>High 16-bit are set as 1 (under decimal system, high 16-bit = -1).</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>Rule：
BASE(axis No. to be remapped)
ATYPE=0 set axis type as 0, low version will report errors if not be set.
BASE(local axis No. to be modified)
ATYPE=0 set axis type as 0
AXIS_ADDRESS(remapped axis No.)= (-1<<16) + local pulse axis No. to be modified
BASE (axis No. to be remapped)
ATYPE=1/7

4. Pulse-axis, encoder axis (sub-card on MotionRT control card) mapping. While mapping, it must set AXIS_ADDRESS at first, then set ATYPE. If there is modification, please reset ATYPE.
<table><tr><td>Bit 0-15</td><td>On sub-card, Axis No. + 1</td></tr><tr><td>Bit 16-31</td><td>Sub-card CARD No.</td></tr></table>Rule:
BASE (axis No. that is to be remapped)
ATYPE = 0, set axis type as 0, it will report error if there is no setting in low version.
BASE (local pulse axis No. that is to be modified)
ATYPE = 0, set axis type as 0.
AXIS_ADDRESS (axis No. to be remapped) = (sub-card No. << 16) + physical axis No. on sub-card + 1
BASE (axis No. that is to be remapped)
ATYPE=X (reset required axis type)

5. Cancel axis mapping: AXIS_ADDRESS = 0
BASE (remapped axis No.)
ATYPE (remapped axis No.) = 0
AXIS_ADDRESS = 0</td></tr>
<tr><td>Grammar</td><td>VAR1 = AXIS_ADDRESS, AXIS_ADDRESS = expression</td></tr>
<tr><td>Controller</td><td>General</td></tr>
<tr><td>Example</td><td>Example 1: ZCAN expansion-axis
AXIS_ADDRESS (6)=2+(32*1) 'map axis 6 to axis 1 of ID2 on ZCAN module
ATYPE(6)=8 'set ZCAN extended axis type, stepper or servo in pulse direction
UNITS(6)=100 'pulse amount 100
SPEED(6)=100 'speed is 100uits/s
ACCEL(6)=1000 'acceleration is 1000units/s/s
MOVE(100) AXIS(6) 'extended axis moves 100units

Example 2: EtherCAT axes mapping by Manual
AXIS_ADDRESS(0)=0+1 'first Ecat drive, No. is 0, mapped as axis 0
AXIS_ADDRESS(2)=1+1 'second Ecat drive, No. is 1, mapped as axis 2</td></tr>
</table>

	AXIS_ADDRESS(1)=2+1 'third Ecat drive, No. is 2, mapped as axis 1 ATYPE(0)=65 'set as Ecat type ATYPE(1)=65 ATYPE(2)=65 **Example 3: EtherCAT axes mapping automatically** **AXIS_ADDRESS** (0)=0 'automatically specify slot0 drive, the start to map axis No. from axis 0 according to the connection sequence (not recommended in this way. example 2 is better) ATYPE(0)=65 'axis 0 is set as ECAT mode **Example 4: change pulse axis No. of EtherCAT controller.** 'before change, operate axis 0 (axis 0 interface on controller) BASE(16) 'axis No. that is remapped ATYPE(16)=0 BASE(0) 'the local pulse axis No. to be modified ATYPE=0 'set local pulse axis 0 as virtual axis **AXIS_ADDRESS** (16)= (-1<<16)+0 'bind with local pulse axis 0, high16 bits = -1. ATYPE(16)=1 'set axis 16 as pulse axis, use local pulse axis 0. Then, at this time, operate axis 0, corresponding to ECAT encoder, operate axis 16, corresponding to controller axis 0 port. **Example 5: galvanometer axis remapping** ATYPE(4)=0 ATYPE(5)=0 BASE(X) 'axis NO. to be mapped **AXIS_ADDRESS** = (-1<<16)+4 'remap the first SCAN axis ATYPE = 21 BASE(Y) 'axis No. to be mapped **AXIS_ADDRESS** = (-1<<16)+5 'remap the second SCAN axis ATYPE = 21 **Example 6: axis of sub-card on MotionRT control card remapping** 'remap axis 0 as axis 16 BASE(16) 'axis No. to be remapped ATYPE=0 BASE(0) 'original axis No. to be modified ATYPE=0 'set axis type as 0 AXIS_ADDRESS(16)=(0<<16)+ 0 + 1 BASE(16) ATYPE =1 'configure axis 16 as pulse axis
Instructions	ATYPE

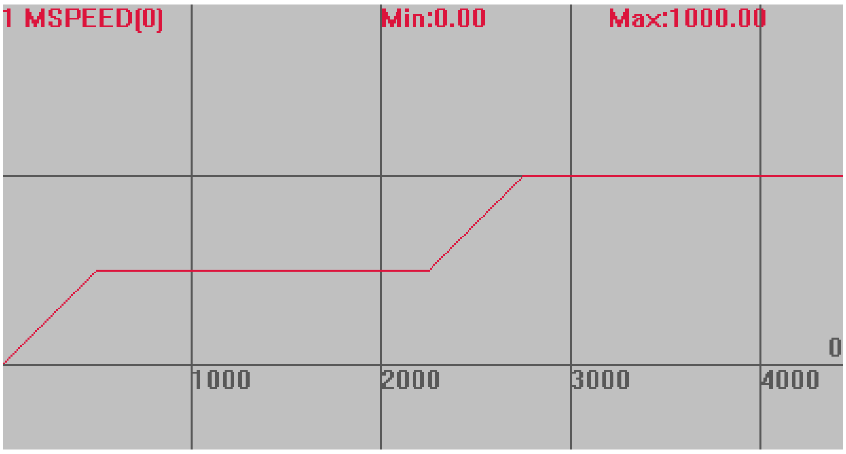
AXIS_ENABLE--Axis Enable

Type	Axis Parameters
Description	**Enable each axis.** EtherCAT Bus axis should be configured, and WDOG=1 general enable must be set.
Grammar	AXIS_ENABLE = 1/0， 1-open enable,0-close enable.
Controller	General
Example	**AXIS_ENABLE**(0) = 1 'open axis 0 enable
Instruction	WDOG

11.3 Speed Parameter Instruction

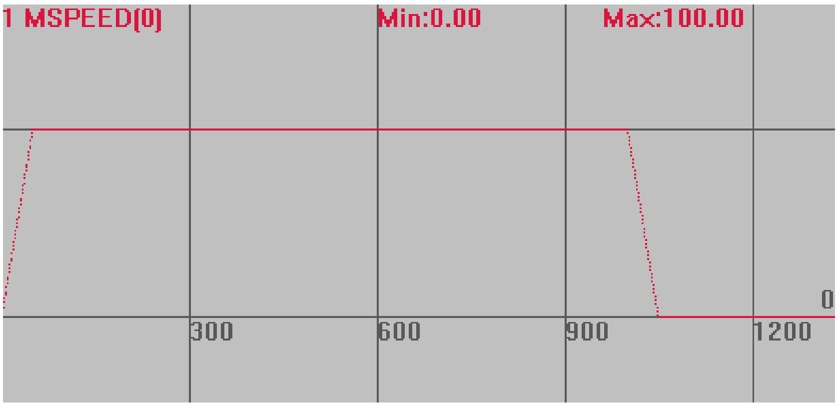
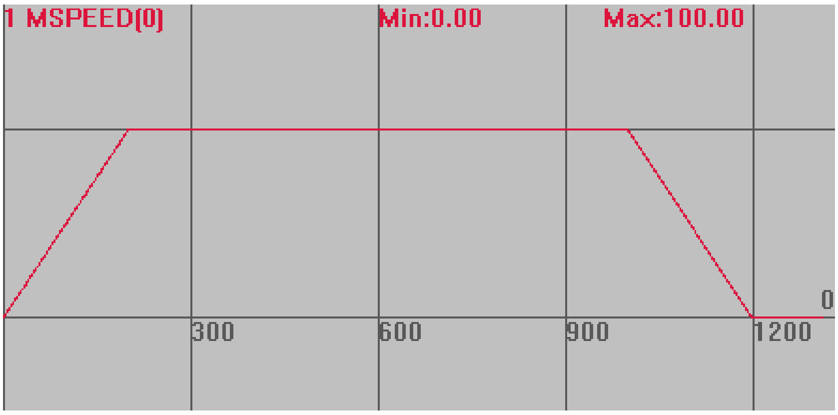
SPEED--Motion Speed

Type	Axis Parameters
Description	**Axis speed, unit is units/s.** When multi-axis is in motion, SPEED as interpolation motion speed. After modification of SPEED, SPEED will take effect immediately, dynamic speed changing can be done in this way, but the moment of speed changing, it may cause speed jumping, which will also cause machine vibration, then we can use SPEED_RATIO to realize smooth speed changing. When SP instruction: FORCE_SPEED is more than SPEED, SPEED will also take effect. (SPEED will not take effect in this situation in firmware version above 140716).
Grammar	VAR1 = SPEED, SPEED = expression
Controller	General
Example	BASE(0) UNITS=100 'pulse amount **SPEED** =500 'set speed of axis 0 as 500units/s ACCEL=1000 'acceleration:1000units/s/s DPOS=0 'coordinate clears TRIGGER 'trigger oscilloscope automatically VMOVE(1) 'continuous motion WAIT UNTIL DPOS(0)>1000 'wait until axis 0 reaches 1000. **SPEED**=1000 'change speed as 1000 Speed Curve: MSPEED(0)=1000(vertical scale)

	
Instruction	FORCE_SPEED, SPEED_RATIO

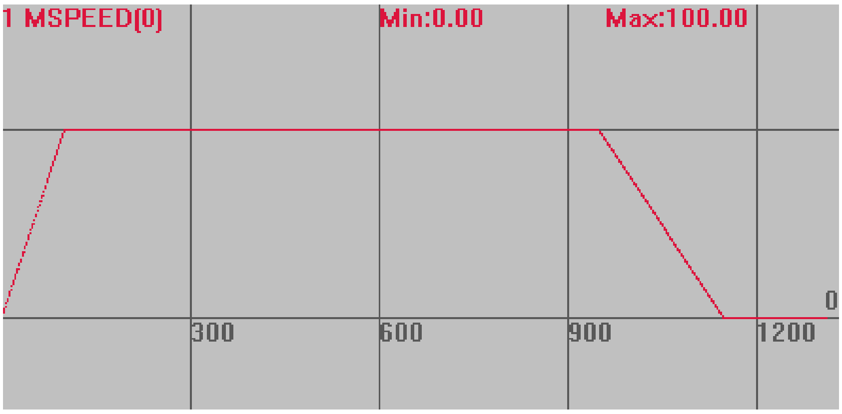
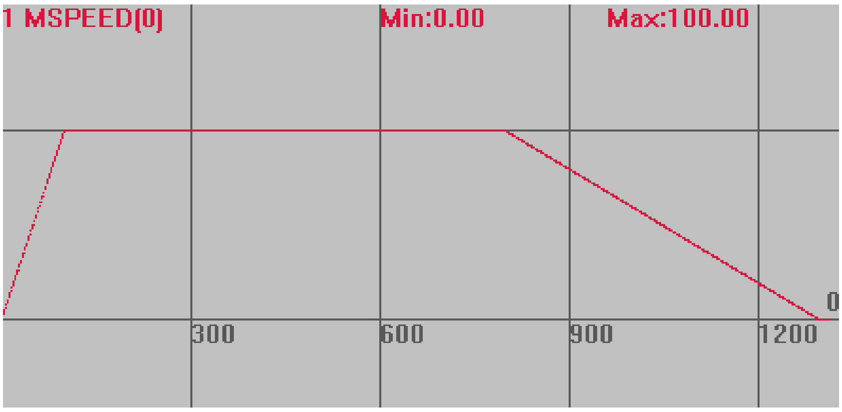
ACCEL--Axis Acceleration

Type	Axis Parameters
Description	**Axis acceleration, unit is units/s/s.** In multi-axes motion, acceleration of interpolation motion will obey main axis. It is better to set acceleration and deceleration before motion starts, and don't change in motion, or will cause change of speed curve.
Grammar	To read:VAR1=ACCEL(axis number) To write: ACCEL(axis number) = expression
Controller	General
Example	**Example 1** BASE(1,2,3,4) 'BASE select axis **ACCEL**=100, 100, 100, 100 'set acceleration of axis 1,2,3,4 ACCEL(2)=200 'set acceleration of axis 2. **Example 2** BASE(0) UNITS=100 'pulse amount DPOS=0 'coordinate clears **ACCEL**=2000 SPEED=100 MOVE(100) Speed curve of Acceleration Process MSPEED(0)=100(vertical scale)

	 Speed curve after acceleration and deceleration **ACCEL**=500 DECEL=500 
Instruction	DECEL, SPEED

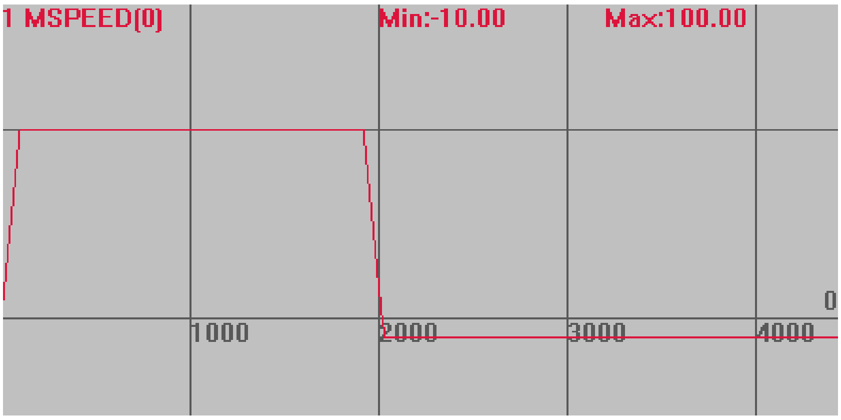
DECEL--Axis Deceleration

Type	Axis Parameters
Description	**Axis deceleration, unit is units/s/s.** In multi-axes motion, deceleration of interpolation motion will obey main axis. When it is set as 0, it will get value of ACCEL, then deceleration and acceleration will be symmetric. It is better to set acceleration and deceleration before motion starts,and don't change in motion,or will cause change of speed curve.
Grammar	VAR1 = DECEL, DECEL = expression
Controller	General
Example	**Example one** BASE(1,2,3,4) **DECEL** =100, 100, 100, 100 'set deceleration of axis 1,2,3,4. **DECEL** (0)=200 'set deceleration of axis 0.

	PRINT DECEL (0) **Example two** BASE(0) 'choose axis 0. SPEED=100 'set speed as 100 units/s **DECEL**=500 'deceleration is 500units/s/s TRIGGER 'trigger oscilloscope automatically MOVE(200) 'move 200units Speed Curve MSPEED(0) vertical scale 100  DECEL=200 
Instruction	ACCEL, FASTDEC

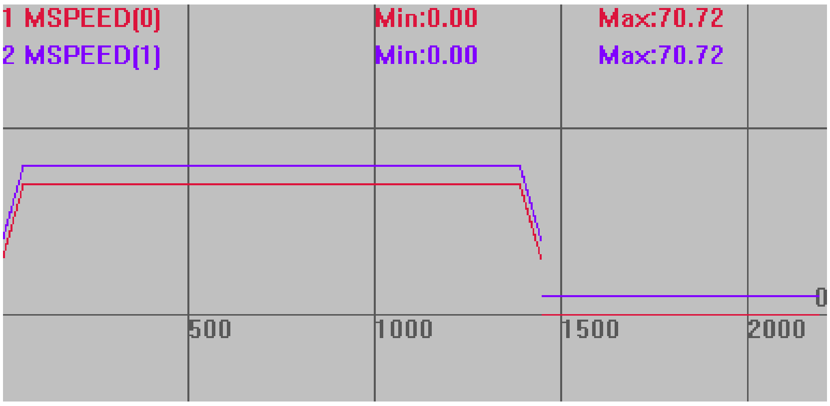
CREEP--Creep Speed

Type	Axis Parameters
Description	**Axis creep speed while homing, which is used to search origin point,** unit is units/s.
Grammar	VAR1 = CREEP, CREEP = expression
Controller	General
Example	BASE(0)

	UNITS=100 ACCEL=1000 DECEL=1000 SPEED = 100 **CREEP** = 10 'set creep speed as 10units/s DATUM_IN=0 'set IN0 as origin point of axis 0 INVERT_IN(0,ON) 'invert electric level. TRIGGER 'trigger oscilloscope automatically DATUM(3) 'search origin point at speed of 100, leave at speed of 10 after meeting origin point. Speed Curve MSPEDD(0) vertical scale 100 
Instruction	DATUM

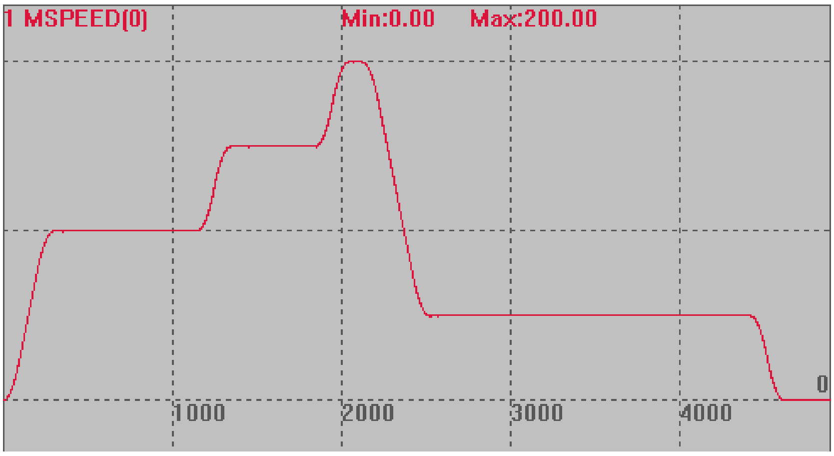
LSPEED--Initial Speed

Type	Axis Parameters
Description	**Axis starting speed, also can be used as stop speed, default value is 0, unit is units/s** As the starting speed of interpolation in multi-axis motion. When the motion needs efficiency, LSPEED starting speed can be set. Please note in most of applications, LSPEED value is recommended to be 0, otherwise, it may cause severe shake.
Grammar	VAR1 = LSPEED, LSPEED = expression
Controller	General
Example	Example BASE(0,1) 'select axis 0 as main axis DPOS=0,0 UNITS=100,100 'pulse amount 100 SPEED=100,100 'main axis speed 100units/s ACCEL=1000,1000 DECEL=1000,1000

	LSPEED=40 'initial speed 40 units/s TRIGGER 'trigger oscilloscope automatically MOVE(100,100) 'motion distance of per axis Speed Curve MSPEED(0)=100(vertical scale), no offset MSPEED(1)=100(vertical scale), offset 10 
Instruction	SPEED

FORCE_SPEED--SP Speed

Type	Axis Parameters
Description	**Forced speed of self-defined speed SP motion, unit is units/s.** This parameter will enter buffer. When FORCE_SPEED is bigger than SPEED, then SPEED value will also limit the maximum speed in motion. (SPEED will not take effect after firmware 140716) If need FORCE_SPEED to decrease to required value before a new motion segment, then set STARTMOVE_SPEED.
Grammar	VAR1 = FORCE_SPEED, FORCE_SPEED = expression
Controller	General
Example	BASE(0) DPOS=0 UNITS=100 'pulse amount100 ACCEL=500 DECEL=500 SPEED = 100 'speed is 100units/s **FORCE_SPEED**=150 'self-defined speed is 150units/s SRAMP=100 'S curve MERGE=ON TRIGGER= 'trigger oscilloscope automatically MOVE(100) 'normal motion without SP MOVESP(100) 'speed is 150

	FORCE_SPEED=200 MOVESP(100) 'speed is 200 **FORCE_SPEED=50** MOVESP(100) 'speed is 50 END Speed Curve: MSPEED(0)=100(vertical scale) 
Instruction	*SP

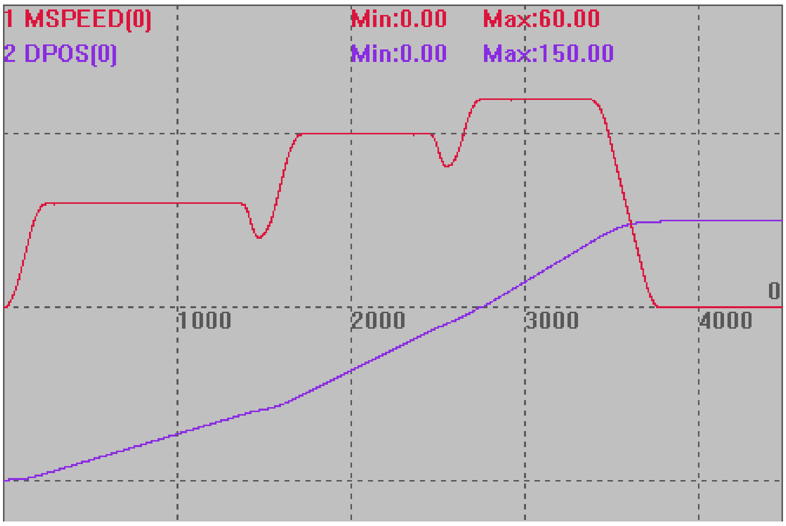
STARTMOVE_SPEED--Start Speed of SP Motion

Type	Axis Parameters
Description	**Starting speed of SP motion, this parameter will enter buffer.** Only valid in motion instruction with SP. Set a big value when this instruction is not required any more. Default value of controller is 1000.
Grammar	VAR1=STARTMOVE_SPEED, STARTMOVE_SPEED=expression
Controller	General
Example	RAPIDSTOP(2) WAIT IDLE(0) BASE(0) ‘select XY axis DPOS = 0 MPOS = 0 ATYPE = 1 ‘pulse stepper / servo UNITS = 100 ‘pulse amount SPEED = 100 ACCEL = 200 DECEL = 200 SRAMP = 100 ‘S curve MERGE = ON ‘open continuous interpolation

<table>
<tr><td></td><td>‘first segment
FORCE_SPEED = 30　　　　　‘the first segment speed is 30
STARTMOVE_SPEED = 1000　‘not set, default value is 1000
ENDMOVE_SPEED = 1000　　‘not set, default value is 1000
MOVESP(40)

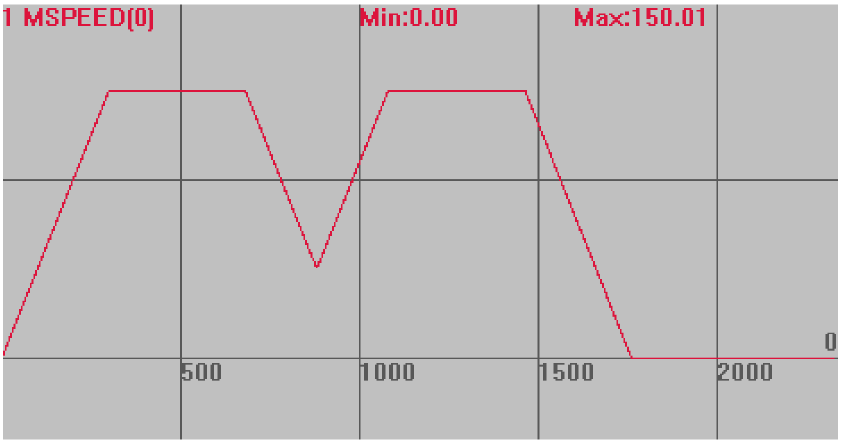
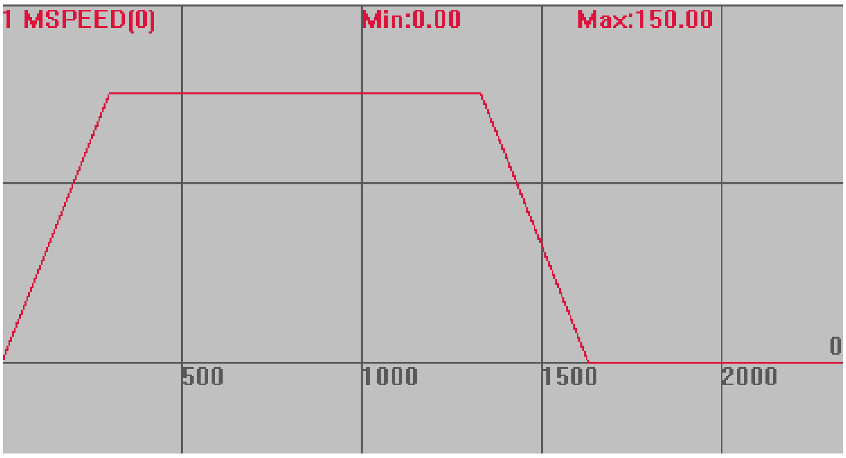
‘second segment
FORCE_SPEED = 50　　　　　‘the second segment speed is 50
STARTMOVE_SPEED = 20　　‘the second segment’s initial speed is 20
ENDMOVE_SPEED = 40　　　‘end speed is 40
MOVESP(50)

‘third segment
FORCE_SPEED = 60　　　　　‘the third segment speed is 60
STARTMOVE_SPEED = 1000　‘not set, default value is 1000
ENDMOVE_SPEED = 1000　　‘not set, default value is 1000
MOVESP(60)
END

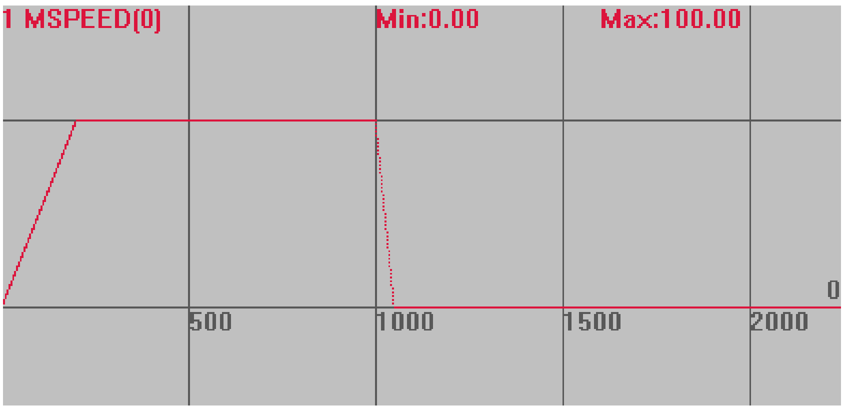
Speed & Position Curve:
MSPEED(0) – vertical scale 50
DPOS(0) – vertical scale 100, offset -100

</td></tr>
<tr><td>Instructions</td><td>FORCE_SPEED, *SP,ENDMOVE_SPEED</td></tr>
</table>

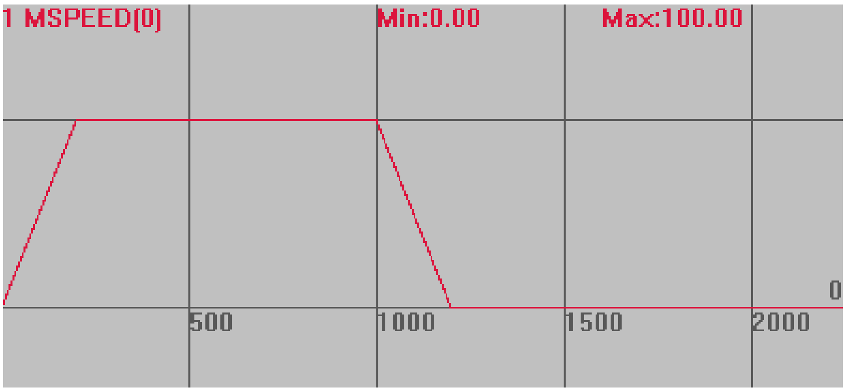
ENDMOVE_SPEED--End Speed of SP motion

Type	Axis Parameters
Description	**End speed of self-defined speed SP motion, this parameter will enter motion buffer.** Only valid when SP motion instructions are used. Set a big value when not used. Default value of controller is 1000.

Grammar	VAR1 = ENDMOVE_SPEED, ENDMOVE_SPEED = expression
Controller	General
Example	BASE(0) DPOS=0 UNITS=100 MERGE=1 'open continuous interpolation SPEED=100 ACCEL=500 DEVEL=500 FORCE_SPEED=150 'limit speed is 150units/s **ENDMOVE_SPEED**=50 'forced end speed is 50units/s TRIGGER 'trigger oscilloscope automatically MOVESP(100) MOVESP(100) Speed Curve with speed limit: MSPEED(0) vertical scale 100  Speed Curve without speed limit (END_SPEED) 
Instruction	FORCE_SPEED, *SP, STARTMOVE_SPEED

FASTDEC--Fast Deceleration

Type	Axis Parameters
Description	**Fast deceleration, unit is units/s/s.** Activated automatically when CANCEL is used and position limit or unusual stop happens. When set value is 0 or less than DECEL, then will set as DECEL automatically.
Grammar	VAR1 = FASTDEC, FASTDEC= expression
Controller	General
Example	BASE(0) 'select axis 0 DPOS=0 UNITS=100 SPEED=100 'set speed as 100 ACCEL=500 DECEL=500 'set deceleration as 500 **FASTDEC**=2000 'set fast deceleration as 2000 TRIGGER 'trigger oscilloscope automatically VMOVE(1) 'continuous positive motion DELAY (1000) 'wait 1 second CANCEL(2) 'motion stops Deceleration Curve MSPEED(0) vertical scale 100  When FASTDEC=10, use DECEL to decelerate.

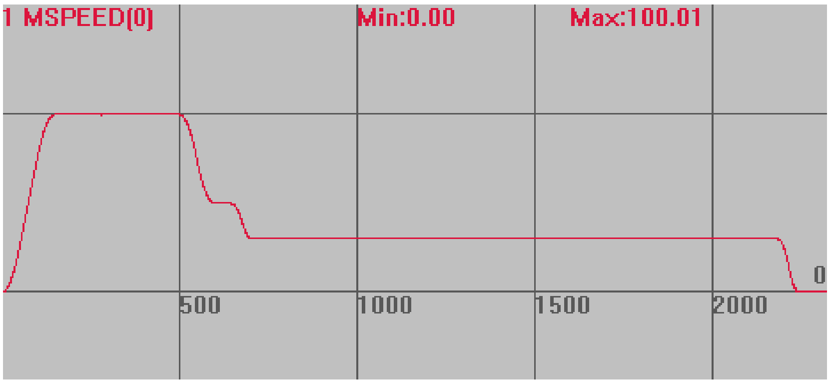
	
Instruction	DECEL

MSPEED--Actual Speed Feedback

Type	Axis Status
Description	**Measured speed feedback of axis, unit is units/s.**
Grammar	VAR1 = MSPEED
Controller	General
Instruction	UNITS, VP_SPEED

SPEED_RATIO--Speed Proportion

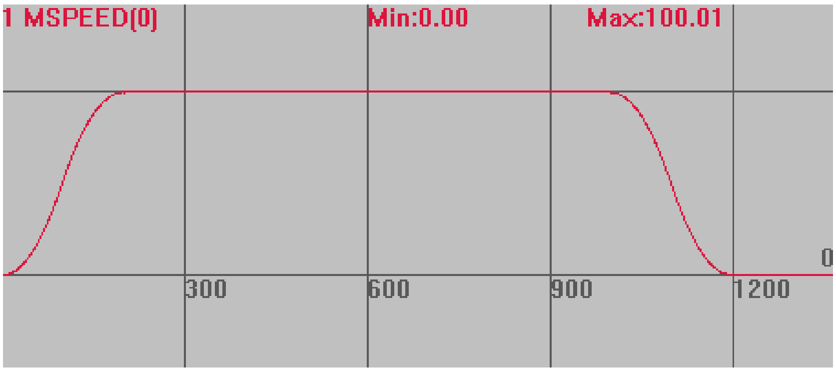
Type	Axis Parameters
Description	**Axis speed proportion ratio:0-1.** Actual axis speed=SPEED*SPEED_RATIO. It is used to smooth change speed of motion in process based on acceleration or deceleration.
Grammar	SPEED_RATIO (axis number) = value value: ratio is 0-1 If not assign axis NO., use defined axis NO. by BASE instruction default. Interpolation motion can be used in all axes, or only be valid in the first axis of BASE. When online command without axis NO., be valid in axis 0 by default.
Controller	Controller with latest hardware version
Example	RAPIDSTOP(2) WAIT IDLE **SPEED_RATIO** = 1 TRIGGER BASE(0) 'select axis 0 DPOS = 0 UNITS = 100

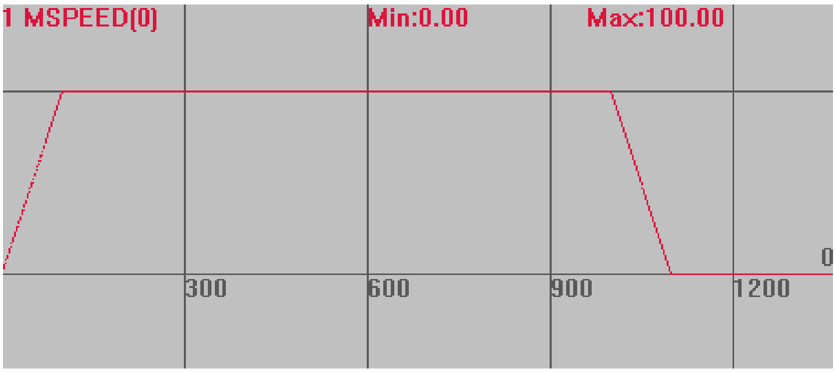
	SPEED = 100 ACCEL = 1000 DECEL = 1000 MERGE = ON SRAMP = 50 MOVE(100) DELAY(500) 'wait 0.5s **SPEED_RATIO** = 0.5 'speed decrease to 50 WAIT UNTIL VP_SPEED < 80 DELAY(100) 'wait 0.1s **SPEED_RATIO** = 0.3 'speed decrease to 30 END Speed Curve VP_SPEED(0) vertical scale 100 
Instruction	FORCE_SPEED,SPEED

SRAMP--Acceleration Curve

Type	Axis Parameters
Description	**S curve setting of acceleration or deceleration process.**
Grammar	VAR1=SRAMP, SRAMP=smoothms smoothms 0-250ms, acceleration or deceleration process time will increase after setting.
Controller	General
Example	BASE(0) 'select axis 0 DPOS=0 UNITS=100 'pulse amount is 100 SPEED=100 'speed is 100units/s ACCEL=1000 'acceleration is 1000units/s/s DECEL=1000 'deceleration is 1000units/s/s **SRAMP**=100 'S curve time is 100ms TRIGGER 'trigger oscilloscope automatically

<table>
<tr><td></td><td>MOVE(100)   'move 100units

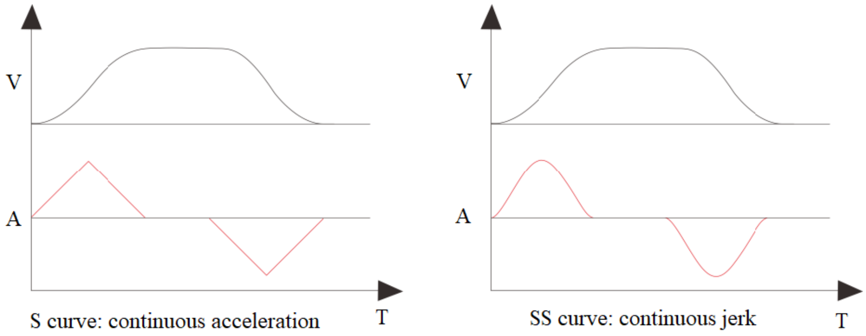
Speed Curve
MSPEED(0) vertical scale 100


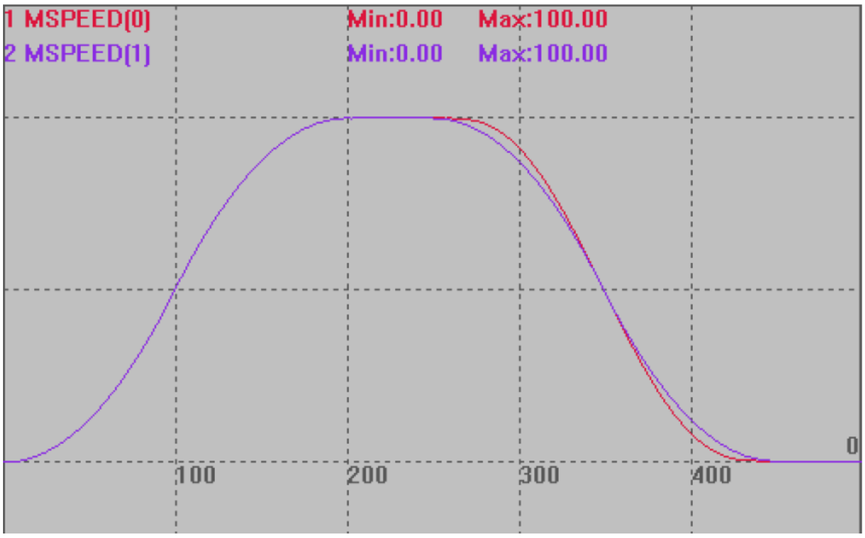
When SRAMP=0


</td></tr>
<tr><td>Instruction</td><td>ACCEL,DECEL</td></tr>
</table>

VP_MODE—Acceleration & Deceleration Curve

Type	Axis Parameters
Description	**Acceleration and deceleration curve's type selection:** 0: default value, use sramp to set S curve. 4: at the very beginning of motion, it uses the max acceleration, then acceleration will gradually decrease to 0 when achieving the highest speed. 6: new added SS curve, which belongs to the curve type of jerk continuity. Deceleration time under SS mode will be more 87% than T mode. Mode 0 is used in this mode's acceleration stage, but it will take effect until decelerating, in this way, continuous small segment interpolations are easy to achieve. 7: new added SS curve, which belongs to the curve type of jerk continuity. If axis parameters or continuous interpolations are modified dynamically, maybe jerk can't be realized, then it will switch to mode 0, therefore,

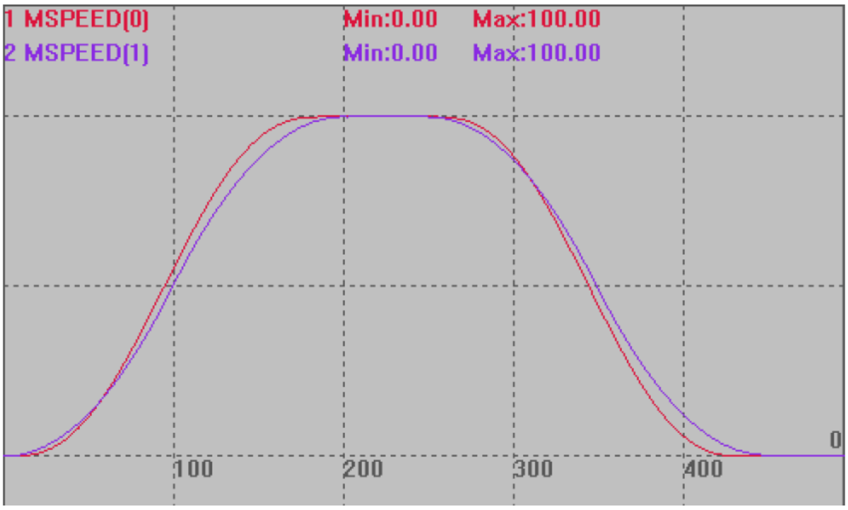
	SRAMP is recommended to set a suitable value. VP_MODE and SRAMP both can smooth the "speed" parameter, followings show difference: 
Grammar	VAR1=VP_MODE / VP_MODE(axis) = mode mode: select mode
Controller	General
Example	**Example 1: mode 6** BASE(0) 'select axis 0 and axis 1 ATYPE=1,1 UNITS=100,100 DPOS=0,0 MPOS=0,0 SPEED=100,100 ACCEL=1000,1000 DECEL=1000,1000 SRAMP=100,100 VP_MODE=6,0 'axis 0 with mode 6, axis 1 with mode 0 TRIGGER MOVE(25) AXIS(0) MOVE(25) AXIS(1) Speed Curve: under mode 6, acceleration stage is not processed, it is only for deceleration. MSPEED(0) vertical scale 50 MSPEED(1) vertical scale 50



Example 2: mode 7

VP_MODE=7,0 'axis 0 with mode 7, axis 1 with mode 0

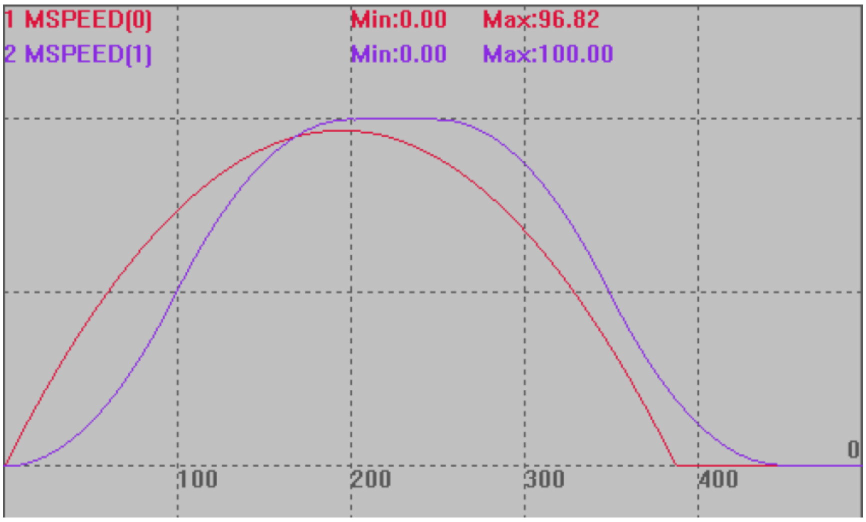
Others are same as example 1, mode 7 processed both acceleration and deceleration stages.



Example 3: mode 4

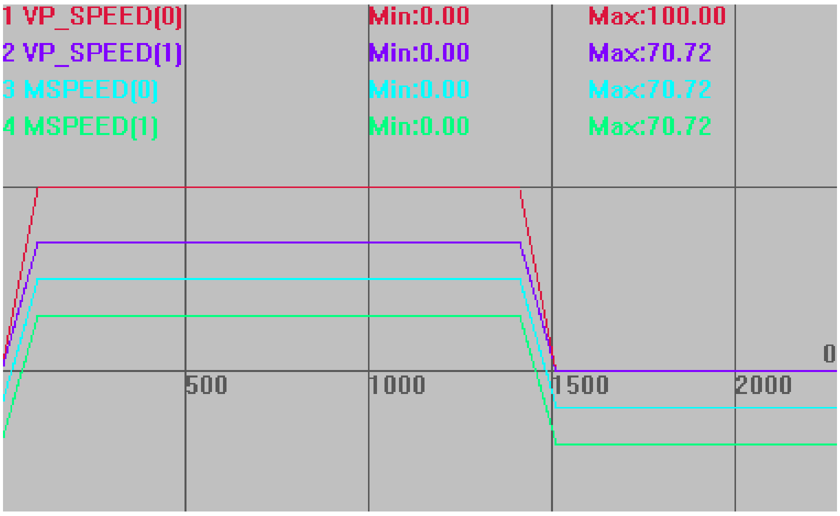
VP_MODE=4,0 'axis 0 with mode 4, axis 1 with mode 0

Others are same as example 1, the max acceleration is at the very beginning of motion, and acceleration will decrease to 0 when achieving the highest speed.

	
Instruction	MSPEED, SPEED, SRAMP

VP_SPEED--Present Motion Speed

Type	Axis Status
Description	**Return present axis motion speed, unit is units/s.** In terms of muti axes interpolation motion, returned speed of main axis is interpolation resultant speed, not component speed of main axis. Returned speed of non-main axis is relevant component speed, the same as MSPEED. VP_SPEED is designed to show multi-axis resultant speed, no minus value, except set SYSTEM_ZSET bit0 value as 0, in this way, it shows single axis speed, and it can be positive or minus value.
Grammar	VAR1 = VP_SPEED
Controller	General
Example	BASE(0,1) DPOS=0,0 'coordinate clears UNITS=100,100 'pulse amount SPEED=100,100 'main axis speed is 100units/s ACCEL=1000,1000 'acceleration is 1000units/s/s DECEL=1000,1000 'deceleration is 1000units/s/s TRIGGER 'trigger oscilloscope automatically MOVE(100,100) 'two axes move 100units respectively Speed Curve: VP_SPEED of main axis is interpolation resultant speed. VP_SPEED of non-main axis is relevant component speed, the same as MSPEED. VP_SPEED(0)=100(vertical scale), no offset VP_SPEED(1)=100(vertical scale), no offset

	MSPEED(0)=100(vertical scale), offset -20 MSPEED(1)=100(vertical scale), offset -40 
Instruction	MSPEED,SPEED

INTERP_FACTOR--Interpolation Speed

Type	Axis Parameters
Description	**Axis participates speed calculation or not, default: participate (1).** This parameter only valid for any axis in linear interpolation or third axis in helical interpolation. Do cancel after motion, or will cause incorrectness to followed motion. When some axes don’t participate speed calculation, calculate out component speed and total motion time of axis which participate interpolation motion, then speed of axis which don’t participate calculation = motion distance/total motion time. See Example Two for reference. Don’t set INTERP_FACTOR of all axes as 0, or will cause infinite actual speed.
Grammar	INTERP_FACTOR=0/1 0-not participate calculation 1-participate calculation
Controller	General
Example	**Example one: All axes participate in speed calculation** BASE(0,1,2) 'axis 0 as main axis DPOS=0,0,0 ATYPE=1,1,1 UNITS=100,100,100 'pulse amount:100 SPEED=100,100,100 'main axis speed:100units/s ACCEL=1000,1000,1000 DECEL=1000,1000,1000 **INTERP_FACTOR=1,1,1** 'axis 0,1,2 participate speed calculation. TRIGGER 'trigger oscilloscope automatically

MOVE(100,200,300)   'component distance of each axis

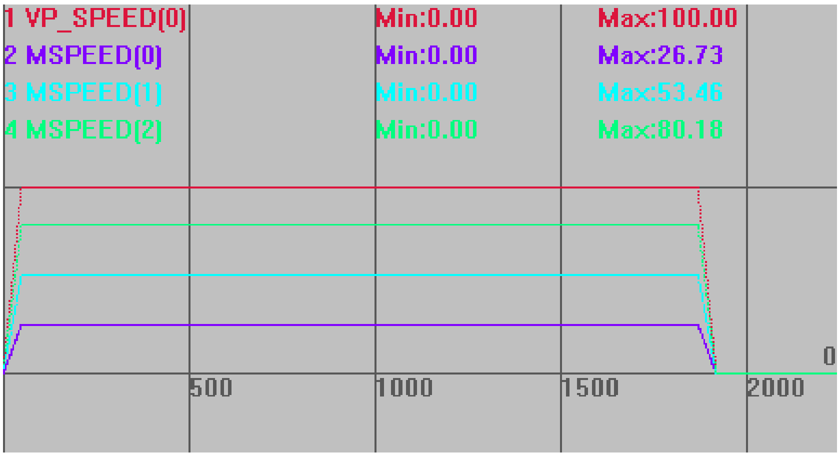
Calculate out component speed of each axis based on resultant motion speed:100.

VP_SPEED(0)=100(vertical scale)

MSPEED(0)=100(vertical scale)

MSPEED(1)=100(vertical scale)

MSPEED(2)=100(vertical scale)

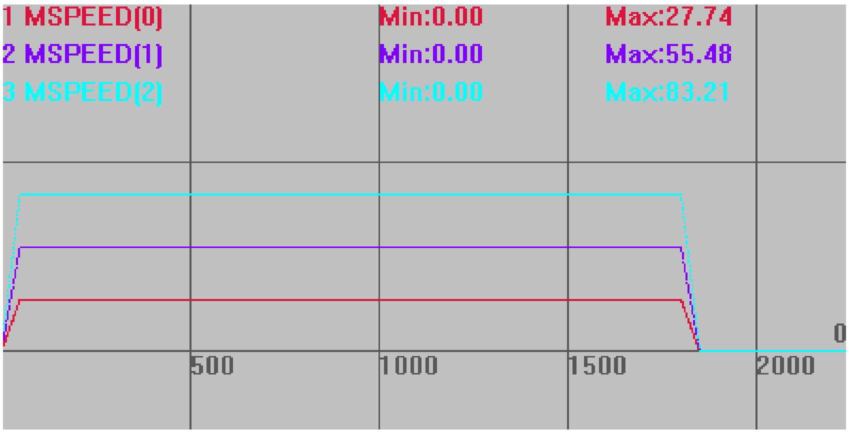


Example Two: Some axes don't participate speed calculation

INTERP_FACTOR=0,1,1   'axis 0 don't participate speed calculation.

Calculate out component speed and total motion time of axis 2 and axis 3, then speed of axis 0=motion distance of axis 0/total motion time.

Scale same as the former.

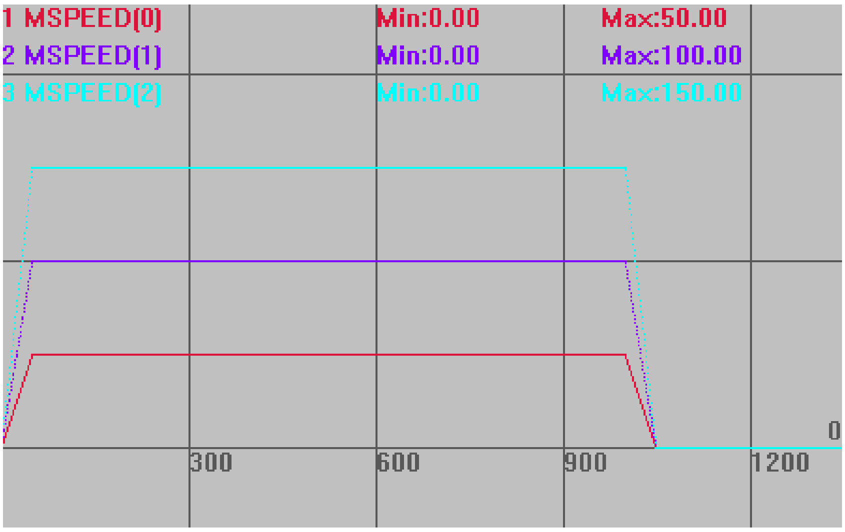


Example Three: only one axis participates speed calculation

INVERT_FACTOR=0,1,0   'only axis 1 participates speed calculation.

Axis 1 in main axis in this situation, speed is 100, total motion time is 200/100, speed of axis 0 and axis 2=motion distance/total motion time.

Vertical scale same as the former example.

	
Instruction	BASE_MOVE

CORNER_ACCEL – Corner Acceleration

Type	Axis Parameter
Description	**Corner acceleration, the unit is units/s/s.** Used to set curve deceleration, default is 0 (not take effect), after setting, replace FULL_SP_RADIUS. When CORNER_MODE sets as “apart mode”, each axis’ set corner acceleration all take effect. Recommend use together with ZSMOOTH_MODE to smooth the speed and curve. Please refer to each axis’ acceleration limit, set machine real allowed corner acceleration.
Grammar	To read: VAR1 = CORNER_ACCEL (axis No.) To write: CORNER_ACCEL (axis No.) = expression
Controller	Valid in ZMC4XX controller’s fast firmware, after 230926.
Example	SPEED = 500, 500, 500, 2000, 313 ACCEL = 8000, 5000, 5000, 4000, 4200 **CORNER_ACCEL** = 5000, 2000, 3000, 3000, 3000
Instruction	ACCEL, SPEED

11.4 Axis Status Checking Instruction

MTYPE--Type of Present Motion

Type	Axis Status

Description	**Type of present motion in process.** In terms of interpolation motion, slave axis always returns to master axis.
Grammar	VAR1 = MTYPE

MTYPE	Motion Type
0	IDLE (no motion)
1	MOVE
2	MOVEABS
3	MHELICAL
4	MOVECIRC
5	MOVEMODIFY
6	MOVESP
7	MOVEABSSP
8	MHELICALSP
9	MOVECIRCSP
10	FORWARD, VMOVE(1)
11	REVERSE, VMOVE(-1)
12	DATUMING
13	CAM
14	FWD_JOG
15	REV_JOG
16	MOVESYNC
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

	<table><tr><td></td><td>MOVE_PARA MOVE_PWM MOVE_ASYNMOVE MOVE_AOUT</td></tr><tr><td>29</td><td>MOVE_DELAY MOVE_WAIT MOVE_SYNMOVE</td></tr><tr><td>31</td><td>MSPHERICAL/ MSPHERICALSP</td></tr><tr><td>32</td><td>MOVE_PT</td></tr><tr><td>33</td><td>CONNFRAME</td></tr><tr><td>34</td><td>CONNREFRAME</td></tr></table>
Controller	General

Example	WHILE 1 'cycle judgment
	IF **MTYPE**=0 THEN
	?"no motion"
	ELSEIF **MTYPE**=1 THEN
	?"Linear Interpolation"
	ELSEIF **MTYPE**=4 THEN
	?"Circular Interpolation"
	…
	ENDIF
	WEND

Instruction	NTYPE, REMAIN_BUFFER

NTYPE--Motion Type of Next Motion

Type	Axis Status
Description	**The next motion type of present motion instruction.** In terms of interpolation motion, slave axis always returns to master axis.
Grammar	VAR1 = NTYPE
Controller	General

Example	WHILE 1 'cycle judgment
	IF **NTYPE**=0 THEN
	?"End the motion"
	ELSEIF **NTYPE**=1 THEN
	?"Linear Interpolation"
	ELSEIF **NTYPE**=4 THEN
	?"Circular Interpolation"
	…
	ENDIF
	WEND

Instruction	MTYPE

AXISSTATUS--Axis Status

Type	Axis Status
Description	**Check axis status.** Show value as per denary, check bit status as per binary.
Grammar	VAR1 = AXISSTATUS

Bit	Description	Value	
1	Alarm: Follow-Up Error Exceeds.	2	2h
2	Communication with Remote Axis Error	4	4h
3	Remote Driver Error	8	8h
4	Positive Hard Limit	16	10h
5	Negative Hard Limit	32	20h
6	Origin Searching	64	40h
7	Hold Signal IN at HOLD Speed	128	80h
8	Error: Follow-Up Error Exceeds.	256	100h
9	Positive Soft Limit Exceeds	512	200h
10	Negative Soft Limit Exceeds	1024	400h
11	CANCEL in Process	2048	800h
12	Pulse Frequency > MAX_SPEED. Please Low the Speed / Reset MAX_SPEED.	4096	1000h
14	"Robot" Command Coordinates Error	16384	4000h
18	Power Abnormal	262144	40000h
19	Buffer of Precision OUT Exceeds	524288	80000h
20	Speed Protection. Axis Speed > MAX_SPEED, it will Alarm.	1048576	100000h
21	Fail to Trigger Special Commands in Motion.	2097152	200000h
22	Alarm Signal Input	4194304	400000h
23	Axis Paused	8388608	800000h

Controller	General
Example	Example one: Read bit directly (it is recommended when programming) When meeting positive limit. VAR = READ_BIT2(4,**AXISSTATUS(0)**) 'check if axis 0 meets positive limit Print VAR 'print result, it is 1, then already met positive limit Example two: Check returned value **?AXISSTATUS(1)** 'check status of axis 1, it is 48 '48=32+16, axis is met positive and negative limit at the same time, it usually happens when limit electric level is not reversed. Example Three: Field bus communication error After enabling motor correctly: Disconnect the communication wiring, then AXISSTATUS will show 4, which means communication error with remote axes. If disconnect encoder wiring, then AXISSTATUS will show 8, which means

	remote drive error.
Instruction	AXIS_STOPPREASON

IDLE--Motion Status

Type	Axis Status
Description	**Axis motion status, only to judge whether motion is in process or stops.** 0-in motion, -1-motion ends. If motion parts are robotics, then in CONNFRAME mode, joint axis will always return IDLE value 0, in CONNREFRAME mode, virtual axis will return IDLE value 0.
Grammar	VAR1 = IDLE
Controller	General
Example	Example One： IF **IDLE(0)** then 'if axis 0 stops BASE(1) MOVE (100) ENDIF Example Two： BASE(0,1) MOVE(100,100) BASE(2,3) MOVE(200,200) **WAIT UNTIL IDLE(0) AND IDLE(1) AND IDLE (2) AND IDLE(3)** 'wait until axis0,1,2,3 stops
Instruction	LOADED, WAIT IDLE

ADDAX_AXIS--Added Axis NO.

Type	Axis Status
Description	**Axis NO. of added axis by instruction ADDAX, -1 means no axis was added.**
Grammar	VAR1 = ADDAX_AXIS
Controller	General
Example	ADDAX(0) AXIS(1) 'add motion of axis 0 to axis 1. ?**ADDAX_AXIS**(1) 'print added axes on axis 1, result is 0. ADDAX(-1) AXIS(1) 'cancel axis add.
Instruction	ADDAX

AXIS_STOPREASON--Axes Stop Reason

Type	Axis Status
Description	**Latch history reasons of axes stop.**
Grammar	Write as 0, which means clear. Latch as per bit, same meaning as AXISSTATUS. Valid in firmware above 20150731
Controller	General

Example	If **AXIS_STOPREASON** AND (512+1024) THEN
	PRINT "axis el stoped"
	ENDIF

Instruction	AXISSTATUS

LINK_AXIS--Link Axis NO.

Type	Axis Status
Description	**Return reference axis NO. of present link motion. Return -1 if there is no link.**
Grammar	VAR1 = LINKAX
Controller	General
Example	CONNECT(2,1) AXIS(0) 'link axis 0 to axis 1. ?**LINK_AXIS**(0) 'print result:1
Instruction	CAMBOX, MOVELINK, CONNECT

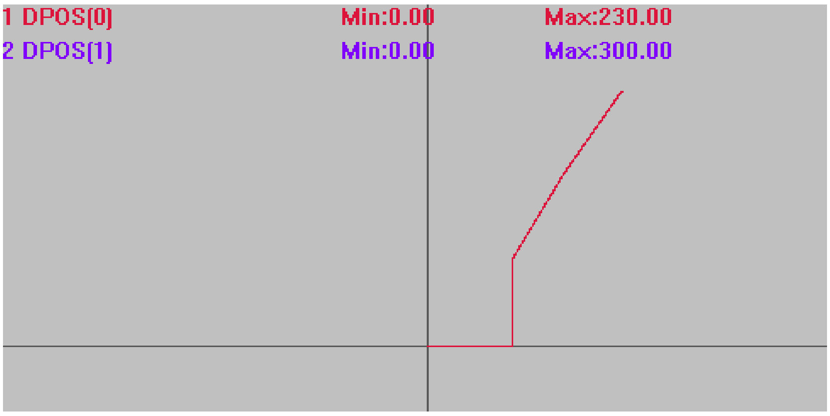
11.5 Motion Look-ahead Instruction

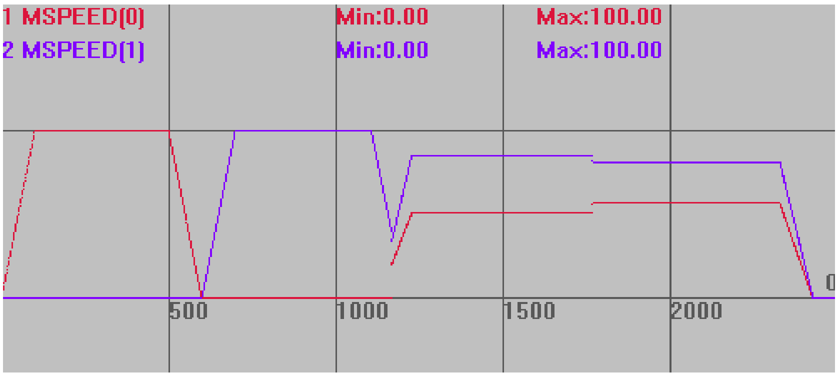
CORNER_MODE--Corner Speed Setting

<table>
<tr><td>Type</td><td>Axis Parameters</td></tr>
<tr><td>Description</td><td>Corner deceleration mode configuration.</td></tr>
<tr><td>Grammar</td><td>CORNER_MODE=mode
mode: different bits indicate different meanings, and bit can be used at the same time.
<table>
<tr><th>Bit</th><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>1</td><td>Reserved</td></tr>
<tr><td>1</td><td>2</td><td>Automatic corner deceleration.
Acceleration, deceleration follow the value of ACCEL, DECEL.
This parameter is activated before MOVE is called.
See DECEL_ANGLE and STOP_ANGLE for the definition of deceleration angle.</td></tr>
</table></td></tr>
</table>

		Reference speed of deceleration angle follows FORCE_SPEED, do set reasonable FORCE_SPEED.
2	4	Reserved
3	8	**Auto speed limit of small circle, when radius is smaller than set value, there is speed limit, if radius is bigger than set value, there is no speed limit.** This parameter is activated before MOVE is called. Speed limit will follow FORCE_SPEED. Limit speed = FORCE_SPEED* actual radius / FULL_SP_RADIUS The radius of limit speed is set by FULL_SP_RADIUS.
4	16	Reserved
5	32	**Auto chamfer setting.** This parameter is activated before MOVE is called. Present MOVE motion will chamfer with former MOVE motion automatically, chamfer radius refers to ZSMOOTH. This chamfer is valid in all axes which are doing interpolation motion, firmware should be above 20150701.
6	64	**Multi-axis interpolation separation speed, automatically corner decelerate.** The same as mode 2, the difference is interpolation motion of mode 2 uses speed parameter of main axis, but interpolation of mode 64 uses speed parameters of each axis. It is valid in the latest firmware above 4 series.
7	128	When MOVE runs robotic arm virtual axis, using joint-axis speed and acceleration to limit combined speed and acceleration at the same time. It takes effect when it is used together with BIT6, the controller firmware “version_build” of ZMC4XX and above should be after 240521. It only supports MOVE line command, doesn’t support circular.
8	256	MOVER command uses SP mode.
9	512	Reserved
10	1024	Max speed limit, if the axis speed exceeds MAX_SPEED, please reduce the speed, it only supports line and screw axis.

Controller	General
Example	Example below only shows function of each bit, functions of multi-bit are also available. For Example, CONNER_MODE=2+8, it means bit 1 and bit 3 are opened, then functions of auto corner deceleration and small circle speed limit are

	opened. **Example One: Corner Speed Limit** BASE(0,1) DPOS=0,0 UNITS=100,100 ACCEL=500,500 'set acceleration DECEL=500,500 'set deceleration SPEED=100,100 'set speed MERGE=ON 'open continuous interpolation **CORNER_MODE**=2 'start corner deceleration DECEL_ANGLE = 15 * (PI/180) 'set angle where starts to decelerate STOP_ANGLE = 45 * (PI/180) 'Set angle where deceleration ends. FORCE_SPEED=100 'geometric deceleration activates TRIGGER 'trigger oscilloscope automatically MOVE(100,0) MOVE(0,100) 'motion angle is over 45°,total deceleration. MOVE(60,100) 'motion angle is 30.96°,between 15° and 45°, geometric deceleration MOVE(70,100) 'motion angle is 4.03°,below 15°, no deceleration. Trace Curve DPOS(0) vertical scale 200 DPOS(1) vertical scale 200  Speed Curve: MSPEED(0)=100(vertical scale) MSPEED(1)=100(vertical scale)



Some precision errors may happen in simulator, it is better to check with actual controllers connected.

Example Two: Speed Limit of Small Circle

BASE(0,1)

DPOS=0,0

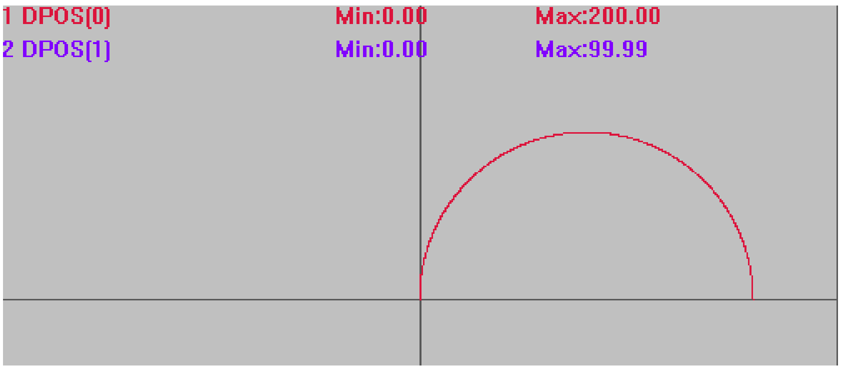
UNITS=100,100

ACCEL=500,500	'set acceleration
DECEL=500,500	'set deceleration
SPEED=100,500	'running speed
CORNER_MODE=8	'start speed limit of small circle
FORCE_SPEED=120	'limit speed of small circle
FULL_SP_RADIUS=60	'speed limit radius is 60
TRIGGER	'trigger oscilloscope automatically
MOVECIRC(200,0,100,0,1)	'when motion radius is over limit, there is no limit speed, and it will follow SPEED, at the speed of 100.

Trace Curve:

DPOS(0) vertical scale 100

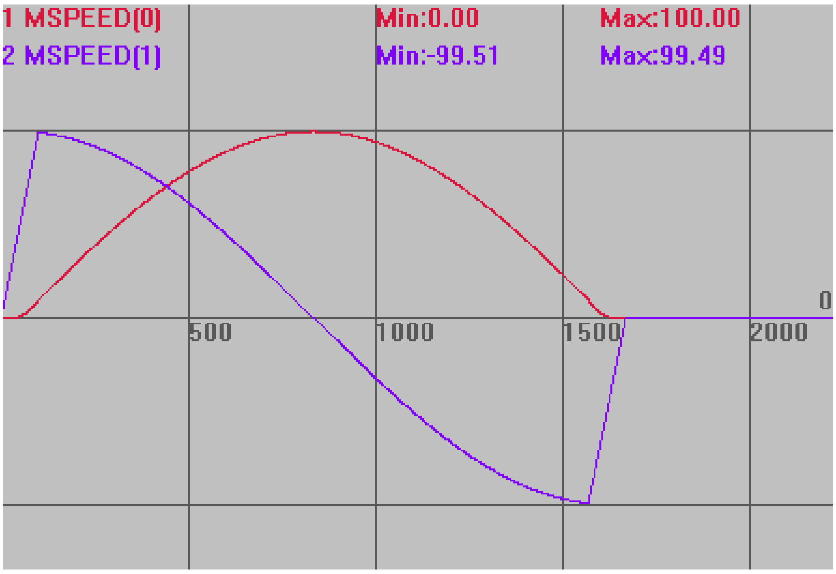
DPOS(1) vertical scale 100



Speed Curve:

MSPEED(0)=100(vertical scale)

MSPEED(1)=100(vertical scale)



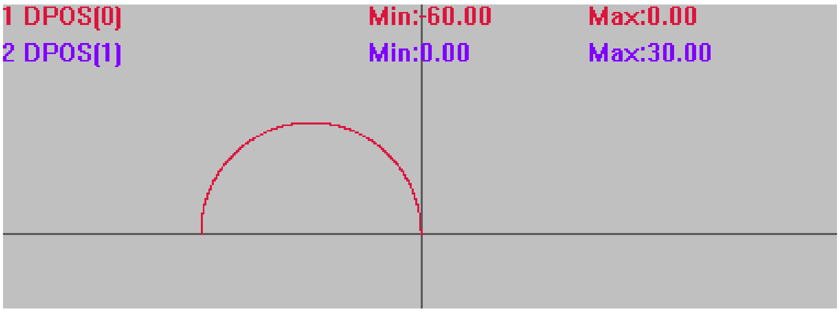
'When radius is smaller than limit value, limit speed = FORCE_SPEED* actual radius/FULL_SP_RADIUS

MOVECIRC(-60,0,-30,0,0)   'now speed 60=120*30/60

Trace Curve:

DPOS(0) vertical scale 100

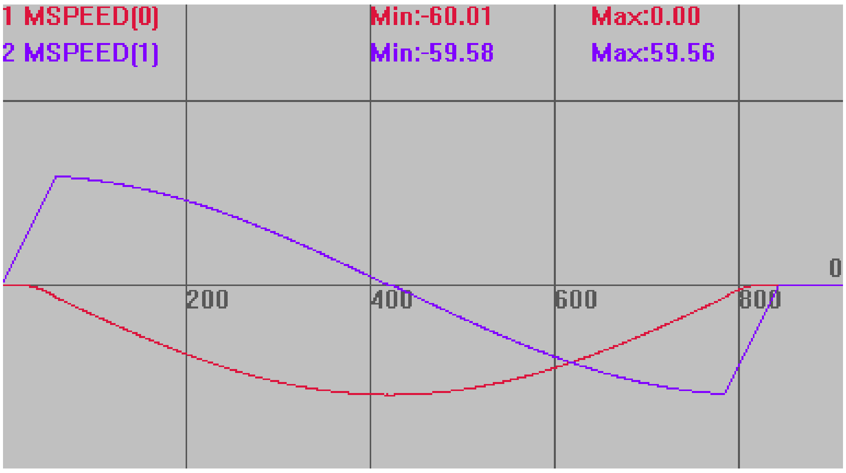
DPOS(1) vertical scale 100



Speed Curve:

MSPEED(0)=100(vertical scale)

MSPEED(1)=100(vertical scale)



Example Three: Chamfer

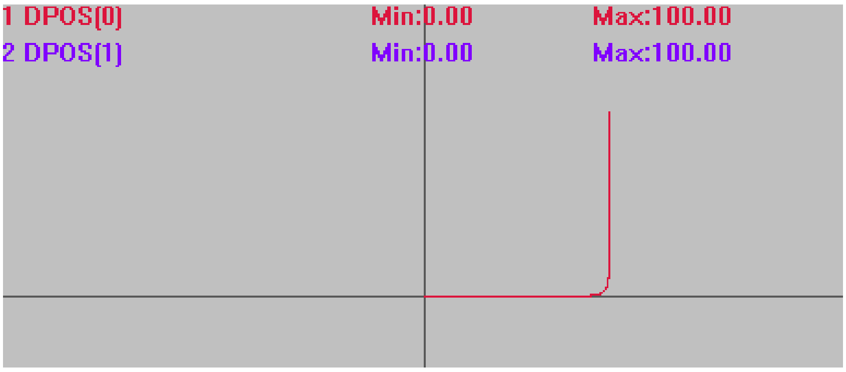
Chamfer mode can be used in linear, circular, helical motion etc. here only shows chamfer linear motion.

```
BASE(0,1)
DPOS=0,0
ACCEL=500,500          'set acceleration
DECEL=500,500          'set deceleration
SPEED=100,100          'running speed
CORNER_MODE=32         'start chamfer
ZSMOOTH=10             'chamfer reference radius.
TRIGGER                'trigger oscilloscope automatically
MOVE(100,0)
MOVE(0,100)            'chamfer between former two linear motions.
```

Interpolation trace with chamfer.

DPOS(0) vertical scale 100

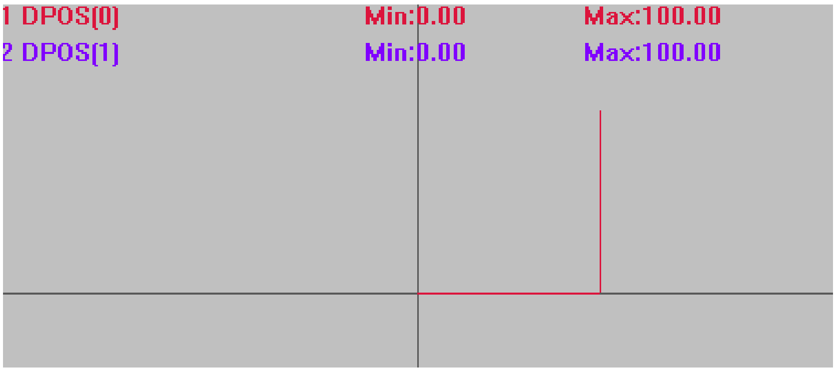
DPOS(1) vertical scale 100



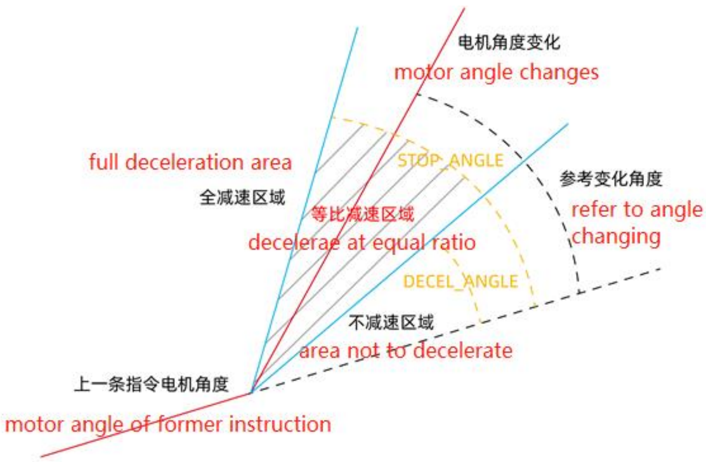
Interpolation trace without chamfer:

DPOS(0) vertical scale 100

DPOS(1) vertical scale 100

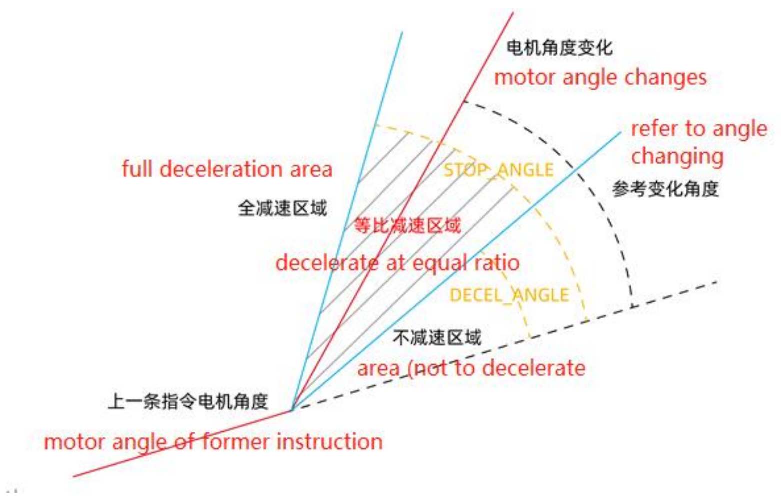
	
Instruction	MERGE, STOP_ANGLE, DECEL_ANGLE, FULL_SP_RADIUS ZSMOOTH.

DECEL_ANGLE--Corner Deceleration Angle

Type	Axis Parameters
Description	**Angle where deceleration starts, unit is rad.** Corner deceleration speed refers to FORCE_SPEED, FOR_SPEED should be set properly. Convert angle to radian: angle*(PI/180). Deceleration Angle means the changing value between reference angle of the motor and its former motion. Please see the below figure. This angle value is not the actual path angle, which converts to motion changing angle and is only for reference.  If the next interpolation motion is under below, then get its absolute value instead. When line links with circle arc, calculate angle according to the tangent direction of arc. DECEL_ANGLE is usually used with STOP_ANGLE together, when angle of actual motion is between DECEL_ANGLE （upper limit） and STOP_ANGLE（lower limit）, then deceleration will happen.

