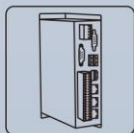
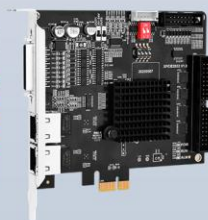
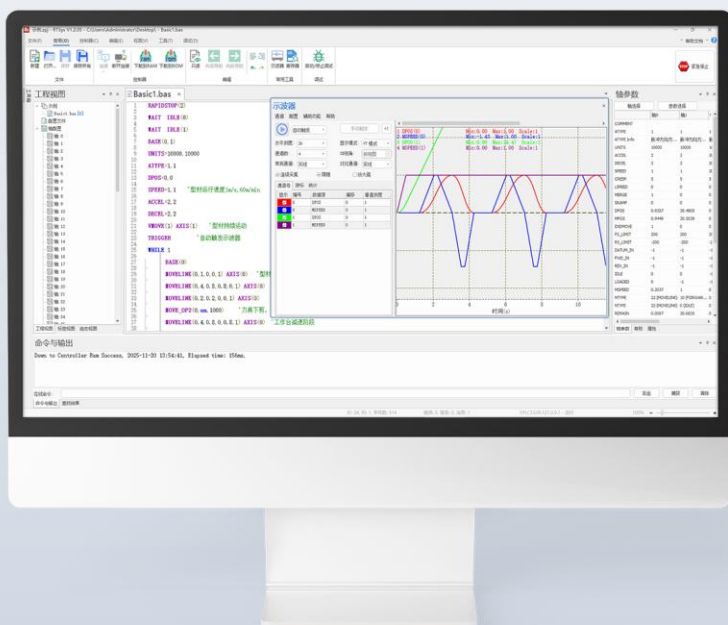
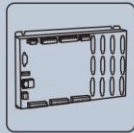


RTBasic Programming Manual

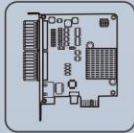
V1.1.6



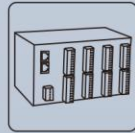
Vision Motion
Controller



Motion
Controller



PC-based Motion
Control Card



IO Expansion
Module



Teach
Pendant

Foreword

Zmotion Technology is a national high-tech enterprise, which focuses on motion control technology R&D (general motion control software & hardware products). In Zmotion, there integrates lots of experienced talents from famous companies or institutions, such as Huawei, Huazhong University of Science and Technology etc. At the same time, Zmotion insists self-innovating and collaborating with comprehensive universities to research basic knowledge of motion control. In this way, Zmotion has already become one of the fastest growing industrial motion control companies in China, also it is the rare company that has completely managed core technologies of motion control and platform technology of real time industrial control software.

All Zmotion products development obeys standard IPD-CMM from Huawei, which means they have stability and reliability of telecom level, excellent compatibility and expansibility of software or hardware. Zmotion provides powerful and convenient RTSys development environment, which can do programming (RTBasic / RTPlc / RTHmi), real-time simulation, and online tracking Debug. In addition, mainstream PC programming languages under different operation systems are supported, such as, C++ / C# / LabVIEW / Python, etc.

This manual is to better 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.

This manual includes the RTSys usage, detailed instructions, program operation logic descriptions, motion principle, controller introduction, the wiring reference between the controller and other components, and it can support multiple communication methods. What's more, application routines of typical industries are provided for programming reference.

Relative Programming Manuals:

Zmotion PC Programming Manual
Zmotion RTPlc Programming Manual
Zmotion RTHmi Programming Manual
Zmotion RTVision Programming Manual
Zmotion RTNC Programming Manual
Zmotion Robotic Arm Instruction Description Manual
RTSys Software User 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

This manual is copyrighted by Shenzhen Technology Co., Ltd., without the written permission of the Zmotion Technology, no person shall reproduce, translate and copy any content in this 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 1 Motion Control Products Introduction	1
1.1. Motion Control Products Overview	1
1.2. Motion Control Products Advantages	2
1.3. Motion Controller Main Functions	3
1.4. Motion Controller Applications	4
1.5. Motion Controller Interfaces	5
1.6. Motion Controller Usages	6
Chapter 2 RTSys Software Program	10
2.1. Program Software Introduction	10
2.2. New Project	11
2.3. Online Command and Output	15
2.4. How to Use Oscilloscope	18
Scope Interface	18
How to Configure Scope	21
How to Import & Export Scope Data	25
How to Sample by SCOPE	26
Scope Needs	26
Scope Usage Routine	27
2.5. Program Debug	31
Enter Program Debug	31
Task & Watch Windows	32
Usage of Debug Tool Bar	33
Breakpoint Debug	33
2.6. “View” Window	35
Chapter 3 Basis of Basic Programming	37
3.1. Programming Basic Knowledge	37
Program	37
■ Common Program Structure	37
■ Sub-Procedure	40
Data	41
■ Data Definition	41
■ Data Type	43
■ Data Operation	45
3.2. RTSys Three Programming Methods	48
Hybrid Programming	48
PLC & BASIC Call Each Other	48

3.3.	Register	49
	TABLE	50
	FLASH.....	51
	VR.....	52
	MODBUS.....	54
3.4.	Multi-Task Program	57
	Concept of Muti-task	57
	Check Multi-task Status	59
	Multi-Task Start & Stop.....	60
	Task Pause & Resume	62
	Basic & PLC Task Call Each Other	63
	Multi-Task Routine	64
3.5.	Three Kinds of Interruption	66
	Power Failure Interrupt.....	66
	External Interrupt	67
	Timer Interrupt.....	67
3.6.	Motion Buffer.....	68
	The Concept of Motion Buffer.....	68
	Motion Buffer.....	69
	Motion Buffer Overflow	70
	Output in Motion Buffer	72
Chapter 4	Communication Methods	74
4.1.	Serial Port Communication	74
	The Serial Port Type.....	74
	Serial Connection Method.....	76
4.2.	ETHERNET Communication	78
4.3.	CAN Bus Communication	81
	CAN Wiring	81
4.4.	U Disk Interface	82
4.5.	EtherCAT Communication.....	84
	EtherCAT Bus Initialization	84
	EtherCAT Bus & Drive Communication.....	88
	Connect to Expansion Module by EtherCAT	89
4.6.	RTEX Bus Communication.....	91
Chapter 5	Motion Control Functions	94
5.1.	Common Motion Modes	94
	Single-axis Jog Motion	94
	Electronic CAM	97
	Electronic Gear	99
	Handwheel	100
5.2.	Interpolation Motion	102
	Concept of Interpolation	102
	Continuous Interpolation.....	106
5.3.	Look-Ahead Processing	106

5.4.	Origin Homing	109
5.5.	Position Limit Related Instructions.....	115
5.6.	Position Latch	118
5.7.	Hardware Comparison Output	119
5.8.	Precision Output.....	121
5.9.	Galvanometer Control System	122
	The Description of Galvanometer	122
	Galvanometer Application Processes	126
5.10.	Robotic Arm.....	132
	The Concept of Robot	132
	Forward & Inverse Kinematics	134
	Valid Robot Functions.....	135
	Robot Examples	136
5.11.	G Code	140
Chapter 6	“Axis” Introductions.....	142
6.1.	Axis	142
6.2.	Axis No.	142
6.3.	Axis Status	144
6.4.	Axis Speed	145
	Speed Curve	145
	SP Speed.....	149
6.5.	Axis Mapping.....	151
6.6.	Axis Type	152
Chapter 7	Motion Instructions	157
7.1.	Single-Axis Motion Instructions	157
	ADDAX -- Motion Superposition.....	157
	CANCEL -- Single-Axis / Axis Group Stop	164
	DATUM -- Homing.....	168
	DATUM_OFFSET -- Origin Position Offset	173
	VMOVE -- Continuous Movement.....	174
	FORWARD -- Forward Movement.....	175
	REVERSE -- Negative Movement.....	176
	MOVEMODIFY -- Motion Position Modification	176
7.2.	Multi-Axis Motion Instructions	178
	RAPIDSTOP – All Axes Stop	178
	MOVE -- Linear Motion	181
	MOVEABS -- Linear Motion (Absolute)	183
	MOVEMODIFY2 -- Motion Position Modification 2	184
	MOVECIRC -- Center Arc.....	186
	MOVECIRCABS -- Center Arc (Absolute).....	188
	MOVECIRC2 -- 3-Point Arc	189
	MOVECIRC2ABS -- 3-Point Arc (Absolute).....	190
	MHELICAL -- Central Helical	191
	MHELICALABS -- Central Helical (Absolute)	193

MHELICAL2 -- 3-Point Helical	194
MHELICAL2ABS -- 3-Point Helical (Absolute)	196
MECLIPSE -- Ellipse.....	197
MECLIPSEABS -- Ellipse (Absolute).....	199
MSPHERICAL -- Space Arc.....	201
MSPHERICALABS -- Space Arc (Absolute).....	204
MOVESPIRAL -- Involute Arc.....	208
MOVESPLINE/MOVESPLINEABS -- Spline Interpolation.....	210
MOVE_TURNABS -- Rotary Table Interpolation.....	212
MCIRC_TURNABS -- Rotary Table Circular Interpolation	213
MOVESMOOTH -- Fillet	215
MOVESP -- SP Motion	216
*SP -- Motion Independent Speed.....	217
MOVESCAN -- Galvanometer (SCAN) Motion	219
MPULSCAN -- Galvanometer (SCAN) Motion 2.....	221
7.3. Special Motion Instructions	222
MOVE_PAUSE -- Motion Pause.....	222
MOVE_RESUME -- Motion Resume.....	224
MOVE_PT -- Unit Distance.....	225
MOVE_PTABS -- Unit Distance (Absolute)	228
MOVE_PVT -- Unit Distance (with Speed Planning)	232
MOVE_PVTABS -- Unit Distance (Absolute & Speed Plan)	235
MOVE_PVTPP -- Unit Distance 2	237
MOVE_PVTPPABS -- Unit Distance 2 (Absolute).....	239
MOVE_PTP -- Point to Point.....	241
MOVE_PTPABS -- Point to Point (Absolute)	244
MOVE_OP -- Output in Buffer.....	247
MOVE_OP2 -- Output in Buffer 2.....	251
MOVE_TABLE -- Table in Buffer.....	252
MOVE_PARA -- Parameters in buffer.....	253
MOVE_PWM -- PWM in Buffer.....	256
MOVE_SYNMOVE -- Synchronous Axis Trigger in Buffer	256
MOVE_ASYNMOVE -- Synch Axis Trigger in Buffer 2	257
MOVE_TASK -- Start Task in Buffer.....	258
MOVE_AOUT -- Analog Signal in Buffer	259
MOVE_DELAY -- Delay in buffer	260
MOVE_WAIT -- Wait in Buffer.....	261
MOVE_CANCEL -- Stop Buffer.....	262
MOVE_HWPSWITCH2 -- Hard Comparison Out in Buffer	262
MOVE_HWTIMER -- Hardware Timer in Buffer.....	263
MOVE_ADDAX -- Motion Superposition	264
MOVELIMIT -- Speed Limit.....	267
7.4. Synchronization Motion Instruction.....	268
CONNECT -- Synchronization Motion	268

CONNPATH -- Synchronization Motion 2	269
CAM -- Cam Table Motion.....	270
CAMBOX -- Following Cam Table Motion	274
MOVELINK -- Auto Cam	277
MOVESLINK -- Auto Cam 2	283
MOVELINK_MODIFY -- Sync Distance Modification.....	286
MOVESYNC -- Synchronous Motion	290
FLEXLINK -- Excite Motion.....	296
7.5. Motion Settings Instructions	299
CLUTCH_RATE -- Link Speed.....	299
ENCODER_RATIO -- Encoder Gear Ratio.....	301
STEP_RATIO -- Motor Gear Ratio.....	302
BACKLASH -- Backlash Compensation.....	302
PITCHSET -- Pitch Compensation	303
PITCH2SET -- 2D Pitch Compensation	308
PITCH_DIST -- Pitch Compensation Distance.....	310
PITCH2_DIST -- 2D Pitch Compensation Distance.....	310
7.6. Robot Instructions	311
CONNFRAME -- Robot Inverse Kinematics	311
CONNREFRAME -- Robot Forward Kinematics.....	313
FRAME -- Robotic Arm Type.....	314
FRAME_STATUS -- Robot Axis Status	314
FRAME_TRANS2 -- Coordinates Transform (FWD / REV).....	315
FRAME_ROTATE -- Workpiece Coordinates Transform	316
FRAME_ROTATE2 -- Coordinates Transform Calculation	318
WORLD_DPOS -- World Coordinate System.....	321
MOVER_L/MOVER_LABS -- Joint Linear Interpolation.....	321
MOVER_C/MOVER_CABS -- Joint Axis Plane Arc	322
MOVER_C3/MOVER_C3ABS -- Joint Axis Space Arc.....	323
FRAME_CAL -- Parameters Correction.....	324
MOVER2_J/MOVER2_JABS -- Robot Point Motion	325
MOVER2_P/MOVER2_PABS -- Robot Point Motion	326
MOVER2_L/MOVER2_LABS -- Robot Linear Motion.....	327
MOVER2_ARCHP/MOVER2_ARCHPABS -- Arch Jog.....	330
MOVER2_ARCHPL/MOVER2_ARCHPLABS -- Line Arch.....	331
MOVER2_ARCHP3 / MOVER2_ARCHP3ABS -- 3D Arch	333
MOVER2_ARC / MOVER2_ARCABS -- Arc Interpolation.....	334
MOVER2_LIMIT -- Joint SPEED & ACCEL Limit Settings	335
ZSMOOTH -- Robot Smooth Coefficient.....	336
MOVER2_ZSMOFFSET -- Smooth Offset Time of Continuous Interpolation	336
FORCE_SPEED -- Robot Motion Speed Settings.....	337
MOVER2 PTP & Path Motion SPEED & ACCEL Config.....	338
Chapter 8 Program Structure & Instruction	342
8.1. Program Symbols.....	342

' -- Add Comments	342
_ -- Change Line	342
: -- Label.....	342
8.2. Data Definitions	342
CONST -- Constant Definitions.....	342
DIM -- Variables Definitions.....	343
LOCAL -- Local Definitions.....	344
GLOBAL -- Global Definitions	344
8.3. Array Operations.....	345
DMINS -- Array Link List Insert	345
DMADD -- Array Increase (in Bulk)	346
DMDEL -- Array Link List Delete.....	346
DMCPY -- Array Copy	347
DMSET -- Array Assign.....	347
DMCMP -- Array Comparison.....	348
DMSEARCH -- Array Search.....	348
SIZEOFARRAY -- Array Space Get	349
8.4. Self-Defined Sub Function.....	350
SUB -- Self-Defined Subfunction SUB	350
SUB_PARA -- SUB Parameters Transferring	352
SUB_IFPARA -- SUB Input Parameters Judge.....	352
GOSUB/CALL -- SUB Call.....	353
GSUB -- Self-Defined Subfunction in G Code.....	353
GSUB_PARA -- GSUB Input Parameters.....	354
GSUB_IFPARA -- GSUB Input Parameters Judge.....	354
END SUB -- Self-Defined Function End.....	355
RETURN -- Function Value Return	355
XSUB -- Customized XSUB Sub-Function	355
RSUB -- Self-Defined RSUB Sub-Function.....	356
8.5. Structural Definitions	357
STRUCTURE -- Structural Body Definitions.....	357
UNION -- Union Definitions	358
8.6. Jump Instructions	360
GOTO -- Forced Jump	360
ON GOSUB -- Conditional Jump	360
ON GOTO -- Conditional Jump 2.....	361
8.7. Conditional Judgement Instructions.....	361
IF -- Condition Judgement Structure.....	361
THEN -- Condition Judgement Structure.....	362
ENDIF -- Condition Judgement Structure	362
ELSEIF -- Condition Judgement Structure	362
8.8. Cycle Instructions.....	362
FOR -- “for” Cycle.....	362
TO -- “for” Cycle Structure.....	363

STEP -- “for” Cycle Structure.....	363
NEXT -- “for” Cycle Structure	363
WHILE -- “while” Cycle Structure.....	363
WEND -- While Cycle.....	364
EXIT -- Exit Cycle	364
REPEAT -- Conditional Cycle	364
UNTIL -- Condition Structure.....	365
8.9. Waiting (Execution) Instructions.....	365
DELAY -- Time Delay	365
WAIT UNTIL -- Time for Condition Met	365
WAIT IDLE -- Wait Until Axes Stop	365
WAIT LOADED -- Wait Until Axes Buffer Clears.....	366
8.10. ZINDEX Pointer Instructions.....	368
ZINDEX_LABEL -- Pointer Index Build	368
ZINDEX_CALL -- SUB Function Access	368
ZINDEX_ARRAY -- Array Access.....	368
ZINDEX_VAR -- Variables Access	369
ZINDEX_STRUCT -- Structure Access	369
ZINDEX_MARK -- Pointer Mark No. Settings	370
Chapter 9 Task Instructions.....	372
9.1. Task Start & Stop	372
RUN -- File Task Start	372
RUNTASK -- SUB TASK Start	372
END -- Task End.....	373
STOP -- File Task Stop.....	373
STOPTASK -- SUB Task Stop.....	374
HALT -- All Tasks Stop.....	374
PAUSE -- All Tasks Pause.....	374
PAUSETASK -- Assigned Tasks Pause.....	374
RESUMETASK -- Assigned Task Resume	375
9.2. File 3 (Z3P) Tasks Instructions	375
FILE3_RUN -- FILE3 Task Execution	375
FILE3_ONRUN -- FILE3 Callback Function.....	375
FILE3_GOTO -- FILE3 Jump (Forced).....	375
FILE3_LINE -- FILE3 Line No.	375
FILE3_LINES -- FILE3 Total Lines	375
FILE3_CHANNEL -- FILE3 File No. / Channel No.	375
FILE3_BUFSIZE -- Buffer Loading Size Settings	375
FILE3_SIZE -- FILE3 Size.....	375
FILE3_DATETIME -- FILE3 Last Modified Date	375
FILE3_BUFLOAD -- FILE3 Loading	375
FILE3_BUFRUN -- Loaded FILE3 Run	375
FILE3_BUF_CLOSE -- FILE3 Close.....	375
FILE3_BUF_SAVE -- Save / Save as	375

FILE3_BUFREAD -- Line Read.....	375
FILE3_BUFLAYER -- Line Layer Read	375
FILE3_BUFELENUM -- Line Total Elements Read.....	375
FILE3_BUFELENAME -- Line Element Name Read.....	375
FILE3_BUFELEPARA -- Line Element Parameters Read	375
FILE3_BUFELEPARA2 -- Element Parameters Read	375
FILE3_BUFELEPALL -- Element All Parameters Read	375
FILE3_BUFELEPALL2 -- Element All Parameters Read	375
FILE3_BUFWRITE -- Row Write.....	375
FILE3_BUFINSERT -- Row Insert.....	375
FILE3_BUFDELN -- Selected Part Delete	375
FILE3_BUFNOTE -- Selected Part Noting	375
FILE3_BUFCOPYN -- N Rows Copy.....	375
FILE3_BUFCUTN -- N Rows Cut	375
FILE3_BUFPASTE -- Row Content Paste	375
FILE3_BUFUNDO -- Last Step Undo.....	375
FILE3_BUFREDO -- Last Step Redo.....	375
FILE3_BUFSEEK -- Content Find.....	375
FILE3_BUFREPLACEALL -- All Replace.....	375
9.3. Task Parameters Instructions.....	375
BASE_MOVE -- Main Axis Specify	375
PROC_STATUS -- Task Status	375
PROC -- Task No.	375
PROCNUMBER -- Current Task No.	375
PROC_LINE -- Task Line No.....	375
PROC_PROGRESS -- Task Instruction Progress	375
PROC_PRIORITY -- Task Priority.....	375
ERROR_LINE -- Error Line No.	375
RUN_ERROR -- Task Error Code.....	375
TICKS -- Task Count Period.....	375
TIME_TICKUS -- Task Timer	375
Chapter 10 Operator & Math Instructions	376
10.1. Arithmetic Operational Instructions.....	376
+ -- Plus.....	376
- -- Minus.....	376
* -- Multiply	376
/ -- Divide.....	376
\ -- Exact Divide.....	376
<< -- Shift Left.....	376
>> -- Shift Right.....	376
MOD -- Remainder Operation	376
ABS -- Absolute Operation	376
10.2. Comparison Operational Instructions.....	376
= -- Comparison or Assignment	376

<> -- Not Equal	376
> -- More Than	376
>= -- More Than / Equal To	376
< -- Less Than	376
<= -- Less Than / Equal To	376
10.3. Logical Operational Instructions	376
AND -- Bit Operation: AND	376
OR -- Bit Operation: OR	376
NOT -- Bit Operation: NOT	376
XOR -- Bit Operation: XOR	376
EQV -- Bit Operation: EQV	376
10.4. Trigonometry Instructions	377
SIN -- Trigonometric Function: SINE	377
ASIN -- Trigonometric Function: Anti-SINE	377
COS -- Trigonometric Function: Cosine	377
ACOS -- Trigonometric Function: Anti-Cosine	377
TAN -- Trigonometric Function: Tangent	377
ATAN -- Trigonometric Function: Anti-Tangent	377
ATAN2 -- Trigonometric Function: Anti-Tangent 2	377
10.5. Exponentiation Operational Instructions	377
EXP -- Exponent	377
SQR -- Square Root	377
LN -- Natural Logarithm	377
LOG -- Logarithm of 10	377
10.6. Data Operational Instructions	377
SET_BIT -- Bit Settings	377
CLEAR_BIT -- Bit 0 Reset	377
READ_BIT -- Bit Read	377
READ_BIT2 -- Read Bit 2	377
FRAC -- Decimal (Returned)	377
INT -- Integer (Returned)	377
SGN -- (Returned)	377
IEEE_IN -- Float Number Combine	377
IEEE_OUT -- Single Byte Extract	377
\$ -- Hexadecimal System	377
10.7. Character String Operational Instructions	377
CHR -- ASCII Code Print	377
HEX -- Hexadecimal System Print	377
STRLEN -- Character String Length (Returned)	377
TOSTR -- Format Output	377
STRCOMP -- Character String Comparison	377
STRFIND -- Character String Search	377
STRCONV -- Encoder Conversion	377
VAL -- String to Number	377

10.8.	Constant Instructions.....	377
	PI -- Circular Constant	377
	TRUE -- True Value	377
	FALSE -- False Value.....	377
	ON -- Open.....	377
	OFF -- Close.....	377
10.9.	Advanced Operational Instructions.....	377
	CRC16 -- CRC Verification Calculation.....	377
	DTSMOOTH -- Table Smooth.....	377
	B_SPLINE -- B Spline Smooth.....	377
	TURN_POSMAKE -- Rotary Coordinates Calculation.....	377
	ZCUSTOM -- Motion Parameters Calculation	377
	ZMATH64 -- 64-Bit Calculation.....	377
	MODBUS_DOUBLE -- Read MODBUS.....	377
Chapter 11	Axis Parameter & Axis Status Instructions.....	378
11.1.	Axis Selection	378
	BASE -- Axis Selection/Axis Group Selection	378
	AXIS -- Temporary Axis Selection	379
11.2.	Basic Parameter Instructions.....	380
	UNITS -- Pulse Amount.....	380
	ATYPE -- Axis Type	381
	AXIS_ADDRESS -- Axis Address Settings.....	384
	AXIS_ENABLE -- Axis Enable.....	387
11.3.	Speed Parameters Instructions	387
	SPEED -- Motion Speed	387
	ACCEL -- Axis Acceleration.....	388
	DECEL -- Axis Deceleration.....	390
	CREEP -- Creep Speed	391
	LSPEED -- Initial Speed	392
	FORCE_SPEED -- SP Speed.....	393
	STARTMOVE_SPEED -- SP Starting Speed	394
	ENDMOVE_SPEED -- SP End Speed	395
	FASTDEC -- Fast Deceleration.....	397
	MSPEED -- Actual Speed Feedback.....	398
	SPEED_RATIO -- Speed Proportion	398
	SRAMP -- Acceleration & Deceleration Curve Settings.....	399
	VP_MODE -- Acceleration & Deceleration Curve Types.....	400
	VP_SPEED -- Current Motion Speed	403
	INTERP_FACTOR -- Interpolation Speed Calculation	404
	CORNER_ACCEL -- Corner Acceleration.....	406
	JERK -- Jerk.....	406
11.4.	Axis Status Checking Instructions	408
	MTYPE -- Current Motion Type.....	408
	NTYPE -- Next Motion Type.....	410

AXISSTATUS -- Axis Status	410
IDLE -- Motion Status	412
ADDAX_AXIS -- Axis No. Add	412
AXIS_STOPREASON -- Axes Stop Reason	412
LINK_AXIS -- Link Axis No.	413
11.5. Motion Look-Ahead Instructions	413
CORNER_MODE -- Corner Speed Setting	413
DECEL_ANGLE -- Corner Deceleration Start	419
STOP_ANGLE -- Corner Deceleration Stop	420
FULL_SP_RADIUS -- Speed Limit Radius	421
SPLIMIT_RADIUS -- Speed Limit Value	422
ZSMOOTH -- Chamfer Radius	422
MERGE -- Continuous Interpolation	423
ZSMOOTH_MODE -- Speed Curve Smoothing Mode	424
ZSMOOTH_START -- Smoothing Mode (ON)	425
ZSMOOTH_END -- Smoothing Mode (OFF)	425
11.6. Motion Buffer Instructions	425
LOADED -- Empty Buffer	425
MOVES_BUFFERED -- Current Buffer Numbers	425
REMAIN_BUFFER -- Remain Buffers	426
MOVE_MARK -- Motion Mark	426
MOVE_CURMARK -- Motion Mark No.	427
LIMIT_BUFFERED -- Motion Buffers Limit	427
11.7. Position Instructions	428
DPOS -- Axis Instruction Position	428
MPOS -- Encoder Feedback Position	428
DEFPOS -- Position Offset	428
OFFPOS -- Offset Position	429
ENDMOVE -- Current Motion Target Position	430
VECTOR_MOVED -- Current Motion Distance	430
REMAIN -- Current Motion Remain Distance	431
VECTOR_BUFFERED -- Remain Distance in Buffer	432
VECTOR_BUFFERED2 -- Target Vector Position	432
ENDMOVE_BUFFER -- End Position in Buffer	433
11.8. Origin Homing Instructions	435
DATUM_IN -- Origin Input Mapping	435
HOMEWAIT -- Homing Reverse-Find Delay	435
11.9. JOG Motion Instructions	436
FAST_JOG -- Jog Input Mapping	436
FWD_JOG -- Positive JOG Input Mapping	438
REV_JOG -- Negative JOG Input Mapping	439
JOGSPEED -- JOG Speed	439
FHOLD_IN -- Hold Input Mapping	440
FHSPEED -- Hold Speed	441

11.10.	“Encoder” Instructions	441
	ENCODER -- Encoder Original Value.....	441
	ENCODER_STATUS -- Encoder Status.....	442
	ENCODER_FILTER -- Encoder Filter.....	442
	PP_STEP -- Encoder Internal Proportion.....	442
	ENCODER_BITS -- Encoder Absolute Value	443
	ENCODER_ID -- Data Dictionary Storage No.	443
	DRIVE_POSMIN -- Encoder Original Min Value.....	444
	DRIVE_POSMAX -- Encoder Original Max Value	444
11.11.	Latch Instructions.....	445
	REGIST -- Position Latch.....	445
	REG_INPUTS -- Latch Input Mapping	451
	REGIST_DELAY -- Latch Delay	451
	MARK -- Latch Trigger	452
	MARKB -- Latch 2 Trigger.....	452
	MARKC -- Latch 3 Trigger.....	453
	MARKD -- Latch 4 Trigger	453
	MARKE -- Latch 5 Trigger.....	453
	MARKF -- Latch 6 Trigger.....	454
	MARKG -- Latch 7 Trigger	454
	MARKH -- Latch 8 Trigger	454
	OPEN_WIN -- Latch Coordinate Range (ON)	455
	CLOSE_WIN -- Latch Coordinate Range (OFF).....	455
	REG_POS -- Position Latch.....	455
	REG_POSB -- Position Latch 2.....	455
	REG_POSC -- Position Latch 3	456
	REG_POSD -- Position Latch 4.....	456
	REG_POSE -- Position Latch 5	457
	REG_POSF -- Position Latch 6	457
	REG_POSG -- Position Latch 7.....	458
	REG_POSH -- Position Latch 8.....	458
11.12.	Position Limit Parameter Instructions.....	459
	FS_LIMIT -- Soft Positive Position Limit	459
	RS_LIMIT -- Soft Negative Position Limit	459
	FWD_IN -- Positive Limit Input Mapping	460
	REV_IN -- Negative Limit Input Mapping.....	461
	ALM_IN -- Alarm Input.....	461
11.13.	On-Position Parameter Instructions	461
	IN_POS -- On Position Mark.....	461
	AXISINP_IN -- On-position Signal Input	462
	IN_POS_DIST -- On-position Distance.....	463
	IN_POS_SPEED -- On-position Speed.....	464
11.14.	Range Limit Parameter Instructions.....	464
	REP_OPTION -- Coordinate Cycle Mode.....	464

REP_DIST -- Coordinate Cycle Position	466
FE -- Current Follow-up Error	466
FE_RANGE -- Follow-up Error when Alarm	467
FE_LIMIT -- Maximum Follow-Up Error	467
11.15. Advanced Setting Instructions	467
INVERT_STEP -- Pulse Mode Settings	467
MAX_SPEED -- Pulse Frequency Limit	469
AXIS_ZSET -- Precision OP	470
AXIS_MODE -- “connect” Motion Hold	471
MOVEOP_DELAY -- Output Delay in Buffer	474
MOVEOP_ADIST -- Close Glue (in Advance)	475
MOVEOP_AHEANDMS -- Time of Early Closing Glue	476
DAC -- Analog Control of Field Bus Axes	477
ERRORMASK -- Operation when Error	480
ZSCAN_CORRECT -- Galvanometer Correction	480
11.16. PID Instructions	481
D_GAIN -- Differential Gain	481
I_GAIN -- Integral Gain	482
OV_GAIN -- Speed Gain	482
P_GAIN -- Proportion Gain	482
VFF_GAIN -- Feedforward Gain	483
AFF_GAIN -- Acceleration Feedforward Gain	483
SERVO -- Closed-Loop Switch	483
Chapter 12 IO Instructions	486
12.1. Input Instructions	486
IN -- Digital Inputs	486
AIN -- Analog Input	486
ZSIMU_IN -- Digital Input Simulation	487
ZSIMU_AIN -- Analog Input Simulation	487
ZSIMU_ENCODER -- Encoder Input Simulation	487
INVERT_IN -- Input Invert	488
IN_SCAN -- Inputs Changes Scan	488
IN_EVENT -- Input Changes Read	489
SCAN_EVENT -- Changes Check	489
IN_BUFF -- Buffer Input Read	490
INFILTER -- Input Filter	490
IN_MSFILTER -- IN Filter Settings	490
LOCALIO_OFFSET -- Local IO Offset	491
12.2. Output Instructions	491
OP -- Output Ports	491
AOUT -- Analog Output	492
READ_OP -- Output Read	493
EXIO_DIR -- EXIO Interface Configuration	493
12.3. Position Comparison Output Instructions	493

PSWITCH -- Position Comparison Output (Soft).....	493
HW_PSWITCH -- Position Comparison Output (Hard).....	496
HW_TIMER -- Hardware Timing.....	502
HW_PSWITCH2 -- Bus Position Comparison Output (Hard).....	507
HW_MINTIME -- HW Min Time Space.....	523
HW_PS2AXISNUM -- PS2 Axis No. Settings.....	524
HW_PS2COUNTS -- PS Comparison Numbers.....	526
12.4. PWM Control Instructions.....	527
PWM_FREQ -- PWM Frequency.....	527
PWM_DUTY -- PWM Duty Cycle.....	527
12.5. Buzzer Control Commands.....	528
SPEAKOUT -- Buzzer Control.....	528
Chapter 13 Communication Instructions.....	529
13.1. Serial Communication Instructions.....	529
SETCOM -- Serial Port Configuration.....	529
ADDRESS -- Controller Station No.	532
COM_UNUSED -- Serial Port Settings.....	532
13.2. CAN Communication Instructions.....	532
CAN -- CAN Communication.....	532
CANIO_ADDRESS -- CAN Communication Settings.....	535
CANIO_ENABLE -- CAN Enable.....	536
CANIO_STATUS -- ZIO Expansion Status.....	536
CANIO_INFO -- CAN Expansion Information.....	537
13.3. Self-Defined Communication Instructions.....	538
GET# -- String Read.....	538
OPEN# -- Self-Defined Ethernet Communication ON.....	539
CLOSE # -- Self-Defined Ethernet Communication OFF.....	540
PRINT# -- Character String Output.....	541
PUTCHAR# -- Character Output.....	542
PORT_TARGET -- IP & Port No. Configuration.....	543
13.4. Print & Output Instructions.....	544
PRINT -- Information Print.....	544
ERRSWITCH -- Information Output Settings.....	545
TRACE -- Information Print 2.....	545
WARN -- Alarm Information.....	546
ERROR -- Error Information.....	546
13.5. Channel Parameter Instructions.....	546
PORT -- Channel No.....	546
PORT_STATUS -- Channel Status.....	548
PORT_MODE -- Channel Mode.....	549
FILE_PORT -- Current Channel File No.	550
PROTOCOL -- Channel Communication Protocol.....	550
ETH_MODE -- EtherNET Port Mode Settings.....	550
SEND_AUTOUP -- Active Report.....	551

SEND_AUTOUP2 -- Active Report 2	551
IFAUTOUP_PORT -- Active Reporting Port.....	552
PORT_KEEPLIVE -- Timeout Detection.....	552
PORT_485DELAY -- Delay Time of Receive & Send	552
13.6. MODBUS Communication Instructions	553
MODBUS_BIT -- Bit Register	553
MODBUS_IEEE -- Word Register (32-Bit Floating)	553
MODBUS_LONG -- Word Register (32-Bit Integer).....	554
MODBUS_REG -- Word Register (16-Bit Integer).....	554
MODBUS_STRING -- Word Register (Byte)	555
MODBUSM_DES -- Modbus Communication Connection.....	555
MODBUSM_DES2 -- Ethernet Communication.....	557
MODBUSM_STATE -- MODBUS Communication Status	559
MODBUSM_REGSET -- Slave Modbus Value Write.....	559
MODBUSM_REGGET -- Slave Modbus Value Read.....	560
MODBUSM_3XGET -- Input Register Read	561
MODBUSM_BITSET -- Slave Coil Write	561
MODBUSM_BITGET -- Slave Coil Read	562
MODBUSM_1XGET -- Isolated Inputs Read	562
13.7. Direct Command Instructions between Controllers	562
SEND_RESULT -- “send” Result Read.....	562
SEND_CMD -- “send” Command.....	563
SEND_CMDAXIS -- “send” Command (with Axis No.).....	563
SEND_ASSIGN -- “send” Command (with Value).....	564
SEND_QUERY -- “send” Command (to TABLE).....	564
SEND_QUERTSET -- “send” Command (to TABLE).....	565
13.8. Sending Instructions between Controller Files	565
SEND_ZAR -- USB Drive operation.....	565
SEND_FALSH -- Data Copy	566
SEND_FILE -- USB Drive Data Copy	566
SEND_IFLASH -- flash Data Copy.....	567
SEND_PERCENT -- Instruction Process Check	567
SEND_CONTROL -- Controller Type Check.....	567
Chapter 14 System Instructions	569
14.1. Controller Encryption Instructions.....	569
APP_PASS -- Password	569
LOCK -- Controller Lock.....	569
UNLOCK -- Controller Unlock	570
14.2. System Time Instructions.....	570
DATE -- System Date.....	570
DATE\$ -- System Date 2.....	570
DAY -- System Week	571
DAY\$ -- System Week 2	571
RTC_DATE -- System Date 3	571

TIME -- System Time	572
TIME\$ -- System Time 2	572
RTC_TIME -- System Time 3	573
14.3. Axis System Parameter Instructions	573
WDOG -- Total Axes Enable	573
DISABLE_GROUP -- Axes Group	573
ERROR_AXIS -- Error Axis No.	574
MOTION_ERROR -- Error Axes List	574
ERROR_SET -- Error Output	574
RADIUS_ERRSET -- Circular Interpolation Check	575
14.4. IP Parameter Instructions	576
IP_ADDRESS -- IP Address	576
IP_ADDRESS2 -- IP Address 2	576
IP_GATEWAY -- IP Gateway	577
IP_NETMASK -- IP Mask	577
IP_IFDHCP -- IP Address Auto-Get	578
IP_IFDHCP2 -- IP Address Auto-Get 2	578
14.5. Controller Information Instructions	578
VERSION_FPGA -- System FPGA Version	578
VERSION_BUILD -- System Firmware Original Date	579
VERSION_DATE -- System Firmware Version	579
VERSION -- System Software Version	579
ID_HARDWARE -- Controller Hardware Model	579
CONTROL -- Controller Software Model	580
SYSTEM_ZSET -- Controller Settings	580
LEDOUT -- Controller Indicator Light	581
SERIAL_NUMBER -- Controller Unique ID	581
SERVO_PERIOD -- System Period	582
SYS_ZFEATURE -- System Specification	582
SYS_IOSET -- Special IO Switch	585
LASER_SET -- Energy Parallel Output Switch	585
ZML_DEFSHIFT -- ZML Device "shift" Time	585
14.6. Log Instructions	586
RTLOG_COUNT -- The Number of Current Logs	586
RTLOG_CLEAR -- Current Logs Clear	586
RTLOG_ADD -- Log Error Message Add	586
RTLOG_CODE -- Log Error No. Get	587
RTLOG_TIME\$ -- Log Error Time Get	587
RTLOG_INFO -- Log Error Message Get	588
RTLOG_INFO2 -- Log Error Message Get 2	588
?* RTLOG -- Current Recorded Logs Get	588
14.7. TABLE Array Instructions	589
TABLE -- System Default Array	589
TSIZE -- Table Size	589

TABLESTRING -- Table Print (in Character String).....	590
14.8. Oscilloscope Instructions	591
TRIGGER -- Oscilloscope Trigger	591
SCOPE -- Data Acquisition.....	591
SCOPE_POS -- Captured Points.....	592
14.9. VR Instructions	593
CLEAR -- VR Clear.....	593
VR -- Power Failure Storage (Float).....	593
VR_INT -- Power Failure Storage (Integer)	593
VRSTRING -- Power Failure Storage (String).....	594
Chapter 15 Storage Instructions	595
15.1. U Disk Instructions	595
FILE -- USB File Operations	595
U_STATE -- U Disk Status	599
U_READ -- U Disk Read (float).....	599
U_READDBL -- U Disk Read (double)	600
U_READ2 -- U Disk Read 2.....	600
U_READ2DBL -- U Disk Read 2 (double)	601
U_READDSB -- DSB File Read	602
U_WRITE -- U Disk Write (float)	603
U_WRITEDBL -- U Disk Write (double).....	604
U_WRITE2 -- U Disk Write 2	604
STICK_READ -- U Disk Read (to Table).....	605
STICK_WRITE -- U Disk Write (from Table).....	605
STICK_READVR -- U Disk Read (to VR)	606
DISK_WRITEVR -- U Disk Write (from VR)	607
DISK_CONFIG -- ZHD HMI “U Drive”	607
15.2. FLASH Instructions	608
FLASH_WRITE -- Flash Write (float)	608
FLASH_WRITEDBL -- Flash Write (double).....	609
FLASH_WRITE2 -- Flash Write 2	609
FLASH_READ -- Flash Read.....	610
FLASH_READDBL -- Flash Read (double)	610
FLASH_READ2 -- Flash Read 2.....	611
FLASH_READ2DBL -- Flash Read 2 (double)	612
FLASHVR -- RAM Data Copy.....	613
FLASH_SECTSIZE -- Variable Numbers in Flash.....	613
FLASH_SECTES -- Flash Block Numbers	614
15.3. File Storage Instructions	614
FILE_ZOPEN -- File Open.....	614
FILE_ZCLOSE -- File Close	615
FILE_ZWRITES -- File Write (Character String)	615
FILE_ZWRITE -- File Write (Character)	616
FILE_ZREAD -- File Read (Character).....	616