Grammar	VAR1 = DECEL_ANGLE，DECEL_ANGLE = expression
Controller	General
Example	Refer to CORNER_MODE routine 1. CORNER_MODE=2 **DECEL_ANGLE** = 25 * (PI/180) 'set start angle of deceleration. STOP_ANGLE = 45 * (PI/180) 'set end angle of deceleration. FORCE_SPEED=SPEED 'FORCE_SPEED must be set.
Instruction	STOP_ANGLE

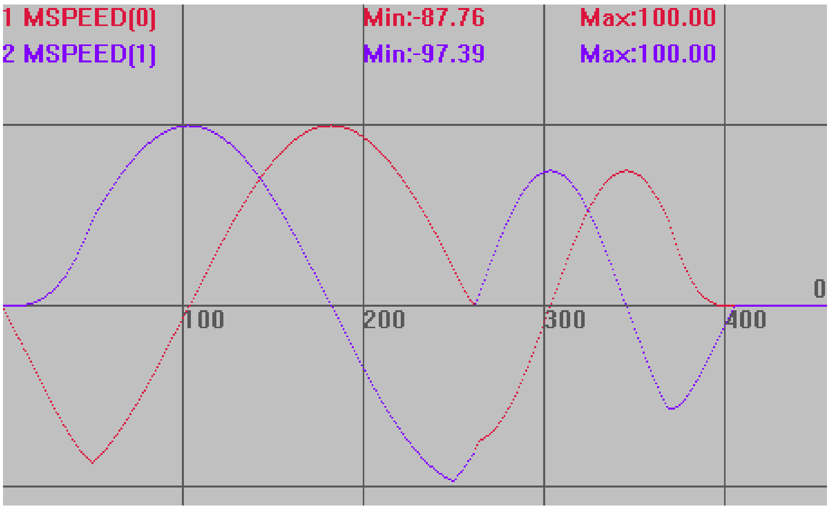
STOP_ANGLE--Corner Deceleration Stops

Type	Axis Parameters
Description	**Angle where deceleration stops, unit is rad.** Corner deceleration speed refers to FORCE_SPEED, FOR_SPEED should be set properly. Convert angle to radian: angle*(PI/180). Deceleration Angle means the changing value between reference angle of the motor and its former motion. Please see the below figure. This angle value is not the actual path angle, which converts to motion changing angle and is only for reference.  If the next interpolation motion is under below, then get its absolute value instead. When line links with circle arc, calculate angle according to the tangent direction of arc. DECEL_ANGLE is usually used with STOP_ANGLE together, when angle of actual motion is between DECEL_ANGLE（upper limit） and STOP_ANGLE（lower limit），then deceleration will happen.
Grammar	VAR1 = STOP_ANGLE，STOP_ANGLE= expression
Controller	General
Example	See example one in CORNER_MODE

	CORNER_MODE=2 DECEL_ANGLE = 25 * (PI/180) **STOP_ANGLE** = 90 * (PI/180) FORCE_SPEED=SPEED 'FORCE_SPEED must be set
Instruction	DECEL_ANGLE, CORNER_MODE

FULL_SP_RADIUS--Speed Limit Radius

Type	Axis Parameters
Description	**Maximum arc radius of speed limit, unit is units.** When radius is over FULL_SP_RADIUS, motion speed will follow the value assigned by user procedure, or if below FULL_SP_RADIUS, motion speed will decrease in proportion. **VP_SPEED=FORCE_SPEED * radius/FULL_SP_RADIUS** It refers to radius of chamfer in auto chamfer mode.
Grammar	VAR1 = FULL_SP_RADIUS, FULL_SP_RADIUS = expression
Controller	General
Example	BASE(0,1) 'select axis 0 as main axis DPOS=0,0 'coordinate clears UNITS=100,100 ATYPE=1,1 'set axis type SPEED=100,100 'main axis speed is 100units/s ACCEL=1000,1000 'acceleration speed is 1000units/s DECEL=1000,1000 'deceleration speed is 1000units/s FORCE_SPEED=150 'self-defined speed is 150units/s CORNER_MODE=8 'open small circle speed limit **FULL_SP_RADIUS**=8 'limit radius is 8 TRIGGER 'trigger oscilloscope automatically MOVECIRC(10,10,0,10,1)'arc radius is 10, no speed limit, and move at speed = 100 MOVECIRC(4,4,0,4,1) 'arc radius is 4, speed limit is activated, and move at the speed 4/8*150=75. Speed Curve: MSPEED(0)=100(vertical scale) MSPEED(1)=100(vertical scale)

	
Instruction	CORNER_MODE, FORCE_SPEED, SPLIMIT_RADIUS

SPLIMIT_RADIUS--Speed Limit Value

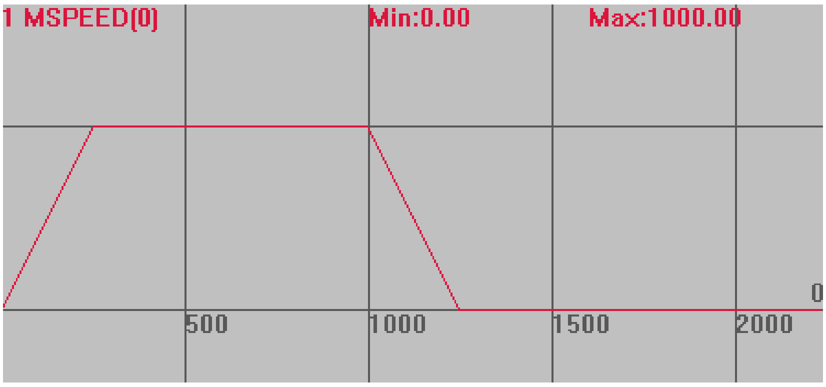
Type	Axis Parameters
Description	**Minimum speed in small circle limit mode, unit is units.** **When value is below LSPEED, follow LSPEED.**
Grammar	VAR1 = SPLIMIT_RADIUS, SPLIMIT_RADIUS = expression
Controller	ZMC4XX series controller with firmware above 170518 supports.
Instruction	CORNER_MODE, FULL_SP_RADIUS

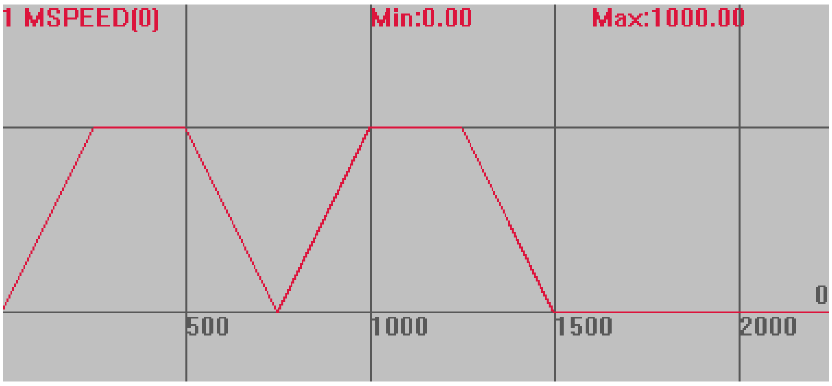
ZSMOOTH--Chamfer Radius

Type	Axis Parameters
Description	**See chamfer radius and CONER_MODE for reference.** Calculate actual corner radius based on corner angle. if exceeds angle, it is 50% of set value(ZSMOOTH). When it is 90°, corner radius is set value (ZSMOOTH).
Grammar	VAR1 = ZSMOOTH, ZSMOOTH=smoothdistance
Controller	General
Example	See Example Three in CONER_MODE
Instruction	CORNER_MODE

MERGE--Continuous Interpolation

Type	Axis Parameters
Description	**Motions in buffer will be integrated without deceleration, which is used**

	to realize continuous interpolation. When MERGE is ON, multi-interpolation motions still decelerate in between, some possible reasons as follow: 1.MERGE is not set successfully, print result to check. 2.Controller is point-to-point model, which means it doesn't support continuous motion. Please contact with manufacturers. 3.CORNER_MODE was set to define corner deceleration, print result to check. 4.SP motion instructions are in process, and ENDMOVE_SPEED and STARTMOVE_SPEED are set, then speed will follow value of these two instructions. 5.Main Axis was switched between interpolation motions, and main axis parameters were also changed. 6.MOVE_DELAY was added between interpolation motions, even MOVE_DELAY was set as 0, it also will cause deceleration.
Grammar	MERGE = ON/OFF
Controller	General
Example	BASE(0) 'select axis 0 DPOS=0 UNITS=100 SPEED=1000 'set speed as 1000units ACCEL=1000 DECEL=1000 **MERGE**=ON 'open continuous interpolation TRIGGER 'trigger oscilloscope automatically MOVE(100) 'move 100units MOVE(100) Speed Curve: MSPEED(0) vertical scale 100  When MERGE=OFF MSPEED(0) vertical scale 100

	
Instruction	*SP, CORNER_MODE

11.6 Motion Buffer Instruction

LOADED--Buffer Empty

Type	Axis Status
Description	**If there are no other motion instructions in buffer except present motion, return TURN.** This instruction can't judge whether axis stops or not. Please use IDLE command to judge.
Grammar	VAR1 = LOAED
Controller	General
Instruction	IDLE, WAIT LOADED

MOVES_BUFFERED--Present Buffer Number

Type	Axis Status
Description	**Return motion instructions number in buffer** Do not use total buffer space minus MOVES_BUFFERED to judge how many buffers are left, since special instructions may consume multi buffers. It is more accurate to use REMAIN_BUFFER.
Grammar	VAR1 = MOVES_BUFFERED
Controller	General
Example	Print **MOVES_BUFFERED** 'result: 0
Instruction	LOADED, LIMIT_BUFFERED, REMAIN_BUFFER

REMAIN_BUFFER--Rest Buffers

Type	Special Axis Status

Description	**Return rest motion buffers number.** Since this status instruction has its own parameters, AXIS can't be omitted when try to modify axis NO.. If returned value is 0, it means buffer space is full, when try to call additional motion instructions of axis, then task will be blocked until there is space in buffer.
Grammar	VAR1 = REMAIN_BUFFER ([mtype]) AXIS(AXISNUM) Default value of Mtype is type of motion which is most complex, such as spherical interpolation.
Controller	General
Example	DIM movetime 'define variable movetime = 0 WHILE movetime < 100 'condition cycle IF **REMAIN_BUFFER**(1) > 0 THEN 'if there is buffer left, call linear motion instruction MOVE(10) movetime = movetime +1 ENDIF WEND 'MOVE(10) was 100 times called
Instruction	LOADED, LIMIT_BUFFERED

MOVE_MARK--Move Mark

Type	Axis Parameters
Description	**MARK number of next motion instruction, and it will enter buffer together with motion instructions.** Every time an instruction was called, MOVE_MARK will increase with 1 automatically. If needs to set required MOVE_MARK value, then set MOVE_MARK before motion instruction. To pause motion between different MARK numbers through MOVE_PAUSE
Grammar	VAR1 = MOVE_MARK, MOVE_MARK = expression
Controller	General
Example	**MOVE_MARK** =1 'set mark number as 1 MOVE(100) **MOVE_MARK** =1 'set mark number as 1 MOVE(100) **MOVE_MARK** =2 'set mark number as 2 MOVE(200) MOVE_PAUSE (2) 'pause before MOVE(200)
Instruction	MOVE_CURMARK

MOVE_CURMARK--Return Move Mark

Type	Axis Status
Description	**Return MOVE_MARK value of present motion instruction in process.**
Grammar	VAR1 = MOVE_CURMARK
Controller	General
Example	MOVE_MARK =1 MOVE(100) MOVE_MARK =2 MOVE(200) MOVE_MARK =3 MOVE(300) WAIT UNTIL **MOVE_CURMARK**=2 'wait until MOVE(200) starts to run, open output 1 OP(1,ON)
Instruction	MOVE_MARK

LIMIT_BUFFERED--Motion Buffer Limit

Type	System Parameters
Description	**Limit motion buffer numbers, it can not exceed maximum buffer value of controller.**
Grammar	LIMIT_BUFFERED=value, VAR1 = LIMIT_BUFFER
Controller	General
Example	Online print commands: >>?LIMIT_BUFFERED 4096 Modify motion buffer number limit: >>LIMIT_BUFFERED=2000 >>?LIMIT_BUFFERED 2000
Instructions	REMAIN_BUFFER,MOVES_BUFFERED

11.7 Instructions Related to Position

DPOS--Axis Instruction Position

Type	Axis Status
Description	**Virtual coordinate position of axis, or required position.** Value written into DPOS will not cause motor motion, it will be converted to OFFPOS offset automatically.

	UNITS as unit.
Grammar	VAR1 = DPOS, DPOS=expression
Controller	General
Example	**DPOS**(0) = 0 'coordinate of axis 0 offsets to 0. ?***DPOS** 'print DPOS, result:0 0 0 0 0 0 0 0 0 0 0 0
Instruction	MPOS, ENDMOVE, OFFPOS, DEFPOS

MPOS--Encoder Feedback Position

Type	Axis Status
Description	**Measured position feedback of axis, unit is units.** Written MPOS value will be converted to OFFPOS amount
Grammar	VAR1 = MPOS, MPOS=expression
Controller	General
Example	**MPOS**(0) = 50 'MPOS offset is 50 ?***MPOS** 'print result:50 0 0 0 0 0 0 0 0 0 0 0
Instruction	DPOS,ENDMOVE

DEFPOS--Position Offset

Type	Coordinate instruction.
Description	**Set the present position as another new absolute position value, which has no influence on the motion in process or motion in buffer.**
Grammar	DEFPOS(pos1 [,pos2[, pos3[, pos4.....]]]) pos1: Absolute position, using units as unit pos2: Absolute position of next axis, using units
Controller	General
Example	**Example 1:** BASE(0,1) 'choose axis 0 and 1 ATYPE=1,1 UNITS=100,100 'set units as 100 DPOS=0,0 'clear DPOS MOVE(100,100) 'axis 0 and axis 1 move 100 WIAT IDLE ?DPOS(0),DPOS(1) 'print present DPOS, both are 100 **DEFPOS**(0,10) 'set present DPOS ?DPOS(0),DPOS(1) 'print present DPOS, DPOS are 0,10 **Example 2:** Different from OFFPOS, DEFPOS is used to change the absolute position. BASE(0,1) 'choose axis 0,1 DPOS=100,100 'set position as 100,100 ?DPOS(0), DPOS(1) 'print position, the present position is 100,100

	DEFPOS(10,20) 'set present position as 10,20 ?DPOS(0), DPOS(1) 'print position, they are 10,20 **DEFPOS**(10,20) 'set the present position again **DEFPOS**(10,20) ?DPOS(0), DPOS(1) 'print position, they are still 10,20 OFFPOS=10,20 'call OFFPOS several times OFFPOS=10,20 ?DPOS(0),DPOS(1)'now present position is 30,60 （10+10+10,20+20+20）
Instructions	DPOS, OFFPOS

OFFPOS--Offset Position

Type	Axis Parameters
Description	**Relative offset to change all coordinates, which has no influence on motion in process or motion in buffer.** After modification was finished, OFFPOS recovers to 0.
Grammar	VAR1 = OFFPOS, OFFPOS = expression
Controller	General
Example	**Example One: relative offset position** BASE(0) MOVEABS(1000) WAIT IDLE **OFFPOS**=-1000 'coordinate offset is 1000 PRINT DPOS(0) 'print result is 0 **Example Two: no change of motion in process** BASE(0) MOVEABS(1000) 'move to absolute position 1000 **OFFPOS**=500 'position offset is 500 WAIT IDLE PRINT DPOS(0) 'print present coordinate position: 1500, motor still runs 1000. **Example Three** DEFPOS is to change absolute coordinate position, while OFFPOS is to change relative coordinate position. BASE(0,1) 'select axis 0, axis 1 DPOS=100,100 'set present position as:100,100 ?DPOS(0), DPOS(1) 'print to check DEFPOS(10,20) 'set present coordinate position as 10,20 ?DPOS(0), DPOS(1) 'print result:10,20 DEFPOS(10,20) 'call DEFPOS several times DEFPOS(10,20) ?DPOS(0), DPOS(1) 'print result:10,20

	OFFPOS=10,20 'call DEFPOS several times **OFFPOS**=10,20 ?DPOS(0), DPOS(1) 'print result: 30,60(10+10+10,20+20+20)
Instruction	DPOS , DEFPOS

ENDMOVE--Target Position

Type	Axis Status
Description	**Absolute target position of present axis motion.** For instructions: VMOVE, DATUM, etc. ENDMOVE is not accurate, it changes as per the motion status.
Grammar	VAR1 = ENDMOVE
Controller	General
Example	BASE(0) SPEED = 10 'speed is 10units/s DPOS = 0 'coordinate clears MOVE(100) 'move 100units WAIT IDLE PRINT **ENDMOVE**(0) 'result:100 MOVE(200) WAIT IDLE PRINT **ENDMOVE**(0) ''result:300
Instruction	DPOS, MPOS, ENDMOVE_BUFFER

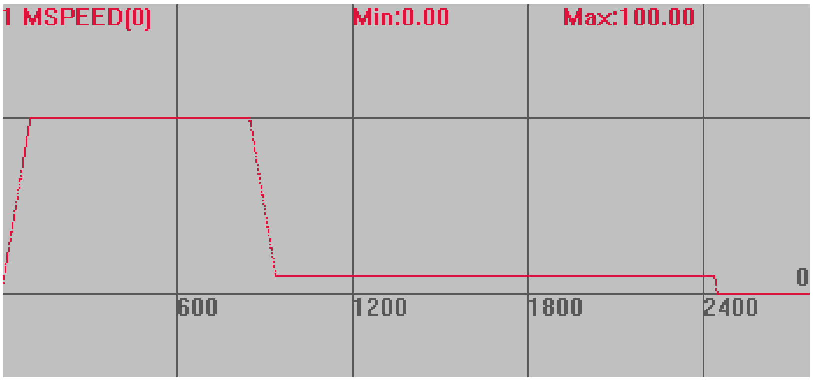
VECTOR_MOVED--Present Motion Distance

Type	Axis Status
Description	**Return motion distance of present axis, unit is units.** This distance is vector distance in terms of muti axes interpolation motion. It is better to clear value of this parameter before use. This command is only valid for motion instructions, invalid in superposition instructions, such as, ADDAX, CONNECT, etc.
Grammar	VAR1=VECTOR_MOVED, VECTOR_MOVED=0
Controller	General_
Example	**Example 1: single axis** **VECTOR_MOVED**=0 'clear parameter MOVE(100) WAIT IDLE ? **VECTOR_MOVED** 'print motion distance of axis 0, result is 100. **Example 2: multi-axis** BASE(0,1) DPOS = 0,0

	VECTOR_MOVE = 0 'manually clear resultant vector motion distance of axis 0 as 0 BASE(2,3) DPOS = 0,0 VECTOR_MOVE = 0 'manually clear resultant vector motion distance of axis 2 as 0 BASE(0,1) MOVE(-300,-400) WAIT IDLE(0) ?VECTOR_MOVED MOVE(300,400) WAIT IDLE(0) ?VECTOR_MOVED 'print resultant vector motion distance of axis 0, result: 500 BASE(2,3) MOVE(30,-40) WAIT IDLE(2) ?VECTOR_MOVED MOVE(30,40) WAIT IDLE(2) ?VECTOR_MOVED
Instruction	ENDMOVE

REMAIN--Rest Target Motion Distance

Type	Axis Status
Description	**Return remain distance need to move, unit is units.**
Grammar	VECTOR_BUFFERED
Controller	General
Example	BASE(0) 'select axis 0 DPOS=0 UNITS=100 SPEED=100 'speed is 100units/s ACCEL=1000 'acceleration is 1000units/s DECEL=1000 TRIGGER 'trigger oscilloscope automatically MOVE(100) 'move 100units WAIT UNTIL **REMAIN**<20 'wait until remain distance is less than 20 SPEED=10 'change speed Speed Curve: MSPEED(0) vertical scale 100

	
Instruction	VECTOR_BUFFERED

VECTOR_BUFFERED--Remain Distance in Buffer

Type	Axis Status
Description	**Return distance of motion in process and motion in buffer, unit is units.** This distance is vector distance in terms of muti axes interpolation motion.
Grammar	VAR1 = VECTOR_BUFFERED
Controller	General
Example	BASE(0) 'select axis 0 UNITS=100 'pulse amount is 100 SPEED=100 'speed is 100units/s ACCEL=1000 'acceleration is 1000units/s/s MOVE(100) 'motion in process is 100units MOVE(300) 'motion in buffer is 300units MOVE(-1000) 'motion in buffer is -1000units ?**VECTOR_BUFFERED** 'return remain motion distance, result is 1400
Instruction	REMAIN

VECTOR_BUFFERED2—Target Vector Distance

Type	Axis Status
Description	**It is used to read the target vector position after current interpolation command is called, unit is units.** VECTOR_BUFFERED2=VECTOR_BUFFERED+VECTOR_MOVED It can be read for HW comparison output.
Grammar	VAR1 = VECTOR_BUFFERED2
Controller	General
Example	BASE(0,1) 'select axis 0 and axis 1 UNITS=100,100 SPEED=100,100 ACCEL=1000,1000

	DECEL=1000,1000 MERGE=0,0 VECTOR_BUFFERED2=0,0 DPOS=0,0 MOVE(100,100) '3 interpolation motions MOVE(200,200) MOVE(-100,-100) ?“start to move” ?VECTOR_BUFFERED 'return remain motion distance, result is 565.68 ?VECTOR_MOVED 'current motion distance, result is 0 ?VECTOR_BUFFERED2 'return required target vector distance, result is 565.68 DELAY(1000) 'delay is 1s ?“in motion” ?VECTOR_BUFFERED 'remain motion distance, result is 470.63 ?VECTOR_MOVED 'the current motion distance, result is 95.05 ?VECTOR_BUFFERED2 'return required target vector distance, result is 565.68 WAIT IDLE 'wait until axis stops ?“motion ends” ?VECTOR_BUFFERED 'remain motion distance, result is 0 ?VECTOR_MOVED 'current motion distance, result is 565.68 ?VECTOR_BUFFERED2 'required target vector position of interpolation command, result is 565.68
Instruction	VECTOR_BUFFERED

ENDMOVE_BUFFER--Final Position in Buffer

Type	Axis Status
Description	**Final target position based on present motion and motion in buffer.** It can be used to realize conversion between absolute position and relative position, see Example Two. Instructions have no fixed distance, such as, VMOVE, DATUM, etc. ENDMOVE is not accurate, it changes as per the motion status. After REP_OPTION cycle coordinate instruction is used, ENDMOVE_BUFFER will decrease as per set value: REP_DIST in REP_OPTION mode, which means minimum precision is REP_DIST (mode1) or 2*REP_DIST (mode0), see Example Three.
Grammar	VAR1 = ENDMOVE_BUFFER

Controller	General
Example	**Example One** BASE(0) SPEED = 10 DPOS = 0 MOVE(100) MOVE(200) PRINT **ENDMOVE_BUFFER**(0) 'print final absolute coordinate, result:300. **Example Two: conversion between absolute and relative** Use ENDMOVW_BUFFER and relative motion instructions together to realize absolute motion, such as MOVE, MSPHERICAL, etc. BASE(0) UNITS=100 SPEED=100 ACCEL=1000 DPOS=0 WHILE 1 MOVE(100- **ENDMOVE_BUFFER**(0))'move to position 100, then stop. WEND **Example Three: returned value of cycle coordinate.** BASE(0) UNITS=100 SPEED=100 ACCEL=1000 DPOS=0 TRIGGER MOVE(1000) REP_DIST=100 'set coordinate cycle range. REP_OPTION=1 'cycle range: 0~100 WHILE 1 ?**ENDMOVE_BUFFER**(0) 'print result:1000,900,800...,100,0 'minimum precision:100 WEND
Instruction	DPOS, MPOS, ENDMOVE

11.8 Instructions for Origin Homing

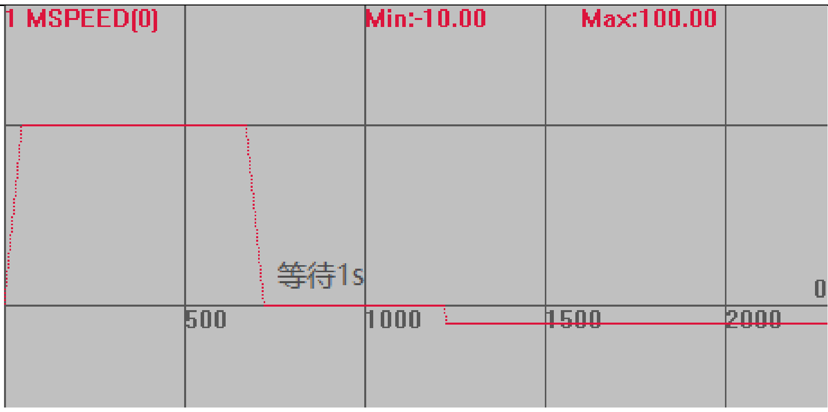
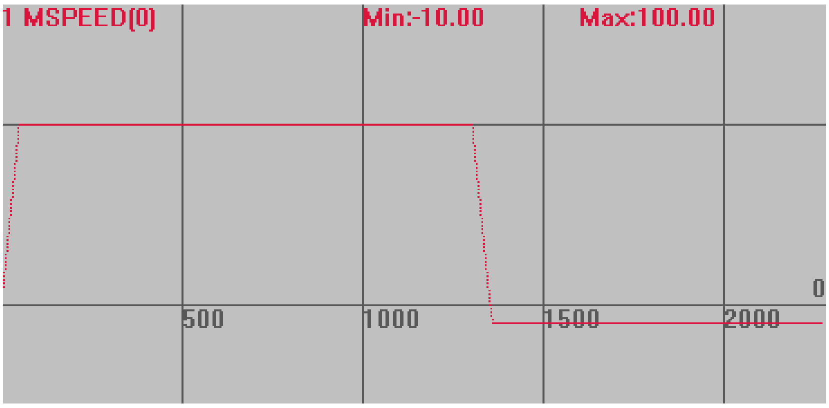
DATUM_IN--Origin Input

Type	Axis Parameters

Description	**Configure general input as origin signal input, -1 means invalid.** Input is valid when signal is OFF in ZMC controller, do use INVERT_IN to reverse the electric level. (except for ECI)
Grammar	VAR1 = DATUM_IN, DATUM_IN = expression
Controller	General
Example	BASE(0,1,2,3) **DATUM_IN** = 6,7,8,9 'origin inputs of axis 0,1,2,3 relate to input 6,7,8,9. INVERT_IN(6,ON) 'reverse origin signal. INVERT_IN(7,ON) INVERT_IN(8,ON) INVERT_IN(9,ON)
Instruction	DATUM, FWD_IN, REV_IN, INVERT_IN

HOMEWAIT—Reversely Find Delay when Homing

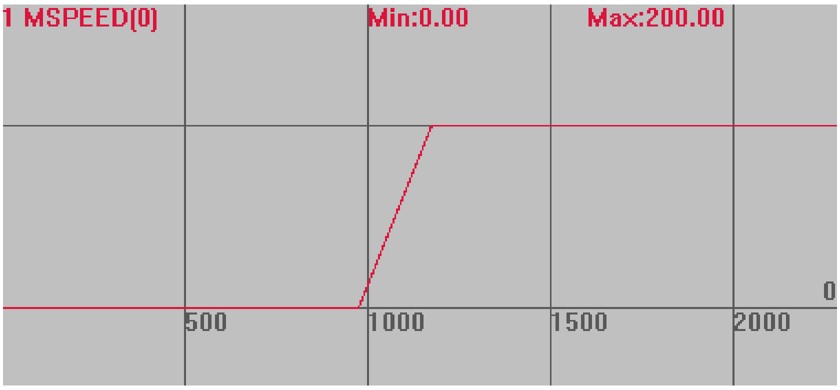
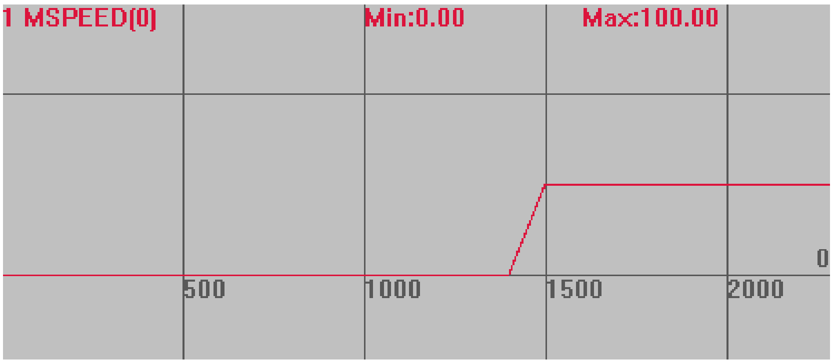
Type	Axis Parameters
Description	**This parameter sets waiting time, units is millisecond.** In terms of pulse servo drives, there needs time delay when back finding origin point. Default value of controller is 2ms.
Grammar	VAR1 = HOMEWAIT, HOMEWAIT = expression
Controller	General
Example	BASE(0) DPOS=0 'axis 0 clears UNITS=100 'coordinate clears ATYPE=1 SPEED=100 'speed of finding origin point is 100units/s. ACCEL=1000,1000 'acceleration is 1000units/s DECEL=1000,1000 'deceleration is 1000units/s CREEP=10 'speed of backing finding origin point DATUM_IN=0 'IN0 as homing signal input INVERT_IN(0,ON) 'reverse signal **HOMEWAIT**=1000 'set time delay of back finding TRIGGER 'trigger oscilloscope automatically DATUM(3) 'positive finding origin point Speed Curve: When meeting IN0, it will stop for 1 second, then back to find origin point at speed of CREEP MSPEED(0) vertical scale 100

	 When HOMEWAIT=2, it almost doesn’t stop. MSPEED(0) vertical scale 100 
Instruction	DATUM

11.9 JOG Motion Instruction

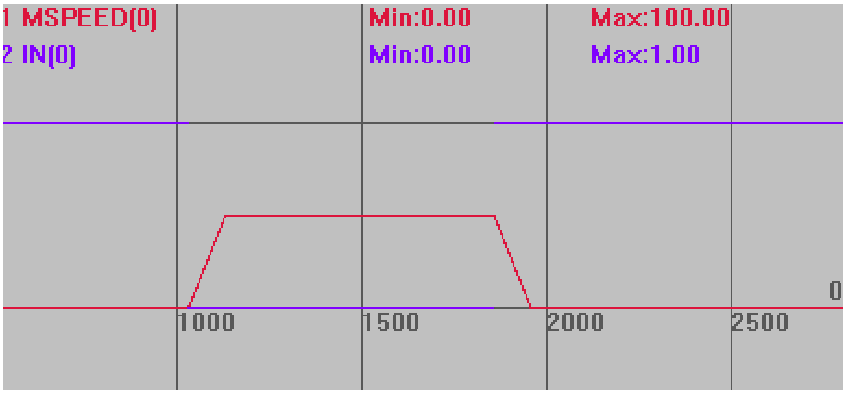
FAST_JOG--Jog Input Mapping

Type	Axis Parameters
Description	**Fast Jog input NO., -1 means invalid.** If fast jog input was set, then speed of motion will follow SPEED, or it will follow JOGSPEED. See Example One. Input is valid when signal is off in ZMC controller, do use INVERT_IN to reverse the electric level. (except for ECI)
Grammar	VAR1 = FAST_JOG, FAST_JOG = expression
Controller	General
Example	BASE(0) 'select axis 0 DOPS=0 'axis o clears UNITS=100 ATYPE=1 SPEED=100 'set speed as 100 units/s ACCEL=500 'set acceleration is 500 units/s

	JOGSPEED=200 'jog move speed is 200units/s **FAST_JOG**(0)=0 'fast jog input of axis 0 is IN0. FWD_JOG(0)=1 'positive jog move input IN1 INVERT_IN(0,ON) 'reverse electric level INVERT_IN(1,ON) TRIGGER 'trigger oscilloscope automatically Speed Curve: When IN0 doesn't input, button IN1 and keep this status, axis speed is JOGSPEED=200 MSPEED(0)=200(vertical scale)  IN0:ON, IN1:ON, axis speed is SPEED=100 
Instruction	REV_JOG, FWD_JOG, SPEED, JOGSPEED

FWD_JOG--Positive JOG Input Mapping

Type	Axis Parameters
Description	**Input number relates to positive JOG input, -1 means invalid.** Input is valid when signal is off in ZMC controller, do use INVERT_IN to reverse the electric level. (except for ECI) When there is input signal, axis will move at speed of JOGSPEED in positive direction.

Grammar	VAR1 = FWD_JOG, FWD_JOG= expression
Controller	General
Example	Example One BASE(0) 'select axis 0 DPOS=0 'axis 0 clears UNITS=100 ATYPE=1 SPEED=100 'set speed as 100 ACCEL=500 'set acceleration is 500 units/s/s DECEL=500 JOGSPEED=50 'JOG speed is 50 **FWD_JOG**=0 'IN0 as positive JOG switch INVERT_IN(0,ON) 'reverse signal TRIGGER 'trigger oscilloscope automatically When IN0 is ON, axis move at speed of 50 in positive direction. When IN0 is OFF, axis motion stops. Speed Curve: MSPEED(0) vertical scale 100 
Instruction	REV_JOG, JOGSPEED, FAST_JOG

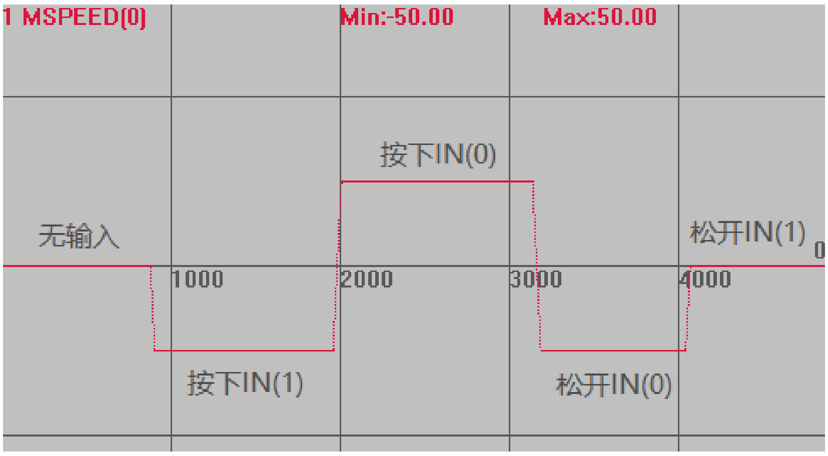
REV_JOG--Negative JOG Input Mapping

Type	Axis Parameters
Description	**Input number relates to negative JOG input, -1 means invalid.** Input is valid when signal is off in ZMC controller, do use INVERT_IN to reverse the electric level. (except for ECI) When there is input signal, axis will move at speed of JOGSPEED in negative direction. When both signals of REV_JOG and FWD_JOG come, axis moves as per FWD_JOG

Grammar	VAR1 = REV_JOG, REV_JOG= expression
Controller	General
Example	See in Example FWD_JOG.
Instruction	FWD_JOG, JOGSPEED, FAST_JOG

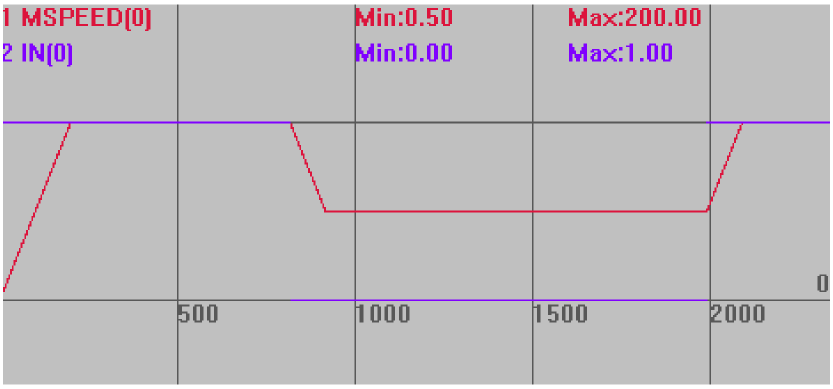
JOGSPEED--JOG Speed

Type	Axis Parameters
Description	**JOG motion speed, unit is units/s.** When REV_JOG or FWD_JOG is activated, motor will run at speed of JOGSPEED slowly. When input port is loosened, motion will stop.
Grammar	JOGSPEED= value, VAR1=JOGSPEED
Controller	General
Example	**Example One:** BASE(0) 'select axis 0 DPOS=0 'axis 0 clears UNITS=100 'pulse amount SPEED=100 'main axis speed is 100 units/s ACCEL=1000 'acceleration is 1000 units/s/s DECEL=1000 'deceleration is 1000 units/s/s TRIGGER 'trigger oscilloscope automatically **JOGSPEED**=50 'JOG speed is 50 FWD_JOG=0 'IN0 as positive JOG input REV_JOG=1 'IN1 as negative JOG input INVERT_IN(0,ON) 'reverse signal INVERT_IN(1,ON) When IN0=ON, axis 0 moves at speed of 50 in positive direction. When IN1=ON, axis 0 moves at speed of 50 in negative direction. When IN0=ON and IN1=ON, axis 0 moves at speed of 50 in positive direction. Speed Curve: MSPPED(0) vertical scale 100

	
Instruction	REV_JOG, FWD_JOG, FAST_JOG

FHOLD_IN--Hold Input Mapping

Type	Axis Parameters
Description	**Hold related input number, -1 means invalid.** If there is input signal, then speed of motion axis is FHSPEED, present is not cancelled. when input signal is cancelled, then motion speed will follow speed defined in procedure. See example one. Input is valid when signal is OFF in ZMC controller, do use INVERT_IN to reverse the electric level. (except for ECI) This parameter is only valid in speed control mode (instruction with sp suffix). If motion is not controlled by speed, such as, CAMBOX, CONNECT, MOVELINK, then it will not activate.
Grammar	VAR1 = FHOLD_IN, FHOLD_IN = expression
Controller	General
Example	BASE(0) 'select axis 0 DPOS=0 'coordinate clears UNITS=100 ATYPE=1 SPEED=100 ACCEL=500 'acceleration is 500 units/s/s DECEL=500 FORCE_SPEED=200 'set speed as 200 units/s FHSPEED=200 'set hold speed as 200units/s **FHOLD_IN**(0)=0 'set hold input of axis 0 as IN0 INVERT_IN(0,ON) 'reverse electric level TRIGGER 'trigger oscilloscope automatically VMOVE(1) 'continuous motion

	When IN0=OFF, axis moves at FORCE_SPEED of 200. When IN0=ON, axis moves at FHSPEED of 100, turn IN0 into off, speed becomes 200. Speed Curve: MSPEED(0) vertical scale 200 
Instruction	FHSPEED

FHSPEED--Hold Speed

Type	Axis Parameters
Description	**Axis holds speed, the speed when FHOLD_IN is activated, unit is units/s.** When input position keeps hold status, it can move at this speed.
Grammar	VAR1 = FHSPEED, FHSPEED = expression
Controller	General
Example	See example in FHOLD_IN
Instruction	FHOLD_IN, SPEED

11.10 Instructions Relate to Encoder

ENCODER—Original Value of Encoder

Type	Axis Status
Description	**Original value of encoder hardware register.** Inner parameters are only valid after correcting ATYPE setting. If drive encoder is multiturn, then multiturn value is read.
Grammar	VAR1 = ENCODER(axis No.)
Controller	General
Example	?***ENCODER** 'print encoder value, result:0 0 0 0 0 0 0 0 0 0 0 0
Instruction	MPOS, ENCODER_RATIO

ENCODER_STATUS--Encoder Status

Type	Axis Status
Description	**Status of encoder EA, EB, EZ.**
Grammar	VAR1 = ENCODER_STATUS <table><tr><td>Bit</td><td>Value</td><td>Description</td></tr><tr><td>0</td><td>1</td><td>EA Status</td></tr><tr><td>1</td><td>2</td><td>EB Status</td></tr><tr><td>2</td><td>4</td><td>EZ Status</td></tr></table>
Controller	General
Example	?***ENCODER_STATUS** 'print encoder status of all axes
Instruction	ATYPE, MPOS

ENCODER_FILTER—Encoder Filter

Type	Axis parameters
Description	**Inner encoder filter setting, motion speed of belt encoder can be uniform, default value is 1, from 0.001~1.** Default 1- no filter, 0.5- 2 periods filter, 0.25- 4 periods filter. ZMC5XXX series controllers support, ZMC 4XXX series with firmware version above 170706 supports.
grammar	ENCODER_FILTER = VALUE
controller	General
Instruction	ENCODER_RATIO

PP_STEP--Encoder Internal Proportion

Type	Axis Parameters
Description	**Internal inputs of encoder will multiply this parameter.** The parameter effect superposes ENCODER_RATIO, default value is 1.
Grammar	PP_STEP = value
Controller	General
Instruction	ENCODER_RATIO

ENCODER_BITS – Encoder Absolute Value Setting

Type	Axis Parameters
Description	**Set SSI/BISS encoder absolute value.**

<table>
<tr><td>Grammar</td><td>ENCODER_BITS = VALUE
<table>
<tr><td>Encoder Type</td><td>Bit</td><td>Value</td><td>Function Description</td></tr>
<tr><td rowspan="4">SSI/BISS</td><td>Bit0-5</td><td>0-32</td><td>The total bit of encoder communication.</td></tr>
<tr><td>Bit6</td><td>64</td><td>Whether it is Gray code</td></tr>
<tr><td>Bit8-10</td><td>256*(0<n<15)</td><td>Invalid bit, BISS = 8 (usually)</td></tr>
<tr><td>Bit16-18</td><td>65536*(0<n<7)</td><td>Frequency division adjustment, default 0-2MHz.</td></tr>
</table>
ATYPE = 48 ‘SSI absolute encoder
ATYPE = 49 ‘BISS absolute encoder
Before using this commands, axis mapping must be done.</td></tr>
<tr><td>Controller</td><td>General</td></tr>
<tr><td>Example</td><td>SSI Example:
BASE(n)
AXIS_ADDRESS = (-1<<16) + 4 ‘map the fourth physical axis position
ENCODER_BITS = 26 ‘26-bit absolute value
ATYPE = 48

BISS Example:
BASE(n)
AXIS_ADDRESS = (-1<<16) + 5 ‘map the fifth physical axis position
ENCODER_BITS = 26
‘26-bit absolute value, it is with 8 state bits automatically
ATYPE = 49</td></tr>
</table>

DRIVE_POSMIN – Encoder Transfer Original Min Value

Type	Axis Parameters
Description	**Set the minimal value of original value range transferred by encoder.**
Grammar	DRIVE_POSMIN = VALUE Set before modifying ATYPE. If MPOS also does coordinates loop, modify REP_OPTION. Range: 32-bit integer, 0 to 2^32-1 DRIVE_POSMAX = 2^32-1 DRIVE_POSMIN = -2^31
Controller	General
Example	BASE(0) AXIS_ADDRESS=(0<<16)+nodenum ENCODER_ID=$60640020 DRIVE_POSMIN=0 DRIVE_POSMAX = 2^18-1 ‘PDOBUFF value is in this range

	ATYPE=25
Instruction	DRIVE_POSMAX

DRIVE_POSMAN – Encoder Transfer Original Max Value

Type	Axis Parameters
Description	**Set the maximal value of original value range transferred by encoder.**
Grammar	DRIVE_POSMAX = VALUE Set before modifying ATYPE. If MPOS also does coordinates loop, modify REP_OPTION. Range: 32-bit integer, 0 to 2^32-1 DRIVE_POSMAX = 2^32-1 DRIVE_POSMIN = -2^31
Controller	General
Example	Same as DRIVE_POSMIN.
Instruction	DRIVE_POSMIN

11.11 Instructions Relate to Latch

REGIST-Position Latch

Type	Position Latch Instruction
Description	**REGIST is used to latch the measured position of axis.** It can latch encoder axis and bus axis. Different controller models can latch different axis types. ZMC4XX series controllers or above with the latest firmware support latching virtual axis and pulse axis. Support drive latch in EtherCAT based motion controller through IO of drive, see the instructions for details. For Rtex, it only supports controller latch. ZMC4XX series controller or above supports 4 latch channels. channels refer to MARK, MARKB, MARKC, MARKD, using REG_INPUTS to define high-speed input that corresponds to each latch channel, default IN ports are 0, 1, 2, 3. For details, please refer to IN of user manual. Latches in EtherCAT based drive and latches in controllers can be used synchronously, but it should support 4-channel latching. For bus latching, please use MARK and MARKB channels. For controller latching, please use MARKC, MARKD channels. Latch mode corresponding rising edge and falling edge are determined by

	PNP or NPN. When latch occurs, MARK status of axis will turn into ON, position latch will be saved in REG_POS. Each axis' latch channel is mutually independently.
Grammar	**Grammar 1:** REGIST(mode) mode: latch mode Rising or falling edge is determined by the inner status of controller. Different inputs type may cause difference, which needs to confirm depend on the actual latch edge. (if sets as rising edge, latch will be triggered when external input port change linking status to cut-ff status in a flash. Instead, if sets as falling edge, latch will be triggered when external input port changed its cut-off status to linking status in a flash.) Pulse axis type uses R0, R1 and Z these 3 latched generally, bus axis type uses R2 and R3 latches:

value	Description
1	Save absolute position in REG_POS when meets rising edge of Z pulse
2	Save absolute position in REG_POS when meets falling edge of Z pulse
3	Save absolute position in REG_POS when meets rising edge of R0 signal
4	Save absolute position in REG_POS when meets falling edge of R0 signal
6	Save absolute position in REG_POS when meets rising edge of R0 signal, save absolute position in REG_POSB when meets rising edge of Z signal
7	Save absolute position in REG_POS when meets rising edge of R0 signal, save absolute position in REG_POSB when meets falling edge of Z signal
8	Save absolute position in REG_POS when meets falling edge of R0 signal, save absolute position in REG_POSB when meets rising edge of Z signal
9	Save absolute position in REG_POS when meets falling edge of R0 signal, save absolute position in REG_POSB when meets falling edge of Z signal
10	Save absolute position in REG_POS when meets rising edge of R0 signal, save absolute position in REG_POSB when meets rising edge of R1 signal.
11	Save absolute position in REG_POS when meets rising edge of R0 signal, save absolute position in REG_POSB when meets falling edge of R1 signal.
12	Save absolute position in REG_POS when meets falling edge of

	R0 signal, save absolute position in REG_POSB when meets rising edge of R1 signal.
13	Save absolute position in REG_POS when meets falling edge of R0 signal, save absolute position in REG_POSB when meets falling edge of R1 signal.
14	Save absolute position in REG_POSB when meets rising edge of R1 signal (in controller with firmware version above 14XXXX, each latch channel is individual and supports 4 latch channels.)
15	Save absolute position in REG_POSB when meets falling edge of R1 signal.
16	Save absolute position in REG_POSB when meets rising edge of Z signal.
17	Save absolute position in REG_POSB when meets falling edge of Z signal.
18	Save absolute position in REG_POSC when meets rising edge of R2 signal.
19	Save absolute position in REG_POSC when meets falling edge of R2 signal.
20	Save absolute position in REG_POSD when meets rising edge of R3 signal.
21	Save absolute position in REG_POSD when meets falling edge of R3 signal.

Grammar 2:

REGIST(100+mode, tableindexn, numes)

mode: latch mode

tableindex: table position of saving continuously latched content. The first table element saves latched numbers, later saves latched coordinates, maximum number = numes-1, write cycle when overflow.

numes: occupied table numbers.

Through mode + 100 to support continuous latch, the result is saved in TABLE

Continuous latch to two channels separately, which can realize rising and falling edge of continuous latch.

ECI: with firmware version above 20150829.

4XXX series Controller: with firmware version above 20170523.

100+mode: only used in single channel, +100 means use continuous latch.

value	Description
1	Save absolute position in REG_POS when meets rising edge of Z pulse

	2	Save absolute position in REG_POS when meets falling edge of Z pulse
	3	Save absolute position in REG_POS when meets rising edge of R0 signal
	4	Save absolute position in REG_POS when meets falling edge of R0 signal
	14	Save absolute position in REG_POSB when meets rising edge of R1 signal
	15	Save absolute position in REG_POSB when meets falling edge of R1 signal
	16	Save absolute position in REG_POSB when meets rising edge of Z signal
	17	Save absolute position in REG_POSB when meets falling edge of Z signal
	18	Save absolute position in REG_POSC when meets rising edge of R2 signal
	19	Save absolute position in REG_POSC when meets falling edge of R2 signal
	20	Save absolute position in REG_POSD when meets rising edge of R3 signal
	21	Save absolute position in REG_POSD when meets falling edge of R3 signal
	23	Save absolute position in REG_POSB when meets rising edge of R0 signal
	24	Save absolute position in REG_POSB when meets falling edge of R0 signal
	33	Save absolute position in REG_POS when meets rising edge of R0 signal, the next time switches to falling edge. Switch in turn.
	34	Save absolute position in REG_POS when meets falling edge of R0 signal, the next time switches to rising edge. Switch in turn.
	35	Save absolute position in REG_POSB when meets rising edge of R1 signal, the next time switches to falling edge. Switch in turn.
	36	Save absolute position in REG_POSB when meets falling edge of R1 signal, the next time switches to rising edge. Switch in turn.
Controller	Controllers with latch input ports	
Example (based on ZMC432)	**Example1: latch position of pulse axis 0 when meets jumping edge of R0 signal, and print** BASE(0) REG_INPUTS=0 'R0-R3 are matched with input port 0 ATYPE=1 'pulse axis **REGIST**(3) 'select R0 latch mode	

```
WAIT UNTIL MARK            'wait until latch be triggered
PRINT REG_POS              'print latch position
```

Example2: latch position of encoder axis 0 when meets jumping edge of R1 signal, and print

```
BASE(0)
REG_INPUTS=0               'R0-R3 are matched with input port 0
ATYPE=3                    'encoder axis
REGIST(14)                 'select R1 latch mode
WAIT UNTIL MARK            'wait until latch be triggered
PRINT REG_POS              'print latch position
```

Example3: latch position of EtherCAT bus axis 0 when meets jumping edge of R2 signal, and print

```
'before latch, please do bus initialization enable, use R2R3 latch.
BASE(0)
REG_INPUTS=$1000           'map latch channel R3-R0 into input 1,0,0,0
REGIST(imode)
IF imode = 15 OR imode = 19 THEN       'latch channel R2
    WAIT UNTIL MARKC           'probe
    ?“mode”, Imode, “latch position REG_POSC”, REG_POSC
ELSELF imode = 20 OR imode = 21 THEN        'latch channel R3
    WAIT UNTIL MARKD           'wait until latch be triggered
    ?“mode”, Imode, “latch position REG_POSD”, REG_POSD
ENDIF
```

Example 4: transfer latched position data between controller and PC, which is used to capture motion, then get the actual position through latch function while the Capture happened.

```
GLOBAL g_start
GLOBAL g_posx, g_posy
WHILE 1
    WAIT UNTIL g_start=1    'wait until PC is triggered.
    REGIST(4) AXIS(0)       'latch input 0, when 24V become 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
```

Example 5: 100+mode continuous latch

```
DIM num
num=1
```

	BASE(6)
	ATYPE=6
	REGIST(100+4,5,100) 'cycle automatically, no need to write in WHILE cycle, table(5) saves latched times, table(6)-table(105) save latched data over 99 every time, table(5) clears 0, restarts to memorize data from table(6).
	WHILE 1
	local test
	test = table (5)
	WAIT UNTIL table (5)
	?reg_pos,TABLE(num),TABLE(0) 'print
	IF num=100 THEN
	num=1
	ELSE
	num=num+1
	ENDIF
	WA 1 'delay 1ms, anti-shake
	WEND
	
	Example 6: bus drive latch, which should be configured DRIVE_PROFILE with probe mode.
	BASE(iaxis) 'select axis No. to latch position
	REGIST(imode) 'latch mode
	IF imode = 3 OR imode = 4 THEN
	WAIT UNITL MARK 'probe 1
	?“mode”, imode, “latch position REG_POS”, REG_POS
	DELAY(100)
	ELSELF imode = 14 OR imode = 15 THEN
	WAIT UNITL MARKB 'wait until latch to be triggered
	?“mode”, imode, “latch position REG_POS”, REG_POS
	DELAY(100)
	ELSELF imode = 11 TEHN
	WAIT UNTIL MARK OR MARKB 'wait until latch to be triggered
	IF MARK THEN
	?“mode”, imode, “latch position REG_POS”, REG_POS
	WAIT UNTIL MARKB
	?“latch position REG_POSB”, REG_POSB
	ELSELF MARKB THEN
	?“mode”, Imode, “latch position REG_POSB”, REG_POSB
	WAIT UNTIL MARK
	?“latch position REG_POSB”, REG_POS
	ENDIF
	DELAY(100)
	ENDIF

Instructions	MARK, MARKB, REG_POS, REG_POSB

REG_INPUTS--Latch Input Mapping

<table>
<tr><td>Type</td><td>Axis Status</td></tr>
<tr><td>Description</td><td>Configure inputs of position latch signals: R0-R3, every 4 bits was mapped to one latch signal input.</td></tr>
<tr><td>Grammar</td><td>VAR1 = REG_INPUTS
<table>
<tr><td>Bit</td><td>Latch signal</td><td>Input range (ZMC306E)</td></tr>
<tr><td>Bit0-3</td><td>R0</td><td>0-3</td></tr>
<tr><td>Bit4-7</td><td>R1</td><td>0-3</td></tr>
<tr><td>Bit8-11</td><td>R2</td><td>0-3</td></tr>
<tr><td>Bit12-15</td><td>R3</td><td>0-3</td></tr>
</table>
Inputs Range: Different controller modes have different inputs ranges.</td></tr>
<tr><td>Controller</td><td>Some ZMC3XX series and ZMC4XX series with latest firmware version support latch function.</td></tr>
<tr><td>Example</td><td>BASE(6)
ATYPE=3
REG_INPUTS=$3210 ‘R0-R3 are mapped to inputs:0,1.2.3
REG_INPUTS=$1111 ‘R0-R3 are all mapped to input 1.</td></tr>
<tr><td>Instruction</td><td>REGIST</td></tr>
</table>

MARK--Latch Trigger

<table>
<tr><td>Type</td><td>Axis Status</td></tr>
<tr><td>Description</td><td>Return value to check if position latch happened.
When REGIST is executing, MARK is true, returned value is -1, once execution of REGIST is finished, MARK becomes false, returned value is 0.</td></tr>
<tr><td>Grammar</td><td>VAR1 = MARK</td></tr>
<tr><td>Controller</td><td>General</td></tr>
<tr><td>Example</td><td>BASE(0) 'select axis 0
MOVE(100) 'move 100units
REGIST(3) 'rising edge trigger, get signal R0.
WAIT UNTIL MARK 'wait to be triggered.</td></tr>
<tr><td>Instruction</td><td>REG_POS, REGIST</td></tr>
</table>

MARKB--Latch 2 Trigger

<table>
<tr><td>Type</td><td>Axis Status</td></tr>
<tr><td>Description</td><td>Return value to check if position latch 2 happened.
When REGIST is executing, MARKB becomes true, returned value is -1, once execution of REGIST is finished, MARKB becomes false, returned value is 0.</td></tr>
<tr><td>Grammar</td><td>VAR1 = MARKB</td></tr>
</table>

Controller	General
Example	BASE(0) 'select axis 0 MOVE(100) 'move 100units REGIST(14) 'rising edge trigger, get signal R1. WAIT UNTIL **MARKB** 'wait to be triggered.
Instruction	REG_POSB, REGIST

MARKC-- Latch 3 Trigger

Type	Axis Status
Description	**Return value to check if position latch 3 happened.** When REGIST is executing, MARKC is true, returned value is -1, once REGIST is finished, MARKC becomes false, returned value is 0.
Grammar	VAR1 = MARKC
Controller	General
Example	BASE(0) 'select axis 0 MOVE(100) 'move100units REGIST(18) 'rising edge trigger, get signal R2 WAIT UNTIL **MARKC** 'wait to be triggered.
Instruction	REG_POSC, REGIST

MARKD-- Latch 4 Trigger

Type	Axis Status
Description	**Return value to check if position latch 4 happened.** When REGIST is executing, MARKD is true, returned value is -1, once REGIST is finished, MARKD becomes false, returned value is 0.
Grammar	VAR1 = MARKD
Controller	General
Example	BASE(0) 'select axis 0 MOVE(100) 'move 100units REGIST(20) 'rising edge trigger, get signal R3 WAIT UNTIL **MARKD** 'wait to be triggered.
Instruction	REG_POSD, REGIST

OPEN_WIN--Coordinate Range for Latch Starts

Type	Axis Parameters
Description	Start coordinate range point of position latch **Reserved**
Grammar	OPEN_WIN = pos

Instruction	REGIST, CLOSE_WIN

CLOSE_WIN-- Coordinate Range for Latch Ends

Type	Axis Parameters
Description	End coordinate range point of position latch. **Reserved**
Grammar	CLOSE_WIN = pos
Instruction	REGIST, OPEN_WIN

REG_POS--Latch Position

Type	Axis Status
Description	**Save latched measurement feedback position (MPOS), unit is units.**
Grammar	VAR1 = REG_POS (axis No.)
Controller	General
Example	MOVE(100) 'move 100units REGIST(3) 'trigger when meets rising edge of signal R0. WAIT UNTIL MARK 'wait until position latch happens PRINT **REG_POS** 'print latched position
Instruction	REGIST, MARK

REG_POSB--Latch 2 Position

Type	Axis Status
Description	**Return MPOS latched by register 2, unit is units.**
Grammar	VAR1 = REG_POSB (axis No.)
Controller	General
Example	MOVE(100) 'move 100units REGIST(16) 'trigger when meets rising edge signal EZ WAIT UNTIL MARKB 'wait until the second latch happens. PRINT **REG_POSB** 'print latched position
Instruction	REGIST, MARKB

REG_POSC--Latch 3 Position

Type	Axis Status
Description	**Return MPOS latched by register 3, unit is units.**
Grammar	VAR1 = REG_POSC (axis No.)
Controller	General

Example	MOVE(100) 'move 100units REGIST(18) 'trigger when meet rising edge signal WAIT UNTIL MARKC 'wait until the second latch happens. PRINT **REG_POSC** 'print latched position
Instruction	REGIST, MARKC

REG_POSD--Latch 4 Position

Type	Axis Status
Description	**Return MPOS latched by register 4, unit is units.**
Grammar	VAR1 = REG_POSD (axis No.)
Controller	General
Example	MOVE(100) 'move 100units REGIST(20) 'trigger when meets rising edge signal WAIT UNTIL MARKD 'wait until the second latch happens. PRINT **REG_POSD** 'print latched position
Instruction	REGIST, MARKD

11.12 Position Limit Parameter Instructions

FS_LIMIT--Soft Positive Limit

Type	Axis Parameters
Description	**Soft positive position limit setting, unit is units.** When FS_LIMIT is bigger than REP_DIST, FS_LIMIT will become invalid, and soft positive position limit function is forbidden. If needs to cancel soft positive position limit, it is not recommended to modify value of REP_DIST, it is better to set FS_LIMIT as a bigger value. Default value of FS_LIMIT is 200000000. Soft position limit can not be as signal reference of homing when use DATUM.
Grammar	VAR1 =FS_LIMIT，FS_LIMIT = expression VAR1 =FS_LIMIT(axisnum)，FS_LIMIT = expression (axisnum)
Controller	General
Example	BASE(0) 'select axis 0 ATYPE=1 'axis type setting UNITS=100 'pulse amount is 100 DPOS=0 'coordinate clears SPEED=100 'speed is 100units/s ACCEL=1000 'acceleration is 1000 units/s/s

	FS_LIMIT=200 'set positive limit as 200units MOVE(300) 'move 300units When axis reaches 200, it will stop, and show error: Axis:0 AXISSTATUS: 200h, FSOFT. If wants to move axis, then only motion in negative direction is allowed. It can be cancelled through set a bigger value. **FS_LIMIT**=2000000 'cancel soft positive position limit setting.
Instruction	RS_LIMIT, FWD_IN, REV_IN

RS_LIMIT--Soft Negative Limit

Type	Axis Parameters
Description	**Soft negative position limit setting, unit is units.** If RS_LIMIT is bigger than REP_DIST, RS_LIMIT will become invalid, and soft negative position limit function is forbidden. If needs to cancel Soft negative position limit, it is not recommended to modify value of REP_DIST, it is better to set RS_LIMIT as a bigger value. Default value of RS_LIMIT is -200000000. Soft position limit can not be as signal reference of homing when use DATUM.
Grammar	VAR1 =RS_LIMIT, RS_LIMIT = expression VAR1 =RS_LIMIT(axisnum), RS_LIMIT = expression (axisnum)
Controller	General
Example	**RS_LIMIT** = -50 'set soft negative limit is 50 units **RS_LIMIT**= - 2000000 'cancel soft negative limit
Instruction	FS_LIMIT, FWD_IN, REV_IN

FWD_IN--Positive Limit Mapping Input

Type	Axis Parameters
Description	**Input number related to positive position limit input in hardware, -1 is invalid.** **When limit signal comes, axes motion will stop immediately at speed of FASTDEC.** Input is valid when signal is off in ZMC controller, do use INVERT_IN to reverse the electric level. (except for ECI)
Grammar	VAR1 = FWD_IN, FWD_IN = expression VAR1 =FWD_IN(axisnum), FWD_IN = expression (axisnum)
Controller	General

Example	BASE(0,1,2,3) 'select axis0,1,2,3 **FWD_IN**=6,7,8,9 'set positive limit inputs. INVERT_IN(6,ON) 'reverse signal INVERT_IN(7,ON) INVERT_IN(8,ON) INVERT_IN(9,ON) When limit signals come, axis status will show error: Axis:0AXISSTATUS:10h,FWD. then only motion in negative direction is allowed.
Instruction	REV_IN, FS_LIMIT, FASTDEC

REV_IN--Negative Limit Mapping

Type	Axis Parameters
Description	**Input number relates to negative position limit input in hardware, -1 is invalid.** **When limit signal comes, axes motion will stop immediately at speed of FASTDEC.** Input is valid when signal is off in ZMC controller, do use INVERT_IN to reverse the electric level. (except for ECI)
Grammar	VAR1 = REV_IN, REV_IN = expression VAR1 =REV_IN(axisnum), REV_IN = expression (axisnum)
Controller	General
Example	See Example in FWD_IN
Instruction	FWD_IN, RS_LIMIT,FASTDEC

ALM_IN--Alarm Input

Type	Axis Parameters
Description	Drive alarm input configuration, -1 means invalid. Once controller gets alarm signal, all axes will stop, acceleration will obey FASTDEC. After Alarm input was set, ZMC controller is valid when off, if signal is normally opened, then use INVERT_IN to reverse electric level; while in ECI controller, it is valid when ON, if signal is normally closed, then use INVERT_IN to reverse electric level.
Grammar	VAR1 = ALM_IN, ALM_IN = expression VAR1 =ALM_IN(axisnum), ALM_IN = expression (axisnum)
Controller	General

Example	BASE(0,1) **ALM_IN** = 10,11 'map alarm signal of axis 0 to input 10, and alarm signal of axis 1 to input 11 INVERT_IN(10,ON) 'reverse electric level. INVERT_IN(11,ON)
Instruction	DATUM_IN, FWD_IN, REV_IN, INVERT_IN, FASTDEC

11.13 On-Position Parameter Instructions

IN_POS – On Position Mark

Type	Axis Parameters
Description	**Read whether axis arrives the position or not, after axis stops, return value -1 means on position, return value 0 means not on position.** IN_PSO_DIST and IN_POS_SPEED must be configured well, or AXISINP_IN must be configured well. When this function is not used, directly use IDLE to judge motion finish status.
Grammar	VAR1 = AXISEMG_IN, AXISEMG_IN = expression VAR1 = AXISEMG_IN (axisnum), AXISEMG_IN (axisnum) = expression
Controller	General, valid in 4xx series controllers with the latest firmware version.
Example	BASE(1) DPOS=0 UNITS=1000 SPEED=100 'set speed as 100 ACCEL=500 DECEL=500 'set deceleration as 500 FASTDEC=2000 'set fast deceleration as 2000 AXISINP_IN(1) = 0 'set input 0 as on-position signal of axis 1 INVERT_IN(0,ON) 'signal reverse MOVE(1000) AXIS(1) ?IN_POS(1) 'print value: 0, not on position WAIT UNTIL IDLE(1) 'wait until axis 1 stops ?IN_POS(1) 'on position signal is triggered, print value: -1, it arrives position
Instruction	IN_POS_DIST, IN_POS_SPEED, AXISINP_IN

AXISINP_IN – On-position Signal Input

Type	Axis Parameters
Description	**Configure axis on-position input, default (-1) means not to use on-**

	position signal. When controller on-position signal took effect, and axis stopped, on-position mark takes effect. When on-position input was set, ZMC controller default OFF is valid, and commonly opened signal uses INVERT_IN to reverse the electric level. ECI controller default ON is valid, commonly-closed signal uses INVERT_IN to reverse electric level.
Grammar	VAR1 = AXISINP_IN, A XISINP_IN = expression VAR1 = AXISINP_IN (axisnum), AXISINP_IN (axisnum) = expression
Controller	General, valid in 4xx series controllers with the latest firmware version.
Example	BASE(0) 'select axis 0 ATYPE=1 UNITS=100 'set pulse amount as 100 DPOS=0 'clear coordinates as 0 SPEED=100 'set speed as 100units/s ACCEL=1000 'set acceleration as 1000units/s/s DECEL=1000 'set deceleration as 1000units/s/s AXISINP_IN(0) = 0 'set input 0 as on-position of axis 0 INVERT_IN(0,ON) MOVE(10000) AXIS(0) TRIGGER ?IN_POS(0) WAIT UNTIL IDLE(0) 'after stopped, on-position changes ?IN_POS(0)
Instruction	IN_POS_DIST, IN_POS_SPEED, IN_POS, INVERT_IN,

IN_POS_DIST – On-position Distance

Type	Axis Parameters
Description	**Configure on-position distance, FE is less than this distance, and MSPEED is less than IN_POS_SPEED, it means it arrives the position.** Parameters start from 0, when this parameter and on-position parameter are used together, on-position mark is controlled by on-position signal.
Grammar	VAR1 = IN_POS_DIST, IN_POS_DIST = expression VAR1 = IN_POS_DIST (axisnum), IN_POS_DIST (axisnum) = expression
Controller	General, valid in 4xx series controllers with the latest firmware version.
Example	BASE(0) ATYPE=4 'with encoder feedback UNITS=1000 SPEED=100 'set speed as 100 ACCEL=1000 DECEL=1000 'set deceleration as 1000 FASTDEC=10000 'set fast deceleration as 10000

	DPOS=0 AXISINP_IN(0) = -1 'cancel on-position signal IN_POS_DIST = 0.5 'on-position distance IN_POS_SPEED =0.5 'on-position speed MOVE(100) DELAY(100) ?FE ?MSPEED ?IN_POS 'print on-position mark WAIT UNTIL IDLE(0) DELAY(100) ?"---------------------" ?FE ?MSPEED ?IN_POS
Instruction	IN_POS_SPEED, IN_POS, AXISINP_IN

IN_POS_SPEED – On-position Speed

Type	Axis Parameters
Description	**Configure on-position speed, MSPEED is less than this speed, and FE is less than IN_POS_DIST, it means it arrives the position.** Parameters start from 0, when this parameter and on-position parameter are used together, on-position mark is controlled by on-position signal.
Grammar	VAR1 = IN_POS_SPEED, IN_POS_SPEED = expression VAR1 = IN_POS_SPEED (axisnum), IN_POS_SPEED (axisnum) = expression
Controller	General, valid in 4xx series controllers with the latest firmware version.
Example	BASE(0) ATYPE=4 'with encoder feedback UNITS=1000 SPEED=100 'set speed as 100 ACCEL=1000 DECEL=1000 'set deceleration as 1000 FASTDEC=10000 'set fast deceleration as 10000 DPOS=0 AXISINP_IN(0) = -1 'cancel on-position signal IN_POS_DIST = 0.5 'on-position distance IN_POS_SPEED =0.5 'on-position speed MOVE(100) DELAY(100) ?FE ?MSPEED ?IN_POS 'print on-position mark

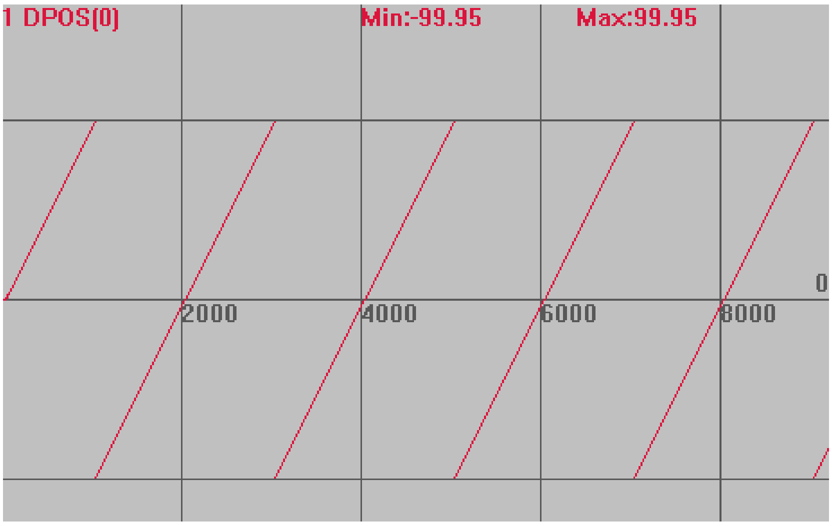
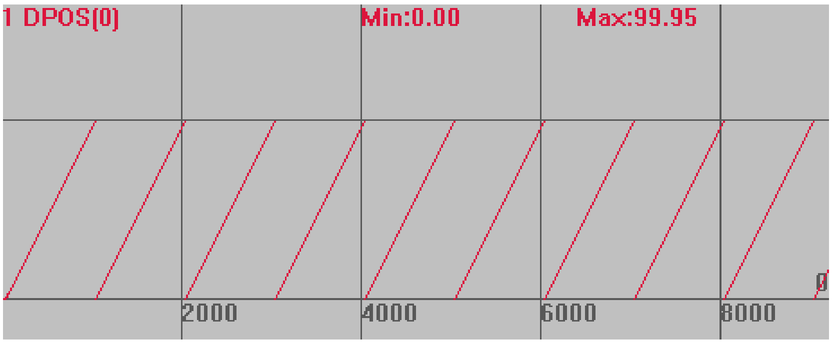
	WAIT UNTIL IDLE(0) DELAY(100) ?"----------------------" ?FE ?MSPEED ?IN_POS
Instruction	IN_POS_DIST, IN_POS, AXISINP_IN

11.14 Range Limit Parameter Instructions

REP_OPTION--Coordinate Cycle Mode

<table>
<tr><td>Type</td><td>Axis Parameters</td></tr>
<tr><td>Description</td><td>Repeat set the coordinate.

It can be used to limit main axis coordinate cycle range of CAM type motion and to realize continuous of multiple CAM profiles.
When in absolute mode, if target position is between coordinate cycle range, then motion is correct, or motion is incorrect.
No influence on relative motion.</td></tr>
<tr><td>Grammar</td><td>VAR1 = REP_OPTION, REP_OPTION = opt
opt, different bits indicate different meanings.
<table>
<tr><td>Bit</td><td>Value</td><td>Description</td></tr>
<tr><td>0</td><td>1</td><td>0-cycle range – REP_DIST to + REP_DIST.
1- cycle range 0 to + REP_DIST.</td></tr>
<tr><td>1</td><td>2</td><td>1 means repeat motion is forbidden in CAMBOX and MOVELIK, once it activates, value recovers to 0.</td></tr>
<tr><td>2</td><td>4</td><td>reserved</td></tr>
<tr><td>4</td><td>16</td><td>1-Don’t use REP_DIST.
0-Use REP_DIST.</td></tr>
</table></td></tr>
<tr><td>Controller</td><td>General</td></tr>
<tr><td>Example</td><td>BASE(0) 'select axis 0
ATYPE=1
UNITS=100 'pulse amount is 100
DPOS=0 'coordinate clears
SPEED=100 'speed is 100units/s
ACCEL=1000 'acceleration is 1000units/s/s
DECEL=1000 'deceleration is 1000units/s/s
REP_DIST=100 'set cycle coordinate range
REP_OPTION=0 'set cycle mode
TRIGGER 'trigger oscilloscope automatically
VMOVE(1) 'continuous motion</td></tr>
</table>

	Coordinate Curve: DPOS(0) vertical scale 100  REP_OPTION=1, MSPEED(0) vertical scale 100 
Instruction	CAMBOX, MOVELINK, REP_DIST

REP_DIST--Coordinate Cycle Position

Type	Axis Parameters
Description	**Auto cycle DPOS and MPOS of axis through REP_OPTION setting.**
Grammar	VAR1 = REP_DIST, REP_DIST = expression
Controller	General
Example	See REP_OPTION as reference
Instruction	REP_OPTION

FE—Current Follow-up Error

Type	Axis Status
Description	**Follow-up error, value=DPOS-MPOS.**

Grammar	VAR1=DPOS
Controller	General
Instruction	MPOS, DPOS

FE_RANGE-- Follow-up Error when Alarm

Type	Axis Parameters
Description	**Follow-up error when alarm happens.**
Grammar	VAR1 = FE_RANGE, FE_RANGE = expression Valid: set assigned value through FE_RANGE AXIS (axis) method.
Example	Refer to SERVO.
Instruction	FE, FE_LIMIT, P_GAIN, D_GAIN, I_GAIN, OV_GAIN, VFF_GAIN, FE_LIMIT.

FE_LIMIT--Maximum Follow-Up Error

Type	Axis Parameters
Description	**Allowed maximum follow-up error, default: 3** When follow error exceeds, real-time error will be caused, and enable relay (WDOG) clears, which means it prevents other generators from running. This limit usually is used for protect default status, such as, machine is locked, encoder feedback is lost, etc. It will report alarms when timeout, 2h/100h/102h are reported usually for AXISSTATUS.
Grammar	VAR1 = FE_LIMIT, FE_LIMIT= expression Valid: set assigned value through FE_LIMIT AXIS (axis) method.
Example	Refer to SERVO.
Instruction	FE, P_GAIN, D_GAIN, I_GAIN, OV_GAIN, VFF_GAIN, FE_LIMIT. SERVO

11.15 Advanced Setting Instruction

INVERT_STEP--Pulse Mode Setting

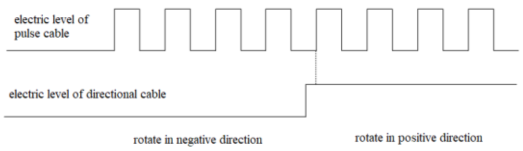
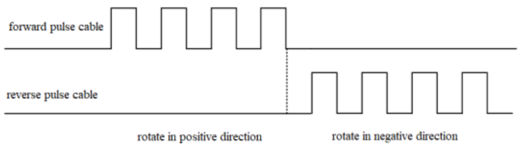
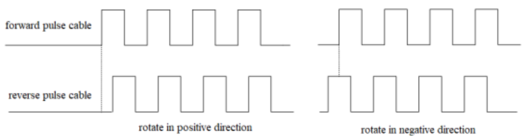
Type	Axis Parameters
Description	**Servo/Step pulse output mode setting.** There are three modes: pulse direction, double pulse and quadrature pulse, the default mode is pulse direction control (mode 0). Now, only some controllers support quadrature pulse. MPOS information involves many complicate modes, such as, MOVE_OP high-precision output mode, so controller can’t support modify MPOS

direction at present, if needs, modify drive or other related parameters, such as, Mitsubishi PA 14.

Grammar

INVERT_STEP = mode

parameters： mode (default is 0) lower 8 bits (bit0-7) indicate mode value, as follow:

<table>
<tr><th>Mode value</th><th>Description</th><th>Reference (positive logical mode0</th></tr>
<tr><td>0-3</td><td>pulse direction. Pulse line + direction line.</td><td>
</td></tr>
<tr><td>4-7</td><td>double pulse (or CW/CCW), positive pulse line + negative pulse line.</td><td>
</td></tr>
<tr><td>8-9</td><td>AB output, quadrature pulse (some controllers are customized)</td><td>
</td></tr>
</table>

Electric levels in different modes: if polarity is reverse, the motion direction will be opposite to original direction.

Mode value	Description	Panasonic setting		Mitsubishi setting
		Pr0.06	Pr0.07	PA13
0	Pulse/direction(pulse positive logic) (positive)	0	3	××01h
1	Pulse/direction(pulse negative logic) (positive)	/	/	××11h
2	Pulse/direction(pulse positive logic) (negative)	1	3	××01h
3	Pulse/direction(pulse negative logic) (negative)	/	/	××11h
4	Double pulse (direction negative logic) (positive)	/	/	××10h
5	Double pulse (direction negative logic) (negative)	/	/	××10h
6	Double pulse (direction positive logic) (positive)	1	1	××00h (default)
7	Double pulse (direction positive logic) (negative)	0 (default)	1 (default)	××00h (default)

	Upper 8 bits(bit8-15) indicate protect time of direction changing, unit is microsecond, value is:0-255 Commonly used modes are 0, 2, 6, 7. If mode is set incorrectly, step motor will lose 1 step position when change direction, if can not confirm motor setting, set change protect time as about 100 ms.
Controller	General
Example	Set as pulse direction mode: **INVERT_STEP** = 256*100+0 'protect time is 100ms, mode is 0. Set as double pulse mode: **INVERT_STEP** = 256*100+6 'protect time is 100ms, mode is 6. Check pulse mode setting: Online input instructions to check, as follow: ?INVERT_STEP(0) 'print axis 0 pulse mode setting value ?*INVERT_STEP 'print all axes pulse mode setting value
Instruction	STEP_RATIO

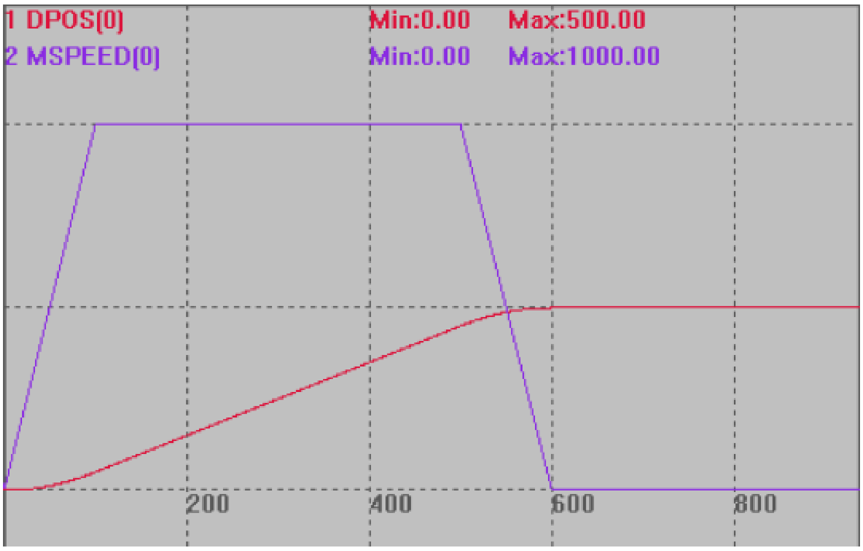
MAX_SPEED--Pulse Frequency Limit

Type	Axis Parameters
Description	**Limit of maximum pulse frequency output.** Once exceed this value, frequency will be limited, and AXISSTATUS will be set. In terms of encoder axis, when set frequency is under 500K, encoder smoothing will start, when set frequency is over 1M, encoder smoothing will be canceled. Default value is 1000000 (the default pulse frequency of old firmware is 500000). When use linear motor, and the speed is too high, it is easy to exceed pulse frequency limit, then it's better to set a bigger value.
Grammar	MAX_SPEED = value
Controller	General
Example	**MAX_SPEED** AXIS(n)=4000000 'set axis n pulse speed limit is 4000000 BASE(6) ATYPE=3 MAX_SPEED =500000 'start encoder filter
Instruction	AXISSTATUS

AXIS_ZSET--Setting of Precision Output

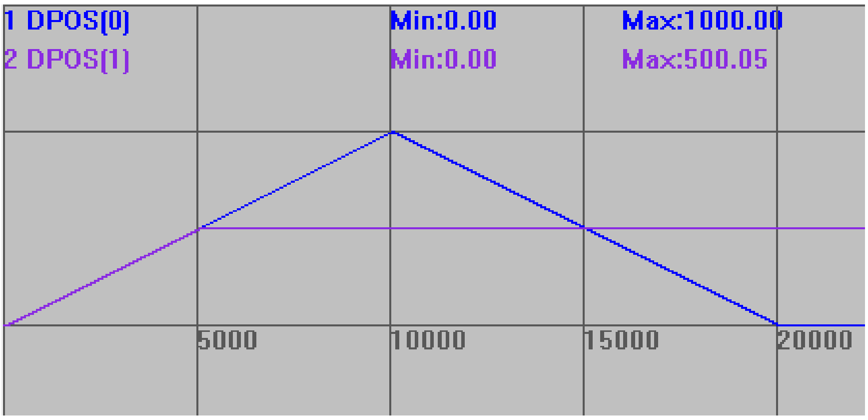
Type	Axis Parameters
Description	**To set precision output function of MOVE_OP, which is used for the**

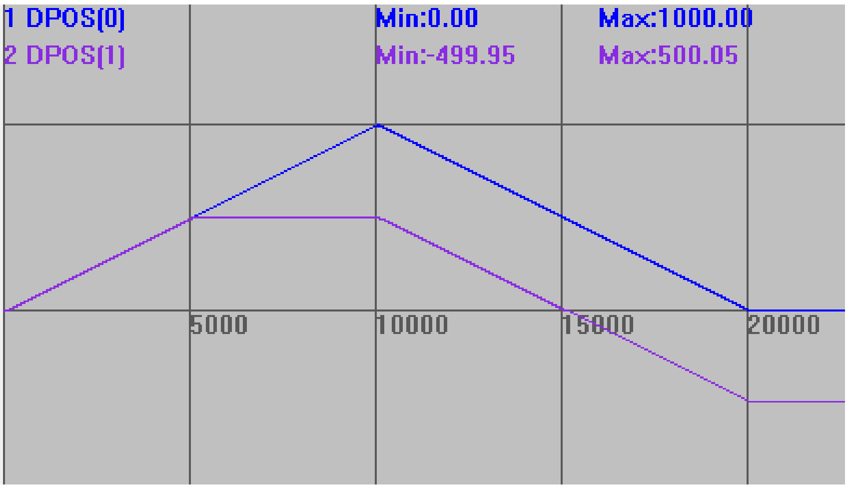
	main axis of axes group. When SYSTEM_ZSET is modified, AXIS_ZSET of present BASE axes will also be modified. In order to fit old procedure, usually it is not recommended to use SYSTEM_ZSET instruction. Parameters: bit 1: 1 - use precision output function of MOVE_OP, 0 - MOVE_OP original method. bit 4: 1 - when encoder axes are attached, use MOVE_OP precision mode based on encoder position, if multiple encoder axes interpolate, then use the mode of configuration of BASE motion main axis. bit 5: 1 – CANCEL (2) / RAPIDSTOP (2) emergency deceleration is DECEL, 0 – emergency deceleration is FASTDEC
Grammar	To read: VALUE=AXIS_ZSET To write: AXIS_ZSET=VALUE
Controller	Firmware version above 20170517
Example	**Example 1: Open Precision Output** BASE(0) ATYPE=1 DPOS=0 SPEED=100 ACCEL=1000 DECEL=1000 AXIS_ZSET(0)=2 'open precision output of MOVE_OP MOVE(100) MOVE_OP(0,1) 'precision takes effect, and select channel 0 **Example 2: Open Multi-Encoder Precision Output Port** Normally, there are 4 channels used for precision output in ZMC4XX series, but some have 8 channels. Suppose there are 3 dispensing positions on device, all need precision output. BASE(0,1,2) 'select axis 0 as main axis AIXS_ZSET(0)=19 'open MOVE_OP encoder precision output for main axis 0 …… BASE(3,4,5) AIXS_ZSET(3)=19 …… BASE(6,7,8) AIXS_ZSET(6)=19 …… **Example 3: Emergency Deceleration Selection** BASE(0) DPOS=0

	ATYPE=1 SPEED=100 ACCEL=10000 DECEL=10000 ‘set deceleration as 1000 FASTDEC = 100000 ‘set fast deceleration as 100000 AXIS_ZSET=32 'deceleration selection TRIGGER ‘trigger oscilloscope automatically MOVE(1000) ‘motion in process MOVE(-2000) ‘motion in buffer DELAY(500) CANCEL(2) ‘emergency stop Please see below, when AXIS_ZEST is not set, the deceleration should be 100000, and it is 10000 when set. 
Instruction	SYSTEM_ZSET, MOVE_OP

AXIS_MODE—connect Motion Holds

Type	Axis parameters]
Description	**Set BIT=1 to prevent CONNECT motion from exiting caused by position limit and soft limit.** BIT1 = 0, when meets position limit, connection “CONNECT” between master axis and slave axis is interrupted. Then, after position limit alarm is cleared, operate master axis now, which means slave axis doesn’t follow anymore. BIT1 = 1, when meets position limit, connection still exists, after position limit alarm is cleared, slave axis still follows. BIT5 = 0, default configuration, tracking MPOS preferentially when main axis is with encoder.

	BIT5 = 1, set cam or cam motion on the main axis, and assign compulsively the DPSO that is to track main axis. Involved instructions: CAMBOX, CONNECT, MOVELINK, MOVESLINK, MOVESYNC, HW_PSWITCH2. Valid in controllers with firmware version 20170616 and above.
Grammar	VAR1 = AXIS_MODE, AXIS_MODE = expression
Example	**Example 1: not set AXIS_MODE parameters** RAPIDSTOP(2) WAIT IDLE BASE(0,1) ATYPE=1,1 UNITS=100,100 DPOS=0,0 SPEED=100,100 ACCEL=1000,1000 DECEL=1000,1000 AXIS_MODE=0,0 'set parameters FS_LIMIT=1000,500 TRIGGER 'trigger oscilloscope automatically CONNECT(1,0) AXIS(1) 'axis 1 connects to axis 0, ratio is 1 MOVE(1000) AXIS(0) MOVE(-1000) AXIS(0) Motion Path: DPOS(0)=1000(vertical scale), no offset DPOS(1)=1000(vertical scale), no offset  Axis 1 accesses limit position, then stops, and disconnects with axis 0, which means they have no any relation on following motions. **Example 2: set AXIS_MODE parameters** RAPIDSTOP(2) WAIT IDLE BASE(0,1) ATYPE=1,1

	UNITS=100,100 DPOS=0,0 SPEED=100,100 ACCEL=1000,1000 DECEL=1000,1000 AXIS_MODE=0,2 FS_LIMIT=1000,500 TRIGGER 'trigger oscilloscope automatically CONNECT(1,0) AXIS(1) 'axis 1 connects to axis 0, ratio is 1 MOVE(1000) AXIS(0) MOVE(-1000) AXIS(0) Motion Path： DPOS(0)=1000(vertical scale), no offset DPOS(1)=1000(vertical scale), no offset  Axis 1 accesses position limit, then stop, but it keeps connection with axis 0, then it moves to position that is in the position limit, and follows axis 0.
Controller	General
Instruction	CONNECT, FS_LIMIT, RS_LIMIT

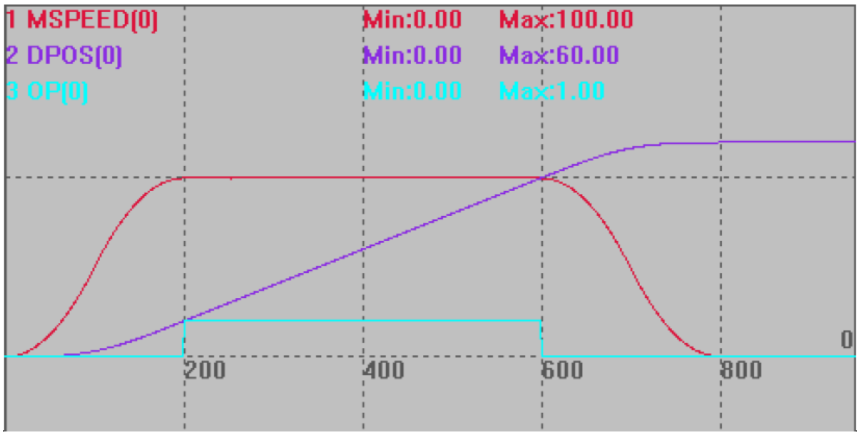
MOVEOP_DELAY-Output Delay in Buffer

Type	Axis parameters
Description	**Make buffer signal delay output for BASE axis.** When high precision output mode of MOVE_OP is used on axis appointed by BASE, it can adjust the actual trigger time of OP, ms (millisecond) as unit, the value can also be decimals format, maximum of the delay time is 100 ms. If value is set as minus, OP can be opened in advance, it is 2ms in advance for stepper, and 20ms in advance for servo. It can be used to stop the glue output in dispensing machine. The command will be affected by axis FE, if you only want to verify the

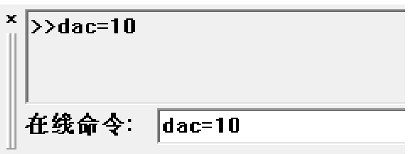
	command function (ignore this affection), you can set ATYPE as 0 or 1 to test.
Grammar	MOVEOP_DELAY= timems timems: delay time.(ms)
Controller	With firmware version above 20170505 supports.
Example	MOVEOP_DELAY = 2 ‘real output time delays 2ms
Instruction	MOVE_OP, HW_PSWITCH, HW_PSWITCH2

MOVEOP_ADIST—Close the glue in advance

Type	Axis parameters
Description	**MOVE_OP can configure that output in advance one certain distance, default 0 means it won’t be taken effect. Compare to default value, an assigned vector distance will be output in advance if it is positive value. If it is the minus, an assigned vector distance will be delayed.** This command is only valid when one single OP is operated by MOVE_OP, and for each axis, their actions have the sequence, namely, former one MOVE_OP operation (that is to change position) is not finished, behind will not act. It will clear automatically after MOVEOP_ADIST entering buffer to avoid affecting other MOVE_OP. Controllers that are with HW functions can open HW precision output through AXIS_ZSET, same as MOVE_OP precision, this command only supports MOVE interpolations or MOVESCAN motions, it doesn’t support cam, MOVE_PT, etc. And cross-small segments output in advance can be achieved through MOVE instruction.
Grammar	MOVEOP_ADIST= distance distance: how far
Controller	4xx series controllers with fast firmware version above 190201.
Example	BASE(0) ATYPE=1 UNITS=100 DPOS=0 MPOS=0 SPEED=100 ACCEL=1000 DECEL=1000 MERGE=1 SRAMP=100 TRIGGER MOVEOP_ADIST = -10 ′delay 10 units (distance), then glue ON MOVE_OP(0,1) MOVE(5) MOVE(50) ′include actual glue ON and glue OFF

	MOVE(5) MOVEOP_ADIST = 10 'make glue off in advance 10 units MOVE_OP(0,0) END Motion curve: MSPEED(0) 100 (vertical scale) DPOS(0) 50 (vertical scale) OP(0) 5 (vertical scale) 
Instruction	MOVE_OP, MOVEOP_DELAY, AXIS_ZEST

DAC--Analog Control of Field Bus Axes

Type	Axis Parameters
Description	**Servo axis DA control directly, speed or torque mode support.** Unit is DA module scale, 12 bits or 16 bits. When doing speed control, see exact unit of drive. When doing torque control, the unit is milli, 100% torque when equals 1000.
Grammar	VAR1 = DAC, DAC = expression
Controller	With EtherCAT port or Rtex port, firmware above 2017 support.
Example	**Example 1: Rtex speed control** Please use Rtex initialization procedure at first, and set ATYPE=51. Then do ZDevelop online instruction, as follow:  At this time, rotation speed of motor is 10r/min, send dac=-10, it will rotate inversely. Also it can send dac instruction in the procedure. Speed unit can refer to drive manual. As follow: Instruction speed

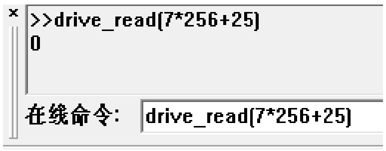
[size]: 32 bits with symbol

[unit]: set through Pr7.25(RTEX speed unit)

Pr7.25	unit
0	[r/min]
1	[instruction unit/s]

[set unit]: maximum speed level in negative direction ~ maximum speed level in positive direction.

When set unit as r/min, it will convert into instruction unit when doing internal calculation, the converted value should be from -800000001h to 7FFFFFFFh



At this time, the unit is 【r/min】

Example 2: Rtex torque control

Please set the first position of drive motor parameter pr6.47 as 0, close 2 DOF control mode, parameter pr3.17 set speed limit, as follow. (Panasonic Rtex manual for reference)

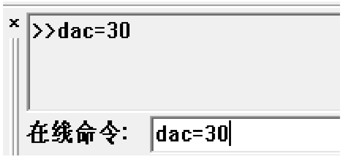
The set value of Pr3.17(speed limit selection) is 1, can switch to speed limit value under torque control through SL_SW.

<table>
<tr><td>Type</td><td>3</td><td>3</td><td>3</td></tr>
<tr><td>No.</td><td>17</td><td>21</td><td>22</td></tr>
<tr><td>Property</td><td>B</td><td>B</td><td>B</td></tr>
<tr><td>Parameter name</td><td>speed limit selection</td><td>Speed limit value is 1</td><td>Speed limit value is 2</td></tr>
<tr><td>Set range</td><td>0-1</td><td>0-20000</td><td>0-20000</td></tr>
<tr><td>Unit</td><td>/</td><td>r/min</td><td>r/min</td></tr>
<tr><td>Function</td><td><table><tr><td rowspan="2">value</td><td colspan="2">SL_SW</td></tr><tr><td>0</td><td>1</td></tr><tr><td>0</td><td colspan="2">Pr3.21</td></tr><tr><td>1</td><td>Pr3.21</td><td>Pr3.22</td></tr></table>Set speed limit value selection mode if turn to torque control.</td><td>Set speed limit when turn to torque control. In torque control, control is carried out within the speed set by the speed limit, and the internal values are limited in the smallest set speed of Pr5.13 (over speed level), Pr6.15 (the over second speed</td><td>WhenPr3.17(speedlimit selection)=1, set SL_SW speed limit as 1h, and internal value is limited in the smallest set speed of Pr5.13 (over speed level), Pr6.15 (the over second speed level0), Pr9.10(maximum over speed level).</td></tr>
</table>

		level),Pr9.10(maximum over speed level)	

Then use Rtex initialization procedure, and set ATYPE=52.

Next, can send dac online instruction in zdevelop, now, motor starts.



Now, the torque of drive motor is 0.03. If dac is too small, motor can not overcome friction force to operate.

Also can send dac in the procedure.

When in the torque control, unit is milli, dac=1000 means 100%.

[size]: 32 bits with symbol

[unit]: 0.1%

[set range]: maximum speed level in negative direction ~ maximum speed level in positive direction.

Maximum torque limit[%]=100×Pr9.07/(Pr9.06×2½)

Example 3: EtherCAT speed control

```
FOR I=0 TO 1                    'first use, set all axes as ordinate pulse type
ATYPE(I)=1
NEXT
SLOT_SCAN(0)                    'bus scan start

IF NODE_COUNT(0,0)>0 THEN
    AXIS_ADDRESS(0)=1     'first drive motor is mapped to axis 0
    ATYPE(0)=66                 '66 as speed control mode
    DRIVE_PROFILE(0)=20  'set speed control as 20
    DELAY (200)
    SLOT_START(0)               'scan bus successfully, start bus

    DRIVE_CONTROLWORD(0)=128     'clear errors of drive motor out
    DELAY (2)
    DRIVE_CONTROLWORD(0)=6
    DELAY (2)
    DRIVE_CONTROLWORD(0)=15
    DELAY (2)
    DELAY(20)
    DATUM(0)                    'clear controller errors out
    BASE(0)
    AXIS_ENABLE=1               'mapped axis enable open
    WDOG=1                      'axis enable
    DAC=10000                   'motor rotates at speed of 10000/s
ENDIF
```

	Example 4: EtherCAT torque control Just make some modification of example 3. ATYPE=67, DRIVE_PROFILE=30. Now, send range of dac is 0~1000, 1000 means 100% torque, if needs inverse select, just send minus value.
Instruction	SERVO

ERRORMASK--Operation when Error

Type	Axis Parameters
Description	**To decide which errors are closed through AND operation between ERRORMASK and AXISSTATUS.**
Grammar	VAR1 = ERRORMASK, ERRORMASK = expression
Controller	General
Example	BASE(0) WDOG=1 'open enable ?AXISSTATUS 'print 16, positive hardware position limit alarm ERRORMASK=16 DELAY(10) 'delay for operation ?WDOG 'print 0, enable is off.
Instruction	AXISSTATUS, WDOG

ZSCAN_CORRECT—Galvanometer Correction

Type	Axis parameters
Description	**Correct galvanometer axis parameters.**
Grammar	ZSCAN_CORRECT(ixy,imode,imaxline,imaxrow,x1,y1,x2,y2,tableindex) ixy: value as 0/1, select two galvanometers. 0-the first galvanometer 1-the second galvanometer. imode: 0-close correction, 1-use partition correction, table input mearused actual postion, 2-use partition correction, table input pulse position needed to achieve, 210701 add this function. imaxline: line, Y direction, the more data is, the higher precision is. imaxroe: row, X direction. x1,y1: bottom right corner position in theory. x2,y2: above right corner position in theory. tableindex: measured actual corrdinate start to save in thr table index, first X then Y, the first line(save in the row sequence), the next line. Attention: XY is the actual physical axis, the first is X, the second is Y, no relation with mapped virtual axis number. Coordinate written is actual pulse position of galvanometer.

	(support decimals)(XY2 protocal coordinate is from 32768 to 32767)
Controller	Valid in controllers with galvanometer axis
Example	TABLE(0, -40.6,-41.2) TABLE(2, 0,-41) TABLE(4, 41,-42) TABLE(6, -41,0) TABLE(8, 0,0) TABLE(10, 41.2,0) TABLE(12, -40.4,41.2) TABLE(14, 0,41.2) TABLE(16, 41,42.4) FOR i=0 TO 17 TABLE(i) = TABLE(i)*500 'all are pulse coordinate NEXT **ZSCAN_CORRECT**(0,1,3,3,-20000,-20000,20000,20000,0)

11.16 Reserved Instructions

D_GAIN--Differential Gain

Type	Axis Parameters
Description	**Differential gain, which is only valid in analog servo.** D_GAIN includes differential gain of axis. Differential output is in direct proportion to change number of follow error, default is 0. It will produce smoothing response if superpose differential gain for system, which means bigger proportion gain can be permitted. Vibration will be caused if too high. Attention: servo gain should be changed when SERVO = OFF to avoid unsteadiness.
Grammar	VAR1=D_GAIN, VAR1=D_GAIN Valid: set assigned axis through D_GAIN AXIS (axis) method
Example	Refer to SERVO
Instruction	SERVO, P_GAIN, D_GAIN, OV_GAIN, VFF_GAIN, FE_LIMIT, FE_RANGE

I_GAIN--Integral Gain

Type	Axis Parameters
Description	**Integral gain, which only valid in analog servo.** Integral outputs through calculating sum total of follow error. Default: 0.

	Position errors when running or in still can be decreased through superposing integral gain into servo system. And overshoot and vibration can be decreased. Therefore, it is applied in constant speed and low-speed process. Attention: servo gain should be changed when SERVO = OFF to avoid unsteadiness.
Grammar	VAR1=I_GAIN, I_GAIN = expression Valid: set assigned axis through I_GAIN AXIS (axis) method
Example	Refer to SERVO
Instruction	SERVO, P_GAIN, D_GAIN, OV_GAIN, VFF_GAIN, FE_LIMIT, FE_RANGE

OV_GAIN--Speed Gain

Type	Axis Parameters
Description	**Speed gain, which is only valid in analog servo.** Speed outputs through multiple changes of MPOS and parameter value of OV_GAIN. Default: 0. In system, add output speed gain and damping equivalence of machine, output will be smooth and proportion gain will be promoted. However, it will cause big follow errors. And vibration, big follow errors will be produced if there is too high output gain. Attention: servo gain should be changed when SERVO = OFF to avoid unsteadiness.
Grammar	VAR1=OV_GAIN, OV_GAIN = expression Valid: set assigned axis through OV_GAIN AXIS (axis) method
Example	Refer to SERVO
Instruction	SERVO, P_GAIN, D_GAIN, OV_GAIN, VFF_GAIN, FE_LIMIT, FE_RANGE

P_GAIN--Proportion Gain

Type	Axis Parameters
Description	**Proportion gain, which is only valid in analog servo.** Proportion outputs through multiple follow errors and P_GAIN. Default: 0 Proportion gain sets the rigidity of servo responses, vibration will be caused if value is too high, but big follow errors will be produced if value is too low. Attention: servo gain should be changed when SERVO = OFF to avoid unsteadiness.
Grammar	VAR1=P_GAIN, P_GAIN = expression Valid: set assigned axis through P_GAIN AXIS (axis) method
Example	Refer to SERVO
Instruction	SERVO, D_GAIN, OV_GAIN, VFF_GAIN, FE_LIMIT, FE_RANGE,

	I_GAIN

VFF_GAIN--Feedforward Gain

Type	Axis Parameters
Description	**Feedforward gain feedbacked by speed, don't support non-bus servo.** Speed feedforward is the multiple value of changes of DPOS and parameter value of VFF_GAIN. Default: 0. 60B1 pdo = axis pulse speed * VFF_GAIN How to calculate closed-loop axis PID: axis pulse speed * VFF_GAIN (as speed feedforward) System follow errors in motion can be decreased and output proportion of speed can be increased through superposing speed feedforward gain. **Note: servo gain should be changed when SERVO = OFF to avoid unsteadiness.**
Grammar	VAR1=VFF_GAIN, VFF_GAIN = expression Valid: set assigned axis through VFF_GAIN AXIS (axis) method
Example	Refer to SERVO
Instruction	SERVO, D_GAIN, OV_GAIN, FE_LIMIT, FE_RANGE, I_GAIN, P_GAIN

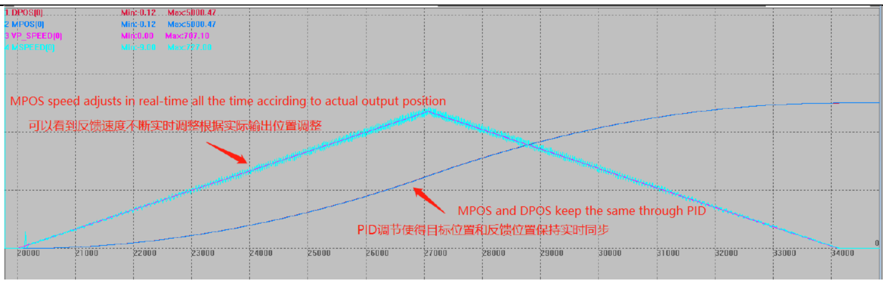
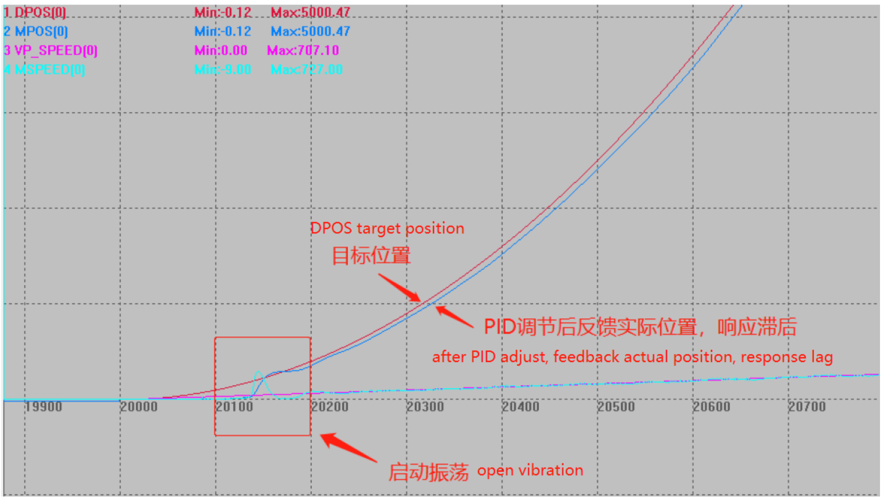
AFF_GAIN -- Acceleration Feedforward Gain

Type	Axis Parameters
Description	**Feedforward gain feedbacked by acceleration, don't support non-bus servo.** 60B2 pdo = (pulse speed change of axis in each period) * AFF_GAIN
Grammar	VAR1=AFF_GAIN, AFF_GAIN = expression Valid: use AFF_GAIN AXIS(axis) to set assigned axis.
Example	Refer to SERVO
Instruction	SERVO, D_GAIN, OV_GAIN, FE_LIMIT, FE_RANGE, I_GAIN, P_GAIN

SERVO—Closed-Loop Switch

Type	Axis Parameters
Description	Close loop switch setting Value range: ON/OFF, default is OFF. It is used as closed-loop control system together with P_GAIN, D_GAIN, I_GIAN, OV_GAIN, VFF_GAIN commands, attention analog servo needs to be used. **Attention:** 1. ATYPE of ECAT bus suits to mode 66 and mode 67. 2. PID can be adjusted to the best according to actual situation only by

	manual, it is recommended to bring load. 3. PID adjust reference
Grammar	VAR1 = SERVO, SERVO = ON/OFF
Controller	General
Example	RAPIDSTOP(2) WAITIDLE TRIGGER 'trigger oscilloscope BASE(0) UNITS=1000 ACCEL=100 DECEL=100 SPEED=1000 CREEP=100 LSPEED=0 MERGE=0 SRAMP=0 DPOS=0 MPOS=0 FE_LIMIT=10 'max follow error limit range can be modified, it will alarm when timeout FE_RANGE=10 'follow error value when set alarm P_GAIN AXIS(0)=100 'proportion gain, and it can use assigned axis method D_GAIN=5 'integration gain I_GAIN=1 'differential gain OV_GAIN=0 'speed gain VFF_GAIN=0 'feedforward gain SERVO=ON 'open closed-loop control BASE(0) MOVE(5000) WAITIDLE SERVO AXIS(0)=OFF 'close closed-loop control, and can through assigned axis, and it is recommended to close after each time usage DELAY(3000) 'stop detection after 3 seconds DPOS(0) MPOS(0) END Global curve is below, it can be seen DPOS and MPOS are adjusted to be real-time synchronous through PID under closed-loop control.

	 Local curve: 
Instruction	

TRANS_DPOS

Type	Axis Status
Description	reserved

Chapter XII Instructions Related to Input and Output

12.1 Instructions Related to Input

IN--Inputs

Type	Input and output functions
Description	**Read inputs, return status of in0-31 if there is no parameter.** Read value is the status reversed by INVERT_IN. IO channel number is related to dial-up switch configuration on expansion module, start value is (16+dial-up value*16), EIO bus expansion IO uses NODE_IO instruction, the value can only be a multiple of 8. See hardware manual for reference. Attention: IO mapped number should be over IO max NO. of controller itself, and can not superpose with controller number.
Grammar	IN([channel1],[channel2]) channel1 start input channel to read. channel2 end input channel to read, return signal input status if no this parameter.
Controller	General
Example	a=**IN**(1) 'read status of input 1
Instructions	OP, INVERT_IN

AIN--Analog Input

Type	Input and output functions
Description	**Read analog input, return scale value of AD conversion module.** 12-bit scale range: 0~4095, mapped voltage: 0-10v. 16-bit scale range: 0~65536, mapped voltage: 0-10v. ZAIO channel number is related to dial-up switch configuration on expansion module, start value is (8+dial-up value*8), ZMIO bus IO expansion AD uses NODE_IO instruction, the value can only be a multiple of 8. See hardware manual for reference. Attention: AIO mapped number should be over AIO max NO. of controller itself, and can not superpose with controller number.
Grammar	Var=AIN(channel) channel analog input channels:0-127

Controller	General
Example	a=**AIN**(1) 'read AD value of channel 1. a=**AIN**(1) *10 /4096 'voltage value of channel 1.
Instructions	AOUT

ZSIMU_IN--Inputs Simulation

Type	Simulator specialized instructions.
Description	**Simulate input of IN.**
Grammar	ZSIMU_IN[([ionum ,] value)] ionum input NO., start from 0, return status of in0-31 if no this parameter value output status
Controller	General
Example	**ZSIMU_IN**(0,1) 'input 0 is ON.
Instructions	IN

ZSIMU_AIN--Analog Inputs Simulation

Type	Simulator specialized instructions
Description	**Simulate analog input of IN.**
Grammar	ZSIMU_AIN(ionum, value)
Controller	General
Example	**ZSIMU_AIN**(0,1024) 'analog input 0
Instructions	AIN

ZSIMU_ENCODER--Encoder Inputs Simulation

Type	Simulator specialized instructions
Description	**Simulate input of encoder.**
Grammar	ZSIMU_IN(axis num, value) axis num axis NO., start from 0 value ENCODER simulation value
Controller	General
Example	**ZSIMU_ENCODER**(0,1024) 'ENCODER=1024
Instructions	ENCODER

INVERT_IN--Reverse Inputs

Type	Special instructions
Description	**Reverse inputs status, it can be checked if inputs were reversed.**

Grammar	INVERT_IN(channel, state); VAR1= INVERT_IN(channel) channel: inputs channels state: ON/OFF
Controller	General
Example	**INVERT_IN**(1,ON) 'it is valid when OFF in terms of special signal, reverse input to avoid input is not valid when limit signal comes.(except ECI series.) FWD_IN(0)=1 'IN1 as positive position limit signal of axis 0.
Instructions	IN

IN_SCAN--Scan Inputs Change Status

Type	Input and output functions
Description	**Scan inputs change status, if returned value is 1(TURN), change happened; if returned value is 0(FALSE), change did not happen.** This function must be used to scan the inputs cycle-by-cycle, returned value is change status between two cycles. Status details can be checked through IN_EVENT, read value is status reversed by INVERT_IN. Only controller with firmware version above 20140214 is valid, scan range has width limit. ZMC00X series only is valid in single task.
Grammar	VAR1=IN_SCAN([channel1][,channel2]) channel1: start channel to be read. channel2: end input channel to be read, scan signal input status if no this parameter.
Controller	General
Example	WHILE 1 IF **IN_SCAN**(0,23) THEN 'scan electric level change of IN0-23. IF IN_EVENT (0) > 0 THEN 'triggered meet rising edge of IN0 PRINT "IN0 UP", IN_BUFF(0) ELSELF IN_EVENT(0) < 0 THEN 'trigger falling edge of IN0 PRINT "IN0 DOWN", IN_BUFF(0) ENDIF ENDIF WEND
Instructions	IN_EVENT, SCAN_EVENT, IN_BUFF

IN_EVENT--Read Input Change

Type	Input and output functions
Description	**Read inputs change details.**

	1-rising edge. -1-falling edge, 0-no change This function should be used with IN_SCAN together.
Grammar	VAR1 = IN_EVENT(IONUM)
Controller	General
Example	See IN_SCAN
Instructions	IN_SCAN, SCAN_EVENT

SCAN_EVENT--Check Change

Type	Input and output functions
Description	**Check change status of expressions.** 1:off- on, -1:on-off, 0:no change. Don’t call the same SCAN_EVENT of SUB in the cycle or the multi-task. Valid in controller with firmware version above 150810, or use IN_EVENT and IN_SCAN instead.
Grammar	ret = SCAN_EVENT (expression) expression any valid expression, result will become BOOL Type.
Controller	General

Example	**Example One: Scan inputs signals**
	WHILE 1
	IF SCAN_EVENT(IN(0))>0 THEN 'trigger rising edge of IN0
	PRINT “IN0 ON”
	ELSELF SCAN_EVENT(IN(0))<0 THEN 'trigger falling edge of IN0
	PRINT “IN0 OFF”
	ENDIF
	WEND
	
	Example Two: Scan register, variables
	WHILE 1
	IF SCAN_EVENT(TABLE(0))>0 THEN
	'trigger rising edge of TABLE0
	PRINT “TABLE0 ON”
	ELSELF SCAN_EVENT(TABLE (0))<0 THEN
	'trigger falling edge of TABLE0
	PRINT “TABLE0 OFF”
	ENDIF
	WEND
	Operate table(0) online, and print results.

Instructions	IN_SCAN, IN_EVENT

IN_BUFF--Read Inputs Buffer

Type	Input and output functions

Description	**Read present inputs scanned by IN_SCAN, return status of in0-31 if no parameters.** Read value is status reversed by INVERT_IN.
Grammar	IN_BUFF([channel1],[channel2]) channel1: start channel to be read, which must be inputs range of IN_SCAN. channel2: last channel to be read, return single input status if no last channel input
Controller	General
Example	See IN_SCAN
Instructions	IN_SCAN

INFILTER—Input Filter

Type	System Parameter
Description	**Local input filter parameter.** The bigger value is, the longer filtering time will last, value is:2-9, default is 2.
Grammar	VAR1 = INFILTER, INFILTER= expression
Controller	General
Example	INFILTER= 5 'increase the filtering time when there is terrible interruption.

IN_SMFILTER – Set IN Filter

Type	Special Command
Description	**Set the filter for one single input.**
Grammar	IN_MSFILTER (channel, timems), VAR1=IN_MSFILTER (channel) channel: input channel timems: the filtering time, the unit is ms, and the precision only can reach system period, up to >200 periods, the default value is 0.
Controller	Valid in 5xx series and the firmware version is above 20230808.
Example	IN_MSFILTER (0,5) 'set IN0 as the filter, the filtering time is 5ms

12.2 Instructions Related to Output

OP--Outputs

Type	Input and output instructions and functions
Description	**Out or read outputs status**

	When it is used in expression, it is regarded as function grammar automatically. IO channel number of ZIO expansion board is related to dial-up code switch configuration, start value is (16+dial-up value*16), EIO bus expansion IO uses NODE_IO instruction, the value can only be a multiple of 8. see hardware manual for reference. Attention: IO mapped number should be over IO max NO. of controller itself, and can not superpose with controller number. Maximum operation output port number is 32.
Grammar	OP([ionum],value) or OP(ionum1, ionum2,value[,mask]) OP([firstnum[,[finalnum]) ionum: output number, starts from 0 value: output status, define multi-port status as bit when operating multiple outputs. ionum1: the first channel to be operated ionum2: the last channel to be operated mask: it is used to assign IOs to be operated, the first and the last c channels both are operated when it is not filled. firstnum: output number, starts from 0. finalnum: output number, starts from 0, it reads single output status if this parameter is not filled.
Controller	General

Example	**Example 1: single operation**
	'reverse output 0
	IF **OP** (0) = ON THEN
	OP (0,OFF)
	ELSE
	OP (0,ON)
	ENDIF
	
	Example 2: regional operation
	OP(0,7,$FF) 'bit0-bit7 full open
	DELAY(1000)
	OP(0,7,0)
	
	OP(8,15,$FF) 'bit8-bit15 full open
	DELAY(1000)
	OP(8,15,0)
	
	OP(0,15,$FFFF) 'bit0-bit15 full open
	DELAY(1000)
	OP(0,15,0)
	
	OP(0,31,$FFFFFFFF) 'bit0-bit31 full open
	DELAY(1000)

	OP(0,31,0)
Instructions	READ_OP,MOVE_OP

AOUT--Analog Output

Type	Input and output instructions and functions
Description	**Analog channel output:** 12-bit scale range: 0~4095, mapped voltage: 0-10v. 16-bit scale range: 0~65536, mapped voltage: 0-10v. AOUT(2) relates to parallel port 0~255, which is used to set the power of laser, such as, valid in ZMC408SCAN and 504SCAN.
Grammar	AOUT(channel) = value channel analog output channels:0-63 DA channel number is related to dial-up switch configuration on expansion module, start value is (4+dial-up value*4), see hardware manual for reference.
Controller	General
Example	**AOUT**(1) = 0 'close output DA channel 1. **AOUT**(1) = 4095 'DA1 output voltage is 10V.
Instructions	AIN

READ_OP--Read Outputs

Type	Input and output functions
Description	**Read outputs status.** Same as OP, output as per bits in terms of multi-output operation.
Grammar	READ_OP ([firstnum[,[finalnum]) firstnum first output number, starts from 0. finalnum last output number, starts from 0, it reads single output status if no this parameter
Controller	General

Example	'reverse output 0
	IF **READ_OP** (0) = ON THEN
	OP (0, OFF)
	ELSE
	OP (0, ON)
	ENDIF

Instructions	OP

EXIO_DIR – Configure EXIO Interface

Type	Input and output functions
Description	**Assign inputs and outputs of EXIO expansion interface as per bit, and it needs to be used together with customized adapter board.** Use Fiber adapter board IO configuration instruction “EXIO_DIR(0, $8FFFF), YAG adapter board IO configuration instruction “EXIO_DIR(0, $FCBFE) and SPI adapter board IO configuration instruction “EXIO_DIR(0, $FFFFA).
Grammar	Command grammar: Exio_Dir(isel, idirbit) Function grammar: Exio_Dir(isel) isel: Exio selection, fix 0 currently idirbit: assign inputs and outputs as per bit, 1–output, 0–input (default)
Controller	ZMC408SCAN
Example	EXIO_DIR(0, $8FFFF) 'Fiber adapter board
Instructions	OP

12.3 Position Comparison Output Instructions

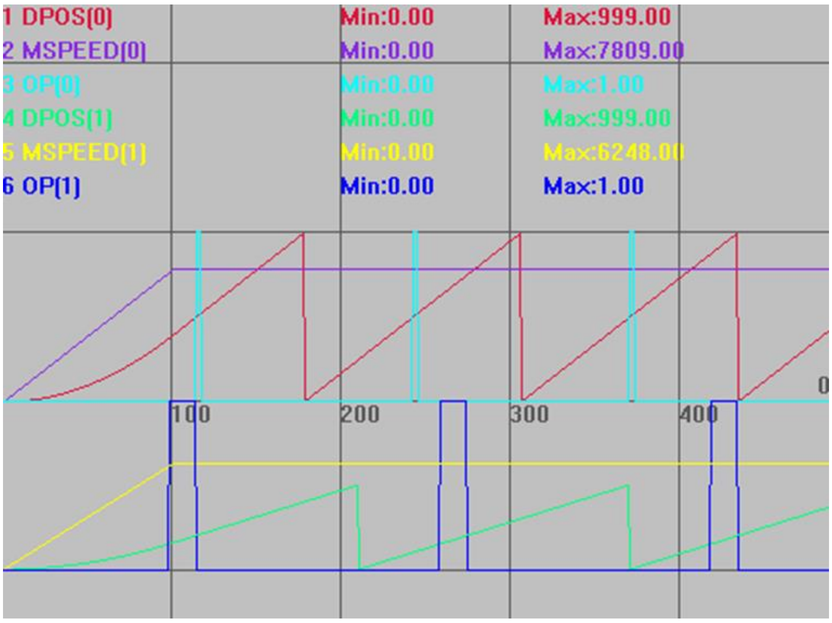
PSWITCH--Position Comparison by Software

Type	Inputs and Outputs Instructions
Description	**Operate outputs based on result of position comparison.** If more than one PSWITCH are mapped to the same output, then relevant comparers should be arranged in order. For pulse type motor, when ATYPE=4, it is the MPOS. Default ATYPE=1/7, it’s DPOS.
Grammar	PSWITCH(num,enable,[,axis,op num,op state,set pos,reset pos]) num: comparer NO., ZM1XX has 16 comparers, NO.:0-15. enable: enable comparers, ON-Start, OFF-Cancel. axis: axis NO. which position is required. op num: IOs to be operated. op state: output status, 1-output is ON in followed position range, 0 output is OFF in followed position range. set pos: set start position that output activates. Unit is units. reset pos: set position that output reset. Unit is units. Different controllers support different comparison numbers, use ?*max to print and check max_pswitch parameters to determine the number.
Controller	General

Example

```
RAPIDSTOP(2)
WAIT IDLE
DELAY(1000)
ERRSWITCH = 3
BASE(0,1)                   'select axis NO.
ATYPE=1,1                   'pulse type stepper or servo
DPOS = 0,0
UNITS = 1,1                 'pulse amount
SPEED = 10000,10000
ACCEL=SPEED(0)*10,SPEED(1)*10
DECEL=SPEED(0)*10,SPEED(1)*10
REP_OPTION=1,1              'set coordinate cycle range: 0 ~ +REP_DIST
REP_DIST=1000,1000
TRIGGER
MOVE(10000,8000)
PSWITCH(0,ON,0,0,ON,500,520)
PSWITCH(1,ON,1,1,ON,300,400)
END
```

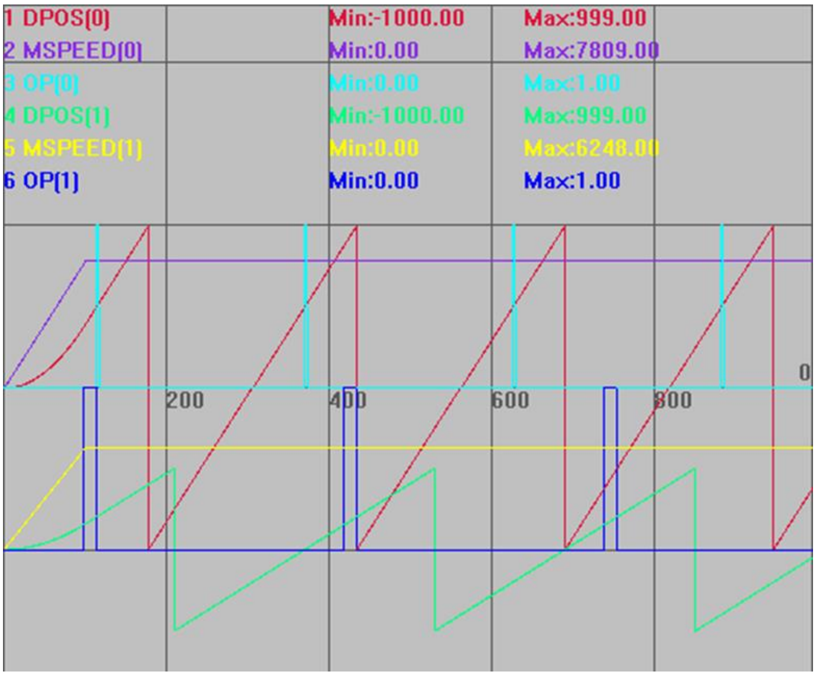
DPOS(0) vertical scale 1000, no offset
MSPEED(0) vertical scale 10000, no offset
OP(0) vertical scale 1, no offset
DPOS(0) vertical scale 2000, offset -2000
MSPEED(0) vertical scale 10000, offset -10000
OP(0) vertical scale 1, offset -1



As former example, just modify some instruction as follow:

```
REP_OPTION=0,0
                'set coordinate cycle range: -REP_DIST~+REP_DIST
```

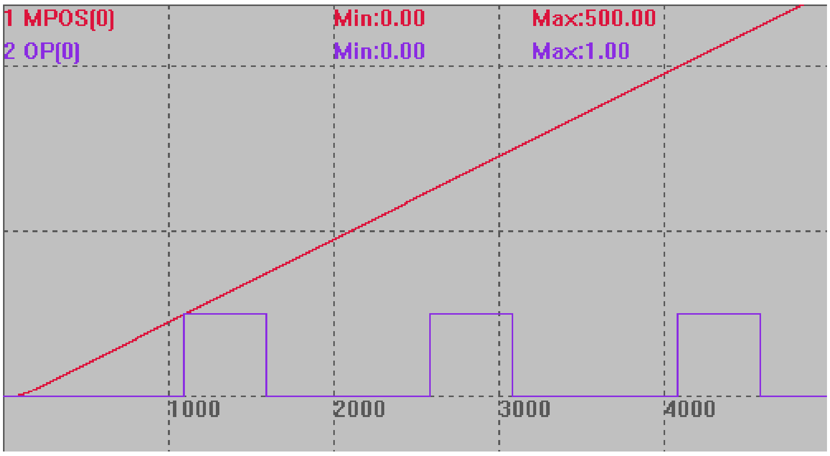
DPOS(0) vertical scale 1000, no offset

	MSPEED(0) vertical scale 10000, no offset OP(0) vertical scale 1, no offset DPOS(0) vertical scale 2000, offset -2000 MSPEED(0) vertical scale 10000, offset -10000 OP(0) vertical scale 5, offset -1 
Instructions	HW_PSWITCH

HW_PSWITCH—Hardware Position Comparison Output

Type	Axis Instructions
Description	**Position Comparison Output by hardware, different axes are mapped to different outputs.** Default mapping relationship: axis 0-5 are mapped to output 0,1,2,3,0,1. There are totally 4 hardware comparison outputs. Two HW_PSWITCH can be called continuously, and the number of called instructions can be gained by related functions. Each compare point is triggered, present output electrical level will be reversed. HW busffers are 1024, totally 1024 HW instrutions can be called continuously. After HW instrution is called, it won't be affected by followed coordinate change caused by related functions, coordinate saved in TABLE should be correct, it is better to modify coordinate by manual, and try to avoid conflict between HW instrution and change caused by auto coordinate cycle(REP_OPTION) . Since coordinate is not determined by procedure in auto coordinate cycle

	mode, not able to confirm if HW is before or after the coordinate, so coordinate in TABLE can also not be confirmed. This instruction only supports pulse axis hardware position comparison output, use HW_PSWITCH2 in fieldbus axis. For pulse type motor, when ATYPE=4, it is the MPOS. Default ATYPE=1/7, it's DPOS.
Grammar	HW_PSWITCH(mode, direction, reserve, tablestart, tableend) Buff=HW_PSWITCH([axisnum]) mode: 1-start comparer, 2-stop and delete comparer that's not finished direction: 0-negative direction of coordinate, 1-positive direction of coordinate, 2-no direction. reserve: reserved tablestart: TABLE NO. that saves first comparison coordinate. tableend : TABLE NO. that saves last comparison coordinate. If not compare all points, mode must be set as 2, stop and delete those compare points through HW_PSWITCH(2) instruction, or will cause abnormal work of output channel.
Controller	ZMC4XX series or above, with firmware version above 20170704. ZMC420SCAN doesn't support HW_PSWTICH.
Example	Testing environment as follow: ZMC432, firmware: 20170709 (the simulator can not run this instruction) BASE(0) 'select axis 0 to output OP(0) by default ATYPE=4 'encoder position as comparison output reference, if no encoder, use pulse type UNITS=100 SPEED=100 ACCEL=500 MPOS=0 OP(0,OFF) TABLE(0,100,150,250,300,400,450) 'MPOS100-150: OP0 opens, MPOS150-250: OP0 closes, MPOS250-300: OP0 opens, MPOS300-400: OP0 closes, MPOS400-450: OP0 opens, MPOS after 450: OP0 closes. **HW_PSWITCH**(2) 'stop and cancel those comparison points not be compared completely. **HW_PSWITCH**(1, 1, 0, 0, 5) 'start comparison output TRIGGER MOVEABS(500) Comparison Output Curve: MPOS(0)=200(vertical scale) OP(0)=2(vertical scale)

	
Instructions	PSWITCH, HW_PSWITCH2

HW_TIMER--Hardware Timing

Type	Special Instructions
Description	**Hardware timer is used to restore electric level after hardware comparison output for a certain time.** **Recommendation: don't exceed 100ms.** Note: There is only 1 HW_TIMER for the controller that is without independent HW. And each calling will stop former calls compulsively. For HW independent controller, each HWOP has one HW_TIMER. The function HW_TIMER of ZMC420SCAN's every output port is independent. When HW_TIMER is not used, please use mode 0 to stop, otherwise, it is ON continuously, then behind comparison output may be affected. Use mode 2 firstly to open hardware timer, then other modes can be used to modify. Galvanometer controllers with firmware version above 20170709 support this function. OP and MOVE_OP can close HW_TIMER pulse that is operating, which means HW_TIMER can be used as PWM, OP outputs, pulse output will be ON, then next OP outputs, pulse output will be OFF. When use MOVE_OP precision output, infinite pulse of precision PWM output can be realized. Use ?*HW_TIMER to see the number of remaining pulses.
Grammar	HW_TIMER(mode, cyclonetime, optime, reptimes, opstate, opnum) done = HW_TIMER_DONE mode 0-stop hardware timer, 1-modify parameters dynamically (start the setting when no modification), 2-start (it can't start repeatedly), 3-stop hardware timer (similar to 0, but

this is for galvanometer controller), when current pulse completed and no output, behind OP still trigger sending pulses.

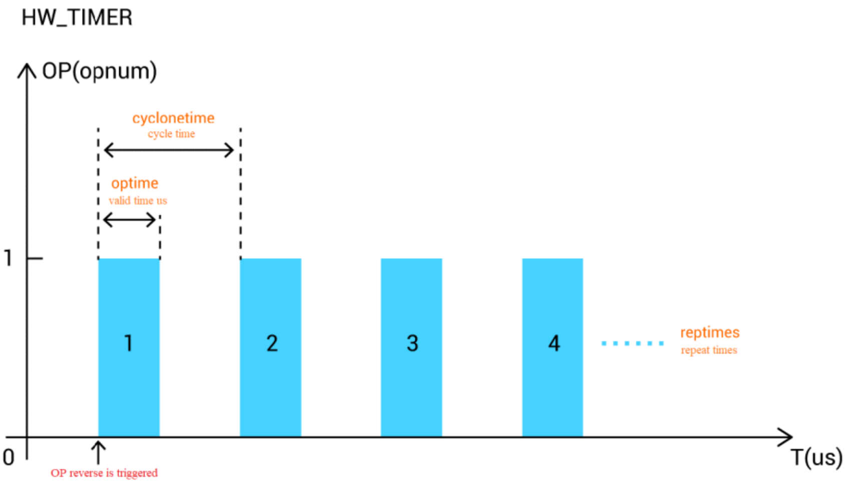
cyclonetime cycle time, us is the unit

optime valid time, us is the unit

reptimes repeat times, start mode, when reptimes=0, HW_TIMER will be softly closed, and continue to output remained undone pulse. When it is -1, output infinitely, except close it manually.

opstate output default status, start to do timing when output port became non default status (initial status of output is OFF, generally the parameter is set as OFF, so it starts to timing when it is ON)

opnum outputs NO., the port must support hardware comparison output.



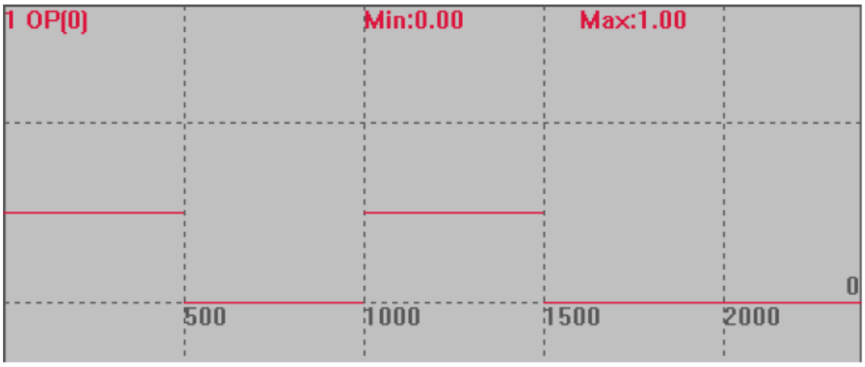
Galvanometer controllers with firmware version above 20170710 add **mode 3**, like 0, it will not output after current pulse finished, but following OP still trigger pulse. For mode 0, mode 0 stops, OP also can't trigger, it must turn on again.

Galvanometer controllers with firmware version above 20170710 add **mode 1**. Mode 1 supports modifying timer parameters dynamically after again sending, but mode 1 can't be used to turn on hardware timer, and it takes effects when comparison in mode 2 is not completed. Mode 2 can't be turned on repeatedly.

ZMC420SCAN 221017 adds **mode 4**, namely, pulse duty changing function, it is used together with mode 1 / 2.

HW_TIMER(mode4, [trigremaintime,][changetimes][changeonetick], reverse, opnum)

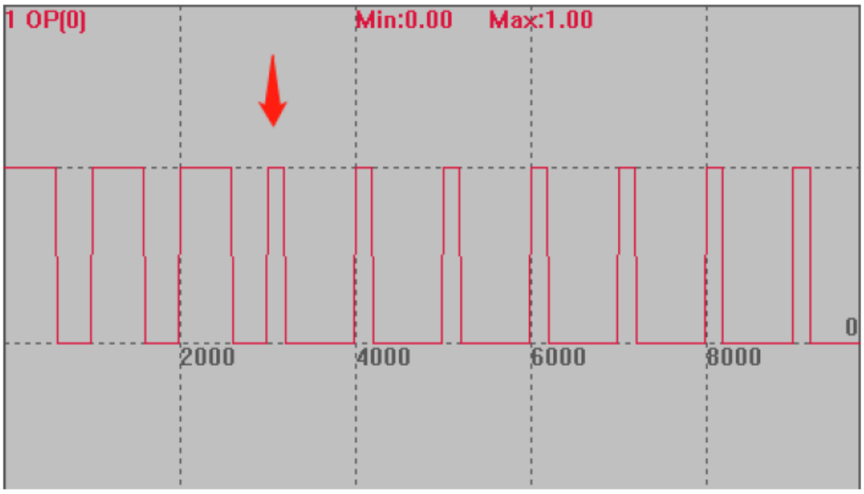
trigremaintine: start to adjust the width of pulse when remaining one certain number of pulses, attention the first pulse doesn't change, 0 means no to use

	changetiems: the number of changed pulses changeonetick: the width change of each pulse, the unit is us, it can be negative. reverse: reverse, set as 0 opnum: output number, the port must support hardware comparison output Note: please set suitable pulse width when mode 4 is applied, if the pulse width that is accumulated exceeds the period, it can't output normally. When mode 4 is not used, set the parameter 2-4 as 0, otherwise, it will take effect next time.
Controller	Valid in some ZMC3XX, ZMC4XX series and above controllers with firmware version 20170704 and above.
Example	Testing: ZMC420SCAN. firmware: 221017(simulator can't run this instruction) For show waveform intuitively, set a bigger period. **Example 1: mode 2, output pulse in cycle** RAPIDSTOP(2) WAIT IDLE(0) BASE(0) ATYPE=1 UNITS=100 SPEED=100 ACCEL=500 DPOS=0 HW_TIMER(0, 1000000, 500000, 2, OFF, 0) 'mode 0 stops hardware timer TRIGGER OP(0, OFF) HW_TIMER(2, 1000000, 500000, 2, OFF, 0) 'when 0 turn to ON, hardware timer trigger START to do timing, arrive 500ms, then turn to OFF. OP(0, ON) Running effect: set 1000ms as period, output two periods, before 500ms of every period start, then later half period close.  **Example 2: mode 1, output pulses in cycle. When mode 2 is turned on,**

use mode 1 to modify dynamically.

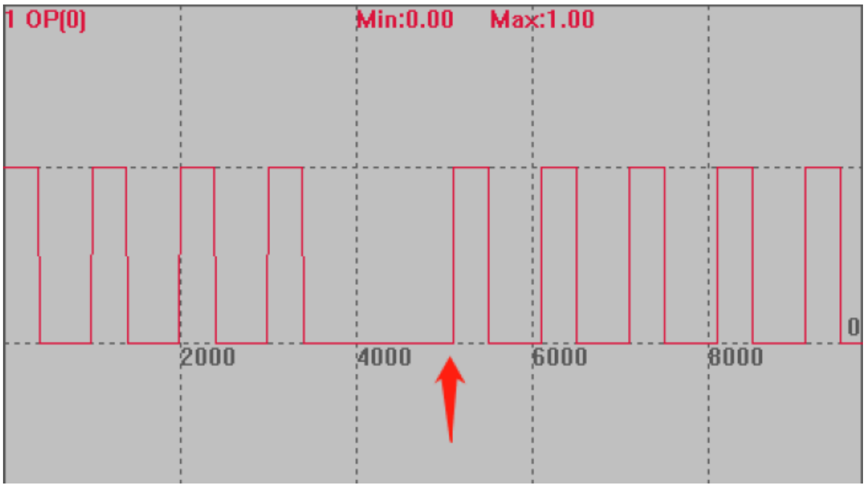
```
BASE(0)
ATYPE=1
UNITS=100
SPEED=100
ACCEL=500
DPOS=0
OP(0, OFF)
TRIGGER
HW_TIMER(2, 1000000, 600000, 10, OFF, 0)
                    'when output 0 turn to ON, hardware timer trigger
                    START to do timing, arrive 500ms, then turn to OFF.
OP(0, ON)
DELAY(2500)          'mode 1 takes effect after 2.5s, modify timing time,
                     the time can't be too long, if the comparison in
                     mode 2 completed, mode 1 can't take effect
HW_TIMER(1, 1000000, 200000, 5, OFF, 0)   'modify dynamically
```

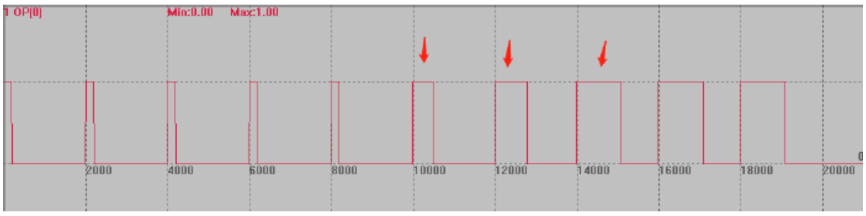
Mode 1 can modify the former timing time (the former can be set as mode 1 or mode 2), if this is changed into mode 2, there is no action for OP when mode 2 is scanned next time.



Example 3: mode 3, following OPs also can trigger when output stops

```
BASE(0)
ATYPE=1
UNITS=100
SPEED=100
ACCEL=500
DPOS=0
OP(0, OFF)
TRIGGER
HW_TIMER(2, 1000000, 400000, 10, OFF, 0)
                    'when output 0 turn to ON, hardware timer trigger
```

	START to do timing, arrive 400ms, then turn to OFF. OP(0, ON) DELAY(3000) 'close output after delay 3s HW_TIMER(3, 1000000, 400000, 10, OFF, 0) OP turns on, mode 2 outputs pulses normally, mode 3 take effects after 3s. stop output, turn on OP trigger HW again, then it can continue to output 10 times (the arrow is the moment when the OP triggers again) If mode 0 is used this time, open OP after stop, HW can't be triggered, it needs to rescan HW command.  **Example 4:** BASE(0) ATYPE=1 UNITS=100 SPEED=100 ACCEL=500 DPOS=0 OP(0, OFF) TRIGGER HW_TIMER(2, 2000000, 200000, 10, OFF, 0) 'when output 0 turn to ON, hardware timer trigger START to do timing, set period as 2s, arrive 200ms, then turn to OFF. OP(0, ON) HW_TIMER(4, 6, 3, 300000, OFF, 0) 'mode 4 is used to control width of some pulses HW_TIMER(4, 0, 0, 0, OFF, 0) 'comparison completed, close mode 4 See below image: 10 pulses are generated under mode 2, and mode 4 takes effect after the reciprocal sixth pulse (because the first one does not change, then the reciprocal sixth has no change, the reciprocal fifth starts to act), it continuously controls 3 pulse widths, and each pulse accumulates 300ms as

	sequence, then fix width output. 
Instructions	HW_PSWITCH2

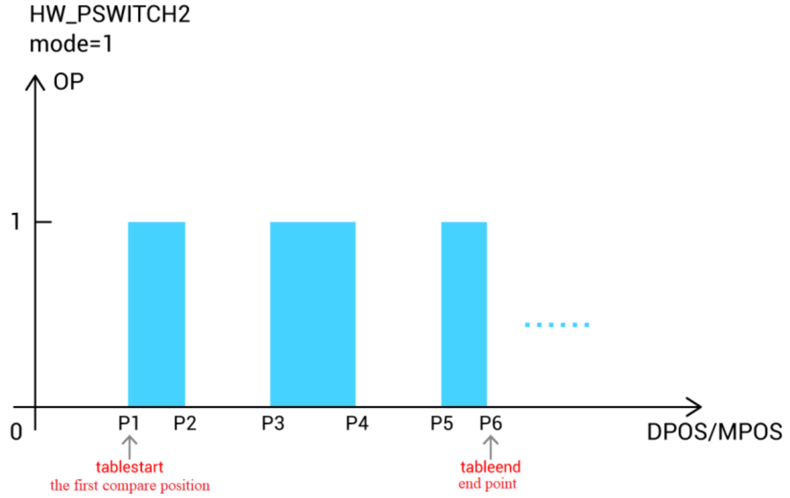
HW_PSWITCH2 -- Bus Hardware Position Comparison OUT

Type	Axis Instructions
Description	**Fieldbus position comparison output by hardware, it also supports pulse axis, but it must use assigned outputs.** There are 4 comparison outputs for ZMC4XX series, choose different outputs as per requirements. Usually, OUT 0/1/2/3 outputs. If the comparison master axis is attached with encoder input, encoder position will be as comparison position automatically, and accurate output time can be adjusted through MOVEOP_DELAY. Different fieldbus drives have different performance, also it can use MOVEOP_DELAY to adjust. HW_PSWITCH2 and MOVE_OP are based on same hardware resources, it is not recommended to call them together in one channel, it can be used in different channels. One comparison can only be done in every system period, see system period by SERVO_PERIOD. Don't change position data in TABLE before comparison is totally finished. Both pulse axis and fieldbus axis support this instruction. For pulse type motor, when ATYPE=4, it is the MPOS. Default ATYPE=1/7, it's DPOS.
Grammar	Command Grammar: HW_PSWITCH2(mode, [...]) Function Grammar: Buff =HW_PSWITCH2([axisnum]) **Mode=1:single axis comparison** HW_PSWITCH2(1,opnum,opstate,tablestart,tableend[,Direction]) mode: 1-start comparer opnum: relevant outputs opstate: output status of the first comparson position tablestart: TABLE number that saves the first absolute comparison coordinate tablesend: TABLE number that saves the last absolute comparison

coordinate

direction: the first coordinate to judge direction, 0-Negative, 1-Positive,-1-no direction

Description: comparison point is written into TABLE, and it will reverse once when reach one comparison position OP.



Mode=2:clear comparison position

HW_PSWITCH2(2)

mode: 2 means stop and delete all comparison positions that are not finished.

Description: if HW_PSWITCH2 doesn't compare all positions, mode must be set as 2, and stop and delete those uncompleted positions through HW_PSWITCH2(2), otherwise, this output channel will work abnormally later.

In vector compare mode, comparison is between VECTOR_MOVED and set positon, it is recommended to set an initial value of VECTOR_MOVED.

Mode=3:vector compare mode

HW_PSWITCH2(3,opnum,opstate,tablestart,tableend)

mode: 3-start comparer

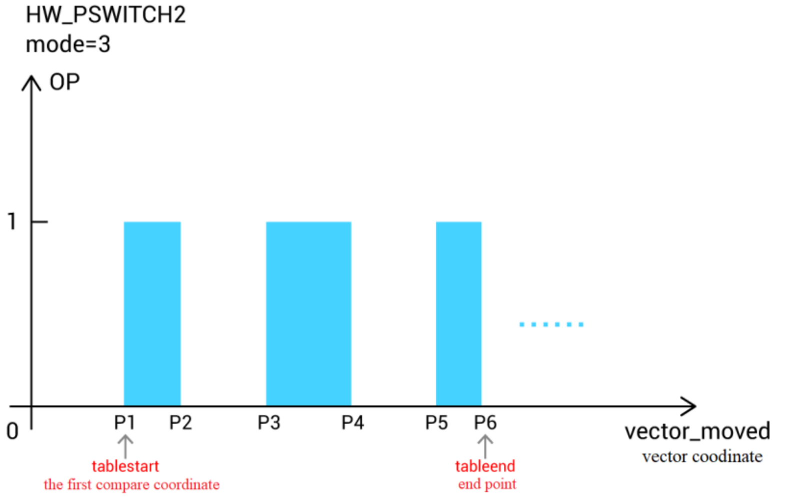
opnum: relevant outputs

opstate: output status of the first comparson position

tablestart: TABLE number that saves the first absolute comparison coordinate

tablesend: TABLE number that saves the last absolute comparison coordinate

Description: compare coordinate is filled in TABLE, OP will reverse once when reach one compare vector position.



Mode=4:vector compare mode, single comparison position.

HW_PSWITCH2(4,opnum,opstate,vectstart)

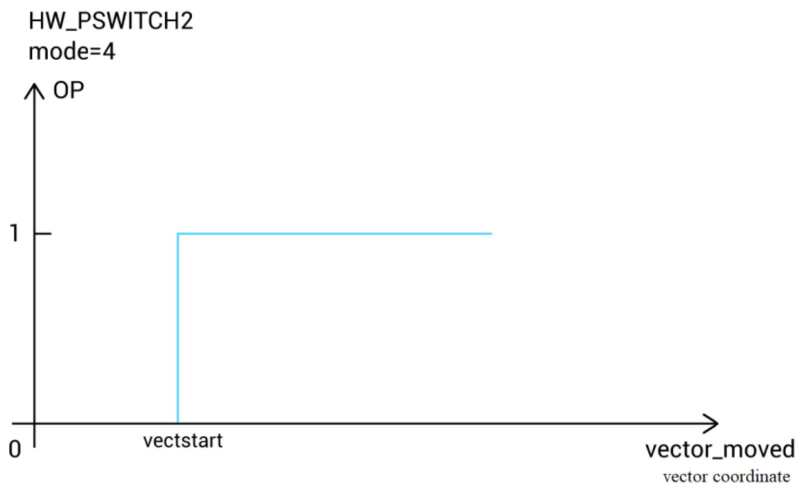
mode: 4-start comparer

opnum: relevant outputs

opstate: output status of the first comparson position

vectstart: absolute coordinate of comparison position

Description: achieve one compare vector position set by sommand, OP reverse, compare ends.



Mode=5: Vector compare mode, cycle pulse mode.

HW_PSWITCH2(5,opnum,opstate,vectstart,repes,cycledis,ondis)

mode: 5-start comparer

opnum: relevant outputs

opstate: output status of the first comparson position.

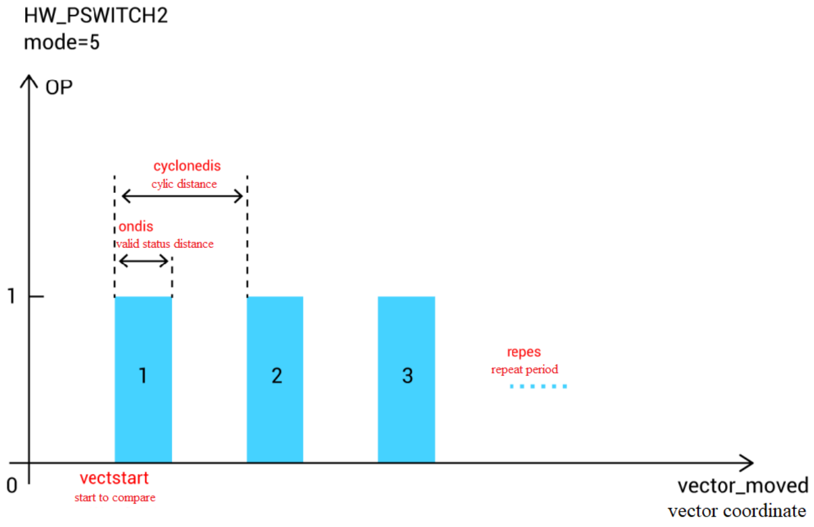
vectstart: compare point VECTOR_MOVED current motion distance

repes: repeat period, two times of comparison will be done in a period, output valid status first, then output invalid status.

cycledis: period distance, output opstate every time after cycledis, output recovers to invalid status after ondis.

ondis: distance that output valid status, (cycledis-ondis) is distance of invalid status.

Description: this mode doesn't need TABLE, and coordinates refer to vector coordinates, start to compare from vectstart, and compare once for each cycledis span, period is compared repeatedly is called repes, and after comparison signal is triggered each time, keep ondis distance, then close the signal to wait for next period.



Mode=6 :vector compare mode, cycle mode, it is used together with HW_TIMER

HW_PSWITCH2(6,opnum,opstate,vectstart,repes,cycledis)

mode: 6-start comparer

opnum: relevant outputs

opstate: output status of the first comparson position

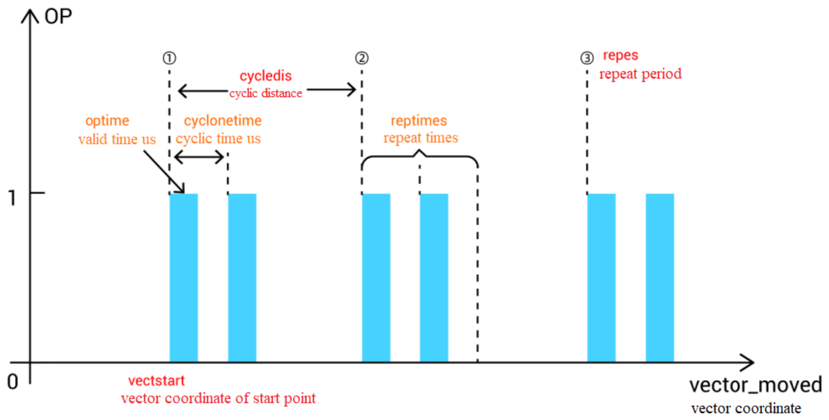
vectstart: compare point VECTOR_MOVED current motion distance

repes: repeat period, two times of comparison will be done in a period, output valid status first, then output invalid status.

cycledis: period distance, output opstate every time after cycledis, output recovers to invalid status after ondis.

Description: this mode doesn't need TABLE, and coordinates refer to vector coordinates, start to compare from vectstart, and compare once for each cycledis span, period is compared repeatedly is called repes, and after comparison signal is triggered each time, pulse width of hold signal is set through HW_TIMER, and HW_TIMER can reverse OP several times when reach one trigger point, after HW_TIMER period moved, wait for next period.

HW_PSWITCH2
mode=6



Mode=7 : it is used together with HW_TIMER

HW_PSWITCH2(7, opnum, opstate, tablestart, tableend [, optimeus, optimes, cyctimeus])

Mode: 7-start comparer, opstate not overturn, used togther with HW_TIMER.

Opnum: relevant ouputs

Opstate: output status of first comparison position

Tablestart: TABLE NO. that saves first comparison point VECTOR_MOVED coordinate

Tableend: TABLE NO. that saves last comparison point VECTOR_MOVED coordinate

[when used togther with hwtimer, it can adjust hwtimer parameters dynamically]

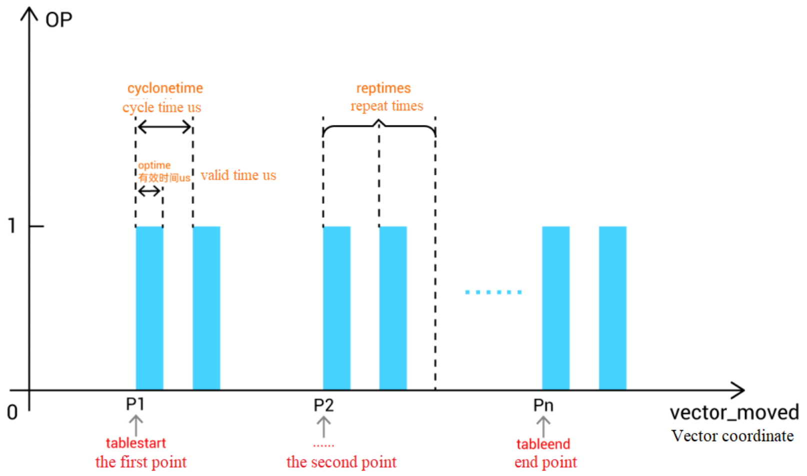
Optimeus: adjust HW_TIMER valid time dynamically.

Optimes: adjust HW_TIMER pulse amount dynamically, 0-not ouput

Cyctimeus: adjust HW_TIMER pulse period time dynamically

Description: compare point is entered into TABLE, and coordinates refer to vector coordinates, OP will be triggered when reach one TABEL compare vector position each time, now, pulse width of OP and compare times triggered each time are controlled by HW_TIMER. Next TABLE position is reached, OP will be triggered again.

HW_PSWITCH2
mode=7



2D, 3D compare mode as follow:

ZMC4XX series with firmware version 170706 or above support, and it must move in sequence to pass the point "fifo".

2D compare: 25, 26, save one point in every two table

3D compare: 35, 36, save one point in every three table

Compare multi-point, output overturn status every time.

HW_PSWITCH2(25, opnum, opstate, maxerr, num, tablepos)

HW_PSWITCH2(26, opnum, opstate, maxerr, num, tablepos, [ophwtimeus, ophwtimes, hwcyctimeus])

HW_PSWITCH2(35, opnum, opstate, maxerr, num, tablepos)

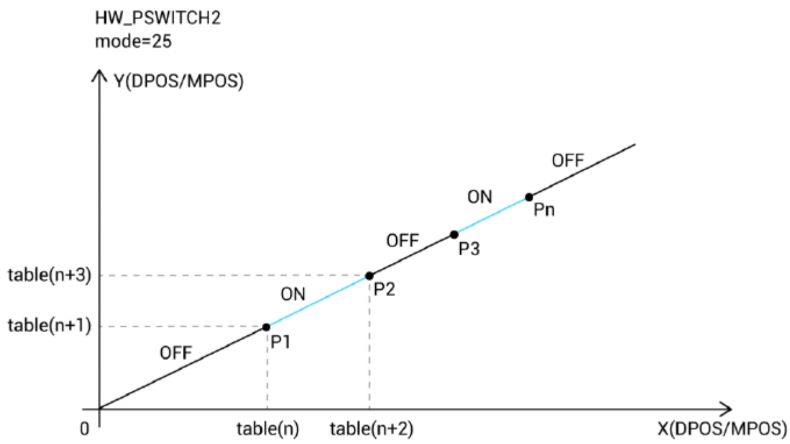
HW_PSWITCH2(36, opnum, opstate, maxerr, num, tablepos, [ophwtimeus, ophwtimes, hwcyctimeus])

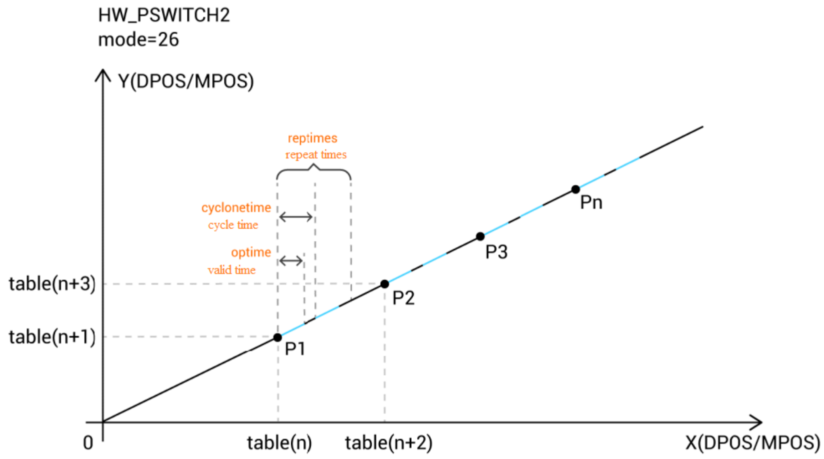
Parameters:

mode: 25,26,35,36 multi-dimension compare mode
opnum: relevant outputs
opstate: output status of first comparison position
maxerr: compare the pulse deviation of each axis left and right position, when in the deviation range, start comparison.
num: comparison position numbers saved in the table
tablepos: TABLE NO. that saves first absolute comparison coordinate
[when used together with hwtimer, can adjust hwtimer parameters dynamically.]
ophwtimeus: pulse time
ophwtimes: pulse numbers
hwcyctimeus: pulse period

Note: under this mode, deviation paremeter maxerr can't be written 0.

Mode = 8: single-axis comparison, same as mode 1, but it will not invert the signal

	HW_PSWITCH2 mode 1 and mode 8 can be used together with HW_TIMER command, but for mode 8, it is mainly used in single-axis fly photoing (each position is triggered by one certain time). **Command Specifc Usage Range :** HW_PSWITCH2 – Mode 1 & HW_TIMER: span one position, trigger once. HW_PSWITCH2 – Mode 8 & HW_TIMER: every position, trigger once. Pulse axis, bus axis, and virtual axis are OK. **Mode = 25: 2D comparison** HW_PSWITCH2(25, opnum, opstate, maxerr, num, tablepos) Description: comparison point is written into TABLE, two consecutive TABLE datas compose of one 2D coordinate, and OP reverse once time when reached one comparison position. Below is the example, blue segment means OP is ON, all kinds of common used interpolation motions support comparison, please pay attention the coordinate of compare point must be accurate, otherwise, following comparison positions will be affected.  **Mode = 26: 2D compare, reused together with HW_TIMER.** HW_PSWITCH2(26, opnum, opstate, maxerr, num, tablepos, [ophwtimeus, ophwtimes, hwcyctimeus]) Description: comparison point is written into TABLE, two consecutive TABLE datas compose of one 2D coordinate, OP will be triggered once when reached one comparison position, and the OP reverse times of each compare point and reverse period are set through HW_TIMER. When reached next TABLE, OP will be triggered again. It is simliar with mode 7 and mode 36.

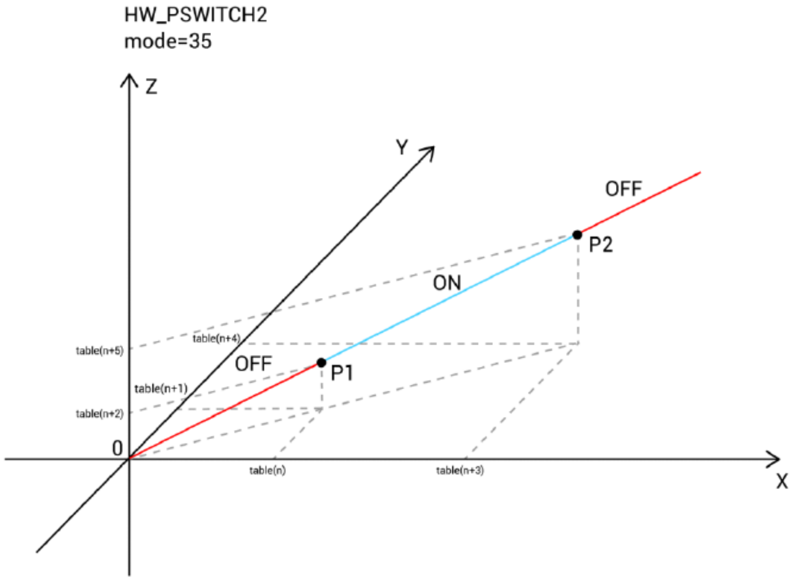


Mode = 35: 3D compare

HW_PSWITCH2(35, opnum, opstate, maxerr, num, tablepos)

Description: comparison point is written into TABLE, three consecutive TABLE datas compose of one 3D coordinate, and OP reverse once time when reached one comparison position.

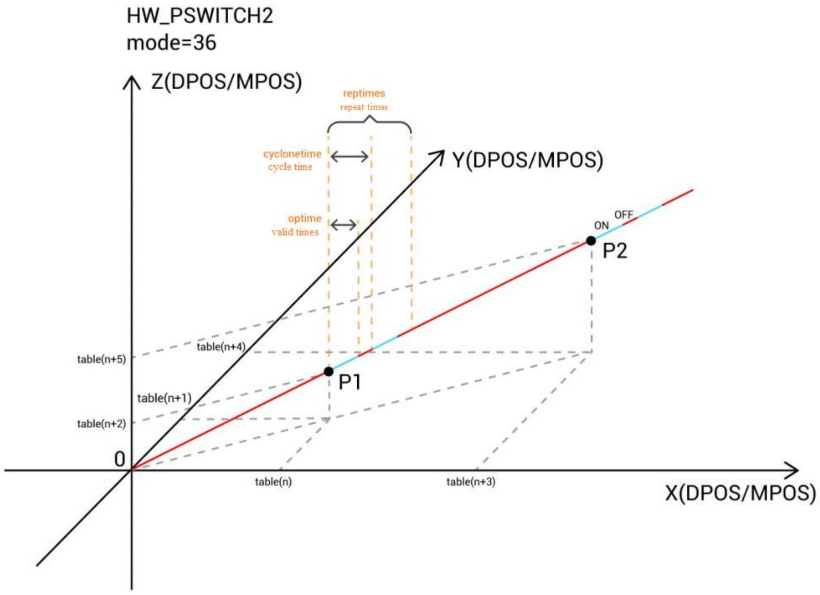
Below is the example, blue segment means OP is ON, all kinds of common used interpolation motions support comparison, please pay attention the coordinate of compare point must be accurate, otherwise, following comparison positions will be affected.



Mode = 36, 3D compare, reused together with HW_TIMER.

HW_PSWITCH2(36, opnum, opstate, maxerr, num, tablepos, [ophwtimeus, ophwtimes, hwcyctimeus])

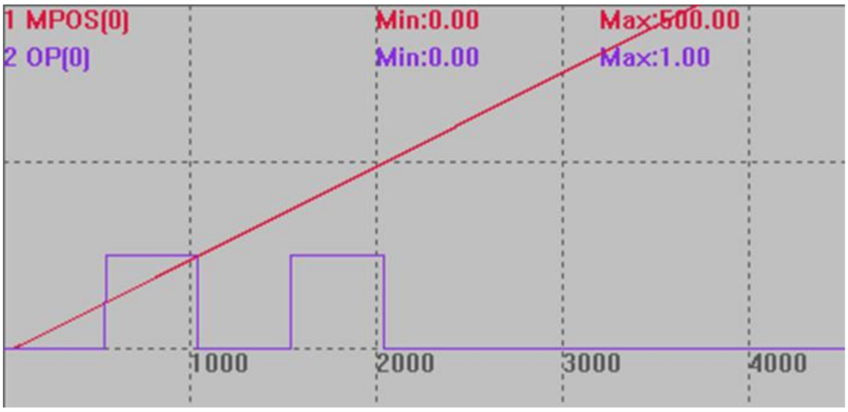
Description: comparison point is written into TABLE, three consecutive TABLE datas compose of one 3D coordinate, OP will be triggered once when reached one comparison position, and the OP reverse times of each compare point and reverse period are set through HW_TIMER. When reached next TABLE, OP will be triggered again. It is simliar with mode 7

<table>
<tr><td></td><td>and mode 26.


</td></tr>
<tr><td>Controller</td><td>Some ZMC3XX, above ZMC4XX Series and ZMC4XX series with firmware version above 20170704 supports.</td></tr>
<tr><td>Example</td><td>Testing: ZMC432, firmware: 20170709(simulator can’t run this instruction.)

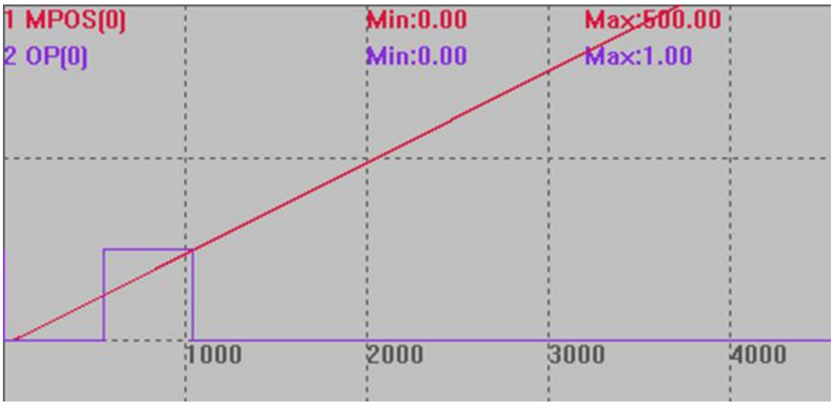
Example One: single axis comparison, mode = 1
Fieldbus enable process is omitted here, see sample: Fieldbus Initialization.
BASE(0)
DPOS=0
MPOS=0
OP(0,OFF)
TABLE(0,50,100,150,200) 'comparison coordinate
HW_PSWITCH2(2) 'stop and delete comparison that not finished.
HW_PSWITCH2(1, 0, 0, 0, 3,1) 'compare 4 positions, operate output 0.
TRIGGER 'trigger oscilloscope
MOVE(500)

Comparison output graph:
When moves to 50, open OUT0; when moves to 100, close OUT0; when moves to 150, open OUT0, when moves to 200, close OUT0.
MPOS(0)=200(vertical scale)
OP(0)=2(vertical scale)</td></tr>
</table>



Example Two: vector compare mode, single axis comparison, mode = 3.
Fieldbus enable process is omitted here, see sample: Fieldbus Initialization.
BASE(0)
DPOS=0
MPOS=0
OP(0,OFF)
TABLE(0,50,100,150,200) 'comparison coordinate
VECTOR_MOVED=100 'set vector start position, it will affect compare point
HW_PSWITCH2(2) 'stop and delete comparison that not finished.
HW_PSWITCH2(3,0,1,0,3,) 'compare 4 positions, operate output 0.
TRIGGER 'trigger oscilloscope
MOVE(300)

Comparison Output Graph:
MPOS(0)=200(vertical scale)
OP(0)=2(vertical scale)



It can be seen signal is operated starting from position 12 of example 1, it only compared later 2 points. This mode compares MPOS (current position) and VECTOR_MOVED (former distance) and set position to compare.