FILE_ZREADLINE -- File Line Read.....	617
FILE_ZSEEK -- File Location.....	618
FILE_ZSEEKLINE -- File Line Location	619
FILE_ZTELL -- File Read & Write Position	619
FILE_ZTELLLINE -- File Line No.....	620
Chapter 16 Interrupt Instructions	621
16.1. Three Interrupt Instructions.....	621
INT_ENABLE -- Interrupt Main Switch	621
ONPOWEROFF -- Power-Failure Interrupt SUB.....	622
INT_ONn -- External Input Interrupt SUB	623
INT_OFFn -- External Input Interrupt SUB	623
ONTIMERn -- Timer Interrupt SUB.....	624
INT_CYCLE -- Interrupt Period Execute	624
16.2. Timer Instructions	625
TIMER_IFEND -- Timer Status.....	625
TIMER_START -- Timer ON	625
TIMER_COUNT -- Timer Accumulation Time	626
TIMER_STOP -- Timer OFF	626
Chapter 17 Fieldbus Instructions	627
17.1. No. Descriptions.....	627
Slot No.	627
Device No.	627
Drive No.....	627
17.2. Basic Instructions.....	627
SLOT_SCAN -- Bus Scan	627
SLOT_START -- Field Bus ON.....	628
SLOT_STOP -- Field Bus OFF.....	629
SLOT_INFO -- Bus Information Get.....	629
?*SLOT -- Field Bus Port Print.....	630
?*ETHERCAT -- EtherCAT Bus Status Print	630
ZTEST -- EtherCAT Information Check.....	631
?*RTEX -- Rtex Status Print.....	634
?*ZML -- ZML Information Print.....	635
17.3. SDO Operation Instructions.....	635
SDO_WRITE -- Data Dictionary Write	635
SDO_WRITE_AXIS -- Data Dictionary Write (with Axis).....	636
SDO_READ -- Data Dictionary Read	637
SDO_READ_AXIS -- Data Dictionary Read (with Axis)	638
SDO_READ2 -- Data Dictionary Read (to MODBUS).....	639
17.4. Device Instructions.....	640
NODE_COUNT -- Connected Device Numbers	640
NODE_STATUS -- Device Status.....	641
NODE_AXIS_COUNT -- Connected Motor Numbers	641
NODE_IO -- Device IO	642

NODE_AIO -- Device Analog	642
NODE_INFO -- Device Information	643
NODE_PROFILE -- PDO Preset	644
NODE_PDOBUFF -- PDO Setting of Specail Devices	645
NODE_PDO_WRBUFF -- PDO Modify by Offset	645
NODE_PDO_RDBUFF -- PDO Read by Offset	646
NODE_REGWRITE -- ESC Register Writing	646
NODE_REGREAD -- ESC Register Reading	647
NODE_PRESET -- Device Preset	647
ZML_INFO -- Device XML Check	648
17.5. Drive Instructions	649
DRIVE_MODE -- Drive Mode	649
DRIVE_PROFILE -- Drive PDO Settings	650
DRIVE_CW_MODE -- Drive Settings	654
DRIVE_CONTROLWORD -- Drive Control Word	655
DRIVE_STATUS -- Drive Status	656
DRIVE_IO -- Drive IO Settings	657
DRIVE_TORQUE -- Drive Torque	658
DRIVE_TORQUEMAX -- Drive Max Torque Limit	658
DRIVE_TORQUEMIN -- Drive Min Torque Limit	659
DRIVE_FE -- Drive Error	659
DRIVE_FE_LIMIT -- Drive Error Limit	659
DRIVE_CLEAR -- Alarm Clear	660
DRIVE_READ -- Parameters Read	660
DRIVE_WRITE -- Parameters Write	663
Chapter 18 ZHD Teaching Box Parameters	666
18.1. Teaching Box Instructions	666
LCD_CONNECT -- LCD No. Settings	666
IP_CONNECT -- IP Connect	666
IP_ADDRESS -- IP Address	666
IP_GATEWAY -- IP Gateway	667
IP_NETMASK -- IP Mask	667
18.2. Controller Instructions	667
LCD_LEDSTATE -- LED State Settings	667
LCD_WDOGTIME -- Time Setting for Screen Power-Off	668
Chapter 19 MotionRT Instructions	669
19.1. MotionRT Instructions	669
CARD_INFO -- Control Card Info Read & Write	669
?*CARD -- Sub-Card Info Print	670
REG_CARD -- Latch Select	670
SLOT_SLAVE -- EtherCAT Redundancy Settings	672
SLOT_PERIOD -- Multi-ECAT Bus Period Settings	672
Chapter 20 Local Slave Port Instructions	673
ZMIO_CONFIG -- Analog Range & Channel State Set / Get	673

ZMIO_INFO -- ZMIO Itself Expansion Check	674
ZMIO_OFFSET -- ZMIO Extended IO Offset	674
ZMAIO_OFFSET -- ZMIO Extended AIO Offset.....	675
Chapter 21 CAD Controls & Instructions	676
21.1. CAD Controls Description.....	677
21.2. CAD Basic Instructions.....	678
CAD_CHANNEL -- CAD Channel	679
CAD_LOADFILE -- Graphic Load from File	679
CAD_LOADTABLE -- Graphic Load from TABLE.....	680
CAD_LOADBIN -- Graphic Load from BIN File.....	680
CAD_CLOSE -- CAD Graphic Loading OFF	680
CAD_TOCODE -- Export Code File (NC / Z3P).....	681
CAD_TABSIZE -- TABLE Space Occupy	681
CAD_TOTABLE -- TABLE Data Export.....	681
CAD_TOBIN -- BIN File Export.....	682
CAD_TAB2CODE -- Code File (from TABLE Data)	682
CAD_PARA -- Parameters Settings	683
CAD_DISPLAY -- Display Settings	683
CAD_MOUSE -- Mouse Mode Settings.....	684
CAD_AUTOVIEW -- Auto-View	684
CAD_RANGE -- Trajectory Range Get.....	684
21.3. CAD “Custom” Control Extended Instructions	685
CAD_SETCOLOR -- CAD Drawing Color Set.....	685
CAD_DRAW -- CAD Graphic Draw	686
CAD_DRAWTAB -- CAD Graphics Draw (from TABLE).....	686
CAD_TRANSPPOINT -- Convert to Pixel Coordinates.....	686
CAD_TRANSPPOINT2 -- Convert to Graphic Coordinates.....	687
CAD_SELECT -- Graphic Box Select.....	688
21.4. CAD Advanced Instructions	688
CAD_MACHTYPE -- Machine Type.....	688
CAD_OPTIMIZE -- Trajectory Optimize.....	689
CAD_EDIT -- Custom Edit.....	690
CAD_MOVE -- Trajectory Translate.....	690
CAD_SCALE -- Trajectory Scaling.....	691
CAD_ROTATE -- Trajectory Rotate.....	691
CAD_DELETE -- Trajectory Delete.....	691
CAD_ACTION -- Action Data Get / Set	691
CAD_ACTIONADD -- Action Add.....	692
CAD_ACTIONDEL -- Action Data Delete	692
CAD_UNDO -- Last Step Undo	692
CAD_REDO -- Last Step Redo	693
CAD_SORT -- Sort Manual.....	693
21.5. CAD Parameters List	693
General Parameters	693

G Code Parameters.....	695
Basic Parameters.....	695
21.6. TABLE Data Format.....	695
Data Format Designing.....	695
DataType Definition.....	696
21.7. Machine Type Technology.....	699
General 3-Axis Machine Tool.....	699
General 2-Axis Machine Tool.....	699
Special 3-Axis Drilling Machine Tool.....	699
Chapter 22 DT Motion Instructions.....	701
MOVEDTSP / MOVEDTABSSP -- DT Linear Motion.....	701
MOVECIRCDTSP / MOVECIRCDTABSSP -- DT Arc.....	701
MOVECIRC2DTSP / MOVECIRC2DTABSSP -- DT 3-Point Arc.....	702
MSPHERICALDTSP / MSPHERICALDTABSSP -- DT Space Arc.....	703
Chapter 23 Simple Routines.....	704
23.1. Common Operation.....	704
IO Operation.....	704
SP Instruction continuous interpolation.....	704
Conversion between String and Data.....	705
Handwheel.....	706
Fly-Shearing.....	707
Position Comparison Output.....	707
Power Failure Storage.....	707
Robot.....	707
Robotic Arm by MOVESYNC Instruction.....	707
Encoder Read.....	711
Self-Defined G codes.....	714
23.2. Module Communication.....	716
CAN Communication.....	716
HMI Communication.....	718
Self-Defined Ethernet Communication.....	721
Communication between Controllers.....	723
String and Self-Defined Communication.....	723
23.3. Bus Initialization.....	724
EtherCAT Initialization.....	724
Rtex Initialization.....	725
Chapter 24 Errors Debugging.....	726
24.1. Common Problems List.....	726
Problems Checking.....	726
24.2. Solutions.....	729
Debugging: Manual Motion.....	729
Debugging: Breakpoint.....	730
Debugging: Oscilloscope Sampling.....	731
Debugging: Register Reading.....	731

Debugging: Remote Commands Enter.....	732
Debugging: Program Information Print	732
Debugging: Fast-IO Detection	732
Debugging: Axis Parameters Status Reading.....	733
Appendix I Error Code List	734
Appendix II Module Expansion.....	763
ZCAN Expansion Module	764
Expansion Wiring.....	764
Resource Mapping	765
IO Mapping.....	765
Axis Mapping.....	768
EtherCAT Expansion Module	769
Expansion Wiring.....	769
Resource Mapping	770
IO Mapping.....	770
Axis Mapping.....	771
Appendix III HMI Communication	772
Controller & HMI Communication.....	772
Connect Controller with HMI	773
Connect with ZHD Series HMI.....	773
Connect to the Third-Party HMI.....	774
Appendix IV EtherCAT Communication	779
Appendix V RTEX Communication	783

Chapter 1 Motion Control Products Introduction

1.1. Motion Control Products 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 Products Advantages

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 Instruction” 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 RTSys 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 RTSys 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. Motion Controller Main Functions

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 RTSys 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		RTSys 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. Motion Controller Applications

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.

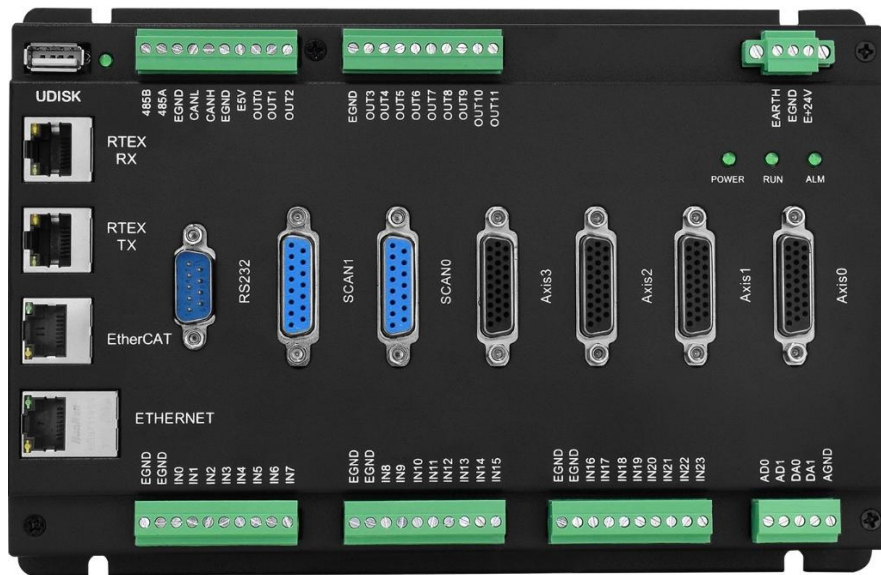
1.5. Motion Controller Interfaces

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. Motion Controller Usages

✧ Prepare Work

Software: install RTSys 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 RTSys 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 RTSys software menu bar "Help" - "RTSys Help" to view the introduction of the various functions of the software, writing tasks and program module for program simulation debugging.

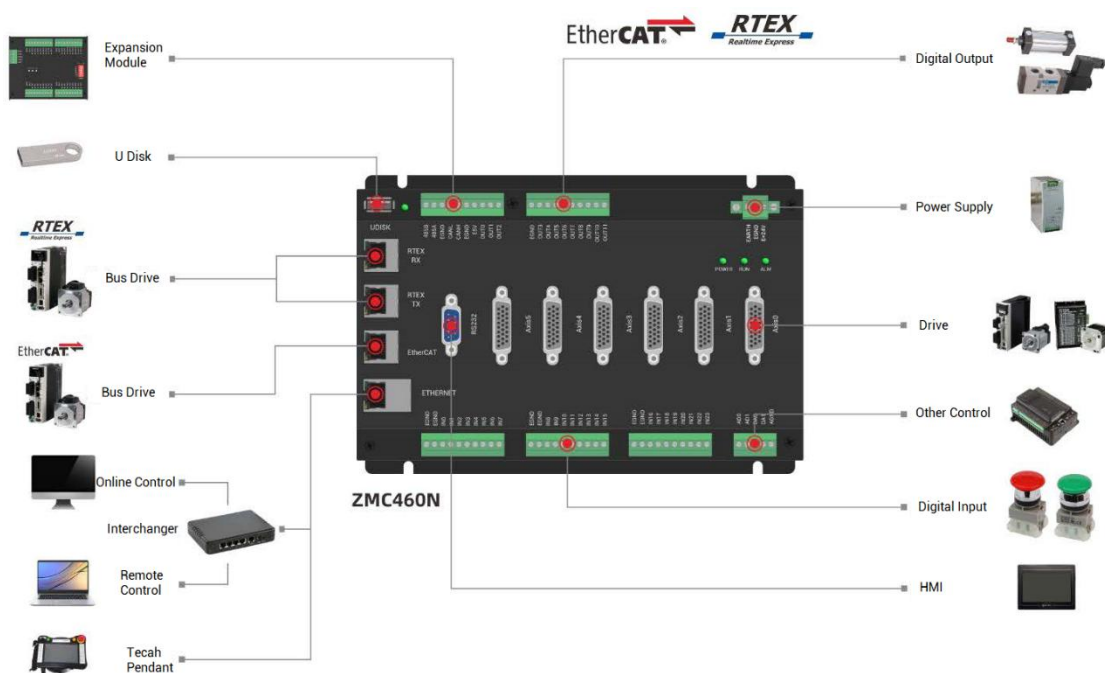
Parameters that need to be set for programming: BASE select the axis number, ATYPE axis type,

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:

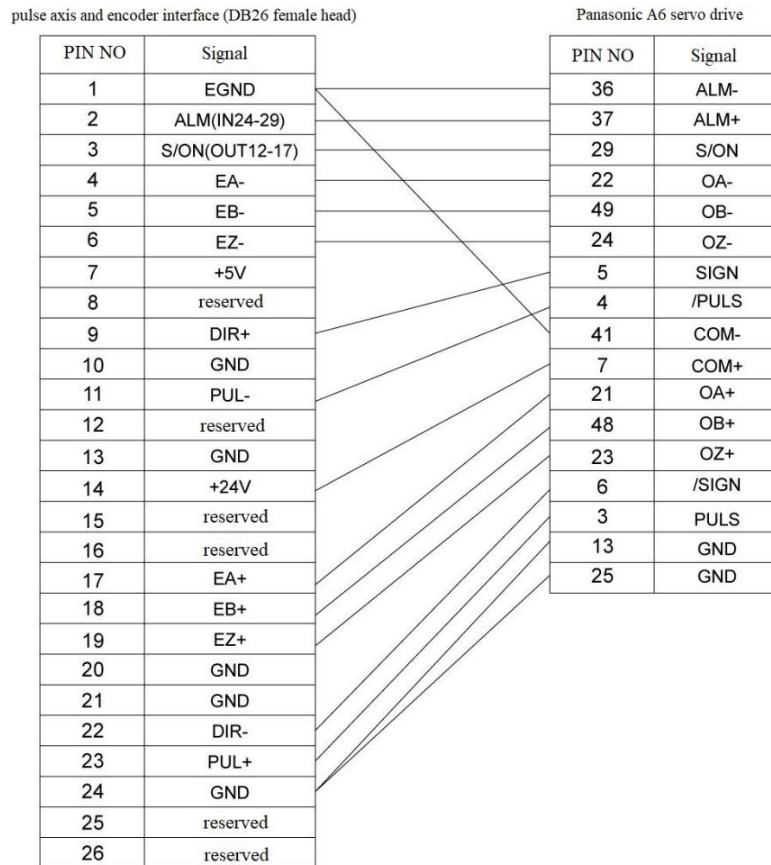


➤ Wiring between computer and controller:

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.



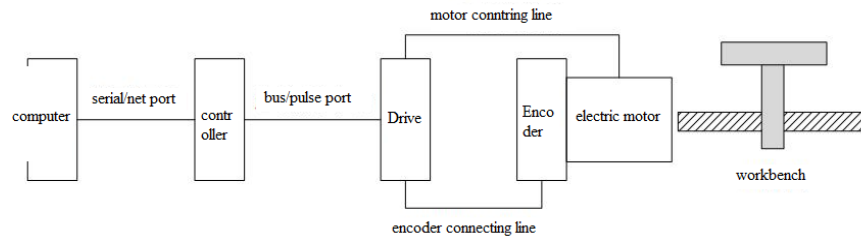
➤ Power wiring:

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 2 RTSys Software Program

2.1. Program Software Introduction

RTSys 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.

RTSys programming software supports RTBasic, RTPlc ladder diagram, RTHmi configuration programming. RTBasic is the Basic programming language used by ZMotion motion controller, and provides all standard program Syntax, 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.

RTBasic 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.

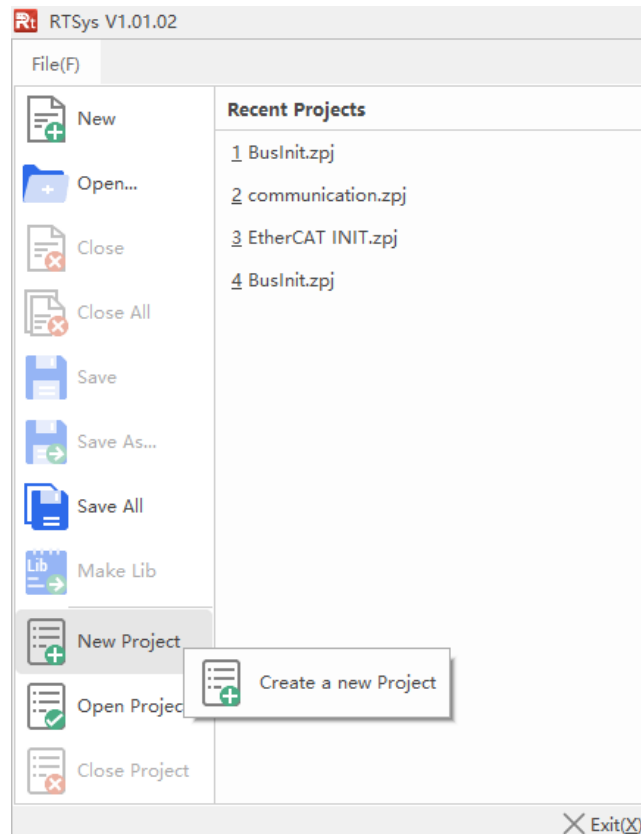
RTBasic has the real-time multi-task property, multi RTBasic procedures can build at the same time and multi-task real-time operation, which makes the complex application simpler.

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.

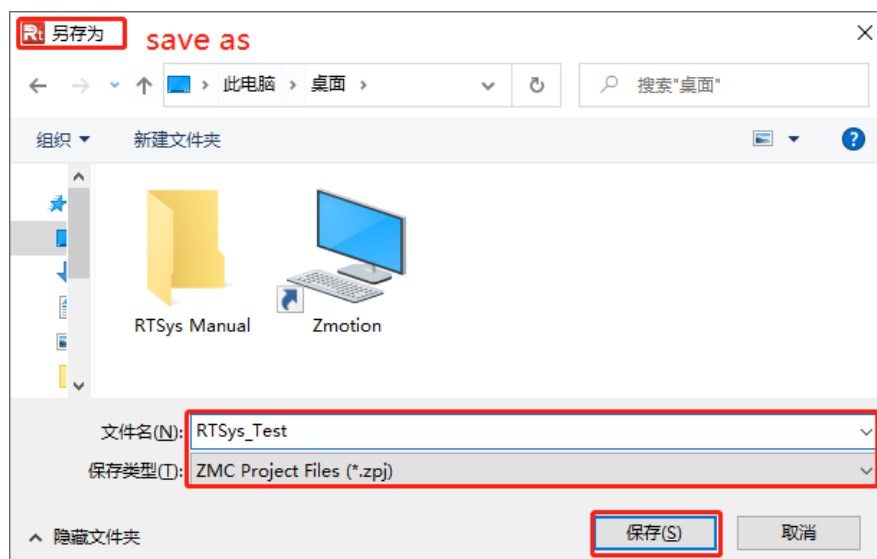
2.2. New Project

Please build a new folder to save the project that is to be built. Open RTSys programming software. Please visit ZMOTION website (www.zmotionglobal.com) to update software version.

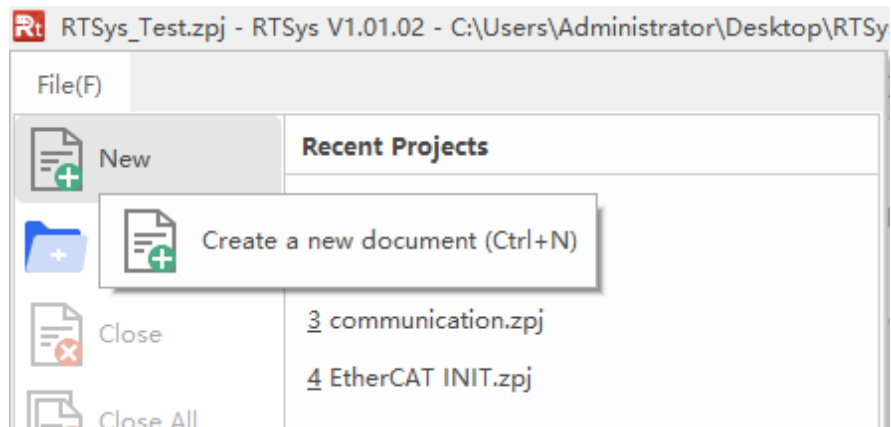
1. New build item: “File” in “Menu” → “New Project”.



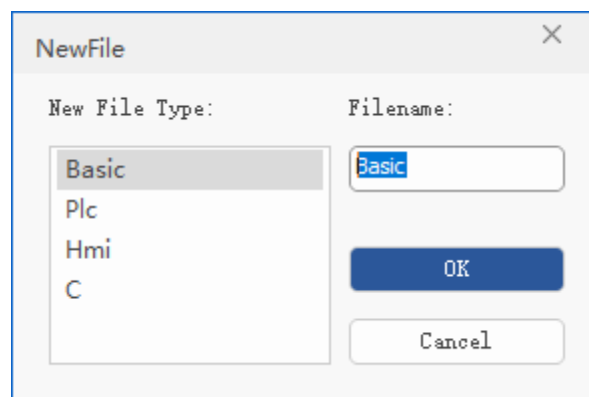
2. Click “New Project”, then “Save as...” will be jumped, select one folder and open it. Input folder’s name and save the project, pay attention to the suffix should be “.zpj”.



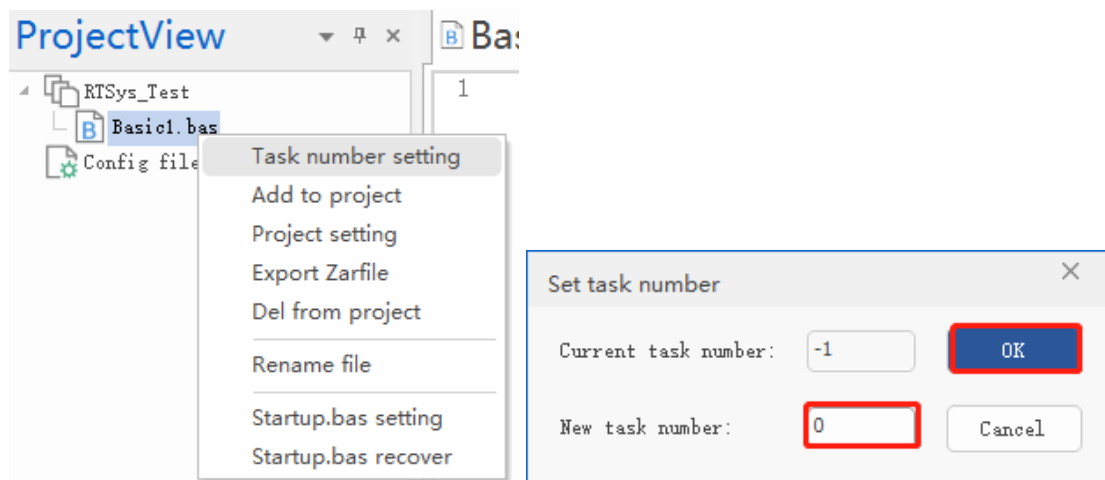
3. New build a file: “File” → “New File”.



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



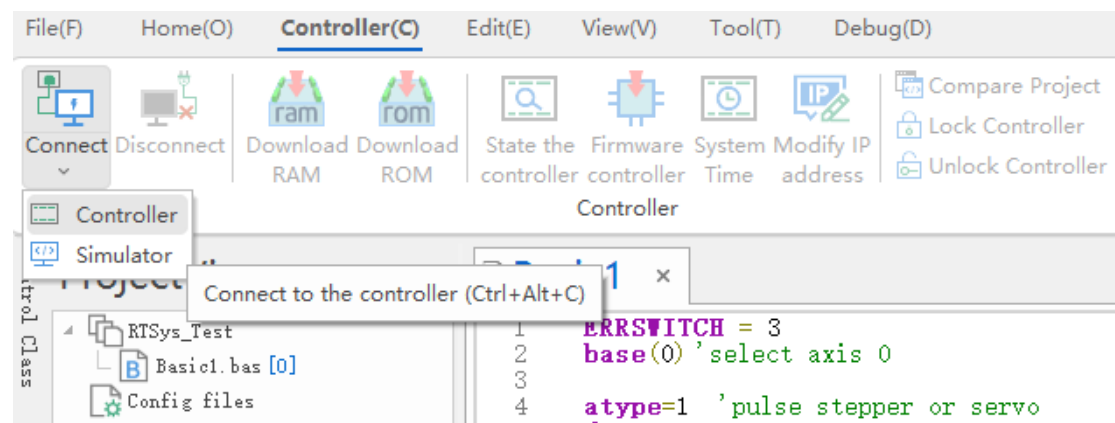
4. Set file as automatic operation: please see the below picture, double click the right position “AutoRun” of “File”, and input task number is “0”.



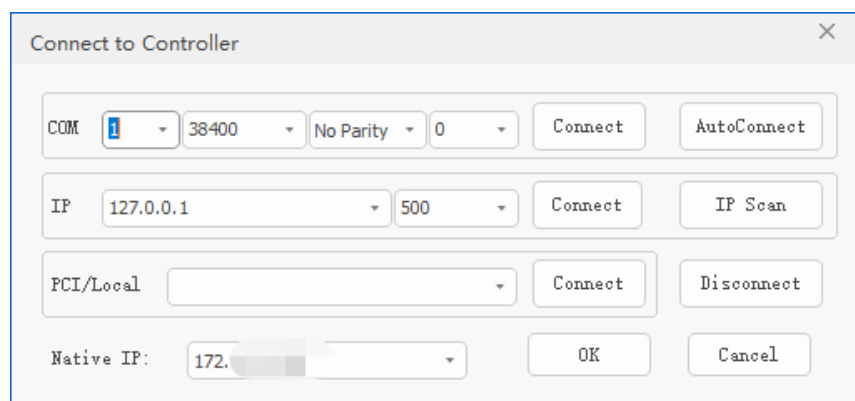
5. Program the procedure: when procedure is programmed, click “save” the file. New built Basic file will be saved automatically into the file in Project zpj.

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.

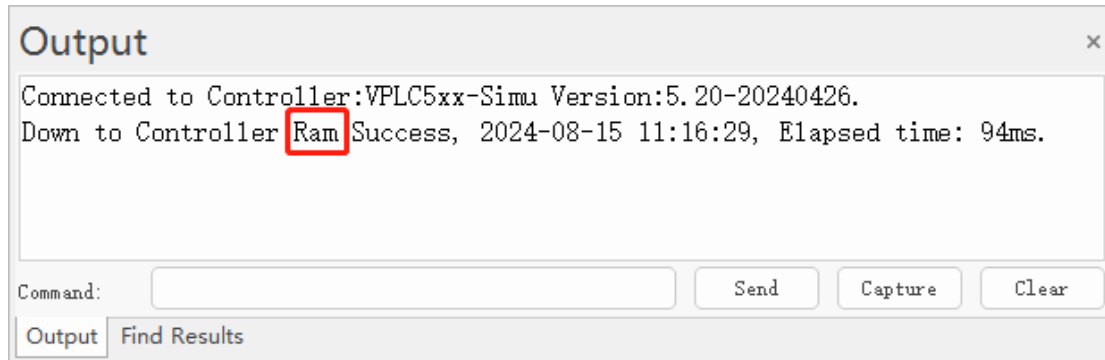


For the detailed method of serial port connection and network port connection, please refer to the "Help" → "RTSys Help" document in the menu bar of RTSys software.

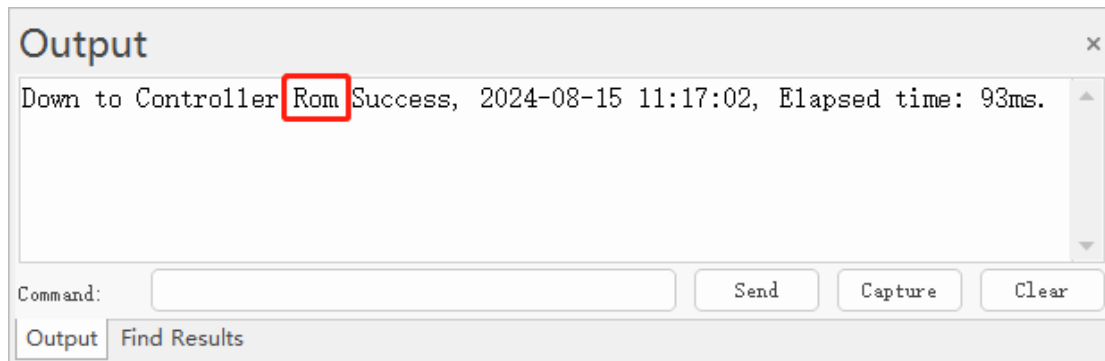
7. Download Program: click “Download RAM” or “Download ROM”. When it is downloaded successfully, Command and Output window will give a hint. Program is downloaded into controller and will run automatically.



- ✓ Download RAM:



- ✓ Download ROM:

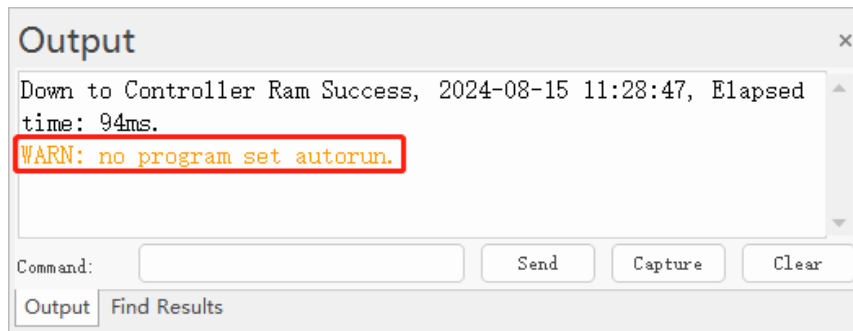


The program will not be saved after the RAM download is powered off, but the program will be saved after the ROM download is powered off. After the program downloaded to the ROM is connected to the controller next time, the program will automatically run according to the task No.

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

➤ What is It?

In “online command & output” window, you can get much information, like, various parameters of the controller, program running results and program error information.

➤ How to Do?

You only need to enter the function commands (?, PRINT, WARN, ERROR, TRACE, etc.) in this window. **And this function becomes valid after connecting to controller or simulator, it is not limited by program running status.**

In addition, English symbols are used for question marks, don't enter Chinese symbols. ERRSWITCH is the control switch of TRACE, WARN, and ERROR commands. Different parameter values correspond to different output effects:

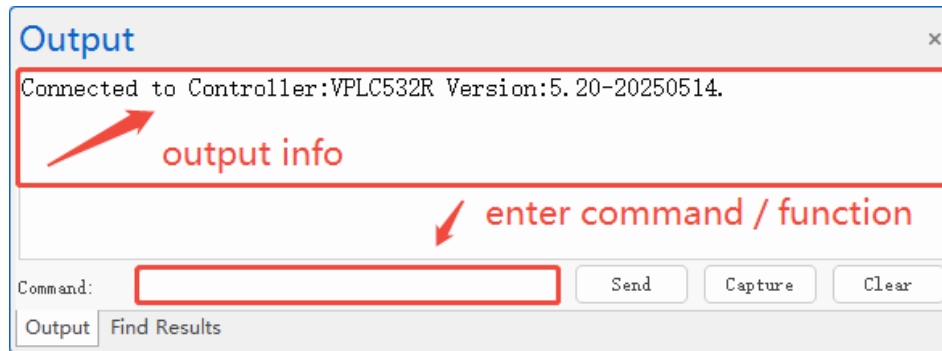
0: TRACK, WARN, ERROR instructions all don't output.

1: only output ERROR instruction.

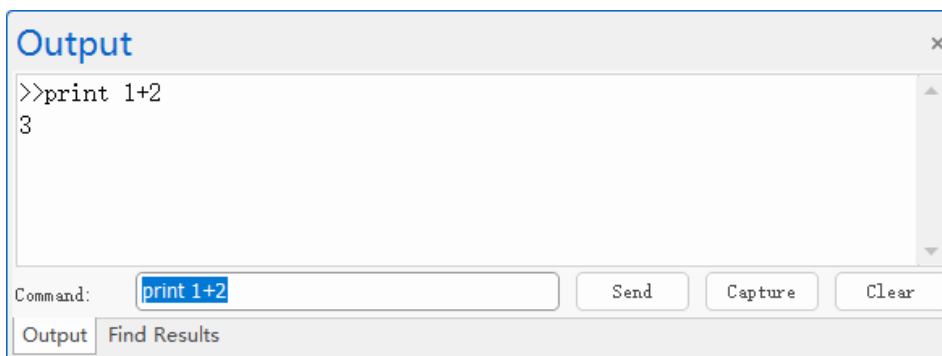
2: output WARN, ERROR instructions.

3: TRACE, WARN, ERROR instructions all output.

The online command and output window is shown below.

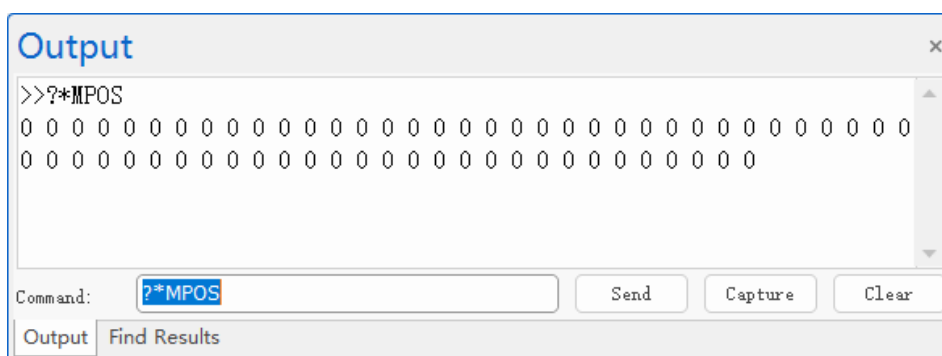


For example, ">>print 1+2", ">>" represents the command input by RTSys online command, send "print 1+2", it will output the calculation result.



--Check Axis States--

Use online command to see status of all axes, please see the below picture. Input "?*mpos", it will print out measurement feedback position (MPOS) of several axes.



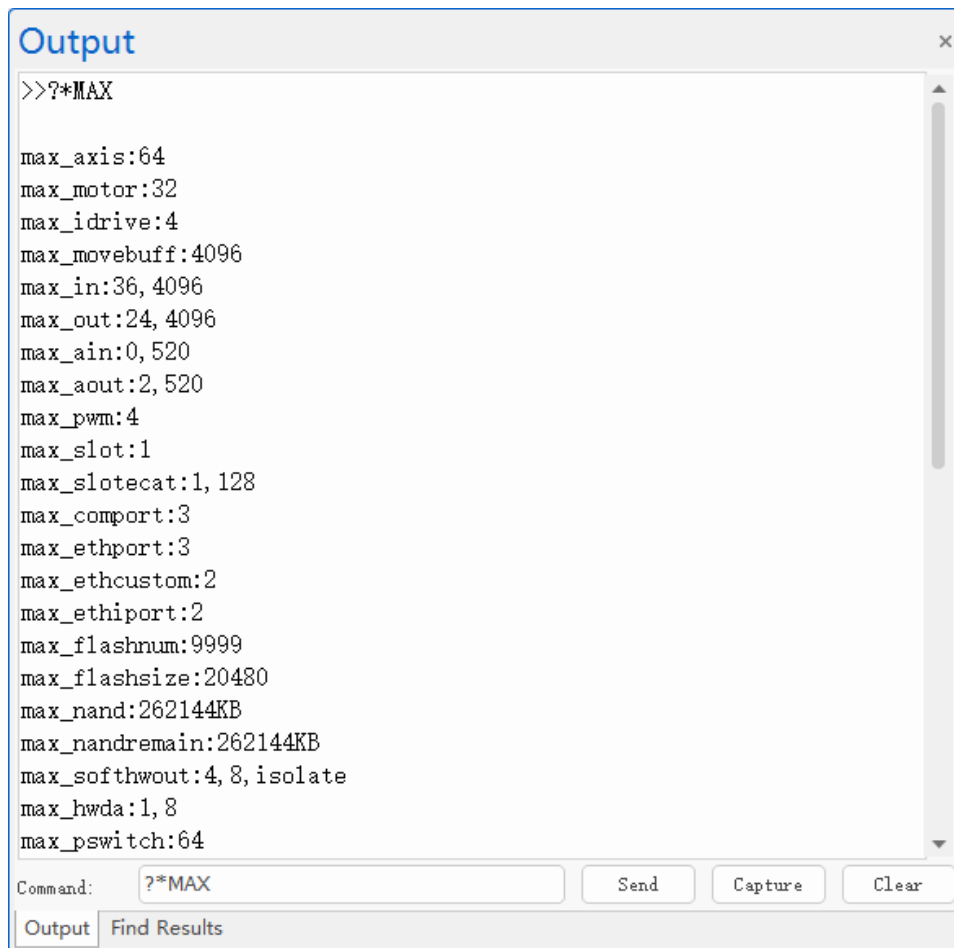
--commonly-used commands--:

*SET: print all parameters' values

*TASK: print task information (when the task is normal, it only prints the task state, when the task appears error, except the task state, there are error task No., specific error line).

*MAX: print all specifications and parameters

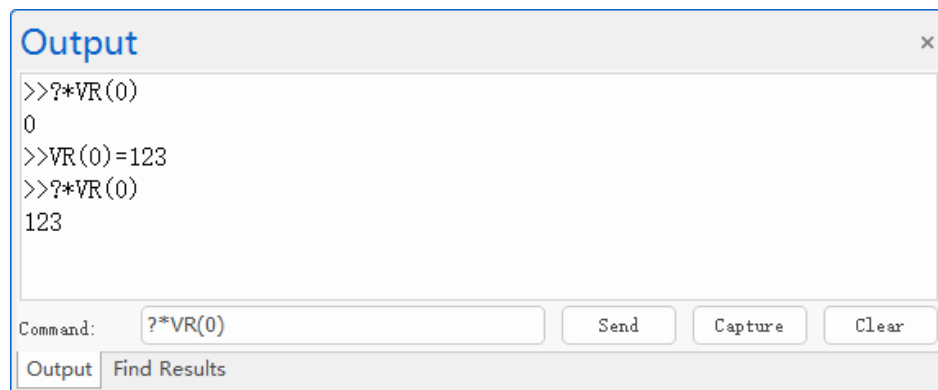
- ?*FILE: print program file information
 - ?*SETCOM: print the present serial port configuration information
 - ?*BASE: print the present task BASE axis list
 - ?*array name: print all elements of array, the array length can't be so long.
 - ?*parameter name: print single parameter of all axes
 - ?*ETHERCAT: print EtherCAT connection setting status
 - ?*RTEX: print RTEX 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
 - ?*CYCLEUP: print whether the cycle report channel is enabled or not.
- After connecting to controller, enter “?*max” to print all specifications and parameters results of controller:



--Modify Variable's Value--

What's more, you also can modify the value of the variable (VR variables, TABLE variables,

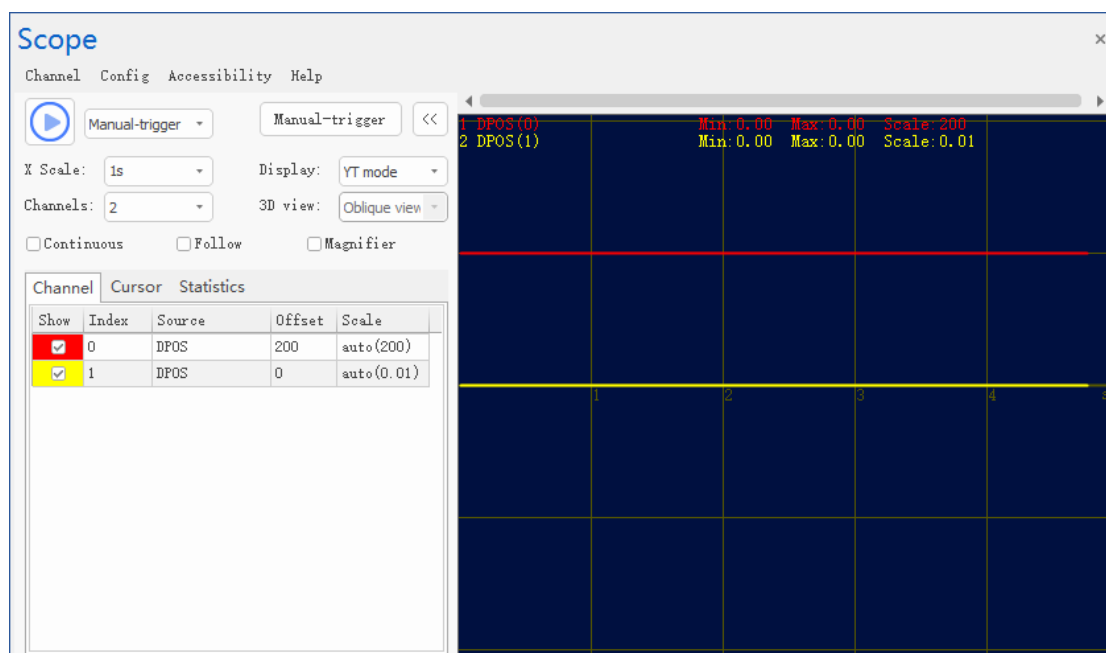
MODBUS variables, global variables, system settings, axis parameters, and axis state variables), for example, modify VR variable value.



2.4. How to Use Oscilloscope

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.	

How to Configure Scope

(1) Scope Config Window

Click menu above "Config" button, then click "parameter configuration".

Parameters config

Basic parameters	
Sampling period(us)	1000
Interval period number	1
Max sampling periods	5000
Auto use end of table	True
Table pos	310000
Export parameters	True
Overlay channel parameters	
Channels	0
Statistics parameters	
Show maximum	True
Show maximum at	True
Show minimum	True
Show minimum at	True
Show magnitude	True
Show average	True
Show Std.Deviation	True

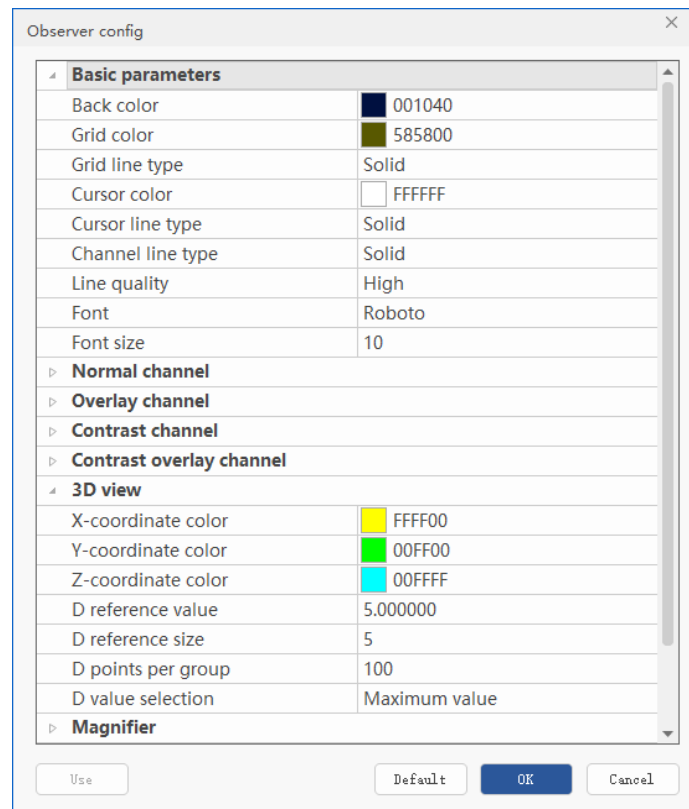
Default
OK
Cancel

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: - use the TSIZE instruction to read. - view in the "Controller Status" window. - 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

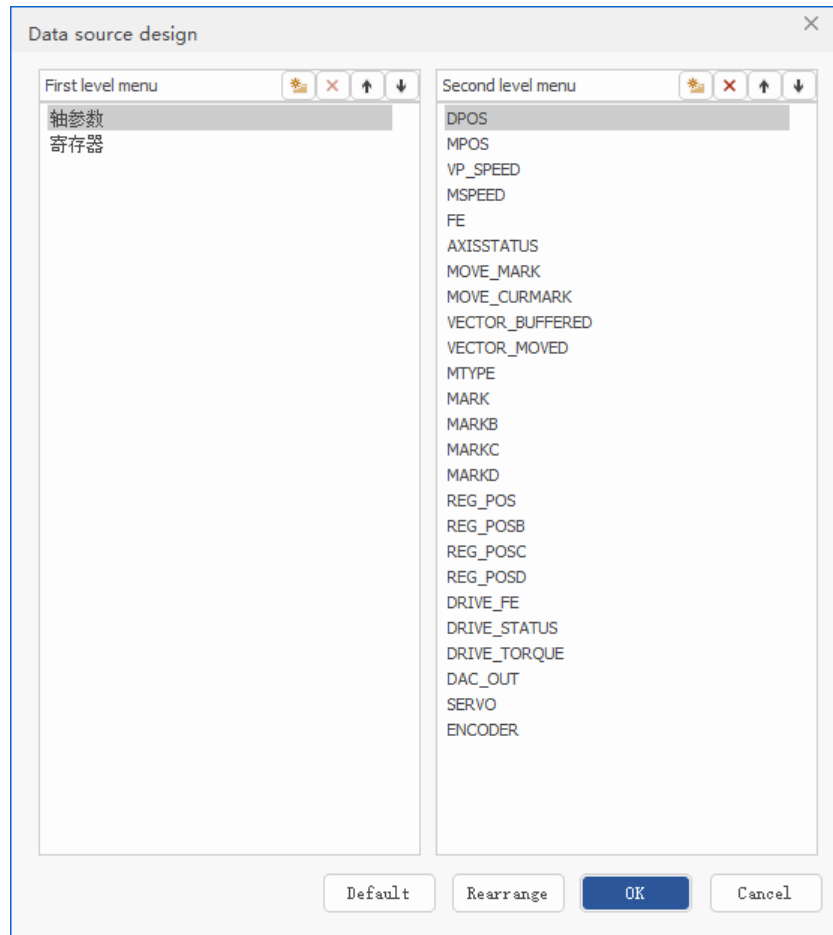
Click menu above "Config" button, then click "observe config", then corresponding window will appear, after configured, click "use" to preview how it is after modified, then click "OK".



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.

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.

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.

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 $4 \times 30000 = 120000$ 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.

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

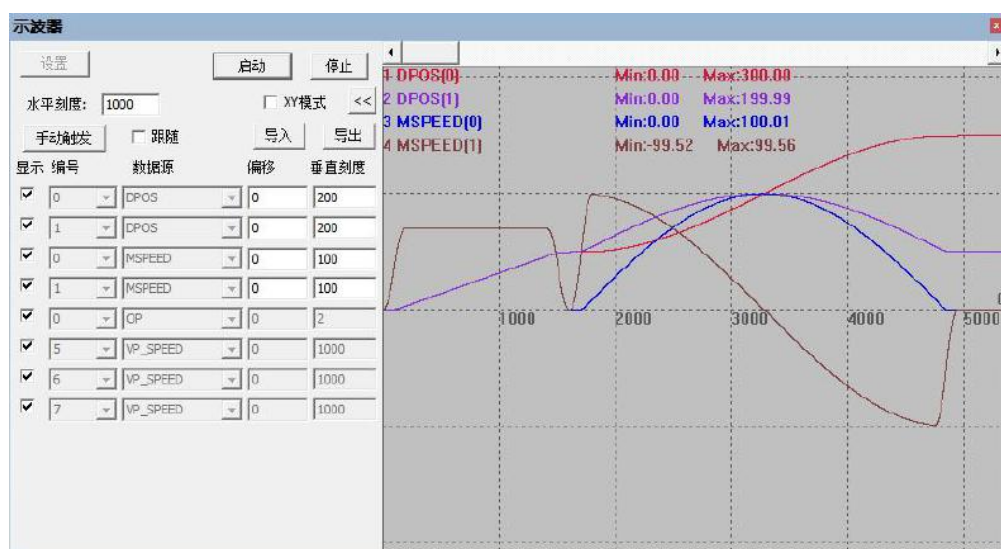
TRIGGER

'trigger oscilloscope automatically

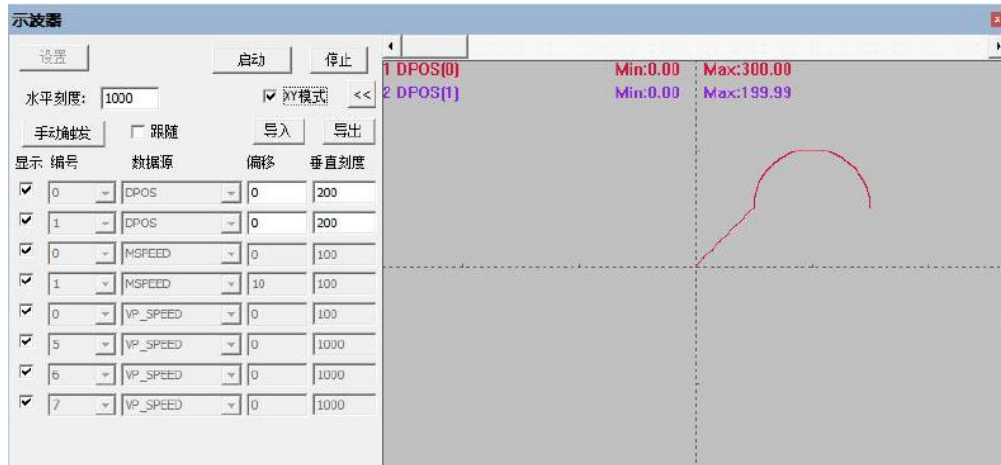
MOVE(100,100)

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)

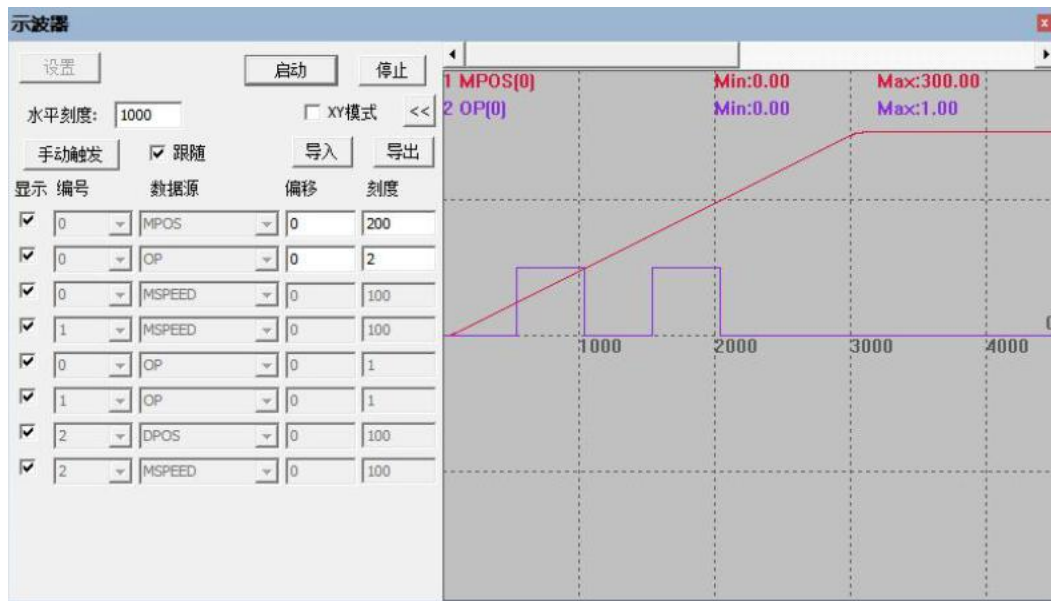
TABLE(0,50,100,150,200) 'coordinate of comparison point

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

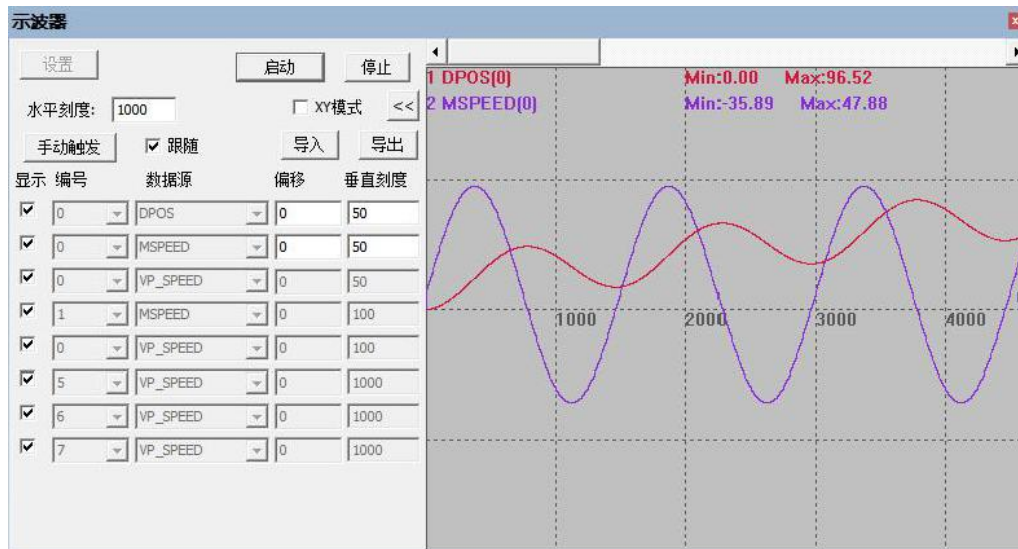
 TRACE deg/stepdeg, X

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 = $300/200 = 1.5\text{s}$



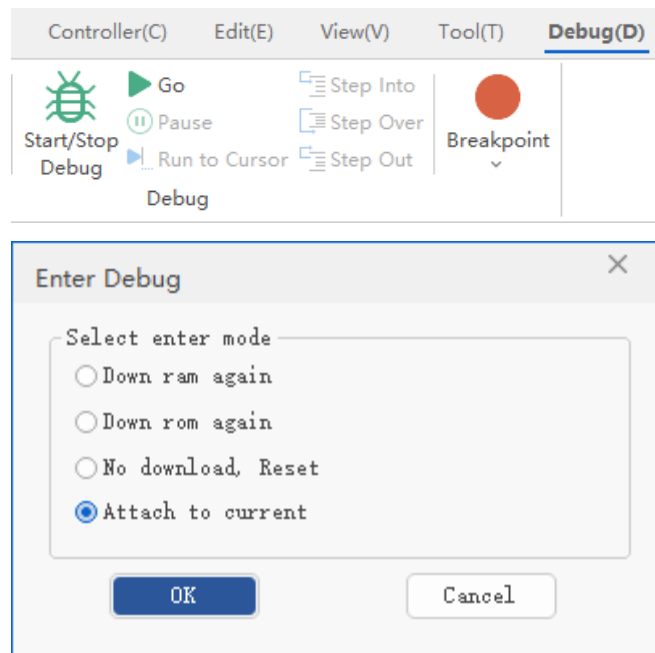
2.5. Program Debug

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 RTSys 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.

Task & Watch Windows

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

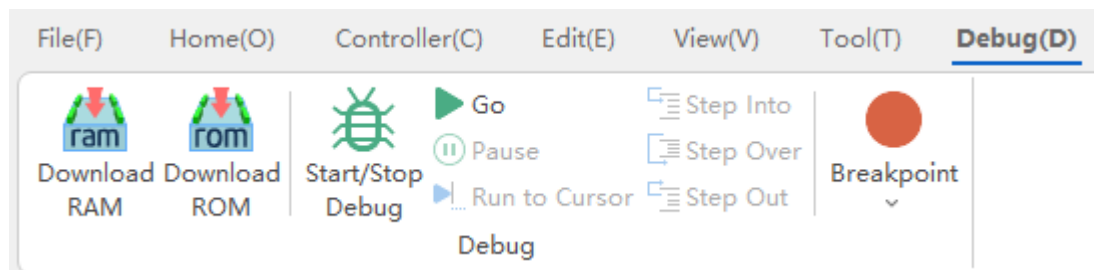


Task window is used to see running status of task, file and task running line number.

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.

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

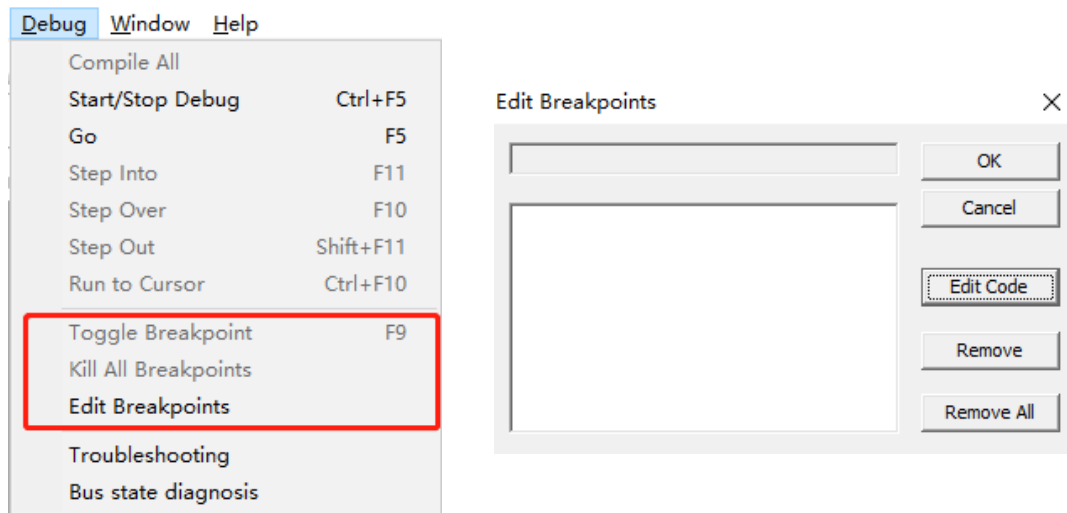
When the program is inconsistent with the controller or the program is not downloaded in time after re-modification, it will cause the line number specified by debugging to be offset. Motions that are currently submitted when paused are not paused.

Breakpoint Debug

Program can be obtained and paused through adding breakpoints

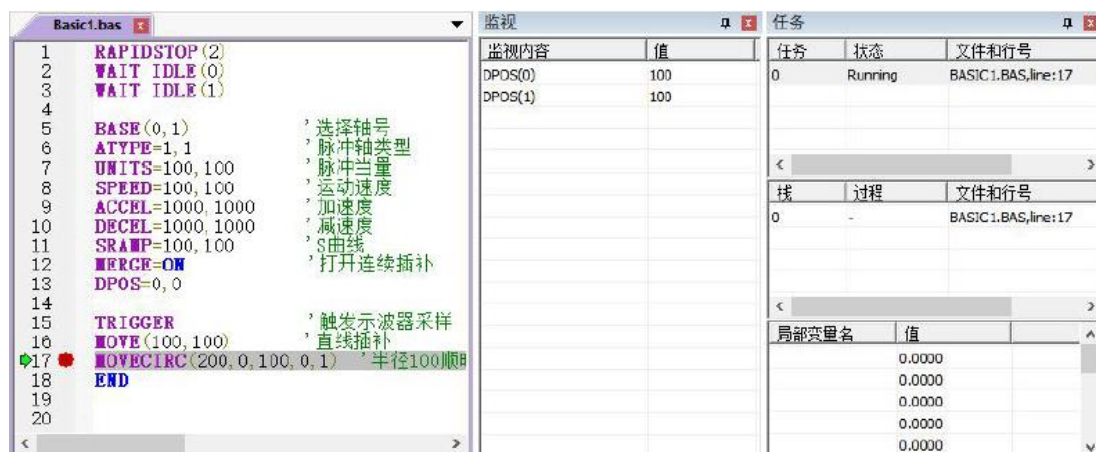
Breakpoint debugging can view the specific running process of program, which is mainly used to judge program logic errors. With watch content and axis parameter changes, you can view the impact of each step of the program execution on registers, variables, arrays, etc.

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. "View" Window

RTSys 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 RTSys software, and open the "RTSys User Manual" to view.



Chapter 3 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

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.

RTBasic 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
ELSEIF 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

Label1:
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

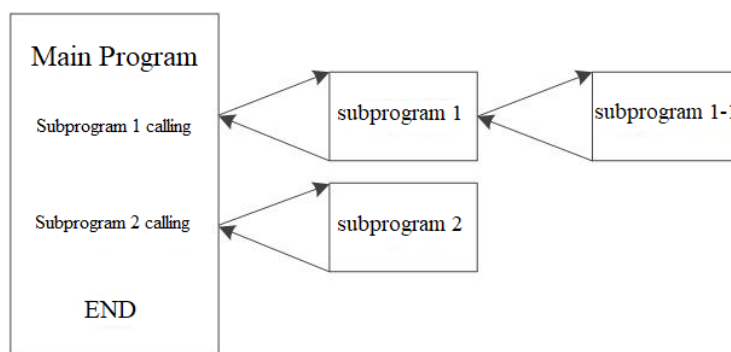
PRINT 'motion finishes

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

```

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.

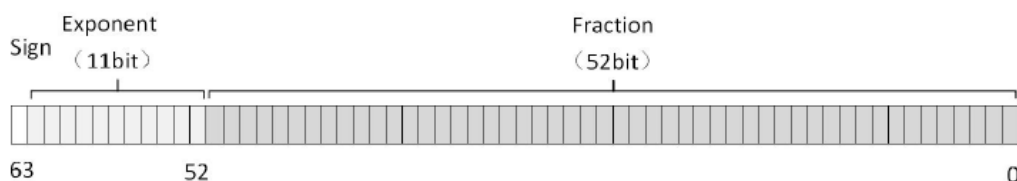
Single-precision floating point 32-bit:

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 & array & variables (≤ZMC3XX series)		
TABLE & array & variables	64 bits floating	1.7E-308 to 1.7E+308

(≥ZMC4XXseries)		
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
<u>DIM</u>	Defined array can be used as character string directly, each element represents a byte.
""	Use "" to define constant type character string directly.
<u>CHR</u>	Convert ASCII to a character string, it only occupies one byte.
<u>MODBUS_STRING</u>	Standard MODBUS protocol defines character string, each 16-bit register stores 2 bytes.
<u>VRSTRING</u>	VR list acts as character string, 1 VR stores 1 byte.
+	Operational character, which is used to combine two characters.
<u>VAL</u>	Convert Number character string to numerical.
<u>TOSTR</u>	Convert numerical to number character string.
<u>STRCOMP</u>	Compare different character strings
<u>DMCPY</u>	Array copy function, also can copy character string.
<u>HEX</u>	Return hexadecimal value, only for print purpose.
<u>DATE\$</u>	Return date in "dd: mm: yyyy" format.
<u>DAY\$</u>	Return the English name of today's week
<u>TIME\$</u>	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 RTSys 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 [SETCOM](#) 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(38400,8,1,0,0,3) '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. RTSys Three Programming Methods

Hybrid Programming

- ✧ ZDevelop software supports 3 kinds of programming methods, they are RTBASIC, RTPLC ladder diagram and RTHMI configuration. It also supports these 3 languages hybrid programming. The programmed procedure through RTSys can be downloaded to ZMOTION motion controller.
- ✧ RTBSIC, RTPLC and RTHMI can run multi-task among them. RTBASIC can run multi-task, RTPLC and RTHMI both only can run one task.
- ✧ For example, see the below, two different RTBASIC 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.

FileView	
FileName	AutoRun
Basic2.bas	0
Basic3.bas	1

FileView	
FileName	AutoRun
Plc1.plc	0
Plc2.plc	

- ✧ Both RTPLC programming and RTBASIC 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.

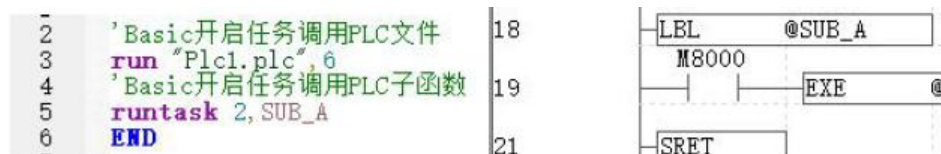
PLC & BASIC Call Each Other

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_REG(0)
	D1	MODBUS_LONG	MODBUS_REG(1)
	D1023	MODBUS_IEEE	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 [OUT](#), 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 RTBASIC.
 - ✧ "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 RTSys software to controller, size of each register on this controller can be checked through RTSys software "Controller" – "State the controller". Also it can input "?*max" in output window to see the amount of each register. Different controllers have different store space.

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 RTBasic 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

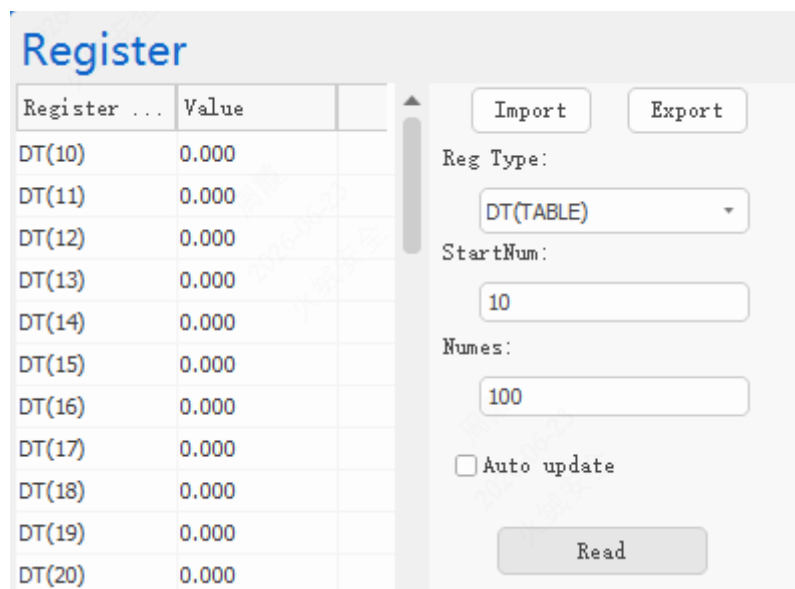
CAM(10,17,100,500) 'Motion track is from TABLE(10) to TABLE(17)

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.



FLASH

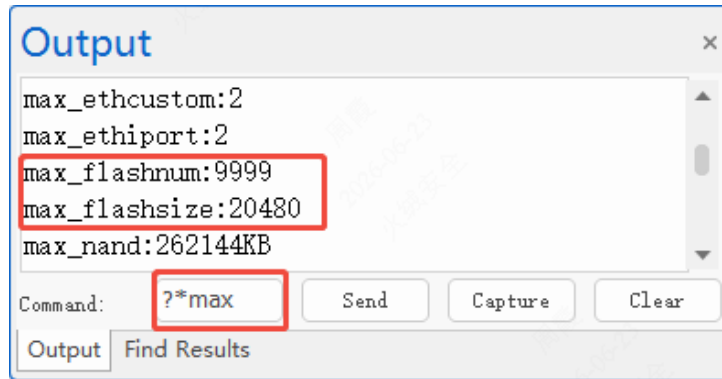
Strictly speaking, FLASH is closely related to the register, but it is not a register, so it is described in this chapter.

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: write VAR, ARRAY1, ARRAY2 data into FLASH block 1 in tur

FLASH_WRITE 1, VAR, ARRAY1, ARRAY2

'read Flash data: 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

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 save as an integer, and use [VRSTRING](#) to save as 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
```

Register ...	Value
VR(0)	666888.000
VR(1)	1433223.000
VR(2)	1433223.000
VR(3)	1000.000
VR(4)	0.000
VR(5)	50.000
VR(6)	5.000
VR(7)	100.000
VR(8)	0.000
VR(9)	1.000

Reg Type: VR

StartNum: 0

Numes: 10

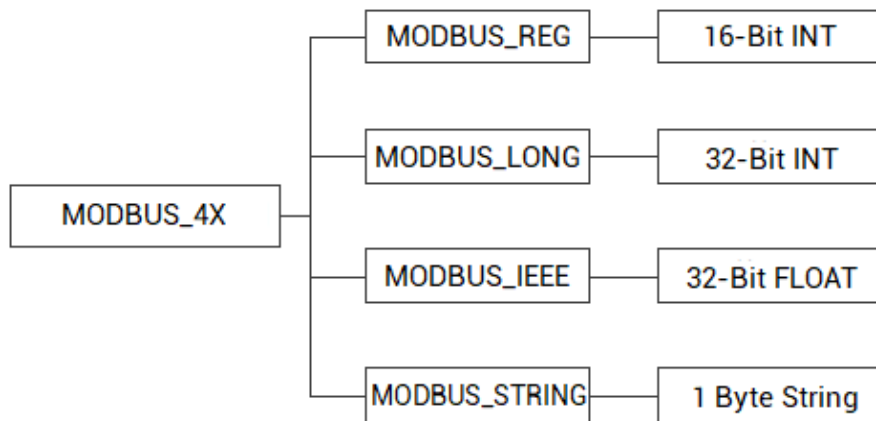
☐ Auto update

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 HMI, it is called MODBUS_4X, see the below:



The MODBUS word registers in the controller occupy 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 No. address.

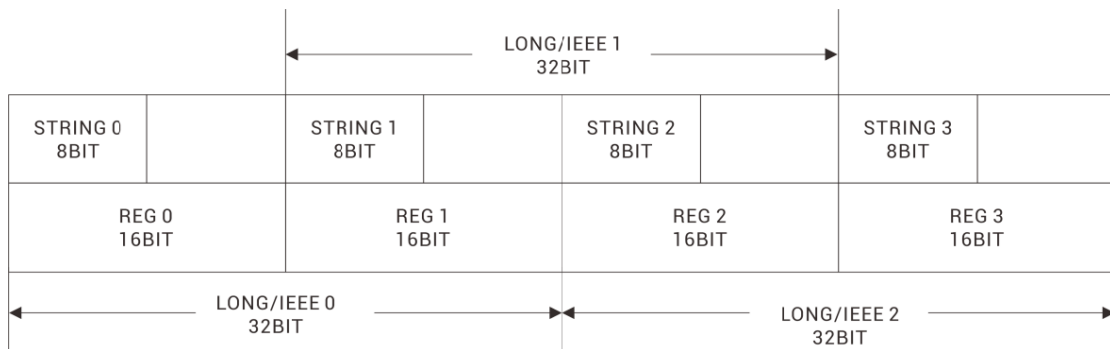
- One MODBUS_LONG(0) occupies two REG addresses, MODBUS_REG(0) and MODBUS_REG(1).
- One MODBUS_LONG(1) occupies two REG addresses, MODBUS_REG(1) and MODBUS_REG(2)
- One MODBUS_IEEE(0) occupies two REG addresses, MODBUS_REG(0) and MODBUS_REG(1).
- One MODBUS_IEEE(1) occupies two REG addresses, MODBUS_REG(1) and MODBUS_REG(2).

Pay attention not to overlap MODBUS_REG, MODBUS_LONG, MODBUS_IEEE addresses in users application programs.

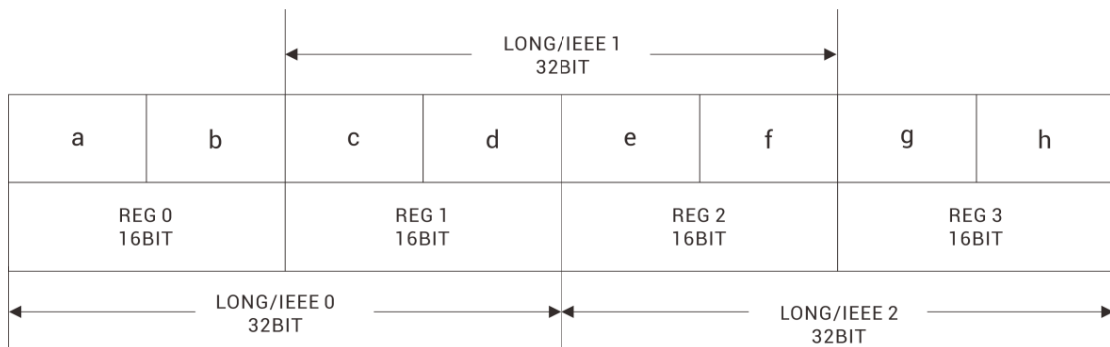
✧ Calculation Method

MORBUS_REG(1) is high bit, MODBUS_REG(0) is low bit, MODBUS_LONG (0) = MODBUS_REG(1) * 2¹⁶ + MODBUS_REG(0).

◇ MODBUS_4X Space Graphic Description:



For example, MODBUS_STRING(0, 20) = "abcdefgh", it will be:



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)

MODBUS_REG(0)=0 '初始化置0	监视内容	值
MODBUS_REG(1)=0 '初始化置0	MODBUS_LONG(0)	70000
'modbus_long赋值70000	MODBUS_REG(0)	4464
'modbus_reg范围-32768~32767	MODBUS_REG(1)	1
MODBUS_LONG(0)=70000		
?MODBUS_REG(0),MODBUS_REG(1)		
'打印出 reg(0)为4464, reg(1)为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 (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

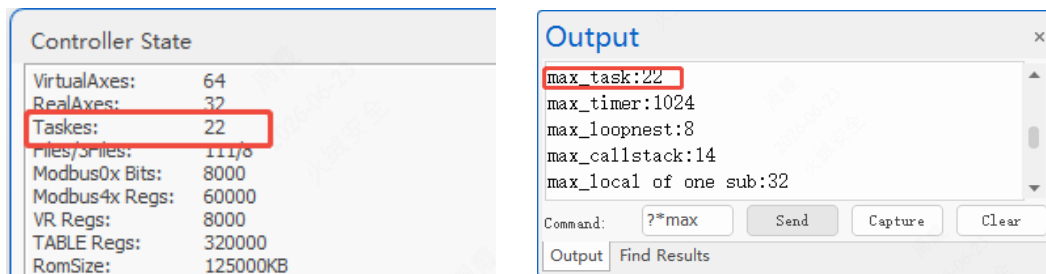
MODBUS Word Register Address	Meaning
0~7999	Customized use for users, MODBUS_REG, MODBUS_IEEE and MODBUS_LONG can be used together
8000~8099	special D register programmed by PLC

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

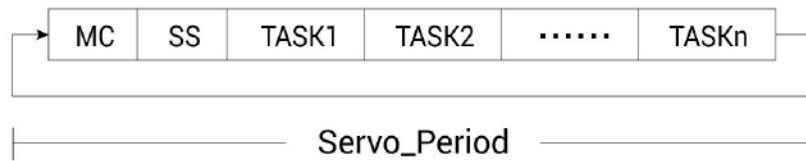
3.4. Multi-Task Program

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 RTSys 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 RTSys 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.

Basic1.bas	0	0	Stopped	BASIC1.BAS,line:31
Basic2.bas	1	1	Stopped	BASIC2.BAS,line:18
Basic3.bas				
Plc1.plc	2	2	Running	PLC1.PLC,line:1
Hmi1.hmi	3	3	Running	HMI1.HMI,line:1

- ✧ 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.

Check Multi-task Status

Task has 3 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.