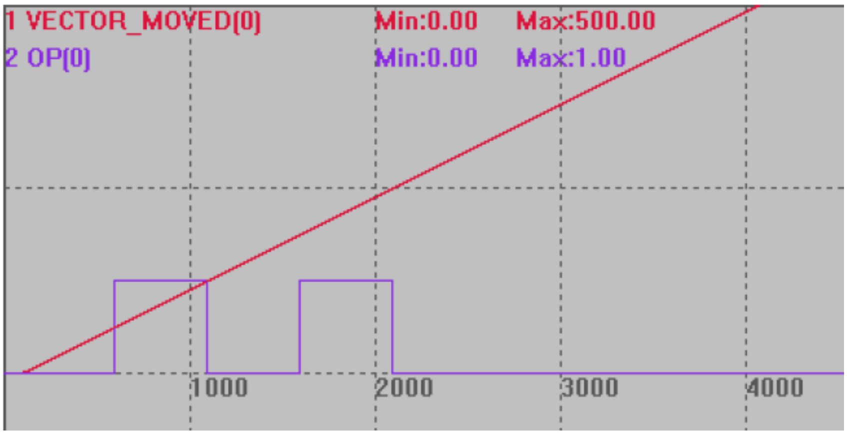
Example Three: mode = 3, vector multi-axis comparison mode, PSO application of XY processed mode.
RAPIDSTOP(2)

```
WAIT IDLE(0)
WAIT IDLE(1)
BASE(0,1)
ATYPE=1,1
SPEED=100,100
ACCEL=1000,1000
DECEL=1000,1000
SRAMP=100,100
DPOS=0,0
MPOS=0,0
OP(0,OFF)
TABLE(0,50,100,150,200)    'compare point coordinate configuration
VECTOR_MOVED=0             'set vector start position
HW_PSWITCH(2)              'stop and delete comparison that is not finished
HW_PSWITCH(3,0,1,0,3)      'compare 4 points, operate OUT0
TRIGGER                    'trigger oscilloscope
MOVE(300,400)
END
```

Compare output: compare according to vector position resultantly interpolated by axis X and axis Y.

VECTOR_MOVED(0) = 200 vertical scale

OP(0) = 2 vertical scale

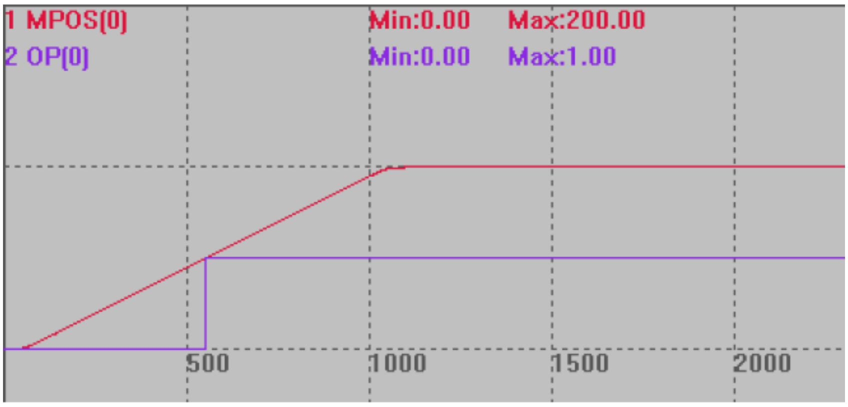


Example Four: mode = 4, single point vector comparison

```
BASE(0)
DPOS=0
MPOS=0
OP(0,OFF)
VECTOR_MOVED(0) = 0        'set the current vector position
HW_PSWITCH(2)              'stop and delete comparison that is not finished
HW_PSWITCH2(4,0,1,100)     'open compare output, mode 4, output 0, the
                           first compare point outputs ON, start to
                           compare from vector position 100, and it
                           only compared once time.
```

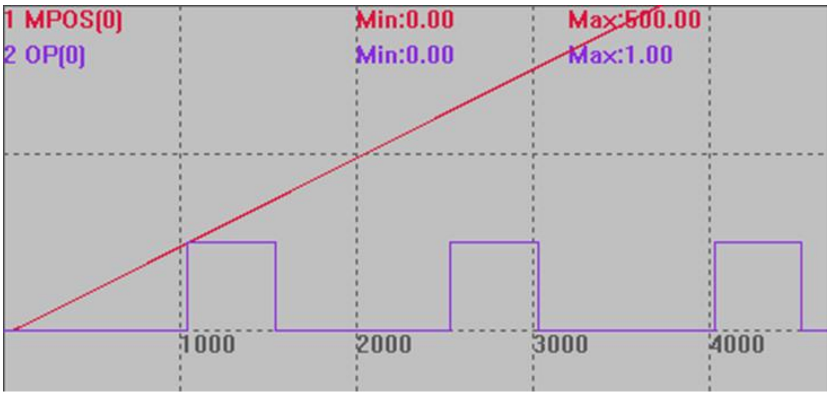
TRIGGER 'trigger oscilloscope to sample
MOVEABS(200)

Compare output:
MPOS(0) = 100 (vertical scale)
OP(0) = 2(vertical scale)



Example Five: cycle compare mode, recover after distance.
Fieldbus enable process is omitted here, see sample: Fieldbus Initialization.
BASE(0)
DPOS=0
MPOS=0
OP(0,OFF)
VECTOR_MOVED=0 'set start position of vector as 0 for observing
HW_PSWITCH2(2) 'stop and delete comparison that not finished.
HW_PSWITCH2(5,0,1,100,3,150,50) 'start to compare at 100, all 3 times cycle distance is 150, output valid distance is 50
TRIGGER 'trigger oscilloscope
MOVE(500)

Comparison Output Graph:
Comparison starts at position 100, compare 3 times, first distance 50 output is valid in one period, and followed distance 100 output is invalid.
MPOS(0)=200(vertical scale)
OP(0)=2(vertical scale)



Example Six: mode = 6, single axis cycle compare mode, time reset.

Fieldbus enable process is omitted here, see sample: Fieldbus Initialization.

BASE(0)

DPOS=0

MPOS=0

OP(0,OFF)

VECTOR_MOVED=0 'set start position of vector as 0 for observing

HW_PSWITCH2(2) 'stop and delete comparison that not finished.

HW_PSWITCH2(6,0,1,100,4,100) 'start to compare at 100, compare 4 times cycle distance of 100, output valid time is determined by HW_TIMER.

HW_TIMER(2,100000,50000,1,off,0) 'output is on, and after 50ms, it is off.

TRIGGER

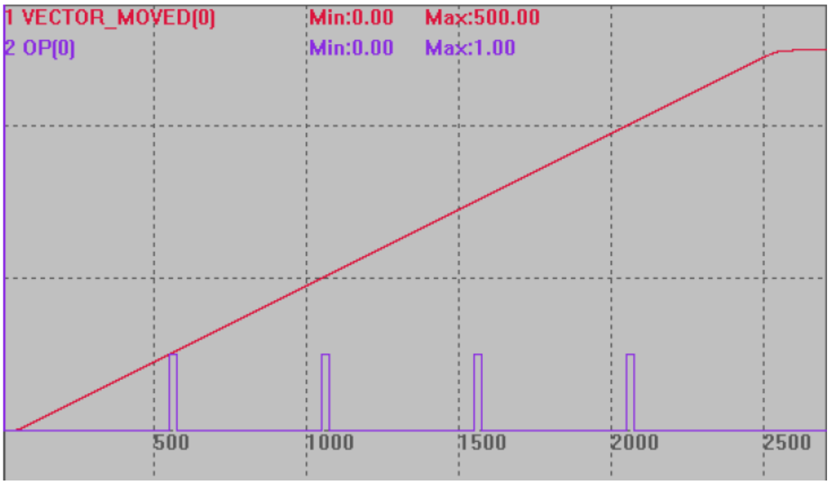
MOVE(500)

Comparison Output Graph:

Comparison starts at position 100, compare 4 times. Turn to OFF 500ms later after output is ON.

MPOS(0)=100(vertical scale)

OP(0)=2(vertical scale)



Example 7: mode = 6, multi-axis cycle comparison mode, time recover.

BASE(0,1)

DPOS=0,0

MPOS=0,0

OP(0,OFF)

TABLE(0,50,100,150,200) 'compare point coordinate configuration

VECTOR_MOVED=0 'set start position of vector as 0 for observing

HW_PSWITCH2(2) 'stop and delete comparison that not finished.

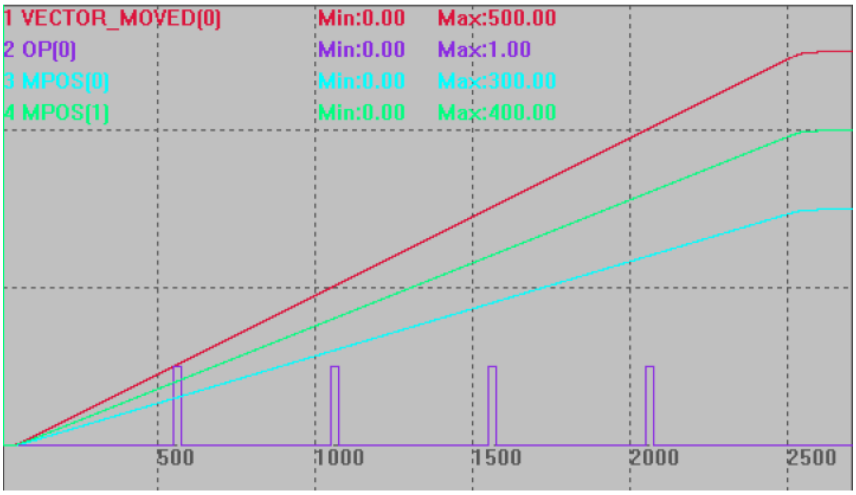
HW_PSWITCH2(6,0,1,100,4,100) 'start to compare at 100, compare 4 times cycle distance of 100, output valid time is determined by HW_TIMER.

HW_TIMER(2,100000,50000,1,off,0) 'output is on, and after 50ms, it is off.
TRIGGER
MOVE(300,400)

Comparison Output Graph:

Coordinates are resultant vector position of two axes, comparison starts at position 100, compare 4 times. Turn to OFF 500ms later after output is ON.

VECTOR_MOVED(0) = 200 (vertical scale)
OP(0)=2(vertical scale)
MPOS(0)=200(vertical scale)
MPOS(1)=200(vertical scale)



Example 8: mode = 25, 2 dimensions hardware position comparison output

BASE(0,1) 'select axis X and axis Y
UNITS=1000,1000
MPOS=100,100
SPEED=100,100
ACCEL=10000,10000
DECEL=10000,10000

'set the current position as point 0
MPOS=0,0
MPOS=0,0

'write XY coordinates of position comparison points to table 10~49 in advance
TABLE(10, 10,0,12,0, 20,0,22,0, 30,0,32,0, 50,0,52,0, 52,10,52,12, 52,20,52,22, 52,30,52,32, 52,40,52,42, 52,50,52,52)
GLOBAL pointNum 'comparison numbers
pointNum = 20

GLOBAL startX, startY 'start point
startX=0

```
startY=0

GLOBAL midX, midY           'middle point
midX=52
midY=0

GLOBAL endX, endY           'end point
endX=52
endY=52

WHILE 1
    IF TABLE(0) = 1 THEN   'compare pulse position, not precision output
        WAIT IDLE
        ?“compare pulse position, not precision output”
        'set parameters
        SYSTEM_ZEST=1
        AXIS_ZEST=1,1

    ELSEIF TABLE(0)=2 THEN 'compare pulse position, precision output
        WAIT IDLE
        ?“compare pulse position, precision output”
        'set parameters
        SYSTEM_ZEST=3
        AXIS_ZEST=3,3

    ELSEIF TABLE(0)=3 THEN 'compare encoder position, not precision
                                output
        WAIT IDLE
        ?“compare encoder position, not precision output”
        'set parameters
        SYSTEM_ZEST=17
        AXIS_ZEST=17,17

    ELSEIF TABLE(0)=4 THEN 'compare encoder position, precision
                                output
        WAIT IDLE
        ?“compare encoder position, precision output”
        'set parameters
        SYSTEM_ZEST=19
        AXIS_ZEST=19,19
    ENDIF

    IF TABLE(0)<>0 AND IDLE(0) = -1  THEN
        TABLE(0)=0
        ?“open”

        HW_PSWITCH2(2)      'clear all comparisons
        HW_PSIWTCH2(25,0,1,10,pointNum,10)      'write to compare,
```

operate OUT0

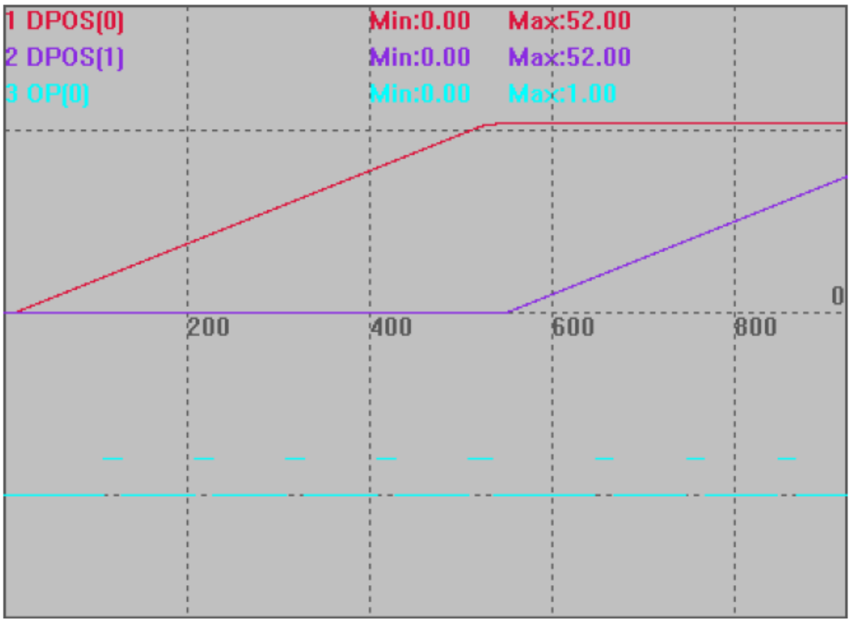
```
        WA 10
        TRIGGER                    'open oscilloscope
        MOVEABS(startX, startY)    'start to move
        MOVEABS(midX, midY)
        MOVEABS(endX, endY)
    ENDIF
WEND
END
```

Compare output:

DPOS(0) = 50 (vertical scale)

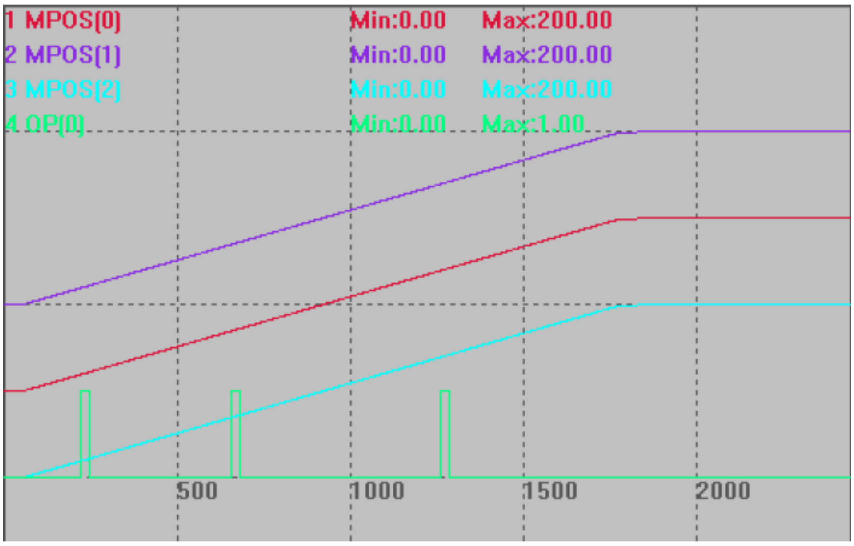
DPOS(1) = 50 (vertical scale)

DPOS(0) = 5 (vertical scale)



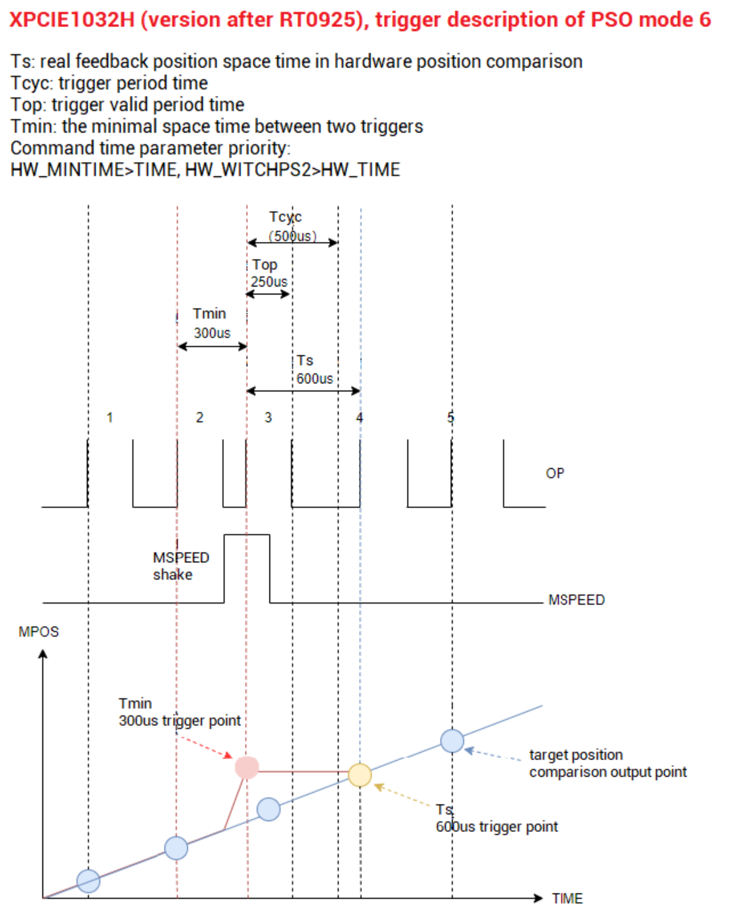
For example 9: mode = 35, 3D hardware position comparison output

```
BASE(0,1,2)
DPOS=0,0,0      'set the current position as 0
MPOS=0,0,0
OP(0,OFF)
TABLE(0, 20,20,20, 40,40,40, 70,70,70, 100,100,100, 140,140,140, 180,180,180)
HW_PSWITCH2(2) 'stop and delete comparison points that are not finished
HW_PSWITCH2(35,0,1,10,6,0) 'start to compare and output, mode 35, OUT0, the first compare point outputs ON, pulse deviation 10, table address 0-18, 6 coordinates.
TRIGGER                    'trigger oscilloscope to sample
```

	MOVEABS(200,200,200) 'move straight Compare output: MPOS(0) = 200 (vertical scale) MPOS(1) = 200 (vertical scale) MPOS(2) = 200 (vertical scale) OP(0) = 2 (vertical scale) 
Instructions	MOVE_HWPSIWTCH2, PSWITCH, HW_PSWITCH, MOVEOP_DELAY, REG_INPUTS

HW_MINTIME – HW Min Time Space

Type	System Parameters
Description	**Set minimal time space of HW precison output (unit is millisecond)** **HW_MINTIME should be bigger than pulse width of HW_TIMER, and smaller than the minimal space.**

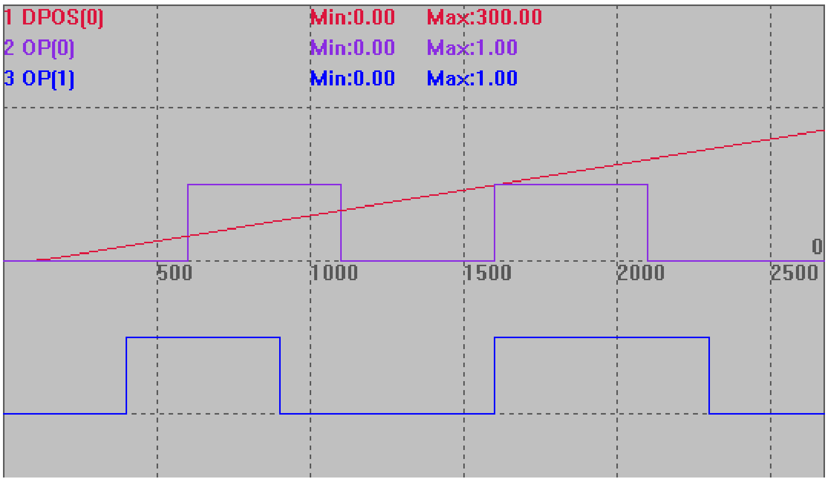
	
Grammar	HW_MINTIME(opnum)=expression opnum: output (OUT / OP) No., the port must support hardware comparison output.
Controller	Valid in special firmware.
Example	HW_MINTIME (1) = 1 ‘1ms at least between former one HW output and behind one HW output for OP 1
Instructions	HW_PSWITCH2, HW_TIMER

HW_PS2AXISNUM—Set PS2 Axis Number

Type	Axis Instructions
Description	Set HW_PSWITCH2 axis NO. to be actually operated, the default value -1 means not modify. Used to reuse HW_PSWITCH2 buffer of axis that is not operated, indicating current motion main axis, and it can do multi-comparison for current main axis.
Grammar	HW_PS2AXISNUM(axisnum1)=axisnum2 axisnum1: buffer axis NO. axisnum2: axis NO. to be actually operated

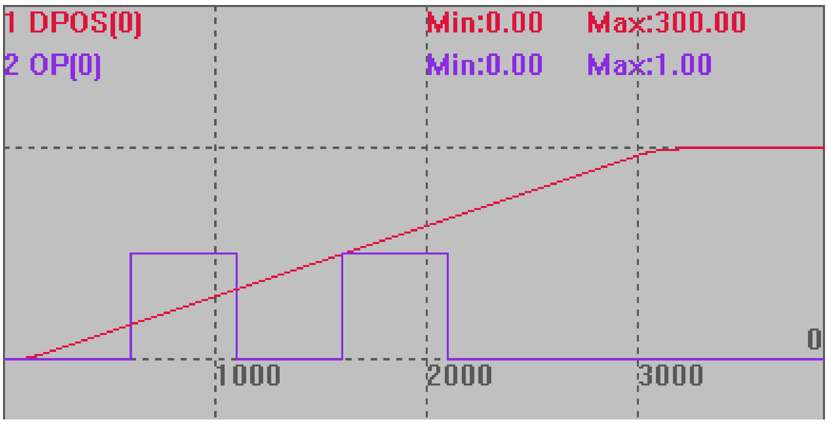
Controller	Valid in 4 series with firmware version above 170705.
Example	Testing: ZMC432, firmware: 20170709(simulator can't run this instruction.)

```
RAPIDSTOP(2)
WAIT IDLE(0)
WAIT IDLE(1)
BASE(0,1)
ATYPE=1,1
UNITS=100,100
SPEED=100,100
ACCEL=500,500
DPOS=0,0
TRIGGER
OP(0, OFF)
OP(1, OFF)

HW_PS2AXISNUM(1)=0      'use axis 1 buffer, compare axis 0 position
VECTOR_MOVED =0
TABLE(0,50,100,150,200)    'set the first comparison coordinate
TABLE(10,30,80,150,220)    'set the first comparison coordinate

HW_PSWITCH2(1, 0, 1, 0, 3,1)                  'the first comparison
HW_PSWITCH2(1, 1, 1, 10, 13,1)  AXIS(1)   'the second comparison, use
                                            axis 1 comparison buffer,
                                            but compare axis 0 position
                                            actually.
MOVE(300)
WAITIDLE(0)
HW_PS2AXISNUM(1)=-1        'cancel
END
```



Instructions	HW_PSWITCH2

HW_PS2COUNTS—PS Comparison Numbers

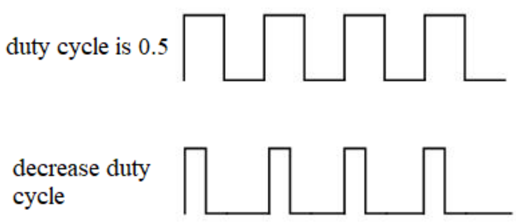
Type	Axis status
Description	HW_PSWITCH2 instruction means the number of compared points actually, when it is HW_PSWITCH2(2), clear as 0.
Grammar	VAL = HW_PS2COUNTS (axisnum) axisnum: axis No.
Controller	ZMC4XX series controller with firmware version above 170706 supports.
Example	Testing: ZMC432 Firmware: 20170711(simulator can't run this instruction) RAPIDSTOP(2) WAIT IDLE(0) BASE(0) ATYPE=1 UNITS=100 SPEED=100 ACCEL=500 DPOS=0 MPOS=0 OP(0,OFF) TABLE(0,50,100,150,200) 'set comparison coordinate HW_PSWITCH2(2) 'stop and delete comparison that not finished HW_PSWITCH2(1, 0, 1, 0, 3,1) 'compare 4 points, operate output0 TRIGGER 'trigger oscilloscope MOVE(300) ?HW_PS2COUNTS 'print 0, not arrive comparison point WAIT IDLE(0) ?HW_PS2COUNTS 'print 4, compared 4 points END 
Instruction	HW_PSWITCH2

12.4 PWM Control Instructions

PWM_FREQ--PWM Frequency

Type	PWM control functions
Description	**PWM frequency setting or reading.** When set duty cycle as 0, PWM can be closed, PWM frequency as 0, it still opens. Don’t set frequency as 0, PWM frequency must be modified before PWM switch.
Grammar	PWM_FREQ (index, freq) or PWM_FREQ (index)=freq index PWM output NO., start from 0 freq frequency, hardware PWM is 1M, software PWM is 2k
Controller	Controllers that support PWM
Example	**PWM_FREQ** (0)=1000 ‘frequency is 1K ?**PWM_FREQ** (0)
Instructions	PWM_DUTY, MOVE_PWM

PWM_DUTY--Duty Cycle of PWM

Type	PWM control functions
Description	**PWM duty cycle setting or reading.** When set duty cycle as 0, PWM can be closed, PWM frequency as 0, it still opens. Don’t set frequency as 0, PWM frequency must be modified before PWM switch. Duty cycle means the ratio of valid electric level to whole period. In one period, output valid electric level first, then output invalid electric level.  PWM actual output is controlled by outputs, so should open outputs, then PWM can output successfully, or will be shielded. Realize first pulse restrain function of laser power supply, first to open PWM function, then open outputs.
Grammar	PWM_DUTY(index, duty) or PWM_DUTY(index)=duty index: PWM output NO., starts from 0 duty: duty cycle value: 0-1, when it is set as 0, then PWM closes.
Controller	Controllers that support PWM
Example	**PWM_DUTY**(0)=0.5

	?**PWM_DUTY**(0) Print result: 0.5
Instructions	PWM_FREQ, MOVE_PWM

12.5 Buzzer Control Commands

SPEAKOUT – Buzzer Control

Type	PWM control functions
Description	**Control the buzzer to speak out.**
Grammar	SPEAKOUT (timems, [freq], [duty]) timems: how long the buzzer sound lasts, the unit is ms. freq: frequency duty: duty cycle frequency and duty cycle are usually set as invalid.
Controller	General
Example	**SPEAKOUT (5000) ‘the buzzer sounds 5s.**

Chapter XIII Instructions Related to Communication

13.1 Serial Communication Instructions

SETCOM -- Serial Port Configuration

<table>
<tr><td>Type</td><td>System Instructions</td></tr>
<tr><td>Description</td><td>Serial port configuration
When controller restart after powered off, parameters of SETCOM will restore as default value, please add SETCOM setting at beginning of procedure.
ZMC00X series don’t suport MODBUS communication as master.
Generally, to switch RS485 station No., please add one delay.</td></tr>
<tr><td>Grammar</td><td>SETCOM(baudrate,databits,stopbits,parity,port[,mode][,variable][,timeout])
baudrate: baudrate: 9600 19200 4800 115200 38400(default) 57600
databits: data bit: 8
stopbits: stop bit: 0/1/2
parity: verify or not:
<table>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>0 (default)</td><td>No verification</td></tr>
<tr><td>1</td><td>Verified while odd</td></tr>
<tr><td>2</td><td>Verified while even</td></tr>
</table>
port: serial PORT number: 0-1. see PORT as reference, it differs from controller modes
mode: protocol:
<table>
<tr><td>value</td><td>Description</td></tr>
<tr><td>0</td><td>RAW data mode, no protocol, at this time, use GET, PRITNT # to transfer data.</td></tr>
<tr><td>4 (default)</td><td>MODBUS Slave (16 bits integer)</td></tr>
<tr><td>14</td><td>MODBUS Master (16 bits integer)</td></tr>
<tr><td>15</td><td>Direct command mode, it can input character string directly through serial port (use line feed to make an end)</td></tr>
</table>
variable: choose registor type, 0-VR, 1-TABLE, 2-MODBUS register in system.
<table>
<tr><td>value</td><td>Description</td></tr>
<tr><td>0</td><td>VR, one VR is mapped to one MODBUS_REG in this situation.
VR is 32 bits float type, REG is 16 bits integer</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td></td><td>type, when VR type value was transmitted to REG type, the decimal part will get lost. if the VR data exceeds positive or minus 15 bits, then the REG data will change.
No loss will happen if REG was transmitted to VR.</td></tr>
<tr><td>1</td><td>TABLE, one table is mapped to MODBUS_REG. (it is not recommended).
Table is 32 bits float type, REG is 16 bits integer type, when TAVLE type value was transmitted to REG type, the decimal part will get lost. If the VR data exceeds positive or minus 15 bits, then the REG data will change.
No loss will happen if REG was transmitted to TABLE.</td></tr>
<tr><td>2 (default)</td><td>MODBUS in system, VR and MODBUS both belong to two independent registor areas in this situation.</td></tr>
<tr><td>3</td><td>VR_INT mode, one VR_INT is mapped to two MODBUS_REG in this situation.</td></tr>
</table>
timeout: this parameter takes effect when MODBUS slave station. Timeout means frame the longest delay time (millsecond ms), for old firmware, the dafault value is 800ms. When the delay time is set too small for some touch screens, it will cause error and it can’t be recovered. Therefore, set this value according to different touch screens. This is added in ZMC4XX series controllers firmware version 20190203. Check and view through ?*SETCOM. If this parameter is shown, which means it supports.

variable parameter is a kind of global configuration, all ports share one.
When registor is set as VR or TABLE, general outputs will be mapped to MODBUS_BIT(0), and general inputs will be mapped to MODBUS_BIT(1000), it is not recommended to use MODBUS_BIT as HMI button in this situation.
When register is set as VR or TABLE, outputs and inputs that are not used will connect together.
</td></tr>
<tr><td>Controller</td><td>General</td></tr>
<tr><td>Example</td><td>Example one: mode0, RAW mode
DIM char1 'define the variable
SETCOM(38400,8,1,0,0,0) 'configured as RAM mode.
WHILE 1
GET #0, char1 'save the character sent to channal 0 in char1
PRINT char1 'print the character received from channal 0 in ASCII code.</td></tr>
</table>

PUTCHAR #0, char1 'send the received character back.
WEND
See related samples in ChapterXIII for reference of encoder read and write of Panasonic A6

Example two: MODBUS communication configuration
SETCOM(38400,8,1,0,0,4,2) 'set serial port 0 as MODBUS slave, baudrate is 38400.
SETCOM(38400,8,1,0,1,14,2,1000) 'set serial port 1 as MODBUS master, baudrate is 38400.
See sample procedure of MODBUSM_DES as reference.

Example three: direct character command mode.
setcom(38400,8,1,0,0,15) 'set serial port 0 as direct character command mode

At this time, directly send related commands to operate the controller in serial port debugging help or other devices.

Before sending:
UNITS=10000
After sending:
UNITS=100

Example four: register mode0
VR(0)=0 'initialize VR(0) and REG(0) as 0
MODBUS_REG(0)=0
SETCOM(38400,8,1,0,0,4,0) 'map VR to MODBUS_REG
VR(0)=100.345 'set VR(0) as100.345
?MODBUS_REG(0) 'print result is 100, since VR is already mapped to REG, reg is integer type, so the fractional part is missed
MODBUS_REG(0)=200 'set REG(0) as 200
?VR(0) 'print result is 200, VR also will change as per the REG

Example five: register mode 2
VR(0)=0 'initialize VR(0) and REG(0) as 0
MODBUS_REG(0)=0
SETCOM(38400, 8,1,0,0,4,2) 'set VR and MODBUS_REG as independent.
VR(0)=100.345 'set VR(0) as100.345
?MODBUS_REG(0) 'print result is 0, no relation between VR and REG

	MODBUS_REG(0)=200 'set REG(0) as 200 ?VR(0) 'print result is 100.345, no relation between VR and REG
Instructions	ADDRESS, PROTOCOL, MODBUSM_DES, PORT

ADDRESS--Controller Station NO.

Type	System Parameters
Description	**MODBUS Protocol based station NO. of all controller serial ports is :1-255. Default value is 1.**
Grammar	ADDRESS = value
Controller	General
Example	Print ADDRESS 'print protocol station NO. Result: 1
Instructions	SETCOM, PORT, PROTOCOL

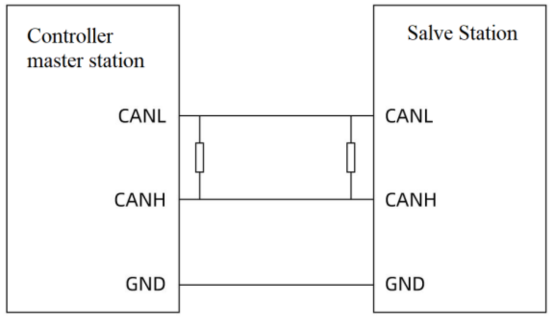
COM_UNUSED—Assign Serial Port

Type	System parameters
Description	**Whether assigned serial ports as per bit are used or not by ZBASIC.** Bit0: serial port 1, value as 1 means not use ZBASIC Bit1: serial port 2, value as 1 means not use ZBASIC
Grammar	COM_UNUSED = value
Controller	Above ZMC5XX series controllers
Example	COM_UNUSED=1 'ZBASIC doesn't use RS232 serial port
Instruction	SETCOM, PORT, PROTOCOL

13.2 CAN Communication Instruction

CAN -- CAN Communication

Type	System Instruction
Description	**Directly receive and send data through CAN.** Multi Controllers communication can be achieved through CAN, but there is only one master station in one same CAN network (CANIO_ADDRESS=32). Please note only newer firmware versions support this function. If it is invalid, please contact manufacturer. Wiring as follow:



CANL-CANL
CANH-CANH
A 120-ohm resistance is connected at both ends of CANL and CANH for resist matching. When link with module expansion that has dial switch, dial the 8th as ON, which means the 120-ohm resistance is connected, no need to connect resistance externally.

Grammar

CAN(channel, function, tablenum)

channel: CAN channel, 0-first channel, -1-default channel.

function: function No.

Value	Description
6	receive data, when there is no data, identifier<0
7	send data
16 (please upgrade firmware)	receive extended data, when there is no data, identifier<0
17 (please upgrade firmware)	send extended data, use 7 to send ordinate data

tablenum: TABLE position where saves data.

When function is 6 or 7, data will be saved in order:

identifier: CAN communication object(cob-id), this value consists of 11 bits, the former 4 bits are function codes, the last 7 bits are node ID. ZCAN uses high-bit data reserved, 0-511 is recommended. When value is less than 0, it indicates there is no data received. Bit11 means whether it is a remote frame.

bytes: number of bytes in the data area, maximum is 8 bytes.

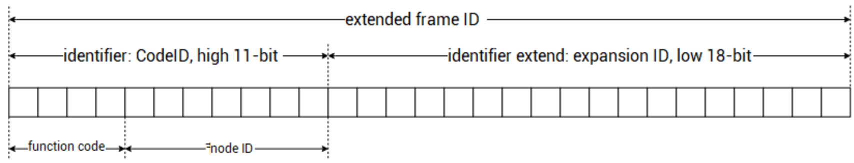
data: data area, byte (0-FF)

When function is 16 or 17:

identifier: CAN communication object(cob-id), this value consists of 11 bits, the former 4 bits are function codes, the last 7 bits are node ID. ZCAN uses high-bit data reserved, 0-511 is recommended. when value is less than 0, it indicates there is no data received. Bit11 means whether it is a remote frame.

identifier extend: extend id, 11 high bits and 18 low bits, that is, there are 29 CAN bits, fill in -1 when no ID exists.

bytes: the number of bytes in the data area, maximum is 8 bytes.

	data: data area, byte(0-FF) The 29-bit extended frame ID is divided into low-order 18 bits and high-order 11 bits starting from the last digit on the right. If the high-bits are less than 11 bits, 0 will be automatically added. The example is as follows:  Extended frame ID is:1753001 The relevant binary is: 1 0111 0101 0011 0000 0000 0001 Binary of 29-bit: 0 0001 0111 0101 0011 0000 0000 0001 High 11-bit: 0 0001 0111 01 (corresponding decimal part is 93) Low 18-bit: 01 0011 0000 0000 0001 (corresponding decimal part is 77825) Then: When CAN17 mode is used, identifier = 93, identifier extend = 77825
Controller	General
Example	**Example one:** 'send TABLE(0,1,8,1,2,3,4,5,6,7,8) 'send cobid=1, 8 bytes: 1-8. **CAN**(0,7,0) 'send data 'receive CANIO_ADDRESS=1 'set as salve, it only sets once. **CAN**(0,6,0) 'receive data ?TABLE(0) **Example two:** 'send TABLE(0,1,10,8,1,2,3,4,5,6,7,8) 'send cobid=1, extend id10, 8 bytes, 1-8 **CAN**(0,17,0) 'send data 'receive CANIO_ADDRESS=1 'set as salve, it only sets once. **CAN**(0,16,0) 'receive data ?TABLE(0) **Example three: CAN mode 17 (avoid blocking), independent task ON** When no device is connected, CAN17 will jump automatically, it will not appear blocking and waiting. CANIO_ADDRESS = 32 'master station GLOBAL FLAG_SEND_NUM 'sending times FLAG_SEND_NUM = 1000 WHILE 1 ?FLAG_SEND_NUM

	IF FLAG_SEND_NUM > 0 THEN
	TABLE(0,32,8,8,1,2,3,4,5,6,7,8)
	STOPTASK 2
	RUNTASK 2,CAN_SEND
	WA 10 'add a bit delay, make sure following sending won't interrupt former sending
	ENDIF
	IF FLAG_SEND_NUM <= 0 THEN
	?"Sending Over"
	EXIT WHILE
	ENDIF
	WEND
	END
	
	GLOBAL SUB CAN_SEND()
	FLAG_SEND_NUM = FLAG_SEND_NUM - 1
	CAN(0,17,0)
	?"Sending Succeed"
	END SUB

Instructions	CANIO_ADDRESS, CANIO_STATUS, CANIO_ENABLE

CANIO_ADDRESS--CAN Communication Setting

Type	System Parameters
Description	**CANID and CAN SPPEED (baud rate) setting of CAN communication on controller.** Can speed of IO expansion should be set through dial-up switches attached, setting value will be saved into FLASH, valid after restarting. There is 16 bits to indicate the setting of CANID and CANSPEED. Lower 8 bits (bit:0-7) indicate CANIO setting, value is 0-32, default value is 32, which means master controller. Upper 8 bits (bit:8-15) indicate CANSPEED, sample values as follow : **Value in upper 8 bits** / **CAN baud rate**: 0 — 500KBPS(default value) 1 — 250KBPS 2 — 125KBPS 3 — 1MBPS Note: 1. Don’t configure multi master controllers in one communication net. 2. Setting of CANID and CANSEED will be effective only after restarting.
Grammar	CANIO_ADDRESS = value
Controller	General
Example	CANIO_ADDRESS=32 'set as master,CAN baud rate is 500KBPS CANIO_ADDRESS=32+256 'set as master,CAN baud rate is 250KBPS

	CANIO_ADDRESS=32+512 'set as master,CAN baud rate is 125KBPS CANIO_ADDRESS=32+768 'set as master,CAN baud rate is 1MBPS CANIO_ADDRESS=1 'set CANID=1 as slave, 500KBPS, can't connect IO expansion. CANIO_ADDRESS=1+256 'set CANID=1 as slave, 250KBPS, used to ZCAN slave station CANIO_ADDRESS=1+512 'set CANID=1 as slave, 125KBPS, used to ZCAN slave station CANIO_ADDRESS=1+768 'set CANID=1 as slave, 1MBPS, used to ZCAN slave station CANIO_ADDRESS=3 'set CANID=3 as slave, 500KBPS, used to ZCAN slave station CANIO_ADDRESS=8+256 'set CANID=8 as slave, 250KBPS, used to ZCAN slave station CANIO_ADDRESS=16+512 'set CANID=16 as slave, 125KBPS, used to ZCAN slave station CANIO_ADDRESS=31+768 'set CANID=31 as slave, 1MBPS, used to ZCAN slave station See CAN communication example.
Instructions	CANIO_ENABLE, CAN, CANIO_STATUS

CANIO_ENABLE--CAN Enable

Type	System Parameters
Description	**Enable or disable internal CAN master function.** When CANIO_ADDRESS is set as 32, default value of CAN_ENABLE is enable.
Grammar	CANIO_ENABLE = ON/OFF
Controller	General
Example	CANIO_ADDRESS = 32 'set master, CAN baudrate is 500KBPS CANIO_ENABLE = ON 'Open CAN master function. CANIO_ENABLE = OFF 'Close CAN master function.
Instructions	CANIO_ADDRESS

CANIO_STATUS--ZIO Expansion Status

Type	System Status
Description	**Get present IO expansion status, the returned value is ON or OFF (1 or 0).** Valid in controllers with firmware above 20140325.

Grammar	CANIO_STATUS(cardnum) cardnum: IO expansion NO. (got from the dial-up switches setting)
Controller	General

Example	**Example one:**
	?***CANIO_STATUS** 'print status of all IO expansion modules.
	
	Example two:
	If **CANIO_STATUS**(1) =0 **THEN**
	'Judge the connection status of IO module
	PRINT "IO Expansion 1 is not connected well"
	ENDIF

Instructions	CAN, CANIO_ENABLE, CANIO_ADDRESS

CANIO_INFO—CAN Expansion Information

Type	System status
Description	**Read current IO expansion information, then return parameter value.** ZMC4XX series controller with firmware version above 170715.
Grammar	CANIO_INFO(canid, isel [, moduleid]) canid: expansion module dial code ID isel: parameters that are read

0	Zmotion
1	DEVICE, device No.
2	
3	
4	
IO numbers	Description
10	the number of IN (all)
11	The number of OP (outputs)
12	The number of AIN (analog inputs)
13	The number of AOUT (analog outputs)
Add	
16	the number of modules
17	Type No. of sub module, and it must be with moduleid.
Reserve	
20	Submodule input
21	Submodule output
22	Submodule AIN
23	Submodule AOUT

For example: coupler connects to the first one expansion submodule, the address is 0, the second expansion submodule, the address is 1, and so on.

[how to modify analog AD/DA of ZMIO300-CAN/ZMOIO310-CAN

	expansion submodule] CANIO_INFO (canid, 17, moduleid) = range type canid: dial code ID of expansion module moduleid: ZMIO submodule information, extended submodules are numbered by coupler connection sequence starting from 0. For some isel, it needs to fill in this parameter. **Range Type**

Type No.	Module Type	Type No.	Module Type	Range
2	AD Module (Input Module)	10	AD Module (Input Module)	0-10V
3		11		-10-10V
4		12		4-20mA
5		13		0-20mA
6		14		0-5V
7		15		-5-5V

Controller	General
Example	**Example 1:** ?**CANIO_INFO** (1,1) ‘print device No. whose expansion module ID is 1 **Example 2:** Modify & Check ranges of the first and the second extended submodules’ AD / DA. CANIO_INFO(0,17,0)=3 ‘modify the AD range as -10-10V ?CANIO_INFO(0,17,0) ‘check the range CANIO_INFO(0,17,1)=11 ‘modify the DA range as -10-10V ?CANIO_INFO(0,17,1) ‘check the range
Instruction	CAN, CANIO_ENABLE, CANIO_STATUS CANIO_ADDRESS -- CAN 通讯设置

13.3 Self-defined Communication Instructions

GET#--Read String

Type	System Instruction
Description	**Get one byte from the channel when the communication mode is RAM mode or self-defined Ethernet mode, and save value into variable.**
Grammar	Grammar1: GET #PORT, VARIABLE Grammar2: GET #PORT, ARRAY[(startindex)] [,maxchares] Grammar3: charesget = GET #PORT, VARIABLE Grammar4: charesget = GET #PORT, ARRAY[(startindex)] [,maxchares]

	PORT: channel NO. VARIABLE: saved variable name Startindex: start address of array Maxchares: maximum bytes to save in array Grammar1/2: if no data was got, channel will jam, usually used in multi-task occasion. Grammar3/4: it will return bytes number that is read. Below version 20150522 only support grammar 1. UDP receive must use grammar 4, use array to receive, the array length should not be smaller than the UDP packet length at one time. UDP reads the entire packet at a time, and if the array length is insufficient, the excess will be discarded. In the UDP_SERVER mode, receive one packet, PORT_TARGET will be the sender automatically, so it can receive multi-salve data.
Controller	General

Example	**Example one:**
	DIM VAR1
	SETCOM(38400,8,1,0,0,0) 'open RAM mode.
	GET #0, VAR1 'get data from channel 0.
	PRINT VAR1 'print data
	
	Example two:
	DIM ARRAY1(101)
	SETCOM(38400,8,1,0,0,0) 'Open RAM mode.
	CHARES = **GET** #0, ARRAY1, 100 'get 100 bytes from channel 0.
	If CHARES > 0 THEN
	ARRAY1(CHARES) = 0 'set the last number as 0
	PRINT ARRAY1 'print string
	ENDIF

Instructions	PORT, PRINT #

OPEN # -- Open Custom Ethernet Communication

Type	System Instruction
Description	**Open Self-defined Ethernet Communication.** Valid in latest firmware version.
Grammar	OPEN #PORT, "mode",portnum[,ipaddress] port: communication channel, see PORT description, select defined net channel (ECUSTOM) mode: master or slave station, TCP_CLIENT -- slave station of TCP mode, TCP_SERVER – master station of TCP mode, UDP_CLIENT – slave station of UDP mode, UDP_SERVER –

	master station of UDP mode, TCP_EXCUTE – use TCP control directly. portnum: TCP port No. or UDP port No., master station is local port No., slave is the other side port No. ipaddress: the other side IP addrerss, character string, it should be filled when as slave TCP_EXCUTE means TCP control, that is, TCP sends BASIC commands, and it must be with \n end character. OPEN#, “TCP_EXECUTE”, portnum is to set “watch” for server side. And this remote port No. can’t set as 502. For other port numbers, it only makes two sides be consistent. UDP_SERVER must receive the other side data first, then send back data. (except use PORT_TARGET to define the other forcibly) The local port No. of UDP_CLIENT is random, it must send to the other side firstly, in this way, the other side could know the port No., and not assigned package will be discarded under this mode. UDP self-defined communication is valid in ZMC4XX series controller with firmware version above 20170628 and valid in XPLC series controller with firmware version above 20170702. “?*open” can print all OPEN port No. information.
Controller	General
Example	**Exapmle 1** OPEN #11, "TCP_SERVER", 10 'set as master OPEN #10, "TCP_CLIENT", 10,"192.168.1.112" 'set as slave **Example 2** OPEN #10, "UDP_SERVER", 1000 'UDP master OPEN #11, "UDP_CLIENT", 60000, "192.168.0.120" 'UDP salve **Example 3** OPEN #3, "UDP_EXECUTE", 500 'direct TCP control See “self-defined Ethernet Communication” for details.
Instructions	PORT, PORT_TARGET, SETCOM, PRINT #

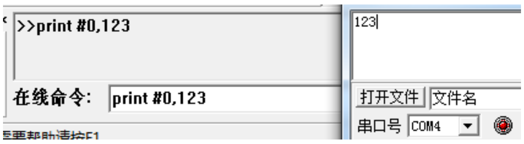
CLOSE # -- Close Self-defined Ethernet Communication

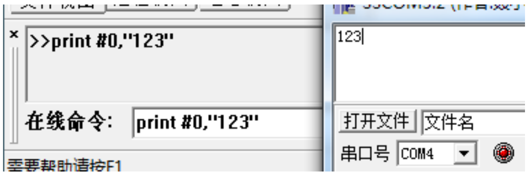
Type	System Instruction
Description	**Close Self-defined Ethernet Communication.** Valid in latest firmware version.
Grammar	CLOSE # ecustomnum ecustomnum: customized net port channel No.

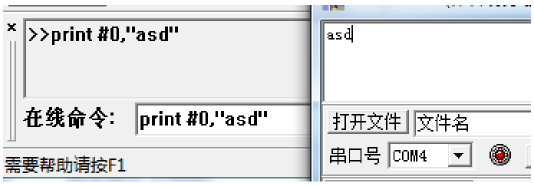
Controller	General
Example	CLOSE # 10 ‘close channel 10
Instructions	OPEN, PORT, PORT_TARGET, SETCOM, PRINT #

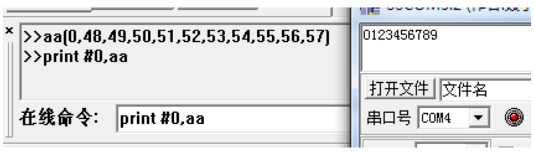
PRINT #--Output Character String

Type	System Instruction
Description	**Output character string in RAM or self-defined Ethernet communication mode, it will stop when meets 0.** One UDP package is sent when evert time it is called. PRINT # will send character string directly (“” will be deleted itself), no need to do ASCII code switch, only one data can be sent in one time, as follow:

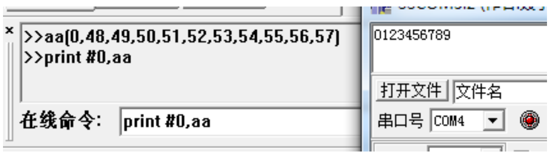


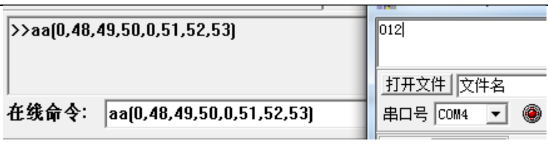




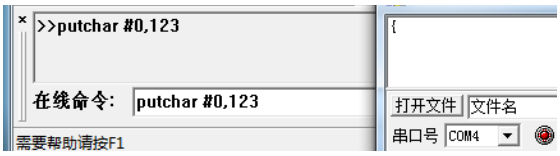
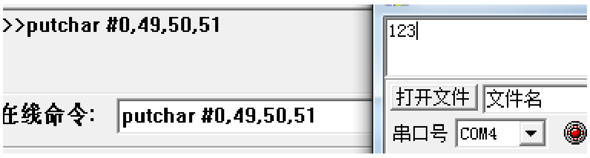
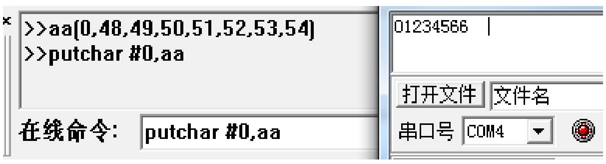


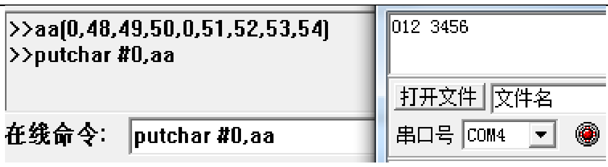
When use PRINT # to send array as ASCII code, it will stop when meets 0, as follow:



	
Grammar	PRINT #PORT, "character string" port: Channel NO.
Controller	General
Example	DIM VAR(10) 'define array VAR = "AAAA" 'assign value to array SETCOM(38400,8,1,0,0,0) 'open RAM mode **PRINT** #0,VAR 'output array through channel 0.
Instructions	PUTCHAR #, GET #, PORT, SETCOM

PUTCHAR#--Output Character

Type	System Instruction
Description	**Output character in RAM or self-defined Ethernet communication mode, it will stop when meets 0.** One UDP package is sent when evert time it is called. Date sent by PUTCHAR # is ASCII code, multi data should be divided by',' it can not send character string directly, as follow:    Array sent by PUTCHAR # is ASCII code, if meets 0 in array, then will output blank space, total array position will be sent, do ensure the array value is correct, as follow: 

	
Grammar	PUTCHAR #PORT,Character port: channel number PUTCHAR #PORT, ARRAY(INDEX, NUMES) port: channel number index: output start position. numes: output byte number, binary type.
Controller	General
Example	DIM VAR1, ARRAY1(10) 'define array VAR1 = $FE 'assign value to VAR1 ARRAY1 = "ABCDEFGHIJ" 'assign value to ARRAY1 SETCOM(38400,8,1,0,0,0) 'open RAW mode. **PUTCHAR** #0, VAR1 'output data of array1 through Channel 0. **PUTCHAR** #0, ARRAY1 'output data of array2 through Channel 0.
Instructions	PORT, SETCOM, GET #

PORT_TARGET—IP and Port NO. configuration

Type	Character string instruction
Description	**Configurate IP address and port NO. of the other side.** ZMC4XX series controller with firmware version above 20170628, XPLC series controller version above 20170702.
Grammar	Command grammar: PORT_TARGET(port))="ipaddress:portnum or PORT_TARGET(port))="ipaddress" port： channel NO. ipaddress： the other IP address portnum： port NO Return grammar： VAR=PORT_TARGET(port)) return IP address character string.
Controller	General
Example	PORT_TARGET="192.168.0.12:502" PORT_TARGET(port))="192.168.0.12" ?PORT_TARGET(port) DIM IPSTRING(100) IPSTRING = PORT_TARGET(port) ?IPSTRING
Instruction	OPEN #_CANIO_ADDRESS -- CAN 通讯设置

13.4 Print and Output Instructions

PRINT--Print Information

Type	Print Output Function
Description	**Print information in output window of ZDEVELOP.** **Additional Name:?** Print parameters value through format: *parameters name; Print special character string through format: * character string.
Grammar	PRINT expression, "string" or ? expression, "string" Expression: valid expression sentences *SET: print all parameters value *TASK: print task information. If task is normal, only output task status. It will output error task NO. and error line NO. when task appear errors. *MAX: print all specification parameters *FILE: print procedure file information *SETCOM: print present serial ports configuration information *BASE: print present BASE list (version above 140123 supports) *array name: print all elements of array, array should not be too long. *parameters name: print one parameter of all axes. ?*ETHERCAT: print EtherCAT bus connection status. ?*RTEX: print Rtex bus connection status ?*FRAME: print robotic parameters, firmware should be above 161022 ?*SLOT: print slot information(EtherCAT or Rtex) ?*PORT: print all communication ports
Controller	General
Example	**Example one:** Input online instructions >>**PRINT** 1+2 Output:3 **Example two:** Input online instructions >>**PRINT** *task 'print all tasks status. 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 three: Input online instructions >> ?*mpos Output:21872.400 0 0 0 0 0 0 0 0 0 0 0
Instructions	TRACE

ERRSWITCH--Information Output Setting

Type	System Parameters
Description	Debugging output switch, control output of TRACE, WARN and ERROR, judge whether debug output commands are actually output information.
Grammar	ERRSWITCH=switch switch; debugging output switch 0-all outputs instructions don't output. 1-only enable ERROR 2-enable WARN and ERROR 3-enable TRACE, WARN and ERROR 4-enable TRACE, WARN and ERROR, and all related motion monitoring instructions.
Controller	General
Example	**Example one:** **ERRSWITCH** = 3 'enable TRACE, WARN and ERROR **Example two:** **ERRSWITCH** = 0 'disable all outputs instructions TRACE DPOS(0) 'can not use trace output dpos of axis 0 ?DPOS(0) 'print is valid, print result is 0 **Example three:** **ERRSWITCH** = 4 'output all information >>MOVE(111) MOVE(111)AXIS(0)TASK(23) 'print present axes and tasks.
Instructions	TRACE, WARN, ERROR

TRACE--Print Information 2

Type	Print Output Function
Description	**Print information in output window of ZDEVELOP.** Controlled by ERRSWITCH setting.
Grammar	Same as PRINT
Controller	General

Example	ERRSWITCH = 3 'all trace functions are enable DPOS(0) =0 **TRACE** "DPOS(0) =" DPOS(0) Print result 0
Instructions	ERRSWITCH , PRINT, WARN, ERROR

WARN--Alarm Information

Type	Print Output Function
Description	**Print alarm information in output window of ZDEVELOP automatically.** Manual print is also available. Controlled by ERRSWITCH setting.
Grammar	print automatically when there is procedure alarm the same as instruction: Print when by Manual
Controller	General
Example	See example of TRACE as reference
Instructions	ERRSWITCH , PRINT, WARN, ERROR

ERROR--Error Information

Type	Print Output Function
Description	**Print error information in output window of Zdevelop automatically.** Manual print is also available. Controlled by ERRSWITCH setting.
Grammar	print automatically when there is procedure alarm the same as instruction: Print when by Manual
Controller	General
Example	See example of TRACE as reference
Instructions	ERRSWITCH , PRINT, WARN, ERROR

13.5 Channel Parameter Instruction

PORT--Channel NO.

Type	Channel Correction Subsidiary Instruction
Description	**When accesses channel parameters, it can select needed port NO.** Choose port NO. when some instructions need to enter related channel.
Grammar	PORT (portnum)

portnum: channel NO.

Different controllers support different channel numbers, check it through instruction "?*port". See example one.

ZMC00x series

Channel NO.	protocol
0	Serial port A
1	Serial port B

ZMC1-2xx series, support MODBUS-TCP when link with HMI.

Channel NO.	protocol
0	RS232 serial port
1	RS485 serial port
2	Ethernet, link 1. port NO.:502
3	Ethernet, link 2. port NO.:502
10	Self-defined net communication channel 1
11	Self-defined net communication channel 2

ZMC3xx series, with 3 serial ports.

Channel NO.	protocol
0	RS232 serial port
1	RS485 serial port
2	RS422 serial port
3	Ethernet, link 1.
4	Ethernet, link 2.
10	Self-defined net communication channel 1
11	Self-defined net communication channel 2

ZMC4xx series, with 2 serial ports.

Channel NO.	protocol
0	RS232 serial port
1	RS485 serial port
2	Ethernet, link 1.
3	Ethernet, link 2.
10	Self-defined net communication channel 1
11	Self-defined net communication channel 2
20	Channel for link between controllers, don't support PORT_STATUS

ECI series, only 1 serial port.

Channel NO.	protocol
0	RS232 serial port
1	Ethernet, link 1.
2	Ethernet, link 2.

Controller	General
Route	**See controller channels** >>?*port Port:0-COM. Port:1-COM. Port:2-ETH. Port:3-ETH. Port:4-ETH. Port:5-ETH. Port:6-ETH. Port:7-ETH. Port:10-ECUSTOM. Port:11-ECUSTOM. Port:12-ECUSTOM. Port:13-ECUSTOM. Port:14-ECUSTOM. Port:15-ECUSTOM. Port:20-CONNECT. Notes: COM: Serial port, ETH: Ethernet port, ECUSTOM: self-defined Ethernet communication port, CONNNECT: channels for controller connection.
Instructions	FILE_PORT, PORT_STATUS, PROTOCOL

PORT_STATUS--Channel Status

Type	Channel Parameters, only for read.
Description	**Return present channel status, serial port will always return 1.** When MODBUS_TCP connection is built, PORT_STATUS will become 1. If relevant link already be as slave, controller will search valid ports as MODBUS_TCP master link, procedure can not confirm the actual port that be called. This function is valid in controller with new firmware.
Grammar	VAR1 = PORT_STATUS(port) port: channel NO. Value / Meaning 0 / No connection 1 / Connected
Controller	General
Example	?**PORT_STATUS**(0) 'return channel status of serial 232, result is1

Instructions	PORT, SETCOM, ADDRESS

PORT_MODE--Channel Mode

Type	Channel Parameters
Description	**PORT Mode Configuration, it can add PORT type dynamically.**

Grammar	Command grammar: PORT_MODE(portnum, mode [,"comname"])
	portnum: PORT No.
	mode: the mode to be set must be a free PORT (ZPORT_MODE_NOTUSED). And it is recommended to set it at the front of the program. The supported setting types:
	ZPORT_MODE_COMUSB = 2 //serial port expanded by USB
	ZPORT_MODE_ETH = 10 //standard ethernet connection
	ZPORT_MODE_ETHCUSTOM = 11 //ethernet custom communication
	"The current mode setting can be checked by ?*PORT or PORT_MODE function".
	comname: for serial ports, it means the name of the serial port. Under Linux, it is generally "/dev/ttySn" or "/dev/ttyUSBn". And the list of serial ports can be checked through the linux command ls /dev, and the name under windows is "COMn".
	
	Function grammar: mode= PORT_MODE(portnum)
	portnum: PORT No.
	return value:
	ZPORT_MODE_NOTEXIST = -1, // exceeds the maximum PORT number
	ZPORT_MODE_NOTUSED = 0, // not used, can be used to set
	ZPORT_MODE_COM = 1, // hardware fixed serial port
	ZPORT_MODE_COMUSB = 2, // USB extended serial port
	ZPORT_MODE_LOCAL = 9, // LOCAL direct channel
	ZPORT_MODE_ETH = 10, // standard network connection
	ZPORT_MODE_ETHCUSTOM = 11, // network custom communication
	ZPORT_MODE_ETHICONNET = 12, // interconnection channel

Controller	Valid in ZMC5XX series controllers with firmware version above 20200302.
Example	Set custom net port: PORT_mode (22,11) Set standard net port: PORT_mode (22,10) Set USB net port: PORT_mode (22,2,"/dev/ttyUSB0")
Instructions	PORT

FILE_PORT--Present Channel File NO.

Type	Channel Parameters
Description	**Return or set default file NO. of present channel.** Access to module variables or array of relevant file after setting. This parameter will be called in Zdevelop automatically, no need to modify it in Zdevelop.
Grammar	VAR1 = FILE_PORT, FILE_PORT = filenum
Controller	General
Example	>>FILE_PORT = 0 ' FILE_PORT parameters of present channel
Instructions	PORT

PROTOCOL--Channel Communication Protocol

<table>
<tr><td>Type</td><td>Channel parameter, only for read</td></tr>
<tr><td>Description</td><td>Return communication protocol in present channel.</td></tr>
<tr><td>Grammar</td><td>VAR1 = PROTOCOL(port)
port: channel NO.
Returned Value
<table>
<tr><td>value</td><td>protocol</td></tr>
<tr><td>0</td><td>RAW data format, no protocol.</td></tr>
<tr><td>3</td><td>MODBUS protocol, controller as slave. (default)</td></tr>
<tr><td>14</td><td>MODBUS protocol, controller as master.</td></tr>
<tr><td>15</td><td>Direct command mode.</td></tr>
</table></td></tr>
<tr><td>Controller</td><td>General</td></tr>
<tr><td>Instructions</td><td>PORT, SETCOM, ADDRESS</td></tr>
</table>

ETH_MODE—Net Port Mode Settings

Type	Channel parameter.
Description	**Net port mode configuration, array type, and each net port is set independently.**
Grammar	ETH_MODE(isel) = imode isel: 0-the first net port, 1-the second net port (EtherCAT) imode: 0- non eth mode, traditional mode, good compatibility, anti-interference is a little bad. 1- eth mode, extremely good anti-interference, when there are multiple links, anti-drops DLL on PC should be used together, and only one standard DLL only can be connected (less touch screens or EtherCAT drives can't support this

	mode). 2- Auto mode (new firmware default value), if net is not connected, it will switch between mode 0 to mode 1 automatically.
Controller	General
Instructions	PORT

SEND_AUTOUP—Active Report

Type	Channel parameter.
Description	**Set the content that is reported actively by net port.**
Grammar	SEND_AUTOUP (port, typecode, string) port: PORT number that supports active reporting. -1 means automatically select the Ethernet link that supports active reporting. After ZMC_SetAutoUpCallBack is called on the PC, the corresponding link automatically supports active reporting typecode: type code string: character string, valid character strings expressions are OK.
Controller	General
Example	SEND_AUTOUP(-1,100,"111211")
Instructions	PORT

SEND_AUTOUP2—Active Report 2

Type	Channel parameter.
Description	**Set the content that is reported actively by net port, send TABLE content.**
Grammar	SEND_AUTOUP2 (port, typecode, tableindex, length) port: PORT number that supports active reporting. -1 means automatically select the Ethernet link that supports active reporting. After ZMC_SetAutoUpCallBack is called on the PC, the corresponding link automatically supports active reporting typecode: type code tableindex: starting number of table position length: the number of sent bytes in binary system, one table element gets one byte.
Controller	General
Example	TABLE = “1234” SEND_AUTOUP2(-1,200,0,4)
Instructions	PORT

IFAUTOUP_PORT—Check Active Reporting Port

Type	Channel parameter.
Description	**Check PORT that supports active reporting.** Check PORT ports that are printed by “?*port” and belong to eth type, 0 – unsupported, 1 – support.
Grammar	VALUE = IFAUTOUP_PORT
Controller	General
Example	Check all net ports channels. ?*IFAUTOUP_PORT
Instructions	PORT

13.6 MODBUS Communication Instruction

MODBUS_BIT--Bit Register

Type	MODBUS bit register
Description	**Modify or read BIT register, Boolean type, which is called 0x register in HMI.** Note: through some particular Modbus_0x registers on HMI, IO status can be read and set directly, in this situation, HMI read the original status of IO, no influence from INVERT_IN. Register (zero based) / Meaning: 10000- : Input IN, each input takes 1 register. 20000- : Output OP, each output takes 1 register. 30000- : S register of PLC, each takes 1 register.
Grammar	MODBUS_BIT(first,[last]) = value first bit register NO.,start from 0. last bit register NO., start from 0.
Controller	General
Example	DIM VAR **MODBUS_BIT**(100) =1 'assign bit100 as 1. VAR = **MODBUS_BIT**(100) 'assign bit100 to variable VAR.

Nested table in Description:

Register (zero based)	Meaning
10000-	Input IN, each input takes 1 register.
20000-	Output OP, each output takes 1 register.
30000-	S register of PLC, each takes 1 register.

MODBUS_IEEE--Word Register-32bits float

Type	MODBUS Word Register

Description	**Modify or read word register, 32 bits float, which is called 4x register in HMI.** Zmotion motion controller contains particular MODBUS word register, which will take possession of 2 register addresses. Controller will call Modbus word register as default selection, if need to modify it, do set the seventh parameter of SETCOM. Note: through particular Modbus_4x registers, some controller status can be read directly from HMI. <table><tr><td>Register (zero based)</td><td>Type</td><td>Meaning</td></tr><tr><td>10000-</td><td>IEEE</td><td>read DPOS, each axis takes 2 registers.</td></tr><tr><td>11000-</td><td>IEEE</td><td>read MPOS, each axis takes 2 registers.</td></tr><tr><td>12000-</td><td>IEEE</td><td>read VP_SPEED, each axis takes 2 registers.</td></tr></table>
Grammar	MODBUS_IEEE (regnum) = value regnum register NO., start from 0.
Controller	General
Example	DIM VAR **MODBUS_IEEE**(100) = 100.10 'assign ieee100 as 100.10 VAR = **MODEBUS_IEEE**(100) 'assign ieee100 to VAR
Instructions	MODBUS_REG, MODBUS_LONG, MODBUS_STRING

MODBUS_LONG--Word Register-32 bits integer

Type	MODBUS Word Register
Description	**Modify or read word register, 32 bits integer, which is called 4x register in HMI.** Zmotion motion controller contains particular MODBUS word register, which will take possession of 2 register addresses. Controller will call Modbus word register as default selection, if need to modify it, do set the seventh parameter of SETCOM.
Grammar	MODBUS_LONG(regnum) = value regnum: register NO., starts from 0.
Controller	General
Example	DIM VAR **MODBUS_LONG** (100) = 100 'assign long100 as 100 VAR = **MODEBUS_LONG**(100) 'assign long100 to VAR.
Instructions	MODBUS_REG, MODBUS_IEEE, MODBUS_STRING

MODBUS_REG--Word Register-16 bits integer

Type	MODBUS Word Register

Description	**Modify or read word register, 16 bits integer, which is called 4x register in HMI.** Zmotion motion controller contains particular MODBUS word register. Register number will differ as per controller type. Controller will call Modbus word register as default selection, if need to modify it, do set the seventh parameter of SETCOM. Note: through particular Modbus_4x registers, some controller status can be read directly from HMI. <table><tr><td>Register (zero based)</td><td>Type</td><td>Meaning</td></tr><tr><td>13000-</td><td>16</td><td>Read DA</td></tr><tr><td>14000-</td><td>16</td><td>Read AD</td></tr></table>
Grammar	MODBUS_REG(regnum) = value regnum register NO., starts from 0
Controller	General
Example	DIM VAR **MODBUS_REG**(100) = 100 'assign reg100 as 100. VAR = **MODBUS_REG**(100) 'assign reg100 to VAR.
Instructions	MODBUS_IEEE, MODBUS_LONG, MODBUS_STRING

MODBUS_STRING--Word Register-Byte

Type	MODBUS Word Register, character string function
Description	**Read character string in MODBUS register according bytes. It is called 4x register in HMI.**
Grammar	MODBUS_STRING(index, chares) index: MODBUS register start NO., starts from 0, register number will differ as per controller type. chares: total character string number to read.
Controller	General
Example	DIM ARR(8) **MODBUS_STRING** (0,8) = "abc" 'save bytes from string 0, all 8 bytes. print **MODBUS_STRING**(0,8) 'print character string, result is abc ARR = **MODBUS_STRING**(0,8) 'assign character string to array.
Instructions	MODBUS_REG, MODBUS_LONG, MODBUS_IEEE

MODBUSM_DES--Modbus Communication Connection

Type	Communication Instructions
Description	**Set or read the Modbus value of slave station from master station.** When there is communication waiting, it will only block the present task, no influence on other tasks.

	When use 485 serial port to do communication with multi-device, it can add wait or delay, wait until the former device succeeded in communicating, then connect to next device to avoid communication failure. Note: don’t write and read multiple MODBUS slave stations at the same moment, especially when there is multi-task, please operate independently.
Grammar	MODBUSM_DES(address[,port], [,time], [,resendset] ADDRESS1 = MODBUSM_DES([port]) address: modbus protocol NO. of slave station port: port NO. of present master station. timer: message timeout setting, default value is 1000ms resendset: timeout message resend setting, 0-not to resend, 1-resend SEND instruction, 2-resend SEND and MODBUSM instructions. Resend message of MODBUSM, slave controller may receive message twice, and scan register twice. Resend SEND message, there is no influence on controllers. Significant symbolic variable can be modified through SEND.
Controller	General

Example	**Example one: multi-master to multi-slave.**
	SETCOM(38400,8,1,0,0,14,2,1000) 'set serial port 0 as modbus master, communication waiting time is 1 second.
	SETCOM(38400,8,1,0,1,14,2,1000) 'set serial port 1 as modbus master, communication waiting time is 1 second.
	
	WHILE 1
	IF IN(0)=1 THEN 'use serial port 0 when IN0 high electric level
	IF IN(1)=1 THEN
	MODBUSM_DES(1,0) 'communicate with slave station port No.1, when IN1 high electric level
	MODBUSM_REGSET(0,10,0) 'copy local register to slave station.
	MODBUSM_REGGET(20,10,20)
	'copy register value of salve station into local station.
	WAIT UNTIL MODBUSM_STATE <> 1 'wait until message ends.
	
	ELSE
	MODBUSM_DES(2,0) 'communicate with slave station port No.2 when IN1 low electric level
	MODBUSM_REGSET(30,10,30)
	'copy local register to slave station.
	MODBUSM_REGGET(40,10,40)
	'copy register value of salve station into local station.
	WAIT UNTIL MODBUSM_STATE <> 1 'wait until message ends.

	ENDIF
	?"channel 0 status=", MODBUSM_STATE 'print communication status
	
	ELSE 'use serial port 1 when in(0) is low electric level
	IF IN(1)=1 THEN
	MODBUSM_DES(1,1) 'communicate with slave station, port No.1 when IN1 high electric level
	MODBUSM_REGSET(50,10,0)
	'copy local register to slave station.
	MODBUSM_REGGET(60,10,60)
	'copy register value of salve station into local station.
	WAIT UNTIL MODBUSM_STATE <> 1 'wait until message ends.
	ELSE
	MODBUSM_DES(2,1) 'communicate with slave station, port No.2, when IN1 low electric level
	MODBUSM_REGSET(70,10,70)
	'copy local register to slave station.
	MODBUSM_REGGET(80,10,80)
	'copy register value of salve station into local station.
	WAIT UNTIL MODBUSM_STATE <> 1 'wait until message ends.
	ENDIF
	?"channel1 status=", MODBUSM_STATE 'print communication status.
	ENDIF
	WEND

Instructions	SETCOM, PROTOCOL, PORT, MODBUSM_DES2

MODBUSM_DES2--Ethernet Communication

Type	Communication Instructions
Description	**Ethernet communication between controllers, it also can be as MODBUS_TCP master communication.** ?*PORT 'print present available communication channels.

```
>>?*port
Port:0-COM.
Port:1-COM.
Port:2-ETH.
Port:3-ETH.
Port:4-ETH.
Port:5-ETH.
Port:6-ETH.
Port:7-ETH.
Port:10-ECUSTOM.
Port:11-ECUSTOM.
Port:12-ECUSTOM.
Port:13-ECUSTOM.
Port:14-ECUSTOM.
Port:15-ECUSTOM.
Port:20-CONNECT.
```

As MODBUS_TCP maser:

Choose one ETH type port as MODBUS_TCP communication channel (the first and last ETH ports are not recommended).

If the chosen port is in possess of slave station, then controller will choose a valid ETH port as MODBUS_TCP master automatically.

As Communication between controllers:

Choose CONNECT type port as communication channel between controllers.

Note: don't write and read multiple MODBUS slave stations at the same moment, especially when there is multi-task, please operate independently.

Grammar

MODBUSM_DES2 (id, port, "desipaddress", [timer], [resendset], [destport502])

id: ID of salve station, default value is 1.

port: support two modes, ?*PORT to confirm channel No. and mode.

For ETH, as MODBUS_TCP master channel

For CONNECT, as connection channel between controllers

desipaddress: IP address of salve station, it is character string.

timer: message delay time setting, default value is 1000ms.

resendset: timeout message resend setting, 0-not to resend, 1-resend SEND instruction, 2-resend SEND and MODBUSM instructions.

destport502: port No., default is 502.

Resend message of MODBUSM, slave controller may receive message twice, and scan register twice.

Resend SEND message, there is no influence on controllers.

Significant symbolic variable can be modified through SEND.

Controller

ZMC4xx series with firmware version above 20170117.

Example

Example one: build MODBUS master communication

No need to consider message loss in MODBUS_TCP master communication mode.

	MODBUSM_des2(1,4,"192.168.0.12")
	'communicate with slave station according to station No. and IP, use controller channel 4, confirm port channel through "?*port"
	WHILE 1
	LASTTICK = TICKS
	FOR i =0 TO 9999
	MODBUS_REG(0)=i
	MODBUSM_REGEST(0,10,0) 'set slave register
	MODBUSM_REGEST(0) = 99
	MODBUSM_REGEST(0,10,0) 'read slave register.
	