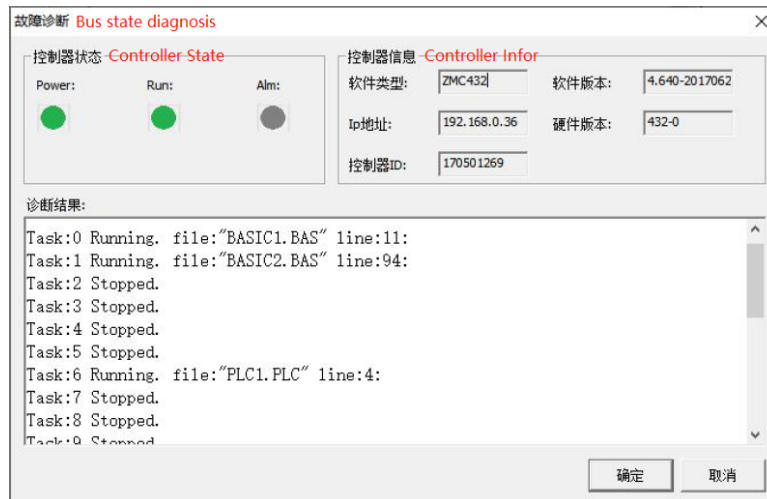
任务	状态	文件和行号
0	Stopped	BASIC1.BAS,line:15
1	Running	BASIC2.BAS,line:107
2	Paused	BASIC1.BAS,line:26
3	Stopped	PLC1.PLC,line:13
6	Running	PLC1.PLC,line:1

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.



Multi-Task Start & 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.

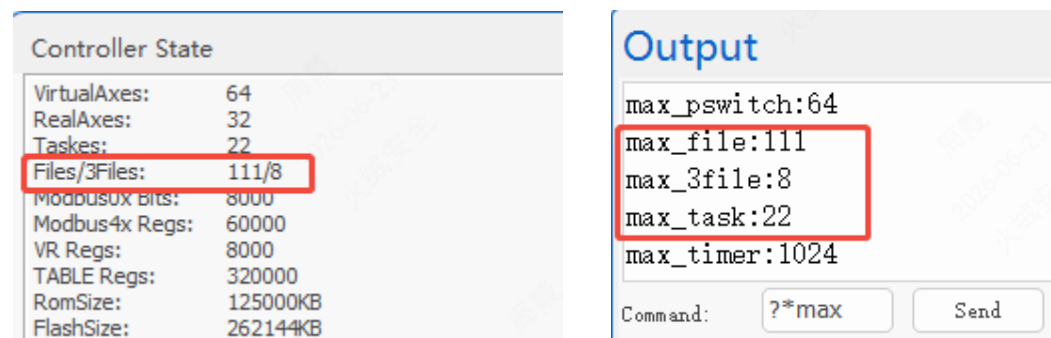
RUNTASK: start a new task and run one SUB or one program with labels.

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.

文件视图		任务		
文件名	自动运行	任务	状态	文件和行号
Basic11.bas	0	0	Running	BASIC11.BAS,line:45
		1	Running	BASIC11.BAS,line:82

Send online command: STOPTASK 0

Output	
>>STOPTASK 0	
Command: STOPTASK 0 Send Capture Clear	

Stop task 0

文件视图		任务		
文件名	自动运行	任务	状态	文件和行号
Basic11.bas	0	0	Stopped	BASIC11.BAS,line:57
		1	Running	BASIC11.BAS,line:82

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.

Task Pause & Resume

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:

RESUMETASK 'continue to running task 1

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

文件视图		任务		
文件名	自动运行	任务	状态	文件和行号
Basic11.bas	0	0	Running	BASIC11.BAS,line:45
		1	Running	BASIC11.BAS,line:82

Send online command to control task is paused or resumed.

Output

```
>>PAUSETASK 0
>>RESUMETASK 0
```

Command: RESUMETASK 0

Send

Capture

Clear

Send: PAUSETASK 0

Task 0 is paused.

文件视图		任务		
文件名	自动运行	任务	状态	文件和行号
Basic11.bas	0	0	Paused	BASIC11.BAS,line:53
		1	Running	BASIC11.BAS,line:82

Send: RESUMETASK 0

Task 0 resumes the operation state.

文件视图		任务		
文件名	自动运行	任务	状态	文件和行号
Basic11.bas	0	0	Running	BASIC11.BAS,line:60
		1	Running	BASIC11.BAS,line:82

Basic & PLC Task Call Each Other

➤ First, Basic calls PLC task.

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

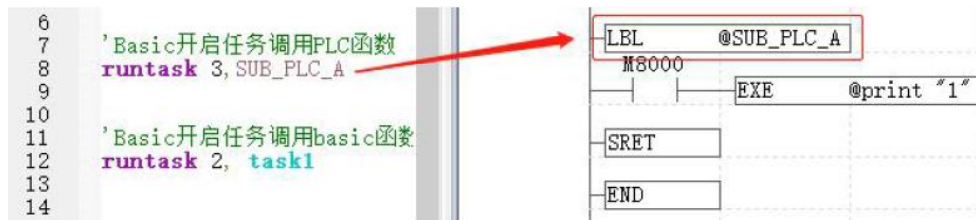
The screenshot shows the '文件视图' (File View) window on the left and the 'Basic1.bas' file editor on the right. The '文件视图' window has a table with two columns: '文件名' (File Name) and '自动运行' (Auto Run). The files listed are Basic1.bas, Basic2.bas, Basic3.bas, Plc1.plc, and Plc2.plc. The '自动运行' column shows '0' for Basic1.bas. A red arrow points from the 'Plc1.plc' file in the list to the 'run' statement in the Basic1.bas file editor on the right.

文件名	自动运行
Basic1.bas	0
Basic2.bas	
Basic3.bas	
Plc1.plc	
Plc2.plc	

The 'Basic1.bas' file editor shows the following code:

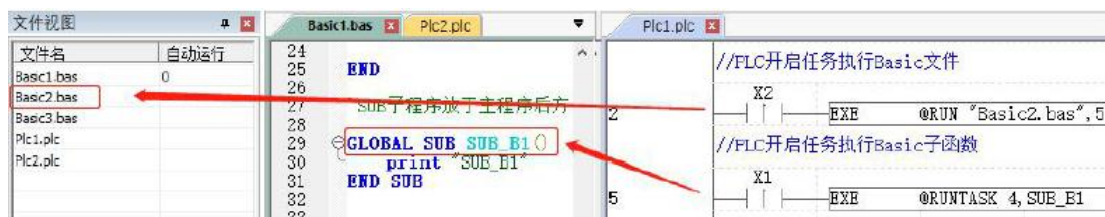
```
1  
2  
3  
4  
5 'Basic开启任务调用PLC文件  
6 run "Plc1.plc", 6  
7
```

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.



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.

任务	状态	文件和行号
0	Running	BASIC1.BAS,line:1

栈	过程	文件和行号
0	-	BASIC1.BAS,line:1

局部变量名	值
	0.0000
	0.0000
	0.0000
	0.0000
	0.0000

There is only auto-task 0 when power-on

Basic1.bas

```

1  RUNTASK 1, taskA
2  RUNTASK 2, taskB
3  RUNTASK 3, taskC
4  END
5
6  taskA:
7  PRINT "atask", TICKS
8  DELAY(1000)
9  GOTO taskA
10
11 taskB:
12 PRINT "btask", TICKS
13 DELAY(1000)
14 GOTO taskB
15
16 taskC:
17 PRINT "ctask", TICKS
18 DELAY(1000)
19 GOTO taskC
20

```

任务	状态	文件和行号
0	Running	BASIC1.BAS,line:2
1	Running	BASIC1.BAS,line:7

栈	过程	文件和行号
0	-	BASIC1.BAS,line:2

局部变量名	值
	0.0000
	0.0000
	0.0000
	0.0000
	0.0000

Task 0 starts task 1 to run

Basic1.bas

```

1  RUNTASK 1, taskA
2  RUNTASK 2, taskB
3  RUNTASK 3, taskC
4  END
5
6  taskA:
7  PRINT "atask", TICKS
8  DELAY(1000)
9  GOTO taskA
10
11 taskB:
12 PRINT "btask", TICKS
13 DELAY(1000)
14 GOTO taskB
15
16 taskC:
17 PRINT "ctask", TICKS
18 DELAY(1000)
19 GOTO taskC
20

```

任务	状态	文件和行号
0	Running	BASIC1.BAS,line:3
1	Running	BASIC1.BAS,line:8
2	Running	BASIC1.BAS,line:12

栈	过程	文件和行号
0	-	BASIC1.BAS,line:3

局部变量名	值
	0.0000
	0.0000
	0.0000
	0.0000
	0.0000

Task 0 starts task 1 and task 2 to run

Basic1.bas

```

1  RUNTASK 1, taskA
2  RUNTASK 2, taskB
3  RUNTASK 3, taskC
4  END
5
6  taskA:
7  PRINT "atask", TICKS
8  DELAY(1000)
9  GOTO taskA
10
11 taskB:
12 PRINT "btask", TICKS
13 DELAY(1000)
14 GOTO taskB
15
16 taskC:
17 PRINT "ctask", TICKS
18 DELAY(1000)
19 GOTO taskC
20

```

任务	状态	文件和行号
0	Stopped	BASIC1.BAS,line:6
1	Running	BASIC1.BAS,line:8
2	Running	BASIC1.BAS,line:13
3	Running	BASIC1.BAS,line:18

栈	过程	文件和行号

局部变量名	值

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

3.5. Three Kinds of Interruption

- ✧ There are 3 types of RTBasic 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.

Power Failure Interrupt

- ✧ 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

```

External Interrupt

- ✧ 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

```

Timer Interrupt

- ✧ 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

```

```

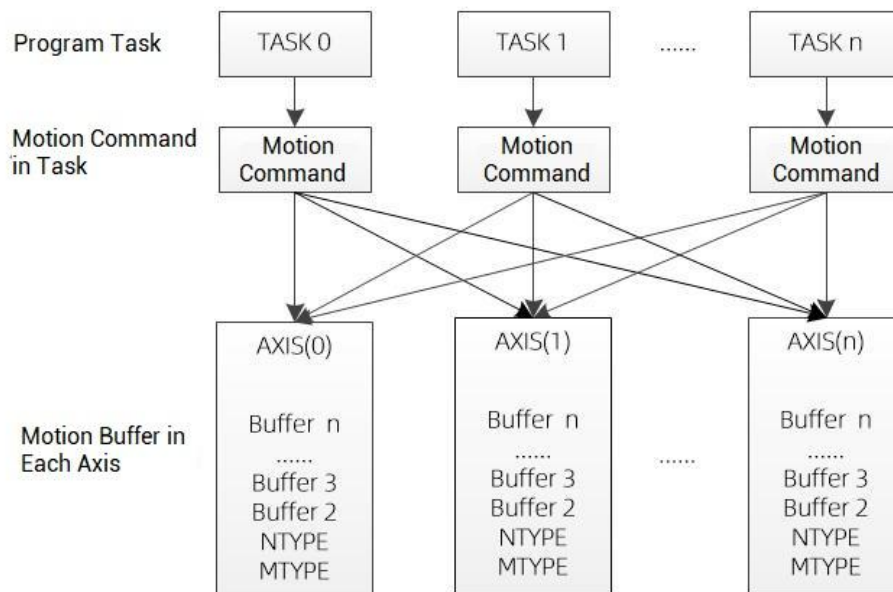
GLOBAL SUB ONTIMER0()      'interruption program
    PRINT "ontimer0 enter"
    'TIMER_START(0,100)    'execute interruption periodically, open timer again in SUB
END SUB

```

3.6. Motion Buffer

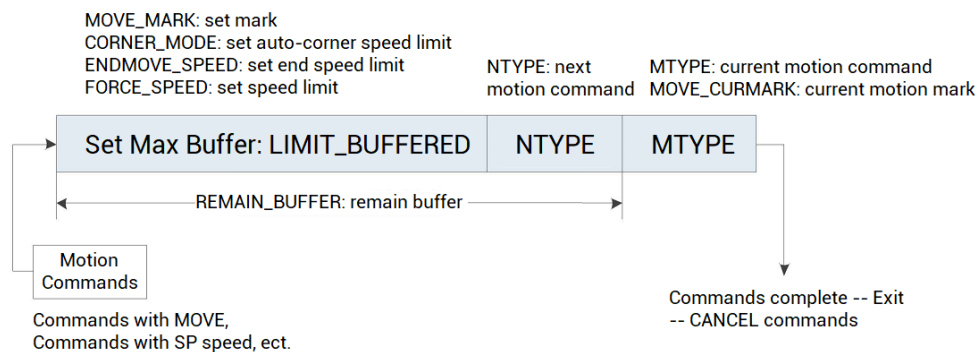
The Concept of Motion Buffer

- ✧ When there are many running commands, the motion controller has one buffer area that can save motions commands, then they will be called out for execution, in this way, program scanning time can be omitted, and the real-time can be promoted. In addition, make the program run without blocked.
- ✧ 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 current running motion instruction, [NTYPE](#) is the first buffer motion instruction.
- ✧ Motion instructions of any program can enter motion buffer of any axis, assigned by axis No.
- ✧ Motion buffer areas of each axis are independent, they don't bother each other.

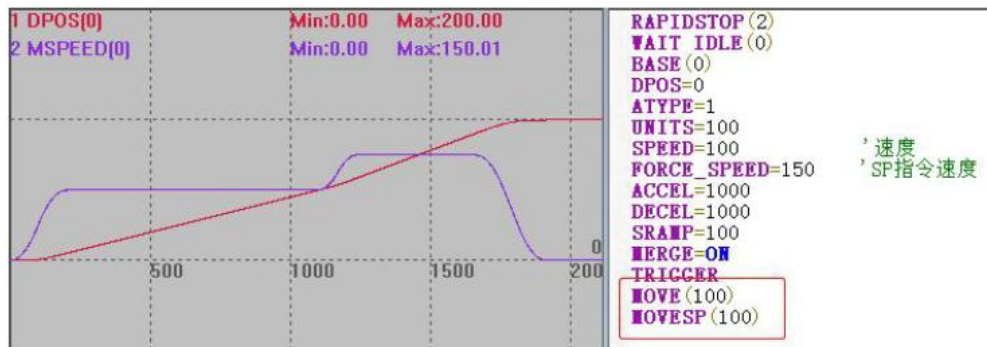


Motion Buffer

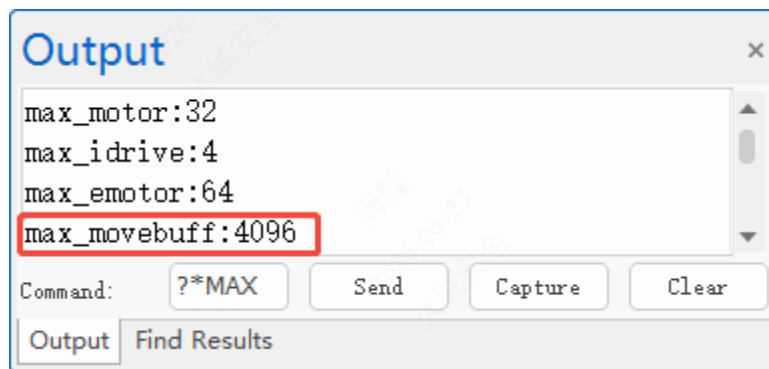
- ✧ During the program scanning, the scanned motion instructions are stored in the motion buffer of the corresponding axis, and the motion instructions are fetched and executed from the motion buffer in the order of first-in, first-out. In addition, it also includes a series of output instructions in 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 STARTMOVE_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 FORCE_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-V2 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.

Motion Buffer Overflow

- ✧ 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 V1.00.02 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 511 circular interpolation instructions, and the value of “i” is 511 after downloading the program, which means that the current FOR loop has not been executed and the program is blocked.

The screenshot displays the Zmotion RTBasic simulator interface. On the left, the program code for 'Basic1.bas' is shown, featuring a FOR loop from i=0 to 1000, with a comment indicating that the buffer can store up to 511 circular interpolation instructions. The '轴参数' (Axis Parameters) window on the right shows the 'MOVES_BUFFERED' value at 458, indicating the buffer is nearly full.

轴选择	参数选择	轴0	轴1
LOADED		0	0
MSPEED		96.8600	24.8500
MTYPE		4	4
NTYPE		4	4
REMAIN		171.8500	0
VECTOR_BUFFERED		144050.3500	144050.3500
VP_SPEED		100	9.5100
AXISSTATUS		0h	0h
MOVE_MARK		459	6
MOVE_CURMARK		0	0
AXIS_STOPREASON		800h	0h
MOVES_BUFFERED		458	458

Motion Buffer Overflow Effect – Circular Interpolation

The screenshot displays the Zmotion RTBasic simulator interface. On the left, the program code for 'Basic1.bas' is shown, featuring a FOR loop from i=0 to 5000, with a comment indicating that the buffer can store up to 4094 linear interpolation instructions. The '轴参数' (Axis Parameters) window on the right shows the 'MOVES_BUFFERED' value at 4094, indicating the buffer is full.

轴选择	参数选择	轴0	轴1
LOADED		0	0
MSPEED		70.7100	70.7100
MTYPE		1	1
NTYPE		1	1
REMAIN		128.7400	0
VECTOR_BUFFERED		579099.0200	579099.0200
VP_SPEED		100	70.7100
AXISSTATUS		0h	0h
MOVE_MARK		4096	6
MOVE_CURMARK		1	1
AXIS_STOPREASON		800h	0h
MOVES_BUFFERED		4094	4094

Motion Buffer Overflow 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.

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

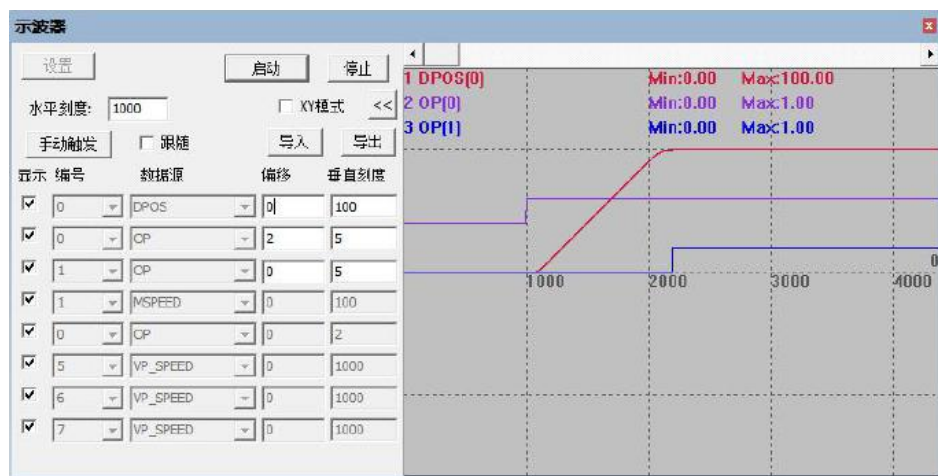

```

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 4 Communication Methods

4.1. Serial Port Communication

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.

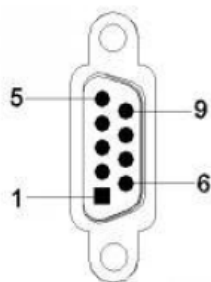
Controller's serial port protocols are MODBUS_RTU, they are as slave station by default. RS232 and RS485 can be set as master station through SETCOM instruction, and communication rate and other parameters are configured through [SETCOM](#) instruction.

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.

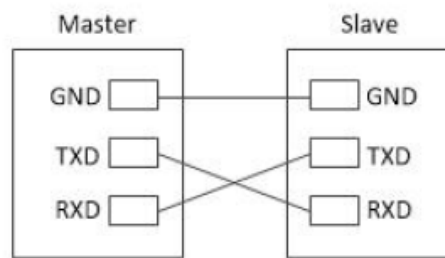
RS232 controller uses DB-9 interface, below is the pin signal description:



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

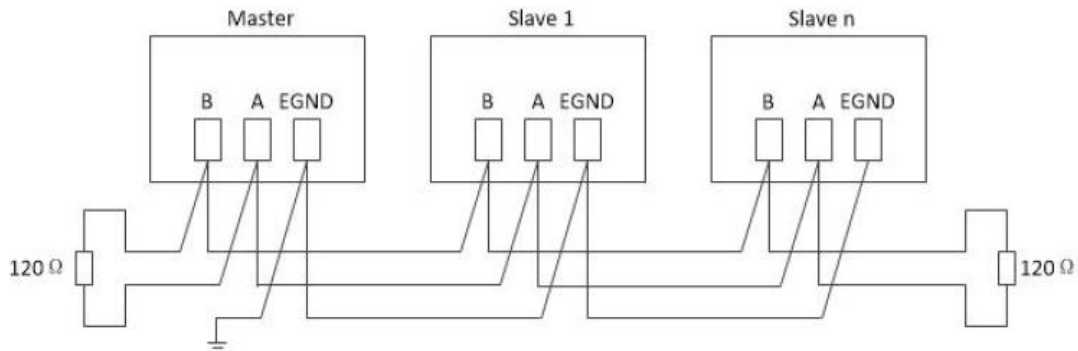
It mainly provides the connection of multiple communication devices of the master/slave station, and theoretically supports 128 nodes, one master and multiple slaves. When it is used as a master station, it can be connected to drives, converters, temperature controllers, etc. to control data read and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

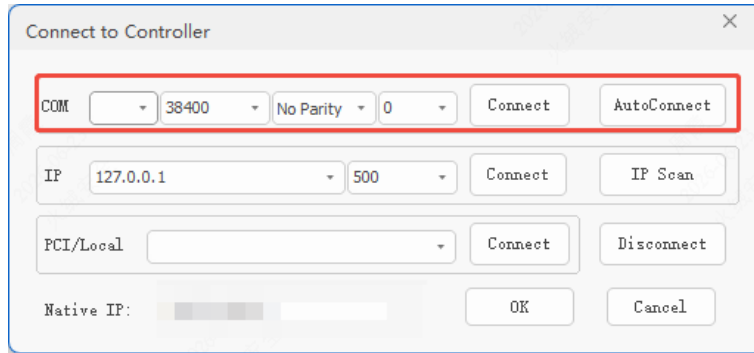
Serial Connection Method

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 RTBasic 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 modifying 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.

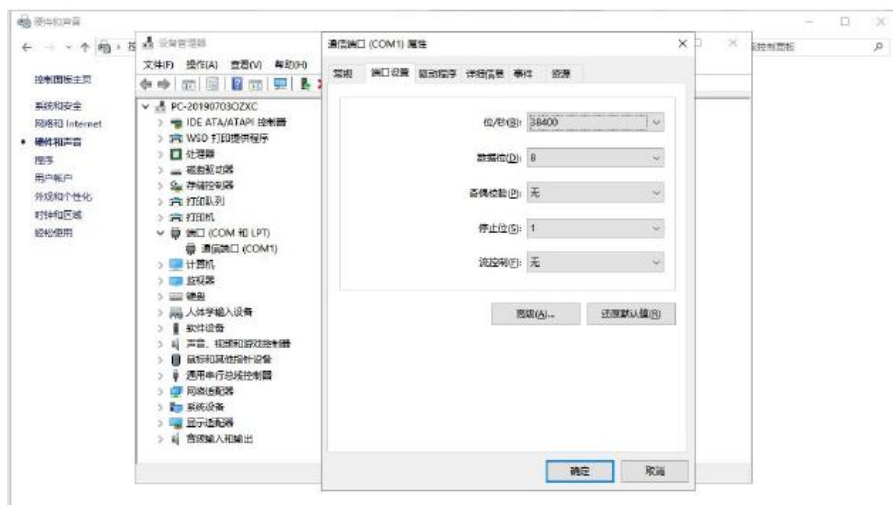
SETCOM instruction mode parameters configuration protocol:

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. ETHERNET 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

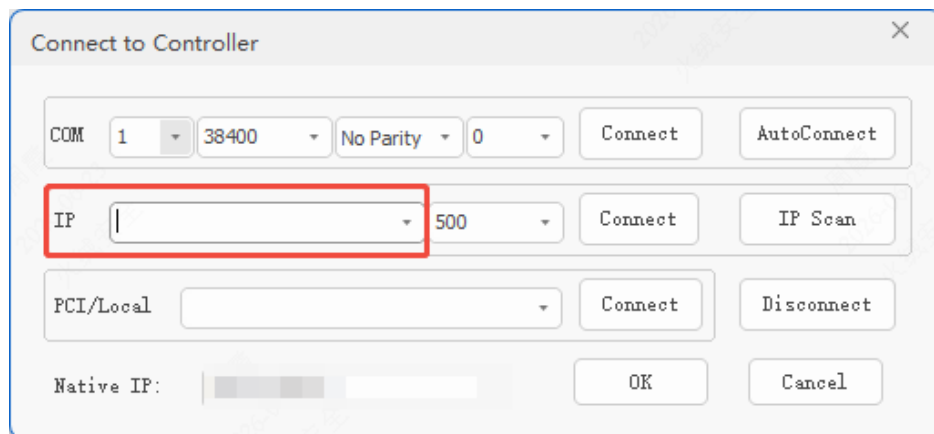
has one EtherNET interface, but there are at least 2 network port channels at the bottom. When net port is needed to connect to multiple equipment, the switch can be used. The number of net port channel can be viewed through ?*PORT instruction.



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

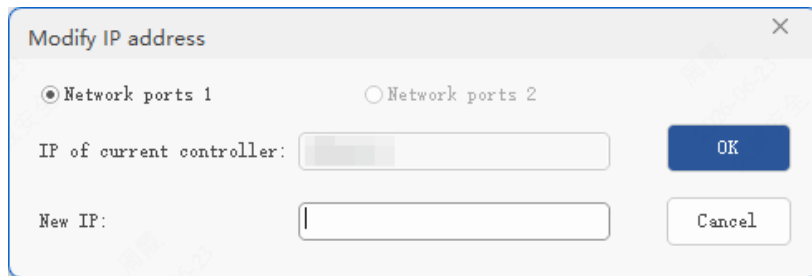
		not. ✚ Check whether the cable insert slot is damaged. ✚ Note: the cable is connected to EtherNET, not EtherCAT.
3	Fail to connect to the controller.	✚ Check whether controller IP address is the same as PC IP address, they must be at the same net segment, see next for details of modification. ✚ If controller 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.	✚ Recommended to use metal interface with shielded cable. For serious interference, using a crystal head network cable without shielding layer will cause unstable communication & 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.

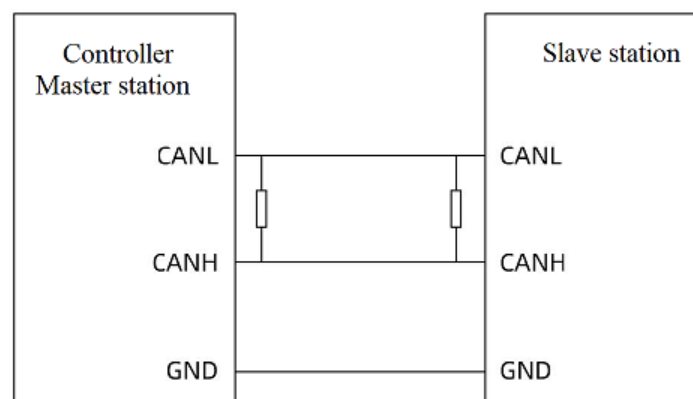
Then, open “Connect to controller” window to connect again.



4.3. CAN Bus Communication

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 integrate 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 Syntax: 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 RTSys 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 file 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
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(filename).BIN or SD(filename).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 “[U disk instructions](#)”

4.5. EtherCAT Communication

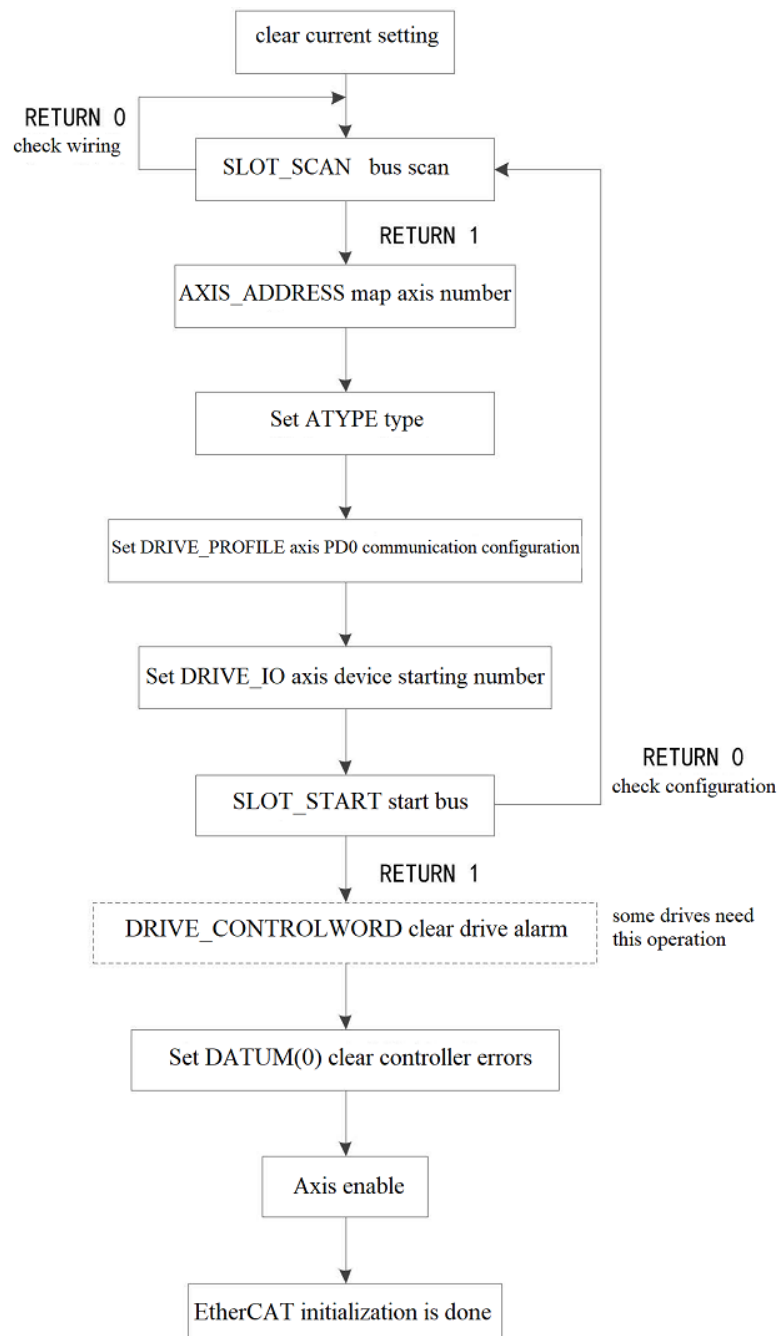
EtherCAT Bus Initialization

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 [routines](#).



Basic concepts involved:

1. NODE node No.: 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 No.: 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 No.: the No. 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 No.), 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 No) 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	Meanings
SLOT_SCAN	Scan the device
SLOT_START	Start EtherCAT connection
SLOT_STOP	Stop EtherCAT connection
ATYPE	Axis Type; ATYPE=65 -- 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 numbers
NODE_STATUS	Devices status
NODE_AXIS_COUNT	Drive numbers (with the device)
NODE_IO	IO serial No. setting of devices
NODE_AIO	AIO serial No 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.

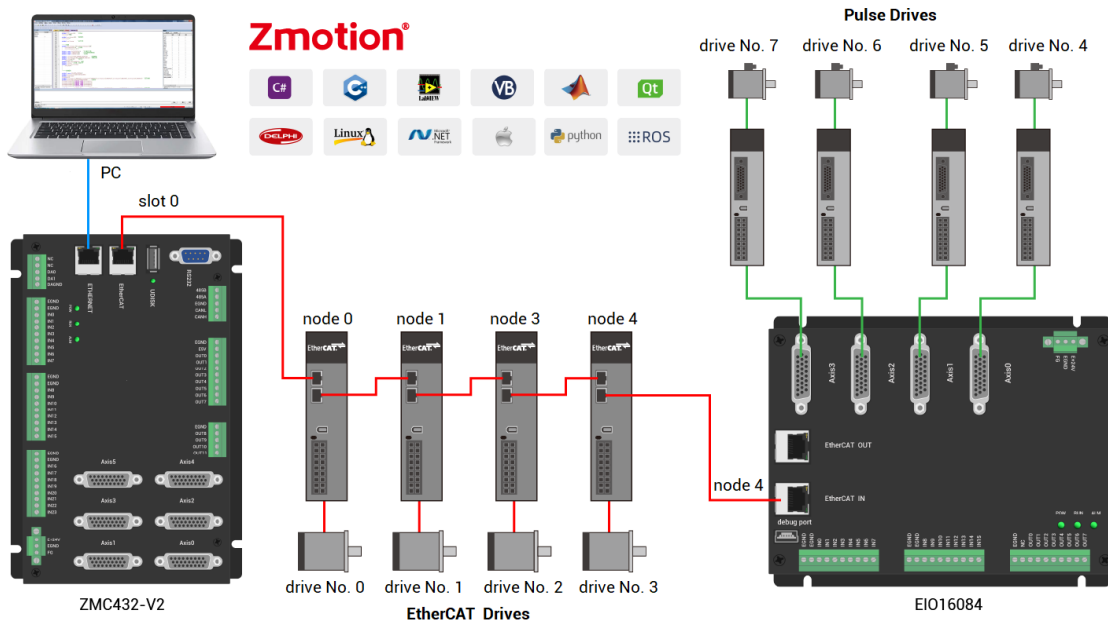
EtherCAT Bus & Drive Communication

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 No. 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 RTECH 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.

Connect to Expansion Module by EtherCAT

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.

Syntax:

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. 0 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:

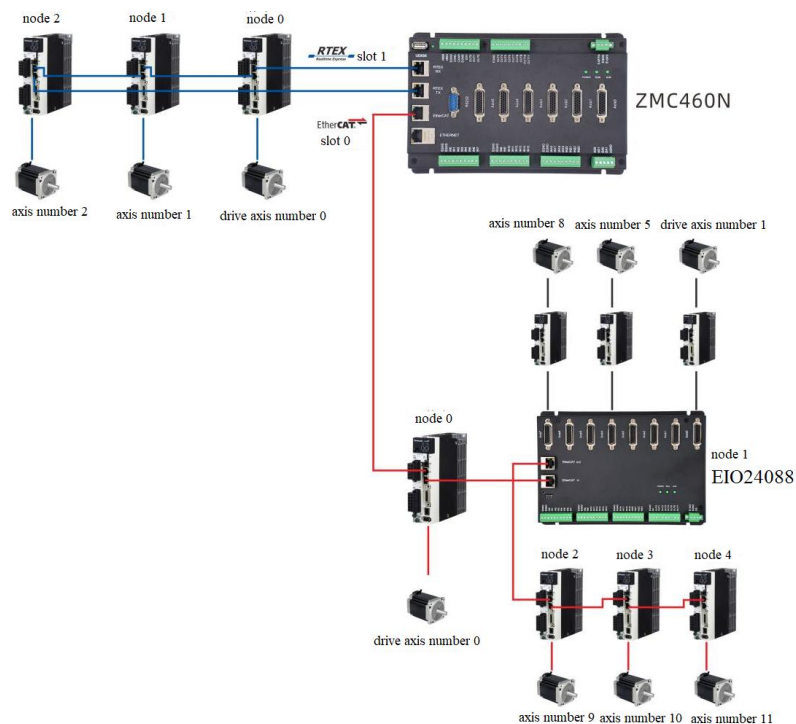
$\text{AXIS\_ADDRESS}(0) = (0 \ll 16) + 0 + 1$ The first ECAT drive, drive number 0, bound as axis 0
 $\text{AXIS\_ADDRESS}(2) = (0 \ll 16) + 1 + 1$ The second ECAT drive, drive number 1, bound as axis 2
 $\text{AXIS\_ADDRESS}(1) = (0 \ll 16) + 2 + 1$ The third ECAT drive, drive number 2, bound as axis 1
 $\text{ATYPE}(0) = 65$ 'Set to ECAT axis type, 65-position 66-speed 67-torque $\text{ATYPE}(1) = 65$
 $\text{ATYPE}(2) = 65$

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.

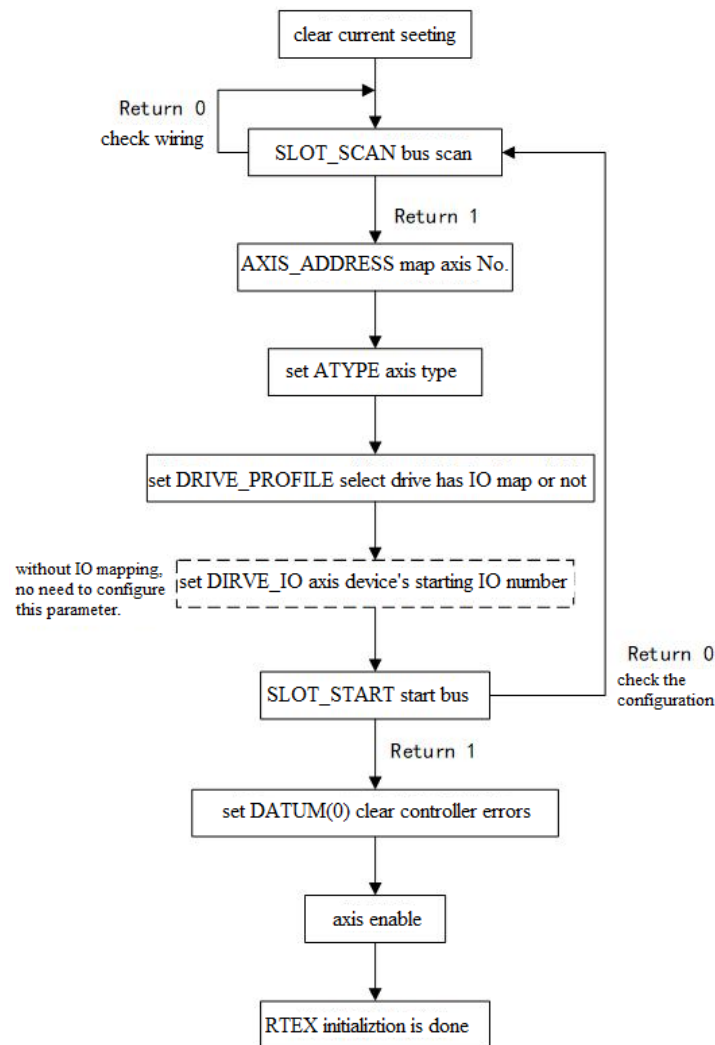
RTEX wiring method refers to below figure:



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	Meanings
SLOT_SCAN	Scan the device
SLOT_START	Start RTEX connection
SLOT_STOP	Stop RTEX connection
ATYPE	Axis Type; ATYPE=50 -- RTEX period position control
AXIS_ADDRESS	Axis address configuration
WDOG	Total enable control of axes
SERVO_PERIOD	Refresh rate
AXIS_ENABLE	Axis enable
?*RTEX	Bus information output
NODE_COUNT	Devices numbers
NODE_STATUS	Devices status
NODE_AXIS_COUNT	Drive numbers (with the device)
NODE_IO	IO serial No. setting of devices
NODE_AIO	AIO serial No 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	Control word of drive
DRIVE_CW_MODE	operation mode of drive control word.
DRIVE_IO	serial number setting of remote IO
DRIVE_CLEAR	Clear drive errors; when without errors, it will alarm.
DRIVE_READ	Drive read parameters
DRIVE_WRITE	Drive writing parameters
AXISSTATUS	Axis status

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

Chapter 5 Motion Control Functions

5.1. Common Motion Modes

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.