	IF MODBUS_REG(0) <> I THEN
	?"REG(0)=" MODBUS_REG(0),"state=" MODBUSM_STATE
	'print error appeared in which communication
	ENDIF
	NEXT
	?LASTTICK-TICK 'print communication time
	WEND
	END
	
	Example two: build communication between controllers.
	When Ethernet environment is not good, there is a small probability to lose message in interconnection communication.
	MODBUSM_des2($fe,20,"192.168.0.25" ,10)
	'controller slave station port is fe, controller master port is 20, confirm timeout time is set as 10ms through "?*port".
	MODBUSM_REGSET(0,10,0) 'copy local register value to slave station.
	WAIT UNTIL MODBUSM_STATE <> 1 'wait until message ends.
	IF MODBUSM_STATE<>0 TEHN
	MODBUSM_REGSET(0,10,0) 'resend if there is error.
	MODBUSM_REGGET(20,10,20) 'copy salve register value into local station.
	ENDIF
	WAIT UNTIL MODBUSM_STATE <> 1 'wait until message ends.
	IF MODBUSM_STATE<>0 then MODBUSM_REGGET(20,10,20)
	'resend if there is error.
	ENDIF
	END

Instructions	ADDRESS, PORT

MODBUSM_STATE--modbus Communication Status

Type	Communication Status

Description	**MODBUS communication status of master station.** Value / Description: 0 — Normal 1 — Waiting for response 2 — Waiting time out 3 — Response error
Grammar	VAR1 = MODBUSM_STATE Present modbus communication status of master station.
Controller	General

Example	SETCOM(38400,8,2,0,1,14,2,1000)
	'485serial port as modbus master station, message time out is 1second.
	MODBUSM_DES(1,1) 'communicate with slave station, port is 1.
	MODBUSM_REGGET(0,10,0) 'get register value of slave station.
	WAIT UNTIL **MODBUSM_STATE** <> 1
	'wait until message ends, maximum 1 second as message time out period, become relevant value after time out or getting message.
	?**MODBUSM_STATE** 'print communication result.
	IF **MODBUSM_STATE**=0 THEN
	?" Normal "
	ELSEIF **MODBUSM_STATE**=2 THEN
	?" Waiting time out "
	ELSEIF **MODBUSM_STATE**=3 THEN
	?" Response error "
	ENDIF

Instructions	PROTOCOL, PORT, SETCOM

MODBUSM_REGSET—Set Save Modbus Value

Type	Communication Instructions
Description	**Assign local save Modbus value to slave station.** **Relevant standard protocol function code is 06 or 16: write save register.**
Grammar	MODBUSM_REGSET (startreg, num, local_reg) startreg modbus start NO. of slave station, starting from 0. num the number of register. local_reg local MODBUS start NO. to get value, starting from 0.
Controller	General
Example	See example one in MODBUSM_REGGET
Instructions	ADDRESS, PROTOCOL, PORT, SETCOM

MODBUSM_REGGET--Read Save Modbus Value

Type	Communication Instructions
Description	**Assign save Modbus value of slave station to local station.** **Relevant standard protocol function code is 03, read save register.**
Grammar	MODBUSM_REGGET (startreg, num, local_reg) startreg modbus start NO. of slave station, starts from 0. num the number of register. local_reg local MODBUS start NO. to get value
Controller	General

Example	**Read absolute encoder of delta**
	GLOBAL DIM flag_abs 'encoder reads correct symbols
	flag_abs = 0
	GLOBAL DIM total_pul 'read the number of total pulses
	SETCOM(38400,8,2,0,1,14) 'set serial 485 as MODBUS master station, baud rate is 38400
	MODBUSM_DES(1,1) 'set serial 485 as salve station, port is 1.p3-00
	MODBUS_LONG(300) = 2 'transfer data with 300,301
	MODBUSM_REGEST(98,2,300) 'set P0-49 = 2, update parameters.
	MODBUS_REGGET(98,2,300)
	TICKS = 1000
	
	WHILE (MODBUS_LONG(300) AND TICKS > 0)
	'wait until P0-49 become 0 or time out after 1 second, which means update succeeded or failed
	MODBUS_REGGET(98,2,300)
	WEND
	
	IF TICKS < 0 THEN
	PRINT "servo upgrade failed"
	flag_abs = 1
	RETURN
	ENDIF
	
	MODBUSM_REGGET(100,6,310)
	IF MODBUS_LONG(310) = 0 THEN 'encoder status is normal
	flag_abs = 0
	total_pul = modbus_long(314)
	ELSE
	PRINT "encoder error"
	flag_abs = 2
	ENDIF
	
	IF flag_abs = 0 TEHN 'correct
	dpos(0) = -total_pul / units(0) 'measured pulse amount is negative, here

	convert it into positive. ENDIF END
Instructions	ADDRESS, PROTOCOL, PORT,, SETCOM

MODBUSM_3XGET--Read Input Register

Type	Communication Instructions
Description	**Assign input Modbus value of slave station to local station.** **Relevant standard protocol function code is 04, read input register.**
Grammar	MODBUSM_3XGET (startreg, num, local_reg) startreg modbus start NO. of slave station, starts from 0. num the number of register. local_reg local MODBUS start NO. to get value
Controller	General
Example	MODBUSM_3XGET(0,9,0) 'copy value of slave register 0-9 to local register 0-9.
Instructions	ADDRESS, PROTOCOL, PORT, SETCOM

MODBUSM_BITSET--Write Coil

Type	Communication Instructions
Description	**Assign local MODBUS bit register value to slave station.** **Relevant standard protocol function code is 05 or 15, write coil..**
Grammar	MODBUSM_BITSET (startreg, num, local_reg) startreg modbus start NO. of slave station, starts from 0. num the number of register. local_reg local MODBUS start NO. to get value
Controller	General
Example	**MODBUSM_BITSET** (0,10,0) 'copy value of local register 0-9 to slave register 0-9.
Instructions	ADDRESS, PROTOCOL, PORT, SETCOM

MODBUSM_BITGET--Read Coil

Type	Communication Instructions
Description	**Assign MODBUS bit register value of slave station to local station.** **Relevant standard protocol function code is 01: read coil.**
Grammar	MODBUSM_BITGET (startreg, num, local_reg) startreg modbus start NO. of slave station, starts from 0. num the number of register. local_reg local MODBUS start NO. to get value

Controller	General
Example	MODBUSM_BITGET (0,10,0) ' copy slave modbus-bit value of register 0-9 to local register:0-9
Instructions	ADDRESS, PROTOCOL, PORT, SETCOM

MODBUSM_1XGET--Read Isolated Inputs

Type	Communication Instructions
Description	**Assign MODBUS bit register value of slave station to local station. Relevant standard protocol function code is 02: read isolated inputs0.**
Grammar	MODBUSM_1XGET (startreg, num, local_reg) startreg modbus start NO. of slave station, starts from 0. num the number of register. local_reg local MODBUS start NO. to get value
Controller	General
Example	MODBUSM_1XGET (0,10,0) ' copy slave modbus-bit value of register 0-9 to local register:0-9
Instructions	ADDRESS, PROTOCOL, PORT, SETCOM

13.7 Direct Command Instructions between Controllers

SEND_RESULT—Read send Result

Type	Communication instruction
Description	**Read send instruction result.** Return value: 0-succeed, others: errors, including error code returned from controller. Resend message of MODBUSM, slave controller may receive message twice, and scan register twice. Resend SEND message, there is no influence on controllers. Significant symbolic variable can be modified through SEND.
Grammar	VAL=SEND_RESULT
Controller	Valid in ZMC4XX series controller, version 20170618 support.
Instruction	SEND_CMD

SEND_CMD—send Command

Type	Communication instruction
Description	**Master controller sends ZMC_DIRECTCOMMAND instruction to slave controller, check result in SEND_RESULT.**

	Send BASIC content: cmdstring(parameter list) Resend message of MODBUSM, slave controller may receive message twice, and scan register twice. Resend SEND message, there is no influence on controllers. Significant symbolic variable can be modified through SEND.
Grammar	SEND_CMD(cmdstring, selectable parameter list) cmdstring: command character string selectable parameter list: numbers can change, no need to add bracket without parameter
Controller	Valid in ZMC4XX series controller, version 20170618 support
Example	SEND_CMD("MOVE",DIS1,DIS2,DIS3) SEND_CMD("MOVEABS",DIS1)
Instruction	SEND_RESULT

SEND_CMDAXIS—send Command

Type	Communication instruction
Description	**Master controller sends ZMC_DIRECTCOMMAND instruction to slave controller, check result in SEND_RESULT.** Send BASIC content: cmdstring(parameter list), AXIS(iaxis) Resend message of MODBUSM, slave controller may receive message twice, and scan register twice. Resend SEND message, there is no influence on controllers. Significant symbolic variable can be modified through SEND.
Grammar	SEND_CMDAXIS (cmdstring,iaxis, selectable parameter list) cmdstring: command character string iaxis: the axis numebr selectable parameter list: numbers can change, no need to add bracket without parameter
Controller	Valid in ZMC4XX series controller, version 20170618 support.
Example	SEND_CMDAXIS("MOVE",IAXIS,DIS1)
instruction	SEND_RESULT, SEND_CMD

SEND_ASSIGN—send Command

Type	Communication instruction
Description	**Master controller sends ZMC_DIRECTCOMMAND instruction to slave controller, check result in SEND_RESULT.** Send BASIC content: cmdstring(parameter list)=value Resend message of MODBUSM, slave controller may receive message

	twice, and scan register twice. Resend SEND message, there is no influence on controllers. Significant symbolic variable can be modified through SEND.
Grammar	SEND_ASSIGN (cmdstring,value, selectable parameter list 可选参数列表) cmdstring: command character string value: assigned content selectable parameter list: numbers can change, no need to add bracket without parameter
Controller	Valid in ZMC4XX series controller, version 20170618 support
Example	SEND_ASSIGN("DPOS",0,0) 'generate DPOS(0)=0 SEND_ASSIGN("DPOS(1)",0) 'generate DPOS(1)=0
instruction	SEND_RESULT

SEND_QUERY—send Command

Type	Communication instruction
Description	**Master controller sends ZMC_DIRECTCOMMAND instruction to slave controller, check result in SEND_RESULT.** Send BASIC content: cmdstring(parameter list) There is no need to add bracket when without parameters. Received content is filled in TABLE according to SEND_QUERYSET configuration. Resend message of MODBUSM, slave controller may receive message twice, and scan register twice. Resend SEND message, there is no influence on controllers. Significant symbolic variable can be modified through SEND.
Grammar	SEND_QUERY (cmdstring, selectable parameter list) cmdstring: command character string selectable parameter list: numbers can change, no need to add bracket without parameter
Controller	Valid in ZMC4XX controller, version 20170618 support
Example	SEND_QUERYSET(0,1) SEND_QUERY("?dpos",0) 'table(0) save DPOS(0) content SEND_QUERY("?REMAIN_BUFFER(1)AXIS(0)",0) 'send character string content (“?REMAIN_BUFFER(1) AXIS(0)”) SEND_QUERYSET(0,2) SEND_QUERY("?dpos(0),dpos(1)") 'return two data from controller
Instruction	SEND_RESULT, SEND_QUERYSET

SEND_QUERTSET—send Command

Type	Communication instruction
Description	**Master controller sends ZMC_DIRECTCOMMAND instruction to slave controller, check result in SEND_RESULT.** Send BASIC content: cmdstring(parameter list) Resend message of MODBUSM, slave controller may receive message twice, and scan register twice. Resend SEND message, there is no influence on controllers. Significant symbolic variable can be modified through SEND.
Grammar	SEND_QUERYSET (dtindex, dtnumes) dtindex: TABLE NO. to save received content. dtnumes: maximum TABLE numbers for save received content.
Controller	Valid in ZMC4XX series controller, version 20170618 support
Example	SEND_QUERYSET (0,1) SEND_QUERYSET (0,1)
Instruction	SEND_RESULT, SEND_QUERY

13.8 Send Instructions bewteen File Connection of Controllers

SEND_ZAR—USB Drive operation

Type	Communication instruction
Description	**Update slave controller procedure through master controller USB Drive, check the result from MODBUSM_STATE.** Resend message of MODBUSM, slave controller may receive message twice, and scan register twice. Resend SEND message, there is no influence on controllers. Significant symbolic variable can be modified through SEND.
Grammar	SEND_ZAR("ufilename") ufilename: filename of USB Drive, it supports array of character and other character string types.
Controller	Valid in ZMC4XX series controller, version 20170618 support.
Example	SEND_ZAR("1.ZAR")
Instruction	MODBUSM_STATE, SEND_PERCENT

SEND_FALSH—Data copy

Type	Communication instruction
Description	**Master controller USB Drive and slave controller FLASH copy each other, ckeck result from MODBUSM_STATE.** Resend message of MODBUSM, slave controller may receive message twice, and scan register twice. Resend SEND message, there is no influence on controllers. Significant symbolic variable can be modified through SEND.
Grammar	SEND_FLASH (dir,uid,flashid) dir: 1-copy USB Drive to controller FLASH, 0-copy controller FLASH to USB Drive uid: file NO. of USB Drive, same rule as U_WRITE flashid: FLASH NO. of controller
Controller	Valid in ZMC4XX series controller, version 20170618 support
Example	SEND_FLASH (1,1,1)
Instruction	MODBUSM_STATE, SEND_PERCENT

SEND_FILE—Copy USB Drive data

Type	Communication instruction
Description	**Master controller disk and slave controller FLASH copy each other, which only supports BIN file and Z3P file, there will return to fail if controller doesn't support file function.** Resend message of MODBUSM, slave controller may receive message twice, and scan register twice. Resend SEND message, there is no influence on controllers. Significant symbolic variable can be modified through SEND.
Grammar	SEND_FLASH (dir,ufile,controlfile) dir: 1-copy USB Drive to controller FLASH, 0-copy controller FLASH to USB Drive ufile: file name of USB Drive contrfile: file name of controller
Controller	Valid in ZMC4XX series controller, version 20170618 support
Example	SEND_FILE(1,"1.bin","1.bin")
Instruction	MODBUSM_STATE, SEND_PERCENT

SEND_IFLASH—Copy flash Data

Type	Communication instruction

Description	**Master controller disk and slave controller FLASH copy each other, the result is checked through MODBUS_STATE.** Resend message of MODBUSM, slave controller may receive message twice, and scan register twice. Resend SEND message, there is no influence on controllers. Significant symbolic variable can be modified through SEND.
Grammar	SEND_IFLASH (dir,id,flashid) dir: 1-copy USB Drive to controller FLASH, 0-copy controller FLASH to USB Drive id: FLASH number of main controller flashid: FLASH number of controller
Controller	Valid in ZMC4XX series controller, version 20170618 support
Example	SEND_FLASH (1,1,1)
Instruction	MODBUSM_STATE, SEND_PERCENT

SEND_PERCENT—Check Instruction Process

Type	Communication insrtuction
Description	**Return percent of instructions that need long time to finish, like SEND, this can be used to show process, 0-100.** Resend message of MODBUSM, slave controller may receive message twice, and scan register twice. Resend SEND message, there is no influence on controllers. Significant symbolic variable can be modified through SEND.
Grammar	percent = SEND_PERCENT ()
Controller	Valid in ZMC4XX series controller, version 20170618 support.
Instruction	SEND_ZAR, SEND_FLASH, SEND_FILE

SEND_CONTROL—Check Controller Type

Type	USB Drive function
Description	**Check relevant controller type to ZAR procedure file of master controller USB Drive, set it in the Zdevelop, avoid mixing master controller and slave controller.**
Grammar	id = ZAR_CONTROL ("ufilename") ufilename: file name of USB Drive, ZAR file
Controller	Valid in ZMC4XX series controller, version 20170618 support.
Example	id = ZAR_CONTROL("1.ZAR") IF(id/3000=3) Then 'ZHD series PRINT "zhd program"

	ENDIF
Instruction	SEND_ZAR, CONTROL

Chapter XIV Instructions Related to System

All date, time parameters or instructions can't be modified after LOCK.

14.1 Controller Encryption Instructions

APP_PASS-- Password

Type	System Instruction
Description	**Controller APP password.** This password can be used to verify ZAR file to be downloaded, if password doesn't match, then ZAR can not be loaded into controller. APP_PASS can not be modified after LOCK. APP_PASS is encrypted through irreversible algorithm, once forgotten, it can not recover any more.
Grammar	APP_PASS(pass) pass: alphabet or number or special sign such as "_", total number can not exceed 16 characters. And it can not set as variable or expression, otherwise, variable or expression name will be regarded as password.
Controller	General
Example	**APP_PASS**(Zmotion)
Instructions	LOCK

LOCK--Lock Controller

Type	System Instruction
Description	**Lock controller, no operation is allowed to controller after lock.** ZPJ project can be modified on PC, but it can not be loaded to controller, but generated zar file still can be loaded into controller. Enumeration operation will be forbidden in controller, such as, print all array data, but specific value can be printed. APP_PASS is encrypted through Irreversible algorithm, once forgotten, it can not recover any more. And pass can not set as variable or expression, otherwise, variable or expression name will be regarded as password.
Grammar	LOCK (pass) pass: alphabet or number or special sign such as "_", total number can not exceed 16 characters
Controller	General

Example	**LOCK**(passwd) 'lock controller, password is passwd
Instructions	UNLOCK

UNLOCK--Unlock Controller

Type	System Instruction
Description	**Unlock Controller.** LOCK password is encrypted through Irreversible algorithm, once forgotten, it can not recover any more.
Grammar	UNLOCK (pass) pass password when used LOCK to lock controller.
Controller	General
Instructions	LOCK

14.2 System Time Instructions

DATE--System Date

Type	System Parameters
Description	**Set system date, it supports power-failure saving, or return number of days since 1/1/2000.** Date can not be modified in simulator.
Grammar	DATE=DD:MM:YYYY or DD:MM:YY
Controller	General
Example	DATE=27:2:13 Online command input >>PRINT DATE Output:4806 It is 2013:2:27 – 2000:1:1 = 4806
Instructions	DATE$, RTC_DATE

DATE$--System Date 2

Type	String Functions
Description	**String function, return date set by DATE in format: DD:MM:YYYY.**
Grammar	DATE$
Controller	General
Example	DATE=27:2:13 Online command input >>PRINT DATE$

	Output: 27:02:2013
Instructions	DATE, RTC_DATE

DAY--System Week

Type	System Parameters
Description	**Set week time of system clock, 0-6, 0 indicates Sunday, it supports power-failure saving** Date can not be modified in simulator. DAY will not change as per DATE or RTC_DATE, they are independent to each other.
Grammar	VAR=DAY, DAY=expression
Controller	General
Example	**Example 1:** DAY = 3 **Example 2:** Online command input >>PRINT DAY Output: 3
Instructions	DAY$

DAY$--System Week 2

Type	String Functions
Description	**String Functions, return week set by DAY.** DAY$ will not change as per DATE or RTC_DATE, they are independent to each other.
Grammar	DAY$
Controller	General
Example	Online command input >>PRINT **DAY$** Output: Wednesday
Instructions	DAY

RTC_DATE--System Date

Type	System Parameters
Description	**Set or get system date, it supports power-failure saving, time starts from 1/1/2000.** The format is not the same as instruction DATE. Return value is integer. Date can not be modified in simulator.

Grammar	RTC_DATE = YYYYMMDD or YYMMDD VAR = RTC_DATE[(days)] Gets the days before and after the specified number of days, such as leap year. Valid in firmware version 20170524 or above.
Controller	General
Example	**RTC_DATE** = 20130227 Online command input >>PRINT **RTC_DATE** Output: 20130227 DIM gRTC_Date,CurDate,curYear,curMonth,curDay ?RTC_DATE gRTC_Date = RTC_DATE CurDate=gRTC_Date mod 20000000 ?CurDate curYear=int(CurDate/10000) ?curYear curMonth=int((CurDate-curYear*10000)/100) ?curMonth curDay=CurDate-curYear*10000-curMonth*100 ?curDay
Instructions	DATE, DATE$

TIME--System Time

Type	System Parameters
Description	**Set system clock time, return total seconds amount after 0 o'clock.** Date can not be modified in system.
Grammar	TIME=hh:mm:ss
Controller	General
Example	**Example One:** TIME=11:14:40 **Example two:** Online command input >>PRINT **TIME** Output: 40541
Instructions	TIME$, RTC_TIME

TIME$--System Time 2

Type	String Functions
Description	**String Functions, return present time in format of 24-hour, hh:mm:ss**

Grammar	TIME$
Controller	General
Example	Online command input >>PRINT **TIME$** Output:11:29:46
Instructions	TIME, RTC_TIME

RTC_TIME--System Time 3

Type	System Parameters
Description	**Set or get system time.** Expression mode is different from TIME.
Grammar	RTC_TIME = hhmmss
Controller	General
Example	Online command input >>**RTC_TIME**=113706 >>PRINT **RTC_TIME** Output: 113706
Instructions	TIME, TIME$

14.3 Axis System Parameter Instructions

WDOG--Total Axes Enable

Type	System Parameters
Description	**Enable all axes.** Use EtherCAT fieldbus, WDOG=1.
Grammar	WDOG=0/1
Controller	General
Example	**WDOG**=1 'enable all axes.
Instructions	AXIS_ENABLE

DISABLE_GROUP--Axes Group

Type	System Instruction
Description	**Set multi axes as one group. Enable of all axes in group will be closed if alarm of any axis in group comes, this is only for EtherCAT axes, no meaning in pulse axes.** Usually used in multi work stations.
Grammar	DISABLE_GROUP(AXIS1, AXIS2, …)

Controller	General
Example	**DISABLE_GROUP**(-1) 'cancel all group setting, alarm will comes: WDOG is closed. **DISABLE_GROUP**(0,5,1) 'axis 0,5,1 as group one. **DISABLE_GROUP**(4,2) 'axis 4,2 as group two. In this situation, alarm of any axis in group one comes, all axes in group one will be disabled, but group two is normal, the same rule as group two.
Instructions	WDOG, AXIS_ENABLE

ERROR_AXIS--Error Axis

Type	System Status
Description	**First axis in which error happens, if return -1, then no error axes.**
Grammar	Var=ERROR_AXIS
Controller	General
Example	?ERROR_AXIS 'print the first error axis, result:-1, no error axis at present.
Instructions	MOTION_ERROR

MOTION_ERROR--Error Axes List

Type	System Status
Description	**List of error axes.** Each bit means one axis, bit0-n indicates axis0-n.
Grammar	Var=MOTION_ERROR
Controller	General
Example	Print **MOTION_ERROR** 'print result, 0-no error
Instructions	ERROR_AXIS

ERROR_SET--Error Output

Type	System Instruction
Description	**Output will open automatically when there is error in BASIC procedure. And it will write error information into relevant MODBUS register, BASIC procedure will recover to run when output status recovers.** Register length is 32 bytes at least.
Grammar	ERROR_SET (outputs,Modbus register address[,errorname]) errorname: set one SUB process for temporary management, this function will be called when there is pause due to Grammar

	errors. Don't use instructions that may cause block, such as WAIT, it should be simple. If Grammar errors happen in SUB process itself, then it will not be processed any more.
Controller	General
Example	**ERROR_SET**(1,200) Mov(30 'there is spell error, now run wrongly, then output 1 opens, and record error information in register. Modbus_string(200,32) ="sample_move.bas,6,e2043" END SUB error_deal() 'call function when there is error ?"enter error to deal sub" 'print 'function process needs to be written END SUB input command: ?MODBUS_STRING(200,32) to check error information. MODBUS_STRING(200,32) ="sample_move.bas,6,e2043" sample_move.bas: file name 6: line number where error happens e2043: error code

RADIUS_ERRSET—Circular Interpolation Check

Type	System Instruction
Description	**Check the configuration of circular interpolation from center of circle to radius.**
Grammar	RADIUS_ERRSET = mode mode: 0- default value, no need to check 1- the radius of start point and end point are different (over 2 pulses), it will correct automatically. 2- It is inconsistent, return 1006 error. Valid after 2022.01.11.
Controller	General
Example	RADIUS_ERRSET = 2 'wrong circular command coordinate, report 1006, then command is not executed. MOVECIRC(200,0,98,0,1) 'draw circular arc RADIUS_ERRSET = 1 'it will correct automatically if circular command coordinate is wrong, and it will hint Center error, radius:98.000000 radiuend:102.000000 diff. ?RADIUS_ERRSET 'set to check
Instructions	MOVECIRC

14.4 IP Parameter Instructions

IP_ADDRESS--IP Address

Type	System Parameters, which are saved into FLASH automatically
Description	**Controller IP address.** Only valid in controllers with Ethernet port, return 32 bits integer when reading, see example one. Modification will take effect immediately, connection will break when use Ethernet, it needs to connect again. When try to connect with controller through single network card, then ensure network card is in the same network segment. If there are multi network cards, then different network cards should use different network segments, set controller network segment same as network card to be connected. When multiple cards, it needs to restart after modifying IP address.
Grammar	IP_ADDRESS = dot.dot.dot.dot
Controller	General
Example	Online command input >>**IP_ADDRESS**=192.168.0.26 >>PRINT **IP_ADDRESS** Output :436250816 Conversion process details as follow： Convert four segments to binary type. 192--1100 0000 168--1010 1000 0--0000 0000 26--0001 1010 Reassociation of binary data 26 0 168 192 0001 1010 0000 0000 1010 1000 1100 0000 Convert to decimal data 436250816
Instructions	IP_GATEWAY, IP_NETMASK

IP_ADDRESS2—IP Address 2

Type	System Parameters, which are saved into FLASH automatically
Description	**The second IP address of ZMC5XX series controller.** Valid in ZMC5XX series controller, default IP address is 192.168.1.11, return 32 bits integer when reading. Modification will take effect immediately, connection will break when use Ethernet, it needs to connect again. When try to connect with controller through single network card, then ensure network card is in the same network segment. if there are multi network cards, then different network cards should use different network segments, set controller network segment same as network card to be connected. When multiple cards, it needs to restart after modifying IP address.
Grammar	IP_ADDRESS2 = dot.dot.dot.dot
Controller	Valid in ZMC5XX series controller
Example	Remote command input: >>**IP_ADDRESS**=192.168.1.26
Instruction	IP_ADDRESS

IP_GATEWAY--IP Gateway

Type	System Parameters, which are saved into FLASH automatically
Description	**Controller IP gateway.** Only valid in controller with Ethernet port, return 32 bits integer when to read.
Grammar	IP_GATEWAY = dot.dot.dot.dot
Controller	General
Example	Online command input >>**IP_GATEWAY** =192.168.0.1 >>PRINT **IP_GATEWAY** Output:16820416 Conversion process see IP_ADDRESS for reference.
Instructions	IP_NETMASK, IP_ADDRESS

IP_NETMASK -- IP Mask

Type	System Parameters, which are saved into FLASH automatically
Description	**Controller IP network mask.** Only valid in controller with Ethernet port, return 32 bits integer when to read.

Grammar	IP_NETMASK=dot.dot.dot.dot
Controller	General
Example	Online command input >>**IP_NETMASK** =255.255.252.0 >>PRINT **IP_NETMASK** Output :16580607 Conversion process see IP_ADDRESS for reference.
Instructions	IP_GATEWAY, IP_ADDRESS

IP_IFDHCP—Get IP Address Automatically

Type	System parameters
Description	**Whether use DHCP set, no use by default.** Get IP address automatically. This is set, fixed IP of ZDEVELOP is useless, it can only be gained automatically. And it needs restart after parameter modification.
Grammar	IP_IFDHCP=1-0 1-get IP automatically 0-use fixed IP
Controller	General
Example	Online command input >>**IP_IFDHCP** =1
Instruction	IP_ADDRESS

IP_IFDHCP2—Get IP Address Automatically 2

Type	System parameters
Description	**Whether the second port of ZMC5XX series controller use DHCP set, no use by default.** Get IP address automatically. This is set, fixed IP of ZDEVELOP is useless, it can only be gained automatically. And it needs restart after parameter modification.
Grammar	IP_IFDHCP2=1/0 1-get IP automatically 0-use fixed IP
Controller	Valid in ZMC5XX series controller
Example	Online command input >>**IP_IFDHCP2** =1
Instruction	IP_ADDRESS

14.5 Controller Information Instructions

VERSION_FPGA--System FPGA Version

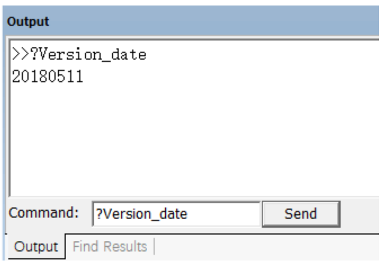
Type	System Status

Description	**System FPGA version No.**
Grammar	VAR1=VERSION_FPGA
Controller	General
Example	?***VERSION_FPGA** 'print FPGA version Result: 240104 “State the controller” – SoftVersion”

VERSION_BUILD--System Firmware Creating Date

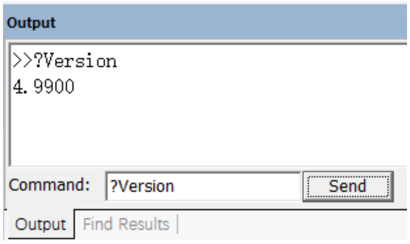
Type	System Status
Description	**The date that creates the system firmware.**
Grammar	VAR1=VERSION_BUILD
Controller	General
Example	?***VERSION_BUILD** 'print firmware creating date Result: 20240111 “State the controller” – SoftVersion”

VERSION_DATE--System Firmware Version

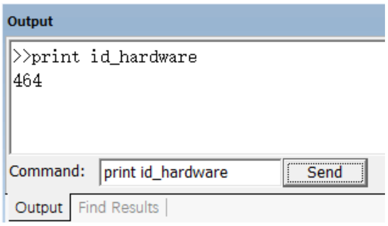
Type	System Status
Description	**System firmware version.**
Grammar	VAR1=VERSION_DATE
Controller	General
Example	?***VERSION_DATE** 'print firmware version, result is 20180511  “State the controller” – SoftVersion”

VERSION--System Software Version

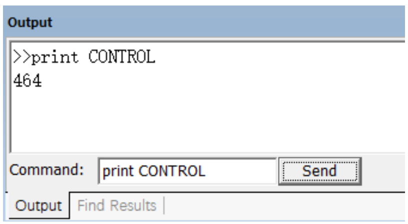
Type	System Status
Description	**System software version NO..**
Grammar	VAR1=VERSION
Controller	General
Example	?***VERSION** 'print software version

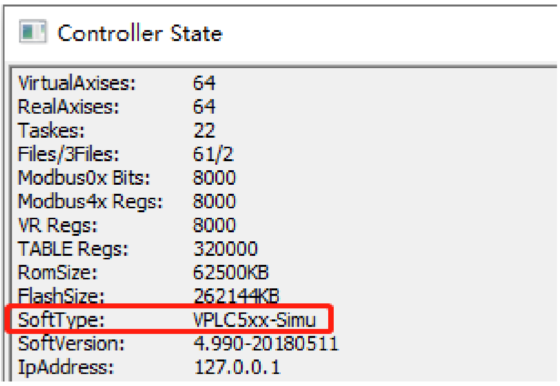
	 "State the controller" – SoftVersion"

ID_HARDWARE--Controller Hardware Type

Type	Parameters to read
Description	**Return controller hardware type.**
Grammar	Val=ID_HARDWARE
Controller	General
Example	Online command input >>PRINT **ID_HARDWARE** 'print controller hardware type. Output: 464  "State the controller" – SoftVersion"
Instructions	[CONTROL](#)

CONTROL--Controller Software Model

Type	Parameters to read
Description	**Return controller software model.**
Grammar	Val=CONTROL
Controller	General
Example	Online command input >>PRINT **CONTROL** 'print controller model name. Output: 464 

<table>
<tr><td></td><td>“State the controller” – SoftType”

</td></tr>
<tr><td>Instructions</td><td>ID_HARDWARE</td></tr>
</table>

SYSTEM_ZSET--Controller Setting

<table>
<tr><td>Type</td><td>System Parameters</td></tr>
<tr><td>Description</td><td>Controller Setting
Set parameters:
bit0: 1-VPSPEED uses interpolation speed by default, 0-VPSPEED uses single axis speed.
bit1: 1-use precision output mode of MOVE_OP, 0-use normal mode of MOVE_OP
bit4: 1-in terms of axis with encoder, use MOVE_OP precision mode of encoder position.
bit7: 1-bus field clock optimization, 0-pulse clock optimization

Once SYSTEM_ZEST opens, all outputs with precision output will change as precision mode. For some controllers, they only can operate one precision output in a controller period. It is not recommended for new versions, instead of using AXIS_ZEST directly to open precision output mode for main axis.

Valid in firmware version above 20170505.
MPOS should follow DPOS before using, since encoder precision function is related to drive response, the smoother speed is, the better precision output will be.

Bus clock optimization is opened after bus opened. Check whether it is used through ?*ETHERCAT.
>>?*ethercat
Slot:0 contain 1 nodes.
dc:ecat-sensitive.Lostcount:0-0</td></tr>
<tr><td>Grammar</td><td>To read: value=SYSTEM_ZSET
To write: SYSTEM_ZSET=value</td></tr>
</table>

Controller	General
Example	SYSTEM_ZSET = 1 ‘set bit0 as 1
Instructions	MOVE_OP, AXIS_ZSET

LEDOUT--Controller Indicator Light

Type	System Instruction
Description	**Operate controller indicator light.** Power indicator light can’t be operated.
Grammar	LEDOUT (num,state) num indicator light number,1-RUN,2-ALM state status, 0-close, 1-open
Controller	General
Example	**LEDOUT**(1,0) 'close RUN indicator **LEDOUT** (2,0) 'close ALM indicator

SERIAL_NUMBER--Unique ID of Controller

Type	System Parameters to Read
Description	**Return unique ID of controller.** It is a unique serial number, generated ZAR file also can be bound with this ID, then this ZAR can only be used in this controller.
Grammar	Var=SERIAL_NUMBER
Controller	General
Example	**Example 1** PRINT **SERIAL_NUMBER** 'print controller ID Print result: 191201941 **Example 2:** 9 bits ID of controller is stored in VR, due to ZMC below 4xx series VR is single-precision float type, which means it only has 8 valid value, it can use 2 VR for store. ?**SERIAL_NUMBER** GLOBAL giA,giB giA = **SERIAL_NUMBER** MOD 100000 'get remainder VR(0)=giA giB = **SERIAL_NUMBER** \ 100000 'exact division VR(1)=giB PRINT giA,VR(0) PRINT giB,VR(1) Print result: 191201941

	1941 1941 1912 1912

SERVO_PERIOD--Fieldbus Communication Period

Type	System Parameters
Description	**Fieldbus servo communication period.** Default value:1000 ms, modification function can be achieved through updating firmware. ZMC 4XX series controller with standard firmware version 20170713, ZMC4XX series controller with "fast" firmware version 20190106, ZMC 5XX series controller with firmware version 20180307 add period-modification function supported for users, but it must be in the range, restart after modification can be taken effect. **When use Rtex drive, set as follow:** When controller period is 500us, set drive P7.20 as 3, set drive P7.21 as 1. When controller period is 1000us, set drive P7.20 as 6, set drive P7.21 as 1. See examples to check Rtex drive motor setting.
Grammar	value=SERVO_PERIOD
Controller	General
Example	Online commands print controller communication period and **Rtex drive motor setting.** >>PRINT **SERVO_PERIOD** 'print servo update period, result is 1000 >>DRIVE_READ(7*256+20)AXIS(0) 'print Rtex axis 0 drive period ratio setting >>DRIVE_READ(7*256+21)AXIS(0) 'print Rtex axis 0 drive period ratio value setting Some firmware modification period: SERVO_PERIOD=500 SERVO_PERIOD=1000
Instructions	SERVO

SYS_ZFEATURE—System Specification

Type	System parameters
Description	**Get system maximum specification.** Valid in ECI controller with firmware version above 150830 and ZMC4xx series controller with firmware version above 170530.
Grammar	num = SYS_ZFEATURE (code) num: return value code: get data type

	0- maximum virtual axis amounts 1- the number that supports motor 2- IN (the number of itself inputs) 3- OUT (the number of itself outputs) 4- AIN (the number of itself analog inputs) 5- AOUT (the number of itself analog outputs) 6- the number of PWM 7- the number of BASIC tasks, no interrupt tasks. 8- the number of bus slot 9- the number of FILE 3 10- the number of serial-port connection 11- the number of ethernet connection 12- the number of custom network connection, 0 – unsupported 13- the number of master stations in network interconnection, 0 – unsupported (for slave station, always support). 14- the number of FLASH blocks 15- the size of FLASH block 16- the number of VR 17- the number of MODBUS_BIT 18- the number of MODBUS_REG 19- the number of timers 20- array space 21- maximum virtual inputs, which corresponds to the number of PLC X registers. 22- maximum virtual outputs, which corresponds to the number of PLC Y registers. 23- maximum virtual analog inputs AIN 24- maximum virtual analog outputs AOUT 25- PLC counter 26- PLC S register 27- PLC V register 28- PLC Z register 29- PLC L register 30- the number of HMI, (net HMI and ontology HMI) 31- the number of HMI itself 32- the largest video memories 33- whether ZINDEX function is supported 34- the number of biggest RTLOG 35- the number of supported sub-card (for former version, they don’t support, for 7XX series, max is 16) 36- the number of current sub-cards (include virtual sub-card) 37- the number of current sub-cards (no virtual sub-card) 39- max specification of PORT (there are adds in 230814) 40- ZV max latches

	41- ZV max tasks recommended 42- max byte space for ZV latching 50- whether supports NC function 51- whether supports CANOPEN 52- whether supports robotic arm 53- the number of ECAT bus slots 54- the number of RTEX bus slots 55- the number of XY2 bus slots 56- whether supports MODBUSM function 57- whether supports SEND command from MODBUSM 58- whether supports U disk 59- the number of network encoders (teaching box handwheel) 60- fastest period 61- lowest period 62- ZAR program space (kbyts) 63- Nandflash space (kbyts) 64- Nandflash remaining space (kybts) 101- whether integrates as ZMIO 102- max ZMIO inputs 103- max ZMIO outputs 104- max ZMIO ADs 105- max ZMIO DAs
Controller	General
Example	?SYS_ZFEATURE (0) 'print maximum axis ?SYS_ZFEATURE (17) 'print the number of MODBUS_BIT
Instruction	/

SYS_IOSET—Special IO Switch

Type	System parameters
Description	**Special IO switch between commonly turn on and commonly turn off .**
Grammar	SYS_IOSET = value 0: default mode, it selects special inputs automatically before compatibility according to controller types, such as, origin position limit, etc. 1: ZMC mode, origin and other special inputs are commonly closed by default. 2: ECI mode, origin and other special inputs are commonly opened by default.
Controller	General
Example	SYS_IOSET = 1

LASER_SET -- Energy Parallel Port Output Switch

Type	System parameters
Description	**Set whether energy parallel port uses AOUT instruction or OP instruction to output.**
Grammar	LASER_SET(isel, value) isel: 1 – set whether energy parallel port uses AOUT command to output. value: 1 – enable AOUT output
Controller	Valid in 504SCAN.
Example	LASER_SET(1,1) 'use AOUT to set laser energy, now, relative original OP instruction is invalid.

ZML_DEFSHIFT – ZML Device "shift" Time

Type	System parameters
Description	**Default shift time of all ZML devices, the unit is ns. After modification, will be saved into FLASH, and please restart.** This function is added after version_build 20240719.
Grammar	value = ZML_DEFSHIFT
Controller	General
Example	Online command print >>PRINT ZML_DEFSHIFT

14.6 Log Instructions

Real time error log function – save common error messages through real-time FIFO, but for some errors that are generated usually or cyclically, it will not write the error log.

This is valid in controllers above ZMC4XX series with firmware above 220907, because there is no enough storage for former controllers.

RTLOG_COUNT – The Number of Current Logs

Type	Log Instructions
Description	**Read the number of logs that are recorded currently, and the max number can be checked through SYS_ZFEATURE(34).**
Grammar	VAL = RTLOG_COUNT()
Controller	Valid in controllers above ZMC4XX series and with firmware above 220907.
Example	?RTLOG_COUNT

Instructions	RTLOG_CLEAR

RTLOG_CLEAR – Clear Current Logs

Type	Log Instructions
Description	**Clear current recorded logs.**
Grammar	RTLOG_CLEAR ([number]) number: the number of logs to be cleared, which starts to clear from the first recorded log, namely, clear all logs. The default is -1.
Controller	Valid in controllers above ZMC4XX series and with firmware above 220907.
Example	RTLOG_CLEAR (10) ‘clear the former 10 logs RTLOG_CLEAR (-10) ‘clear all logs
Instructions	RTLOG_COUNT

RTLOG_ADD – Add Error Message of Log

Type	Log Instructions
Description	**For application level, system logs can be used to record error messages.**
Grammar	RTLOG_ADD (code, “string”) code: error No. string: character string, error message
Controller	Valid in controllers above ZMC4XX series and with firmware above 220907.
Example	RTLOG_ADD (1, “error”)
Instructions	RTLOG_CLEAR

RTLOG_CODE – Get Error No. of Log

Type	Log Instructions
Description	**Get error No. of log.**
Grammar	VAL = RTLOG_CODE ([index]) index: log selection, default (0) means recent logs, when index is -1, which means the most former logs.
Controller	Valid in controllers above ZMC4XX series and with firmware above 220907.
Example	RTLOG_CLEAR (-1) 'clear FOR i = 1 TO 20 RTLOG_ADD(i,"123") 'write into log WA(10) NEXT ?RTLOG_CODE(0) 'get log No.

	Print result: 20
Instructions	RTLOG_CLEAR

RTLOG_TIME$ – Get Error Time of Log

Type	Log Instructions
Description	**Get error time of log.**
Grammar	String = RTLOG_TIME$ ([index]) index: log selection, default (0) means recent logs, when index is -1, which means the most former logs.
Controller	Valid in controllers above ZMC4XX series and with firmware above 220907.
Example	RTLOG_CLEAR (-1) FOR i = 1 TO 20 RTLOG_ADD(i,"123") WA(10) NEXT ?RTLOG_TIME$(0) 'get recent log time Print result: 2022/10/09 16:53:11
Instructions	RTLOG_CODE

RTLOG_INFO – Get Error Message of Log

Type	Character String Functions
Description	**Get error information of logs.**
Grammar	String = RTLOG_INFO ([index]) index: log selection, default (0) means recent logs, when index is -1, which means the most former logs.
Controller	Valid in controllers above ZMC4XX series and with firmware above 220907.
Example	RTLOG_CLEAR (-1) FOR i = 1 TO 20 RTLOG_ADD(i,"123") WA(10) NEXT ?RTLOG_INFO(0) 'get recent log information Print result: 123
Instructions	RTLOG_INFO2

RTLOG_INFO2 – Get Error Message of Log (2)

Type	Character String Functions

Description	**Get error information of logs, including error No. and error time.**
Grammar	String = RTLOG_INFO ([index]) index: log selection, default (0) means recent logs, when index is -1, which means the most former logs.
Controller	Valid in controllers above ZMC4XX series and with firmware above 220907.
Example	RTLOG_CLEAR (-1) FOR i = 1 TO 20 RTLOG_ADD(i,"123") WA(10) NEXT ?RTLOG_INFO(0) 'get recent log information Print result: err20, 2022/10/09 16:53:11,123
Instructions	RTLOG_INFO

?* RTLOG – Clear Current Recorded Logs

Type	Log Instructions
Description	**Rapidly print recent 10 logs.**
Grammar	?*RTLOG
Controller	Valid in controllers above ZMC4XX series and with firmware above 220907.
Example	RTLOG_CLEAR (-1) FOR i = 1 TO 20 RTLOG_ADD(i,"123") WA(10) NEXT ?*RTLOG Print result: RTLOG_COUNT: 20 err20,2022/10/09 17:02:48,123 err19,2022/10/09 17:02:48,123 err18,2022/10/09 17:02:48,123 err17,2022/10/09 17:02:48,123 err16,2022/10/09 17:02:48,123 err15,2022/10/09 17:02:48,123 err14,2022/10/09 17:02:48,123 err13,2022/10/09 17:02:48,123 err12,2022/10/09 17:02:48,123 err11,2022/10/09 17:02:48,123
Instructions	RTLOG_CODE

14.7 TABLE Array Instructions

TABLE--System Default Array

Type	System Array
Description	**Default global array in system, all procedure can access.** **Buffer area for data record, cam data list, screw compensation list, robotic arm parameters, all are saved in TABLE.**
Grammar	TABLE(index) = value,VAR1 = TABLE(index), TABLE(index [, value1..])
Controller	General
Example	**TABLE**(0)=10 'assign 10 to table(0) **TABLE**(10,100,200,300) 'table(10) is assigned as 100, table(11) is assigned as 200, table(12) is assigned as 300.
Instructions	TSIZE

TSIZE – Table Size

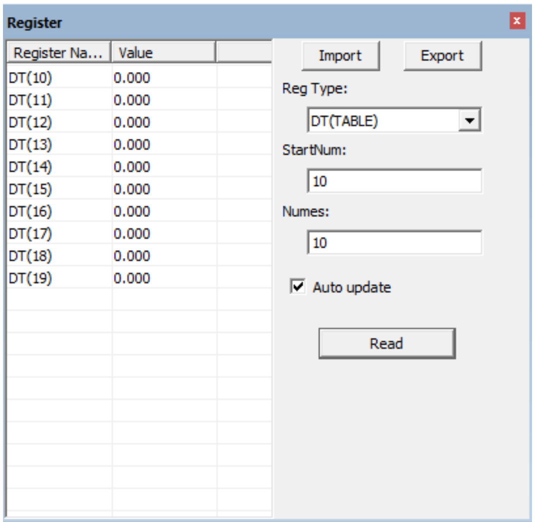
Type	System Parameters
Description	**The number of all elements in TABLE, the size can be modified.** Do modify the table size at first of procedure, that is, before other array definitions (the best is the first line code). Don’t exceed TABLE maximum space.
Grammar	Var=TSIZE TSIZE=Value
Controller	General
Example	Read: PRINT TSIZE 'print table size of controller Set: TSIZE=10000 'set table size, don’t exceed max size of controller table.
Instructions	TABLE

TABLESTRING—Print table in String format

Type	Character string function
Description	**Print data in table according to string format.** Data-converse automatically, printed data is ASCII Code.
Grammar	TABLESTRING(index, length) index: initial address of print data length: data length to print
Controller	General
Example	**Example 1**

<table>
<tr><td></td><td>
TABLESTRI NG(0,5)= "abc"   'save string, starts from tablestring(0)
PRINT TABLESTRING(0,5)   'print saved string
'print result: abc

Example 2
TABLE(100,68,58,92)
PRINT TABLESTRING(100,3) 'print data in string format, converse into ASCII Code
'print result： D:\

In ZDevelop, “view-register”, can check every position’s data in TABLE register.


</td></tr>
</table>

14.8 Instructions Related to Oscilloscope

TRIGGER – Trigger Oscilloscope

Type	System Instruction
Description	**Start to execute data sampling through oscilloscope.** Valid in firmware version above 150723, the function can be used together with ZDevelop – Scope. When oscilloscope opens continuous acquisition, don’t call TRIGGER instruction in Basic, it is recommended to trigger manually.
Grammar	TRIGGER
Controller	General
Example	**TRIGGER** ‘oscilloscope starts to sample the speed of each motion, then save them in TABLE (scope – functrion coniguration). MOVE(10000)

Instructions	SCOPE

SCOPE – Data Acquisition

Type	System Instruction
Description	**Data acquisition, then save into TABLE, 8 types data can be sampled at the same time.** Use TRIGGER to open automatic sampling, sampling time = sampled period * sampled numbers
Grammar	SCOPE(enable[, period]) SCOPE(enable, period, table_start, table_stop, p0 [,p1 [,p2 [,p3 [,p4 [,p5 [,p6 [,p7]]]]]]]) enable: enable or not period: system period, it is generally 1ms, which can be viewed by SERVO_PERIOD table_start: TABLE starting position that saves sampling time. table_stop: TABLE end position, minus the starting position is the number of samples p0~p7: sampling data type, equally divided and stored in the TABLE
Controller	General
Example	BASE(0) ATYPE=1 UNITS=100 DPOS=0 SPEED=100 ACCEL=1000 **SCOPE**(ON,10,0,1000,DPOS(0),MSPEED(0)) 'sample dpos and mspeed every 10ms and store them in TABLE 0~1000, 0~499 for dpos, 500~1000 for mspeed, total sampling 1000/2*10=5s TRIGGER 'start sampling MOVE(10000)
Instructions	TRIGGER, SCOPE_POS

SCOPE_POS – Point Numbers Acquisition

Type	System Instruction
Description	**It is only read, return the number of points sampled by SCOPE currently.**
Grammar	VAR = SCOPE_POS
Controller	General
Example	BASE(0) ATYPE=1

	UNITS=100 DPOS=0 SPEED=100 ACCEL-1000 SCOPE(ON,10,0,1000,DPOS(0),MSPEED(0)) ‘sample configuration TRIGGER ‘start to sample MOVE(10000) WHILE 1 **?*SCOPE_POS** ‘return to current saved sampled points WEND
Instructions	SCOPE

14.9 Instructions Related to VR

CLEAR--Clear VR

Type	System Instruction
Description	**Clear all data in VR.**
Grammar	CLEAR()
Controller	General
Example	CLEAR() 'clear all data in VR
Instructions	VR

VR—Power Failure Storage

Type	System Instruction
Description	**Power failure saving type register, 32 bits float.** Register numbers differ in different controller models Used to saved float type data, use same space with VR_INT and VR_STRIN.
Grammar	VR(index) = value , VAR1 = VR(index)
Controller	General
Example	**VR**(0) = 10 aaa = **VR**(0) ?aaa Print result: 10.5800
Instructions	VR_INT, VRSTRING

VR_INT--Integer Stored when Power Failure

Type	System Instruction

Description	**Power failure saving type register, 32 bits integer.** Register numbers differ in different controller models Used to saved integer type data, use same space with VR and VR_STRIN.
Grammar	VR_INT(index) = value, VAR1 = VR_INT(index)
Controller	General
Example	VR(0) = 10.58 aaa = VR_INT(0) ?aaa Print result: 10, only keep integer part.
Instructions	VR, VRSTRING

VRSTRING--String Stored when Power Failure

Type	String Functions
Description	**Power failure saving type register, used to save string.** String data will be saved as ASCII, use same space with VR_INT and V.one character consumes one VR space.
Grammar	VRSTRING (index[, chares]) index start VR NO., starts from 0. chares total character numbers to be read
Controller	General
Example	Online command input >> **VRSTRING** (0, 8) = "abc" 'save string. >>PRINT **VRSTRING** (0, 8) Output: abc
Instructions	VR, VR_INT

14.10 Instructions Related to 7XX Series

CARD_INFO – Read & Write Control Card Information

Type	Read and write control card information.
Description	**Read information grammar:** var = CARD_INFO (cardnum, sel) cardnum: sub card No., 0-N-1 (N: control card numbers; when there is no sub card, N is -1) sel: information No.

Value	Description
0	The total number of returned sub cards, now fill 0 for cardnum.
1	DEVICE, device No., hardid.
2	VERSION, version
3	Sub card DIP, only valid for PCI card
4	Reserved

5	Sub card unique No., unique No. of the last PCI card is used as the unique No. of RT.
6	Reserved special No.
7	Reserved
8	IO offset, 8 aligns, it will make the order automatically when powered on.
9	AIO offset
10	the number of INs
11	the number of OPs
12	the number of AINs
13	the number of AOUTs
14	Reserved
15	the number of OP of HW
16	the number of pulse axes
17	the number of encoder axes
18	the number of Ecat bus
19	the number of scan
20	the number of 3D galvanometers
21	Power failure storage, the number of VR, it is without VR generally for XPCI.
22	Offset of VR, numbering automatically
23	the number of PWM
24	PWM starting No. on sub card.

Write information grammar:

CARD_INFO (cardnum, sel) = value

cardnum: sub card No., 0 - default

sel: information No.

Value	Description
8	IO offset, 8 aligns, it will make the order automatically when powered on. Note: value must be multiple of 8.
9	AIO offset, it will make the order automatically when powered on.

Grammar	GLOBAL value value=CARD_INFO(0 --1 -1 , 1) read control card device No. ?value END

?*CARD – Print Control Card Information

Type	Control Card Instruction
Description	Print sub card information (in ZDevelop – send ?*CARD in “output”)
Grammar	?*CARD HardId: hardware version Pul: pulses In: inputs Op: outputs Ad: analog inputs Da: analog outputs

	Pwm: PWM numbers flash: flash size size: ROM size serial: card No. license: parameters configuration
Example	?*CARD Print result: >>?*CARD Card0: is XPci, HardId:7133-0 Pul:4 In:16 Op:16 Ad:0 Da:0 Pwm:4 flash:ef4016h size:4194304 serial:221090003 license:AX64 M08 ZV HW

REG_CARD – Control Card Latch

Type	Control Card Instruction
Description	Select latch. When multiple sub cards support latch, switch is valid. It is used together with REGINPUTS and REGIST, when REGIST is called, it supports switch immediately. Latch position: REG_POSE, REG_POSF, REG_POSG, REG_POSH. Latch channel: MARKE, MARKF, MARKG,MARKH (it can extend 8 channels at most).
Grammar	REG_CARD = value Use specific latch of axis, set REG_CARD = -1, then set its remainder value as the card No. of IO that operates latch.
Example	/

Chapter XV Instructions Related to Storage

Zmotion motion controller has internal FLASH memorizer, some models have external memorizer interfaces, such as USB Drive, SD card, see hardware manual for reference.

USB drive or SD card should be converted to FAT format.

15.1 U Disk Instructions

FILE--Operate Files in USB Drive

<table>
<tr><td>Type</td><td>Files Instructions</td></tr>
<tr><td>Description</td><td>Upload and search files in controller or in USB drive.
Choose functions as per relevant string.
fat 32 and fat 16 can be read by U disk, it is invalid in ntfs.
It is recommended to use USB2.0 for controllers below 4xx series, and USB3.0 is used for 4xx controllers or above.
VPLC 5 series are Linux systems, read filename is case-sensitive, name must be capital.</td></tr>
<tr><td>Grammar</td><td>value = FILE "function" ,…

<table><tr><td>"LOAD_ZAR"</td><td>FILE"LOAD_ZAR", "filename"
Load the upgrade ZAR file in USB drive.
filename: procedure file name

If upgrade fails, it will return 0 and output reasons through WARN.
If it succeeds, ZAR file will start automatically, which means return value of TRUE is useless.

⚠ interruption marks in debugging mode will be cleared after upgrade is finished.</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td>"LOAD_TCF"</td><td>FILE"LOAD_TCF", "filename", tableindex, maxsize
Read unique TCP file.
filenam: file name.
tableindex: start TABLE NO. to save data.
maxsize: total TABLE index to save data.
<table>
<tr><td>TABLE0</td><td>Mark Point</td><td></td></tr>
<tr><td>TABLE1</td><td>Total Points</td><td></td></tr>
<tr><td>TABLE2</td><td>Glue Head No.</td><td></td></tr>
<tr><td>TABLE100</td><td>Type of Point</td><td>First Point</td></tr>
<tr><td>TABLE101</td><td>X coordinate</td><td></td></tr>
<tr><td>TABLE102</td><td>Y coordinate</td><td></td></tr>
<tr><td>TABLE103</td><td>Z coordinate</td><td></td></tr>
<tr><td>TABLE104</td><td>Reserved</td><td></td></tr>
<tr><td>TABLE105</td><td>Type of Point</td><td>Second Point</td></tr>
<tr><td>TABLE106</td><td>X coordinate</td><td></td></tr>
<tr><td>TABLE107</td><td>Y coordinate</td><td></td></tr>
<tr><td>TABLE108</td><td>Z coordinate</td><td></td></tr>
<tr><td>TABLE109</td><td>Reserved</td><td></td></tr>
</table></td></tr>
<tr><td></td><td>"LOAD_BYTE"</td><td>FILE"LOAD_BYTE", "filename", tableIndex, maxsize, offset
Load files by bytes.
filename: File Name, when reading csv file, data needs to be put in the first column.
tableindex: Start TABLE index to save data.
maxsize: Total TABLE index to save data.
offset: file bytes offset where starts to read.
<table>
<tr><td>TABLE0</td><td>Total Bytes</td></tr>
<tr><td>TABLE1</td><td>Byte is read firstly.</td></tr>
<tr><td>TABLE2</td><td>Second byte</td></tr>
<tr><td>TABLEn</td><td>The Nth byte</td></tr>
</table></td></tr>
<tr><td></td><td>"FIND_FIRST"</td><td>FILE "FIND_FIRST", type, vr [,dir]
Search files in USB drive. Add selectable parameters, dir, character string parameter.
type:1-file/2-folder/".extend" file suffix name.
vr: found result will be saved in vrstring(vr), if data exceeds VR space, then will be saved in MODBUS_STRING.
dir: assign the path to search, in character string expression format. When not assign, it will search from root directory of U disk by fault.</td></tr>
<tr><td></td><td>"FIND_NEXT"</td><td>FILE "FIND_NEXT", vr
Search the next file in USB drive.
vr: result will be saved in vrstring(vr), if data exceeds VR space, then will be saved in MODBUS_STRING.</td></tr>
</table>

	"FLASH_FIRST"	FILE "FLASH _FIRST", type, vr Search FLASH file, only support BIN and Z3P format. Type:1-file/2-folder/".extend" file suffix name. vr: result will be saved in vrstring(vr), if data exceeds VR space, then will be saved in MODBUS_STRING.
	"FLASH_NEXT"	FILE "FLASH _NEXT", vr Search the next flash file. vr: result will be saved in vrstring(vr), if data exceeds VR space, then will be saved in MODBUS_STRING.
	"FLASH_DEL"	FILE "FLASH_DEL", “fileorddir” Delete selected file. file: full name of file, and with extension name. Support delete folder, can be with symbol drive A, C Drive C means FLASH catalogue, without drive symbol is FLASH catalogue by default. Drive A means USB drive.
	"DELETE"	FILE "DELETE", "filename" Delete selected file in USB drive. For controllers below ZMC4XX, they only support Z3P and BIN type files. For controllers below ZMC4XX, the firmware version is after 20240719, controllers’ NAND flash type add CSV type. Filename: full name of file, with extension name.
	"COPY_FROM"	FILE "COPY_FROM", "FLASH file name"[, "file name in USB drive"] Copy flash file to USB drive, support BIN and Z3P file. Rule of FLASH file name: SD block number. BIN Like: SD0.BIN is flash block 0, SD1.BIN is flash block 1
	"COPY_TO"	FILE "COPY_TO", "file name in USB drive"[, "FLASH file name"] Copy file in USB drive to flash, support BIN and Z3P.
	"FLASH_COPY"	FILE "FLASH_COPY" "src","des" Support copy for folder, can be with symbol drive A,C. C Drive means FLASH catalogue A Drive means USB Drive.
	"MAKE_DIR"	FILE "MAKE_DIR" "path"
	"PATHD"	FILE "PATHD" "dir" Valid in VPLC5XX and VPLC7XX whose firmware version is 230909 and above. D disk character is added, you can manually adjust the mapping path, then special path can be achieved.
	function: function selection	
Controller	General, controllers with USB interface support U Disk function.	

Example	Example 1: download zar update procedure

<table>
<tr><td>Example</td><td>
Example 1: download zar update procedure
<pre>
DIM   result   'define variable
IF U_STATE=TRUE THEN   'check if USB drive was inserted.
    Result = FILE "find_first",".zar",10   'scan first zar file, save it in VR
    IF result=TRUE THEN   'check if scan succeeded
        FILE"load_zar",VRSTRING(10,20) 'download scanned zar file that matches name in VR
    ENDIF
ENDIF
END
</pre>
Example 2: find zar update procedure
<pre>
FILE "find_next",10   'find next zar file result to save in vrstring(10)
FILE "find_prev",20   'find former zar file result to save in vrstring(20)
</pre>
Example 3: FLASH and USB drive data copy each other
<pre>
DIM   a,aa(8)
a=10
FOR   i=0 TO 7
    aa(i)=i
NEXT
WHILE 1
IF   SCAN_EVENT(IN(0))> 0 THEN
    FLASH_WRITE 1,a aa
        FILE"copy_from","sd1.bin"   'copy flash 1 data to USB Drive sd1
        PRINT "copy flash data to USB Drive"
    ELSEIF SCAN_EVENT(IN(1))> 0 THEN
        FILE"copy_to","sd1.bin"   'read sd1 data, then write into flash 1
        PRINT "write USD drive data into flash"
        FLASH_READ 1,a,aa
        PRINT *aa
    ENDIF
WEND
END
</pre>
Example 4: read/delete USB drive file
<pre>
FILE "LOAD_BYTE","00.txt",200,10,0
        'read USB Drive 00.txt file data to save in the tenth address of start table(200), offset is 0, read starts from the first character.
FILE "DELETE" , "sd0.bin"   'delete sd0.bin file in USD drive
</pre>
00.txt file content：ZMOTION
Read result: the first position saves the number of characters, followings save character data in sequence.
</td></tr>
</table>

寄存器名	值
DT(200)	7.000
DT(201)	90.000
DT(202)	77.000
DT(203)	79.000
DT(204)	84.000
DT(205)	73.000
DT(206)	79.000
DT(207)	78.000
DT(208)	0.000
DT(209)	0.000

U_STATE--USB Drive Status

Type	System Status Functions
Description	**Check if USB Drive was inserted.** Ture - inserted, False - not inserted. Only valid in controllers with external interface. When U drive is inserted, if there is no program running in controller, AUTORUN.ZAR will load automatically. But it takes effect only when no any program is downloaded into controller, which means it is still factory status (4XX series controllers with 20170423 firmware version or above). If the program is running, it won't load automatically, at this time, please load in the program manually. Don't put too many files in USB Drive. fat 32 and fat 16 can be read by U disk, it is invalid in ntfs. It is recommended to use USB2.0 for controllers below 4xx series, and USB3.0 is used for 4xx controllers or above.
Grammar	Val=U_STATE
Controller	Controllers with USB interface
Example	?**U_STATE** 'print USB status If **U_STATE** = TRUE TEHN 'USB Drive was inserted. U_READ 1, VAR, ARRAY1, ARRAY2(1) 'read data in USB Drive. ENDIF
Instructions	U_READ, U_WRITE

U_READ--Read USB Drive

Type	Storage Instructions
Description	**Read data from external memorizer (USB DIRVE) to variable or array.** This instruction is only valid in controllers with external interface. File is saved as 32 bits ieee float type in sequence, one variable or one array element consumes one float. use PC to make file ready first, then use

	U_READ to read. fat 32 and fat 16 can be read by U disk, it is invalid in ntfs. It is recommended to use USB2.0 for controllers below 4xx series, and USB3.0 is used for 4xx controllers or above.
Grammar	U_READ sect_num,[,varname][,arrayname][,arrayname(a)][,arrayname(a,length)] sect_num: file number, related to SD【filenum】.BIN. varname: variable name arrayname: array name, TABLE and VR are also regarded as array a: index to operate in array length: array elements number to operate
Controller	Controllers with USB interface
Example	If U_STATE = TRUE THEN 'USB drive was already inserted. **U_READ** 1, VAR, table(0), ARRAY2(1) 'read data in file SD1 from USB Drive. ENDIF
Instructions	U_WRITE, U_READ2, U_STATE

U_READDBL-- Read from USB – double

Type	Storage Instructions
Description	**Read data from external memorizer to variable or array.** Same as U_READ, but U_READ reads float type with 32-bit, U_READDBL reads double type with 64-bit. This instruction is only valid in controllers with external memorizer interfaces. File is saved as 32 bits ieee float type in sequence, one variable or one array element consumes one float. use PC to make file ready first, then use U_READ to read. fat 32 and fat 16 can be read by U disk, but ntfs is invalid. It is recommended to use USB2.0 for controllres below 4xx series, and for 4xx series and above support USB3.0.
Grammar	U_READDBL sect_num, [,varname] [,arrayname] [,arrayname(a)] [,arrayname(a,length)] sect_num: file number, related to SD【filenum】.BIN. varname: variable name arrayname: array name, TABLE and VR are also regarded as array a: index to operate in array length: array elements number to operate
Controller	Controllers with USB interface and version above 4xx series, valid in firmware above 20190128.
Example	If U_STATE = TRUE THEN 'USB drive was already inserted. **U_READDBL** 1, VAR, TABLE(0), ARRAY2(1)

	'read USB Drive data in file SD1 ENDIF
Instructions	U_READ, U_WRITE, U_STATE

U_READ2-- Read USB Drive 2

Type	Storage Instructions
Description	**Read data from external memorizer (USB DIRVE) to variable or array, and it supports set the start position of file reading.** This instruction is only valid in controllers with external memorizer interfaces. File is saved as 32 bits ieee float type in sequence, one variable or one array element consumes one float. use PC to make file ready first, then use U_READ to read. fat 32 and fat 16 can be read by U disk, but ntfs is invalid. It is recommended to use USB2.0 for controllres below 4xx series, and for 4xx series and above support USB3.0.
Grammar	U_READ sect_num, star_num [,varname] [,arrayname] [,arrayname(a)] [,arrayname(a,length)] sect_num: file number, related to SD【filenum】.BIN. start_num: starting position of reading file varname: variable name arrayname: array name, TABLE and VR are also regarded as array a: index to operate in array length: array elements number to operate
Controller	Controllers with USB interface
Example	If U_STATE = TRUE THEN 'USB drive was already inserted. **U_READ2** 1, 10, VAR, table(0), ARRAY2(1) 'read USB Drive data in file SD1 ENDIF
Instructions	U_READ, U_WRITE, U_STATE

U_READ2DBL-- Read from USB 2 – double

Type	Storage Instructions
Description	**Read data from external memorizer to variable or array, start position to read can be set.** Same as U_READ2, but U_READ2 reads float type with 32-bit, U_READ2DBL reads double type with 64-bit. This instruction is only valid in controllers with external memorizer interfaces. File is saved as 32 bits ieee float type in sequence, one variable or one array

	element consumes one float. use PC to make file ready first, then use U_READ to read. fat 32 and fat 16 can be read by U disk, but ntfs is invalid. It is recommended to use USB2.0 for controllres below 4xx series, and for 4xx series and above support USB3.0.
Grammar	U_READ2DBL sect_num, star_num [,varname] [,arrayname] [,arrayname(a)] [,arrayname(a,length)] sect_num: file number, related to SD【filenum】.BIN. star_num : start position of reading in files varname: variable name arrayname: array name, TABLE and VR are also regarded as array a: index to operate in array length: array elements number to operate
Controller	Controllers with USB interface and version above 4xx series, valid in firmware above 20190128.
Example	If U_STATE = TRUE THEN 'USB drive was already inserted. **U_READ2DBL** 1, 10, VAR, TABLE(0), ARRAY2(1) 'read USB Drive data in file SD1, starts from 10 ENDIF
Instructions	U_READ, U_WRITE, U_STATE

U_READDSB--Read DSB File

Type	Storage Instructions
Description	**Read DSB file.** Only valid in controllers with external interface. fat 32 and fat 16 can be read by U disk, but ntfs is invalid. It is recommended to use USB2.0 for controllres below 4xx series, and for 4xx series and above support USB3.0.
Grammar	U_READDSB "filename", tableindex, maxsize [, flag] filename file name TableIndex start TABLE index to save Maxsize total TABLE index numer to save flag reserved.

Number	Contents	
0	Mark bit: the number of variables in one line. (reserved)	
1	Stitch Count	
2--29	File name, 28 characters	
30	times of colors change	
31	+X:	
32	-X:	
33	+Y:	

34	-Y:	
35	AX: +	
36	AY: -	
37	MX: +	
38	MY: +	
39	PD:	
40	L_limt	coordinate of left limit
41	R_limt	coordinate of right limit
42	U_limt	coordinate of up limit
43	D_limt	coordinate of lower limit
99	Total lines have been read	
100	Line Type	First point
101	parameters1: X relative distance	
102	parameters2: Y relative distance	
103	parameters3: X coordinate relates to point stitch starts	
103	parameters4: Y coordinate relates to point stitch starts	
105	Line Type	Second point
106	parameters1: X relative distance	
107	parameters2: Y relative distance	

Controller	Controllers with USB interface
Instructions	U_READ, U_WRITE, U_STATE

U_WRITE—Output to USB Drive

Type	Storage Instructions
Description	**Store variables or arrays, single element or some elements of array are saved into external memorizer.** This instruction is only valid in controllers with external interface. File is saved as 32 bits ieee float type in sequence, one variable or one array element consumes one float. use PC to make file ready first, then use U_READ to read. It is recommended to use USB2.0 for controllres below 4xx series, and for 4xx series and above support USB3.0.
Grammar	U_WRITE sect_num,[,varname][,arrayname][,arrayname(a)][,arrayname(a,length)] sect_num: file number, related to SD【filenum】.BIN. varname: variable name arrayname: array name, TABLE and VR are also regarded as array a: index to operate in array length: array elements number to operate

Controller	Controllers with USB interface
Example	If U_STATE = TRUE THEN 'USB drive was already inserted. **U_WRITE 0**, TABLE(0, 10) 'write TBALE 0-10 into USB drive SD0 ENDIF
Instructions	U_READ, U_STATE

U_WRITEDBL—Output to USB – double

Type	Storage Instructions
Description	**Store variables or arrays, single element or some elements of array are saved into external memorizer.** Same as U_WRITE, but U_READ outputs float type with 32-bit, U_WIRTEDBL outputs double type with 64-bit. This instruction is only valid in controllers with external memorizer interfaces. File is saved as 32 bits ieee float type in sequence, one variable or one array element consumes one float. use PC to make file ready first, then use U_READDBL to read. fat 32 and fat 16 can be read by U disk, but ntfs is invalid. It is recommended to use USB2.0 for controllres below 4xx series, and for 4xx series and above support USB3.0.
Grammar	U_WRITEDBL sect_num, [,varname] [,arrayname] [,arrayname(a)] [,arrayname(a,length)] sect_num: file number, related to SD【filenum】.BIN. varname: variable name arrayname: array name, TABLE and VR are also regarded as array a: index to operate in array length: array elements number to operate
Controller	Controllers with USB interface and version above 4xx series, valid in firmware above 20190128.
Example	If U_STATE = TRUE THEN 'USB drive was already inserted. **U_WRITEDBL** 1, VAR, TABLE(0), ARRAY2(1) 'write data of TABLE 0-10 in file SD0 ENDIF
Instructions	U_READ, U_WRITE, U_STATE

STICK_READ—Read USB Drive to Table

Type	Storage Instructions
Description	**Copy data of external memorizer to TABLE.** When value=TRUE, which means it is successful, or means failure.

	Recommended to use U_READ instead. This instruction is only valid in controllers with external interface. fat 32 and fat 16 can be read by U disk, but ntfs is invalid. It is recommended to use USB2.0 for controllres below 4xx series, and for 4xx series and above support USB3.0.
Grammar	value = STICK_READ (filenum, table_start [,format]) filenum file name, relates to SD【filenum】 table_start TABLE NO. where operation starts. format file format that will get from external memorizer. Value / Description 0(default) / Float format, .BIN 1 / Text format, .CSV
Controller	Controllers with USB interface
Example	**STICK_READ** (0,10,0) 'copy bin file named SD0 in external memorizer to TABLE, starts to save from TABLE(10).
Instructions	U_READ, STICK_READVR

STICK_WRITE--Table to USB Drive

Type	Storage Instructions
Description	**Copy data in table to external memorizer (USB DRIVE).** When value=TRUE, which means success,or means failure. Recommended to use U_WRITE instead. This instruction is only valid in controllers with external interface. fat 32 and fat 16 can be read by U disk, but ntfs is invalid. It is recommended to use USB2.0 for controllres below 4xx series, and for 4xx series and above support USB3.0.
Grammar	value = STICK_WRITE(filenum, table_start [,length [,format]]) filenum file name, relates to SD【filenum】, max is 9999 table_start TABLE NO. where operation starts. length TABLE data number to operate,default value is 128. format file format that will be written into memorizer. Value / Description 0(default) / Float format, .BIN 1 / Text format, .CSV
Controller	Controllers with USB interface
Example	**STICK_WRITE**(0,0,128,0) 'copy front 128 elements of table and save them into external memorizer in float format, then generate SD0.BIN file.
Instructions	U_WRITE, STICK_WRITEVR

STICK_READVR--USB Drive to VR

Type	Storage Instructions
Description	**Copy data in external memorizer to VR.** When value=TRUE, which means success,or means failure. Recommended to use U_READ instead. This instruction is only valid in controllers with external interface. fat 32 and fat 16 can be read by U disk, but ntfs is invalid. It is recommended to use USB2.0 for controllres below 4xx series, and for 4xx series and above support USB3.0.
Grammar	value = STICK_READVR (filenum,vr_start [,format]) filenum file name, relates to SD【filenum】 table_start VR NO. where operation starts. format file format that will get from external memorizer. Value / Description 0(default) / Float format, .BIN 1 / Text format, .CSV
Controller	Controllers with USB interface
Example	**STICK_READVR** (0,20,0) 'copy SD0 bin file of external memorizer to VR, starts from VR(20).
Instructions	U_READ, STICK_READ

STICK_WRITEVR--VR to USB Drive

Type	Storage Instructions
Description	**Copy data in VR to external memorizer (USB DRIVE).** When value=TRUE, which means success,or means failure. Recommended to use U_WRITE instead. This instruction is only valid in controller with external interface. It is recommended to use USB2.0 for controllres below 4xx series, and for 4xx series and above support USB3.0.
Grammar	value = STICK_READVR (filenum, vr_start [,format]) filenum file name, relates to SD【filenum】 table_start VR NO. where operation starts. length VR data number to operate,default value is 128. format file format that will be written into external memorizer. Value / Description 0(default) / Float format, .BIN 1 / Text format, .CSV
Controller	Controllers with USB interface
Example	STICK_WRITEVR (0,0,128,0) 'copy front 128 elements of table and

	save them into external memorizer in float format, then generate SD0.BIN file.
Instructions	U_WRITE, STICK_WRITE

15.2 FLASH Instructions

FLASH_WRITE--Write Flash

Type	Storage Instructions
Description	**Store variables or arrays, single element or some elements in array are saved into flash, support power failure storage.** Storage type in Flash is sequential, the read sequence should be same as sequence to save. Storage times are limited in Flash, don't operate exceeding limit. Don't operate FLASH in motion process, or will influence on motion execution.
Grammar	FLASH_WRITE sect_num [, varname] [, arrayname] [, arrayname(a)] [, arrayname(a,length)] sect_num: FLASH block number, different types are differenet. varname: variable' name arrayname: array' name, TABLE and VR are also regarded as array a: index to operate in array length: array elements number to operate
Controller	General
Example	**Example 1** FLASH_WRITE 1, VAR, ARRAY1, ARRAY2(1) 'write VAR,ARRAY1,ARRAY2(1) data to flash block 1 in sequence. **Example 2** TABLE(1)=123.456 FLASH_WRITE 1, TABLE(1) TABLE(1)=200 FLASH_READ 1, TABLE(1) ?TABLE(1) 'print result: 123.45600 **Example 3:** FLASH storage is float precision, for 32 bits integer data, it should use 2 MODBUS_REG to store MODBUS_LONG. MODBUS_LONG(1)=123456 'store in MODBUS_REG(1) and MODBUS_REG(2) FLASH_WRITE 1, MODBUS_REG(1,2) 'select 2 elements from MODBUS_REG, write them into FLASH block as FLASH_WRITE 1, MODBUS_REG(1), MODBUS_REG(2) MODBUS_LONG(1)=100

	FLASH_READ 1, MODBUS_REG(1,2) ?MODBUS_REG(1) 'print result: -7616 ?MODBUS_REG(2) 'print result: 1 ?MODBUS_LONG(1) 'print result: 123456
Instructions	FLASHVR, FLASH_READ

FLASH_WRITEDBL--Write Flash--double

Type	Storage Instructions
Description	**Store variables or arrays, single element or some elements in array are saved into flash, support power failure storage.** Same as FLASH_WRITE, but FLASH_WRITE saves float type with 32-bit, FLASH_WRITEDBL saves double type with 64-bit. Storage type in Flash is sequential, the read sequence should be same as sequence to save. Storage times are limited in Flash, don't operate exceeding limit. Don't operate FLASH in motion process, or will influence on motion execution.
Grammar	FLASH_WRITEDBL sect_num [, varname] [, arrayname] [, arrayname(a)] [, arrayname(a,length)] sect_num: FLASH block number, different types are differenet. varname: variable's name arrayname: array' name, TABLE, VR and MODBUS are also regarded as array a: index to operate in array length: array elements number to operate
Controller	Valid in ZMC4XX series controllers with firmware 20190128 or above.
Example	FLASH_WRITEDBL 1, table(0,4) 'write 5 elements (table(0)~table(4)) into FLASH 1.
Instructions	FLASHVR, FLASH_READ

FLASH_READ--Read Flash

Type	Storage Instructions
Description	**Read data from internal Flash to variable or array.** Storage type in Flash is sequential, sequence to read should be same as sequence to write. When read flash block that hasn't been written before, it will show message: Warn file: "BASIC1.BAS" line:5 task:0, File:C\SD10.BIN open error, not load., but this will not influence on use. Don't operate FLASH in motion process, or will influence on motion execution.

Grammar	FLASH_READ sect_num [, varname] [, arrayname] [, arrayname(a)] [, arrayname(a,length)] sect_num: FLASH block number, different types are differenet. varname: variable's name arrayname: array' name, TABLE and VR are also regarded as array a: index to operate in array length: array elements number to operate
Controller	General
Example	FLASH_READ 1, VAR, ARRAY1, ARRAY2(1) 'read data in flash block 1, and then save into VAR, ARRAY1, ARRAY2(1).
Instructions	FLASHVR, FLASH_WRITE, FLASH_READDBL

FLASH_READDBL--Read Flash--double

Type	Storage Instructions
Description	**Read data from internal Flash to variable or array.** Same as FLASH_READ, but FLASH_READ saves float type with 32-bit, FLASH_READDBL saves double type with 64-bit. Storage type in Flash is sequential, the read sequence should be same as sequence to save. Storage times are limited in Flash, don't operate exceeding limit. Don't operate FLASH in motion process, or will influence on motion execution.
Grammar	FLASH_READDBL sect_num [, varname] [, arrayname] [, arrayname(a)] [, arrayname(a,length)] sect_num: FLASH block number, different types are differenet. varname: variable's name arrayname: array' name, TABLE and VR are also regarded as array a: index to operate in array length: array elements number to operate
Controller	Valid in ZMC4XX series controllers with firmware 20190128 or above.
Example	FLASH_READDBL 1, table(6,5) 'read data in FLASH 1, starts from 6, read 5 elements, then put them in table(6) ~ table (10)
Instructions	FLASHVR, FLASH_READ, FLASH_WRITE

LASH_READ2--Read Flash (2) -- double

Type	Storage Instructions
Description	**Read data from internal Flash to variable or array.** Storage type in Flash is sequential, sequence to read should be same as

	sequence to write. When read flash block that hasn’t been written before, it will show message: Warn file: “BASIC1.BAS” line:5 task:0, File:C\SD10.BIN open error, not load., but this will not influence on use. Don’t operate FLASH in motion process, or will influence on motion execution.
Grammar	FLASH_READ2 sect_num start_num [, varname] [, arrayname] [, arrayname(a)] [, arrayname(a,length)] sect_num: FLASH block number, different types are differenet. start_num: start position of file inside reading. varname: variable’s name arrayname: array’s name, TABLE and VR are also regarded as array a: index to operate in array length: array elements number to operate
Controller	General
Example	FOR i = 0 TO 10 TABLE (i) = 120 + i NEXT FLASH_WRITE 1, TABLE (0,10) 'write data in flash FOR i = 0 TO 11 TABLE (i) = 0 NEXT FLASH_READ2 1,2 TABLE(4,5) 'read data starting from table(2) in sequence, and all 5 data are put into table(4) – table (8) FOR i = 0 TO 11 ? “TABLE“, i,TABLE(i) NEXT END
Instructions	FLASH_READ, FLASH_WRITE

FLASH_READ2DBL--Read Flash (2)--double

Type	Storage Instructions
Description	**Read data from assigned position of internal Flash to variable or array.** Same as FLASH_READ, but FLASH_READ saves float type with 32-bit, FLASH_READ2DBL saves double type with 64-bit. Storage type in Flash is sequential, sequence to read should be same as sequence to write. When read flash block that hasn’t been written before, it will show message:

	Warn file: "BASIC1.BAS" line:5 task:0, File:C\SD10.BIN open error, not load., but this will not influence on use. Don't operate FLASH in motion process, or it will influence on motion execution.
Grammar	FLASH_READ2DBL sect_num start_num [, varname] [, arrayname] [, arrayname(a)] [, arrayname(a,length)] sect_num: FLASH block number, different types are differenet. start_num: in file, starting position of reading. varname: variable's name arrayname: array's name, TABLE and VR are also regarded as array a: index to operate in array length: array elements number to operate
Controller	Valid in ZMC4XX series controllers with firmware 20190128 or above.
Example	FOR i = 0 TO 10 TABLE (i) = 120 + i NEXT FLASH_WRITE 1, TABLE (0,10) 'write data in flash FOR i = 0 TO 11 TABLE (i) = 0 NEXT FLASH_READ2DBL 1,2 TABLE(4,5) 'read data starting from table(2) in sequence, and all 5 data are put into table(4) – table (8) FOR i = 0 TO 11 ? "TABLE", i,TABLE(i) NEXT END
Instructions	FLASH_READ2, FLASH_WRITE, FLASH_READDBL

FLASHVR--Copy RAM Data

Type	Storage Instructions
Description	**Copy data from RAM to FLASH.** TABLE data are saved in the last FLASH block of ZMC00x series (except ZMC005), and ECI series controller, since FLASH_WRITE and FLASH_READ can also operate this block, do arrange logic well to avoid conflict. TABLE data are saved in an independent area in other controller series, which can't operate.

	Don’t operate FLASH in motion process, or will influence on motion execution.
Grammar	FLASHVR (function) function select function -1 Save all TABLE in FLASH. and recover data to TABLE automatically when power on. -2 Cancel function: recover data to TABLE automatically when power on.
Controller	General
Example	**FLASHVR** (-1) 'save data from table into FLASH, recover flash data to table when power on.
Instructions	FLASH_WRITE

FLASH_SECTSIZE--Variable Numbers in Flash

Type	System Status Functions
Description	**Read the number of varibles that can be saved in flash block.** Different controllers have different numbers.
Grammar	value = FLASH_SECTSIZE
Controller	General
Example	? FLASH_SECTSIZE print result 20480 'ZMC1xx series 1024 'ZMC00x series ...
Instructions	FLASH_SECTES

FLASH_SECTES--Flash Block Number

Type	System Status Functions
Description	**Read the total numbers of FLASH inside the controller.** Different controllers have different numbers. In terms of ZMC00X series, FLASHVR will use last block to avoid conflict.
Grammar	value = FLASH_SECTES
Controller	General
Example	?FLASH_SECTES 'print flash block numbers print result 128 'ZMC005
Instructions	FLASH_SECTSIZE

15.3 File Storage Related Instructions

FILE_ZOPEN – Open File

Type	Storage Instructions
Description	**According to the mode assigned by "mode" parameter, open one file, then return file handle.**
Grammar	Function Grammar: handle = FILE_ZOPEN (path, mode) path: file path, character string, the flash drive is defined as c and U drive is defined as a, such as, c:\filename.txt means the file "filename.txt" of flash. The default is file in flash. (for VPLC5XX and VPLC7XX, you can custom disk character by "FILE "PATHD" "dir"" command.) mode: the mode of open, character string "w" means open through writing method (delete all, then write), "r" means open through reading method, "rw" means it can be opened through writing or reading. Note: "w" mode will delete the data that was written before.
Controller	Valid in ZMC4XX Series controllers with firmware version above 20170601.

Example	**Example 1:**
	FILE "copy_to","test.z3p","test1.z3p"
	'files in U drive are copied into FLASH
	f1 = FILE_ZOPEN("c:\test1.z3p","r") 'open test1.z3p file of FLASH
	IF f1 >= 0 THEN
	?"open FLASH existing file through "r" mode"
	ELSE
	?"error, open fails"
	ENDIF
	FILE_ZOPEN(f1)
	
	Example 2:
	f1 = FILE_ZOPEN("c:\test.z3p","rw")
	num = **FILE_ZWRITES**(f1,"0123456789")
	FILE_ZSEEK(f1,0,0)
	FILE_ZREAD(f1,30,10)
	IF num = 10 AND TABLE(31) = 49 AND TABLE(39) = 57 THEN
	?"write and read FLASH file Success"
	ELSE
	?"error, fail to write and read FLASH file"
	END IF
	FILE_ZCLOSE(f1)

Instructions	FILE_ZCLOSE

FILE_ZCLOSE – Close File

Type	Storage Instructions
Description	**Close file handle “handle” indicated file, if the handle is invalid, warn will output.**
Grammar	Command Grammar: FILE_ZCLOSE (handle) handle: returned file handle of function “FILE_ZOPEN”.
Controller	Valid in ZMC4XX Series controllers with firmware version above 20170601.

Example	FILE "copy_to","test.z3p","test1.z3p"
	'files in U drive are copied into FLASH
	f1 = FILE_ZOPEN("c:\test1.z3p","r") 'open test1.z3p file of FLASH
	IF f1 >= 0 THEN
	?"open FLASH existing file through “r” mode"
	ELSE
	?”error, open fails”
	ENDIF
	FILE_ZCLOSE(f1)

Instructions	FILE_ZOPEN

FILE_ZWRITES – Write File into Character String

Type	Storage Instructions
Description	**Write character string “string” into file handle “handle” assigned file, and the function returns the number of wrote characters.**
Grammar	Command Grammar: FILE_ZWRITES (handle, string) Function Grammar: num = FILE_ZWRITES (handle, string) handle: returned file handle of function “FILE_ZOPEN”. string: normal character string to be written
Controller	Valid in ZMC4XX Series controllers with firmware version above 20170601.

Example	FILE “copy_to”, “test.z3p”, “test.z3p”
	FOR i = 0 TO 9
	TABLE (30 + i)
	NEXT
	
	f1 = FILE_ZOPEN("c:\test.z3p","rw")
	num = **FILE_ZWRITES**(f1,"0123456789")
	FILE_ZSEEK(f1,0,0)
	FILE_ZREAD(f1,30,10)
	IF num = 10 AND TABLE(31) = 49 AND TABLE(39) = 57 THEN
	?"write and read FLASH file Success"
	ELSE
	?"error, fail to write and read FLASH file"

	END IF FILE_ZCLOSE(f1)
Instructions	FILE_ZOPEN, FILE_ZCLOSE, FILE_ZSEEK, FILE_ZREAD.

FILE_ZWRITE – Write File into Character

Type	Storage Instructions
Description	**Write "count" character into the file assigned by file handle "handle", the character locates in the starting position of index "tableindex" in table, and the function returns to the number of actually writing in. In addition, there is the limit for character writing number, if it exceeds, write the max number into.**
Grammar	Command Grammar: FILE_ZWRITE (handle, tableindex, count) Function Grammar: num = FILE_ZWRITE (handle, tableindex, count) handle: returned file handle of function "FILE_ZOPEN". tableindex: the index of table, and put the character to be written to starting table of tableindex, each character is saved into each position of table. count: the number of characters to be written into.
Controller	Valid in ZMC4XX Series controllers with firmware version above 20170601.
Example	FILE "copy_to","test.z3p","test.z3p" FILE "copy_from","test.z3p","test1.z3p" f1 = FILE_ZOPEN("a:\test1.z3p","rw") FILE_ZWRITE(f1,100,512)
Instructions	FILE_ZOPEN, FILE_ZCLOSE

FILE_ZREAD – Read Character from File

Type	Storage Instructions
Description	**Read the character "maxchars" from the file assigned by "handle" of file handle, the read character is saved into index "tableindex" starting position of table.** If the file end character is read, stop in advance. The function returns the number of reading characters, also, there is the limit for reading number, if it exceeds, according to the max number.
Grammar	Command Grammar: FILE_ZREAD (handle, tableindex, maxchars) Function Grammar: num = FILE_ZREAD (handle, tableindex, maxchars) handle: returned file handle of function "FILE_ZOPEN". tableindex: the index of table, and put the read character to starting table of tableindex, each character is saved into each position of table. maxchars: the maximum number of characters scheduled to be read. If the read encounters the end of the file, the number may be insufficient. You can get the read number by returning the value

Controller	Valid in ZMC4XX Series controllers with firmware version above 20170601.
Example	Refer to FILE_ZWRITES routine.
Instructions	FILE_ZOPEN, FILE_ZCLOSE, FILE_ZSEEK, FILE_ZWRITES

FILE_ZREADLINE – File Line Reading

Type	Storage Instructions
Description	**Read a line from the file indicated by the file handle "handle" to the tableindex position of the table, and read maxchars characters at most.** The excess part will not be read, and the read position of the file will remain at the first unread character. Reading starts from the current reading position of the file, and the reading result does not contain the line break of the end, but the file reading position will skip the line break, and the function returns the number of characters read. The number of reads has a maximum limit, and if it exceeds, it will be read according to the maximum limit.
Grammar	Command Grammar: FILE_ZREAD LINE (handle, tableindex, maxchars) Function Grammar: num = FILE_ZREAD LINE(handle, tableindex, maxchars) handle: returned file handle of function "FILE_ZOPEN". tableindex: the index of table, and put the read character to starting table of tableindex, each character is saved into each position of table. maxchars: the maximum number of characters that can be saved into table. If it exceeds maxchars, remain part will not be read, and the read position of the file will remain at the first unread character.
Controller	Valid in ZMC4XX Series controllers with firmware version above 20170601.
Example	DIM handle, num handle = FILE_ZOPEN("test.txt", "r") FILE_ZSEEKLINE(handle, 0, 2) 'the beginning of the last line num = FILE_ZREADLINE(handle, 0, 1000) 'locate to the end of the line after reading, the line number remains unchanged num = FILE_ZREADLINE(handle, 0, 1000) 'has reached the end of the line, return 0 ?num, table(0) '0 0 ?FILE_ZTELLLINE(handle) 'the line number read above remains FILE_ZSEEKLINE(handle, -1, 2) 'the beginning of the penultimate line num = FILE_ZREADLINE(handle, 0, 1000) 'read the penultimate line, locate to the beginning of the penultimate line FILE_ZSEEKLINE(handle, -1, 1) 'the beginning of the previous line, here is the beginning of the penultimate line FILE_ZSEEKLINE(handle, 17, 0) 'locate to line number 17 (starting from 0), which is the beginning of line 18

	FILE_ZSEEKLINE(handle, 0, 1) 'move to the beginning of this line num = FILE_ZREADLINE(handle, 0, 1000) FILE_ZCLOSE(handle)
Instructions	FILE_ZSEEKLINE, FILE_ZTELLLINE

FILE_ZSEEK – File Location

Type	Storage Instructions
Description	**File positioning, move read and write position of file to assigned position of pos and mode.**
Grammar	Command grammar: FILE_ZSEEK (handle, pos, mode) handle: the file handle returned by the function FILE_ZOPEN pos: the position or offset to be positioned, depending on the mode, it can take a negative number mode: the reference position when performing the positioning function, which can be 0, 1, or 2. **Mode:** 0: represents the file header as the reference, pos indicates the position to be located, and the reading and writing position of the file will be located at pos after execution. In this case, the negative value of pos will be treated as 0, that is, it will be located at the beginning of the file. 1: represents the current position as the reference, pos indicates the positioning offset, after execution, the file reading and writing position will move pos relative to the current position, a positive value means moving to the end of the file, and a negative value means moving to the beginning of the file. 2: represents the end of the file as the reference, and pos indicates the positioning offset. After execution, the reading and writing position of the file will move relative to the end of the file to the head of the file -pos. In this case, the effective value of pos must be less than or equal to 0, and a positive value will be used as 0 processing, which means positioning to the end of the file.
Controller	Valid in ZMC4XX Series controllers with firmware version above 20170601.
Example	Refer to FILE_ZWRITES routine.
Instructions	FILE_ZOPEN, FILE_ZCLOSE, FILE_ZREAD, FILE_ZWRITES

FILE_ZSEEKLINE – File Line Location

Type	Storage Instructions
Description	**File line positioning, move the reading and writing position of the file to the beginning of the line specified by line and mode. The number of**

	characters that can be processed at one time is limited. If the positioning offset is too large or the number of long lines is large, the positioning may not be completed. The current line position can be obtained through FILE_ZTELLLINE, and it can be positioned through multiple executions if necessary.
Grammar	Command grammar: FILE_ZSEEKLINK (handle, line, mode) handle: the file handle returned by the function FILE_ZOPEN line: the line No. or offset to be positioned, depending on the mode, it can take a negative number mode: the reference position when performing the positioning function, which can be 0, 1, or 2. **Mode:** 0: represents the file header as the reference, line indicates the line No. to be located, and file is positioned to the beginning of the “line” indication line after execution, 0 means locating at the beginning of the file. 1: represents the current position as the reference, line indicates the positioning offset, 0 means positioning to the beginning of the current line, and a negative value means positioning to the direction of file head. 2: represents the end of the file as the reference, line indicates the positioning offset, 0 means positioning to the beginning of the last line, and a negative value means positioning to the direction of file head.
Controller	Valid in ZMC4XX Series controllers with firmware version above 20170601.
Example	Refer to FILE_ZREADLINE routine.
Instructions	FILE_ZREADLINE, FILE_ZTELLLINE

FILE_ZTELL – File Reading and Writing Position

Type	Storage Instructions
Description	**Return the current reading and writing position of “handle” file, which starts from 0.**
Grammar	Function grammar: pos = FILE_ZTELL (handle) handle: the file handle returned by the function FILE_ZOPEN
Controller	Valid in ZMC4XX Series controllers with firmware version above 20170601.
Example	handle = FILE_ZOPEN("test_w.txt", "r") 'use the generated file FILE_ZSEEK(handle, -1, 2) num = FILE_ZREAD(handle, 100, 5) ? num, TABLE(100), TABLE(101) '1 51 0 ?FILE_ZTELL(handle) 'equal to the file size FILE_ZCLOSE(handle)
Instructions	FILE_ZOPEN, FILE_ZCLOSE

FILE_ZTELLLINE – File Line No.

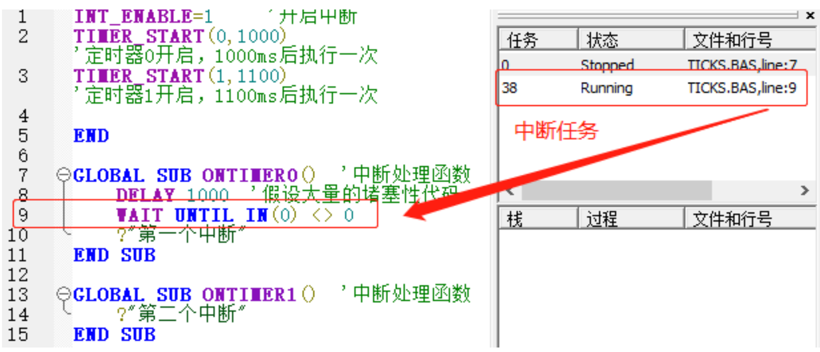
Type	Storage Instructions
Description	**Return the current line No. of handle file, the line No. starts from 0. The line No. can only be obtained in the case of a line operation, and the non-line operation will cause the line number to be invalid, and the return value is -1.**
Grammar	Function grammar: pos = FILE_ZTELL (handle) handle: the file handle returned by the function FILE_ZOPEN
Controller	Valid in ZMC4XX Series controllers with firmware version above 20170601.
Example	Refer to FILE_ZREADLINE routine.
Instructions	FILE_ZREADLINE, FILE_ZSEEKLINE

Chapter XVI Instructions Related to Interrupt

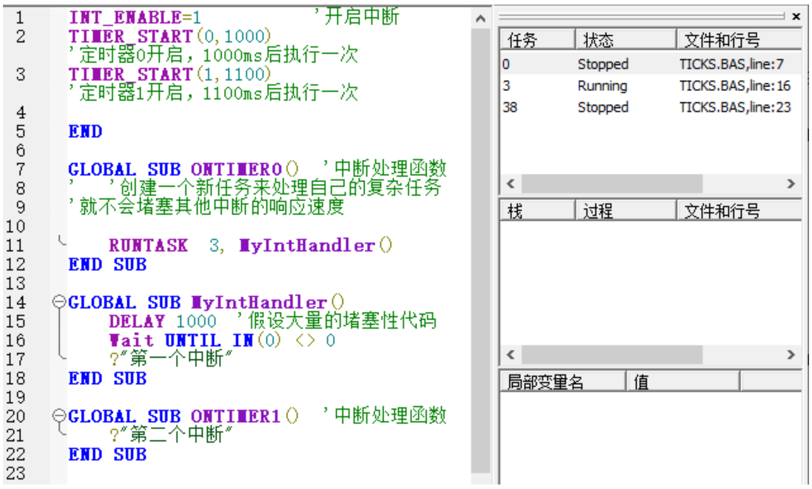
16.1 Three Interrupt Instructions

INT_ENABLE--Main Switch of Interrupt

<table>
<tr><td>Type</td><td>System Parameters</td></tr>
<tr><td>Description</td><td>Master switch of interrupt.
Interrupt switch is OFF by default to avoid entering interrupt before procedure initialization is finished.

There is only one task to respond all interruption signals inside the controller, and there is a fixed interrupt task NO., if deals with too much function, especially with longer code, it will cause all interruption responses become slower, even cause interruption blocking, then influence other interruption execution.
Solutions:
(1)Decrease the amount of interrupt, because many applications can be handled with scan round.
(2)For an extreme long function, call a single task to deal with the complicated interruption task, avoid blocking other interruption responses.</td></tr>
<tr><td>Grammar</td><td>INT_ENABLE=switch
<table><tr><td>Value</td><td>Description</td></tr><tr><td>0 (Default)</td><td>OFF</td></tr><tr><td>1</td><td>ON</td></tr></table></td></tr>
<tr><td>Controller</td><td>General</td></tr>
<tr><td>Example</td><td>Incorrect demonstration, interrupt blocks.
As follow, timer interrupt 0 is ON, IN(0) is 0, interrupt blocks in the line 9, there is no print result, so timer interrupt 1 can’t execute.


Correct demonstration:
When there needs to deal with a lot of codes, build a task in the interrupt, as</td></tr>
</table>

<table>
<tr><td></td><td>follow task 3, execute below procedures, print “the second interrupt”, timer interrupt 0 blocks, this has no influence on timer interrupt 1.


Relevant codes as follow:
INT_ENABLE=1 'open interrupt
TIMER_START(0,1000) 'timer 0 opens, cycle time is 1000ms
TIMER_START(1,1100) 'timer 1 opens, cycle time is 1100ms
END

GLOBAL SUB ONTIMER0() 'interrupt handler function
'build a new task to handle with complicated task to avoid blocking respond speed of other interrupt.
RUNTASK 3, MyIntHandler()
END SUB

GLOBAL SUB MyIntHandler()
DELAY 1000 'suppose a lot of blocking codes
WAIT UNTIL IN(0) <> 0
?"the first interrupt"
END SUB

GLOBAL SUB ONTIMER1() 'interrupt handler function
?"the second interrupt"
END SUB</td></tr>
<tr><td>Instructions</td><td>ONPOWEROFF, ONTIMERn</td></tr>
</table>

ONPOWEROFF--Power-Failure Interrupt SUB

Type	Interruption
Description	**Entrance of power-failure interrupt, it must be global SUB process.** Controller has only 1 power-failure interrupt. Time to execute power-failure interruption is limited, only several sentences can be written.
Grammar	GLOBAL ONPOWEROFF()

	… END SUB
Controller	General
Example	INT_ENABLE=1 dpos(0)=vr(0)　　　'read saved value when power-on, recover coordinate dpos(2)=vr(2) END GLOBAL SUB **ONPOWEROFF**() 　vr(0) = dpos(0)　　'save coordinate 　vr(1) = dpos(1) 　vr(2) = dpos(2) END SUB
Instructions	INT_ENABLE

INT_ONn—External Input Interrupt SUB

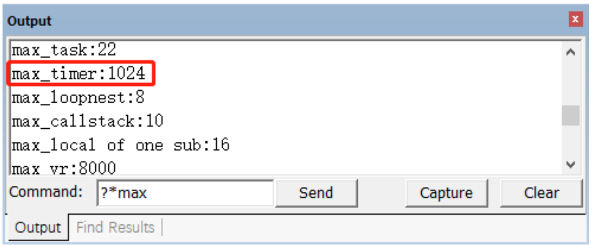
Type	Interrupt
Description	**Entrance of external input interrupt, rising edge trigger, it must be global SUB process.** Only valid in controller with firmware that supports PLC. Interrupt INPUTS: IN0-31.
Grammar	GLOBAL SUB INT_ONn()　n is input NO. 　… END SUB
Controller	Controller with firmware that supports PLC
Example	INT_ENABLE=1　　　'open interrupt END GLOBAL SUB INT_ON0()　'interrupt procedure 　print "triggered when meets rising edge of IN0" END SUB
Instructions	INT_OFFn, INT_ENABLE

INT_OFFn--External Input Interrupt SUB

Type	Interrupt
Description	**Entrance of external input interrupt, falling edge trigger, it must be global SUB process.** Only valid in controller with firmware that supports PLC. Interrupt INPUTS: IN0-31.
Grammar	GLOBAL SUB INT_OFFn()　n is input NO. 　… END SUB

Controller	Controller with firmware that supports PLC
Example	INT_ENABLE=1 'open interruption END GLOBAL SUB INT_OFF0() 'interrupt procedure print "triggered when meets falling edge of IN0" END SUB
Instructions	INT_ONn, INT_ENABLE

ONTIMERn--Timer Interrupt SUB

Type	Interrupt
Description	**Entrance of timer interruption, it must be global SUB process.** Timer number is based on controller model. Link controller through ZDevelop software, remote command send “?*max” see max_timer. 
Grammar	GLOBAL SUB ONTIMERn() n is timer NO. … END SUB
Controller	General
Example	INT_ENABLE=1 'open interruption TIMER_START(0,100) 'timer 0 open, cycle time is 100ms END GLOBAL SUB ONTIMER0() 'interrupt procedure print "ontimer0 enter" 'TIMER_START(0,100) 'select execution in cycle from sub END SUB
Instructions	INT_ENABLE, TIMER_START

INT_CYCLE—Interrupt Period Execution

Type	System parameters
Description	**Interrupt period execute BASIC functions, each SERVO_PERIOD executes once.** Valid in ZMC4XX series controller and version above 20170630.
Grammar	Command grammar: INT_CYCLE(function, taskid [, subname]) parameters:

	function: 1-start, 2-stop taskid: BASIC task NO., but BASIC itself is useless subname: SUB name executed by period, the procedure must be enough brief and refined Function grammar: var = INT_CYCLE(function, taskid) parameters: function: 3 -return status, 1-enable, 0-stop 4 -return time, the former execute time us 5 -return time, the longest execution time us 6 -the longest limitation us of return time 7 -error code when return to error situation, if BASCI interrupt function execute wrongly, errors will be set. 8 -return error line NO. taskid: used BASIC task NO.
Controller	General
Example	DIM times INT_CYCLE(1,1,intisr) END GLOBAL SUB intisr() times=times+1 MOVE_PT(1,1) 'run in every period END SUB
Instruction	INT_ENABLE

16.2 Timer Instructions

TIMER_IFEND--Timer Status

Type	System Functions
Description	**Return value to check if timer ends**
Grammar	value=TIMER_IFEND (timernum) Returned value is 0: timer is in process, timer interrupt was not executed. Returned value is 1: timer ends, timer interrupt starts to execute. Print result of TIMER_IFEND is 0 before timer starts in controller with firmware that supports PLC. Print result of TIMER_IFEND is 1 before timer starts in controller with firmware that doesn’t supports PLC.
Controller	General
Example	INT_ENABLE=1 'open interrupt

	?**TIMER_IFEND**(0) 'timer didn't start, print result is 0 'print result is 1 if doesn't support PLC. TIMER_START(0,2000) 'timer 0 starts, cycle time is 2s ?**TIMER_IFEND**(0) 'print result is 0, timer is in process DELAY(2000) ?**TIMER_IFEND**(0) 'timer ends, timer interrupt starts to execute.
Instructions	ONTIMERn, TIMER_START

TIMER_START--Open Timer

Type	System Instruction
Description	**Start system timer, only execute 1 time.**
Grammar	TIMER_START(timernum, time_ms) timernum timer NO.:0-(timer number-1) time_ms time of timer, unit is millisecond. Time 100 and above are cumulative timers.
Controller	General
Example	See sample in TIMER_IFEND
Instructions	ONTIMERn, TIMER_IFEND

TIMER_COUNT – Timer Accumulation Time

Type	System Instruction
Description	**Read accumulation time of timer, and the data remains when power off, which means it needs to clear manually.**
Grammar	value = TIMER_COUNT (timernum) timernum: timer No., 0 – the number of timers reduce 1
Controller	General
Example	INT_ENABLE=1 'enable interrupt TIMER_COUNT(0)=0 'clear TIMER_START(0,2000) 'timer 0 starts, timing 2s DELAY (2000) ?TIMER_COUNT(0) 'print cumulative time: 2000 TIMER_STOP(0) 'stop timer 0 END
Instructions	ONTIMERn, TIMER_START

TIMER_STOP--Stop Timer

Type	System Instruction

Description	**Stop system timer compulsively.**
Grammar	TIMER_STOP (timernum) timernum NO. 0-(timer number-1)
Controller	General
Example	INT_ENABLE=1 'open interruption TIMER_START(0,2000) 'timer 0 starts, cycle time is 2s ?TIMER_IFEND(0) 'print result:0, no execution DELAY(2000) ?TIMER_IFEND(0) 'print result:0, timer interrupt starts. **TIMER_STOP**(0) 'stop timer 0 ?TIMER_IFEND(0) 'print result:0, timer did not start
Instructions	ONTIMERn, TIMER_IFEND

Chapter XVII Instructions Related to Bus

17.1 Number Description

Slot NO.

Slot NO. means the interface number on motion controller, default is 0. When there are multi field bus slot ports, then use command ?*slot to check.

In instructions description, we will use SLOT to refer slot NO. for short.

When motion controller supports single bus, slot NO. is 0. When supports double bus, EtherCAT bus slot NO. is 0, RTEX bus slot is 1.

```
>>?*slot

Slot:0-ETHERCAT.
```

```
>>?*slot

Slot:0-ETHERCAT.
Slot:1-RTEX.
```

Device NO.

Device No. means all device's numbers connected on one slot position, it starts from 0 and increases in order, the total number can be checked through instruction: NODE_COUNT(slot).

In instructions description, node represents the device number.

Drive NO.

Controller can distinguish the drive connected to slot, start from 0, increase as connection order.

Drive NO. is different from Device NO., suppose 3 devices are connected to controller, the first 2 are IO expansion modules, the last is drive, then now the device number of drive node=2, the drive number is 0.

17.2 Basic Instructions

SLOT_SCAN-- Bus Scan

Type	Field Bus Instructions
Description	**Scan Field Bus.** Use RETURN to check if scan is done successfully, if it succeeds, the

	returned value is -1, or is 0. In terms of Rtex, it will report error if is fails. If the controller doesn't support field bus, then the returned value is 0. There is Returned Error if no connected devices in Rtex Controller, while no error will return in the same situation in EtherCAT controller. After scan, then use NODE to read information of connected devices, and use DRIVE instructions to configure.
Grammar	SLOT_SCAN (slot) slot: Slot No. of EtherCAT or RTEX, 0-default
Controller	Controllers with EtherCAT or Rtex.

Example	aa: 'mark aa
	SLOT_SCAN(0) 'bus scan.
	? RETURN 'print returned value,-1:success, 0:failure.
	IF RETURN THEN
	?NODE_COUNT(0) 'return connected devices number.
	ELSE
	?"scan failed"
	DELAY (1000) 'wait for 1 second.
	GOTO aa 'go to aa:, scan again.
	ENDIF

Instructions	SLOT_START, SLOT_STOP

SLOT_START--Start Field Bus

Type	Field Bus Instructions
Description	**Start Field Bus.** Use RETURN to check if Field Bus starts successfully, it will return -1 when ON, 0 means it fails. Execute after a successful slot scan: SLOT_SCAN. Set AXIS_ADDRESS, ATYPE, DRIVE_PROFILE well before execution.
Grammar	SLOT_START (slot [,opstate]) slot slot No. of EtherCAT or RTEX, 0-default. opstate EtherCAT status of beginning, 4-SAFEOP, 8-OP(default). NODE_PDOBUFF can be modified after ON, it can open SAFEOP firstly, then set initial PDO status, then open OP. This is valid in ZMC4XX series with firmware version above 20170515.
Controller	Controllers with EtherCAT or RTEX.
Example	aa: 'mark aa SLOT_SCAN(0) 'bus scan IF RETURN THEN 'start axis configuration if scan successes AXIS_ADDRESS(0)=1 'map the first drive to axis 0. ATYPE(0)=65 'axis AType 65: Position Mode DRIVE_PROFILE(0)=0 'cycle scan configuration of PDO. SLOT_START(0) 'start field bus.

	ELSE
	  ?"Scan Failed"
	  DELAY (1000) 'wait for 1 second.
	  GOTO aa 'go to aa:, scan again.
	ENDIF

Instructions	SLOT_STOP, SLOT_SCAN, NODE_PDOBUFF

SLOT_STOP--Field Bus Stops

Type	Field Bus Instructions
Description	**Field Bus stop.** Use RETURN to check if Field Bus Stops successfully, if it succeeds, the returned value is -1, or is 0. If field bus stops, axis enable will disappear.
Grammar	SLOT_STOP (slot) slot: slot NO. of EtherCAT or RTEX, 0-default.
Controller	Controllers with EtherCAT or Rtex.

Example	aa: 'markaa
	SLOT_SCAN(0) 'scan slot 0.
	IF RETURN THEN 'start axis configuration if scan succeeds
	  AXIS_ADDRESS(0)=1 'map the first dive to axis 0.
	  ATYPE(0)=65 'axis AType 65:Position Mode
	  DRIVE_PROFILE(0)=0 'cycle scan configuration of PDO.
	  SLOT_START(0) 'start field bus.
	  WHILE 1
	    IF SCAN_EVENT(IN(0))>0 THEN
	      'trigger while rising edge of In(0).
	    SLOT_STOP(0) 'stop Field Bus
	    ENDIF
	  WEND
	ELSE
	  ?"Scan Failed"
	  DELAY (1000) 'wait for 1 second.
	  GOTO aa 'go to aa:, scan again.
	ENDIF

Instructions	SLOT_START, SLOT_SCAN

SLOT_INFO – Get Bus Information

Type	EtherCAT Command
Description	**Read bus information** It reads after scanning the bus.

<table>
<tr><td>Grammar</td><td>Only Read: var = SLOT_INFO (slot, sel)
slot: slot No., default is 0
sel: information No.
<table>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>0</td><td>slot type:
SLOT_TYPE_NULL = 0,// invalid
SLOT_TYPE_ECAT = 1,// ECAT
SLOT_TYPE_RTEX = 4, // RTEX
SLOT_TYPE_HLINK = 5,// Huawei HLINK</td></tr>
<tr><td>4</td><td>Read AL state:
1--Init—initialization state
2--pre-operational—pre-operation state
4--safe-operational—safe operation state
8--operational—running state (enable)</td></tr>
<tr><td>5</td><td>Read whether disconnect is detected, it is used for redundancy mode, 0529 adds.
0—normal, 1--break</td></tr>
<tr><td>14</td><td>How many bytes of PDO sending</td></tr>
<tr><td>15</td><td>How many bytes of PDO receiving</td></tr>
<tr><td>16</td><td>How many devices in total, it must scan at first, then read</td></tr>
<tr><td>17</td><td>How many motors in total, it must scan at first, then read</td></tr>
</table></td></tr>
<tr><td>Controller</td><td>VERSION_BUILD: 240529 above version</td></tr>
</table>

?*SLOT--Print Field bus Ports

Type	EtherCAT Subsidiary Instruction
Description	**Check Field Bus Port NO. and Type.**
Grammar	?*SLOT
Controller	Controllers With Field Bus Port.
Example	?***SLOT** Print Result as Follow: Slot:0-ETHERCAT 'there is only one EherCAT port, Slot No. is 0.

?*ETHERCAT--Print EtherCAT Bus Status

Type	EtherCAT Subsidiary Instruction
Description	Use this instruction while debugging to get status of devices connected. Only valid after a successful field bus scan.
Grammar	?*ETHERCAT
Controller	Controllers with EtherCAT

Example	?***ETHERCAT**
	Result as follow:
	`Slot:0 contain 1 nodes.`
	`Lostcount:0-0.`
	`Node:0 status:1 manid:7595h productid:0h axises:1 Alstate:8 Node_profile:0.`
	`BindAxis:0 Drive_profile:0 Controlword:fh drive_status:1237h Drive_mode:8h target:ffffe067h encode:ffffe068h.`
	
	Slot 0 contain 1 nodes: 1 device is connected on slot 0
	Lostcount 0-0: the number of lost data package.
	Node: Node No. connected to device.
	Status: Node connection status, see NODE_STATUS for details
	Mainid: Manufacturer ID
	Productid: Device ID.
	Axises: Total axes of device.
	Alstate: OP Status of EtherCAT device.
	Node_profile: Profile setting of device.
	Bindaxis: Axes No. mapped to controller.
	Drive_profile: Device PDO setting.
	Controlword: Control Word.
	Drive_status: Present device status, see DRIVE_STATUS for details.
	Drive_mode: device control mode.
	Target: motor position.
	Encode: Encoder position.

Instructions	PRINT

?*RTEX--Print Rtex Status.

Type	Rtex Subsidiary Instruction
Description	Use this instruction while debugging to get status of devices connected. Only valid after a successful field bus scan.
Grammar	?*RTEX
Controller	Controllers with Rtex interface.

Example	?*RTEX
	Result as follow:
	`Slot:0 contain 1 nodes.`
	`Lostcount:0-0.`
	`Node:0 status:1 manid:616e6150h devicetype:31h axises:1 Alstate:1.`
	`BindAxis:0 Drive_profile:0 Controlword:80h drive_status:3c1h target:fffffe5ch encode:fffffe5ch.`
	Slot 0 contain 1 nodes: 1 device is connected on slot 0
	Lostcount 0-0: the number of lost data package.
	Node: Node No. connected to device.
	Status: Node connection status, see NODE_STATUS for details
	Mainid: Manufacturer ID
	Productid: Device ID.
	Axises: Total axes of device.
	Alstate: OP Status of EtherCAT device.
	Node_profile: Profile setting of device.
	Bindaxis: Axes No. mapped to controller.
	Drive_profile: Device PDO setting.
	Controlword: Control Word.
	Drive_status: Present device status, see DRIVE_STATUS for details.
	Drive_mode: device control mode.
	Target: motor position.
	Encode: Encoder position.

Instructions	PRINT

ZTEST—Check EtherCAT Bus Information

Type	EtherCAT subsidiary instruction
Description	**It can see much information while debugging.**
Grammar	Ztest(30,10,nodeid) nodeid = device No., n—(n-1)
Controller	Controllers with EtherCAT interface.
Example	**Example 1: check present PDO and key data dictionary** ztest(30,10,nodeid) nodeid = device No., 0--(n-1) >>ztest(30,10,0) TestDriver_ecat para1:10,para2:0! reg:1c12:0 value:0x1 reg:1c12:1 value:0x1600 reg:1600:0 value:0x3 reg:1600:1 value:0x60400010 reg:1600:2 value:0x607a0020 reg:1600:3 value:0x60600008 reg:1c13:0 value:0x1

	reg:1c13:1 value:0x1a00 reg:1a00:0 value:0x2 reg:1a00:1 value:0x60410010 reg:1a00:2 value:0x60640020 reg:6040:0 value:0xf reg:6041:0 value:0x1237 reg:6060:0 value:0x8 reg:6061:0 value:0x8 reg:6064:0 value:0x10ac4 reg:607a:0 value:0x10ac4 reg:603f:0 value:0x0 **Example 2: check device AL status, 1-init, 2-preop, 3-safeop, 8-op** Check all ECAT combined AL status. >>ztest(30,1) TestDriver_ecat para1:1,para2:0! al:0x8 code:0x0. alctrl:0x8 ztest(30,2,nodeid) nodeid = device NO.，0---(n-1) single AL status check. >>ztest(30,2,0) TestDriver_ecat para1:2,para2:0! al:0x8 code:0x0. alctrl:0x8 **Example 3: message loss check** >>ztest(30,12) TestDriver_ecat para1:12,para2:0! Slot:0 contain 1 nodes. Lostcount:0-0. The first data: no response numbers The second data: numbers of clock conflict **Example 4: check if support defined device** ztest (30,20,manufacuter ID, product ID, version No.) >>ztest(30,20,$41b,0,11) Id:0x41b ProductCode:0x0 version:0xb support. >>ztest(30,20,$41b,145,11) Id:0x41b ProductCode:0x91 version:0xb not support.
Instruction	PRINT

?*ZML – Print ZML Information

Type	EtherCAT Bus Instruction
Description	**Check ZML device list added in current project and the usage situation.** ZML file is the compressed file of XML file, which is used to expand functions for specific device from controllers. Valid in ECAT controllers with firmware version above 20221021.
Grammar	?*ZML
Controller	Controllers with EtherCAT interface.
Example	>>?*zml Print result: Vender:41bh id:1ab0h ver:11h used:1.
Instruction	PRINT, ZML_INFO

17.3 SDO Operational Instructions

SDO_WRITE--Write Data Dictionary

Type	Field Bus Instructions are only for EtherCAT.
Description	**Write Data Dictionary through device No. and Slot No.** Use RETURN to check if data is written successfully, -1 means it succeeds, 0 means it fails. Execute after successful devices connection and field bus scan. Only for Data Dictionary that can be written.
Grammar	SDO_WRITE (slot, node, index, subindex, type, value) slot: Slot No., 0-default node: Device No., starts from 0. index: Data Dictionary NO., add "$" before the value to indicate hexadecimal, such as $6060. subindex: subsidiary NO. 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 Value.
Controller	Controllers With EtherCAT interface.

Example	SLOT_SCAN(0)
	IF NODE_COUNT(0)>0 THEN
	SDO_WRITE(0,0,$6060,0,2,8)
	'control mode of Device 0 is 8, Position Control Mode.
	ENDIF

Instructions	SDO_WRITE_AXIS, SDO_READ

SDO_WRITE_AXIS--Write Data Dictionary

Type	Field bus instructions are only for EtherCAT.
Description	**Write through axis NO. SDO.** Use RETURN to check if data is written successfully, -1 means it succeeds, 0 means it fails. Execute after successful devices connection and field bus scan. Only for Data Dictionary that can be written.
Grammar	SDO_WRITE (axis, index, subindex, type, value) axis: axis NO. index: Data Dictionary NO., add “$” before the value to indicate hexadecimal, such as $6060. subindex: subsidiary NO. 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 Value.
Controller	Controllers with EtherCAT interface.

Example	Please use the followed sample before a EtherCAT device is connected successfully.
	SLOT_SCAN(0) 'Scan Field Bus
	IF NODE_COUNT(0)>0 THEN
	AXIS_ADDRESS(0)=1 'map the first dive to axis 0.
	ATYPE(0)=65 'axis AType 65: Position Mode
	DRIVE_PROFILE(0)=0 'cycle scan configuration of PDO.
	SDO_WRITE_AXIS(0,$6060,0,2,8)
	'control mode of axis 0 is 8, Position Control Mode.
	ENDIF

Instructions	SDO_WRITE, SDO_READ_AXIS

SDO_READ--Read Data Dictionary

Type	Field Bus Instruction is only for EtherCAT.
Description	**Read Data Dictionary through device No. and Slot No.** Use RETURN to check if data is written successfully, -1 means it succeeds, 0 means it fails. Execute after successful devices connection and field bus scan. It reads data dictionary that can be read. Please don't read and write SDO frequency.
Grammar	SDO_READ (slot, node, index, subindex, type, tablenum) slot: Slot No., 0-default node: Device No., starts from 0. index: Data Dictionary NO., add "$" before the value to indicate hexadecimal, such as $6060. subindex: subsidiary No. type: Data Type 1 Boolean 2 Integer 8 3 Integer 16 4 Integer 32 5 Unsigned 8 6 Unsigned 16 7 Unsigned 32 8 float 9 string tablenum: TABLE that saves read data directly input this command in "online command", when tablenum is -1, won't store, print directly.
Controller	Controllers with EtherCAT interface.

Example	SLOT_SCAN(0)
	IF NODE_COUNT(0)>0 THEN
	SDO_READ (0,0,$6061,0,2,0) 'read control mode of device 0, save data into table(0)
	?table(0) 'print data
	ENDIF

Instructions	SDO_READ_AXIS, SDO_WRITE

SDO_READ_AXIS--Read Data Dictionary

Type	Field Bus Instructions are only for EtherCAT.
Description	**Read Data Dictionary through Axis NO..** Use RETURN to check if data is written successfully, -1 means it succeeds,

	0 means it fails. Execute after successful devices connection and field bus scan. Only for Data Dictionary that can be read.
Grammar	SDO_READ (axis, index, subindex, type, value) axis: axis NO. index: Data Dictionary NO., add "$" before the value to indicate hexadecimal, such as $6060. subindex: subsidiary NO. type: Data Type 1 Boolean 2 Integer 8 3 Integer 16 4 Integer 32 5 Unsigned 8 6 Unsigned 16 7 Unsigned 32 tablinum: read TABLE position that saves data.
Controller	Controllers with EtherCAT interface.

Example	Please use the followed sample before a EtherCAT device is connected successfully.
	
	SLOT_SCAN(0) 'Scan Field Bus
	IF NODE_COUNT(0)>0 THEN
	AXIS_ADDRESS(0)=1 'map the first dive to axis 0.
	ATYPE(0)=65 'axis AType 65: Position Mode
	DRIVE_PROFILE(0)=0 'cycle scan configuration of PDO.
	SDO_WRITE_AXIS(0,$6060,0,2,8)
	'control mode of axis 0 is 8, Position Control Mode.
	SDO_READ_AXIS(0,$6061,0,2,0)
	'read the control mode of device No.1, data is saved into table(0)
	'print data
	ENDIF

Instructions	SDO_READ, SDO_WRITE_AXIS

17.4 Device Instructions

NODE_COUNT--Connected Device NO.

Type	Field Bus Instructions
Description	**The total number of devices connected through Bus.** Only valid after a successful field bus scan.
Grammar	Only for Read: var = NODE_COUNT (slot)

	slot: slot NO., 0-default Print directly, see example one. Use as data directly, see example two.
Controller	Controllers with EtherCAT or Rtex.
Example	**Example one** SLOT_SCAN(0) ? **NODE_COUNT**(0) 'print devices NO. connected in slot 0. **Example two** SLOT_SCAN(0) IF **NODE_COUNT**(0) = 3 THEN 'define linking device number, then axes mapping, axis atype setting and other procedure code block. ENDIF
Instructions	NODE_INFO

NODE_STATUS--Device Status

Type	Field Bus Instructions
Description	**Device status, which is valid after successful field bus scan.** BIT / Meaning 0 / Indicates if node exists,1-exist,0-not exist. 1 / Communication status,1-error,0-normal. 2 / Node Status, 1-error,0-normal. When value is 1, then bit0=1, bit1, bit2=0, device communication is normal. When value is 3, then bit0, bit1=1, bit2=0, error in device communication.
Grammar	Only for read: var = NODE_STATUS (slot, node) slot slot NO., 0-default node device NO., starts from 0.
Controller	Controllers with EtherCAT or Rtex.

Example	SLOT_SCAN(0)
	IF NODE_COUNT(0)>0 THEN
	?**NODE_STATUS**(0,0)
	'print status of device 0, value is 1, communication is normal.
	ENDIF

Instructions	NODE_INFO

NODE_AXIS_COUNT--Connected Motor NO.

Type	Field Bus Instructions
Description	**Connected motors of each device.** Only valid after successful field bus scan.

Grammar	Only for Read: var = NODE_AXIS_COUNT (slot, node) slot slot NO., 0-defaul node device NO., starts from 0.
Controller	Controllers with EtherCAT or Rtex.

Example	SLOT_SCAN(0)
	IF NODE_COUNT(0)>0 THEN
	? **NODE_AXIS_COUNT** (0,0) 'print connected device number.
	ENDIF

Instructions	NODE_INFO

NODE_IO--Device IO

Type	EtherCAT Field Bus Instructions
Description	**IO start NO. setting of device, Input and output start NO. are the same in one device.** Setting value should be times of 8. Only valid after successful field bus scan. Usually used in EIO expansion module for IO configuration, also valid in devices with IOs.
Grammar	To read: var=NODE_IO (slot, node) To write: NODE_IO(slot, node)=iobase slot slot NO., 0-default. node device NO., starts from 0.
Controller	Controllers with EtherCAT or Rtex

Example	SLOT_SCAN(0)
	IF NODE_COUNT(0)>0 THEN
	NODE_IO(0,0)=32 'set IO start NO. as 32 in device 0.
	?**NODE_IO**(0,0) 'print the IO start NO.
	ENDIF

Instructions	NODE_AIO

NODE_AIO--Device Analog

Type	Field Bus Instructions
Description	**AIO start NO. setting of device, Analog Input and output start NO. are the same in one device.** Only valid after successful field bus scan. Usually used in EIO expansion module for AIO configuration, also valid in devices with AIOs.
Grammar	To read: var=NODE_AIO (slot, node[,idir]) To write: NODE_AIO(slot, node[,idir])=Aiobase slot slot NO., 0-default. node device NO., starts from 0.

	idir choose AD or DA 0-default, set both AIN and AOUT, but only read AIN. 3-AIN 4-AOUT
Controller	Controllers with EtherCAT or Rtex

Example	SLOT_SCAN(0)
	IF NODE_COUNT(0)>0 THEN
	NODE_AIO(0,0,3)=3 'set Ain start NO. as 3 in device 0.
	?**NODE_AIO**(0,0,3) 'print AIO start NO. of device 0.
	ENDIF

Instructions	NODE_IO

NODE_INFO--Device Information

Type	EtherCAT Bus Instructions
Description	**Read information of field bus devices.** Only valid after successful field bus scanned.

<table>
<tr><td>Grammar</td><td>For reading: var=NODE_INFO (slot, node, sel, [,moduid])
slot: slot No., 0-default.
node: device No., starting from 0.
sel: information No.
Moduid: parameters can be selected, it is used when checking submodule information, parameter value is n-1, n means the number of submodules.

0-VENDER, Manufaturer No.
1-DEVICE, Device No.
2-VERSION,Version
3-ALIAS, Name to distinguish drive.
4-reserved
5-connection breaks, by BIT, 240531 adds.
6-ethernet state, by BIT, 240531 adds.

IO numbers as follow::
10- the number of IN
11- the number of OP (OUT)
12- the number of AIN (analog inputs)
13- the number of AOUT (analog outputs)
14- the bytes of PDO sending, 230823 adds
15- the bytes of PDO receiving, 230823 adds
16- the number of submodules
17- check submodule type, it needs moduid
18- lag time of scanned device (ns)
19- sync offset of EtherCAT (ns)
30- device special parameters
0x300-0x307-package losing information of Esc

Writing is valid in 4xx series controllers of fast firmware version above 20190201.
For writing: NODE_INFO (slot, node, sel) = value
slot: slot No., 0-default.
node: device No., starting from 0.
sel: information No.
<table>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>3</td><td>ALIAS, alias for distinguish from drives, if drive uses eeprom to save, this can be used to modify.</td></tr>
<tr><td>19</td><td>Sync offset iof Ecat (ns), it can be modified after bus scanned and before scan open.</td></tr>
</table></td></tr>
<tr><td>Controller</td><td>Controllers with EtherCAT or Rtex</td></tr>
</table>

Example	SLOT_SCAN(0)
	IF NODE_COUNT(0)>0 THEN
	?**NODE_INFO**(0,0,10) 'read how many IN of device 0.
	?**NODE_INFO**(0,0,0) 'read manufaturer No. of device 0.
	?**NODE_INFO**(0,0,11) 'read how many OP of device 0.
	ENDIF

Instructions	SLOT_SCAN

NODE_PROFILE--PDO Reserved Setting

Type	Field Bus Instructions
Description	**Profile setting of field bus devices.** Reserved, modification is not allowed after field bus starts.
Grammar	To Read: var= NODE_PROFILE(slot,node) To Write: NODE_PROFILE(slot, node) = iprofile[, reserve]
Controller	Controllers with EtherCAT or Rtex

NODE_PDOBUFF--PDO Setting of Specail Devices

Type	Field Bus Instructions
Description	**Support PDO of special EtherCAT devices.** Read or write PDO of devices except for IO and non-axis devices, such as power supply devices. For IO or axis based devices, there is already related axes parameters and IO instructions to access to its PDO, so this instruction is useless here. SLOT_START will read present PDO list automatically in advance and then write available present PDO values. It is better to modify the PDO list or present values of relative data dictionary through SDO before calling SLOT_START. It can be modified after SOD_START is executed. Also, set SOD_START as SAFEOP status first, then try to initilize PDO status, finally set SOD_START as OP status again.

Grammar	Command Grammar: NODE_PDOBUFF (slot, node, index, subindex, type) Function Grammar: Buff=NODE_PDOBUFF (slot, node, index, subindex ,type) Parameters: slot: slot NO., 0-default. node: device NO., starts from 0. index: Data Dictionary NO., add "$" before the value to indicate hexadecimal, such as $6060. subindex: Subsidiary NO. type: Data Type 1 Boolean 2 Integer 8 3 Integer 16 4 Integer 32 5 Unsigned 8 6 Unsigned 16 7 Unsigned 32
Controller	Controllers with EtherCAT, valid in 4 series with firmware version above 20170508.
Example	>>**NODE_PDOBUFF**(0,0, $6040, 0, 3) = 15 >>? **NODE_PDOBUFF** (0,0, $6041, 0, 3)
Instructions	SDO_WRITE, SDO_READ

NODE_PDO_WRBUFF – Offset Modify PDO

Type	Field Bus Instructions.
Description	**Modify PDO according to offset.**
Grammar	Command Grammar: NODE_PDO_WRBUFF (slot, node, offset, tableindex, size) slot: slot No., 0 – default node: device No., 0 – offse: PDO byte offset tableindex: TABLE starting No. that saves data size: the number of data bytes to be written
Controller	Controllers with EtherCAT, ZMC4XX series, firmware above 20170515.
Example	>>NODE_PDO_WRBUFF(0, 0, offset, tableindex, size)
Instructions	NODE_PDO_RDBUFF

NODE_PDO_RDBUFF – Offset Read PDO

Type	Field Bus Instructions.

Description	**Read PDO according to offset.**
Grammar	Command Grammar: NODE_PDO_RDBUFF (slot, node, offset, tableindex, size) slot: slot No., 0 – default node: device No., 0 – offse: PDO byte offset tableindex: TABLE starting No. that saves data size: the number of data bytes to be written
Controller	Controllers with EtherCAT, ZMC4XX series, firmware above 20170515.
Example	>>NODE_PDO_RDBUFF (0, 0, offset, tableindex, size)
Instructions	NODE_PDO_WRBUFF

NODE_REGWRITE – ESC Register Writing

Type	Field Bus Instructions, only for EtherCAT
Description	**ESC register writing through device No. (node) and slot No.** Check from "RETURN" value, -1: reading succeed, 0: reading failed. Please connect the device well, and scan the bus, then it can be executed. Address must be valid (address that can be written).
Grammar	Command Grammar: NODE_REGWRITE (slot, node, address, bytes, value) slot: slot No., 0 – default node: device No., 0 – address: register address, if there is "$', which means hexadecimal, for example, $980 bytes: data length, 1, 2, 4 value: data value
Controller	Controllers with EtherCAT, 220418 adds this function
Example	>>NODE_REGWRITE (0, 0, address, bytes, value)

NODE_REGREAD – ESC Register Reading

Type	Field Bus Instructions, only for EtherCAT
Description	**ESC register reading through device No. (node) and slot No.** Check from "RETURN" value, -1: reading succeed, 0: reading failed. Please connect the device well, and scan the bus, then it can be executed. Address must be valid (address that can be written).

Grammar	Command Grammar: NODE_REGREAD (slot, node, address, bytes, modbusindex) slot: slot No., 0 – default node: device No., 0 – address: register address, if there is “$’, which means hexadecimal, for example, $980 bytes: data length, 1, 2, 4 modbusindex: MODBUS register No. that saves read data, -1: not to save, print (output) directly.
Controller	Controllers with EtherCAT, 220418 adds this function
Example	>>NODE_REGREAD(0, 0, address, bytes, modbusindex)

NODE_PRESET--Device Preset

Type	EtherCAT Field Bus Instructions
Description	**Preset of field bus devices, field bus will start in advance even if there is no connected devices (drive,IO etc) after preset was done.** Modification is not allowed once field bus starts. Use NODE_STATUS to check if any devices are connected after preset. If the type of preset value doesn’t match actual value, then field bus can not start. If no preset value for added devices, and no scan process to detect them, then field bus also can not start. Valid in controllers with firmware version above 20160601.
Grammar	Command Grammar1: NODE_PRESET (slot, node, manuid, productid) Command Grammar2: NODE_PRESET (slot, -1), clear all preset. Function Grammar1: VALUE=NODE_PRESET (slot,node), check if there is preset. Function Grammar1: VALUE=NODE_PRESET (slot), check the maximum number of preset. slot: slot NO., 0-default. node: Device NO., starts from 0. manuid: Manufacturer ID, see NODE_INFO for reference. productid: Deveice ID serial NO., see NODE_INFO for reference.
Controller	Controllers with EtherCAT or Rtex
Example	**NODE_PRESET**(0,-1) 'clear previous setting. **NODE_PRESET** (0,0, $83, 5) 'set the first NODE as OMRON drive. SLOT_SCAN(0) ?"SCAN RESULT:",RETURN, "MAX", NODE_COUNT(0) 'it will show the total number of device is 1. 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
Instructions	NODE_STATUS

ZML_INFO – Check Device XML

Type	EtherCAT Bus Instructions
Description	**Check something about XML file of EtherCAT device.** ZML file is the compressed file of XML file, which is used to expand functions for specific device from controllers.
Grammar	**Function Grammar:** para = ZML_INFO(infosel, venderid, devid, [version,]) infosel: operation No. Value / Description: 0 – VENDER, manufacturer No. 1 – DEVICE, device No. 2 – VERSION, version 10 – the number of fixed IN of device 11 – the number of fixed OP of device 12 – the number of fixed AIN of device 13 – the number of fixed AOUT of device 19 – Shift time configuration, the unit is ns venderid: manufacturer ID devid: device ID version: version No., optical **command grammar:** ZML_INFO(infosel, venderid, devid, [version,]) = para Mode 19 is only supported now. Note: several devices may share one XML file, at this time, they will change at the same time when modifying several devices. XML file can be checked through venderid, devid or ?*ETHERCAT after scanned.
Controller	Controllers with EtherCAT
Example	>>?ZML_INFO(19, $418108, $9252) print: 0 Drive_Vender = NODE_INFO(Bus_Slot,i,0) 'read drive manufacturer Drive_Device = NODE_INFO(Bus_Slot,i,1) 'read device No. ZML_INFO(19, Drive_Vender,Drive_Device)=500000 'modify refresh period before scan SYSTEM_ZSET = SYSTEM_ZSET OR 128 SLOT_SCAN(Bus_Slot) LOCAL LOCAL_zmlinfo,LOCAL_nodeinfo LOCAL_zmlinfo=ZML_INFO(19, Drive_Vender,Drive_Device)

Nested infosel table:

Value	Description
0	VENDER, manufacturer No.
1	DEVICE, device No.
2	VERSION, version
10	the number of fixed IN of device
11	the number of fixed OP of device
12	the number of fixed AIN of device
13	the number of fixed AOUT of device
19	Shift time configuration, the unit is ns

	LOCAL_nodeinfo=NODE_INFO(0,0,19)
	IF LOCAL_zmlinfo = LOCAL_nodeinfo THEN
	    ?"XML Success to write in"
	ELSE
	    ?"XML Fail to write in"
	ENDIF

Instructions	SLOT_SCAN

17.5 Drive Instructions

DRIVE_MODE—Drive Mode

Type	Axis Parameters
Description	**Control Mode of Drive, Mapped data dictionary is 0x6060.** Only valid after correct ATYPE setting.(set as 65/66/67)
Grammar	To Read: var=DRIVE_MODE (axis) To Write: DRIVE_MODE (axis)= value Axis： Axis NO.
Controller	Controllers with EtherCAT or Rtex

Example	SLOT_SCAN(0)
	... 'axis enable process
	IF NODE_COUNT(0)>0 THEN
	  DRIVE_MODE(0)=8 'set axis 0 as position control mode.
	  ? **DRIVE_MODE**(0) 'print control mode value of axis 0.
	ENDIF

DRIVE_PROFILE--Drive PDO Setting

Type	EtherCAT axis Parameters
Description	**PDO Sending or Receiving setting of each axis.** Only valid after correct Atype setting. (set as 65/66/67) Consult the manufacturer for more. **EtherCAT:** When DRIVE_PROFILE=-1, it indicates that controller will follow the default PDO list in drive, only valid in controller with firmware above 20160601. If default PDO list did not contain OX6060, then not able to use datum(21) for homing. -1-default drive setting, only valid in controller with firmware above 20160601 0-default setting, csp position mode.

	{0x60400010, 0x607a0020, 0x60600008},
	//control word target position mode
	{0x60410010, 0x60640020},
	//status word position feedback.
	
	1-csp position mode + torque feedback
	{0x60400010, 0x607a0020, 0x60600008},
	// control word target position mode
	{0x60410010, 0x60640020, 0x60770010},
	// status word position feedback present torque
	
	2-csp position mode + torque feedback + latch 1up
	{0x60400010, 0x607a0020, 0x60b80010, 0x60600008},
	//control word target position probe setting, mode
	{0x60410010, 0x60640020, 0x60770010, 0x60b90010, 0x60ba0020},
	// status word, position feedback, present torque, probe status, probe position
	
	3-csp position mode + torque limit + torque feedback +rising edge of latch 1
	{0x60400010, 0x607a0020, 0x60b80010, 0x60720010, 0x60600008},
	//control word, target position, probe setting, torque limit, mode
	{0x60410010, 0x60640020, 0x60770010, 0x60b90010, 0x60ba0020},
	// status word, position feedback,present torque,probe status,probe position
	
	4-csp position mode + torque feedback + drive IO input
	{0x60400010, 0x607a0020, 0x60600008},
	//control word, target position, mode
	{0x60410010, 0x60640020, 0x60770010, 0x60fd0020},
	// status word, position feedback, present torque, drive IO input
	
	5-csp position mode + torque feedback + drive IO output + drive IO input
	{0x60400010, 0x607a0020, 0x60fe0120,0x60600008},
	//control word, target position, IO output, mode
	{0x60410010, 0x60640020, 0x60770010, 0x60fd0020},
	//status word, position feedback, present torque, drive IO input
	
	6-for special drive
	
	6-for special drive
	
	8- for special drive
	
	9- firmware above 160504
	{0x60400010,0x607a0020,0x60fe0120,0x60b80010,0x60720010,0x60600008},
	//control word, target position, IO output(32 IOs), probe setting, torque limit, mode
	{0x60410010,0x60640020,0x60770010,0x60fd0020, x60b90010,0x60ba0020},
	// status word, position feedback, present torque, drive IO input(32), probe

	status, probe position
	
	10-firmware above 160504, and support drive_fe.
	{0x60400010,0x607a0020,0x60fe0120,0x60b80010,0x60720010,0x60600008},
	//control word, target position, IO output, probe setting, torque limit, mode
	{0x60410010,0x60640020,0x60770010,0x60fd0020,0x60b90010,0x60ba0020, 0x60f40020},
	// status word, position feedback, present torque, drive IO input, probe status, probe position, drive_fe
	
	11-firmware above 160504, test special for probe
	{0x60400010, 0x607a0020, 0x60b80010, 0x60600008},
	//control word, target position, probe setting, mode
	{0x60410010,0x60640020,0x60770010,0x60b90010,0x60ba0020, 0x60bb0020, 0x60bc0020, 0x60bd0020},
	// status word, position feedback, present torque, probe status, probe position1/position2/position3/position4
	
	12-firmware above 160504, for special drive
	{0x60400010, 0x607a0020, 0x60600008},
	//control word, target position, mode
	{0x60410010, 0x60640020, 0x60fd0020},
	//status word, position feedback, drive IO input
	
	13-firmware above 160504, speed forward, acceleration feedforward.
	{0x60400010, 0x60B20010, 0x607a0020, 0x60B10020, 0x60600008},
	//control word, acceleration feedforward, target position, speed feedforward, mode
	{0x60410010, 0x60640020, 0x60770010, 0x60fd0020, 0x606c0020},
	// status word, position feedback, present torque, IO, actual speed
	
	17-firmware above 160504, switchable mode: csp/csv/cst
	{0x60400010,0x60710010,0x60ff0020,0x607a0020,0x60b80010,0x60720010,0x60600008},
	//control word, cyclic torque, cyclic speed, target torque, target position, probe mode, torque limit, mode
	{0x60410010,0x60770010,0x60640020,0x60fd0020, 0x60b90010, 0x60ba0020, 0x60bb0020},
	//status word, present torque, position feedback, IO, probe status, probe position 1/position 2/
	
	18-firmware above 160504, switchable mode: csp/csv/cst + torque feedback read
	{0x60400010,0x60710010,0x60ff0020,0x607a0020,0x60b80010,0x60720010, 0x60600008},
	//control word, cyclic torque, cyclic speed, target position, probe setting, torque limit, mode
	{0x60410010,0x60770010,0x60640020,0x60fd0020, 0x60b90010,

	0x60ba0020, 0x60bb0020, 0x60bc0020, 0x60bd0020},
	//status word, present torque, position feedback, IO, probe status, probe position 1/ position 2 / position 3 / position 4.
	
	20-firmware above 160504, csp position + csvspeed
	{0x60400010, 0x60ff0020, 0x607a0020, 0x60600008},
	//control word, target speed, target position, mode
	{0x60410010, 0x60640020},
	// status word, position feedback
	
	21-firmware above 160504, csp position + csvspeed + torque feedback
	{0x60400010, 0x60ff0020, 0x607a0020, 0x60600008},
	//control word, target speed, target position, mode
	{0x60410010, 0x60640020, 0x60770010},
	// status word, position feedback, present torque
	
	22-firmware above 160504, csp position + csvspeed + torque feedback + rising edge of latch 1(color mark triggered)
	{0x60400010, 0x60ff0020, 0x607a0020, 0x60b80010, 0x60600008},
	//control word, target speed, target position, probe setting, mode
	{0x60410010, 0x60640020, 0x60770010, 0x60b90010, 0x60ba0020},
	//status word, position feedback, present torque, probe status, probe position
	
	23-firmware above 160504, csp position + csvspeed + torque feedback + rising edge of latch 1 + torque limit
	{0x60400010,0x60ff0020,0x607a0020,0x60b80010,0x60720010, 0x60600008},
	//control word, target speed, target position, mode, probe setting, torque limit, mode
	{0x60410010, 0x60640020, 0x60770010, 0x60b90010, 0x60ba0020},
	//status word, position feedback, present torque, probe status, probe position
	
	24-firmware above 160504, csp position + csv speed + IO input + position+ torque feedback
	{0x60400010, 0x60ff0020, 0x607a0020, 0x60600008},
	//control word, target speed, target position, mode
	{0x60410010, 0x60640020, 0x60770010, 0x60fd0020},
	//status word, position feedback, present torque, drive IO input
	
	25-firmware above 160504, csp position + csv speed + IO input +position+ torque feedback
	{0x60400010, 0x60ff0020, 0x607a0020, 0x60fe0120,0x60600008},
	//control word, target speed, target position, IO output, mode
	{0x60410010, 0x60640020, 0x60770010, 0x60fd0020},
	// status word, position feedback, present torque, drive IO input
	
	30-firmware above 160504, csp position + cst torque
	{0x60400010, 0x60710010, 0x607a0020, 0x60600008},
	//control word, target torque, target position, mode

	{0x60410010, 0x60640020},
	//status word, position feedback
	
	31-firmware above 160504, csp position + cst torque + torque feedback
	{0x60400010, 0x60710010, 0x607a0020, 0x60600008},
	//control word, target torque, target position, mode
	{0x60410010, 0x60640020, 0x60770010},
	//status word, position feedback, present torque
	
	32-firmware above 160504, csp position + cst torque + torque feedback + rising edge of latch 1
	{0x60400010, 0x60710010, 0x607a0020, 0x60b80010, 0x60600008},
	//control word, target torque, target position, probe setting, mode
	{0x60410010, 0x60640020, 0x60770010, 0x60b90010, 0x60ba0020},
	//status word, position feedback, present torque, probe status, probe position
	
	33-firmware above 160504, csp position + cst torque + torque feedback + rising edge of latch 1 + torque limit
	{0x60400010,0x60710010,0x607a0020,0x60b80010,0x60720010,0x60600008}
	//control word, target torque, target position, probe setting, torque limit, mode
	{0x60410010, 0x60640020, 0x60770010, 0x60b90010, 0x60ba0020},
	//status word, position feedback, present torque, probe status, probe position
	
	34-firmware above 160504, csp position + cst torque + torque feedback +drive IO input
	{0x60400010, 0x60710010, 0x607a0020, 0x60600008},
	//control word, target torque, target position, mode
	{0x60410010, 0x60640020, 0x60770010, 0x60fd0020},
	//status word, position feedback, present torque, drive IO input
	
	35-firmware above 160504, csp position + cst torque + torque feedback + IO input +IO output
	{0x60400010, 0x60710010, 0x607a0020, 0x60fe0120,0x60600008},
	//control word, target torque, target position, IO output, mode
	{0x60410010, 0x60640020, 0x60770010, 0x60fd0020},
	//status word, position feedback, present torque, drive IO input
	
	Rtex field bus
	0-No dirve IO mapping
	1-with drive IO mapping (set start address through DRIVE_IO)

Grammar	To read: var= DRIVE_ PROFILE(axis) To write: DRIVE_ PROFILE(axis)= value axis: axis NO.
Controller	Controllers with EtherCAT or Rtex

Example	SLOT_SCAN(0)
	IF NODE_COUNT(0)>0 THEN
	AXIS_ADDRESS(1)=1
	ATYPE(1)=65
	DRIVE_ PROFILE(1)=-1 'set 1PDO as -1, default setting
	? **DRIVE_ PROFILE**(1) 'print PDO setting of axis 1
	ENDIF

Instructions	ATYPE, NODE_PROFILE

DRIVE_CW_MODE--Drive Setting

Type	Axes Parameters
Description	**Drive sets parameters.** Only valid after correct Atype setting. (set as 65/66/67) For convenience, some version can operate this directly. Note: don't modify control word of RTEX freely. 0-Controller will adjust DRIVE_CONTROLWORD automatically, now DRIVE_CONTROLWORD is invalid in this situation. 1-Allow to set DRIVE_CONTROLWORD by manual in this situation.
Grammar	To read: var=DRIVE_CW_MODE(axis) To write: DRIVE_CW_MODE(axis)=value axis: axis NO.
Controller	Controllers with EtherCAT or RTEX
Instructions	ATYPE, DRIVE_CONTROLWORD

DRIVE_CONTROLWORD--Drive Control Word

Type	Axis Parameters
Description	**Drive control word, set as per bit.** Only valid after correct ATYPE setting. (set as 65/66/67) For EtherCAT based drive, the mapped data dictionary is 0x6040. This parameter will change automatically as per the WDOG / AXIS_ENABLE when Atype of controller is 65, in order to enable the drive. Main bits setting as follow, see relevant drive manual for details.

Command	bits of the controlword					PDS Transitions
	bit 7 fault reset	bit 3 enable operation	bit 2 quick stop	bit 1 enable voltage	bit 0 switch on	
Shutdown	0	–	1	1	0	2, 6, 8
Switch on	0	0	1	1	1	3
Switch on + Enable operation	0	1	1	1	1	3+4 (*1)
Enable operation	0	1	1	1	1	4, 16
Disable voltage	0	–	–	0	–	7, 9, 10, 12
Quick stop	0	–	0 (*2)	1	–	7, 10, 11
Disable operation	0	0	1	1	1	5
Fault reset		–	–	–	–	15

For Rtex based drive

This parameter will be set automatically, if needs to set by manual, then set DRIVE_CW_MODE as 1 first. Don't modify freely, bit meaning as following, see Panasonic Rtex manual for details.

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Servo_On	0	0	Gain_SW	TL_SW	HM_Ctrl	0	0
bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
Hard_Stop	Smooth_Stop	Pause	0	SL_SW	0	EX-OUT2	EX-OUT1

Grammar

To Read: var=DRIVE_CONTROLWORD(axis)

To Write: DRIVE_CONTROLWORD(axis)=value

axis: axis NO.

Controller

Controllers with EtherCAT or Rtex

Example

Example one: EtherCAT

```
SLOT_SCAN(0)
IF NODE_COUNT(0)>0 THEN
    AXIS_ADRESS(0)=1
    ATYPE(0)=65                         'position control mode
    DRIVE_PROFILE(0)=0                  'set PDO as 0
    DRIVE_CONTROLWORD(0)=128            'servo errors clear
    DELAY (100)
    DRIVE_CONTROLWORD(0)=6              'servo shutdown
    DELAY (100)
    DRIVE_CONTROLWORD(0)=15             'servo switch on
ENDIF
```

Example two: Rtex

```
SLOT_SCAN(0)
IF NODE_COUNT(0)>0 THEN
    AXIS_ADRESS(0)=1
    ATYPE(0)=50                         'position control mode
    DRIVE_PROFILE(0)=1                  'servo with IO
```

	DRIVE_CW_MODE=1 'set control data word by manual. **DRIVE_CONTROLWORD**(0)=128 'servo enable. ENDIF
Instructions	ATYPE, DRIVE_CW_MODE

DRIVE_STATUS--Drive Status

Type	Axis Status
Description	**Check drive status as per the bit value.** Only Valid after correct Atype Setting. (set EtherCAT based drive as 65/66/67, set RETX based drive as 50/51/52) **For EtherCAT based drive, the relevant data dictionary(PDO) is 6041.** See the drive manual for details.

Status Word	PDO state	
xxxx xxxx x0xx 0000 b	not ready to switch on	initialization, not done
xxxx xxxx x1xx 0000 b	switch on disabled	initialization, done
xxxx xxxx x01x 0001 b	ready to switch on	main power is off
xxxx xxxx x01x 0011 b	switch on	servo enable off/servo prepared
xxxx xxxx x01x 0111 b	operation enabled	servo enable on
xxxx xxxx x00x 0111 b	quick stop active	stop rapidly
xxxx xxxx x0xx 1111 b	fault reaction active	abnormal (alarm) judge
xxxx xxxx x0xx 1000 b	fault	abnormal (alarm) status

For Rtex Based Drive

Status Dictionary of Rtex drive as follow, see related Panasonic Rtex manual for details (chapter 4-2-3.)

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Servo_Active	Servo_Ready	Alarm	Warning	Torque_Limited	Homing_Complete	In_Progress	In_Position
bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
SI-MON5/E-STOP	SI-MON4/EX-SON	SI-MON3/EXT3/STOP	SI-MON2/EXT2/RET	SI-MON1/EXT1	HOME	POT/NOT	NOT/POT

Grammar	Only for read: toq = DRIVE_STATUS (axis) axis: axis NO.
Controller	Controllers with EtherCAT or Rtex

Example	SLOT_SCAN(0)
	IF NODE_COUNT(0)>0 THEN
	ATYPE(0)=65 'position control mode
	? **DRIVE_STATUS** (0) 'print device status of axis 0
	ENDIF

Instructions	ATYPE, DRIVE_PROFILE

DRIVE_IO--Drive IO Setting

<table>
<tr><td>Type</td><td>Axis Parameters</td></tr>
<tr><td>Description</td><td>Configure the IO start NO. of drive, number of input and output are the same.
Only valid when DRIVE_PROFILE setting supports Drive IOs reading or writing.</td></tr>
<tr><td>Grammar</td><td>To read: var= DRIVE_IO (axis)
To write: DRIVE_IO (axis)=value
axis: Axis NO.

For inputs and outputs function and address of EtherCAT fieldbus servo, see data dictionary 60FD, 60FE for reference. (attention: some manufacturers don’t set standard protocol address, please see relevant drive manual)

Rtex Fieldbus Servo inputs and outputs function related controller address=DRIVE_IO+NO.
<table>
<tr><td>DRIVE_IO and below NO.</td><td>Servo function</td></tr>
<tr><td colspan="2">Inputs</td></tr>
<tr><td>0</td><td>NOT/POT</td></tr>
<tr><td>1</td><td>POT/NOT</td></tr>
<tr><td>2</td><td>HOME</td></tr>
<tr><td>3</td><td>SI-MON1/EXT1</td></tr>
<tr><td>4</td><td>SI-MON2/EXT2</td></tr>
<tr><td>5</td><td>SI-MON3/EXT3</td></tr>
<tr><td>6</td><td>SI-MON4/EX-SON</td></tr>
<tr><td>7</td><td>SI-MON5/E-STOP</td></tr>
<tr><td colspan="2">Outputs</td></tr>
<tr><td>0</td><td>EX-OUT1</td></tr>
<tr><td>1</td><td>EX-OUT2</td></tr>
</table></td></tr>
<tr><td>Controller</td><td>Controllers with EtherCAT or Rtex</td></tr>
<tr><td>Example</td><td>SLOT_SCAN(0)
...   'axis enable process
IF NODE_COUNT(0)>0 THEN
    DRIVE_IO(1)=32   'start IO NO. of axis1(drive 1) is set as 32,
    DIM var   'define variable.
    var=DRIVE_IO(1)   'assign start NO. of axis 1 to var.
    ?var   'print IO start NO. of axis 1. Directly.
ENDIF</td></tr>
<tr><td>Instructions</td><td>NODE_IO, DRIVE_PROFILE</td></tr>
</table>

DRIVE_TORQUE--Drive Torque

Type	EtherCAT axis status
Description	**Current drive torque value.** Only valid after correct ATYPE and DRIVE_PROFILE setting.
Grammar	Only for read: var=DRIVE_TORQUE(axis) axis: axis NO.
Controller	Controllers with EtherCAT or Rtex

Example	SLOT_SCAN(0)
	IF NODE_COUNT(0)>0 THEN
	ATYPE(0)=65 'position control mode
	DRIVE_PROFILE(0)=1 'set PDO mode as 1, with torque feedback.
	? **DRIVE_TORQUE** (0) 'print torque of axis 0.
	ENDIF

Instructions	ATYPE, DRIVE_PROFILE

DRIVE_FE--Drive Error

Type	Axis Status
Description	**Read present error overrun of drive, mapped object dictionary is 0x60F4**. Only valid after correct ATYPE setting.(set as 65/66/67) For ZMC408SCAN, valid in ATYPE=22 and with MPOS, then return drive receive and MPOS deviation.
Grammar	Only for read: var=DRIVE_FE (axis) axis: Axis NO.
Controller	Controllers with EtherCAT or Rtex

Example	SLOT_SCAN(0) 'field bus scan.
	IF NODE_COUNT(0)>0 THEN
	AXIS_ADDRESS(0)=1 'assign the first axis as axis 0
	ATYPE(0)=65 'axis Type as 65, position control mode.
	DRIVE_PROFILE(0)=0
	'set PDO message mode through DRIVE_PROFILE.
	?DRIVE_FE (0) 'read following error value of axis 0.
	ENDIF

Instructions	DRIVE_FE_LIMIT

DRIVE_FE_LIMIT--Drive Error Limit

Type	Parameters of EtherCAT based axis
Description	**Set present error overrun of drive.**

	Only Valid after correct Atype Setting. (set as 65/66/67) Reserved.
Grammar	To Read: var=DRIVE_FE_LIMIT(axis) To Write: DRIVE_FE_LIMIT(axis)= value axis: axis NO.
Controller	Controllers with EtherCAT or Rtex
Instructions	DRIVE_FE

DRIVE_CLEAR--Alert Clear

Type	Field Bus Instructions
Description	**Operate present BASE axis, clear alert of drive.** Return succeeds or not through RETURN. Error 6015 will come when use this instruction if there is no error in drive, but no effect on procedure execution.
Grammar	BASE(axis NO.) DRIVE_CLEAR(para) 0 Clear present alert 1 Clear history alert 2 Clear external input alert
Controller	Controller with EtherCAT or RETX interface.

Example	SLOT_SCAN(0)
	IF NODE_COUNT(0)>0 THEN
	BASE(0) 'choose axis0.
	DRIVE_CLEAR(0) 'clear present alert.
	ENDIF

Instructions	DRIVE_READ, DRIVE_WRITE

DRIVE_READ--Read Parameters

Type	Field Bus Instructions, only valid in Rtex controller.
Description	**Operate Base axis, read drive parameters.** Check value of RETURN to judge if the operation is done successfully or not.
Grammar	DRIVE_READ(para [,vr_index]) 1: Parameters class*256+ Parameters NO. (Pr7.20=7*256+20) 2: Parameters=130, read latch status, BIT0 and BIT1 indicate status of 2 channels. 3: Parameters=$10000+(ssid), read information of Rtex drive system, string type data will be saved in VRSTRING. 4: Parameters=$20000+(Alert Code)+($1000*index), read alert information. 5: Parameters = $30000 + (monitor code) + ($1000*index), read monitor information.

vr_index: save the read data in VR, if this parameter is omiited, then print on the output box directly.

Normal parameters as follow:

1.Servo Parameters

Parameter	function	value
Pr0.00	Set motor turn direction	0: CW 1: CCW
Pr0.01	Control mode setting	Set as 0: half-closed control
Pr0.08	One round pulse amount	0-8388608 (as per actual motor)
Pr0.09	Gear ratio molecule set	0-1073741824
Pr0.10	Gear ratio denominator set	1-1073741824
Pr4.01	Positive limit position signal setting	OFF:00818181h (8487297） ON: 00010101h (65793)
Pr4.02	Negative limit position signal setting	OFF:00828282h (8553090） ON: 00020202h (131586)
Pr4.03	Home signal setting	OFF:00A2A2A2h (10658466） ON: 00222222h (2236962)

2.Rtex Communication Parameters

Pr7.20	Rtex communication period	-1： make Pr.91 set take effect. 3： 0.5ms 6： 1.0ms
Pr7.21	Rtex instruction update period ratio	1： 1 times 2： 2 times
Pr7.91	Rtex communication period expansion	62500 ns 125000 ns 250000 ns 500000 ns 1000000 ns 2000000 ns

3.Syestem ID Command

SSID	Definition
$01	Manufacturer name
$05	Device type
$12	Drive type NO.
$13	Drive List NO.
$14	Drive software version
$15	Drive type
$22	Motor type NO.
$23	Motor List NO.

4.Alarm Command

Function code	Function	Index
$000	Read present alert / alert record	0: alert code of this time 1: alert code of former time 2: alert code of former two times ... 14: alert code of 14 times before
$001	Clear present alert	0: clear alert of this time
$011	Clear all alerts	0: clear alert record
$021	Clear errors of external distance sensor	0: clear latch errors through external distance sensor of serial communication type. Then, please disconnect control power and restart.