Single-axis Jog Motion

Related instructions:

Instruction	Description	Usage
VMOVE	Continuous motion, positive or negative direction	VMOVE(motion direction)
CANCEL	Single-axis stops, 4 kinds 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.

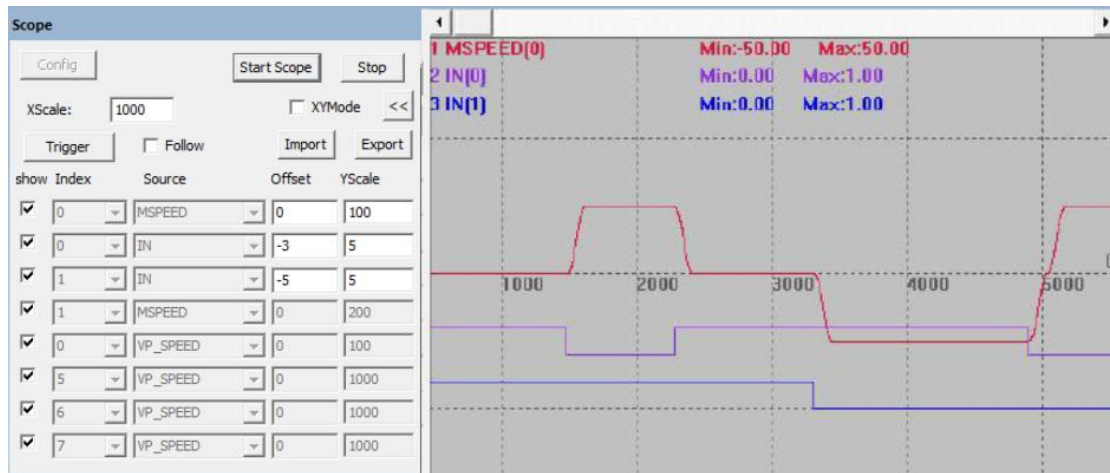
```

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.

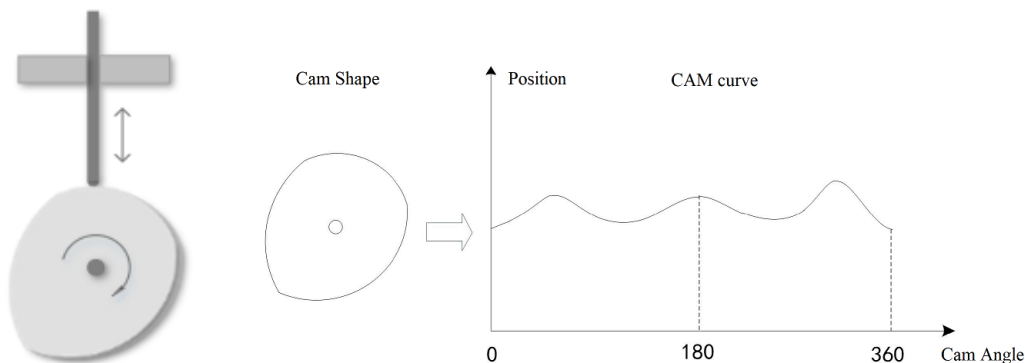


Electronic CAM

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

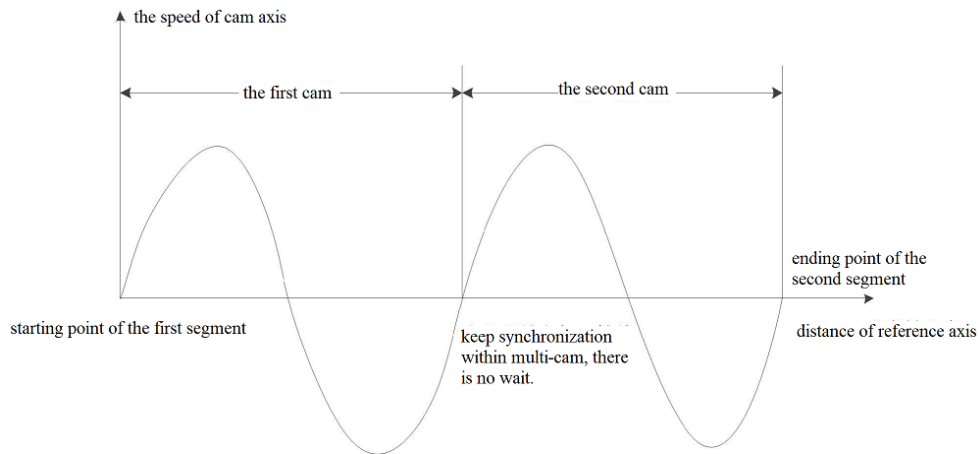
CAMBOX: follow cam watch motion

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.

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.

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 $n \times \text{ratio}$ 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

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.

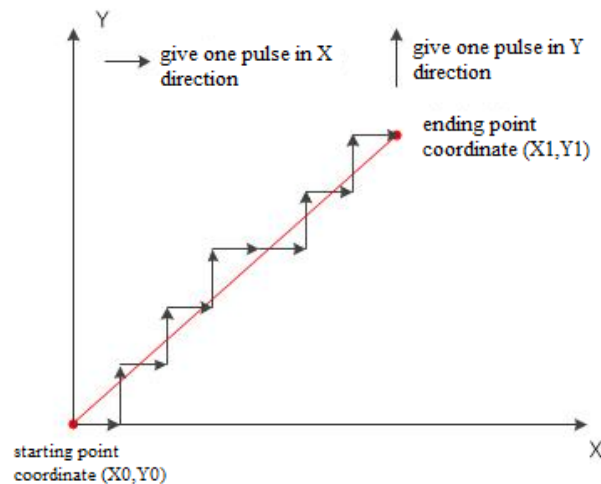
➤ Linear Interpolation:

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 a short distance in Y direction, if the terminal is also below now, continuing to move in Y, it will move a 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.

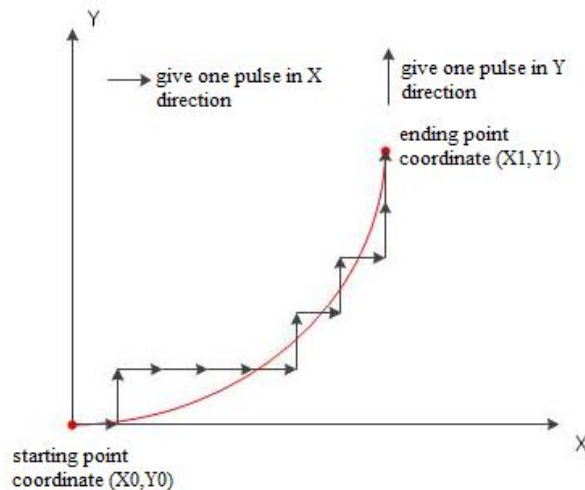
ZMotion controller uses hardware interpolation, and the interpolation precision is within one pulse, so it is still smooth when the path is amplified.

If the axis needs to move from point (X_0, Y_0) to point (X_1, Y_1) 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.

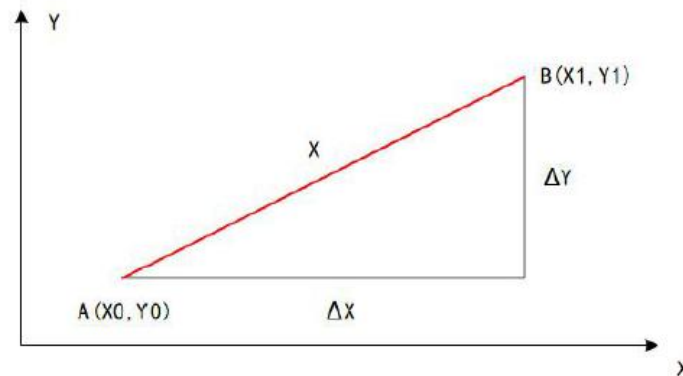
➤ Two-axis linear interpolation

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: $X=[(\Delta X)^2+(\Delta Y)^2]^{1/2}$

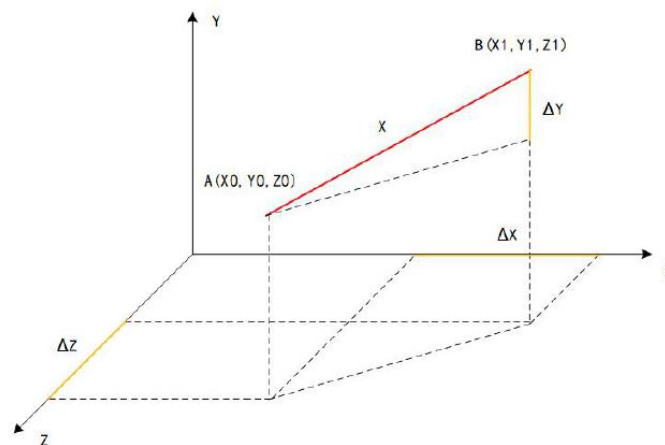
Axis 0 actual speed: $S_0=S \times \Delta X/X$

Axis 1 actual speed: $S_1=S \times \Delta Y/X$



➤ 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:



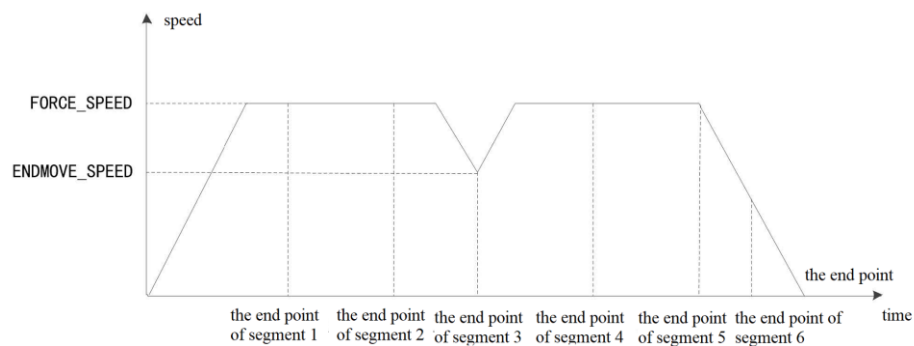
➤ Multi-axis linear interpolation

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.

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 pursuing 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

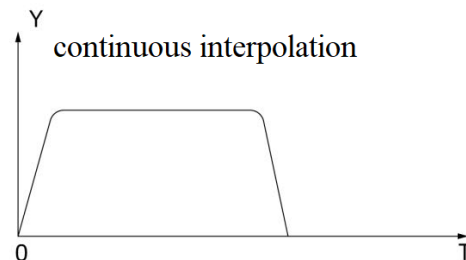
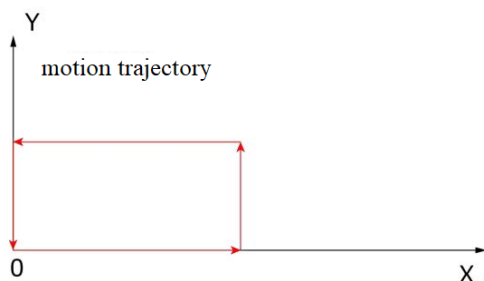
is protected, the machining efficiency is greatly affected, so a look-ahead command is provided to automatically determine whether to reduce the corner speed to a reasonable value at the corner, which will not affect the processing accuracy but also improve the processing speed. This is the role of the 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.

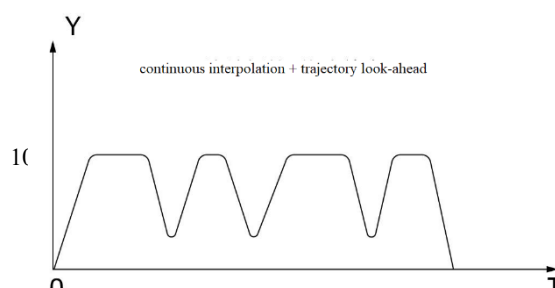
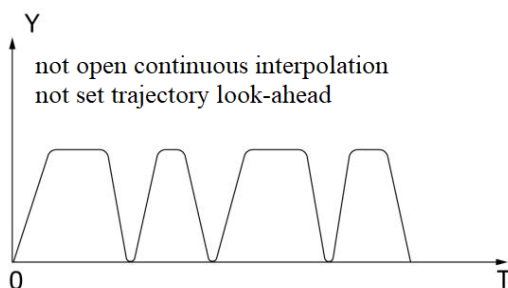
Mode 1: after opening continuous interpolation, obtained the speed of main axis changes with time, please see the right figure. The speed of main axis is consecutive, and it doesn't



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 model1, 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 CORNER_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 Homing

➤ Related Instructions:

Instruction	Description	Usage
DATUM	To select origin homing modes	DATUM (homing mode value)
DATUM_IN	To map origin switch	DATUM_IN = IN No.
FWD_IN	To map position limit +	FWD_IN = IN No.
REV_IN	To map position limit -	REV_IN = IN No.
SPEED	Set the speed of origin finding	Set the speed
CREEP	Set the reverse-crimp speed of origin finding	Set the value of speed
INVERT_IN	To invert the input signal	INVERT_IN = (IN No., ON/OFF)

For high-precision automatic-equipment, they have its own reference coordinate systems. The movement of the workpiece can be defined as the movement on the coordinate system. And the origin of the coordinate system is the motion's starting position. 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 motion controller provides a variety of homing methods, which are set through **DATUM** (set different values to achieve required homing modes). Please note this is the only for one axis, when there is multi-axis, set DATUM for each axis.

When returning to zero, please connect to origin switch (to indicate the origin position), and the position limit (+&-) switch (both are sensors., after the sensor detects a signal, it means 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**;
- the positive and negative limit switches are set by FWD_IN and REV_IN respectively;
- after the limit signal of the controller takes effect, the axis will stop immediately, and the stop deceleration is FASTDEC.
- for ZMC motion controller, 0 means the signal is valid, and when the input is in the OFF state, it means that the origin/limit is reached, but for normally-open type signal, please use INVERT_IN to invert the level.

➤ **DATUM (set homing modes):**

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. DPOS value is reset to 0 while correcting MPOS.
2	The axis runs reversely at CREEP speed until the Z signal appears. It will stop directly when it touches the limit switch. DPOS value is reset to 0 while correcting MPOS.
3	The axis runs forward at SPEED until it touches the home switch. Then, the axis moves inversely at CREEP speed until it leaves the home switch. In process of finding origin, it will stop directly when meeting position limit +. In process of creeping, it will stop directly when meeting position limit -. DPOS value is reset to 0 while correcting MPOS.
4	The axis runs inversely at SPEED until it touches the home switch. Then, the axis runs forward at CREEP speed until it leaves the home switch. In process of finding origin, it will stop directly when meeting position limit -. In process of creeping, it will stop directly when meeting position limit +. DPOS value is reset to 0 while correcting MPOS.
5	The axis runs forward at SPEED until it touches the home switch. Then, the axis runs inversely at CREEP speed until it leaves the home switch. Next, it will continue running inversely at CREEP until it meets the Z signal. It will stop directly when it touches the limit switch. DPOS value is reset to 0 while correcting MPOS.
6	The axis runs inversely at SPEED until it touches the home switch. Then, the axis runs forward at CREEP speed until it leaves the home switch. Next, it will continue running forward at CREEP until it meets the Z signal. It will stop directly when it touches the limit switch. DPOS value is reset to 0 while correcting MPOS.
8	The axis runs forward at SPEED speed until touching the origin switch. It will stop directly when it touches the limit switch.

9	The axis runs reversely at SPEED 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 homing mode (6098h). The default value of 0 means “using 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: 6060h PDO (switch the current mode) → 6098h SDO (set homing mode) → 6099h SDO (set speed) → 609Ah SDO (set acceleration) → PDO (switch the control word to start homing).

Z signal homing must be configured with Z signal ATYPE.

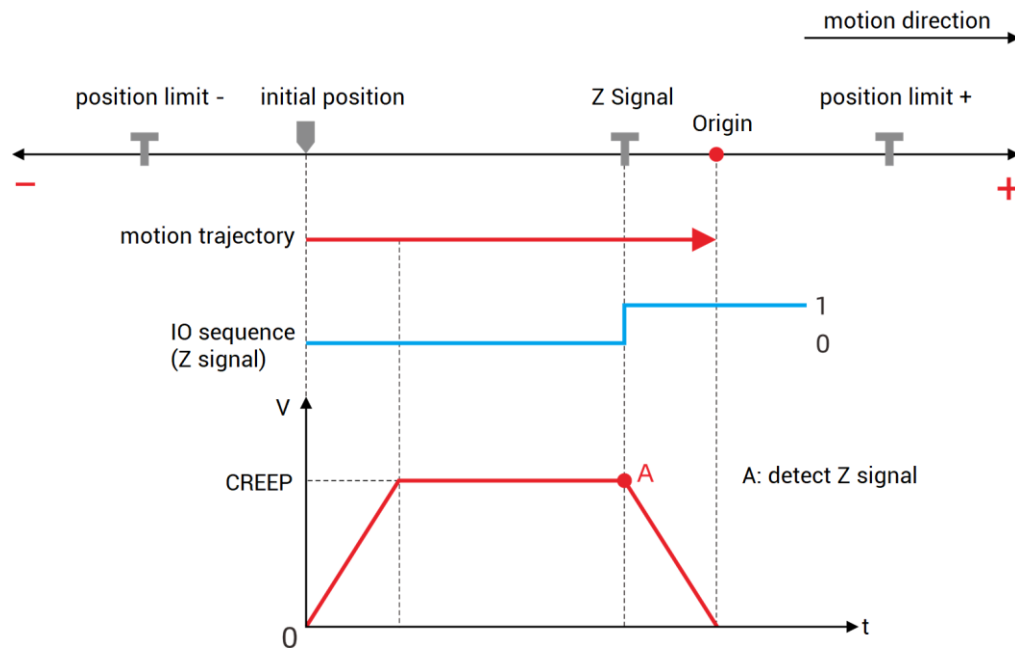
For the case the origin is in the middle of the positive and negative limits, add 10 to each mode, which means that the movement will not be canceled if the limit is encountered during the zero-return process, but will continue to search for the origin in the reverse direction. Other conditions are the 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 Homing Methods:

1) Mode 1

Like below shown, use DATUM(1) command to make the axis run forward at CREEP speed, it will decelerate and stop after meeting the Z signal. And the position when stopped is the origin, the DPOS value is reset to 0 at this time. While homing, it will stop immediately if it touches the position limit.

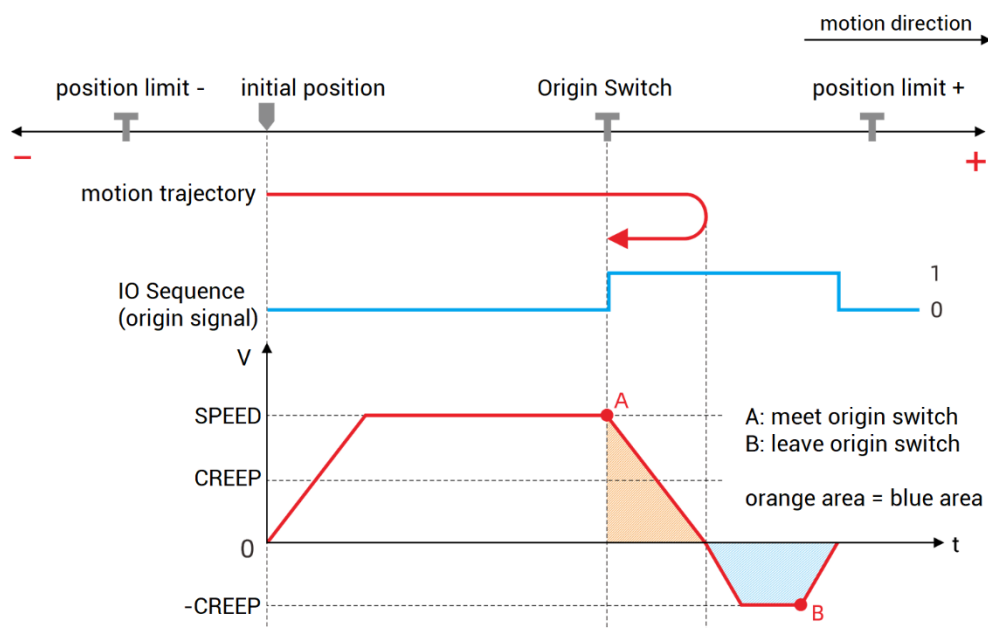


2) Mode 2

Use DATUM (2), same principle as mode 1, but the motion direction is opposite.

3) Mode 3

Like below shown, use DATUM(3) command to make the axis run forward at SPEED speed, it will decelerate after meeting origin switch. When it is stopped (decelerate to 0), it will move inversely at CREEP, until it left the origin switch. After axis is stopped, reset the DPOS value to 0, and now the position when stopped is the origin. While homing, it will stop immediately if it touches the position limit.

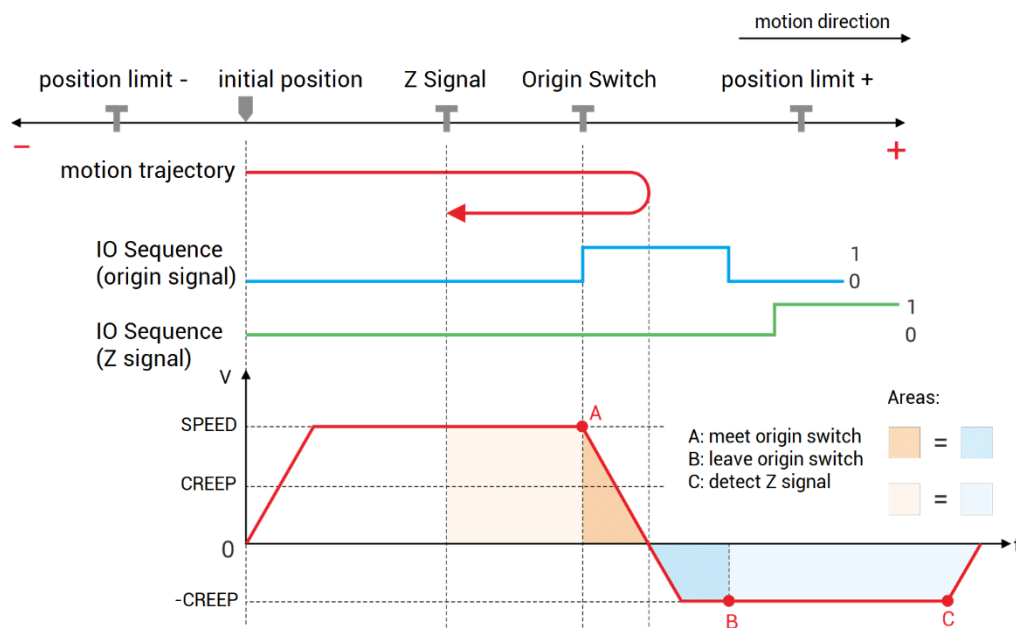


4) Mode 4

Use DATUM (4), same principle as mode 3, but the motion direction is opposite.

5) Mode 5

Like below shown, use DATUM(5) command to make the axis run forward at SPEED speed, it will decelerate after meeting origin switch. When it is stopped (decelerate to 0), it will move inversely at CREEP, until it meets the Z signal. After axis is stopped, reset the DPOS value to 0, and now the position when stopped is the origin. While homing, it will stop immediately if it touches the position limit.

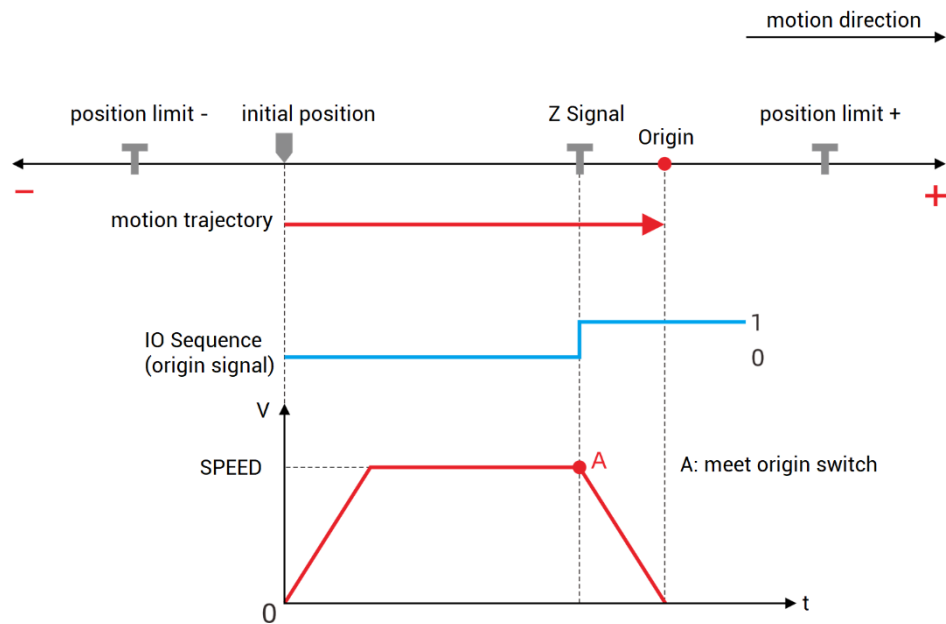


6) Mode 6

Use DATUM (6), same principle as mode 5, but the motion direction is opposite.

7) Mode 8

Like below shown, use DATUM(8) command to make the axis run forward at SPEED speed, it will decelerate and stop after meeting the origin switch. And the position when stopped is the origin, the DPOS value is reset to 0 at this time. While homing, it will stop immediately if it touches the position limit.



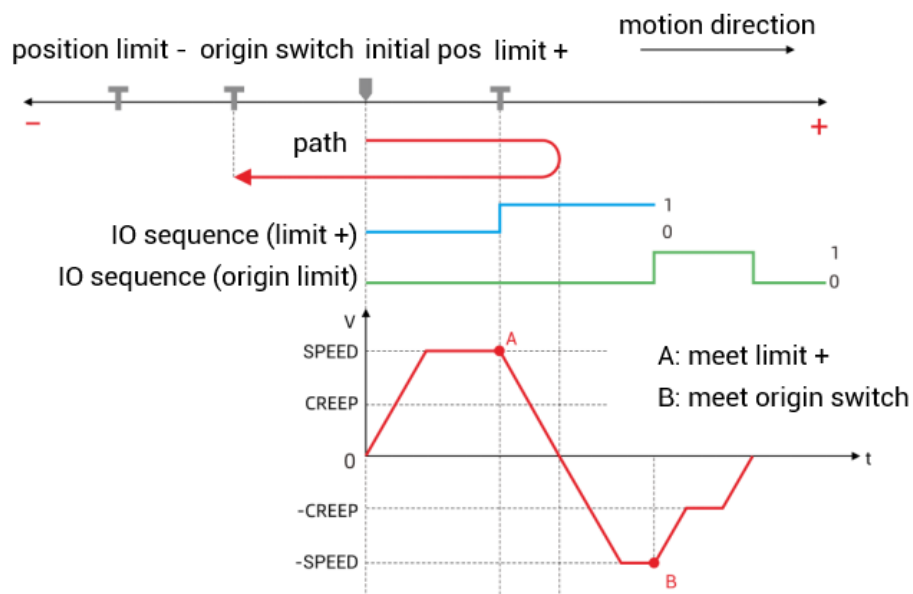
8) Mode 9

Use DATUM (9), same principle as mode 8, but the motion direction is opposite.

9) Mode 13

The axis runs forward at SPEED speed, it will not alarm and stop after meeting the position limit, but it will move inversely at SPEED, after meeting the origin switch, it will decelerate to CREEP, until leaving the switch. Homing is completed; position is cleared.

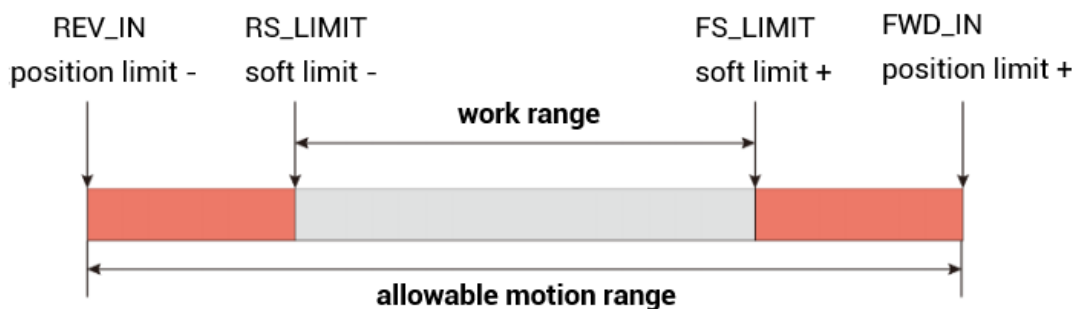
If it meets the origin signal at first, same as mode 3.



5.5. Position Limit Related Instructions

Instruction	Description	Usage
FS_LIMIT	Positive software limit setting	FS_LIMIT = positive limit position
RS_LIMIT	Negative software limit setting	RS_LIMIT = negative 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 is 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 v_0 when it stops, and the fast deceleration FASTDEC is a . The calculation formula: $v_2 - v_0 = 2as$, 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 "SOFTLIMIT 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 "SOFTLIMIT RS", *DPOS
RS_LIMIT = -200000000    'close software limit position
FS_LIMIT = 200000000
END

```

Print result:

```

Axis:0 AXISSTATUS:200h,FSOFT
SOFTLIMIT FS 101
Axis:0 AXISSTATUS:400h,RSOFT
SOFTLIMIT 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

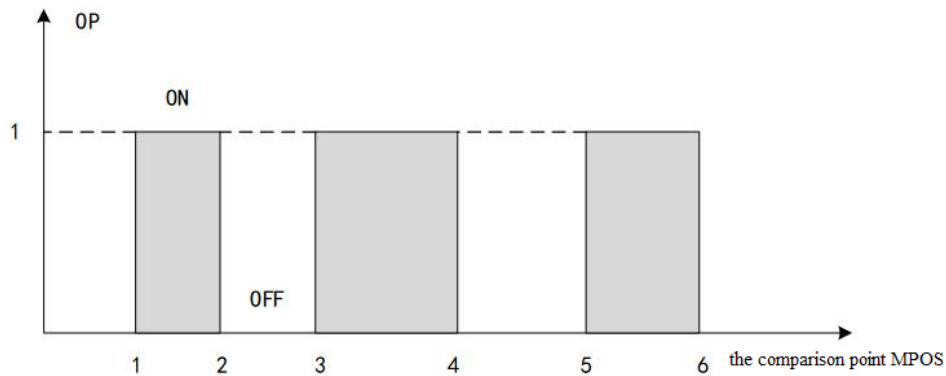
Related instructions:

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:

Instructions	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.

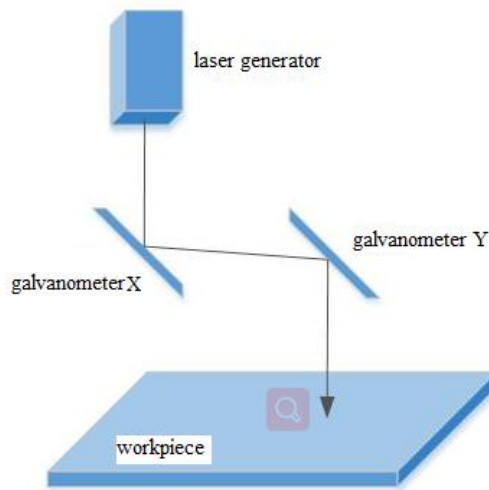
When there is no encoder feedback, the precision output function automatically uses the command position DPOS to compare the trigger. The motor always has a certain following error (following error = DPOS-MPOS). When the encoder feedback is used, the encoder feedback MPOS trigger will be more accurate. 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

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.

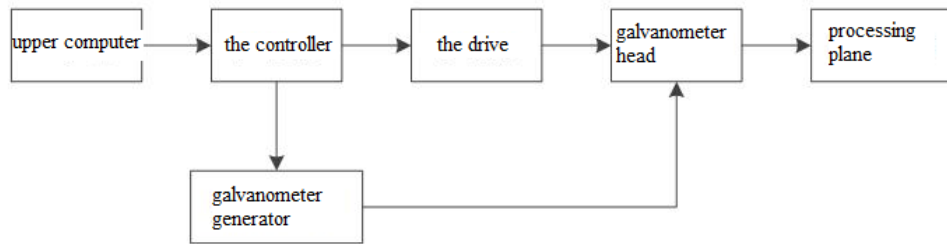


There are two basic movements of laser galvanometer movement: jump movement and the marking movement.

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.

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.

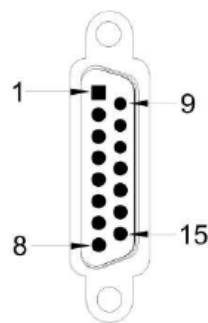
It is convenient to work and has strong anti-counterfeiting ability.

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.



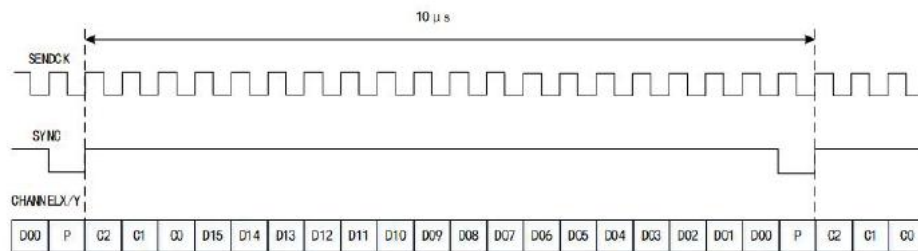
Pin No.	Signal	Description
1	CLOCK-	Clock signal -
2	SYNC-	Synchronization signal -
3	X channel-	Galvanometer X channel signal -
4	Y channel-	Galvanometer Y channel signal -
5	NC	Reserve
6	STATUS	Galvanometer status signal -

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.

Galvanometer Application Processes

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)

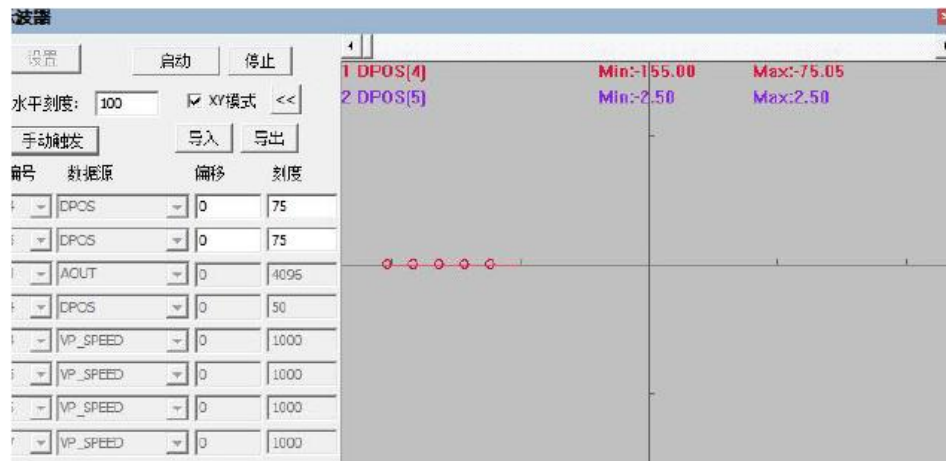
```

        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

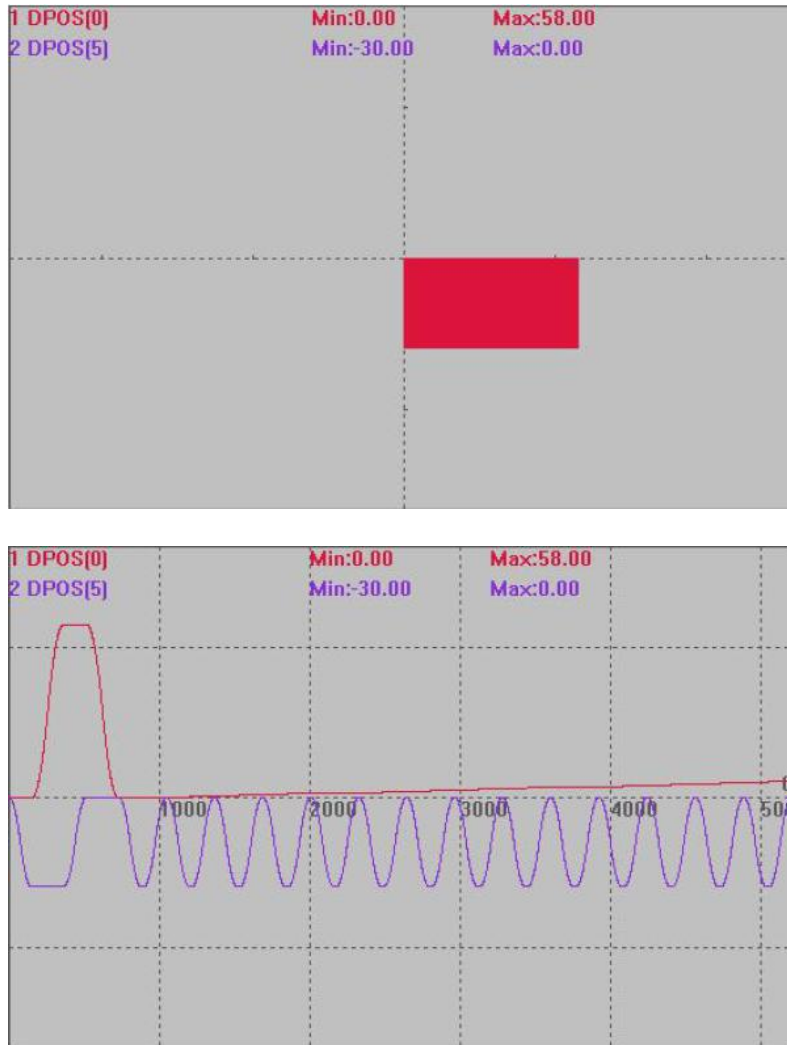
        MOBE_OP(0,ON)
        TRIGGER
        MOVE(0,-30)
        WAITIDLE
        MOVE(58,0)
        WAITIDLE
        MOVE(0,30)
        WAITIDLE
        MOVE(-58,0)
        WAITIDLE

        '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".

The 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.

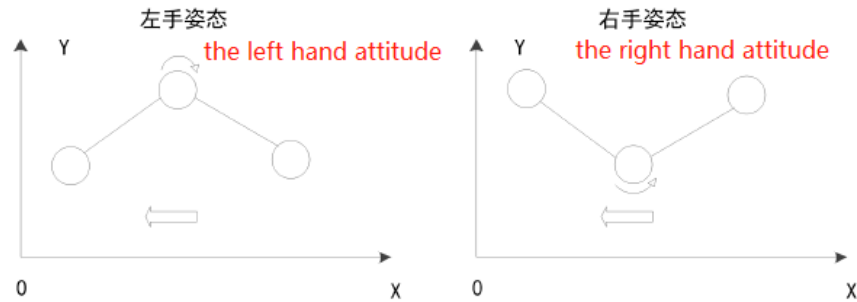
Coordinate system transformation refers to the transformation from the original coordinate system to another coordinate system when describing the same space. In the use of the manipulator, it is often used to determine the coordinate system of the workpiece.

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 several workpiece coordinate systems 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. Posture

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.

Forward & Inverse Kinematics

The establishment of the manipulator is set by the [CONNREFRAME](#) (positive solution) instruction and the [CONNFRAME](#) (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.

Valid Robot Functions

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.