5. Monitor

Function code	Function	Index
$01	Position deviation, instruction unit	0: instruction deviation after filtering
$02	Encoder resolution ratio, pulse/round	0: motor encoder resolution ratio
$04	Instruction position, instruction unit	0: internal instruction position after filtering
$05	Actual speed, Pr7.25	0: actual speed of motor
$06	Internal instruction torque, 0.1%	0: arrived motor instruction torque
$07	actual speed, instruction unit	0: actual position of motor
$09	Latch position 1, instruction unit	0: actual speed of latch CH1 motor
$0A	Latch position 2, instruction unit	0: actual speed of latch CH2 motor

Controller

Controllers with Rtex

Example

Only valid when field bus starts successfully

Example one

```
IF NODE_COUNT(0)>0 THEN
    BASE(0)                         'choose axis 0
    DRIVE_READ(7*256+11,0)
                              'read value of parameter Pr7.11, save in Vr(0)
    ?vr(0)                          'print
ENDIF
```

Example two

	IF NODE_COUNT(0)>0 THEN
	    BASE(0) 'choose axis 0
	    DRIVE_READ($10000+$01) 'read vendor name, print.
	ENDIF

Instructions	DRIVE_WRITE, DRIVE_CLEAR

DRIVE_WRITE--Write Parameters

Type	Field Bus Instructions, only valid in Rtex controller.
Description	**Operate Base axis, read drive parameters.** Check value of RETURN to judge if the operation is done successfully or not.
Grammar	DRIVE_WRITE(para [,vr_index]) 1: Parameters class*256+ Parameters NO. (Pr7.20=7*256+20) 2: Special parameters=128, here write EEPROM (now value=1) 3: Special parameters =$40000+typecode+(set value*256), use homing latch position function of drive value: parameters' value

1.Servo Parameters

Parameter	function	value
Pr0.00	Set motor turn direction	0: CW 1: CCW
Pr0.01	Control mode setting	Set as 0: half-closed control
Pr0.08	One round pulse amount	0-8388608 (as per actual motor)
Pr0.09	Gear ratio molecule set	0-1073741824
Pr0.10	Gear ratio denominator set	1-1073741824
Pr4.01	Positive limit position signal setting	OFF:00818181h (8487297) ON: 00010101h (65793)
Pr4.02	Negative limit position signal setting	OFF:00828282h (8553090) ON: 00020202h (131586)
Pr4.03	Home signal setting	OFF:00A2A2A2h (10658466) ON: 00222222h (2236962)
Relative torque:		
Pr0.13	The first torque limit	0~500%
Pr5.21	Torque limit selection When torque controls, Pr0.13 is fixed.	Please see below:

Set value	TL_SW=0		TL_SW=1	
	negative	positive	negative	positive
0, [1]	Pr0.13			
2	Pr5.22	Pr0.13	Pr5.22	Pr0.13

<table>
<tr><td></td><td colspan="2"><table><tr><td>3</td><td colspan="2">Pr0.13</td><td colspan="2">Pr5.22</td></tr><tr><td>4</td><td>Pr5.22</td><td>Pr0.13</td><td>Pr5.22</td><td>Pr5.25</td></tr></table></td></tr>
<tr><td>Pr5.22</td><td>The second torque limit</td><td>0~500%</td></tr>
<tr><td>Pr5.25</td><td>torque limit in positive</td><td>0~500%</td></tr>
<tr><td>Pr5.26</td><td>torque limit in negative</td><td>0~500%</td></tr>
<tr><td colspan="3">Relative speed:</td></tr>
<tr><td>Pr3.12</td><td>Acceleration time setting</td><td>0~10000ms (reach 1000.r/min)</td></tr>
<tr><td>Pr3.13</td><td>Deceleration time setting</td><td>0~10000ms (reach 1000.r/min)</td></tr>
<tr><td>Pr3.14</td><td>S acceleration and deceleration time setting</td><td>0~10000ms</td></tr>
<tr><td>Pr3.17</td><td>Speed limit selection: select speed limit value mode when torque controls</td><td><table><tr><td rowspan="2">Set value</td><td colspan="2">SL_SW</td></tr><tr><td>0</td><td>1</td></tr><tr><td>[0]</td><td colspan="2">Pr3.21</td></tr><tr><td>1</td><td>Pr3.21</td><td>Pr3.22</td></tr></table>When set as 1, select it as SL_SW value of RTEX communication instruction.</td></tr>
<tr><td>Pr3.21</td><td>Speed limit value 1</td><td>0~20000.r/min</td></tr>
<tr><td>Pr3.22</td><td>Speed limit value 2</td><td>0~20000.r/min</td></tr>
</table>

2.Rtex Communication Parameters

Pr7.20	Rtex communication period	-1: make Pr.91 set take effect. 3: 0.5ms 6: 1.0ms
Pr7.21	Rtex instruction update period ratio	1:1 times 2: 2 times
Pr7.91	Rtex communication period expansion	62500 ns 125000 ns 250000 ns 500000 ns 1000000 ns 2000000 ns

3.Homing Latch Mode

typecode	Description
$50	Monitor under position latch status
$51	Position latch 1 starts
$52	Position latch 2 starts
$53	Position latch 1 and 2 start
$54	Position latch 1 stops
$58	Position latch 2 stops
$5c	Position latch 1 and 2 stop

4.Set value

Set value = latch position 1 setting + ($10* latch 2 setting)

	<table><tr><td>bit7</td><td>bit6</td><td>bit5</td><td>bit4</td><td>bit3</td><td>bit2</td><td>bit1</td><td>bit0</td></tr><tr><td colspan="4">LATCH_SEL2</td><td colspan="4">LATCH_SEL1</td></tr></table><table><tr><td>Latch 1/2 setting</td><td>Description</td></tr><tr><td>$0</td><td>Z Phase signal trigger</td></tr><tr><td>$1</td><td>EXT1 logic rising edge</td></tr><tr><td>$2</td><td>EXT2 logic rising edge</td></tr><tr><td>$3</td><td>EXT3 logic rising edge</td></tr><tr><td>$9</td><td>EXT1 logic falling edge</td></tr><tr><td>$10</td><td>EXT2 logic falling edge</td></tr><tr><td>$11</td><td>EXT3 logic falling edge</td></tr></table>
Controller	Controllers with Rtex

Example	Only valid when field bus starts successfully
	Example one
	IF NODE_COUNT(0)>0 THEN
	BASE(0) 'choose axis 0
	DRIVE_READ(7*256+11,0)
	'read value of parameter Pr7.11, save in Vr(0)
	?vr(0) 'print
	ENDIF
	
	Example two
	IF NODE_COUNT(0)>0 THEN
	BASE(0) 'choose axis 0
	DRIVE_READ($10000+$01) 'read vendor name, print.
	ENDIF

Instructions	DRIVE_READ, DRIVE_CLEAR

Chapter XVIII ZHD Teaching Box

18.1 Teaching Box Commands

LCD_CONNECT – LCD No. Setting

Type	Teaching Box Commands
Description	This command is used when controller and teaching box communicates successfully, and it is operated in teaching box terminal. Set the other side controller's LCD (HMI) No. that is to be connected, default is 0. When controller supports multi-HMI, it can be set as other values. After configuration, teaching box needs restarting, then it can be connected.
Grammar	VAR1=LCD_CONNECT LCD_CONNECT=VAR1
Example	>> LCD_CONNECT=1 >> LCD_CONNECT=0

IP_CONNECT – IP Connection

Type	Teaching Box Commands
Description	This command is used when controller and teaching box communicates successfully, and it is operated in teaching box terminal. Set the other side controller's IP that is to be connected. It will reconnect after each time modification.
Grammar	IP_CONNECT = dot.dot.dot.dot
Example	>> LCD_CONNECT=1 >> LCD_CONNECT=0

IP_ADDRESS – IP Address

Type	Teaching Box Commands
Description	This command is used when controller and teaching box communicates successfully, and it is operated in teaching box terminal. Specify teaching box's IP address, the default is 192.168.0.10
Grammar	IP_ADDRESS = dot.dot.dot.dot
Example	>> IP_ADDRESS = 192.168.0.14

IP_GATEWAY – IP Gateway

Type	Teaching Box Commands
Description	This command is used when controller and teaching box communicates successfully, and it is operated in teaching box terminal. Specify teaching box's IP gateway, the default is 192.168.0.1
Grammar	IP_GATEWAY = dot.dot.dot.dot
Example	>> IP_GATEWAY = 192.168.0.1

IP_NETMASK – IP Mask

Type	Teaching Box Commands
Description	This command is used when controller and teaching box communicates successfully, and it is operated in teaching box terminal. Specify teaching box's IP mask, the default is 255.255.0.0
Grammar	IP_NETMASK = dot.dot.dot.dot
Example	>> IP_NETMASK = 255.255.0.0

18.2 Controller Commands

LCD_LEDSTATE – LED State Setting

Type	System Screen Parameters
Description	This command is used when controller and teaching box communicates successfully, and it is operated in controller terminal. Set LED state on teaching box, it is set by BIT, default is 1, and turn on the LED. It needs both firmware support of controller and teaching box. Valid in VERSION_BUILD = 230801 or above.
Grammar	VAR = LCD_LEDSTATE (lcdnum) LCD_LEDSTATE (lcdnum) = VAR lcdnum: controller LCD (HMI) No., default is 0
Controller	Controller that supports ZHMI (RTHMI)
Example	LCD_LEDSTATE(0) = 1

LCD_WDOGTIME – Time Setting for Screen Power-Off

Type	System Screen Parameters
Description	This command is used when controller and teaching box communicates successfully, and it is operated in controller terminal. Set the time to process screen power-down problem, the unit is ms. Press the emergency stop switch automatically (physical key No. is 5) when operating this time and there is no communication, when it is 0, this function is OFF. Valid in 5XX series with firmware version 20180404 or above, and valid in 4XX series with firmware version 20170721 or above.
Grammar	LCD_WDOGTIME (lcdnum) = time lcdnum: controller LCD (HMI) No., default is 0 time: time, the unit is ms
Controller	Controller that supports ZHMI (RTHMI)
Example	LCD_WDOGTIME(0) = 100

Chapter XVIIII MotionRT Commands

19.1 MotionRT Commands

CARD_INFO – Read & Write Control Card Info

<table>
<tr><td>Type</td><td>Control Card Commands</td></tr>
<tr><td>Description</td><td>Read / Write control card information.</td></tr>
<tr><td>Grammar</td><td>
--How to Read--
var = CARD_INFO (cardnum, sel)
cardnum: sub card No., 0 ~ N ~ 1 (N: the number of control cards. When no subcard, N is 1)
sel: information No.
<table>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>0</td><td>The totals of sub-card that are returned, now cardnum should be 0.</td></tr>
<tr><td>1</td><td>DEVICE, device No., hardid.</td></tr>
<tr><td>2</td><td>VERSION, version</td></tr>
<tr><td>3</td><td>Sub-card dial code, it is only valid in PCI.</td></tr>
<tr><td>4</td><td>Reserved</td></tr>
<tr><td>5</td><td>Unique No. of sub-card, the last PCI sub-card’s unique No. is the unique No. of RT.</td></tr>
<tr><td>6</td><td>Reserved for special No.</td></tr>
<tr><td>7</td><td>Reserved</td></tr>
<tr><td>8</td><td>IO offset, it automatically sort by default when power on.</td></tr>
<tr><td>9</td><td>AIO offset</td></tr>
<tr><td>10</td><td>The number of IN</td></tr>
<tr><td>11</td><td>The number of OP</td></tr>
<tr><td>12</td><td>The number of AIN</td></tr>
<tr><td>13</td><td>The number of AOUT</td></tr>
<tr><td>14</td><td>Reserved</td></tr>
<tr><td>15</td><td>The number of HW OP</td></tr>
<tr><td>16</td><td>The number of pulse axes</td></tr>
<tr><td>17</td><td>The number of encoders</td></tr>
<tr><td>18</td><td>The number of ECAT, 0 / 1</td></tr>
<tr><td>19</td><td>The number of scan, 0 / 1</td></tr>
<tr><td>20</td><td>The number of 3D SCAN</td></tr>
<tr><td>21</td><td>Support power-down, the number of VR, usually XPCI doesn’t bring with VR.</td></tr>
<tr><td>22</td><td>VR offset, sorting automatically</td></tr>
<tr><td>23</td><td>The number of PWM</td></tr>
<tr><td>24</td><td>PWM starting No. on sub-card</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>--How to Write--
CARD_INFO (cardnum, sel) = value
cardnum: sub card No., 0 - default
sel: information No.
<table>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>8</td><td>IO offset, it automatically sort by default when power on.
Note: value must be multiple of 8</td></tr>
<tr><td>9</td><td>AIO offset, it automatically sort by default when power on.</td></tr>
</table></td></tr>
<tr><td>Example</td><td>GLOBAL value
valu=CARD_INFO (0,1) ‘read control card device No.
?valu
end</td></tr>
</table>

CARD – Print Sub-Card Info

<table>
<tr><td>Type</td><td>Control Card Commands</td></tr>
<tr><td>Description</td><td>Print sub-card information (send command in ZDevelop / RTSys).</td></tr>
<tr><td>Grammar</td><td>?*CARD
HardId: hardware version
Pul: pulse
In: the number of IN
Op: the number of OP
Ad: the number of AIN
Da: the number of AOUT
Pwm: the number of Pwm
flash: flash size
size: ROM size
serial: card No.
license: parameters configuration</td></tr>
<tr><td>Example</td><td>?*CARD
>>?*CARD

Card0: is XPci, HardId:7133-0 Pul:4 In:16 Op:16 Ad:0 Da:0 Pwm:4
flash:ef4016h size:4194304 serial:221090003 license:AX64 MO8 ZV HW</td></tr>
</table>

REG_CARD – Latch Selection

<table>
<tr><td>Type</td><td>Control Card Commands</td></tr>
<tr><td>Description</td><td>Select latch IN. When several sub-card support latching, it can switch.
Used together with REGINPUTS, REGIST. After calling REGIST, it can switch immediately.</td></tr>
</table>

	Latch Position: REG_POSE, REG_POSF, REG_POSG. REG_POSH Latch Channel: MARKE, MARKF, MARKG, MARKG (it can be expanded to 8 channels latching at most)
Grammar	REG_CARD = value Set REG_CARD = 1 to use axis specified latch. When other values are set, which means it specifies the card No. for latching.

SLOT_SLAVE – EtherCAT Redundancy Configuration

Type	Special parameter
Description	Configure two ECAT channels as one channel, then support hot redundancy backup. Notes: 1. It is set when bus stops. 2. After setting, it only can operate islot 1, that is, islot 2 can't be operated. 3. When bus initializing, the wiring must be normal, otherwise, initialization will fail, or appear error of scanned devices numbers, you can check the configuration by "?SLOT_SLAVE(islot1).
Grammar	SLOT_SLAVE(islot1)=islot2 Islot1: master bus, must connect to the first device's IN Islot2: slave bus, must connect to the last device's OUT
controller	VPLC7XX controllers whose firmware version should be above 240520, multi-card is valid.
Example	SLOT_SLAVE(0)=1

Chapter XX Commands of Local Slave Interface

ZMIO_CONFIG – Set/Get Analog Range & Channel State

<table>
<tr><td>Type</td><td>Local slave level interface RS485 bus command for XPLC300 / ZMC432M.</td></tr>
<tr><td>Description</td><td>Read or configure expansion sub-module’s AD/DA channel switch states and range types.</td></tr>
<tr><td>Grammar</td><td>To read: var = ZMIO_CONFIG (sel, moduleid)
To write: ZMIO_CONFIG (sel, moduleid, value)
sl: function No.
mduleid: expansion submodule address
value: configure expansion submodule channel value or range type

--sel:
<table>
<tr><td>sel</td><td>Description</td></tr>
<tr><td>1</td><td>Configure expansion submodule AD/DA range type.</td></tr>
<tr><td>2</td><td>Configure expansion submodule analog AD channel switch state.</td></tr>
</table>

--range data values of “value”
<table>
<tr><td>type</td><td>value</td><td>range</td><td>type</td><td>value</td><td>range</td></tr>
<tr><td rowspan="6">AD</td><td>2</td><td>0-10V</td><td rowspan="6">DA</td><td>10</td><td>0-10V</td></tr>
<tr><td>3</td><td>-10-10V</td><td>11</td><td>-10-10V</td></tr>
<tr><td>4</td><td>4-20mA</td><td>12</td><td>4-20mA</td></tr>
<tr><td>5</td><td>0-20mA</td><td>13</td><td>0-20mA</td></tr>
<tr><td>6</td><td>0-5V</td><td>14</td><td>0-5V</td></tr>
<tr><td>7</td><td>-5-5V</td><td>15</td><td>-5-5V</td></tr>
</table>

--channel data values of “value”, 4 groups of values correspond to 4 channels of AD module
<table>
<tr><td>AD Channel</td><td>CH3</td><td>CH2</td><td>CH1</td><td>CH0</td></tr>
<tr><td>value</td><td>8</td><td>4</td><td>2</td><td>1</td></tr>
</table>
When multiple channels are configured and opened, “value” is the total of all opened channel value. For example, if CH1 and CH2 are opened, then value will be 6 (CH2+CH1=6).</td></tr>
<tr><td>Controller</td><td>XPLC300, ZMC432M</td></tr>
<tr><td>Example</td><td>ZMIO_CONFIG (1, 0, 10)
‘configure DA range type of submodule whose address is 0 as 0-10V
ZMIO_CONFIG (2, 0, 15)
‘open AD all channels of submodule whose address is 0
?ZMIO_CONFIG (1, 0)
‘get AD/DA range types of submodule whose address is 0
?ZMIO_CONFIG (2, 0)
‘get the AD channel switch state of submodule whose address is 0</td></tr>
</table>

ZMIO_INFO – Check ZMIO Itself Expansion

<table>
<tr><td>Type</td><td>Local slave level interface RS485 bus command for XPLC300 / ZMC432M.</td></tr>
<tr><td>Description</td><td>Used to check XPLC300B series controllers' ZMIO expansion.</td></tr>
<tr><td>Grammar</td><td>Grammar 1: var = ZMIO_INFO (sel)
Grammar 2: var = ZMIO_INFO (17, node)
sel: function No.
node: module No., starting from 0, each one module is connected, the No. + 1
moduleid: expansion submodule address

--sel:
<table>
<tr><th>Function No.</th><th>Content</th></tr>
<tr><td>10</td><td>Max IN</td></tr>
<tr><td>11</td><td>Max OP</td></tr>
<tr><td>12</td><td>Max AIN</td></tr>
<tr><td>13</td><td>Max AOUT</td></tr>
<tr><td>16</td><td>The number of modules</td></tr>
</table></td></tr>
<tr><td>Controller</td><td>XPLC300, ZMC432B</td></tr>
<tr><td>Example</td><td>?ZMIO_INFO(10) 'print itself ZMIO expansion's max IN.
?ZMIO_INFO(11) 'print itself ZMIO expansion's max OP.
?ZMIO_INFO(12) 'print itself ZMIO expansion's max AIN.
?ZMIO_INFO(13) 'print itself ZMIO expansion's max OP.
?ZMIO_INFO(16) 'print how many modules for itself ZMIO
?ZMIO_INFO(17,0) 'print the type No. of the first module expanded</td></tr>
</table>

Chapter XXI Simple Routines

21.1 Common Operation

IO Operation

```
WHILE 1                        'cycle
    IF  IN(0) = ON  THEN       'input 0 is on
        OP(2, OFF)             'close output 2.
    ELSE
    OP(2, ON)                  'open input 2
    ENDIF
WEND
```

SP Instruction continuous interpolation

```
ERRSWITCH = 3                  'output all information
BASE(0,1,2,3)                  'choose X Y Z U
RAPIDSTOP(2)
WAIT IDLE

DPOS = 0,0,0,0
ATYPE =1,1,1,1                 'pulse based stepper or servo
UNITS = 100,100,100,100        'pulse amount,100 pulse per mm
SPEED = 200,200,200,200        'FROCE_SPEED will be limited by this speed
ACCEL = 2000,2000,2000,2000    'acceleration set
DECEL = 2000,2000,2000,2000    'deceleration set
MERGE = ON                     'open continuous interpolation
CORNER_MODE=0                  'close corner speed mode, set STARTMOVE_SPEED,
                                ENDMOVE_SPEED by manual
'DECEL_ANGLE = 15 * (PI/180)   'angle of deceleration starts, 15 degrees
'STOP_ANGLE = 45 * (PI/180)    'angle of the lowest speed, 45 degrees

WHILE                          'cycle
    IF  IN(0) = ON THEN        'start motion when input 0 is on
        TRACE "start movesp"
```

```
        'start move, single speed of every distance and stop speed are different.
        FORCE_SPEED = 100       'the first speed is 100
        ENDMOVE_SPEED = 10  'end move speed of the first segment is 10
        MOVESP(100,0)

        FORCE_SPEED = 150       'the second speed is 150
        ENDMOVE_SPEED = 15
        STARTMOVE_SPEED = 15 'start speed here is more than end speed of first motion, so
                              start speed will obey end speed of former motion, it is 10.
        MOVESP(0,100)

        FORCE_SPEED = 200        'the third speed is 200
        ENDMOVE_SPEED = 20
        STARTMOVE_SPEED = 20   'start speed here is more than end speed of second motion,
                                so start speed will obey end speed of former motion, it is
                                15.
        MOVESP(0,100)

        FORCE_SPEED = 300        'the fourth speed is 300, actually is 200 limited by SPEED
        ENDMOVE_SPEED = 30
        STARTMOVE_SPEED = 30   'start speed here is more than end speed of second motion,
                                so start speed will obey end speed of former motion, it is
                                20.
        MOVESP(0,-100)
        WAIT IDLE                    'wait until motion stops
        DELAY(100)                   'time delay
    ENDIF
WEND
END
```

Conversion between String and Data

```
DIM val1,val2,array1(15)     'define variable and array
val1=1234                    'assign variable 1 as 1234
FOR i=0 TO 14
    array1(i)=0              'clear arrays
NEXT
```

```
?val1,val2                  'print two variables
?*array1                    'print array

array1=TOSTR(val1)          'data convert to string, and assign to array
?*array1                    'print array value again
array1=TOSTR(val1)+"1asf"'combine tostr and other string
?*array1

val2=val(array1)            'string convert to data, and assign to variable 2
?val2                       'print variable 2 for determine
```

'conversion only valid between number and string, other strings, such as alphabet, symbol etc, are not allowed.

Handwheel

Handwheel is an encoder, which is usually used to correct work-piece coordinate. Handwheel motion is similar to mechanical gears motion and linear motion with different proportions.

```
ERRSWITCH = 3          'output all information
CONST AXISHAND = 0
BASE(AXISHAND)         'choose axis 0 as handwheel input
ATYPE=6                'pulse + direction, for orthogonal input handwheel, use 3
BASE(1)                'axis 1 will as slave axis of handwheel
ATYPE=1                'stepper
DPOS = 0
UNITS = 100            'pulse amount, 100 pulse per mm
SPEED = 200
ACCEL = 3000
DECEL = 3000
SRAMP = 20
CLUTCH_RATE = 0        'use speed and acceleration to limit.

DIM POSLAST            'record former position
POSLAST = DPOS
WHILE 1                'choose handwheel by manual to link to multiplying ratio
    IF IN(0) = ON AND IN(1) = OFF THEN
        CONNECT(1,AXISHAND)      'link to axis 0, ratio is 1.
    ELSELF IN(1) = ON AND IN(0) = OFF THEN
```

```
        CONNECT(10, AXISHAND)       'link to axis 0,ratio is 10
    ELSELF IN(0) = ON AND IN(1) = OFF THEN
        CONNECT(50, AXISHAND)  'link to axis 0, ratio is 10, in terms of stepper motor, if
                               ratio is too high, it may cause package lost or the motion
                               lasts for a long time.

    ELSELF MYTYPE = 21 THEN       'cancel CONNECT
        CANCEL
    ENDIF

    IF POSLAST<> DPOS THEN
        POSLAST = DPOS
        TRACE DPOS
    ENDIF
WEND
END
```

Fly-Shearing

Please refer to example 2 of MOVELINK command.

Position Comparison Output

Please refer to Chapter XI "Position Comparison Output", there are hardware position comparison output and software position hardware comparison.

Power Failure Storage

Please refer to ONPOWEROFF routines.

Robot

Please refer to Chapter IV "6 FOD Robotic Arm", there are hardware position comparison output and software position hardware comparison.

Robotic Arm by MOVESYNC Command

This example is one SCARA.zpj project file, it includes “building.bas” and “motion.bas” files. Please refer to below contents.

“Building.bas”

```
DELAY (1000)
dim axis_j0,axis_j1,axis_jz,axis_jv,axis_x,axis_y,axis_z,axis_RZ

axis_j0 =0 'axis' big arm
axis_j1 =1 'axis' small arm
axis_jz =2 'axis' telescopic axis
axis_jv =3 'axis' rotate axis
axis_x =7  'big arm of simulation axis
axis_y =8  'small arm of simulation axis
axis_z =9  'telescopic axis of simulation axis
axis_RZ =10 'rotate axis of simulation axis

dim L1  'big arm length
dim L2  'small arm length
dim L3  'offset in X direction
dim ZDis 'rotate axis turns one, the motion distance of axis Z
L1=400
L2=400
L3=0
ZDis=0   '0 decouples RZ and Z

dim u_m1 'the number of pulses when motor 1 turns one circle
dim u_m2 'the number of pulses when motor 2 turns one circle
dim u_mz 'the number of pulses when motor z turns one circle
dim u_mv 'the number of pulses when motor v turns one circle
u_m1=131072
u_m2=131072
u_mz=131072
u_mv=131072
```

```
dim i_1    'joint 1 transmission ratio
dim i_2    'joint 2 transmission ratio
dim i_z    'joint z transmission ratio
dim i_v    'joint v transmission ratio
i_1=80
i_2=50
i_z=2
i_v=6750/384

dim u_j1  'actual pulses of joint 1 in one circle
dim u_j2  'actual pulses of joint 2 in one circle
dim u_jz  'actual pulses of joint z in one circle
dim u_jv  'actual pulses of joint v in one circle
u_j1=u_m1*i_1
u_j2=u_m2*i_2
u_jz=u_mz*i_z
u_jv=u_mv*i_v
dim p_z   'pitch of axis Z
p_z=20

"""set joint axis
BASE(axis_j0,axis_j1,axis_jv,axis_jz)  'select axis No. of joint axis
DPOS= 0,0,0,0   'homing
atype=0,0,0,0    'axis type is pulse axis
UNITS=u_j1/360,u_j2/360,u_jv/360,u_jz/p_z
                                    'set axis Z' units as 1mm pulses, set other axes as 1° pulses
speed=100,100,100,100         'set speed
accel=1000,1000,1000,1000
decel=1000,1000,1000,1000
CLUTCH_RATE=1000,1000,1000,1000    'use joint axis' speed and acceleration to limit
merge=on       'open continuous interpolation
SRAMP = 100
FS_LIMIT = 1000,1000,1000,1000
RS_LIMIT = -1000,-1000,-1000,-1000
```

```
""'set virtual axis
BASE(axis_x,axis_y,axis_RZ,axis_z)
ATYPE=0,0,0,0  'set as virtual axis
TABLE(1000,L1,L2,u_j1,u_j2,u_jv,L3,ZDis) 'fill in parameters according to manual
UNITS=1000,1000 ,u_jv/360,1000        'motion precision, set it in advance, because it can't be
                                       modified during motion
speed=100,100,100,100   'set speed
accel=1000,1000,1000,1000
decel=1000,1000,1000,1000
FS_LIMIT = 1000,1000,1000,1000
RS_LIMIT = -1000,-1000,-1000,-1000

GLOBAL g_op
g_op= 0
while 1
  if g_op=1then
    BASE(0,1,2,3)   'configure joint axis
    CONNFRAME(1,1000,7,8,9,10)   'axis 7,8,9,10 are virtual axis X,Y,Z,W, open inverse
connection
    g_op= 0        'reset
    ?"Inverse Mode"
  elseif g_op= 2 then
    base(7,8,9,10)  'select virtual axis No.
    CONNREFRAME(1,1000,0,1,2,3)  'axis 0/1/2/3 are joint axes, open forward connection
    g_op= 0
    ?"Forward Mode"
    endif
wend
end
```

“Motion.bas”

```
GLOBAL SUB MOVESYNCTEST()
    ERRSWITCH = 4
    TRIGGER
```

```
OP(1,ON)
g_op =1
DIM
WAIT_X,WAIT_Y,WAIT_Z,WAIT_A,CONVREG_Axis,CONVREG_POS,CONVSYNCGO_Ti
me,CONVSYNC_Time,CONVBack_Time

WAIT_X = -26
WAIT_Y = 362
WAIT_Z = 0
WAIT_A = 0
'run general parameters
CONVREG_Axis = 11
CONVREG_POS = 350
CONVSYNCGO_Time = 300 'chasing time
CONVSYNC_Time = 2000    'synchronization time
CONVBack_Time = 500

BASE(7,8,9,10)    'run at waiting position
MOVEABS(WAIT_X,WAIT_Y,WAIT_Z,WAIT_A)
WAIT UNTIL IDLE(7)

DIM convREG_X,convREG_Y,convREG_Z,convREG_A,convAngleX,convAngleY

'//the conveyor belt is in the 225-degree direction of the coordinate system, the positive angle
of the belt dpos with the X-axis is 225, and the angle with the Y-axis is 135
convREG_X = 56    'coordinate when triggering position latch
convREG_Y = 462
convREG_Z = 0
convREG_A = 0
convAngleX = 225/180*PI  'angle radian between belt positive and robotic arm X
convAngleY = 225/180*PI - PI/2 'angle radian between belt positive and robotic arm Y,
please note the radian value can't be negative value, if it is minus, it needs +2*PI.

BASE(11)    'conveyor axis
UNITS = 1910738.4889
```

SPEED = 30

VMOVE(1)

MPOS=350

WAIT UNTIL DPOS(11) < 407 'wait for the conveyor to be brought to the trigger location

' 'Pay attention to the writing method of separated axes. Make sure that all axes are included every time. If you do not need to follow, write the following axis as -1, but do not use the following mode of -1 casually.

' 'here is only doing chasing, no synchronization

base(7)

MOVESYNC(convAngleX,CONVSYNCGO_Time,CONVREG_POS,CONVREG_Axis,convREG_X)

BASE(8)

MOVESYNC(convAngleY,CONVSYNCGO_Time,CONVREG_POS,CONVREG_Axis,convREG_Y)

BASE(9)

MOVESYNC(0,CONVSYNCGO_Time,0,-1,convREG_Z)

BASE(10)

MOVESYNC(0,CONVSYNCGO_Time,0,-1,convREG_A)

'synchronization motion

base(7)

MOVESYNC(convAngleX,CONVSYNC_Time,CONVREG_POS,CONVREG_Axis,convREG_X)

BASE(8)

MOVESYNC(convAngleY,CONVSYNC_Time,CONVREG_POS,CONVREG_Axis,convREG_Y)

BASE(9)

MOVESYNC(0,CONVSYNC_Time,0,-1,convREG_Z)

BASE(10)

MOVESYNC(0,CONVSYNC_Time,0,-1,convREG_A)

'here, synchronization stops, return to upper of the designated placing position

BASE(7)

MOVESYNC(0,CONVBack_Time,0,-1,WAIT_X)

```
    BASE(8)
    MOVESYNC(0,CONVBack_Time,0,-1,WAIT_Y)
    BASE(9)
    MOVESYNC(0,CONVBack_Time,0,-1,WAIT_Z)
    BASE(10)
    MOVESYNC(0,CONVBack_Time,0,-1,WAIT_A)

ENDSUB
```

Read Encoder

Encoder read of Panasonic A6

```
'***********************absolute encoder*****************************
SETCOM(38400,8,1,0,1,0)          'set ports 485 as self-defined protocol

GLOBAL DIM tempchar              'receive one byte

GLOBAL DIM neqbuff(2)            'send identification code, code of 485 is: 81H, 05H
neqbuff(0) = $81
neqbuff(1) = $05

GLOBAL DIM eotbuff(2)            'receive identification code, code of 485 is: 80H,04H
eotbuff(0) = $80
eotbuff(1) = $04

GLOBAL DIM ackbuff               'receive response, 06H
ackbuff = $06

GLOBAL DIM cmdbuff(20)           'send command array
GLOBAL DIM getbuff(20)           'receive string

GLOBAL DIM getnum                'received bytes
getnum = 0

GLOBAL DIM highdata              'multi circles data of encoder
GLOBAL DIM lowdata               'single circle data of encoder

runtask 4,get_char               'start task to receive string
```

```
MODBUS_REG(0)=0
WHILE   1
        IF MODBUS_REG(0) = 1   THEN            'judge if receives data
                MODBUS_REG(0)=0
                getmpos(1,45)                  'read single and multi circles value of station 1
        ENDIF
WEND
END

'read coordinate
GLOBAL SUB getmpos(sifunum,rcr)      'read absolute position of servo 1
        cmdbuff(0) = $00
        cmdbuff(1) = sifunum
        cmdbuff(2) = $d2
        cmdbuff(3) = rcr

        neqbuff(0) = $80 + sifunum
        neqbuff(1) = $05

        eotbuff(0) = $80
        eotbuff(1) = $04

        getnum = 0
        putchar #1,neqbuff
        TICKS = 2000                                   'delay
        WAIT UNTIL(getnum = 2) OR TICKS< 0
            IF getnum = 2 THEN                         'if gets 2 characters
                IF (getbuff(0) = $80 + sifunum) and (getbuff(1) = $04) THEN
                                                       'send command if receive commands
                    getnum = 0
                    PUTCHAR #1,cmdbuff                 'send command to read encoder.
                    TICKS = 2000
                    WAIT UNTIL (getnum = 3) OR TICKS < 0
                    IF (getbuff(0) = $06) AND (getbuff(1) = $80) AND (getbuff(2) = $05)  THEN
                                                       'if get send requirement, send response.
                        getnum = 0
                        PUTCHAR #1,eotbuff             'send message, wait to receive data
```

```
            TICKS = 2000
            WAIT UNTIL (getnum = 15) OR THEN < 0
            IF getnum = 15 THEN          '11-10 is multi circles data, 9-7 is single circle data.
              PUTCHAR #1,ackbuff
              highdata = getbuff(11) * $100 + getbuff(10)
              lowdata = getbuff(9) * $10000 + getbuff(8) * $100 +getbuff(7)
        PRINT getbuff(11),getbuff(10),getbuff(9),getbuff(8),getbuff(7),getnum
            ELSE
              PRINT getnum,getbuff(0) ,getbuff(1),     "read 1 again after time out "
            ENDIF
         ELSE
            PRINT getbuff(0) ,getbuff(1),              "read 2 again after time out "
         ENDIF
      ELSE
         PRINT getbuff(0), getbuff(1),                 "drive no response"
      ENDIF
    ELSE
      PRINT                                            "drive no response "
    ENDIF
END SUB

'serial port receive
GLOBAL SUB get_char()
    WHILE 1
        GET #1,tempchar
        getbuff(getnum) = tempchar
        getnum = getnum + 1
    WEND
END SUB
```

Self-defined G code

```
ERRSWITCH = 3        'output all information
BASE(0,1,2,3)        'choose X Y Z U, don't modify freely, since there is rule in G01.
RAPIDSTOP
WAIT IDLE
```

```
DPOS = 0,0,0,0
ATYPE=1,1,1,1                'pulse based stepper or servo
UNITS = 100,100,100,100     'pulse amount, 100 pulse per mm
SPEED = 200,200,200,200
ACCEL = 2000,2000,2000,2000
DECEL = 2000,2000,2000,2000
MERGE = ON                   'open 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 motion
    IF IN(0) = ON THEN       'start motion when input 0 is on
        'run a box
        G91                  'relative position
        G01 X100 Y0
        G01 X0 Y100
        G01 X-100 Y0
        G01 X0 Y-100
        WAIT IDLE            'wait until motion stops
        DELAY(100)           'time delay
    ENDIF
WEND
END     'for avoid executing following SUB file again, make this file exit automatically when
                             running

'relative position mode:
GLOBAL SUB G_INT()
DIM coor_rel
coor_rel = 1                 'relative position mode
END SUB

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))          'relative position
    ELSE
        LOCAL xdis, ydis, zdis, udis
        IF sub_ifpara(0) THEN                        'if there are parameters
            xdis = sub_para(0)
        ELSE
            xdis = ENDMOVE_BUFFER(0)
        ENDIF
        IF sub_ifpara(1) then
            ydis = sub_para(1)
        ELSE
            ydis = ENDMOVE_BUFFER(1)
        ENDIF
        IF sub_ifpara(2) THEN
            zdis = sub_para(2)
        ELSE
            zdis = ENDMOVE_BUFFER(2)
        ENDIF
        IF sub_ifpara(3) THEN
            udis = sub_para(3)
        ELSE
            udis = ENDMOVE_BUFFER(3)
        ENDIF
        MOVEABS(xdis,ydis,zdis,udis)          'absolute position
    ENDIF
END SUB

'absolute position mode
GLOBAL GSUB G90()
    TRACE "G90 entered"
    coor_rel = 0
END SUB

'relative
GLOBAL GSUB G91()
    TRACE "G91 entered"
```

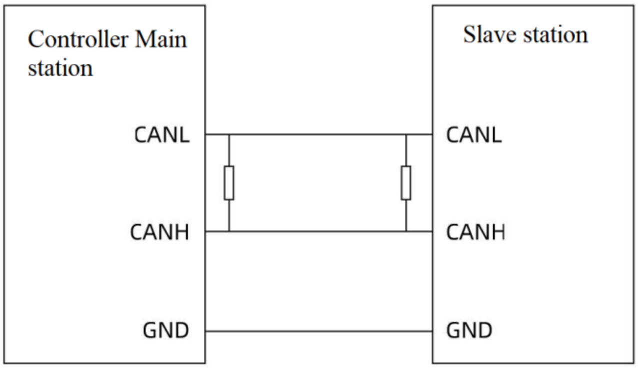
```
	coor_rel = 1
END SUB

'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)
	TRACE"G00 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))
	ELSE
		local xdis, ydis, zdis, udis
		IF sub_ifpara(0) THEN
			xdis = sub_para(0)
		ELSE
			xdis = ENDMOVE_BUFFER(0)
		ENDIF
		IF sub_ifpara(1) THEN
			ydis = sub_para(1)
		ELSE
			ydis = ENDMOVE_BUFFER(1)
		ENDIF
		IF sub_ifpara(2) THEN
			zdis = sub_para(2)
		ELSE
			zdis = ENDMOVE_BUFFER(2)
		ENDIF
		IF sub_ifpara(3) THEN
			udis = sub_para(3)
```

```
        ELSE
            udis = ENDMOVE_BUFFER(3)
        ENDIF
        MOVEABS(xdis,ydis,zdis,udis)
    ENDIF
END SUB
```

21.2 Module Communication

CAN Communication

Wiring between controllers.



CANL - CANL

CANH - CANH

Add a resistance of 120 Ohm between CANL and CANH

Procedure in Master

```
CANIO_ADDRESS = 32          'set as master
STOPTASK1
RUNTASK 1,task_canget       'start to get task
GLOBALIF_send
IF_send= 1

WHILE 1
    IF if_send = 1 THEN
        TABLE(0,0,8,1,2,3,4,5,6,7,8) 'send 8 bytes to controller which can-cob-id is 0.
        CAN(0,7,0)                   '0-CAN channel,7-send data,0-start table address of data
        if_send = 0
    ENDIF
```

```
WEND
END

GLOBAL SUB task_canget()        'receive task
    WHILE 1
     CAN(0,6,10)                      'receive data, data will be saved after table(10). Table(10)
                                 means CANID, when it is <0, which means no data.
                                'table(11)  means  data  number  received.  table(12)……means
data.
        IF(table(10) >= 0  )  THEN            'judge if data was received
            ?"ID of controller that sends data:",table(10)
            ?"data bytes number:",table(11)
            ?"data:"table(12),table(13),table(14),table(15),table(16),table(17),table(18)
        ENDIF                        '0-CAN channel,7-send data,0-start table address of data
    WEND
END SUB
```

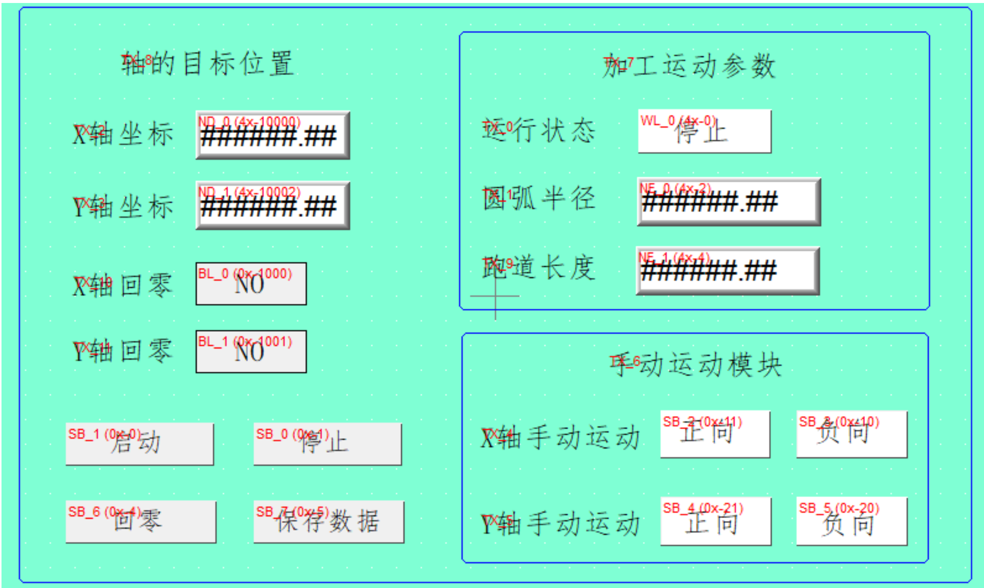
Procedure in Slave

```
CANIO_ADDRESS = 0               'slave
STOPTASK  1
RUNTASK 1,task_canget           'start to receive task
GLOBAL if_send
if_send = 0

  WHILE 1
        WAIT UNTIL if_send = 1  'wait until data comes.
        TABLE(0,32,1,$ff)           'send 1 byte to controller which can-cob-id is 32,it is FF.
        CAN(0,7,0)                  '0-CAN channel,7-send data,0-start table address of data
        if_send = 0
  WEND
  END

GLOBAL SUB task_canget()             'receive task
    WHILE 1
        CAN(0,6,10)                   'receive data, data will be saved after table(10). Table(10)
                                  means CANID, when it is <0, means no data.
                                'table(11)  means  data  number  received.  table(12)……means
```

```
data.
        IF(table(10) >= 0  )  then          'judge if data was received
            ?" ID of controller that send data:",table(10)
            ?" data bytes number:",table(11)
            ?"data：  "table(12),table(13),table(14),table(15),table(16),table(17),table(18)
            if_send = 1
        EDNIF
    WEND
END SUB
```

HMI Communication

This HMI starts from 0, some other HMIs start from 1, all is mapped to address 0 in controller.




```
'**********initialization module*********
ERRSWITCH = 3                  'output all information
RAPIDSTOP(2)
WAIT IDLE
BASE(0,1)                      'choose X Y
UNITS = 100,100
SPEED = 100,100
ACCEL = 1000,1000
DECEL = 1000,1000
SRAMP = 100,100

DIM run_state                  'run status
```

```
run_state = 1                    '1-stop, 1-run, 2-homing
MODBUS_REG(0) = run_state 'show running status

DIM radius,length                'radius, length
radius = 100                     'default radius value
length = 300                     'default length value
FLASH_READ 0,radius,length
MODBUS_IEEE(2) = radius     'show radius value
MODBUS_IEEE(4) = length     'display length value

DIM home_done                    'finished sign position of homing, 0-not finish,1-homing finish
home_done = 0                    'enter not go homing status when power is on

MODBUS_BIT(0) = 0            'start, button resets
MODBUS_BIT(1) = 0            'stop, button resets
MODBUS_BIT(4) = 0            'homing, button resets
MODBUS_BIT(5) = 0            'save data, button resets

MODBUS_BIT(1000)=0           'axis X, homing sign is 0
MODBUS_BIT(1001)=0           'axis Y, homing sign is 0

STOPTASK 2
RUNTASK 2, guidetask            'start manual run task

'**********button scan module*********
WHILE 1                              'scan HMI button input
    IF MODBUS_BIT(0)= 1 THEN    'press start button
        MODBUS_BIT(0) = 0          'button resets

        IF run_state = 0 THEN          'standby stopped
            IF home_done = 0 THEN                'if homing motion is done, start motion
                TRACE "before move need home"
            ELSEIF home_done = 1 THEN            'if homing motion is done, start motion
                TRACE "move start"
                STOPTASK 1                 'software is safe, stop task 0
                RUNTASK 1, movetask     'start run process task 1
            ENDIF
        ENDIF

    ELSEIF MODBUS_BIT(1) = 1 THEN         'stop to press button
        TRACE "move stop"
        MODBUS_BIT(1) = 0                      'button resets
        RAPIDSTOP(2)
        STOPTASK 1
        RAPIDSTOP(2)
```

```
            run_state = 0                          'stop sign
            MODBUS_REG(0) = run_state              'display state
    ENDIF

    IF MODBUS_BIT(4) = 1 THEN                      'press homing button
          MODBUS_BIT(4) = 0                        'homing reset

          IF run_state= 0  THEN
                stoptask 1
                runtask 1, home_task               'start homing task
          ENDIF
    ENDIF

    '''reserve data
    IF MODBUS_BIT(5) = 1 THEN                      'save data, press button
            MODBUS_BIT(5) = 0                      'save data, button resets

            print "write data into FLASH "
            radius = MODBUS_IEEE(2)
            length = MODBUS_IEEE(4)
            FLASH_WRITE 0,radius,length            'write data into fan sector
      ENDIF
WEND
END

'**********process motion module*********
movetask:                               'run task: draw arc + runway
      run_state =1                      'enter run state
      MODBUS_REG(0) = run_state

      radius = MODBUS_IEEE(2)           'read radius
      length = MODBUS_IEEE(4)           'read length

      TRIGGER
      BASE(0,1)                         'choose XY axis
      MOVEABS(0,0)

      MOVE(length,0)                    'move runway path starts from the origin
      MOVECIRC(0,radius*2,0,radius,0)
      MOVE(-length,0)
      MOVECIRC(0,-radius*2,0,-radius,0)
      WAIT IDLE(0)

      run_state = 0                     'enter standby state
      MODBUS_REG(0) = run_state
END
```

```
'**********homing task*********
home_task:
	TRACE "enter home task"
	run_state = 2			'homing sign
	MODBUS_REG(0) = run_state	'display state

	TRIGGER

	BASE(0,1)
	CANCEL(2) AXIS(0)		'first axis 0, stop axis 1
	CANCEL(2) AXIS(1)
	WAIT IDLE(0)
	WAIT IDLE(1)

	'DATUM(3) AXIS(0)		'homing of actual device axis 0
	'DATUM(3) AXIS(1)		'homing of actual device axis 1
	MOVEABS(0) AXIS(0)		'homing of virtual device axis 0
	MOVEABS(0) AXIS(1)		'homing of virtual device axis 1

	WAIT IDLE(0)
	MODBUS_BIT(1000)=1		'set sign, indictates axis 0 has done homing motion

	WAIT IDLE(1)
	MODBUS_BIT(1001)=1		'set sign, indictates axis 1 has done homing motion

	home_done = 1
	TRACE "home task done"

	run_state = 0			'return to standby state
	MODBUS_REG(0) = run_state
END

'**********manual motion*********
guidetask:
WHILE 1
	IF run_state = 0 THEN		'judge if stops or not
		BASE(0)
		IF MODBUS_BIT(10) = 1 THEN		'left
			MODBUS_BIT(10) = 0
		VMOVE(-1)
		ELSEIF MODBUS_BIT(11) = 1 THEN	'right
			MODBUS_BIT(11) = 0
			VMOVE(1)
		ELSEIF MTYPE = 10 OR MTYPE = 11 THEN	'not be VMOVE motion
		CANCEL(2)
		ENDIF
```

```
            BASE(1)
            IF MODBUS_BIT(20) = 1 THEN            'left
                MODBUS_BIT(20) = 0
                VMOVE(-1)
            ELSEIF MODBUS_BIT(21) = 1 THEN     'right
                MODBUS_BIT(21) = 0
                VMOVE(1)
            ELSEIF MTYPE = 10 OR MTYPE = 11 THEN  'not be VMOVE motion
                CANCEL(2)
            ENDIF
        ENDIF
        DELAY(100)
WEND
END
```

Self-defined Ethernet Communication

```
OPEN #11, "TCP_SERVER",1000     'use self-defined Ethernet channel 2, as master, port NO. is 10

GLOBAL DIM tempchar
GLOBAL CONST datamax=20
GLOBAL DIM datanum
            datanum=0
GLOBAL DIM DATA(datamax)

STOPTASK1
RUNTASK 1,aa
WHILE 1
    tempchar = 0                    'clear former characters.
    GET #11,tempchar                'get single character, save it in tempchar
    PRINT tempchar                  'print ASCII code of character
    DATA(datanum) = tempchar  'save in array
    datanum = datanum + 1
    IF datanum = datamax then    'if exceed array space
        datanum = 0
        FOR i = 0 to datamax-1    'clear data in array.
            Data(i) = 0
        NEXT
```

```
        ENDIF
        IF    tempchar = 59 then             'if meet character ;stops.

                PRINT #11,"ok1245"
        ENDIF
                DELAY(10)
WEND
END

SUB aaa()
        WHILE 1
        tempchar = 0                    'clear former characters.
        GET #10,tempchar            'get single character ,save it in tempchar
        PRINT tempchar                'print ASCII code of character
        DATA(datanum) = tempchar    'save in array
        datanum = datanum + 1
        IF datanum = datamax then    'if exceed array space
                datanum = 0
                FOR i = 0 to datamax-1     'clear data in array.
                        DATA(i) = 0
        NEXT
        ENDIF
        IF  tempchar = 59 then            'if meet character ;stops.
                PRINT #10,"aaaaaaaaaa"
        ENDIF
        WEND
END SUB
```

Communication between controllers

Valid in the latest firmware, download master and slave procedure separately into different controllers, and connect with controller's net interface with reticle.(can use switch)

'''''''''''''''''''''''''''''''**master controller**

```
DIM i,j,lasttick
MODBUS_REG(j)=0
MODBUSM_DES2($1 , 20, "192.168.0.11")    'controller IP can't be same
WHILE 1
```

```
	lasttick=TICKS
	FOR i =0 TO 9999
		MODBUS_REG(0) = i
		MODBUSM_REGSET(0,1,0)
		MODBUS_REG(0) = 99
		MODBUSM_REGGET(0,1,0)
		IF MODBUS_REG(0) <> i THEN PRINT "MODBUS_REG(0)=" MODBUS_REG(0),
"MODBUSM_STATE=" MODBUSM_STATE
	NEXT
	?lasttick-TICKS
WEND
END
''''''''''''''''''''slave controller
DIM j
ADDRESS=1
MODBUS_REG(j)=0
WHILE 1
	IF MODBUS_REG(0) <> 0 then
	  SPEAKOUT(100)
	ENDIF
WEND
END
```

String and Self-defined Communication

```
SETCOM(38400,8,1,0,0,0)       'set as RAM mode
DIM TEMPVAR                   'define variable
DIM VALUE
DIM CHLIST(10)                'define array
FOR i=0 TO 9
	GET #0, TEMPVAR           'read data through channel 0
	CHLIST(i)=TEMPVAR         'read data store in array in sequence
Next
TRACE CHLIST                  'debugging
VALUE = VAL(CHLIST)           'convert to variables
PRINT #0, TOSTR(CHLIST)       'convert to string
```

21.3 Bus Initialization

EtherCAT Initialization

Requirements: controllers with EtherCAT interface, servo based drive must support EtherCAT fieldbus, valid in ZDevelop version above 2.5.

This only sets fieldbus enable operation, others should be set in upper computer, like pulse amount, axis speed, motion path, etc.

```
'''''''''''''''''''''initialization preparation
RAPIDSTOP(2)
WAIT IDLE
FOR i=0 to 10                       'cancel former Fieldbus axes setting
    ATYPE(i)=0
NEXT
'''''''''''''''''''''EtherCAT fieldbus initialization
SLOT_SCAN(0)                        'start scan
IF RETURN THEN
    ?"fieldbus scan successfully", "linked devices number：  "NODE_COUNT(0)
    ?
    ?"start to map axes NO."
    AXIS_ADDRESS(0)=0+1             'Map Axes NO.
    ATYPE(0)=65             'EtherCAT Type, 65-position, 66-speed control,67-torque control
    DRIVE_PROFILE(0)= -1            'servo PDO functions
                                    when atype is 66, set as 20, when atype is 67, set as 30
    DISABLE_GROUP(0)                'each axis as one group
    ?"axes map finished"
    DELAY (100)

    SLOT_START(0)                   'start fieldbus
    IF RETURN THEN
        ?"fieldbus starts successfully"
        ?"start to clear drive errors(set according to drive data dictionary)"
        DRIVE_CONTROLWORD(0)=0         'clear errors for cooperating with servo
        DELAY (10)
        DRIVE_CONTROLWORD(0)=128       'when bit7=1, force servo to clear errors
```

```
        DELAY (10)
        DRIVE_CONTROLWORD(0)=0          'clear errors for cooperating with servo
        DELAY (10)
        DATUM(0)                        'clear all axis errors of controller
        DELAY (100)
        ?"ready to enable axes "
        AXIS_ENABLE(0)=1                'axis 0 enable
        WDOG=1                          'main switch of enable
        ?"axis enable finished"
    ELSE
        ?"fail to fieldbus start"
    ENDIF
ELSE
    ?"fail to scan fieldbus"
ENDIF
END
```

Rtex Initialization

Requirements: controllers with RTEX interface, use Panasonic RTEX servo drive.

```
RAPIDSTOP(2)
WAIT IDLE
FOR i=0 to 10                       'cancel former Fieldbus axes setting
    ATYPE(i)=0
NEXT

SLOT_SCAN(0)                        'start to scan
IF RETURN THEN
    ?" fieldbus scan successfully ",  "linked devices number： "NODE_COUNT(0)
    ?" start to map axes NO."
    AXIS_ADDRESS(0)=0+1             'map axis NO.
    ATYPE(0)=50          'Rtex Type, 50-position control, 51-speed control, 52-torque control
    DRIVE_PROFILE(0)=1              'servo mode that support IO mapping
    DISABLE_GROUP(0)                'each axis as one group
    ?"axis NO. mapping finished"
    DELAY (100)

    SLOT_START(0)                   'start fieldbus
    IF RETURN THEN
```

```
        ?"open Fieldbus successfully"
        DATUM(0)
        DELAY (100)
        ?"ready to enable axes"
        AXIS_ENABLE(0)=1                    'enable axis 0
        WDOG=1                              'main switch of enable
        ?" axes enable finished "
    ELSE
        ?" Fail to open fieldbus "
    ENDIF
ELSE
    ?"Fail to scan fieldbus"
ENDIF
END
```

Chapter XXII Error and Debug

Due to wrong wiring, procedure logic problems, instructions errors, etc., motor will not run as expected, controller will show errors.

How to find out reason and solve problems, the first rule is to close other software, then use Zdevelop to debug, following functions should be known: manual motion debugging, interrupt debugging, oscilloscope collection, register check, remote commands, procedure information print, fast IO test.

22.1 List of Common Problem

Problem	Debugging Solutions
Motor doesn't run	Manual Motion Debug
Register value on HMI is not correct	Check Register
Not run as expected in procedure	Interrupt Debug + Procedure Information Print
inputs or outputs don't work	Fast IOs Test
Machine shakes too much	Oscilloscope Collection

Problem Checking

If there is procedure error, first check procedure problem:

⚠ when procedure motion appears errors, ZDevelop software will show error information. If there is no error information, check through ?*task command, then double-click error information, it will turn to procedure error position autoamtically. Relevant codes as follow, see "Error Code List".

Problem	Possible Reasons
2043:Unknown function is met	Function is not supported by controller.
stop of error:2049 Line not ended.	1.some commands must occupy a whole line. 2.no need () for calling GSUB.
stop of error:2033 Unknown label is met	1.not-defined variable/array 2.not-defined SUB process 3.with defined array, but defined commands aren't executed, maybe relevant files are not set run automatically.
2048:Function can only be used in expression	Function must be with return value, but no need in the start position.
2064:Param few	Function parameters are too less.
2063:Param too many	Function parameters are too more.
2072:Need = sign	Not write "=".

2060:Syntax format error	There exists grammar error of instruction.
error:1010	Pause repeat
error:1011	No motion, so can’t pause.

The procedure running errors are solved, it still exists abnormal operation. If motor doesn’t work, then check following settings.

1. Reason of Drive

Since drive motor is not set inverse IO level by default, so limit position will appear errors. Solution: set limit position level inversion according to drive manual. For example, for Panasonic servo, parameters should be: Pr4.01=010101h(65793), Pr4.02=020202h(131586). For other brands, please see relevant manual.

Relevant parameters	Default values (decimal)	Position control/full-closed control	
		Signal name	logic
Pr4.00	00323232h (3289650)	SI-MON5	Normally open (ON)
Pr4.01	00818181h (8487297)	POT	Normally close (NC)
Pr4.02	00828282h (8553090)	NOT	Normally close (NC)

Signal name	Sign NO.	Set value	
		Normally open (ON)	Normally close (NC)
invalid	-	00h	It can’t be set
Positive drive forbids input	POT	01h	81h
Negative drive forbids input	NOT	02h	82h

2. Reason of procedure

1) UNITS value is too small, this causes extreme slow motion speed. This can’t be distinguished by naked eyes.

2) there is abnormal status of motor(limit position, alert…), then it can’t move, print AXISSTATUS value to judge.

3）pulse can’t sent correctly when there is wiring error of motor.

4) axis enable OP port close (for servo motor, it needs to open).

5) program processing makes the motor unable to move, download the empty program.

6) drive motor alarm.

Following reasons only for fieldbus axis:

7) fail to fieldbus scan, print return value.

8) WDOG main enable and AXIS_ENABLE axis enable instructions are closed.

9) there are errors of drive status setting. See drive manual for details.

3. **Problem checking steps:**

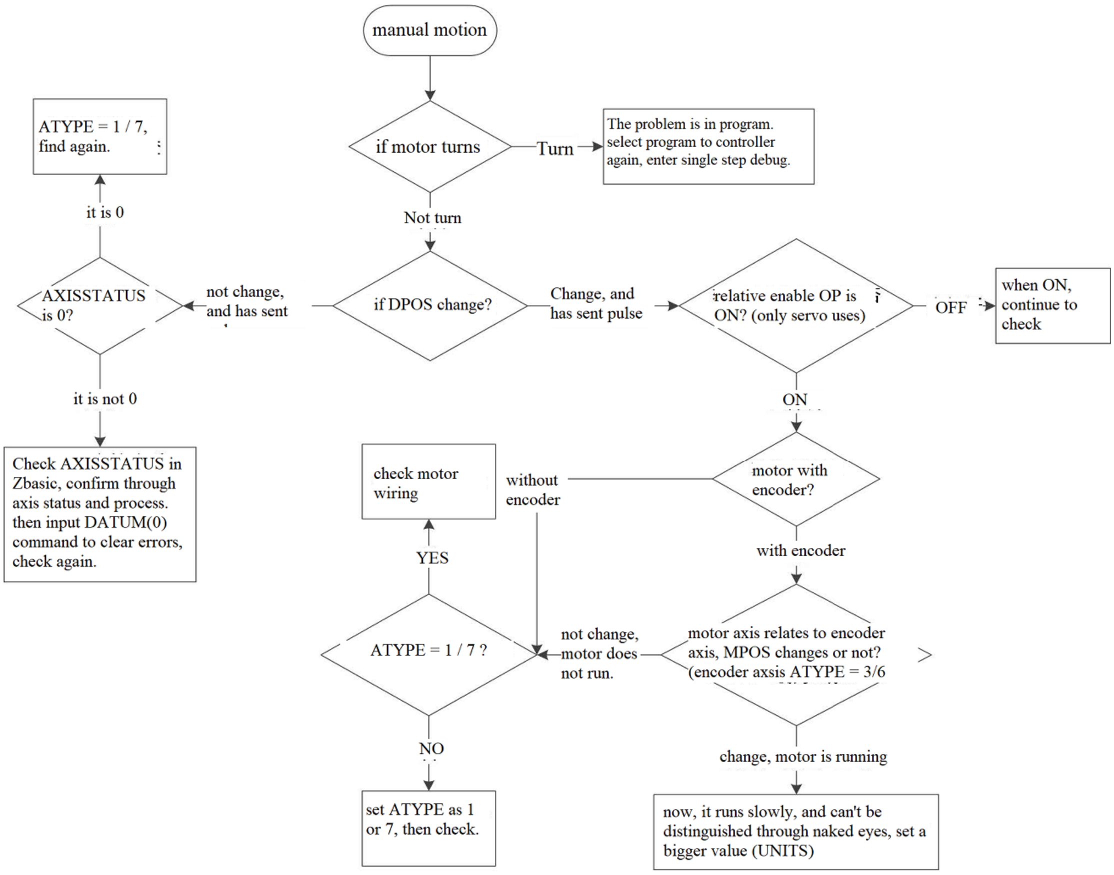
1) use ZDevelop software.

2) close all other software and procedures linked with the controller, except ZDevelop, to avoid these external elements make influence on operation.

3) use ZDevelop download an empty program to controller, for avoid making influence by internal elements.

4) open ZDevelop, “view”-“motion by manual” and “view”-“axis parameters”

5) following steps only for **pulse axis.**



Motor only can do single-direction motion, possible reasons as follow:

1.motor is in the limit position status, see AXISSTATUS.

2.wrong motor control mode, set INVERT_STEP as relevant mode (double pulse / pulse + direction).

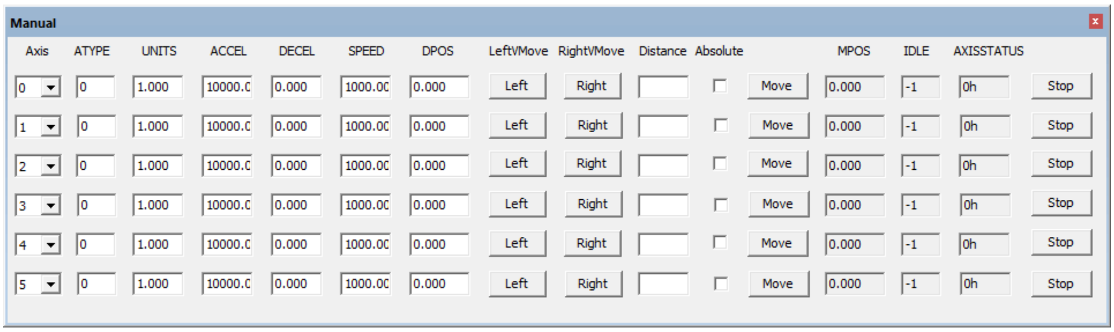
3.check if motor wiring is correct.

22.2 Solutions

Manual Motion Debug

Use “Manual” motion to check if there are wiring problems.

Close all related software except ZDevelop, and use ZDevelop to link with controller, download an empty procedure. And select axis No. through VIEW-AXIS PARAMETERS, manually set axis type ATYPE, pulse amount UNITS, speed SPEED, acceleration ACCEL, deceleration DECEL, etc., then click view -- manual, operate motor by manual.



Operation Method: hold LeftMove/RightMove, motor will run continuously, it stops when releasing. “DPOS” (command position) shows how many pulses sent now (the unit is units). Fill in “Distance” parameter well, then check “Move”, please note there is one “Absolute”, when it is checked, motor will run to “you filled distance” directly, when not checked, it will keep moving according to the distance parameter (relative motion).

When click left or right, followed situation will happen:

1. Motor doesn’t run, DPOS changes.

A. pulse was sent from controller, check if there is drive alarm, check motor wiring.

B. UNITS is too small, motor is running slightly, not able to observe.

2. Motor only runs at one direction

Check the motor control mode, controller only supports double pulse or pulse + direction 2-axis control mode, orthogonal pulse is not supported.

3. Motor only runs when operate one side (left or right).

A. check motor wiring.

B. motor control mode and controller control mode are different, default controller control mode is pulse + direction, use INVERT_STEP to modify.

4. Motor doesn’t run, DPOS also doesn’t change.

Check if there is an alarm from AXISSTATUS.

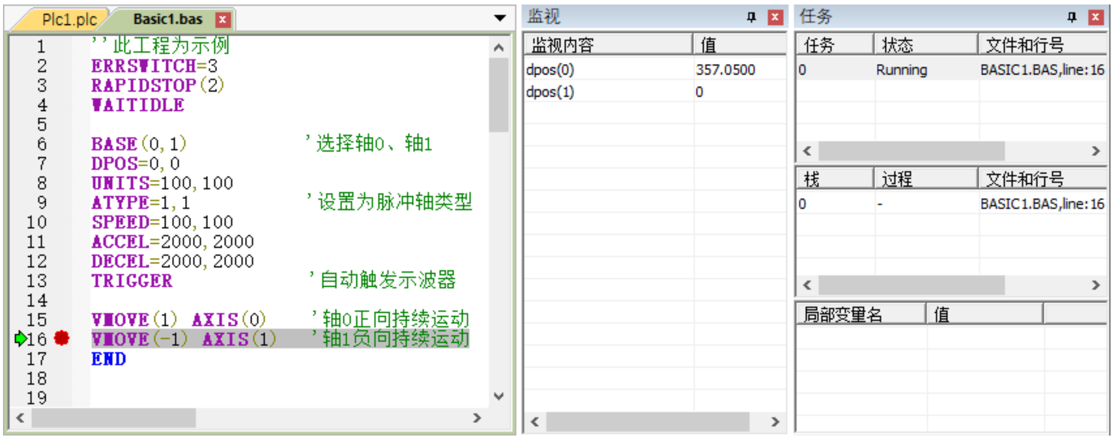
5. For controller that is supplied by dual-power, please check whether IO 24V is wired

and supplied normally.

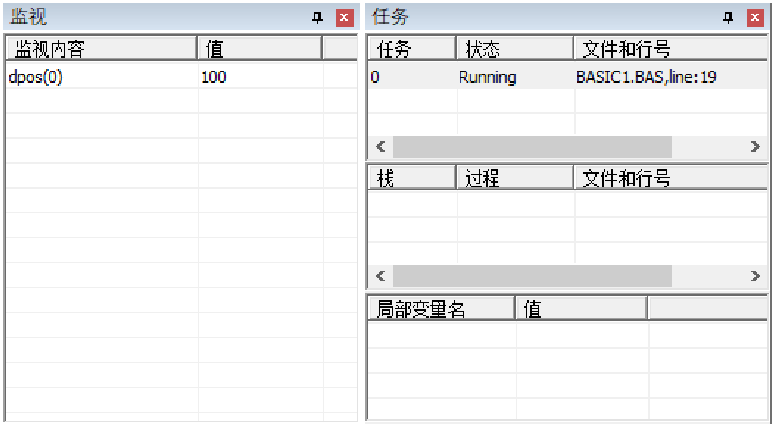
Interrupt Debug

Interrupt debugging is used to check procedure process or logic and judge procedure logic errors. Also, it checks influence on registers, variables, arrays, etc. together with monitor content. Debugging procedure should be same as controller procedure.

Press F9 to add interruption points. In Develop, connect controller well, then click debug-start/stop debug to enter debugging mode.



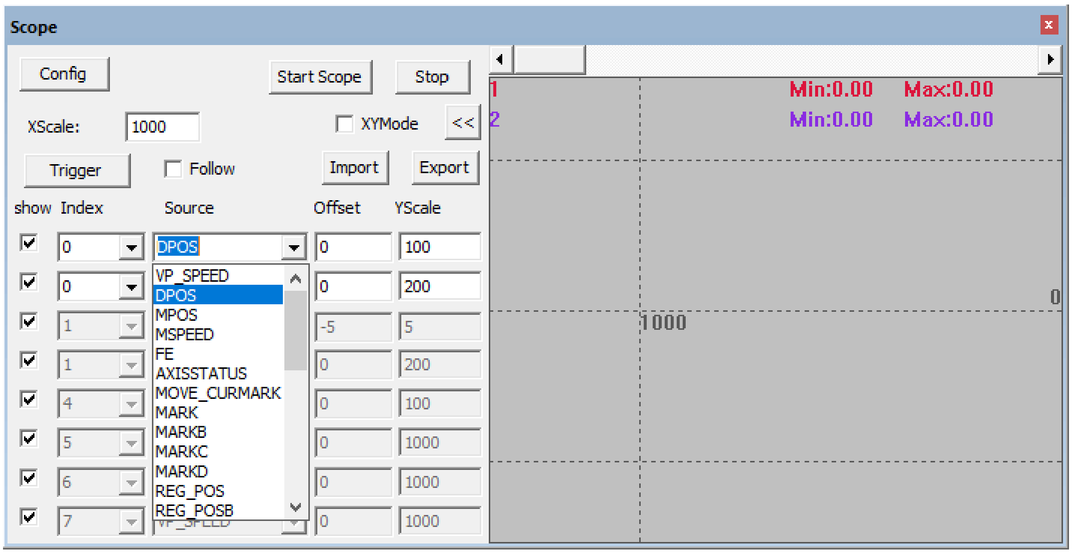
In debug mode, task running process, monitoring item, sub process, local variables in subsidiary functions all can be checked.



Oscilloscope Collection

Oscilloscope can collect all kinds of data types, click “view-oscilloscope-source” to check.

Oscilloscope is usually used to judge actual speed and position of motor.

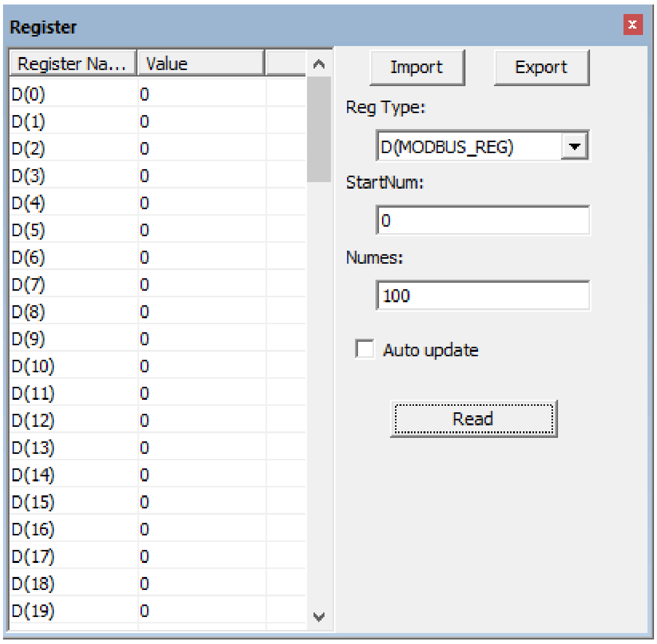


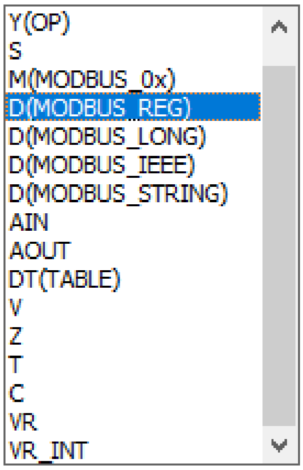
XYmode is used to check two-dimensional trajectory, source of first 2 channels should be set as DPOS or MPOS.

If machine shakes too much, then use oscilloscope to collect encoder feedback MSPEED, to check if wave shape is smooth, if it is smooth, which means pulse delivery is stable, then continue to check if speed curve is too steep and if speed is too big in constant speed mode

Register Check

In Develop, click view-register to check registers data. including register types: IN, OP, MODBUS_4X, MODBUS_0X, TABLE, VR, AIN, AOUT.





It is only valid in controller that supports PLC

If registers in HMI did not change as expected, now check relative register type and number of touch screen to confirm this register value has changed or not. If it changed, there is communication problem between controller and HMI, check the wiring and communication parameters setting. If it not changed, there is procedure logic problem. Check if program is executed correctly or not through interrupt debug.

Remote Commands

Online command executes instructions that are sent out, which is usually used to check if controller instructions are normal.

Such as, when procedure is already executed, but motor doesn’t run as expected. For this situation, it can’t confirm the result from other tasks in procedure or controller function. Now, download an empty program (without code) into controller, then send remote commands to check possible reasons.

Print Program Information

Print information in different procedures to check if procedure is executed or the number of execution or which relative executed parameters.

The print instruction is PRINT, also the omit type of print is “?” (English character).

```

dim ccv(20),aa,bb
ccv="ad"
aa=100
bb=300

base(0,1,2)
atype=1,1,1
units=200,200,200
speed=100,100,100
accel=1000,1000,1000

stoptask 1
runtask 1,aaa

while 1
    ?bb
    move(bb)
wend
end


sub aaa(num)
  while 1
    ?aa
    move(aa)
  wend
end sub
```

Fast IOs Test

Connect Ddevelop with controller, click view-input/output, link output and input one by one(EGND must be linked) at the same time, then operate output, check output status in Develop at the same time.

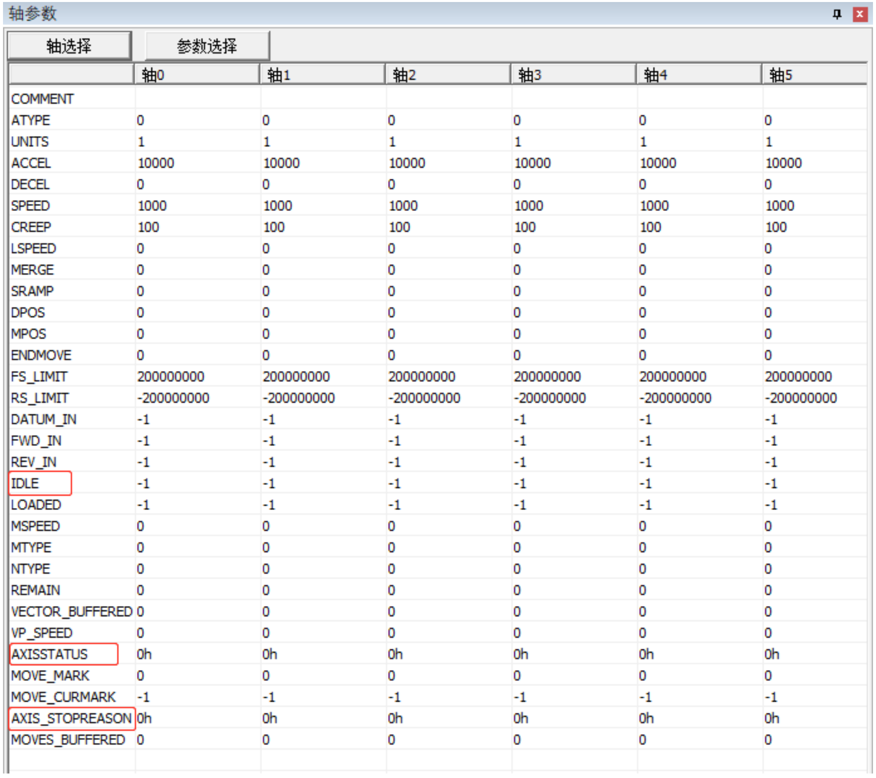
Some controllers need additional 24V power to supply IOs.

If checking expansion module IOs, first need to confirm there is a 120Ω resistance between CANL and CANH and DIP setting is correct, then check wiring of main power and IOs power supply is right. At last, check as above steps.

Axis Parameters Status Judge

Axis parameters can be checked on PARAMETER, such as ATYPE, UNITS, SPEED, etc. and can be modified directly here. But not valid for parameters read.

Judgements of axis running state: IDLE, AXISSTATUS and AXIS_STOPREASON.


轴参数

轴选择 参数选择

	轴0	轴1	轴2	轴3	轴4	轴5
COMMENT						
ATYPE	0	0	0	0	0	0
UNITS	1	1	1	1	1	1
ACCEL	10000	10000	10000	10000	10000	10000
DECEL	0	0	0	0	0	0
SPEED	1000	1000	1000	1000	1000	1000
CREEP	100	100	100	100	100	100
LSPEED	0	0	0	0	0	0
MERGE	0	0	0	0	0	0
SRAMP	0	0	0	0	0	0
DPOS	0	0	0	0	0	0
MPOS	0	0	0	0	0	0
ENDMOVE	0	0	0	0	0	0
FS_LIMIT	200000000	200000000	200000000	200000000	200000000	200000000
RS_LIMIT	-200000000	-200000000	-200000000	-200000000	-200000000	-200000000
DATUM_IN	-1	-1	-1	-1	-1	-1
FWD_IN	-1	-1	-1	-1	-1	-1
REV_IN	-1	-1	-1	-1	-1	-1
IDLE	-1	-1	-1	-1	-1	-1
LOADED	-1	-1	-1	-1	-1	-1
MSPEED	0	0	0	0	0	0
MTYPE	0	0	0	0	0	0
NTYPE	0	0	0	0	0	0
REMAIN	0	0	0	0	0	0
VECTOR_BUFFERED	0	0	0	0	0	0
VP_SPEED	0	0	0	0	0	0
AXISSTATUS	0h	0h	0h	0h	0h	0h
MOVE_MARK	0	0	0	0	0	0
MOVE_CURMARK	-1	-1	-1	-1	-1	-1
AXIS_STOPREASON	0h	0h	0h	0h	0h	0h
MOVES_BUFFERED	0	0	0	0	0	0

1.IDLE: judge if motion instruction of axis finishes or not, motion in process-0, motion finishes-1. For axis status, often use WAIT IDLE(AXSI NO.) to judge.

2.AXISSTATUS can check all status of axis. Show value with decimal, judge status with relevant binary, there can appear several errors at the same time.

3.AXIS_STOPREASON latch stop reasons, write 0 to clear, latch AXISSTATUS information as per bit.

Appendix I Error Code List

Code	Error Code Description	Possible Reason	Solution
External Error Code			
201	Invalid Sub-Module		Set ZMIO_OFFSET as minus or the No. that exceeds starting IN & OUT No., and it must be the times of 8, such as, ZMIO_OFFSET = -8.
210	Oversize File		Downloaded zpj project or zar file is too large, please check controller specification by “?*max”.
211	File Size Error		
212	State Error	When in Resume, it is non-pause state.	When in Resume, it is in non-pause state, see if RT running state is switched too fast, this usually appears MotionRT7.
213	Download & Upload File Error, Package Loss	Appear when calling PC function.	
214	Downloaded File Length Verify Error		
215	Insufficient Buffer Length	when the sending character string is too long, return this.	Check the length of character string command. It can’t be too long.
217	Unsupported Controller Function		
218	Wrong Called & Transferred Parameters		
219	Downloading Error, Multi-File Downloading		See if several files now are downloading at the same time.
220	Filename Error, Be with Special Character		See if the filename has unsupported character.
221	Too Long Filename		See if the filename’s length exceeds.
222	Invalid File		The file may not exist, open it, see whether there is alarm, and check which one.
223	Locked, Password Protection		Enter correct password.
224	Locked, Password Protection 2		Don’t unlock the

			controller too frequently, and the password must be correct.
225	Unknown Error		
226	Disk Space Error		
227	Firmware Version Error		1. Check whether it is the latest one – update. 2. See if it matches with your controller – change correct one.
228	File Open Error		
229	Connection Error		
230	Fail to “bind”		
231	File Read Error		
232	File Write Error		
233	Link Encrypted		
234	Firmware Error		1. Update dll.
235	File Delete Error		
236	Path Error		
237	File Close Error		
240	XPCI Sub-Card State Error		
241	XPCI Sub-Card Memory Resource Error		
242	Sub-Card Setting Error		
243	Unsupported Sub-Card		
Chip Self-Checking Error			
260	Hardware Error, LED Shrinks		
261	Disk Unformatted		
262	RTC Error		
263	NORFLASH Error		1. There is strong interference, please restart, if still, contact with us. 2. For XPCIE, please check wiring cable, make sure it is good.
264	RAM Error		1. Make sure internet stable. 2. Haware error, contact with us.
265	NANDFLASH Error		Same as code 263.
266	U Disk Error		1. See whether U disk is plugged in stably.