Robot Examples

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 ml	'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 °

```

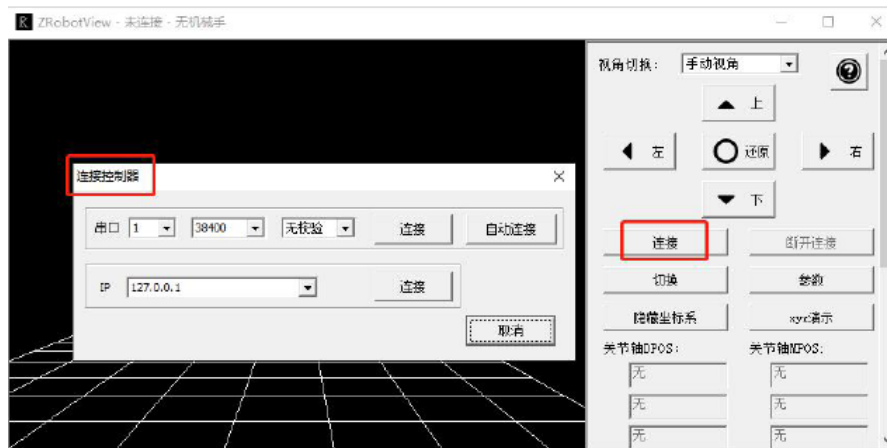
```

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
WNEED
END

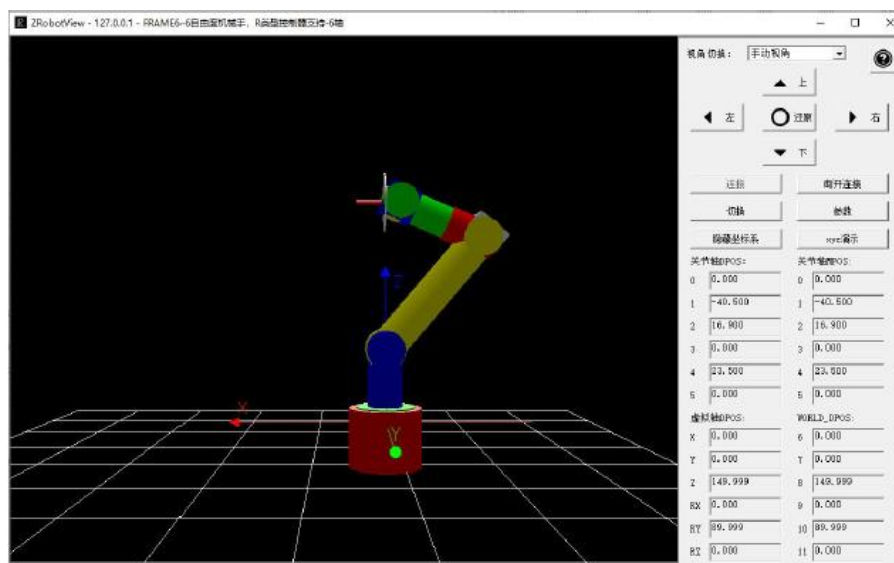
```

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 RTSys 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 6 “Axis” Introductions

6.1. 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 No.

1. Pulse Axis No.

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).

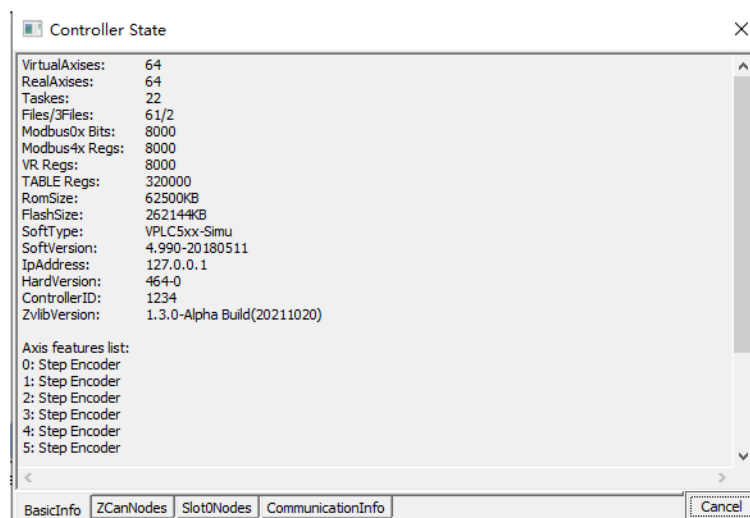


Take the following controller status as an example. For what type of axis each pulse axis interface supports, please refer to the description of the Axis features list in the "State the Controller" window. Step is pulse output, and Encoder is encoder feedback.

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 No.

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_STEP](#) 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.

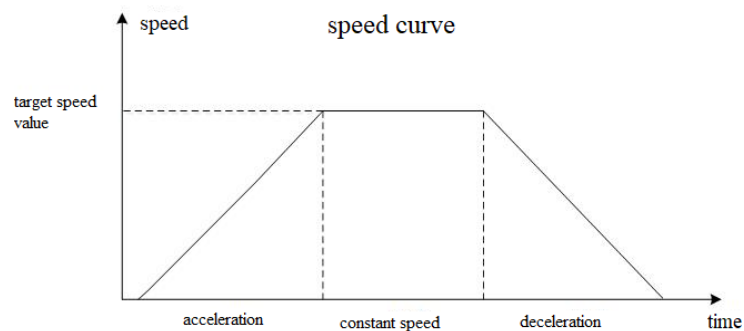
The IDLE command is used to judge whether the motion command added to the axis is completed. It returns 0 during motion and -1 when the motion ends. Generally, the WAIT IDLE (axis number) statement is used in the program to judge the state of the axis.

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.

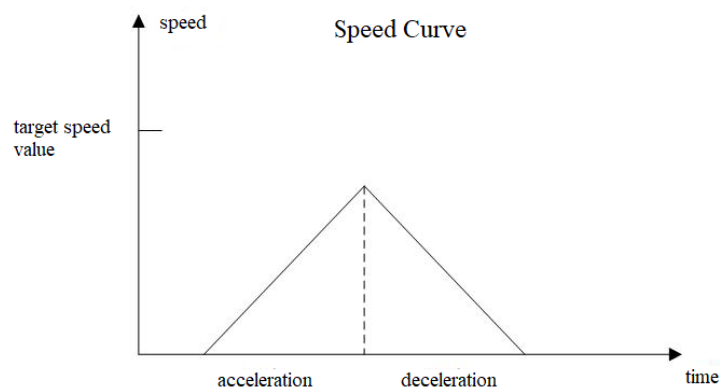
6.4. Axis Speed

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
```

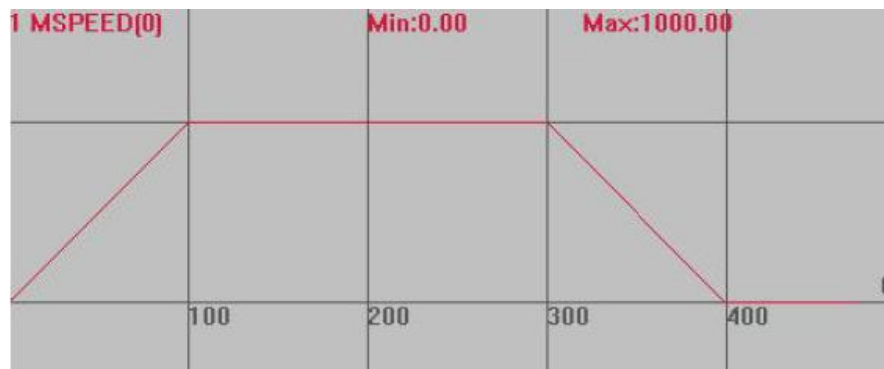
```
DECEL = 10000
```

```
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

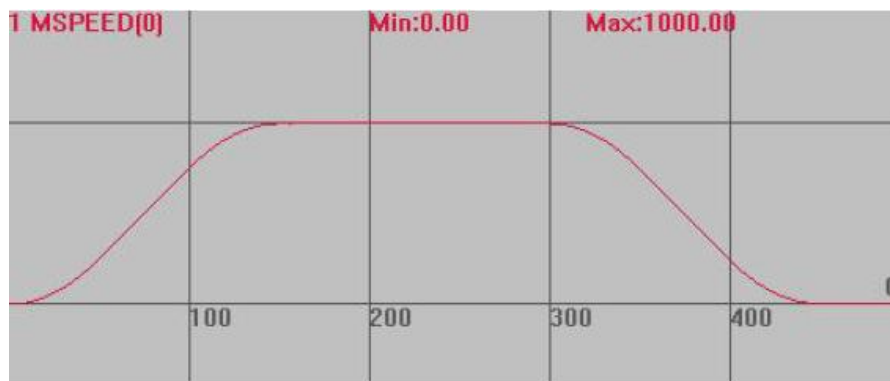
DECEL = 10000

SRAMP=50

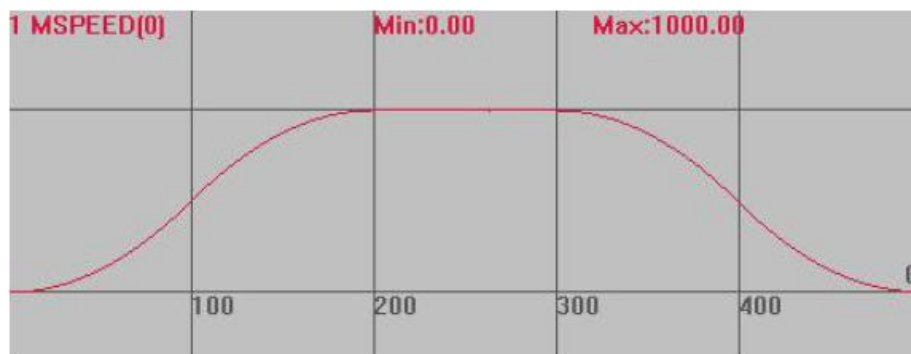
TRIGGER

MOVE(300)

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

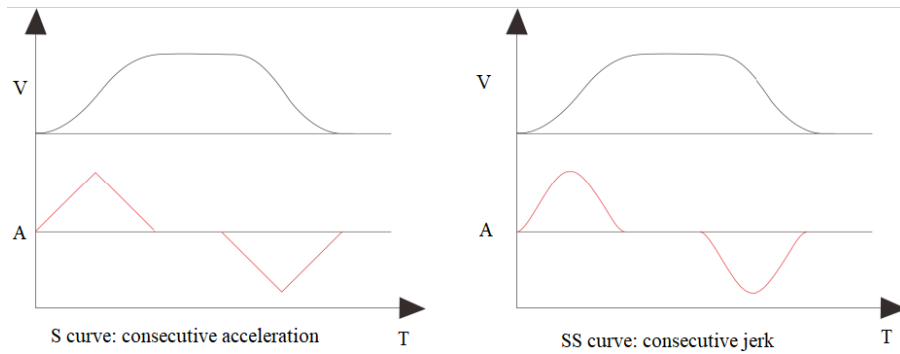


When SRAMP=100, obtain the below S curve: acceleration and deceleration process become longer.



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.



Routine: compare S curve with SS curve

```

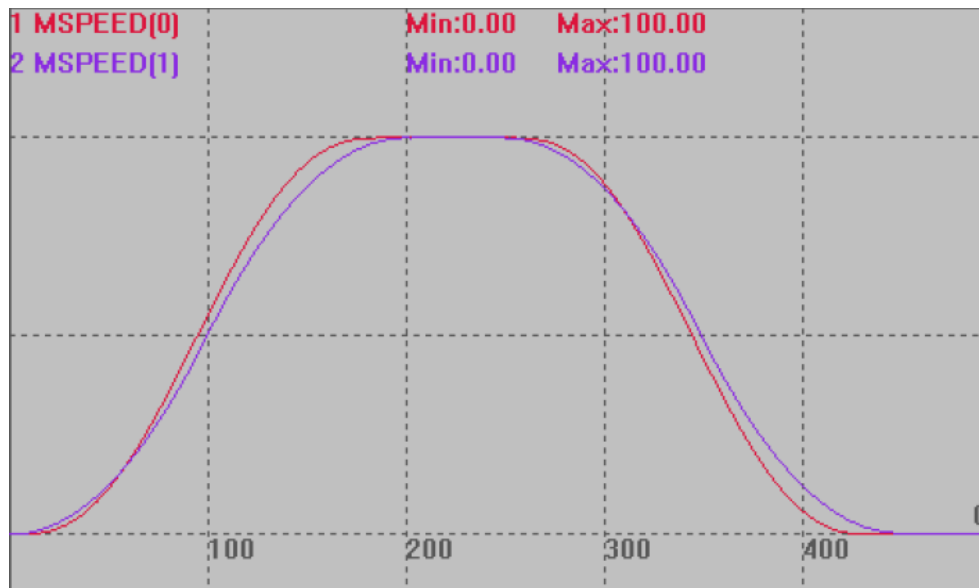
BASE(0,1)
ATYPE=1,1
UNITS=100,100
DPOS=0,0
MPOS=0,0
SPEED=100,100
ACCEL=1000,1000
DECEL=1000,1000
SRAMP=100,100
VP_MODE=7,0
TRIGGER
MOVE(25)  AXIS(0)
MOVE(25)  AXIS(1)
END

```

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.



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

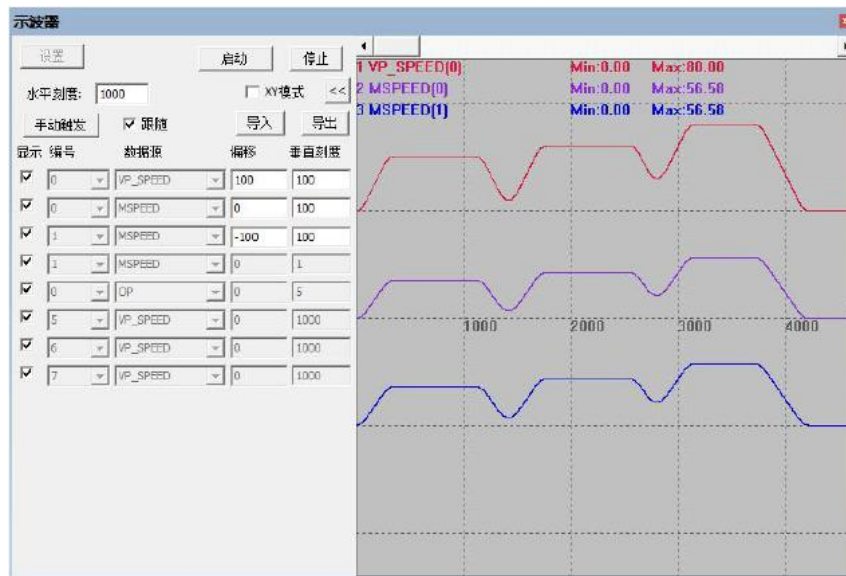
```
DECEL = 200,200
SRAMP=100,100          'S curve
MERGE= ON              'start continuous interpolation
TRIGGER

'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 No.](#) 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 Syntax 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:

Axis 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 unsuccessfully, 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 unsuccessfully, galvanometer warning 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.
26	Customized encoder : use C language to update encoder position, closed-loop is supported ; Valid in firmware after version_build 240702.
48	SSI absolute encoder
49	BISS absolute encoder
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.

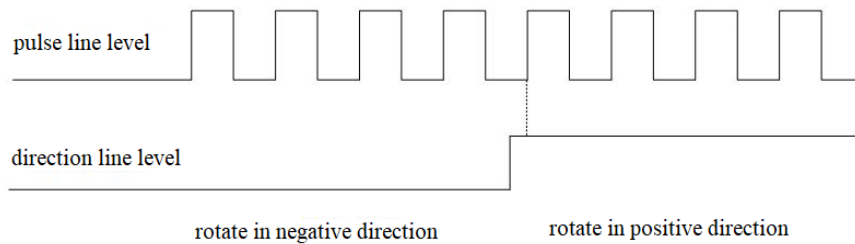
There are 3 modes of controller pulse output: pulse + direction, dual-pulse, orthogonal pulse. They are configured through INVERT_STEP instruction, the default is pulse + direction mode.

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.

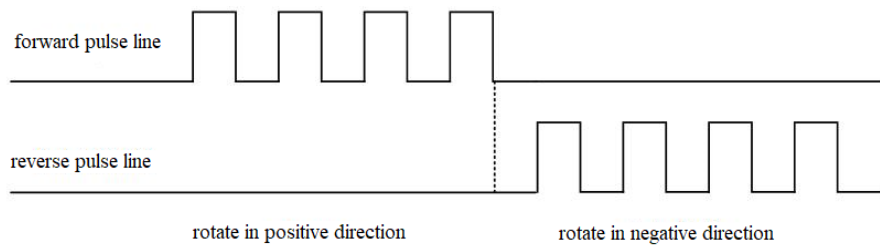
DIR+, DIR- output direction signal, different levels of this signal are relative to different

rotation direction. This mode occupied the most in drive.



2) CW/CCW: dual-pulse work mode

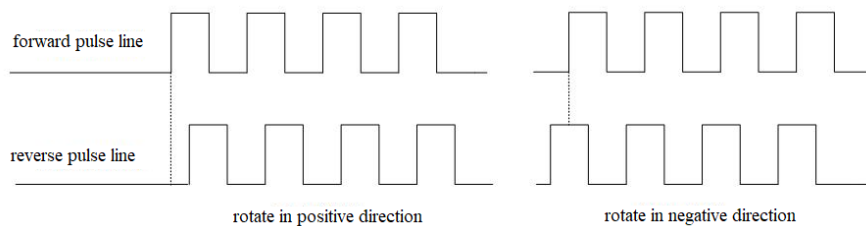
Two lines both output pulse signal, CW means output pulse signal in positive direction, CCW means output pulse signal in negative direction. Usually, they are differential output, the phase difference angle between the two signals is determined by the phase lead or lag.



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.

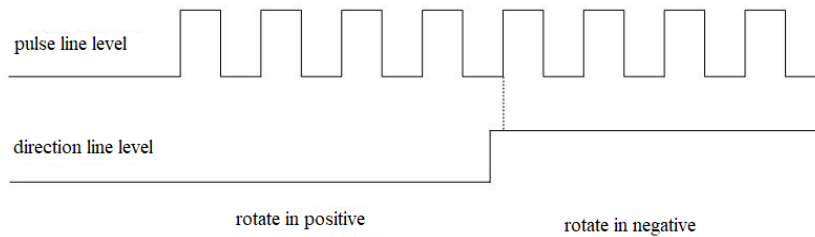
The function of counting or encoding is achieved by the phase difference between the two pulses.



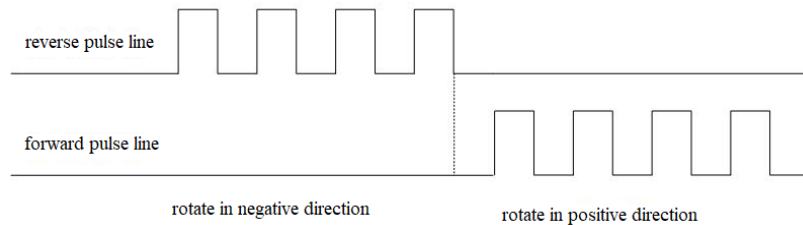
Polarity reversal

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



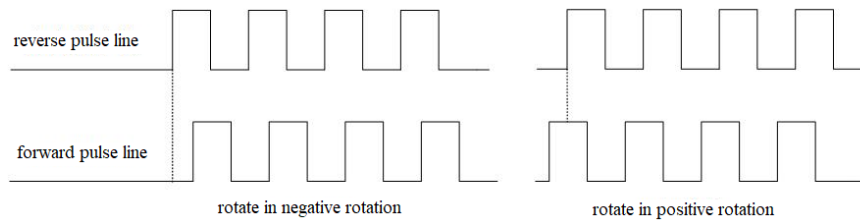
2) CW/CCW: dual-pulse work mode



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 7 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 supports up to 4096 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. After superposition, don't use absolute motion commands for the superposed axis, otherwise, it may cause wrong position. When using ADDAX to realize superposition, the added value is not units but the pulses. <u>Conversion relationship:</u> Distance of superimposing axis *units of superimposing axis /units of superimposed axis = distance of superimposed axis <u>For example:</u> If UNITS of axis A is 100, of Axis B is 50, make their superposition move 100. <i>Situation 1:</i> add motion of axis A to axis B, no axis A moves 100, then the axis B moves $100 \times 100 / 50 = 200$ <i>Situation 2:</i> add motion of axis B to axis A, now axis B moves 100, then the axis A moves $100 \times 50 / 100 = 50$. Motions can't be added to each other simultaneously between 2 axes, that is, when adding motion of axis A to axis B, B can't be added to axis A any more. ● 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 superimposing 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. For the motion controller that is with slow running speed, it needs to delay one cycle after axis superposition, which is used to avoid precision error.
Syntax	<u>Make superposition:</u> ADDAX (superposing axis No.) AXIS (superposed axis No.) <u>Cancel superposition:</u> ADDAX(-1) AXIS (superposed axis No.) <i>This superposition is added in controllers above 4xx series with 20220708 firmware version or above.</i> ADDAX(srcaxis ,[imode], [para]) destaxis: superposing target axis No. srcaxis: superposing source axis' No. imode: superposition mode 0: default value, single-axis superposition, it is compatible with "previous superposition of direct pulses adding". 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 / ratioout 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 respectively by ADDAX(-1, 3, 0) or ADDAX(-1) AXIS (the superposed axis No.) 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 inversely superposed at destaxis, srcaxis2 is inversely superposed at destaxis2. Note: two axes should be cancelled together, ADDAX(-1, 5, -1) or ADDAX(-1) AXIS (superposed axis No.) 6: SCAN axes superposition, 2 axes must be galvanometer axes. When the entire scan is superposed, please call XY separately. BASE(destaxis) 'scan axis ADDAX(srcaxis, 6) 'scan axis 2
Controller	General

Example**Example 1:**

```

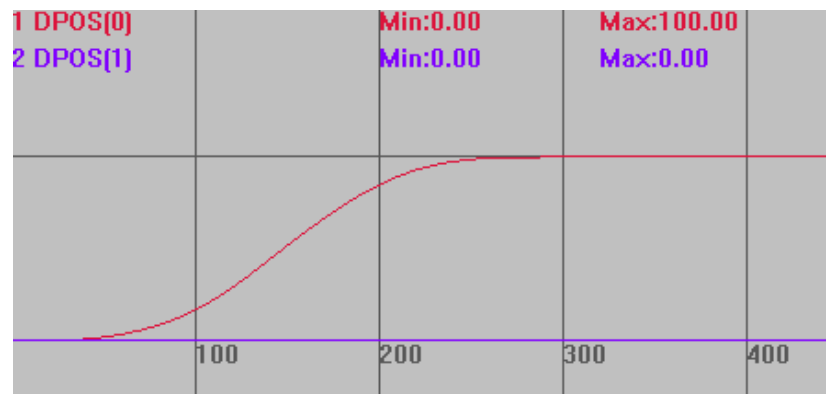
BASE(0,1)
ATYPE=1,1
UNITS=100,200      'set UNITS of axis 0 as 100, and axis 1 as 200
SPEED=1000,1000    'set speed as 1000
ACCEL=10000,10000  'set acceleration as 10000
DECEL=10000,10000  'set deceleration as 10000
ADDAX(0) AXIS(1)  'add motion of axis 0 to axis, superpose according to
                    'the number of pulse
DELAY(1)           'set delay time according to controller cycle
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

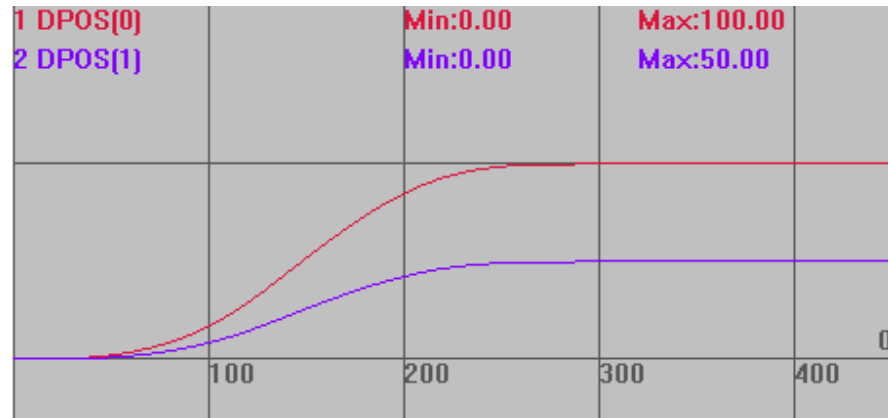
DPOS(1) vertical scale 100



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)

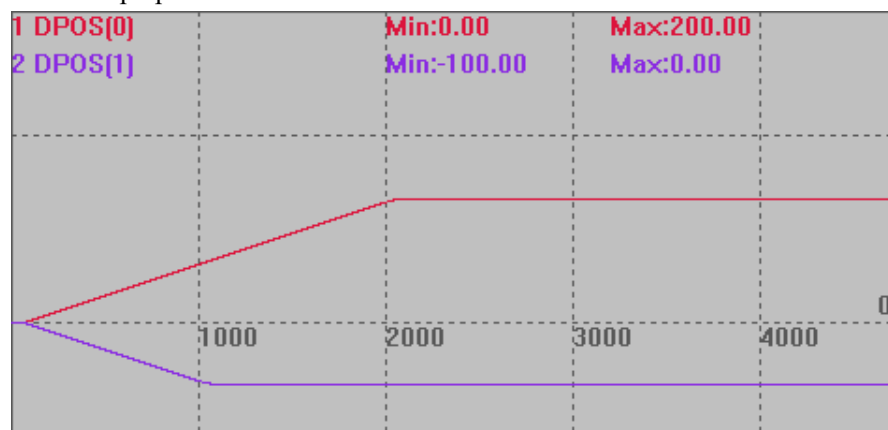
WAIT IDLE

'wait running ends

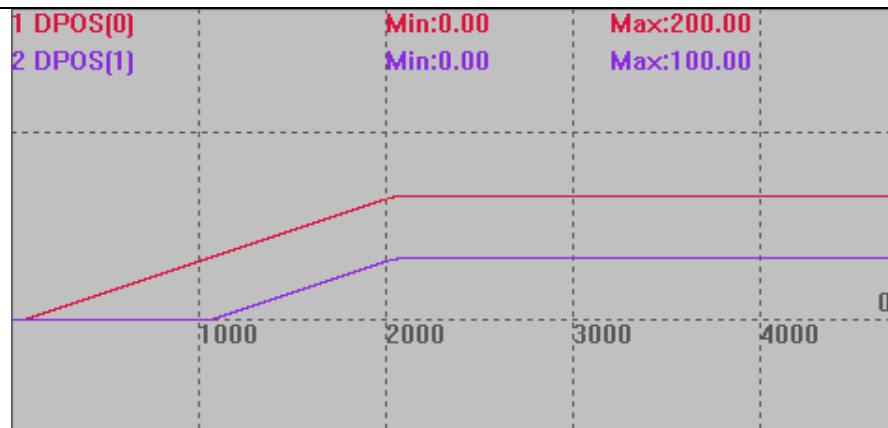
ADDAX(-1) AXIS(1)

'cancel superposition

Before superposition:



After superposition: 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

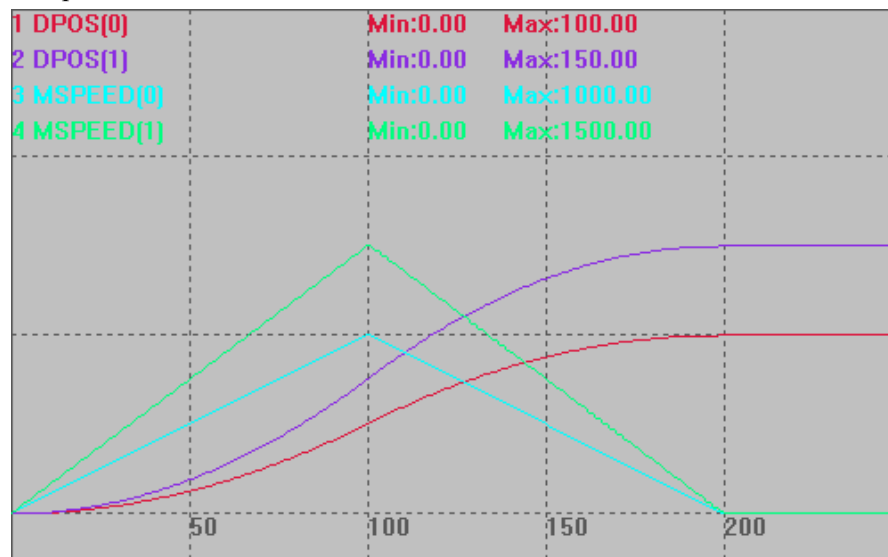
MOVE(100) AXIS(0)

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

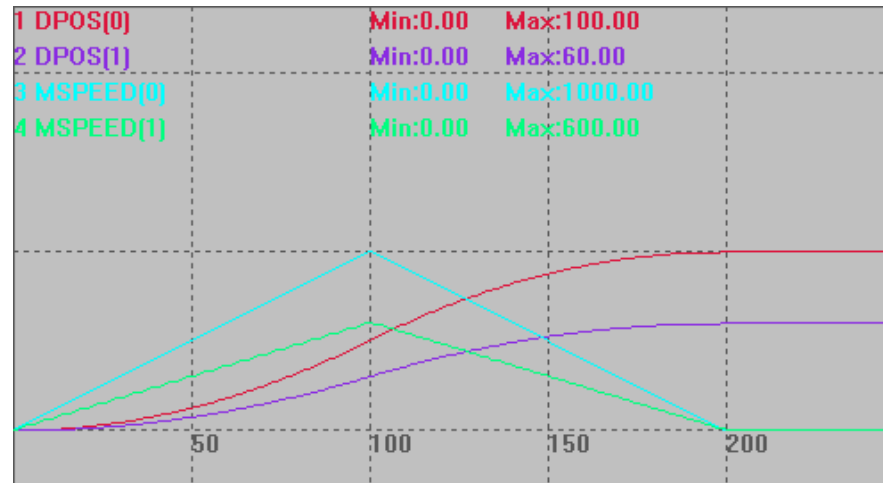
BASE(1) 'select superposed axis

```

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.

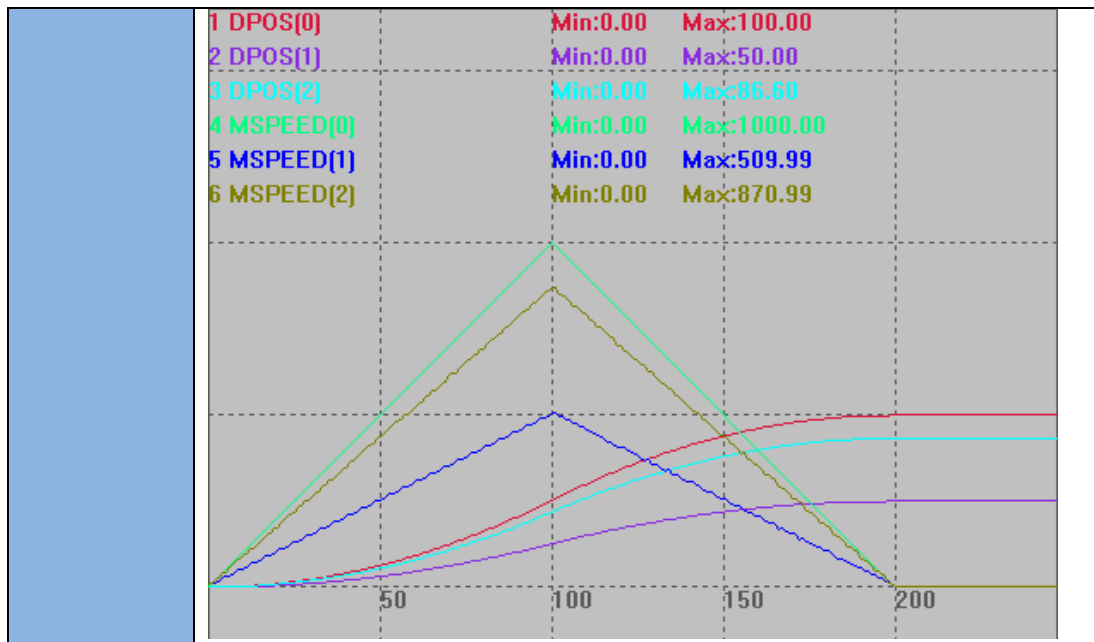


Example 5: mode 3

```

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
DELAY            'set delay time according to controller cycle
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 -- Single-Axis / Axis Group Stop

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 CANCEL 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.										
Syntax	CANCEL (mode) Mode: mode selection <table border="1"> <tbody> <tr> <td>0(default)</td><td>Cancel the motion in process</td></tr> <tr> <td>1</td><td>Cancel the motion in buffer</td></tr> <tr> <td>2</td><td>Cancel motions in process and in buffer, stop speed refers 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 refers to deceleration “DECEL”.</td></tr> </tbody> </table> 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.	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”.
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”.										
Controller	General										

Example**Example 1: mode = 0**

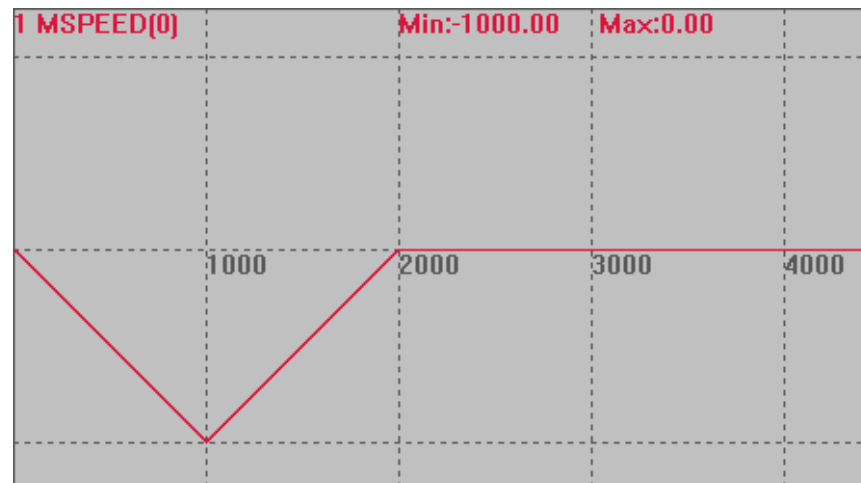
```

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**

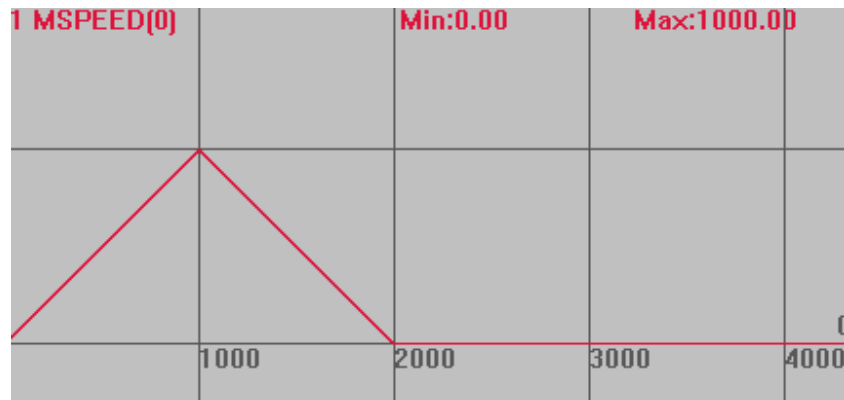
```

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.

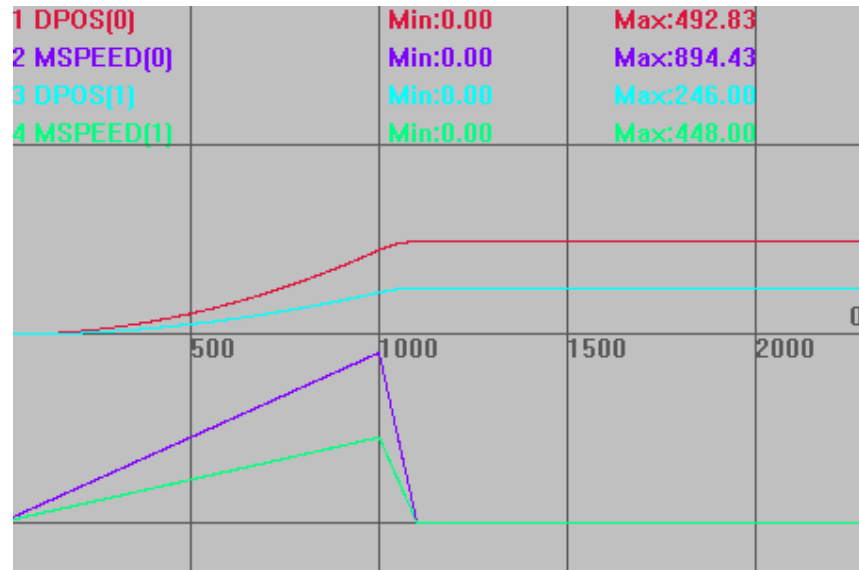
Motion trajectory and speed curve:

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

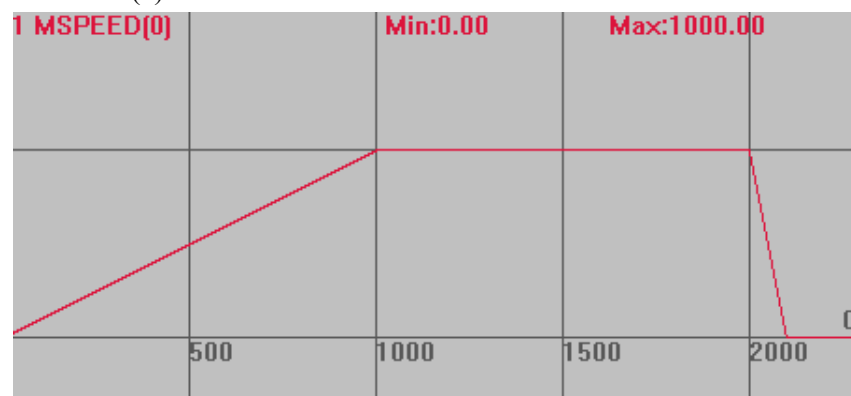
DELAY(2000)

'delay 2 seconds

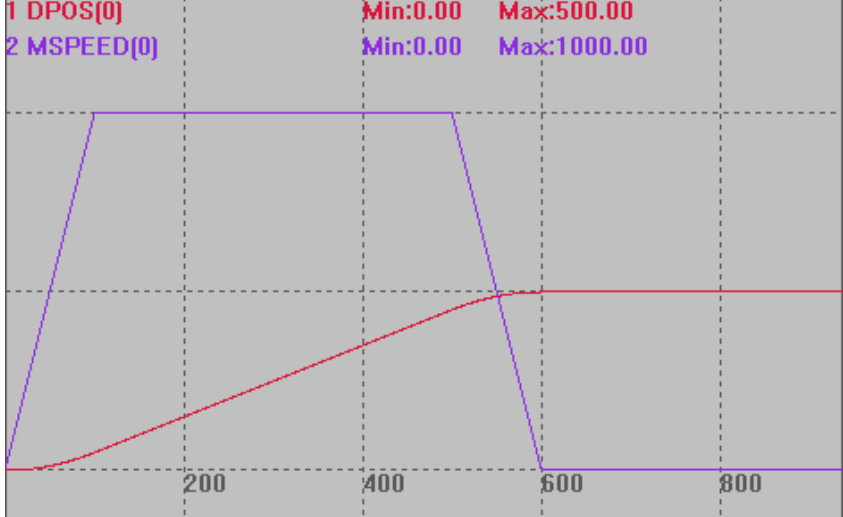
CANCEL(3) 'now directly stop sending pulse, axis stops immediately

Motion Trajectory

MSPEED (0) vertical scale 1000

**Example 5: mode = 4**

BASE(0)

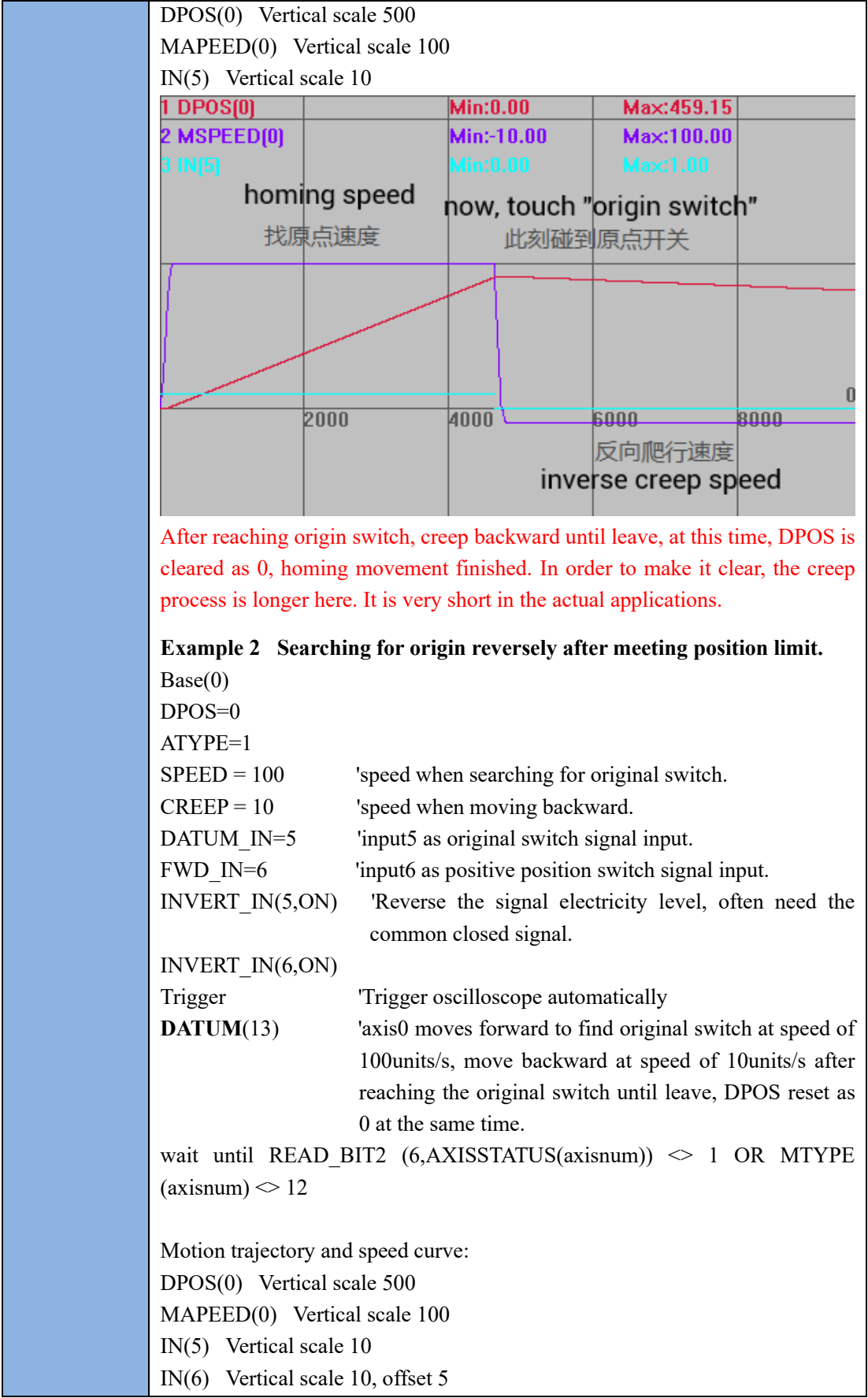
	DPOS=0 ATYPE=1 UNITS=100 SPEED=1000 ACCEL=10000 DECEL=10000 'set deceleration as 10000 FASTDEC=10000 'set fast deceleration as 100000 TRIGGER 'trigger oscilloscope automatically MOVE(1000) 'the current motion DELAY(-2000) 'the motion in buffer DELAY (500) CANCEL(4) 'emergency stop, deceleration is 10000 Motion Trajectory DPOS (0) vertical scale 500, no offset MSPEED (0) vertical scale 500, no offset 
Instructions	RAPIDSTOP , DECEL , FASTDEC

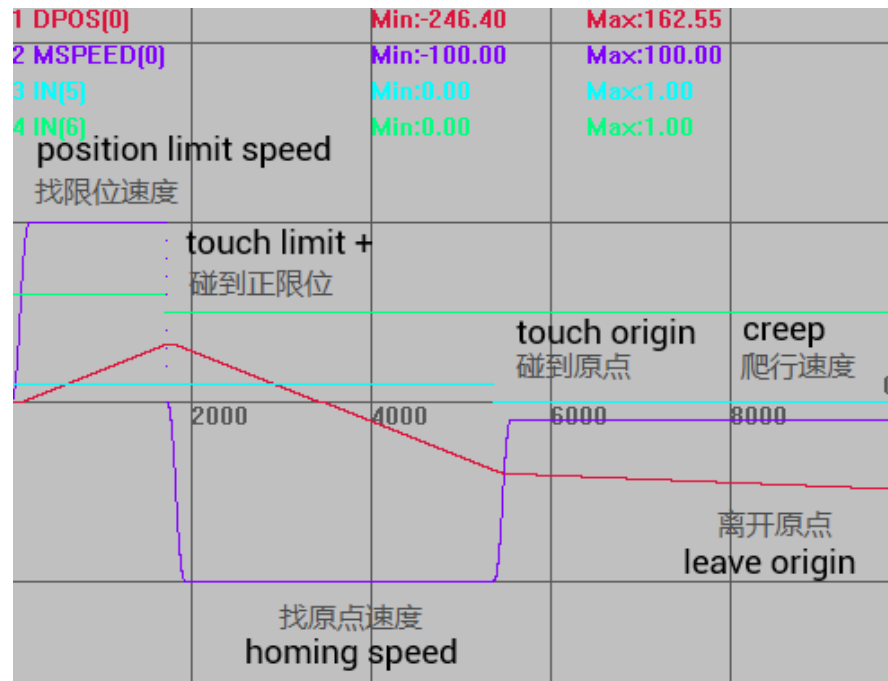
DATUM -- Homing

Type	Single Axis Motion Instruction
Description	Origin (home position or zero position) finding movement of single axis. Origin switch is set by DATUM_IN, positive and negative position limit switches are set by FWD_IN and REV_IN respectively. ➤ For ZMC Controller It is valid when it is 0, when IN is OFF, which means it arrived origin / position limit. It is always ON type signal, using INVERT_IN to invert the electric level. ➤ For ECI Controller It is valid when it is 1, when IN is ON, which means it arrived origin /

	position limit. It is always OFF type signal, using INVERT_IN to invert the electric level. For Z signal homing, please configure ATYPE that is with Z signal, ATYPE = 4 / 7. When LSPEED is configured, it will stop emergency when found the origin, and the position when decelerated to LSPEED is the origin position. When there are several axes for homing, 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 manually. Absolute commands and MOVER commands can't be added behind DATUM. For drive homing mode, at this time, no feedback from DATUM command. Please use drive state returned value of drive itself homing.												
Syntax	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 + 100 (mode 100+n and 110+n corresponds n and 10+n), indicating the relevant MPOS will be cleared automatically after encoder connected (only valid in ZMC4XX controllers). DATUM (0) AXIS (Axis No.) to clear assigned axis' error state. <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Clear error states of all axes.</td></tr> <tr> <td>1</td><td>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.</td></tr> <tr> <td>2</td><td>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.</td></tr> <tr> <td>3</td><td>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.</td></tr> <tr> <td>4</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. 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.</td></tr> </tbody> </table>	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.
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 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.
	6	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.
	8	Axis runs forward at speed of SPEED, until meeting origin switch, it will stop immediately when met limit switch.
	9	Axis runs reverse at speed of CREEP, until meeting origin switch, it will stop immediately when met limit switch.
	21	Use EtherCAT drive homing function, now mode 2 is valid. Set drive homing mode (6098h), default 0 = drive current homing mode is used. Using axis SPEED, CREEP, ACCEL and DECEL to multiple UNITS, then automatically set drive 6099h and 609Ah. <u>Action sequence:</u> 6060h, switch PDO current mode – 6098 homing mode – 6099 speed – 609A acceleration – 6060 switch to current mode.
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.		
Controller	General	
Example	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:	





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.

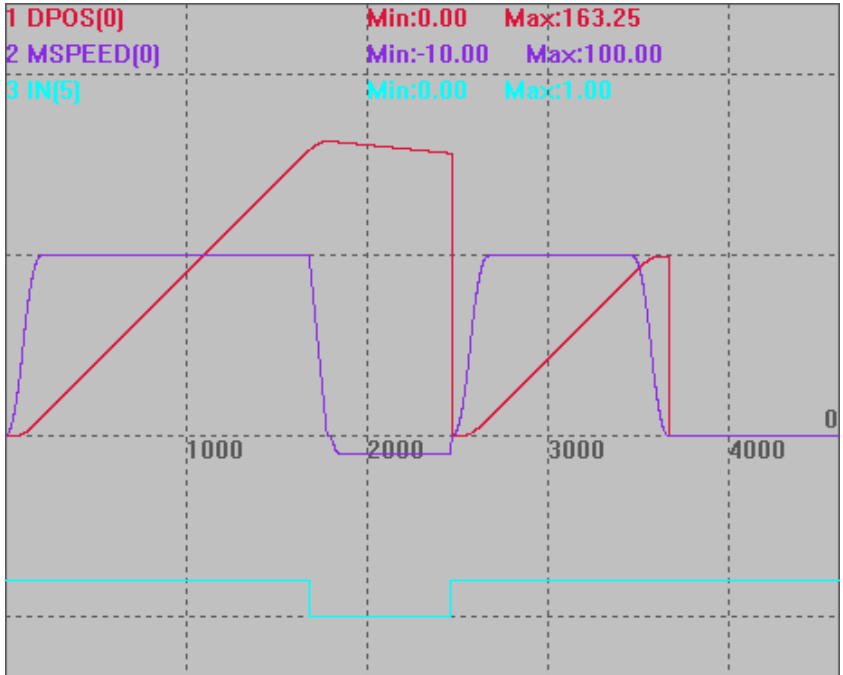
Description of bit 13, bit 12, bit 10:

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 <table><tr><td rowspan="12">Initialization mode</td><td>11h</td><td>Z Phase</td></tr><tr><td>12h</td><td>HOME ↑ *2</td></tr><tr><td>13h</td><td>HOME ↓ *3</td></tr><tr><td>14h</td><td>POT ↑ *2</td></tr><tr><td>15h</td><td>POT ↓ *3</td></tr><tr><td>16h</td><td>NOT ↑ *2</td></tr><tr><td>17h</td><td>NOT ↓ *3</td></tr><tr><td>18h</td><td>EXIT1 ↑ *2</td></tr><tr><td>19h</td><td>EXIT1 ↓ *3</td></tr><tr><td>1Ah</td><td>EXIT2 ↑ *2</td></tr><tr><td>1Bh</td><td>EXIT2 ↓ *3</td></tr><tr><td>1Ch</td><td>EXIT3 ↑ *2</td></tr><tr><td>1Dh</td><td>EXIT3 ↓ *3</td></tr></table>	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				
		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																														

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.
Syntax	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 'reverses 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. Note: VMOVE command can't do hardware position comparison output with multi-axis vector mode.
Syntax	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 0 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 10, no offset
Instructions	FORWARD , REVERSE

FORWARD -- Forward Movement

Type	Single Axis Motion Instruction
Description	BASE selects axis to move forward. REVERSE is switched after CANCEL.
Syntax	Forward [axis(axis No.)]
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.
Syntax	reverse [axis(axis No.)]
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 -- Motion Position Modification

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 MOVEMODIFY will interrupt the continuity of motion speed. When MOVEMODIFY is used in multi-axis, the motion may not be linear interpolation movement. If MOVEMODIFY and MOVEMODIFY2 commands are used in 32-bit controller, not recommend set big motion parameters, otherwise, it may cause “abnormal changes”. If you need big parameters, try to set one gear ratio, or use 64-bit controller.
Syntax	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

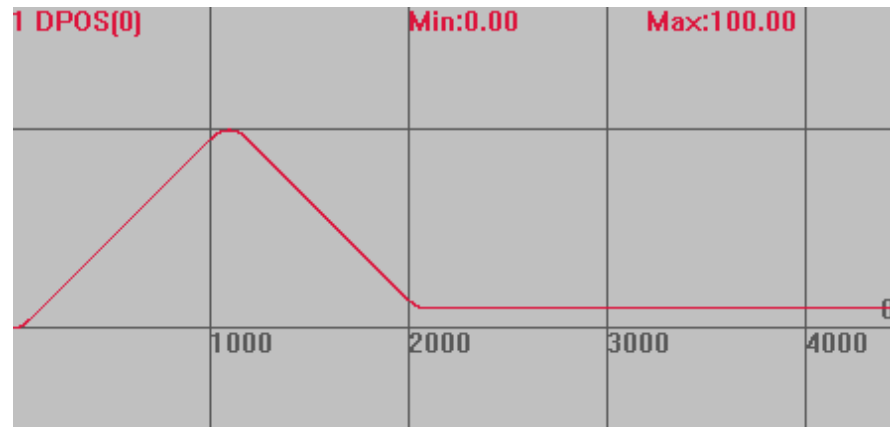

```

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:

```

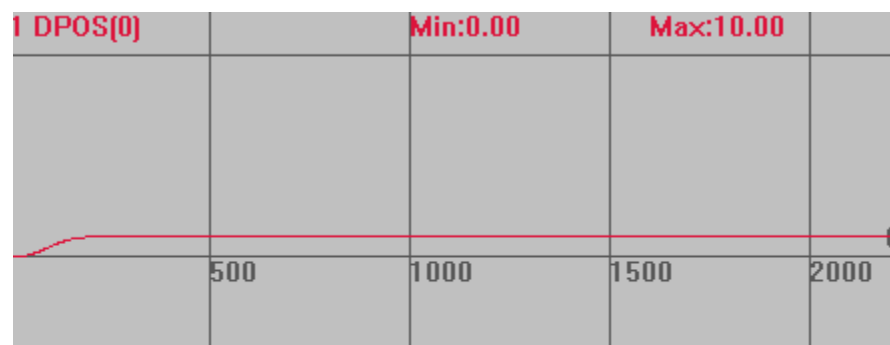
MOVEMODIFY(100)

```

MOVEMODIFY(10) 'axis will move to position 10 directly, MOVEMODIFY will not enter the motion buffer.

Motion trajectory:

DPOS(0) Vertical scale 100

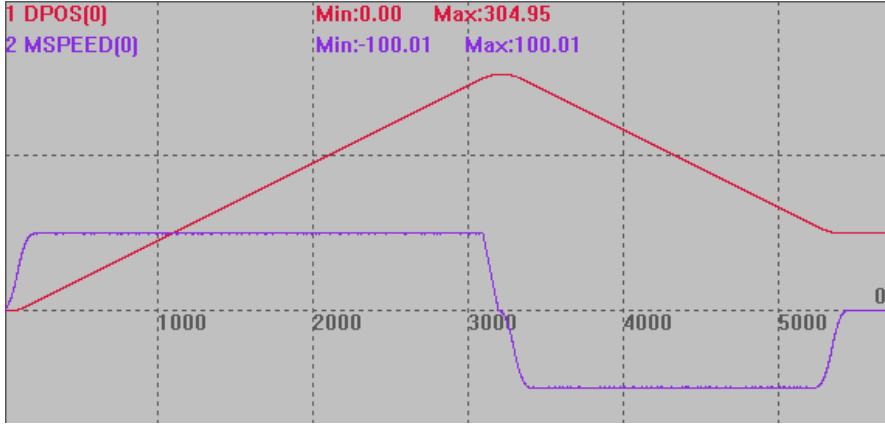
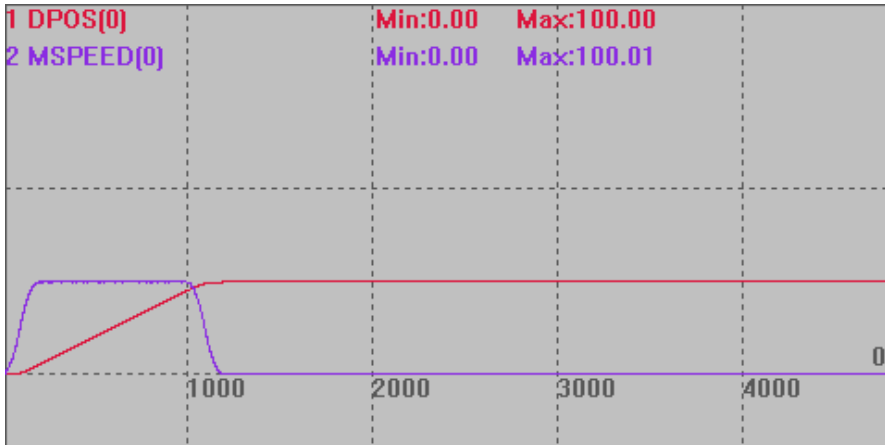


Example 2

```

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)

```

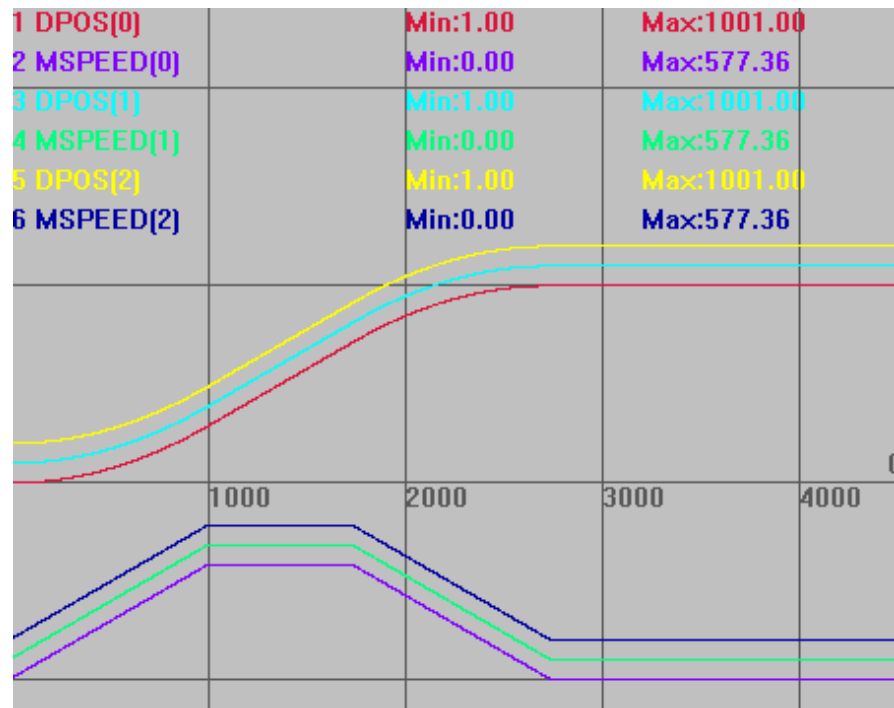
	WAIT UNTIL DPOS >=300 'modify the target position until axis reaches 300. MOVEMODIFY (100) 'Change the target position to 100, the axis will decelerate to stop, then move inversely. Use WAIT, motion trajectory: DPOS(0) vertical scale 200 MSPEED(0) vertical scale 200  Not use WAIT, move to position 100 directly. DPOS(0) vertical scale 300 
Instructions	MOVEMODIFY2

7.2. Multi-Axis Motion Instructions

RAPIDSTOP – All Axes Stop

Type	Multi-Axis Motion Instruction

Description	All axes stop immediately, if axes are 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 you need absolute position after RAPIDSTOP, please use “WAIT IDLE” to wait the movement to stop at first. And if there is motion pulse command before RAPIDSTOP, then the first one RAPIDSTOP stops the motion, the second time will cancel axis pause state.										
Syntax	RAPIDSTOP (mode) Mode: mode selection <table border="1"> <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.	0(default)	Cancel motion in process	1	Cancel motion in buffer	2	Cancel motions in process and in buffer, stop speed refer to fast deceleration FASTDEC	3	Stop pulse delivery immediately	4	Cancel motions in process and in buffer, stop speed refer to deceleration DECEL
0(default)	Cancel motion in process										
1	Cancel motion in buffer										
2	Cancel motions in process and in buffer, stop speed refer to fast deceleration FASTDEC										
3	Stop pulse delivery immediately										
4	Cancel motions in process and in buffer, stop speed refer to deceleration DECEL										
Controller	General										
Example	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										

**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

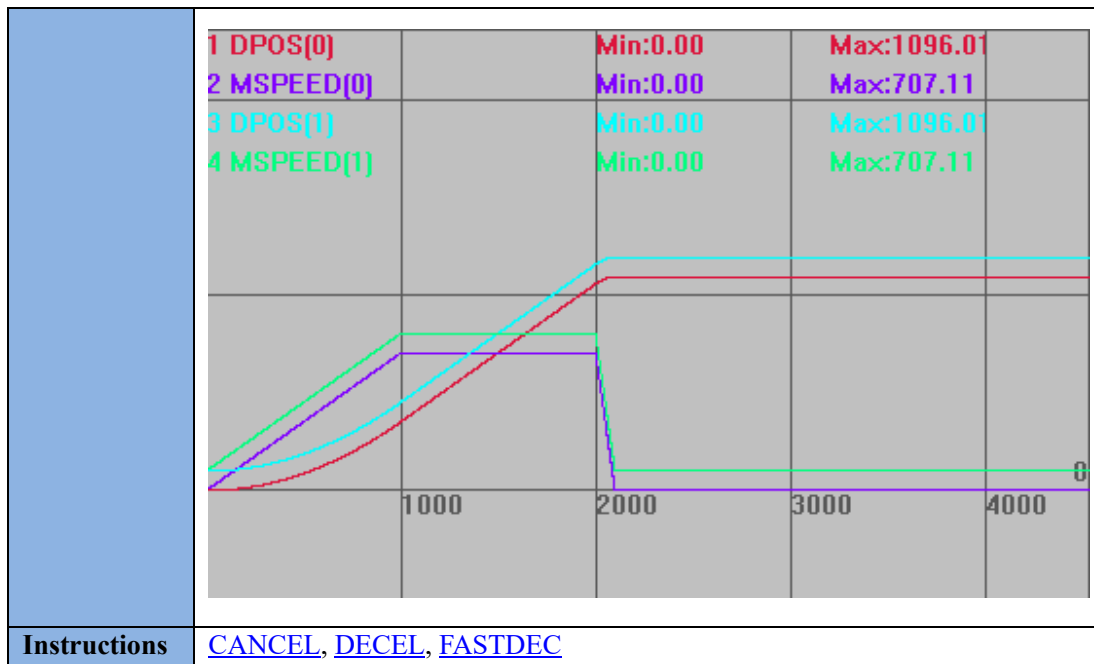
Motion trajectory and speed curve:

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



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: $X = \sqrt{X_0^2 + X_1^2 + X_2^2 + \dots + X_n^2}$ Motion time: $T = X / \text{speed of main axis}$.
Syntax	MOVE(distance1 [,distance2 [,distance3 [,distance4...]]]) ● 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 main axis speed 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. WAIT IDLE 'wait until the motion stops.

PRINT *DPOS

'Printed result:500,1000,1500

Each axis's speed 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

1 MSPEED[0]		Min:0.00	Max:26.73	
2 MSPEED[1]		Min:0.00	Max:53.46	
3 MSPEED[2]		Min:0.00	Max:80.18	
4 VP_SPEED[0]		Min:0.00	Max:100.00	



Example 2

BASE(0,1)

ATYPE=1,1

UNITS=100,100

SPEED=100,100

ACCEL=1000,1000

DECEL=1000,1000

DPOS=0,0

MPOS=0,0

Trigger

MOVE(100,100)

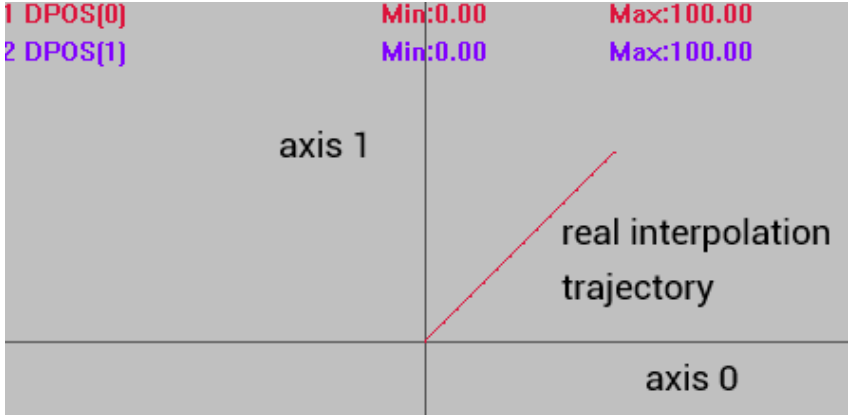
Interpolation trajectory

DPOS(0) horizontal scale 100

DPOS(1) vertical scale 100

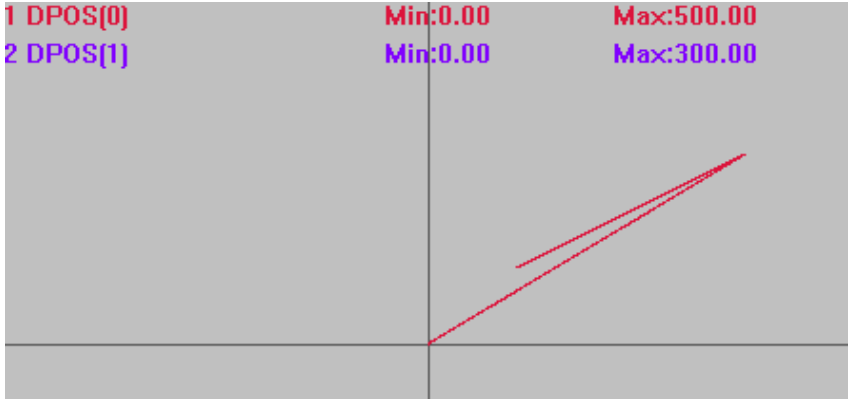
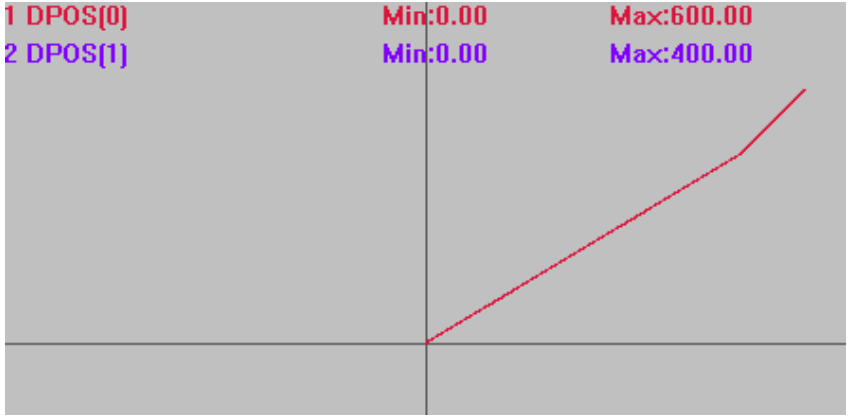
'pulse equivalent configuration

'Trigger the oscilloscope automatically

	
Instructions	MOVEABS , *SP

MOVEABS -- Linear Motion (Absolute)

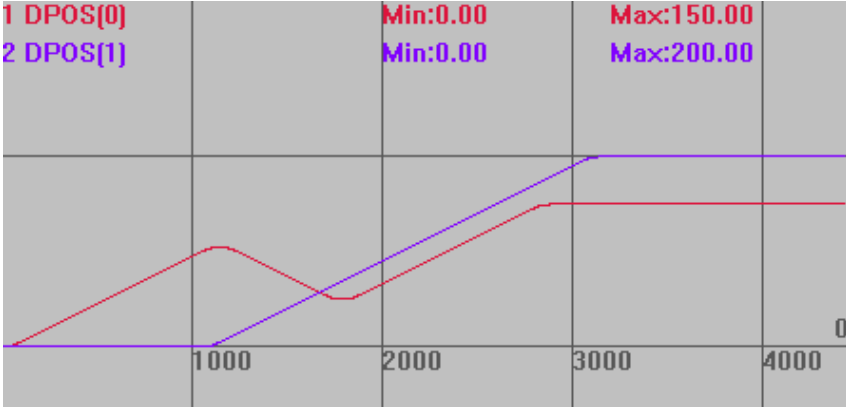
Type	Multi-Axis Motion Instruction
Description	Linear Interpolation movement, it moves absolutely to defined coordinate. This instruction can be used in continuous interpolation movements by adding SP, see *SP for reference.
Syntax	MOVEABS(position1[, position2[, position3[, position4...]]]) ● position1 -coordinate of first axis ● position2 -coordinate of next axis
Controller	General
Example	<pre>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</pre>

	 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

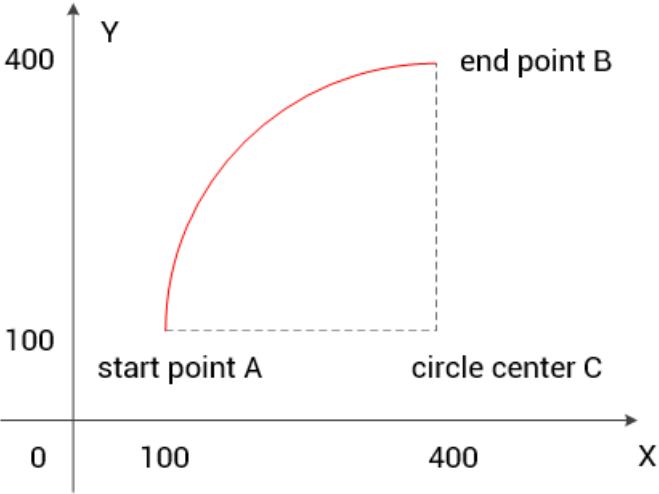
MOVEMODIFY2 -- Motion Position Modification 2

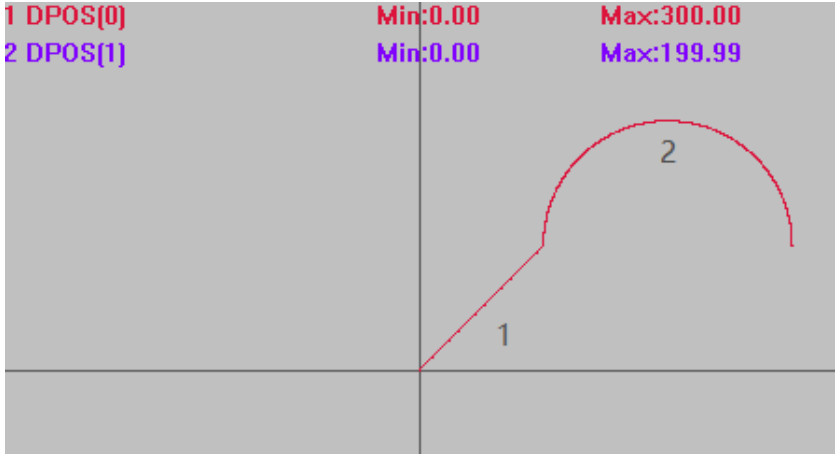
Type	Multi-Axis Motion Instruction
Description	Force the previous motion to stop, move to a new target position at former speed and acceleration. If there isn't motion in the former, then the result caused by this instruction is the same as MOVEABS, but each axis' motion is independent and will not enter 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.
Syntax	MOVEMODIFY2 (abspos1, abspos2,[...]) abspos1 BASE -Target position of axis 1

	abspos2 BASE -Target position of axis 2
Controller	Special Firmware: ZMC3XX series with firmware version above 20161209. ZMC4XX series with firmware version above 20170509.
Example	Example 1 <pre> 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 </pre> Example 2 <pre> 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: </pre>

	The vertical scale is the same as the above. 
Instructions	MOVEMODIFY

MOVECIRC -- Center Arc

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).
Syntax	MOVECIRC (end1, end2, centre1, centre2, direction) ● end1: the first axis' end point coordinate -- relative to starting point. ● end2: the second axis' end point coordinate -- relative to starting point ● center1: the first axis' center coordinate -- relative to starting point ● center2: the second axis' center coordinate -- 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 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

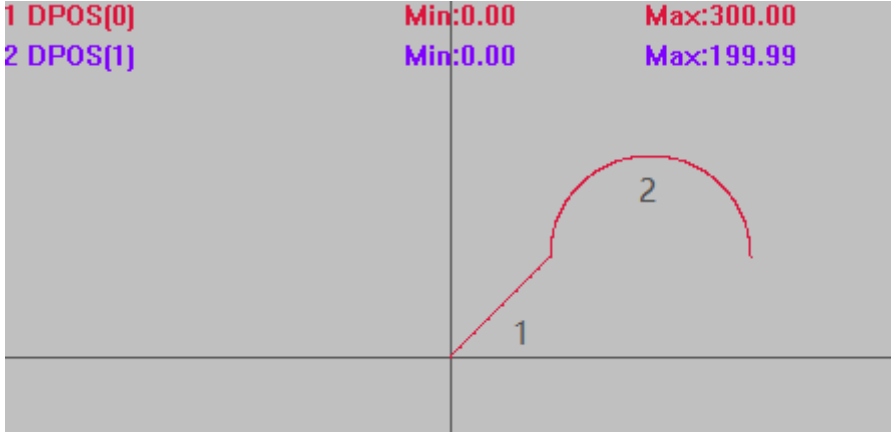
MOVECIRCABS -- Center Arc (Absolute)

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.
Syntax	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	<pre> 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) </pre>

	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
	1 DPOS[0] Min:0.00 Max:200.00 2 DPOS[1] Min:0.00 Max:100.00 (200,0)
Instructions	MOVECIRC , MOVECIRC2 , *SP

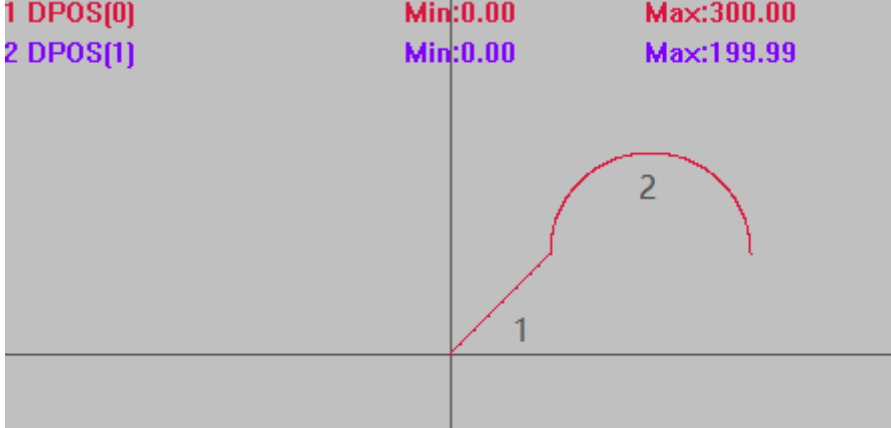
MOVECIRC2 -- 3-Point Arc

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.
Syntax	MOVECIRC2(mid1, mid2, end1, end2) ● mid1: middle point coordinate of the first axis -- relative to start point. ● mid2: middle point coordinate of the second axis -- relative to start point. ● end1: end point coordinate of the first axis -- relative to start point. ● end2: end point coordinate of the second axis -- 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	MOVECIRC , MOVECIRC2ABS , *SP

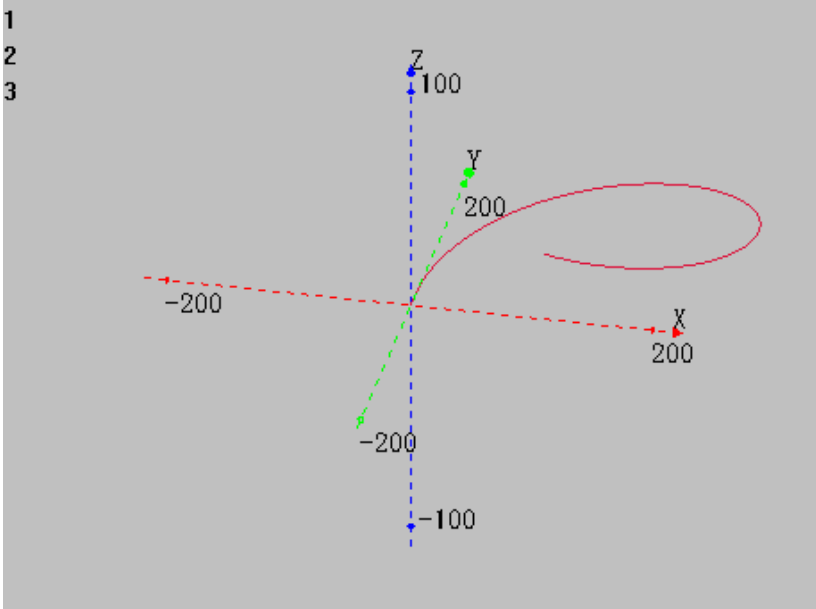
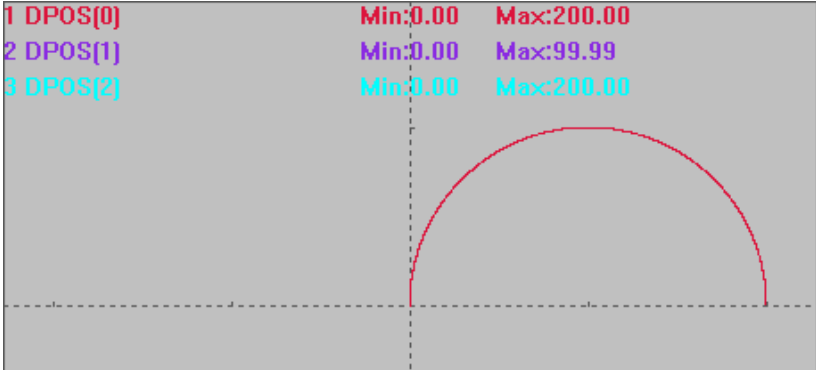
MOVECIRC2ABS -- 3-Point Arc (Absolute)

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.
Syntax	MOVECIRC2ABS(mid1, mid2, end1, end2) ● mid1: middle point coordinate of the first axis, absolute ● mid2: middle point coordinate of the second axis, absolute ● end1: end point coordinate of the first axis, absolute ● end2: end point coordinate of the second axis, absolute 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 , MOVECIRC2ABS , *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.				
Syntax	MHELICAL(end1,end2,centre1,centre2,direction,distance3,[mode]) ● end1: first axis' end point coordinate -- relative to start point. ● end2: second axis' end point coordinate -- relative to start point. ● center1: first axis' circle center coordinate -- relative to start point. ● center2: second axis' circle center coordinate -- relative to start point. ● direction: 0-anticlockwise 1-clockwisemode ● distance3: motion distance of the third axis ● mode: speed calculation of the third axis <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0(default)</td><td>Third axis participates interpolation speed calculation.</td></tr> </tbody> </table>	Value	Description	0(default)	Third axis participates interpolation speed calculation.
Value	Description				
0(default)	Third axis participates interpolation speed calculation.				