			2. Interface error, contact with us.
267	FPGA Error		1. Contact with us.
268	Ethernet Hardware Error		1. Contact with us.
Software Error			
271	Backup Power Error		1. Contact with us.
272	Sub-Card Doesn’t Exist		
273	ID File Lost		1. Contact with us.
274	System File Lost		1. Contact with us.
275	No Master Control, Appear in Sub-Card		
276	Program File Verify Error		1. For ZMC0XX, see whether is ROM file. 2. Check if ZAR file is correct.
277	Program File Error		1. See whether it lacks program file.
278	ZAR File “apppass” Error		
279	ZAR File ID Error		
280	Too Many BAS Files		1. Check controller supported BAS file numbers (?*max – max_file), then see whether it exceeds.
281	Sub-Card ID Conflict / Multi-Master Conflict		
282	Unsupported Function		1. The controller doesn’t support this function. 2. See if it is controller new function, then update the firmware.
283	“set” File Error	Parameter file lost.	1. Please modify needed parameters that are saved into flash, then generate as set file automatically.
284	ZAR File Not Matched with Controller		
285	Image File Error		
286	Font File error		
287	.c File Function Syntax Error	Usually C language program error	1. Check your edited C language program. 2. If bottom layer C

			language error, please contact with us to update.
288	Above Abnormal, Alarm Again when Powered On.		
289	Too Fast Reset		
290	Drive Program Init Error		
291	fpga Error		1. Please restart. 2. Contact with us.
292	Insufficient mmap		
293	MotionRT Trial Expires		1. Check whether the License is configured 2. For trial mode, please restart MotionRT.
zmotion			
1000	Motion Offset Error		
1001	Must Be Interpolation State		
1002	No Motion Buffer		
1003	Can’t Be in Interpolation State		
1004	Slave Axis is Moving		
1005	Unsupported Motion Control Function		
1006	Arc Position Error		
1007	Ellipse Para AB Error		
1008	Motion Module Para IN Error		
1009	In Motion, Operate Not Allowed		
1010	Repeat Run “Pause” & Others		
1011	In IDLE, Can’t Do Pause, etc.		
1012	Now Motion Doesn’t Support Pause		
1013	Pause Point Not Found		
1014	Unsupported ATYPE		
1015	ZCAN ATYPE Conflict		
1016	Unsupported Axis Function		
1017	FRAME Correction Data Error		
1018	Too Less FRAME Correction Data		
1019	Too Less Met FRAME Data		
1020	Too Less FRAME Data Auxiliary Para		
1021	Too Small Span Between FRAME Correct Data, < Joint-Axis Numbers		

1022	FRAME IN Coordinate Error		
1023	In FRAME, No Way to Modify Coordinates		
1024	FRAME Inverse Kinematic Error		
1025	Not FRAME Status		
1026	FRAME HAND Error		
1027	Can’t Switch Attitude in Interpolation		
1028	UNITS of Special Joint-Axis & Virtual-Axis Should Be Same		
1029	FRAME Called INIT Para Error	Distance / angle para exceeds, note the angle unit.	
1030	CORNERMODE 7-Bit Set Already, But Unsupported Motion		
1031	CORNERMODE 7-Bit Set Already, But Not in FRAME.		
1032	AXIS_ADDRESS Error		
1033	Too Many Interpolation Axes		
1034	INTCYCLE Time Out		
2000	RTBASIC Module Offset, Para in Module Error		
2001-2020	Internal Error in RTBASIC Module		1. Check if there are same definition name of different types’ variables according to the message hint. 2. Check if it prints unassigned variable in the function.
2021	Manual Stop		
2022	Task Stops Because Other Tasks Error		
2023	Operate RO Para		1. Read only parameter can’t be modified.
2024	Array Exceeds		1. Modify the number of defined arrays (table index starts from 0).
2025	Variables > Controller Allowed		1. Check ?*max – max_sub, if more, please select other models.
2026	Arrays > Controller Allowed		
2027	Array Space > Controller Allowed		

2028	SUB > Controller Allowed		
2029	Mark Name Error	Command edit error / no note for Chinese	1. See if there are unsupported symbols (especially in function, variable definition).
2030	Too Long Mark Name		1. See if function or variable definitions are too long, controller has own limit.
2031	No “)”		1. See if the parentheses are full. 2. See if the parentheses are correct or if there is special character in the middle.
2032	Unknown Character	Command para’s comma is Chinese symbol.	1. See if there are unsupported symbols.
2033	Unknown Name in Expression	Undefined variable / array	1. See if commands are edited correctly, and whether commands are supported.
2034	SUB Can’t Be Used in Expression		1. Modify SUB usage method, SUB function only can be called.
2043	Unknown Command Mark (Now Line 1st Mark Name)	Command editing error	1. See if there are wrong editing commands, and if commands are supported.
2044	Stack Overflow	SUB function recursive calling > system allowed times	1. Check “?*max” – “max_callstack”, see the limit, each controller is different. 2. See if functions are called mutually.
2045	Too Complex Math expression		1. Check and modify the math expression, each controller is different.
2046	No End Quote Mark Found		1. Check and modify quote mark.
2047	No Returned Value for The		1. Cancel related

	Command, Can't Read		commands / functions in expression or use in online command.
2048	Function Must Return a Value, Not at Beginning of First Line		1. See if executed function that needs return value but the expression didn't output the value, like, directly run SIN(1).
2049	Special Command Must Be One Separate Line	Some commands must occupy one whole line.	1. Edit related commands in separate line.
2050	Para / Array Needs Index		
2051	Variables Can't Use Index		
2052	Array Redefine & Inconsistent Length	Define same arrays many times.	1. Check if the array is redefined, and see if the length is not same.
2053	Array Defined Length Para Error, Minus / Oversize		
2054	Mark Defined as SUB	SUB progress mark can't be defined again.	1. Check if the function name is redefined.
2055	Mark Defined as Parameter		1. Check if the parameter name is redefined.
2056	Mark Reserved, Can't Use		1. Check defined mark name, see if conflict with existed para command, (mpos, ..)
2057	Unrecognizable Character	Like "&" can't be identified by system	1. See if undefined value is transferred.
2058	SUB Calling Out Stack (Repeat)		1. Check SUB calling logic and correct it.
2060	Syntax Format Error		1. Find the wrong line, and correct it.
2061	Parameter Overflow		
2062	Function Para Range Error		1. Check the command parameter or custom function parameter, see if the range exceeds. 2. Check task No., if it is more than allowed by ?*max – max_task

			3. Check if there inputs too many contents in executing line.
2063	Too Many Function Parameters		
2064	Too Less Function Parameters		
2065	Lack Operands		1. Check if the operation expression is full.
2066	Lack Operands after Operators		
2067	Lack Operands before Operators		
2068	Unknown Operators		
2069	Lack Binary Operators	2 commands are in one line, one operator is needed.	
2070	CALL Must Call SUB		1. Correct the content to be called after CALL, it must be SUB type.
2071	No AUTO, Won't Start		
2072	Lack Assignment Symbol	When there is no comma between data, use "space".	1. Correct expression information, add needed assign symbol
2073	Empty File		
2074	SUB Defined Mark Name Conflicts		1. Check and correct parameter definition.
2075	Task to Be ON is Running		1. Check if there is conflict on executing task No.
2076	Multi-Para, Please Use Comma		1. Check parameter used format, correct.
2077	No "<"		1. Check if the brackets are full
2078	Too Much "IF"		1. Cut down the number of IN, each controller is different.
2079	Too Much "Loop"		1. Cut down the number of LOOP, see "?*max -- max_loopnest", each controller is different.
2080	Too Less Interpolation Axes		1. Correct the number of axes.
2081	CONST Can't Be Modified.	It will report and error when defined constant data is assigned again.	1. Correct CONST usage.
2082	Command Doesn't Support Online Sending		

2083	Too Many SUB Defined Para		1. Cut down local parameters (< ?*max -- max_local of one sub), each controller is different.
2084	SUB with Para Can't Use in GOTO		1. Use GOTO command correctly.
2085	Too Many LOCAL Defined Para		1. Cut down the number, each controller is different.
2086	LOCAL Variable Name & File Variable Name / Others Conflict		1. Correct related content.
2087	LOCAL Doesn't Support Array Definition.		1. Correct LOCAL command usage.
2088	GSUB Defined Para Letters Repeat		1. Correct repeated part.
2089	GSUB Defined Para Only Can Be Single Letter		1. Correct non-single part.
2090	RO Parameter Error		1. Read only parameter can't be modified.
2091	GSUB_IFPARA Usage Error		
2092	Divisor is 0		1. See if one divisor in the code is 0, correct.
2093	Over Buffer		
2094	Online Commands Blocking Too Long (Time)		1. Check the network. 2. Check if there are too much data transferring in a short time. 3. Make sure link stable, and bandwidth is OK.
2095	Same Para Name		1. Rename.
2096	Use Uninitialized Value		1. Initialize the value.
2097	Axis No. Conflicts		1. Correct axis mapping
2098	Data Type Error		
2099	Inside Error		1. Usually defined name conflict, correct.
2100	Too Many SCANEDGE		
2101	ZINDEX Type Mismatch		
2102	ZVOBJ > Allowed Numbers		
2103	Inconsistent ZVOBJ Definition		
2104	ZINDEX Value Error		
2110	Call Command Not Enabled		
2120	Structure Define Conflict, Can't Define Multi at the Same Time		

2121	Name & System Command Conflict		
2122	Structure Definition Can’t Be Recursive		
2123	Structure “item” & Structure Name Conflict		
2124	Syntax Error, Lack Structure Type		
2125	Structure “item” Error		
2126	Lack Structure Element		
2127	Lack Structure Variable		
2128	Structure > Allowed Numbers		
2129	Structure Element > Allowed		
2130	Structure Type Undefined		
2131	Data Type Undefined		
2132	Structure Define Should Be Before Codes		
2133	Can’t Dynamically Delete Static Definition		
2134	Lack Array Type		
2150	Function Return Not Immediate Done		
2151	Function Can’t Immediate Return Because Now Expression Doesn’t Support		
2152	Dynamic Stake is Overflow		
2200	Function Calling Not Done		
2901	System Error, Too Many Defined Mark (variable, array, process…)		
3201	Over Buffer		
3202	File Abnormal End		
3203	Program Structure Command Lack Something	no THEN after IF.	1. Compatible error, see if the command is full, such as, no THEM in IF.
3204	Internal State Error		
3205	Unsupported Function		1. The controller doesn’t support this function. 2. See if it is controller new function, then update the firmware.
3206	Internal Calling Para Error		1. You used controller unsupported function,

			please see whether controller supports. 2. EtherCAT config file is not loaded.
3212	Unknown Error		
3230			
3231	Insufficient Resources		
3232			
3233	OS Return Error		
3242	"os" Error		
3243	U Disk Uninserted		1. Check if the U disk is inserted and is stable. 2. Check if the U disk that can be known by your controller.
3244	File Opened Again		
3245	Oversize File		
3248	Filename Error		
3249	Too Long Filename		
3250	No This File		
3301	Arc 3 Points are in One Line		1. Correct command usage.
3302	2 Parallel Lines, No Intersection Point		1. Correct command usage.
3401	MODBUS Master Para Error		1. See if MODBUS master para is correct, see if the length exceeds.
3402	Message Response Timeout		1. See communication configuration. 2. See if there is blocking / unstable network.
3403	Message Length > Max Buffer		1. Sending data exceeds the limit.
3404	MODBUS Message Bytes / ID Error		
3405			
3406			
3407	MODBUS Return Para Error		
3408	MODBUS Return Doesn't		

	Support		
3410	Receive Data in Blocking		
3420	MODBUS Salve Returns Unsupported Function Code		
3421	MODBUS slave station returns invalid function codes		
3422	MODBUS Slave Return Address Space Error		1. Master and slave addresses are not matched, please read & write not existed register addresses, then set correct one.
3423	MODBUS Slave Return Data Length Error		
3424	MODBUS Slave Return Length Too Long		
3501	ZCAN Return No Sub-Card		
3502	ZCAN Return No Sub-Card Related Axis		
PLC (Controller Side)			
4001			
4002	Parameter Error		
4003	Unknown Type		
4004	Unknown Function	Unknown function is called, or the called function is not GLOBAL type.	Modify the syntax usage.
4005	Stack Too Many STL		
4006	Too Many Stakes		
4007	Too Complex Program, Too Much BLOCK		
4008	No Stack BLOCK		
4009	No Stack STL		
4010	No Stack	when using MPP and MPS commands, MPP > MPS	Determine error position, then modify the script., MPP used times should be same as MPS.
4011	MC Can't Be in the Middle of STL		
4012	MC Level Error		
4013	STL Only Can Be Main File Main Task		
4014	File Content Error		

4015	RET Must Be after STL		Correct it.
4016	> Register Range		
4017	< register Range		
4018	L Not Defined		
4019	Don’t Support G Code Function		
4020	COTO Can’t Cross PLC & BASIC	In Basic, you used GOTO to jump to PLC.	Use syntax correctly.
4021	Only One PLC Main Task		
4022	Syntax Error		
4023	FOR NEXT Error, Mismatched		Use syntax correctly.
4024	FOR NEXT Error, no NEXT		Use syntax correctly.
4026	FOR MC Mixed Use		Use syntax correctly.
4027	FOR STL Mixed Use		Use syntax correctly.
4030	Must Use in PLC Main Task		
4031	Must Use in Interrupt		
4032	Too Less Parameters		
4033	Too Many Parameters		
4034	Multiples of 8		
4035	Register Mark Error		
4036	Register Type Error	STL command and others use wrong register type	Determine alarm position and correct the script.
4037	Too Many LV		
4038	Read-Only		
PLC (PC Side)			
4501			
4502			
4503	Insufficient Memory	The memory exceeds allowed	Optimize the memory.
4504	Reflow to Busbar	The soft component is connected direct, and parallel to the busbar.	Delete the connection that is not correct.
4505	Reflow	Direct connect, and parallel to other soft components without the soft component.	Delete the connection that is not correct.
4506	AND Command Can’t Connect to Busbar Directly.	No other elements between AND type and busbar.	Add the component, or delete this ladder diagram.
4510	Not Full, No OUT Command in the Right	No output command is connected after ANB command or after the component.	Determine where is wrong, then correct it.

4511	In the Rightmost, it is Not OUT Type.		Check if it is output type in the rightmost side.
4512	Rightmost Must Be Separate	2 output types of components that are in the rightmost side are connected.	Check if they are connected in the rightmost.
4513	OUT Type Must Be in Right Most	Output types' component is not the rightmost side of ladder diagram, but in the middle or left side.	Check if there is output type is in the middle or left.
4514	Unsupported Command Type		Check where the unsupported type is, and correct it.
4515	Inside Error		
4516	Inside Error		
4517	Empty Register	There is no any value in called register.	Assign the register that is to be called.
4518	DOT Value Exceed		
4519	Register Exceed	Called register exceeds the number of registers.	Modify the usage range of the calling register to be within the specified number range
4520	Too Many Characters		
4521	Register Type Error	Script used register type is not controller standard register type.	Check if the used register type is consistent, valid.
4522	Register Value Error	Register value input by the component / command is wrong.	Find the wrong component or command (wrong register value).
4523	Too Many Registers		
4524	Too Less Registers	The component didn't set the register.	Add register for the component or command that didn't set register.
4525	STL Usage Error		
4526	RET Usage Error	RET should be used after STL.	
4527	RET Repeat	RET command or component is used again.	Delete one.
4528	END / LBL Position Error		
4529	Function Can't Be Connected to		

	Busbar Directly		
4530	No Push when Out the Stack	when using MPP and MPS commands, MPP > MPS	
4531	Too Many MPS	Keep using MPS over 11 times	Determine the usage times, and correct it.
4532	Register Type Usage Error	You used supported register type.	Use correct register type.
4533	ANB Error, Insufficient Blocks	No others after ANB, real numbers of used is not consistent with needed.	
4534	ORB Error, Insufficient Blocks	No others after ORB, real numbers of used is not consistent with needed.	
4535	ANB Error, Can’t Combine after OUT	After OUT command, then you call ANB.	
4536	ORB Error, Can’t Combine after OUT	After OUT command, then you call ORB.	
4537	AND Can’t Be Connected to Busbar Directly	AND command or other components are connected to busbar directly, before, there is no other command / component.	Determine where is the wrong position, then correct the script or LAD.
4538	OR Can’t Be Connected to Busbar Directly	OR command or other components are connected to busbar directly, before, there is no other command / component.	Determine where is the wrong position, then correct the script or LAD.
4539	OR Can’t Be after OUT		
4540	STL & MC Can’t Be Shared		
4541	MC Can’t Be Connected to Busbar Directly		
4542	@Register Without Brackets		
4543	Note Error		

4544	Too Many LAD Columns	LAD columns > controller allowed	Too many LAD columns, please delete some.
4545	OUT Type Can't Be Connected to Busbar Directly.	Output types component connects to busbar directly.	Delete the corresponding OUT type component.
HMI			
5000	LCD No. Error	HMI running tasks > controller allowed.	See if it is more than allowed (allowed can be known from ?*max – max_hmi), more, please select other controllers.
5001	HMI File Error	Inside error	Please contact with us.
5002	LCD No. Conflict	Multiple HMI file use same LCD No.	See if there ae same LCD No.
5003	Unsupported Object	Inside error	Please contact with us.
5004	Insufficient Memory	Too small memory setting for VPLC7 or other controllers don't support.	1. For VPLC7XX, adjust "config -- hmisize". 2. Contact with us.
5005	"Control" Error	One abnormal "layer" value set by PC software is transferred.	Contact with us.
5006	Window No. Exceed	You set too large window No.	1. Set it as a small one. 2. If it is full, contact us.
5007	Invalid Window No.	1. In base window, you opened one window that doesn't exist. 2. One invalid window is opened by the HMI_SHOW - WINDOW.	Check if you opened the base window that had been opened by the command already.
5008	HMI Content Error	Inside error	Contact with us.
5009	Same Window No.	Two HMI files or several windows use same window No.	See if they are same.
5010	Object Property Lost	Inside error	Contact with us.
5011	>1 KeyboardShow in Keyboard		
5012	ACTION Type Error	Because action value is abnormal in PC configuration.	Contact with us.

5013	Too Many Events		
5014	Back to Last Window Failed		
5015	Can't OFF Base Window		Check HMI file's base window, and check script "close" logic.
5016	No Related Character in Font	This will not alarm, but the character that can't be known will not be shown.	
5017	Must Use in HMI Task		
5018	Wrong Control Type	Because the control is operated by the command but it doesn't support.	Check parameter configuration and related HMI window, see if they are consistent.
5019	Control ID Not Exist		
5020	Control ID Conflict	Different controls are set same component No.	Correct it.
5021	LCD No. Error	PC host computer error	PC host computer error
5022	No Valid LCD Found		
5023	LCD No Opened		
5024	LCD No Data		
5025	Program Reset		
5026	LCD Opened		
5027	Not Network LCD	PC host computer error (300 uses internal LCD No., the HMI with x uses network LCD No.)	
5028	Unsupported Compress	Reserved	Reserved
5029	Unsupported Color Depth	Controller doesn't support that.	Contact with us.
5030	Unsupported Data Type	Inside error	Contact with us.
5031	Device No. Error		
5032	LCD_SET Can't Use	Reserved	
5033	Don't Set REDRAW in DRAW	In draw function, you used set_redraw command.	"set_redraw" is one refresh function that must be used in refresh function.
5034	DRAW Function Only Can Be DRAW	"draw" command is used in refresh function.	Draw command (usually the beginning of draw) must use in draw function.
5035	Can't Call in DRAW	The command that operates control is used in draw function.	Commands that operate control to show, control state can't be used in draw function.
5036	Fixed Inner LCD Resolution		
5037	LCD Resolution Beyond	Set resolution >	You can check x and y

		controller allowed	parameter (?*max – max_hmi), that is, the resolution size.
5038	Library File Name Error	Called library file name is wrong while using text library.	Check the control “text library” or the command, correct the name.
5039	Too Many Characters		
5040	Object Property Lost	Inside error	Contact with us
5041	No KeyboardShow in keyboard		
5042	Too Many States		
5043	Unsupported Draw Property		
5044	Remote Communication Device Name Error		
5045	Remote Communication Data No Update		
5101	Invalid Date Format	You used invalid format while using SYSTEM command (like, not the format of % + letter).	Use correct data format: % + letter
5102	Control Not Exist	You use the command to operate the control that doesn’t exist (such as, online change control text).	Use correct control ID.
5103	Too Many / Less Polygon Points	The polygon points is <2 / >32.	Check the point numbers, and better to use DRAW_POLYGONS.
5104	No Free Scroll Bar	You used auto-allocate ID syntax while initializing the scroll bar.	Note to release ID No. that is not used, if you need more, please contact with us.
5105	Invalid Scroll Bar ID	You called invalid scroll bar ID when using scroll bar command (such as, you used ID (>31)).	Use correct initialization scroll ID.
5106	Unsupported Function	Controller used unsupported HMI control / command.	Contact with us to see if there is new firmware.
5107	Not Load Image	You don’t import the image while CAD command is	Please use CAD control to import corresponding graphics, then do other

		executing.	CAD operations.
5108	File Broken	Usually appears when importing broken format of strong formats (bin file).	Reexport broken bin file.
5019	Menu Para Error		
5110	Not Enough table Space when Exporting, then Overflow		Make table space large through the command or change one controller if now it is the max space.
5111	Unsupported Data Type		Change as correct data type.
5112	Unsupported Control Type		Change control ID, and use correct control.
5113	Array Overflow	Exceed max value	Check array size, and see whether transferred array max value exceeds or not.
5114	Inside Error	Error in HMI inside	Contact with us.
5115	Channel Overflow		
EtherCAT Bus Errors			
6000	EtherCAT Module Error, SLOT No. Error		
6001	Inside Error, Unsupported Function		
6002	No Stack		
6003	Unknown		
6004	“mbox” Occupied		
6005	Parameter Error		
6006	Supported Device Types Exceed		
6009	NODE Operated Exceed		
6010	Slave State Error		
6011	Unsupported Slave		
6012	Insufficient Resources		
6013	Slave Device Respond Timeout		Slave doesn’t respond when master writes data several times for a long time (like, >400ms), please check from drive error, time when problem appears, and controller performance, etc.
6014	Insufficient Buffer		
6015	Respond Package WKC Error		Slave returned WKC

			counts is wrong, please check specific reason.
6016	Too Long SDO Respond Content		Slave respond SDO length is too long, please check if the sent SDO data type is correct.
6017	SDO Respond Error		The transmission of the SDO read or write operation is actively rejected by the servo and terminated. The cause of the error needs to be analyzed in combination with the specific SDO content sent, such as reading a data object that does not exist in the data dictionary, or writing PDO data during operation.
6018	SDO Respond Data Length Error	Usually because sent SDO data type is incorrect or unsupported.	Check SDO data type, if it is correct.
6019	WKC Timeout		Slave returned WKC timeout, please check from drive error, controller performance, time when problem appears, etc.
6020	STATE Switch Timeout		SoE state switching timeout, that is, master doesn’t get correct respond from slave after s long time requesting on switch the state, please check from drive error, controller performance, time when problem appears, etc.
6021	SDO ABORT, Drive Return Error	Data dictionary reading or writing error / write drive function that is not supported.	The transmission of the SDO read or write operation is actively rejected by the servo and terminated. The cause of

			the error needs to be analyzed in combination with the specific SDO content sent. Generally, because sent incorrect / unsupported SDO.
6022			
6023	NODE PROFILE Error		
6024	Axis PROFILE Error		
6025	Too Many Axes		Bus axis numbers exceed allowed, please check and correct.
6026	Exceed Custom PDO Buffers		
6027	Too Many Custom Numbers		
6028	Don’t Modify PROFILE after ON		
6029	PDO Package Length > System Allowed		Check “profile” setting, for functions that will not use, don’t configure PDO.
6030	Scan First		
6031	Too Many Devices		
6032	Over buff Length		
6035	Preset & profile Conflict		
6036	Too Many PDO		
6037	Special Profile, Drive doesn’t Support it.		
6038	“preset scan” Not Matched		
6039	“preset” Empty		
6040	No Scan		
6042	Device Not Support		
6045	Mail Timeout		
6046	Data Lost		
6047	Data Type Error		
6048	PDO Not Support		
6049	Unsupported Sub-Module		
6050	Too Many Submodules		
6051	Unknown Submodule		
6055	Operated PDO Type Length Not Matched		
6056	PDO R & W Content Not Found		
6057	PDO Key Content Error (like, DRIVE_STATUS)		
6058	AL State Reading Error		
6059	AL State Error, Non-OP State		
6060	Drive Error		

6061	Insufficient XmlEsi Buffer		
6065	IO PDO Must Byte Offset (≠0)		
6066	IO PDO Not Continuous		
6067	DA PDO Type Conflict, Only Can Be Single Type		
6068	ZML File SM Info Lost		Correct xml file, and convert it to zml again, contact with us to add.
6069	ZML File Key Info Lost		
6070	ZML File Needs More Space		
6071	Wrong ECAT Module Numbers		
6072	ZML File Message Repeat		
6073	Module startup Doesn't Support CA Method		
6208	RTEX Drive ID Conflict		
6209	Scan Timeout	Usually because cable.	
6210	RTEX Initialize Failed		
6211	RTEX Scan Result Error		
6212	RTEX Device Type Error		
6213	RTEX Message Timeout		
6214	RTEX SDO Message Error		
6500-6520	EIO Error		
6501	PDO Length Settings Error		
6502	Mail Length Settings Error		
6503	Don't Modify RO Data Dictionary		Read only data dictionary can't be modified.
6504	Too Many Data Dictionary Arrays		
6510	PDO Written Content Error, First Level Index Content Error		
6511	PDO Data Content Repeat		
6512	PDO Content Error, Dictionary No. Error		
6513	PDO Content Error, Dictionary Sub No. Error		
6514	PDO Content Error, Dictionary Length Error		
6515	Too Many PDO		
6516	Slave AML Alarm		
6517	Slave WDOG Alarm, PDO Package Loss All the Time		
NC Module			
7003	Unsupported Syntax when		

	Analyzing		
7006	No Info about which Axis & Channel of The Command		
7008	ACOS Operation Command Parameters Out of Range		
7009	ASIN Operation Command Parameters Out of Range		
7010	The Divisor Can’t Be 0		
7011	The Exponent Must Be an Integer When the Base is Negative.		
7012	Character Error	Because there are characters that can’t be analyzed.	
7014	Wrong Digit Format		
7032	Inner Syntax Error		
7034	Flat Switching Error	One wrong flat value is transferred.	
7037	Unsupported Operation Command		
7041	Feed Speed is 0, Can’t Run G1		
7043	Feed Speed is 0, Can’t Run G2, G3		
7057	Appear Unused Axis Parameter		
7066	Too Long of This Line Code	256 characters can be edited in one line.	
7069	Arc Start Point = End Point by Radius Method		
7077	No “=” in Assignment Command		
7078	Illegal G Code		
7096	Lack “[” after ATAN		
7097	Lack “[” after Operation Command		
7098	Illegal N Code > 999999		
7100	Illegal M Code		
7101	Arc Para R & IJK Mixed Use		
7102-7119	Muti Axis Specified Info (A~F, H~L, P~Z, 7012=A、7107=H、7112=P), Can’t Know Which One		
7121	Negative Can’t Be Squared		
7124	Negative in G Code		
7127	Negative in M Code		
7132	Brackets Embedded in One		

	Bracket for Noting		
7133	Syntax Error, No Read Value		
7134	Digit Lost		
7135	Read Value is Not Integer		
7136	Inner Syntax Error		
7142	Illegal Para & Variable Address		
7147	Arc Parameter Lack		
7153	Too Small Arc Radius to Arrive End		
7156	ATAN No "/"		
7161	Inner Syntax Error		
7168	≥2 Same Type Command of G		
7169	≥2 Same Type Command of M		
7170	Can't Open NC File		
7171-7188	Inner Syntax Error		
7196	0 / - Value in LN		
7197	Arc Radius R is 0		
7200	Empty Analysis Code		
7201	No Symbol, Integer Can't Be with "+ / -" Symbol.		
7202	No Integer (without symbol) Read		
7203	No Symbol, Read Number Can't Be with "+ / -" Symbol.		
7204	No Real Number (without symbol) Read		
7220-7246	Unused Key Words in Code Line, 7220-7228: XYZABCUVW, 7229-7231: FST, 7232-7241: EDHIJKLPQR		
7301	Command Type / Command Code Not Exist		
7302	Register Command Existed Already		
7303	Microprogram No. Exceed		
7304	Command Group No. Exceed		
7305	Command Code Exceed		
7306	Expand Type Not Exist		
7307	Command Priority Exceed		
7308	Empty Command (for the function "call")		
7309	Command Not Exist		
7311	Command of Priority 0 & Others Appear at the Same Time		

7312	Here are Multi Motion Commands / Commands with Coordinate Para Synchronously		
7313	Inner Error		
7314	Inner Error		
7315	Inner Error		
7319	Too Many Parameters Checked		
7320	Lack Parameter to Check		
7350	Illegal Parameter Checking		
7351	Undefined G Code		
7352	Undefined Length Unit		
7353	Undefined Type / Mode		
7354	Inner Error		
7355	Unknown Coordinate Axis	You used unset axis.	
7356	No p, IJK Para for Scaling Command		
7357	Scaling Command's Para P & IJK Mixed Use		
7358	Inner Error		
7359	Tool Compensation Radius > the Cutting Arc Radius		
7360	Inner Error, Preset GOTO Numbers > Allowed (256)		
7361	Invalid Channel No.		
7362	Target Channel is Running Other Tasks		
7363	No Run Task of Target Channel	Pause / stop the free channel task	
7364	Can't Switch Work Plane, Unit after ON Tool Compensation / Coordinate Rotate Function		
7365	Can't Use Gsub Expand Command in "Online Output"		
7366	G04.1 Q_ Transferred Wrong Q Arguments	Q arguments doesn't contain its own channel.	
		Transferred channel No. by Q arguments is out of the range.	
7367	G04.1 Q_ Wait P Signal Error	Channels that wait mutually, signal P are different.	
7370	No WHILE / DO for Microprogram		
7371	WHILE & END Should Be		

	Together		
7372	WHIE > Allowed Layers (5)		
7373	Not Found "GOTO" Target Address		
7374	Subprogram Calling > Allowed Layers Embedded	8 layers can be embedded at most.	
7375	Empty Subprogram File		
7376	Subprogram File Oversize	<10k	
7379	Unsupported Microprogram No.		
7380	Mian Axis is In Motion of Feed Axis, Can't Modify Main Axis Speed & Ratio	When the main axis and feed axis are same axis, please switch motion and speed well.	
7400	Parameter Error		
7401	Para Address No Open		
7402	Invalid Para Address		
7403	Invalid Para Value	Parameter value exceeds valid range.	
7404	No Permission to R & W Para		
7405	Can't Modify Para Description		
7407	Para Form File Reading Error		
7408	Parm Form File Writing Error		
7410	G Code File (Expand Type) is Full	Max: 12 expansion names	
7411	Expand G Code File Type Failed	Transferred invalid suffix name	
7412	Para Form File ON Failed		
7413	Dynamically Expand Para Failed	Expand the number of parameters that can be expanded, parameter address is out of range.	
7414	System Variable Unused	Access invalid system variables	
7421-7436	Your Assigned Axis Direction, which Exceeds the Range of the First Journey (soft limit)		
9912	Pull-Down List, Control Call Function Error		
PC Side Error			

20000	PC Wrong Offset		
20001			
20002	Wrong Parameter		
20003	Timeout	fifo buffer blocked	
20004			
20005			
20006	Operating System Error		
20007	Serial Port Open Failed		
20008	Ethernet Open Failed		Check if IP is correct, see if it can be scanned, the network link is normal?
20009	Handle Error		Check if the network breaks because of wiring / unstable network.
20010	Sending Error		
20011	File Error: Unsupported Head File, Unrecognizable		
20012	File Length Error		
20013	Too Many Filename		Check project related filename, whether it is too long, if yes, then correct.
20014	File Not Exist		Check project file's related folder, maybe one file lacks, usually it appears when deleted the file without IDE operation.
20015	ZLB Library File Error		
20016	File Not Compile	Usually because PLC file is not compiled.	Generally, PLC file has no compile.
20018	Firmware File Error	Usually because firmware file is broken.	
20020	Incorrect Firmware File		Check if updated firmware is consistent with controller model.
20021	Unsupported Function		1. Check the controller model, whether it supports this function 2. Check if there is new function for the controller, if yes, try to update firmware.
20022	"mmap" Failed RT LOCAL Open Failed	RT memory config is too large.	

20023	“xplcterm” Runs Incorrectly / No Enough Permission		
20024	No Card / No Drive in PCI Link		
20025	Drive Enumerate Failed		
20026	Interface Enumerate Failed		
20027	Unknown		
20028	PCI Card Not Exist		
20029	Too Many PCI Cards Connected		
20030	Insufficient IN Buffer Length		1. Check if defined function name, para name are too long. 2. Check the firmware version, if it is low, then this length of IN command name is not supported.
20031	Password Protection, Return after LOCK		
20032	Password Protection, Too Fast to Unlock.		
20033	File Open Failed		
20034	Unsupported Function		
20035	Too Long Message		
20036			
20037	Too Many Parameters		
20038	Report Para Numbers Error		
20039	No Assigned Para in Report Para		
20040	MotionRT Connect Failed MotionRT Not Opened		
20100	Response Buffer Length Not Enough		
30000	Above 30000 – ZAUX Auxiliary Library Errors		

Appendix II Module Expansion

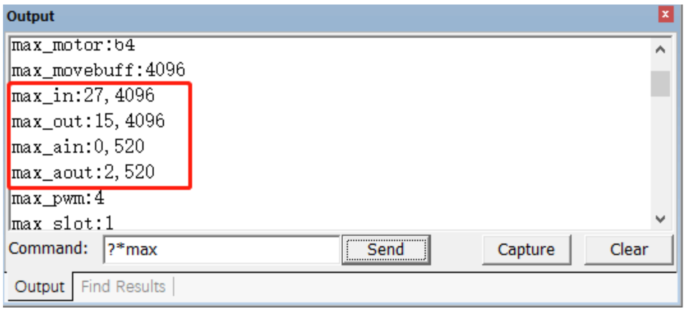
Module expansion is used to expanse pulse-axis, digital inputs & outputs and analog inputs & outputs when there is no enough axis resource and IO resource on controller. Pulse-axis extension is only valid in expansion module with pulse interface, which means bus axis can't be expanded.

IO (digital input and output): IO points of ZMC4XX series and above can reach 4096.

AIO (analog input and output): AIO points of ZMC4XX series and above can reach 520.

ZCAN fieldbus expansion: it only can extend 4 pulse axes, but it is not recommended to use axis expanse board too much, controllers with multiple pulse axes can be used.

Maximum IO expansion points can be check in hardware manual, or input "?*max" in the "COMMAND AND OUTPUT".



For connection way, there are ZCAN fieldbus and EtherCAT fieldbus module expansion, their expansion wiring and resource mapping methods are different.

For product series, there are three module expansion, ZCAN, EtherCAT and ZMIO300. ZMIO300 series communication modules are CAN communication module and EtherCAT communication module.

All controllers include CAN bus interface, but EtherCAT interface is only valid in EtherCAT fieldbus Controller.

After expansion module and controller wiring, there needs to operate map, then expanded IO and axis resource become useful. CAN fieldbus expansion map method differs from EtherCAT bus, the mapped NO. should not repeat in the whole control system when do map, if IO NO. range of controller or expansion module repeat, only one is valid.

ZCAN Expansion Module

Expansion wiring

When CAN fieldbus links with multi CAN expansion module, all CANL and CANH interface of CAN communication module link together separately, and connect a 120ohms resistance between 2 sides.

Expansion module CAN ports:

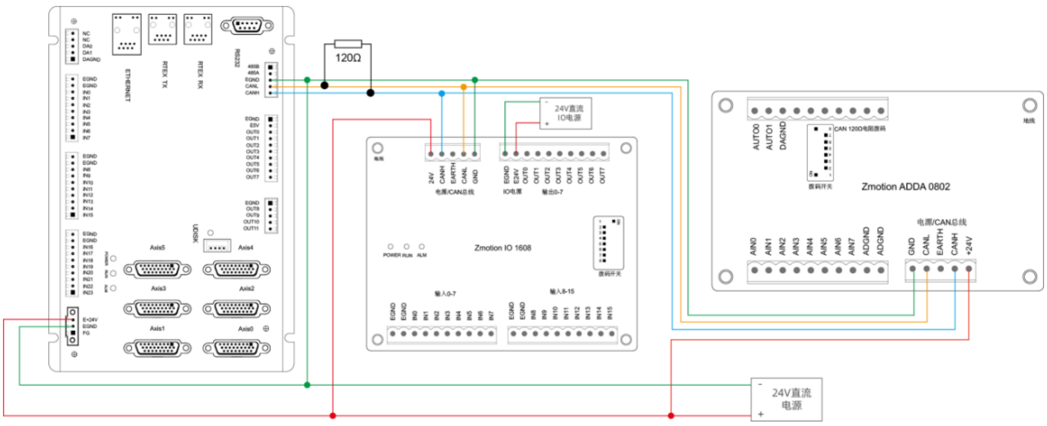
Stitch NO.	Name	Description
1	GND	Internal power position
2	CANL	CAN differential data-
3	EARTH/SHIELD	Shield layer
4	CANH	CAN differential data+
5	+24V	Internal power 24V input

Wiring method of controller and CAN expansion module as followed picture, connect a 120 ohms resistance between CANL and CANH, and the eighth bit of the last CAN communication module DIP as ON(there connected a 120ohms resistance between CANL and CANH), others no need to operate, just operate the terminal expansion module.

CAN communication must link with relevant GND, or main power of controller and expansion module should be the same one, prevent expansion modules from burning out.

ZCAN wiring can refer this: ZMC432+ZIO1608M+ZAIO0802M, CAN expansion uses a twisted-pair shield cable, and the shield layer is grounded.

ZIO expansion module needs main power and IO power, double power supply, it will be useless when no IO power. ZAIO expansion module only needs main power supply.



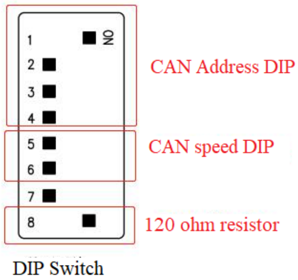
Resource mapped

ZCAN expansion module resource become useful after mapping, IO map use dial switch setting of expansion module itself, axis map uses AXIS_ADDRESS instruction.

There is slight difference of mapped NO. rule between IO and AIO, details as follow.

IO mapped

ZCAN expansion board usually with 8-bit dial switch, dial ON open, as follow:



1-4: 4-bit CAN ID is used to ZCAN expansion module IO address map, relative value is 0-15.

5-6: CAN communication speed, relative value is 0-3, and there are 4 speed values.

7: reserved

8: a 120ohms resistance, dial ON, which means there has connected a 120ohms resistance between CANH and CANL

When dial 1-4 to choose CAN address, set relevant expansion IO NO. range according to CAN dial address, set every bit OFF value is 0, ON as 1, address combined value=dial code 4 × 8 + dial code 3 × 4 + dial code 2 × 2 + dial code 1.

Dial switch should be dialed well before power on, dial again after power on is invalid, which means it needs power on again.

Digital start IO mapping from 16, and increases as multiple of 16

Dial 1-4 combination value	Start IO NO.	End IO NO.
0	16	31
1	32	47
2	48	63
3	64	79
4	80	95
5	96	111
6	112	127
7	128	143
8	144	159
9	160	175
10	176	191
11	192	207
12	208	223
13	224	239
14	240	255
15	256	271

Dial 1-4 combination value	Start AD NO.	End AD NO.	Start DA NO.	End DA NO.
0	8	15	4	7
1	16	23	8	11
2	24	31	12	15
3	32	39	16	19
4	40	47	20	23
5	48	55	24	27
6	56	63	28	31
7	64	71	32	35
8	72	79	36	39
9	80	87	40	43
10	88	95	44	47
11	96	103	48	51
12	104	111	52	55
13	112	119	56	59
14	120	127	60	63
15	128	135	64	67

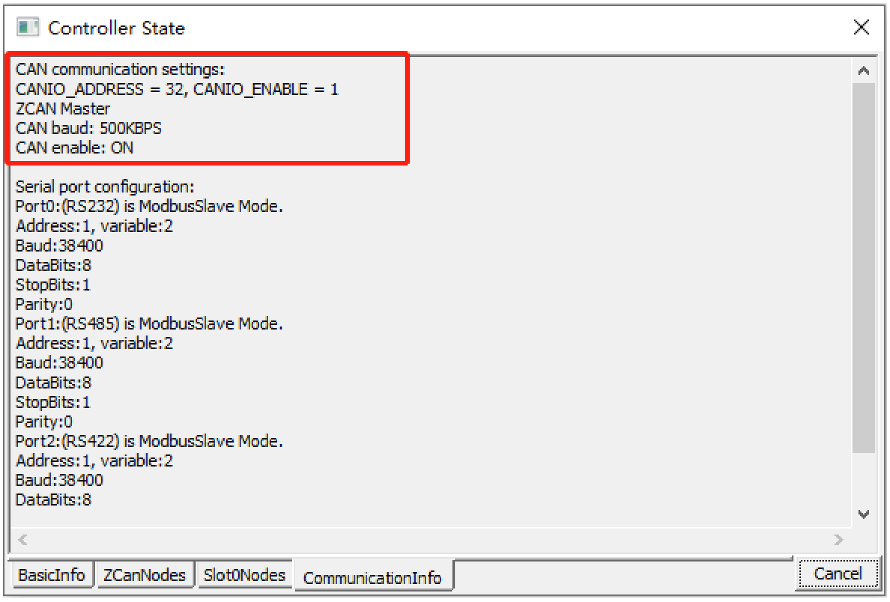
Dial 5-6 choose CAN fieldbus communication speed, speed combination value=dial 6×2+dial 5×1, combination value is from 0 to 3, relative speed as follow:

Dial 5-6 value	CANIO_ADDRESS high 8-bit value	CAN communication speed
0	0（is relevant to decimal 128）	500KBPS（default）
1	1（is relevant to decimal 256）	250KBPS
2	2（is relevant to decimal 512）	125KBPS
3	3（is relevant to decimal 768）	1MBPS

CAN communication speed of controller is set through CANIO_ADDRESS instruction, also there are 4 choices of speed parameters, but should be same as combination value related to expansion module communication speed value, then can mutual communication.

CANIO_ADDRESS instruction also can set CAN communication main station and slave station, default value is 32, as main port, set others as slave port.

CAN communication configuration can be check in “State the controller”.



Dial switch setting notes:

Expansion module dial switch according to IN of present IO points and OP maximum (external IO interface numbers+ pulse axis IO interface numbers)

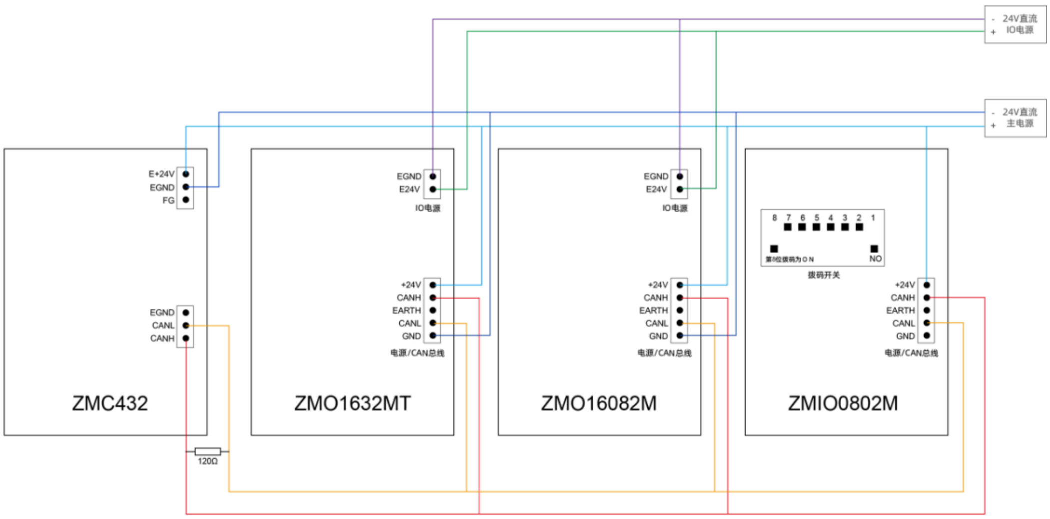
For example, controller has 28 IN and 16 OP itself, which means start address of the first expansion module should exceed 28, and address dial should be set as combination value 1 according to IO map rule(binary combination value is 0001, relevant dial 1-4 from right to left, dial 1 as ON, dial others as OFF), here the IO NO. on expansion module is 32-47, and 29-31, empty IO NO. won’t be used.

Following expansion module continues to dial set as IO point sequence.

When IO NO. range of controller and expansion module is the same, only one is valid. Recommended to reset dial, then IO NO. of the whole control system will not repeat.

ZCAN expansion module IO map configuration example:

Control module configuration: a ZMC432+a ZIO1632MT+a ZIO16082M+a ZAIO0802M



CAN wiring method refers to former picture, set correct dial code ID of every module, and set the eighth-bit dial code of the last expansion module as ON (means connected a 120ohms between CANL and CANH), use ZDevelop software to link controller, click “controller-state the controller” to check ZCAN node information, including all device information linked with CAN fieldbus.

Set CAN ID of ZIO1632 as 1, expanded digital input IO NO. are 32-47, all 16 amounts. Expanded digital output IO NO. are 32-63, all 32 amounts.

Set CAN ID of ZIO16082 as 1, expanded digital input IO NO. are 64-79, all 16 amounts. Expanded digital output IO NO. are 64-71, all 32 amounts. And there are 2 pulse axes.

Set CAN ID of ZAIO0802 as 1, expanded analog input AD NO. are 40-47, all 8 amounts. Expanded analog output DA NO. are 20-21, all 2 amounts.

控制器状态

CanID	硬件ID	轴数	输入	输出	AD	DA
Local	432-0(ZMC432)	32	30(0-29)	18(0-17)	0	2(0-1)
1	48(ZIO1632)	0	16(32-47)	32(32-63)	0	0
3	26(ZIO16082)	2	16(64-79)	8(64-71)	0	0
4	10(ZAIO0802)	0	0	0	8(40-47)	2(20-21)

基本信息 | ZCan节点 | 槽位0节点 | 通讯配置　　取消

Axis mapped

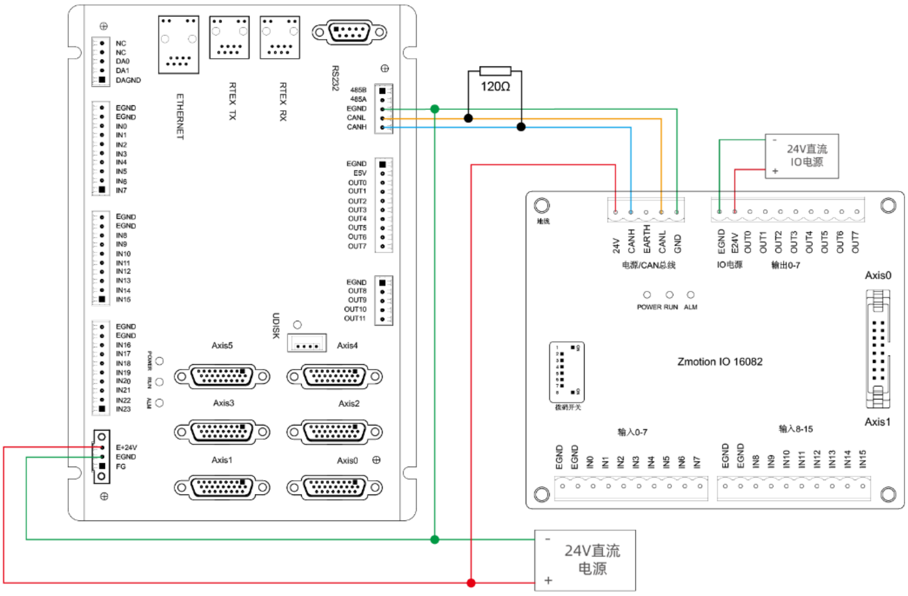
When expanse pulse-axis in CAN fieldbus expansion way, ZIO16082M can be chosen, and expanse 2 pulse axes.

Expanse axis needs to axis map, use AXIS_ADDRESS instruction, map rule as follow:

AXIS_ADDRESS(axis NO.)=(32*0)+ID 'local axis port 0 of expansion module

AXIS_ADDRESS(axis NO.)=(32*1)+ID 'local axis port 1 of expansion module

ID is the combination value of expansion module 1-4 bits address dial code.



After set axis parameters, it can use expansion axis, for example:

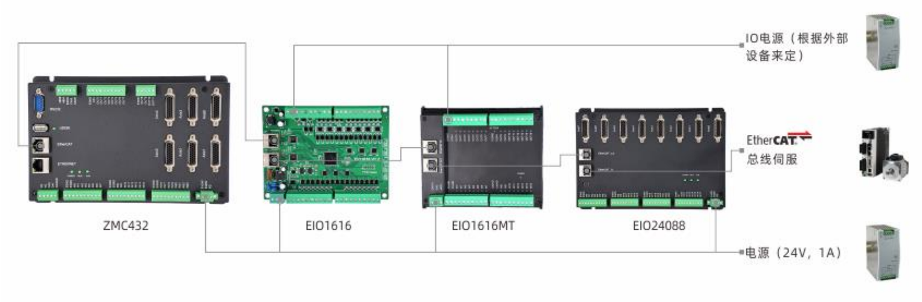
```
ATYPE(6)=0                      'set as virtual axis
AXIS_ADDRESS (6)=1+(32*0)       'map axis 0 of CAN expansion module (ID=1) to axis 6
ATYPE(6)=8                      'ZCAN expansion axis, stepper in pulse direction or servo
UNITS(6)=100                    'pulse amounts 100
SPEED(6)=100                    'speed 100units/s
ACCEL(6)=1000                   'acceleration 1000units/s^2
MOVE(100) AXIS(6)               'expansion axis move 100units
```

EtherCAT Expansion Module

Expansion Wiring

EtherCAT expansion module wiring only needs EtherCAT interfaces of every module link with each other. EIO series expanse board with 2 EtherCAT interfaces, EtherCAT port 0 links with main controller, EtherCAT port 1 links with lower expanse board or drive device, they can not be used wrongly.

EIO expansion wiring reference: ZMC432+EIO1616+EIO1616MT+EIO24088.



Resource mapped

IO map on EtherCAT bus uses NODE_IO instruction(digital) and NODE_AIO instruction(analog), axis map uses AXIS_ADDRESS instruction.

Slot NO. and device NO. follow the liking sequence with controller, and start from 0.

IO mapped

NODE_IO instruction sets start NO. of device digital IO, single device input and output start NO. is the same. It should wait until fieldbus scan successfully, then set. NODE_AIO and NODE_IO instructions are the same basically.

Grammar:

NODE_IO(slot, node)=iobase

slot: slot NO., 0-default

node: device NO., start from 0

ioBASE: mapped IO start NO., result only is the times of 8

NODE_AIO(slot, node[,idir])=aiobase

slot: slot NO., 0-defualt

node: device NO., start from 0

idir: select AD/DA. 0-default, and set AIN and AOUT at the same time, but only read AIN, 3-AIN, 4-AOUT.

IO mapped example: ZMC432 controller links with 2 EtherCAT expansion module as sequence. Configuration: a ZMC432 + a EIO1616MT + a ZMIO-4AD.

```
SLOT_SCAN(0)                    'scan fieldbus
IF NODE_COUNT(0)>0 THEN         'judge there is device on slot 0
    NODE_IO(0,0)=32             'set device 0 IO start NO. of slot 0 as 32
    NODE_AIO(0,1,3)=8           'set device 1 AIN start NO. of slot 0 as 8
```

ENDIF

Axis mapped

Fieldbus axis needs to be axis mapped, use AXIS_ADDRESS instruction, operation ways as follow:

AXIS_ADDRESS(axis NO.)=(slot NO. <<16)+drive NO. +1

Axis map should be written in the fieldbus initialization procedure, after fieldbus is scanned, before open fieldbus.

For example:

AXIS_ADDRESS (0)=(0<<16)+0+1 'the first ECAT drive, drive NO. is 0,binding with axis 0

AXIS_ADDRESS (0)=(0<<16)+0+1 'the second ECAT drive, drive NO. is 1, binding with axis 2

AXIS_ADDRESS (0)=(0<<16)+0+1'the third ECAT drive, drive NO. is 0,binding with axis 0

ATYPE(0)=65 'set as ECAT axis type, 65-position, 66-speed, 67-torque

ATYPE(1)=65

ATYPE(2)=65

Appendix III HMI Communication

Controller and HMI Communication Introduction

Controller or HMI usually is linked through serial port or net port, serial port and net port of controller use MODBUS protocol, HMI with MODBUS communication protocol can be used with Zmotion controller, also with ZHD series HMI, which is developed by Zmotion itself. ZHD400X as follow:

When controller uses MODBUS protocol communicate with the third party, the data should be passed in the MODBUS register. Controller will program more flexibly and free matched with ZHD series HMI.

There are some differences between controller MODBUS address and other manufacturers' HMI address map relations. The relation between controller and HMI modbus register address as follow:

Controller MODBUS address starts from 0, when do communication with WEINVIEW, all address starts from 0, so they are relative.

Controller MODBUS_BIT(0) is relevant to WEINVIEW MODBUS_0X-0, Boolean type.

Controller MODBUS_REG(0) is relevant to WEINVIEW MODBUS_4X_0, word register type.

When do communication with MCGS, MCGS address starts from 1, controller address starts from 0, so HMI address adds 1.

Controller MODBUS_BIT(0) is relevant to MCGS MODBUS_0X_1, Boolean type.

Controller MODBUS_REG(0) is relevant to MCGS MODBUS_4X_0, Word register type.

Controller procedure can use ZBASIC and PLC to program, for ZHD series, use HMI.

Connect Controller with HMI

Normal step:

1. Download program written in ZDevelop into controller.

2. HMI program written by relevant programming software is downloaded and saved in HMI.

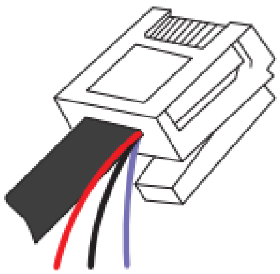
3. After program is downloaded, choose serial port or net port link with HMI, and controller run offline.

Connect with ZHD Series HMI

It is convenient to connect Zmotion Controller with ZHD300X and ZHD400X Series of Zmotion HMI, HMI procedure can be downloaded into controller, followings are ZHD400X usage methods, the difference between ZHD300X and ZHD400X is, the former is RS232 serial communication, the latter is net communication, but other configurations are the same.

ZHD400X matches with a net line, and link it to EtherNET controller net port, three cables are led out from the edge of the crystal head of the network cable, namely the power cable of the teaching box and the emergency stop signal cable. The red cable is the positive pole of the 24V power supply, the black cable is the negative pole of the 24V power supply, and the purple cable is the emergency stop signal cable.

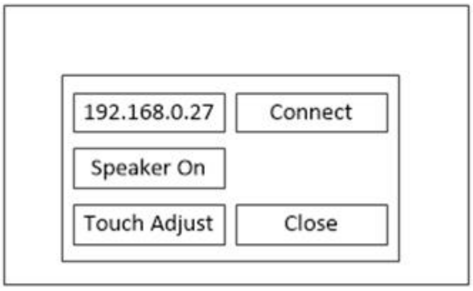
Main power of HMI and controller can be the same.



Steps for connecting HMI to controller:

1.use ZDevelop software to write HMI program, then connect to controller, download the program into ROM for storage when power off, next, disconnect controller and ZDevelop and power on the HMI.

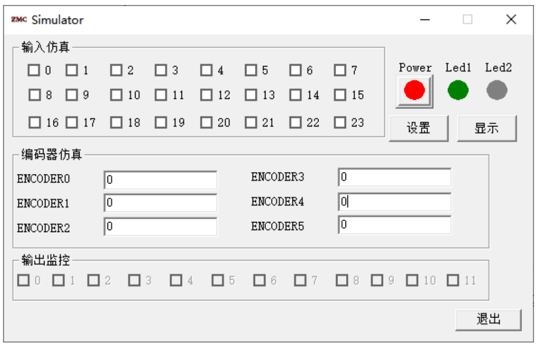
2.connect the ZHD400X directly to the network port of the controller using the provided cable, and then click on the four corners of the screen in the order of drawing a Z, 2 times in a row, wake up the screen, and a setting window will pop up to perform touch correction, controller IP modification, etc.



3.below is the setting window, and gain the current connected controller IP address from jumped window automatically, please confirm IP is correct, then click Connect. Now, HMI shows starting basic content.

4.if there is no real HMI, it can download HMI program into simulator, then simulate on XPLC screen platform.

After connecting simulator and downloading program, click “显示” , the simulation page will appear.



Connect to the third-party HMI

The touch screen that supports the standard MODBUS protocol can communicate with ZMOTION motion controller, communication data is put in the MODBUS register to transmit, and support connecting to the controller through the serial port or the network port.

When the touch screen and the controller establish a communication connection, the connection is mainly operated on the touch screen side, and the corresponding serial port or network port parameters should be matched when connecting.

The available register types for communication are as follows: MODBUS_BIT (Boolean), MODBUS_REG (16-bit integer type, MODBUS_LONG (32-bit integer type), MODBUS_IEEE (32-bit floating point type), MODBUS_STRING (8-bit byte type).

Communication example between the controller and the third-party touch screen: Take the communication between the controller and the Weilun screen as an example to expand the use of the touch screen.

1. Download controller program

Program of controller is programmed by ZDevelop software and downloaded into controller.

2. Download HMI program

Program of touch screen is programmed by EasyBuilder software, after programming, then open "system parameter setting" window, please see below.

1) Add devices to be connected with touch screen

Device list shows local touch screen and local device, if there is local device, please double click this line, if there isn't, click "new build device/servicer" like the below, then device property window will jump.

2) Set device property

Like above, select device type, first select MODBUS IDA communication protocol, then select according to actual connection method of touch screen and controller.

There is different between serial port communication and net port communication, please see following for details.

If connects through serial port:

Device type: select mode MODBUS RTU (Zero-BASEd Addressing)

Interface type: select serial port (RS485/RS232)

COM: Baud rate matched with communication port and other parameters, see below, now, the parameter must be same as port parameter connected on controller. After setting, confirm that system parameter setting window is closed.

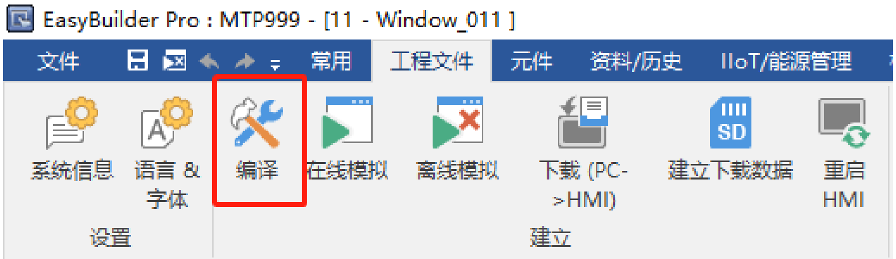
If connects through net port:

Device type: select mode MODBUS RTU (Zero-BASEd Addressing), interface type will changed into Ethernet automatically.

IP: fill the IP address and port number of controller that is to be connected currently. See below:

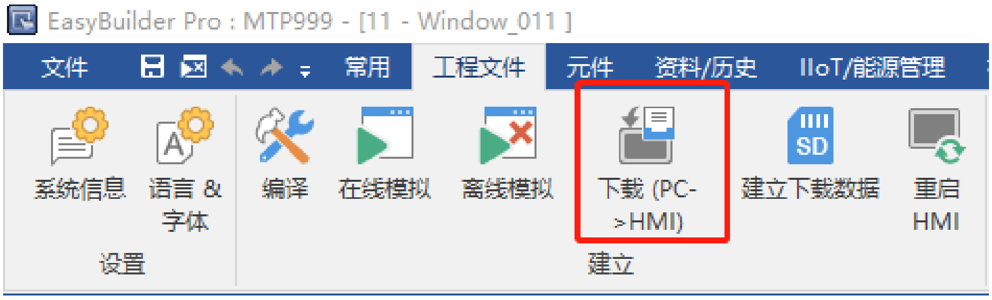
After setting, confirm that system parameter setting window is closed.

After system parameters are set, compile written configuration program, click “编译”, open the window.



Click right corner “开始编译”, if compiling successfully, there will print information, and “开始编译” will become “编译”, if program is incorrect, the window will print error information, then need to modify program until it compile successfully.

If it is successfully, connect touch screen to PC and download the program.



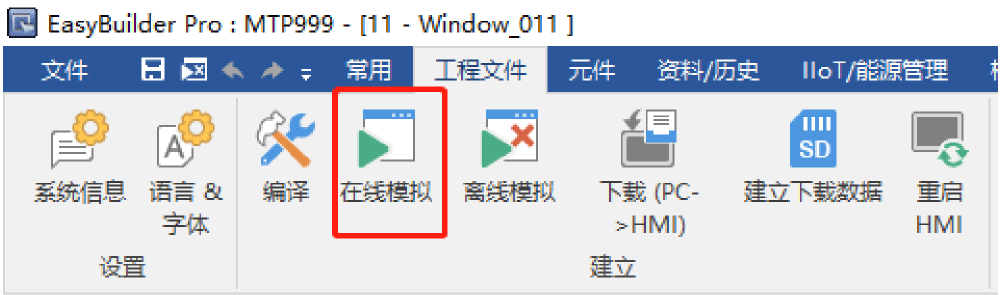
Click download, program is downloaded into HMI through Ethernet, and after downloading, the program has been written into touch screen, now, disconnect touch screen and PC.

3. Touch screen communicates with controller

After the program on the controller side is successfully downloaded to the controller, and the program on the touch screen side is successfully downloaded to the touch screen, it can be disconnected from the PC, and the touch screen and the controller can be connected. At this time, the touch screen and the controller can communicate with each other.

4. Controller and touch screen simulate offline

If there is no controller or touch screen, it can use simulator. The ZDevelop program is downloaded to the simulator and only supports network port connection. According to the above steps, when setting the system parameters of the EasyBuilder software, select the device type as MODBUS IDA—MODBUS TCP/ IP (Zero-BASEd Addressing), fill in the IP address of the simulator IP: 127.0.0.1, select "Online Simulation" to connect the controller program and the configuration program for simulation.



After clicking the online simulation, the compilation will start automatically, and the compilation result will correctly open the following touch screen simulation interface, which can be operated at this time. If the compilation is unsuccessful, an error message will be reported.

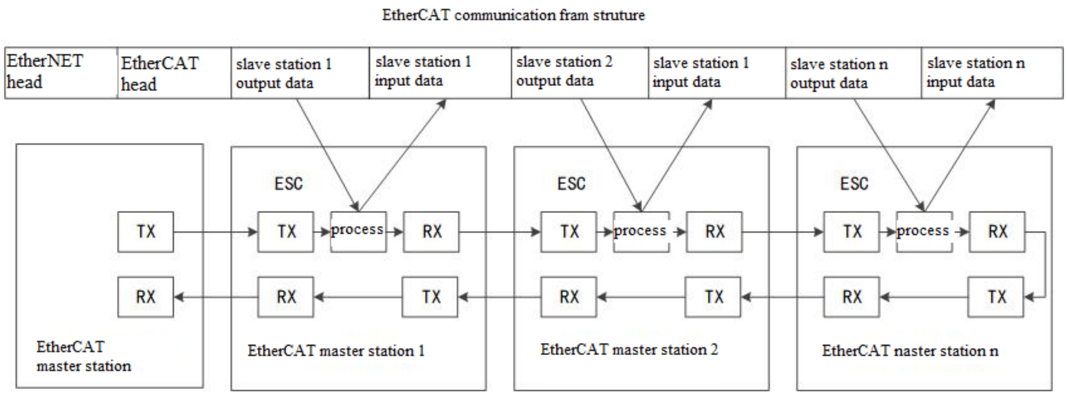
Touch screen simulation interface:


轴的目标位置
X轴坐标
171.55
Y轴坐标
0.00
X轴回零
OK
Y轴回零
OK
启动
停止
回零
保存数据
加工运动参数
运行状态
停止
圆弧半径
100.00
跑道长度
300.00
手动运动模块
X轴手动运动
正向
负向
Y轴手动运动
正向
负向
Fast Sel

Appendix IV ETHERCAT Communication

EtherCAT bus is a real-time industrial field bus communication protocol based on Ethernet development architecture. It is currently one of the fastest industrial Ethernet technologies, providing nanosecond-level precise synchronization, high performance, flexible topology, low cost, high precision, the application is simple and so on.

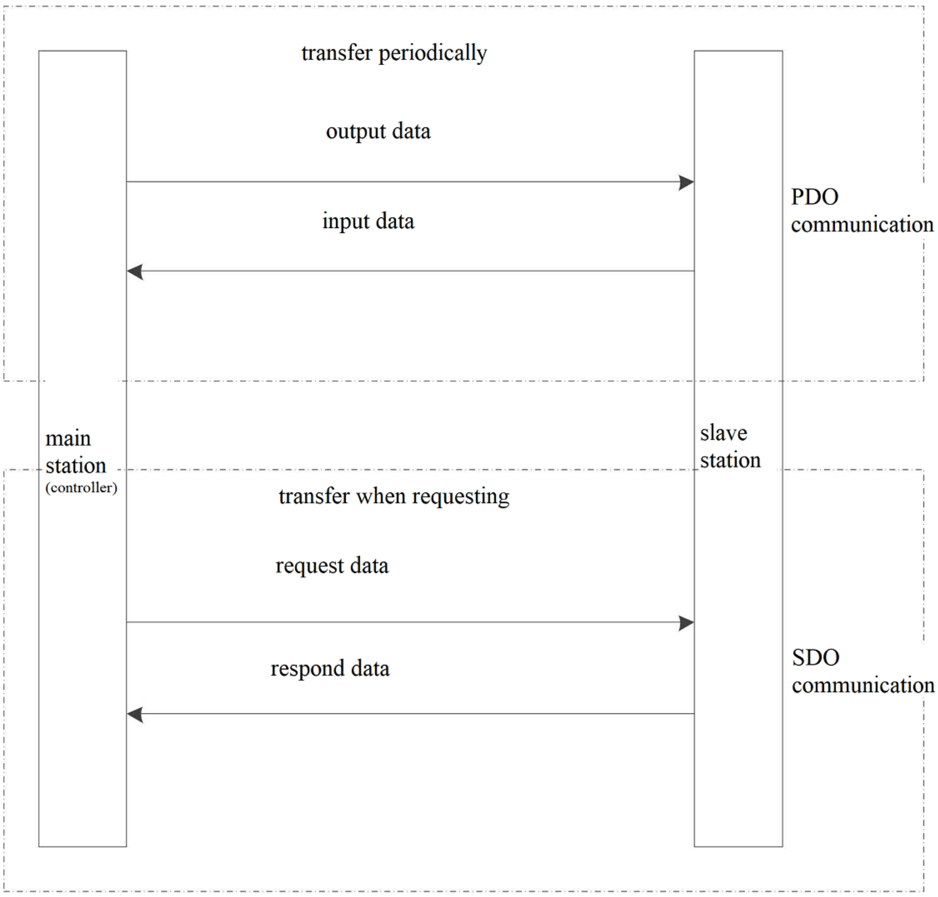
EtherCAT takes full advantage of the full-duplex nature of Ethernet, using master-slave mode media access control. The EtherCAT network is obviously different from the ordinary Ethernet. In the same EtherCAT network, there is only one EtherCAT master station, and the EtherCAT slave station has a chip ESC (EtherCAT Slave Controller) specially processing EtherCAT communication data. The ESC chip can take out the data sent by the master station to the slave station when the EtherCAT data frame passes, and insert the data that the slave station needs to transmit to the master station into the EtherCAT data frame, the last EtherCAT slave station in the network ESC automatically close the loop and return the processed messages to the master station in turn. The data transmission diagram is shown in the figure below:



Controller EtherCAT communication port and EtherCAT slave station transfer data through COE (CANopen over EtherCAT) protocol.

There are 2 ways to transmit data between controller and slave station, one is data-transmitting periodically as defined time, this is called PDO(Process Data Object), another is request-response data-transmitting, this is called SDO(Service Data Object).

EtherCAT fieldbus communication process:



Process Data Object (PDO)

PDO means periodical data interaction function between master station and slave station in EtherCAT Bus network. PDO data is used for periodical data reading and controlling, and write & read speed is fast. When master station and slave station interact data through PDO, one side sent the data, another side no need to respond. When controller controls EtherCAT slave station through motion instructions, then controller and slave station interact data through PDO.

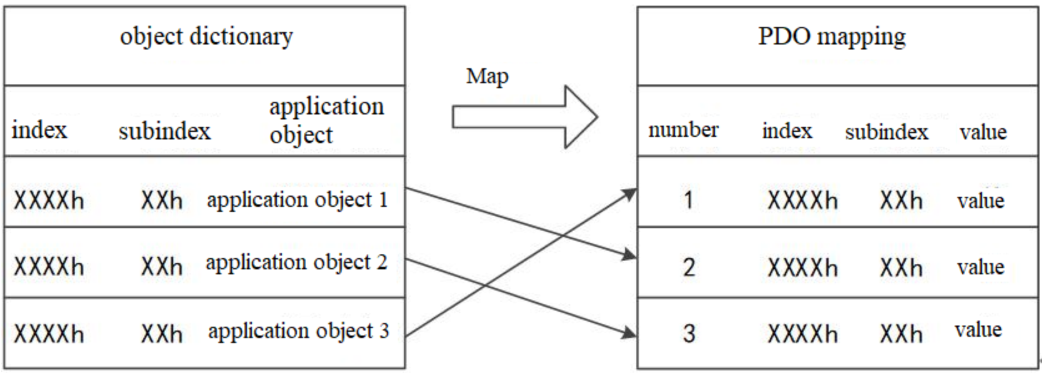
Drive PDO must be configured in EtherCAT initialization, and PDO list of drive is configured by DRIVE_PROFILE. Currently, there are more than 20 kinds of configuration to be selected, each configuration includes data dictionary description. If DRIVE_PROFILE can't meet, PDO can be self-defined, using SDO related instructions to operate data dictionary for configuring needed PDO.

The PDO list can be regarded as an array space. Each array element stores different function codes. The PDO executes the operations corresponding to these function codes in one cycle. These function codes are called the data dictionary. The data dictionary uses 4-digit hexadecimal numbers. To indicate that the planning method is through the corresponding PDO mapping and PDO parameter index in the object dictionary.

There are two types of PDOs: TxPDO for transmission and RxPDO for reception. A node's TxPDO is to transmit data from this node to other nodes, while RxPDO is to receive data transmitted by other nodes. A node has 4 TxPDOs and 4 RxPDOs respectively. Each byte in the data field of the PDO message is used for data transmission, so the message utilization rate is high.

All transfer data in the PDO must be mapped in the object dictionary:

After configuration, the transmission sequence of PDO is: application object 3, application object 1, application object 2.



Service Data Objects (SDO)

SDO data is used to send communication data when the master needs to read or write the parameters of the slave. In this way, only the master station can read or write the data of the slave station. After the master station sends the data, the slave station needs to respond.

SDO can be used to access the object dictionary of the remote node, read or set the data in it. The self-defined configuration of the read-write PDO list of the data dictionary is realized through the instructions SDO_READ, SDO_READ_AXIS and SDO_WRITE, SDO_WRITE_AXIS.

SDO messages contain index and sub-index information so that objects can be easily located in the object dictionary, and the complex data structures in the object dictionary can be easily accessed through SDO. The triggering method of SDO is command response type, that is, after the SDO client sends a read/write request, the SDO server must respond, both the client and the server can actively terminate the transmission of SDO, the request message and the response message pass through different COB-IDs differentiate.

SDO can transmit data of any length. If the data to be transferred exceeds 4 bytes, a fragmented transfer must be performed. The last piece of data contains an end marker.

Data Dictionary:

EtherCAT communication operation object dictionary, which is an ordered group of objects, each object is addressed with a 4-bit hexadecimal index value, in order to allow access to a single element in the data structure, an 8-bit sub-element is defined at the same time. Index, multiple data objects are combined into a data dictionary, also known as PDO list.

Each node has an object dictionary that contains all the parameters describing the device and its network behavior. Refer to the following table for the structure of the object dictionary. The relevant range of the object dictionary of the node is between 0x1000-0x9FFF.

Index	Content
0x0001-0x0FFF	Protocol type description, data type, line rule type description, configuration form information.
0x1000-0x1FFF	Communication area
0x2000-0x5FFF	Function property of object self-defined by device manufacturer, it is used to set functional codes and static parameters.
0x6000-0x9FFF	Data object defined by line rule, it is used for device controlling and monitoring.
0xA000-0xFFFF	Reserved

Index 1600h~17FFH use RxPDO mapping configuration, when configured, it will be allocated to 1C12h. Index 1A00h~1BFFH use TxPDO mapping configuration, when configured, it will be allocated to 1C13h.

Normal data dictionary reference:

Index	subindex	Name	Data Range	Data Type	W/R	PDO	Control mode	EEPROM
6040h	00h	Control word	0-65535	U16	RW	RxPDO	All	NO
6041h		State word			RO			
6060h		Control mode set	-128~127	18	RW			YES
6061h		Control mode check			RO	TxPDO		NO
6071h		Target torque	-32768 ~ 32767	I16	RW	RxPDO	tq, cst	YES
6072h		Max torque	0-65535	U16			All	
6077h		Actual torque	-32768~32767	I16	RO	TxPDO	All	NO
607Ah		Target position	-2147483648 ~ 2147483647	I32	RW	RxPDO	pp, csp	
607Eh		Motor Polarity	0-255	U8	RW	NO	All	YES
6091h	01h	Electronic gear ratio numerator	1~4294967295	U32				
	02h	Electronic gear ratio denominator						
6098h	00h	Homing mode	-128~127	18		RxPDO	hm	
60FDh	00h	Digital input	0~4294967295	U32	RO	TxPDO	All	NO
	01h	Digital output			RW	RxPDO		YES
60FFh	00h	Target speed	-2147483648 ~ 2147483647	I32	RW		pv, csv	NO

(W/R: write or read. W: write, R: read, RO: read only, RW: write only)

Different values indicate different functions, and please refer to drive manuals description to set. When some parameters are set well, they are written into drive failure storage memory, and restart drive for taking effect.

Appendix V RTEX Bus

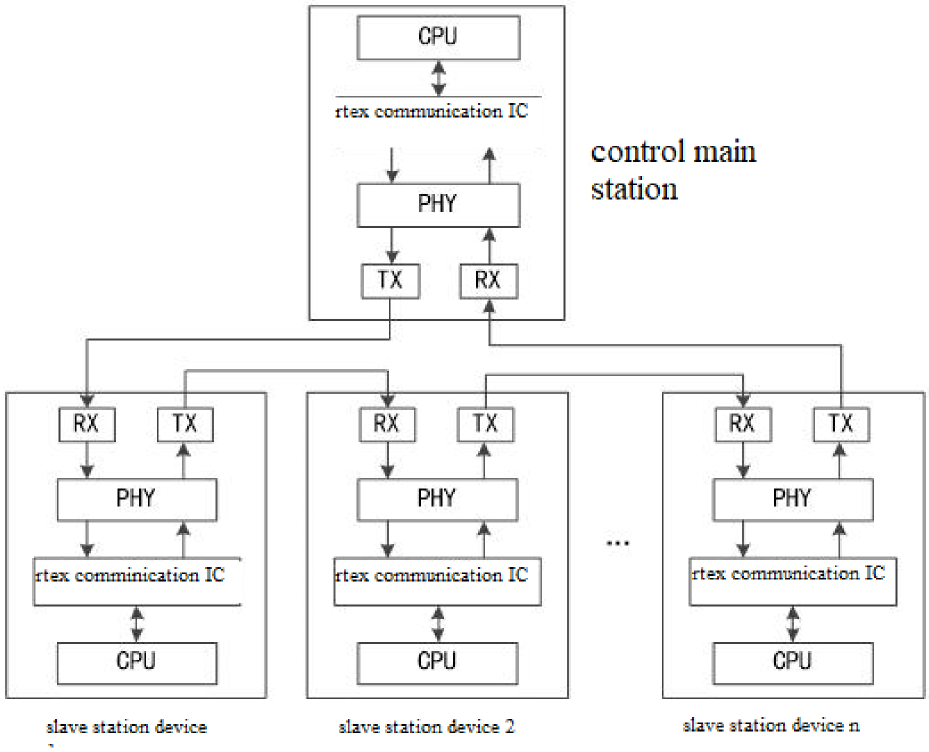
Some models of ZMOTION motion controller support RTEX bus, RTEX fieldbus axis, EtherCAT bus axis, pulse axis joint interpolation. RTEX bus is high-speed network fieldbus developed by Panasonic, which suits to real-time bus of small system, and the pipeline composed of small equipment is more flexible and faster.

Currently, RTEX bus supports 32 nodes, and each whole data package includes 32 nodes output information and feedback information, it has 64 data blocks totally. Additionally, RTEX bus provides control word register and status word register. And each data block is 16bytes, only including needed position, speed information and other command word and status word. Take main station based on RTEX bus as core, main computer sends control commands to all nodes once, at the same time, it gains all feedback signals of nodes, then it can complete control for all node outputs.

The master station equipped with the RTEX communication IC and the slave station are connected in a ring to form a multi-axis servo communication system. The structure is as follows, and the PHY is the physical layer chip. Shielded twisted pair cables should be used for connecting wires.

When the synchronization bit of communication and servo is established, the timing of command reception and corresponding transmission is uncertain. Whether the synchronization is completed can be judged by reading the current state of the command.

The RTEX communication IC includes a sending memory, a receiving register, a control register and a status register. The sending memory is used to store data instructions, and the receiving register is used to store the response data.



RTEX parameter writing and reading use DRIVE_READ and DRIVE_WRITE to operate below drive parameters.

Relative setting parameters:

Type	No.	Property	Name	Range	Unit	Description
0	00	C	Set rotate direction	0~1	-	Set relation between indication direction and motor rotation direction. 0-CW is positive, 1-CCW is positive
0	01	R	Set control mode	0~6	-	Set control mode of servo drive 0-half closed loop control position/speed/torque control mode can switch 1-full closed loop control only position control (contour/period)
0	08	C	Instruction pulse amount as per round of motor rotates	$0\sim 2^{23}$	pulse	Set the number of pulse when motor rotates one round
0	09	C	Electronic gear ratio numerator	$0\sim 2^{30}$	-	Set the numerator of electronic gear ratio
0	10	C	Electronic gear ratio denominator	$0\sim 2^{30}$	-	Set the denominator of electronic gear ratio

0	13	B	The first torque limit	0~500	%	Set the first limit value of torque output by motor, parameter value is limited by the max torque of motor.
3	12	B	Acceleration time	0~10000	ms	Set the time to accelerate
3	12	B	Deceleration time	0~10000	ms	Set the time to decelerate
3	14	B	S acceleration and deceleration time	0~1000	-	To do S curve process for acceleration and deceleration
3	17	B	Speed limit value selection	0~1	-	Select speed limit: 0-speed limit 1, 1- speed limit 2
3	21	B	Speed limit value 1	0~20000	r/min	Set speed limit value, internal value is limited by Pr5.13 (pass speed level), Pr6.15 (the second pass speed level) and the minimal and internal set speed of pass speed protection level.
3	22	B	Speed limit value 2	0~20000	r/min	set speed limit value when Pr3.17 (speed limit selection) = 1 and SL_SW = 1. Internal value is limited by Pr5.13 (pass speed level), Pr6.15 (the second pass speed level) and the minimal and internal set speed of pass speed protection level.
5	21	B	Torque limit selection	1~4	-	Set torque limit selection method: positive / negative. When sets as 0, set internal as 1.
5	22	B	The second torque limit	0~500	%	Set the second limit value of torque output by motor, and motor max torque limit value
5	25	B	Positive torque limit	0~500	%	When Pr5.21(torque limit selection) = 4 and TL_SW = 1, set positive torque limit, parameter values are

						used for max torque limit of motor.
5	25	B	Negative torque limit	0~500	%	When Pr5.21(torque limit selection) = 4 and TL_SW = 1, set negative torque limit, parameter values are used for max torque limit of motor.
5	26	A	Software limit function	0~3	-	Set valid / invalid software position limit function when in contour position control (pp). Valid software position limit value, it is set through Pr3.11 (positive software position limit value) and Pr7.12 (negative software position limit value) 0-two sides software position limit are valid. 1-only negative side is valid 2-only positive side is valid 3-both are invalid Due to this set value and invalid position limit signal (PSL/NSL), RTEX communication state is 0, also it is 0 when homing reset does not finish.
7	11	A	Positive software position limit value	-1073741823 ~ 1073741823	Instruction unit	When positive/negative software position is over limit, RTEX communication state PSL/NSL will become ON(=1) Positive software position limit value must be bigger than negative value.
7	12	A	Negative software position limit value			
7	20	R	RTEX communication period	-1~12	-	Set RTEX communication cycle -1: set Pr7.91 as valid 3: 0.5ms 6: 1.0ms

7	21	R	RTEX instruction update period ratio setting	1~2	-	Set the ration of RTEX communication period and instruction update cycle. Set value = instruction update value / communication period 1: 1 time 2: 2 times
7	22	R	RTEX function expansion	-32768 ~ 32767	-	bit0 sets RETX communication data 0: 16 bytes mode 1: 32 bytes mode bit1 uses TMG_CNT multi-axis synchronization mode, please set as 0 when no use. 0: half-synchronization in axes (some are not synchronized) 1: full-synchronization in axes When in bit4 half closed loop control, external distance sensor position information function setting. 0: invalid 1: valid (when in full closed loop, it is no relevant to this bit setting, it can monitor external distance sensor position)
7	91	R	RTEX communication period expansion	0~2000000	ns	When Pr7.20 = 1, RTEX communication period only can be set as 62500, 125000, 250000, 500000, 1000000 and 2000000. Otherwise, there will be Err93.5 “parameter setting abnormal protection 4”

A: always valid

B: prohibit modifying parameters when motor is in motion or when the instruction is sending.

C: it becomes valid after controlling power reset, controlling software reset mode of RTEX communication reset instruction or property C parameter valid mode finished.

R: control power restart and become valid

RTEX communication cycle (Pr7.20, Pr7.91) and instruction update cycle (Pr7.21) need to be consistent with cycle of upper equipment. At the same time, RTEX expansion functions (Pr7.22) should be same as upper equipment, if they are different, the motion can't be executed.

Mode setting example as below: communication period is 0.5ms, instruction update period is 1ms, half closed loop, 16 bytes mode, under half synchronization mode in axes.

Pr0.01=0 (half closed loop control)

Pr7.20=3 (communication cycle is 0.5ms)

Pr7.21=2 (instruction update cycle 1ms=05ms*2 times)

Pr7.22=0 (16 bytes mode, half synchronization mode in axes)

(when Pr7.20 is not equal -1, Pr7.91 can be set optionally)

Possible reasons if drive doesn't move:

Number	Item	Description
0	No reason	Reason of not rotation can't be checked, usually in rotatable status
1	Servo is not in preparable status	Main power of drive doesn't input Alarm emerges. Synchronization between communication and servo doesn't finish. Attribute C parameter validation mode processing under restart command is medium
2	Servo is not enabled	Servo ON command is not input, Servo_On bit is 0, EX_SON (external servo ON input) did allocation, signal is OFF, etc.
3	Drive prohibit input is valid	When Pr5.05=0 ~ 1 (sequence when driving is prohibited, except for immediate stop), when Pr5.04=0 (driving prohibition input is valid), when the positive direction driving prohibition input (POT) is ON, the action command is positive direction, when the negative direction driving prohibition input (POT) is ON, the action command is negative direction. When Pr5.05=2 (sequence when driving is prohibited, except for immediate stop), when Pr5.04=0 (driving prohibition input is valid), it is no relation with motion instruction input or not, when the positive/negative direction driving prohibition input (POT) is ON, it stops.
4~5	Torque limit setting is small	Valid torque limit setting value, it is set below extra 5%.
7	Low position instruction input frequency	Position instruction of each control period is below 1 unit.

10	Command speed of RTEX communication is small.	For RTEX communication command speed, they are set below 30r/min.
11	Manufacturer use	-
12	Command torque of rtex communication is small.	For RTEX communication command torque, decrease to below 5% of extra torque.
13	Speed limit is small	When Pr3.17=0, Pr3.21 speed limit value is set under 30r/min. When Pr3.17=1, speed limit value of parameter assigned by SL_SW (Pr3.21/Pr3.22) is set under 30r/min.
14	Others	Not in 1~13, motor doesn’t rotate (small instruction, heavy load, lock, crash, drive, motor malfunction, etc.).