	<table><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.	1	Third axis keep independent.							
1	Third axis keep independent.									
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 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:  <table><tr><td>1 DPOS[0]</td><td>Min:0.00</td><td>Max:200.00</td></tr><tr><td>2 DPOS[1]</td><td>Min:0.00</td><td>Max:99.99</td></tr><tr><td>3 DPOS[2]</td><td>Min:0.00</td><td>Max:200.00</td></tr></table>	1 DPOS[0]	Min:0.00	Max:200.00	2 DPOS[1]	Min:0.00	Max:99.99	3 DPOS[2]	Min:0.00	Max:200.00
1 DPOS[0]	Min:0.00	Max:200.00								
2 DPOS[1]	Min:0.00	Max:99.99								
3 DPOS[2]	Min:0.00	Max:200.00								
Instructions	MHELICAL2 , MHELICALABS , *SP									

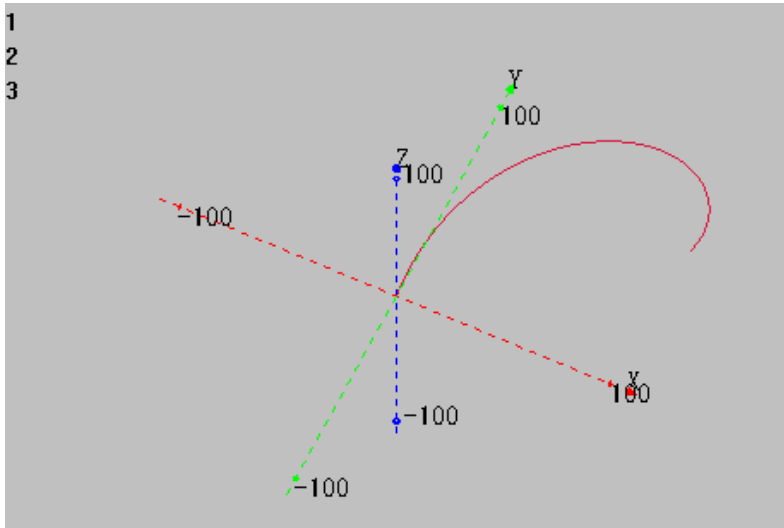
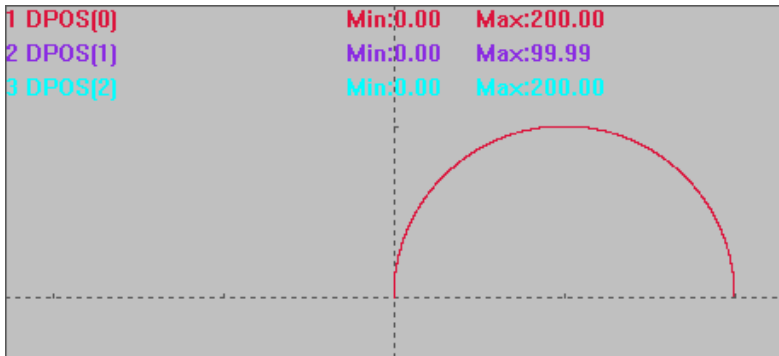
MHELICALABS -- Central Helical (Absolute)

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 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.						
Syntax	MHELICALABS(end1,end2,centre1,centre2,direction,distance3,[mode]) ● end1: motion coordinate of the first axis, absolute ● end2: motion coordinate of the second axis, absolute ● center1: motion center point of the first axis, absolute ● center2: motion center point of the second axis, absolute ● direction: 0-anticlockwise 1-clockwisemode ● distance3: motion distance of the third axis, absolute ● mode: speed calculation of the third axis <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <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> </tbody> </table> Ensure the coordinate is correct, or the actual motion path will be wrong.	Value	Description	0(default)	Third axis participates interpolation speed calculation.	1	Third axis keep independent.
Value	Description						
0(default)	Third axis participates interpolation speed calculation.						
1	Third axis keep independent.						
Controller	General						
Example	<pre> 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 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: </pre>						

	Axis 0 and axis 1 interpolation trajectory under XY mode:
Instructions	MHELICAL , MHELICAL2ABS , *SP

MHELICAL2 -- 3-Point Helical

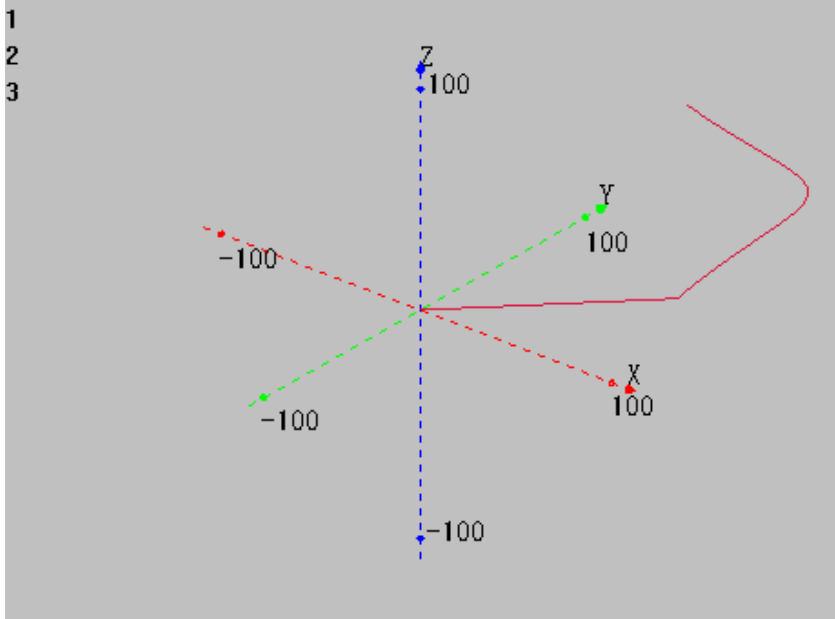
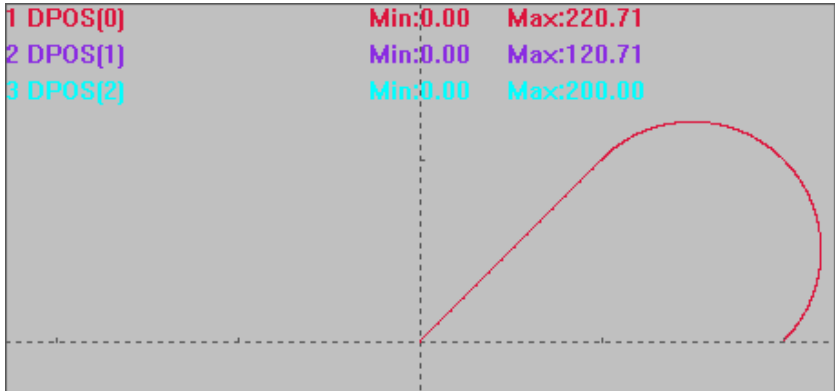
Type	Multi-Axis Motion Instruction
Description	Helical Interpolation, arc at the center, absolute motion. The first and second axis in BASE list will execute circular interpolation, the third axis will execute helical, in relative motion way. This instruction can be used in continuous interpolation movements by adding SP, see *SP for reference. It can't generate a full circle in Z direction, please use MHELICAL or MHELICALABS.
Syntax	MHELICAL2(mid1, mid2, end1, end2, distance3,[mode]) ● mid1: middle point coordinate of the first axis -- relative to start point. ● mid2: middle point coordinate of the second axis -- relative to start point.

	● end1: end point coordinate of the first axis -- relative to start point. ● end2: end point coordinate of the second axis -- relative to start point. ● distance3: motion distance of the third axis -- 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.	Value	Description	0(default)	Third axis participates interpolation speed calculation.	1	Third axis keep independent.			
Value	Description									
0(default)	Third axis participates interpolation speed calculation.									
1	Third axis keep independent.									
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:  <table><tr><td>1 DPOS[0]</td><td>Min:0.00</td><td>Max:200.00</td></tr><tr><td>2 DPOS[1]</td><td>Min:0.00</td><td>Max:99.99</td></tr><tr><td>3 DPOS[2]</td><td>Min:0.00</td><td>Max:200.00</td></tr></table>	1 DPOS[0]	Min:0.00	Max:200.00	2 DPOS[1]	Min:0.00	Max:99.99	3 DPOS[2]	Min:0.00	Max:200.00
1 DPOS[0]	Min:0.00	Max:200.00								
2 DPOS[1]	Min:0.00	Max:99.99								
3 DPOS[2]	Min:0.00	Max:200.00								

Instructions	MHELICAL , MHELICAL2ABS , *SP

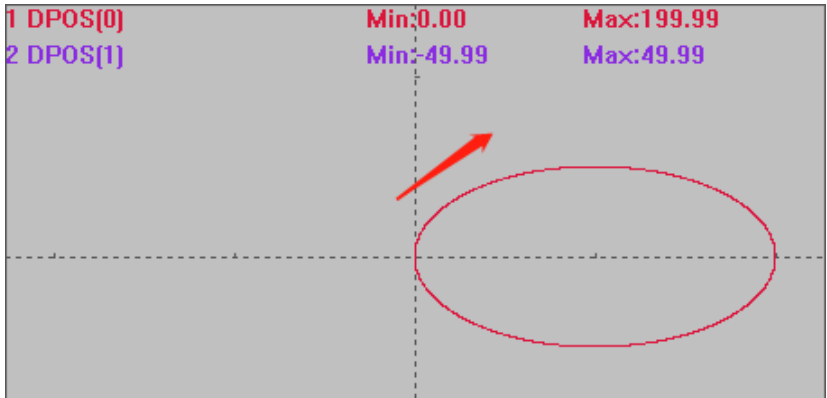
MHELICAL2ABS -- 3-Point Helical (Absolute)

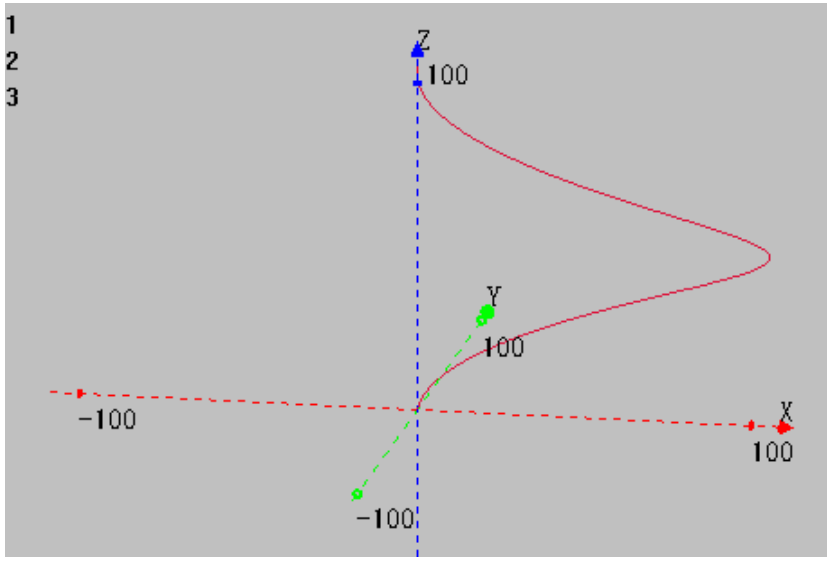
Type	Multi-Axis Motion Instruction						
Description	Helical Interpolation, arc at the center, absolute motion. The first and second axis in BASE list will execute circular interpolation, the third axis will execute helical, in relative motion way. This instruction can be used in continuous interpolation movements by adding SP, see *SP for reference. It can't generate a full circle in Z direction, please use MHELICAL or MHELICALABS.						
Syntax	MHELICAL2(mid1, mid2, end1, end2, distance3,[mode]) ● mid1: middle point coordinate of the first axis, absolute ● mid2: middle point coordinate of the second axis, absolute ● end1: end point coordinate of the first axis, absolute ● end2: end point coordinate of the second axis, absolute ● 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 border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <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> </tbody> </table> Ensure the coordinate is correct, or the actual motion path will be wrong.	Value	Description	0(default)	Third axis participates interpolation speed calculation.	1	Third axis keep independent.
Value	Description						
0(default)	Third axis participates interpolation speed calculation.						
1	Third axis keep independent.						
Controller	General						
Example	<pre> 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: </pre>						

1 2 3									
	Axis 0 and axis 1 interpolation trajectory under XY mode:								
	<table><tr><td>1 DPOS[0]</td><td>Min:0.00</td><td>Max:220.71</td></tr><tr><td>2 DPOS[1]</td><td>Min:0.00</td><td>Max:120.71</td></tr><tr><td>3 DPOS[2]</td><td>Min:0.00</td><td>Max:200.00</td></tr></table> 	1 DPOS[0]	Min:0.00	Max:220.71	2 DPOS[1]	Min:0.00	Max:120.71	3 DPOS[2]	Min:0.00
1 DPOS[0]	Min:0.00	Max:220.71							
2 DPOS[1]	Min:0.00	Max:120.71							
3 DPOS[2]	Min:0.00	Max:200.00							
Instructions	MHELICAL2 , MHELICALABS , *SP								

MECLIPSE -- Ellipse

Type	Multi-Axis Motion Instruction
Description	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.
Syntax	MECLIPSE (end1, end2, centre1, centre2, direction, adis, bdis[, end3]) ● end1: end point coordinate of the first axis -- relative to start point. ● end2: end point coordinate of the second axis -- relative to start point. ● center1: center point coordinate of the first axis -- relative to start point.

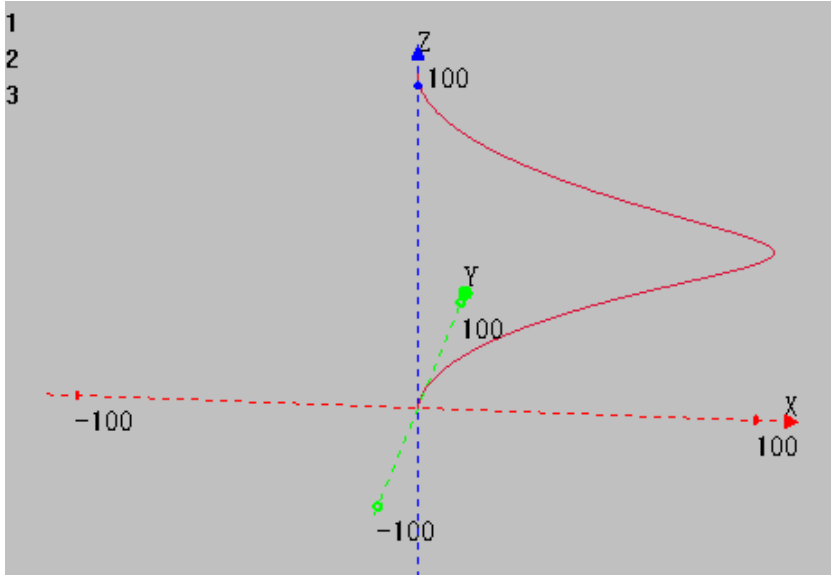
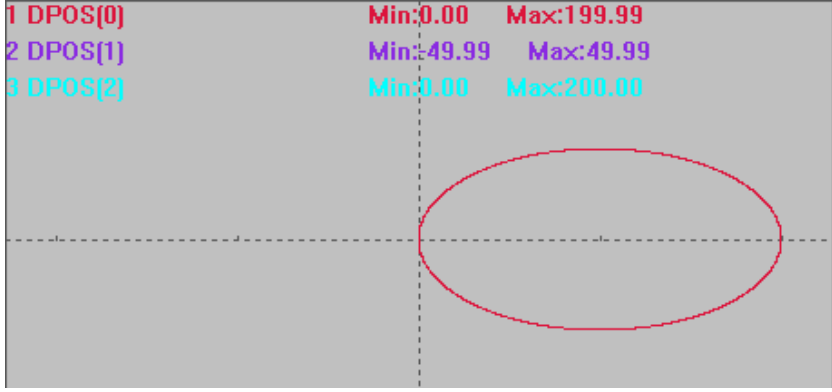
	- center2: center point coordinate of second axis -- relative to start point. - direction: 0-anticlockwise 1-clockwise <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Clockwise</td></tr> <tr> <td>1</td><td>Anticlockwise</td></tr> </tbody> </table> - adis: ellipse radius of first axis, semi-major / semi-minor axis. - bdis: ellipse radius of the second axis, semi-major / semi-minor axis. 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.	Value	Description	0	Clockwise	1	Anticlockwise
Value	Description						
0	Clockwise						
1	Anticlockwise						
Controller	General						
Example	Example 1 No 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) '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)						

	<pre> 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 clockwise 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. </pre>
	
Instructions	MECLIPSEABS , *SP

MECLIPSEABS -- Ellipse (Absolute)

Type	Multi-Axis Motion Instruction
Description	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. It can draw one full ellipse (controller 64-bit accuracy). When the starting point and end point is closing, for the circle trajectory, use the relative mode.
Syntax	MECLIPSEABS(end1, end2, centre1, centre2, direction, adis, bdis[, end3])

	● end1: end point coordinate of the first axis, absolute ● end2: end point coordinate of the second axis, absolute ● center1: center point coordinate of the first axis, absolute ● center2: center point coordinate of the second axis, absolute. ● direction: 0-anticlockwise 1-clockwise <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Clockwise</td></tr> <tr> <td>1</td><td>Anticlockwise</td></tr> </tbody> </table> ● adis: ellipse radius of the first axis, semi-major / semi-minor axis ● bdis: ellipse radius of the second axis, semi-major / semi-minor axis. 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.	Value	Description	0	Clockwise	1	Anticlockwise
Value	Description						
0	Clockwise						
1	Anticlockwise						
Controller	General						
Example	Example 1: no helical <pre> 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 </pre> Example 2 with helical <pre> BASE(0,1,2) </pre>						

	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:  Axis 0 and axis 1 interpolation trajectory: 
Instructions	MECLIPSE , *SP

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.										
Syntax	MSPHERICAL(end1,end2,end3,centre1,centre2,centre3,mode[,distance4][,distance5]) ● 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 border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <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> </tbody> </table> ● distance4: add the fourth axis as helical motion, appoint the relative motion distance of axis 4. This axis is not involved in speed calculation. ● distance5: add the fifth axis as helical motion, appoint the relative motion distance of axis 5. This axis is not involved in speed calculation. ● distance6: add the sixth axis as helical motion, appoint the relative motion distance of axis 6. This axis is not involved in speed calculation. Ensure coordinates are correct, otherwise, the real motion path will be wrong.	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.
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.										
Controller	General										
Example	<pre> 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 </pre>										

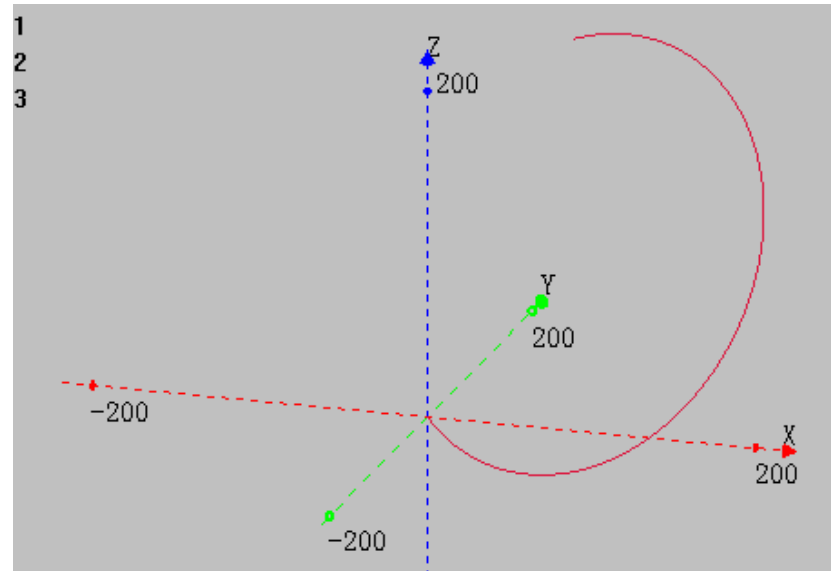
generated below due to mode differences.

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:

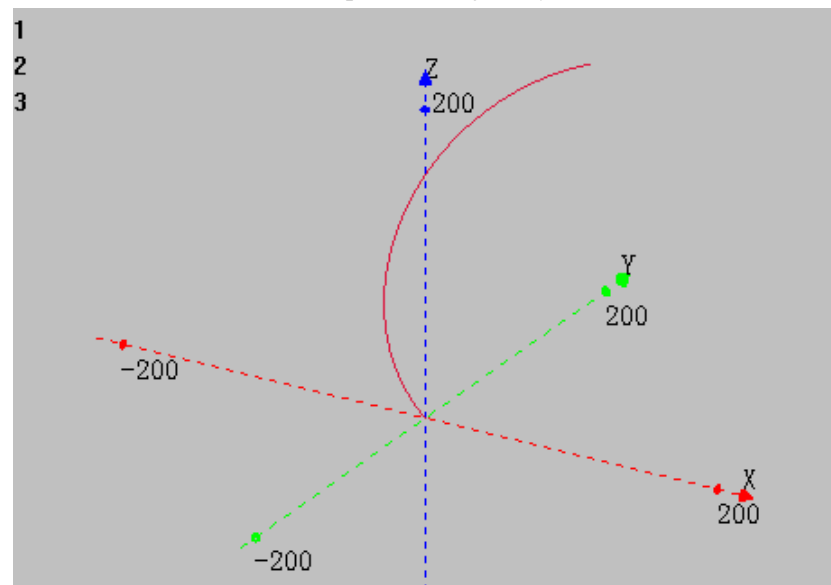


mode 1:

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:

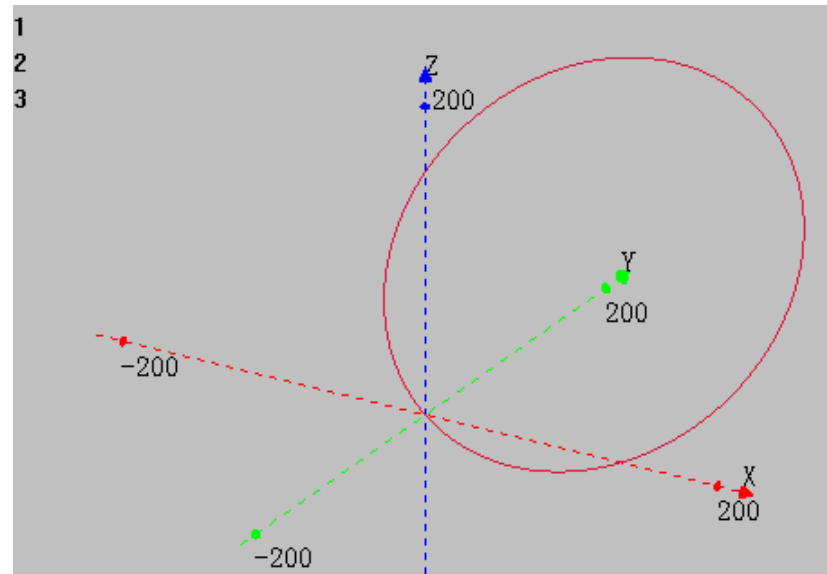


mode 2:

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:

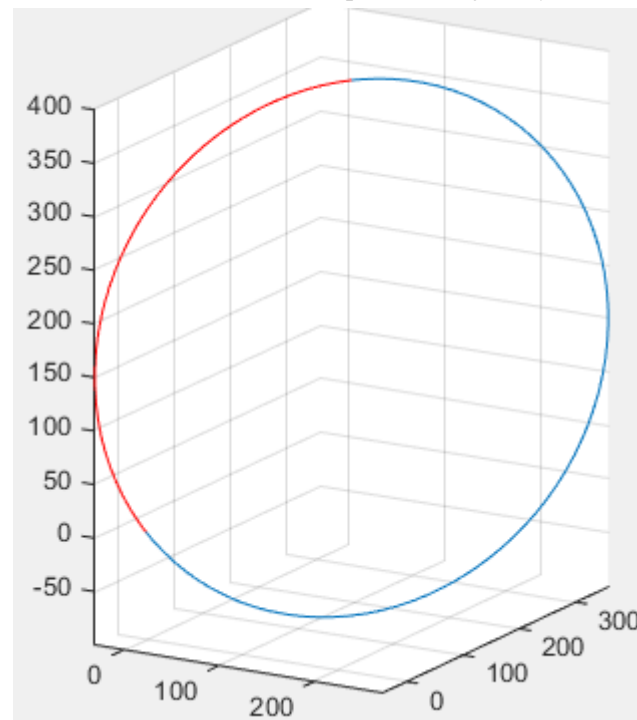


mode 3

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)

Type	Multi-Axis Motion Instruction
Description	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.										
Syntax	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 border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <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> </tbody> </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. ● distane6: add the sixth axis as helical motion, appoint the relative motion distance of axis 6. This axis is not involved in speed calculation. Ensure coordinate is correct, otherwise, the actual motion path will be wrong.	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.
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.										
Controller	General										
Example	<pre> 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 </pre>										

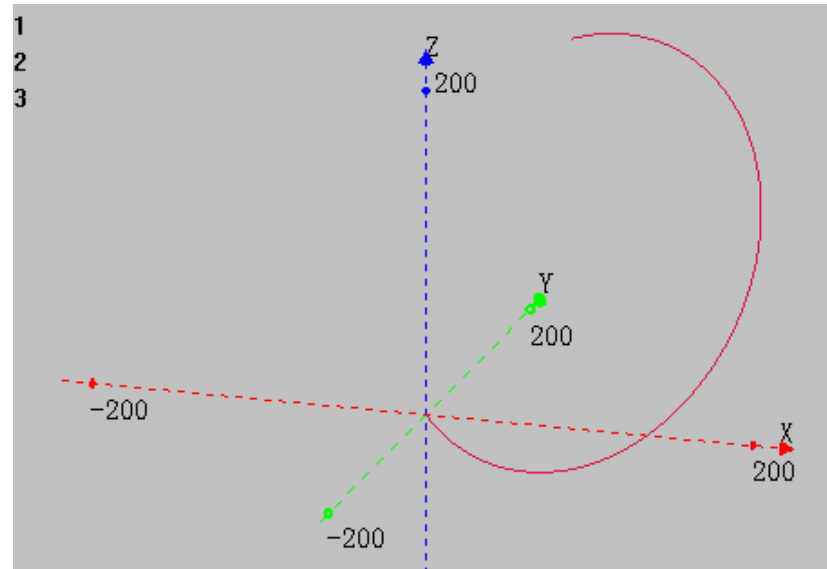
Suppose central point is (120,160,150), radius is 250, then below shows motion trajectories of 4 modes.

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:

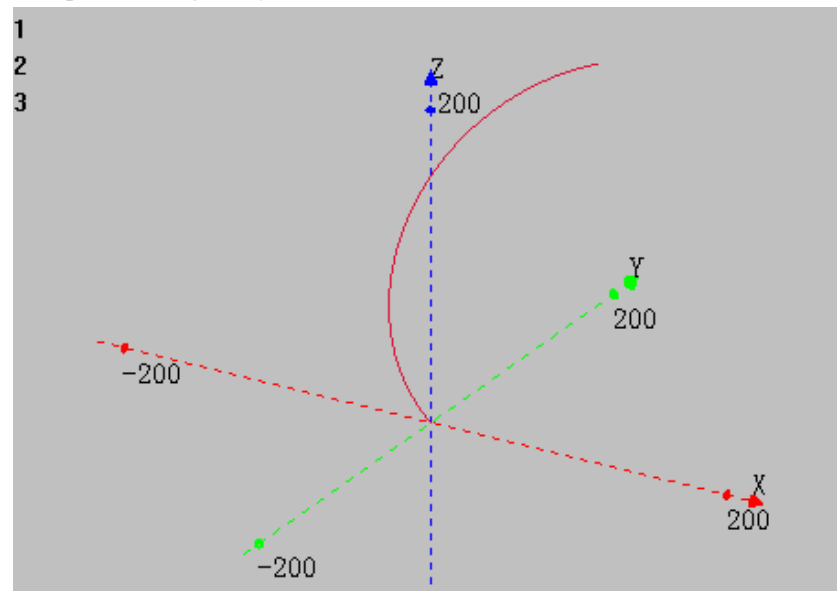


mode 1:

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:



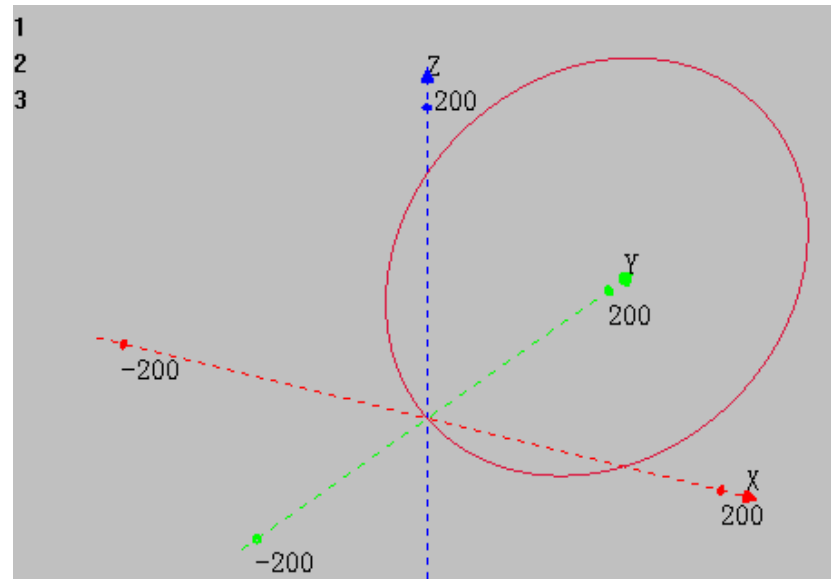
mode 2:

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:

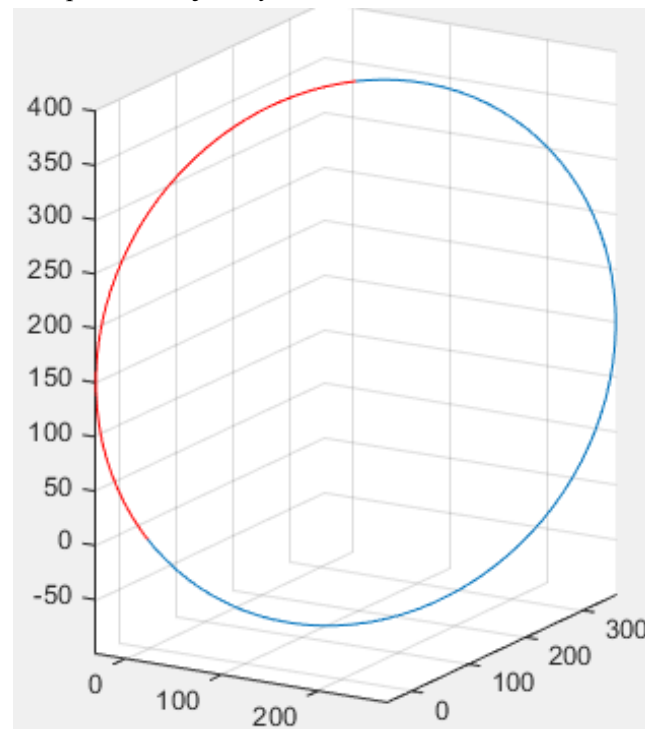


mode 3

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:

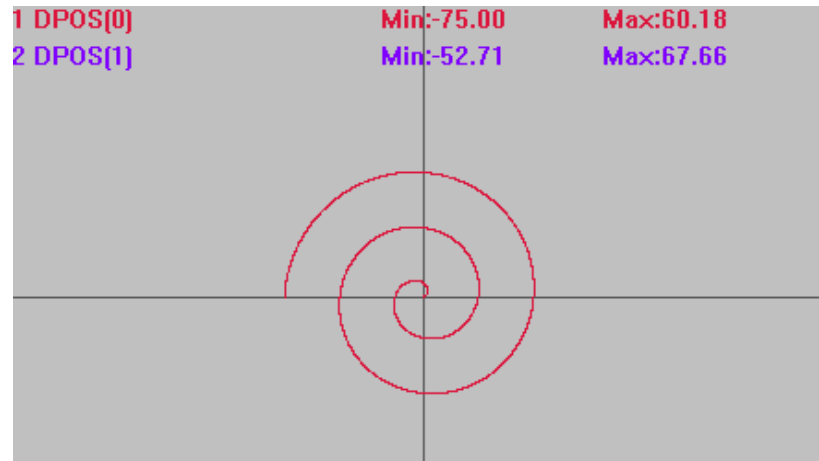


Instructions

[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.
Syntax	MOVESPIRAL(centre1,centre2,circles,pitch[,distance3][,distance4]) ● 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	<pre>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</pre> 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)

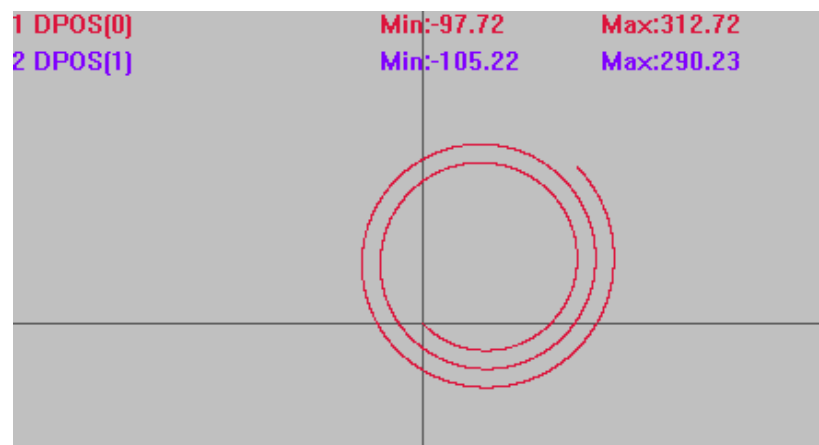
'start radius is 100, central point is (100,100), rotate 2.5 circles anticlockwise, diffusion distance of each circle is 30.

Interpolation Path

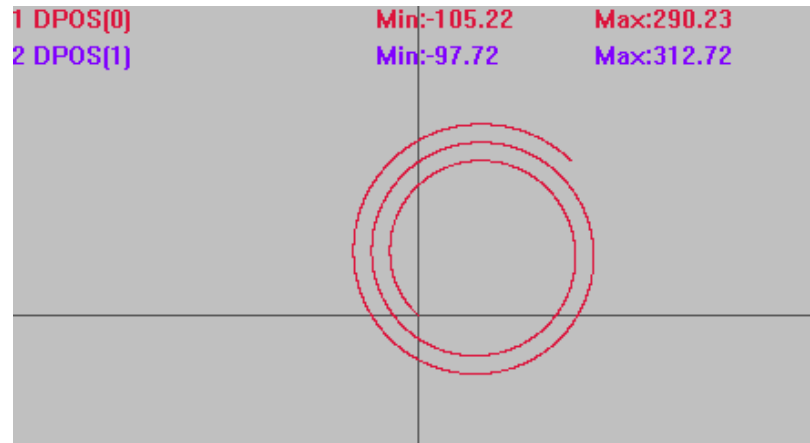
(if path circle amount is not full displayed, make captured gap proper bigger)

DPOS(0) vertical scale 300

DPOS(1) vertical scale 300

**MOVESPIRAL** (100,100,-2.5,3.)

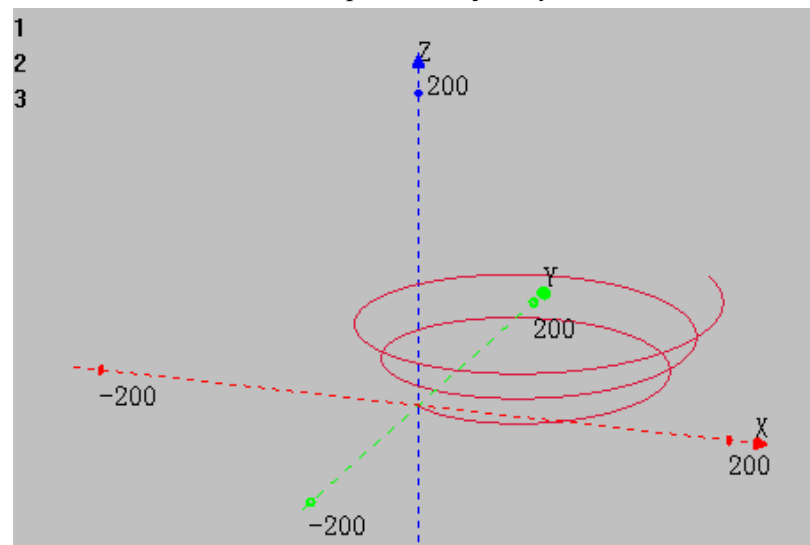
'when the number of rotations is negative (-2.5), rotate clockwise

**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.

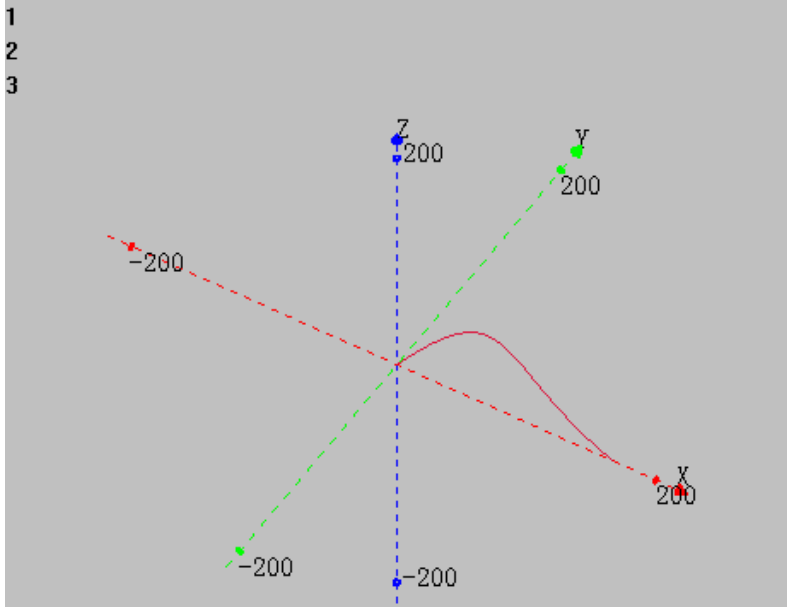
Axis 0, axis 1 and axis 2 interpolation trajectory under XYZ mode:



MOVESPLINE/MOVESPLINEABS -- Spline Interpolation

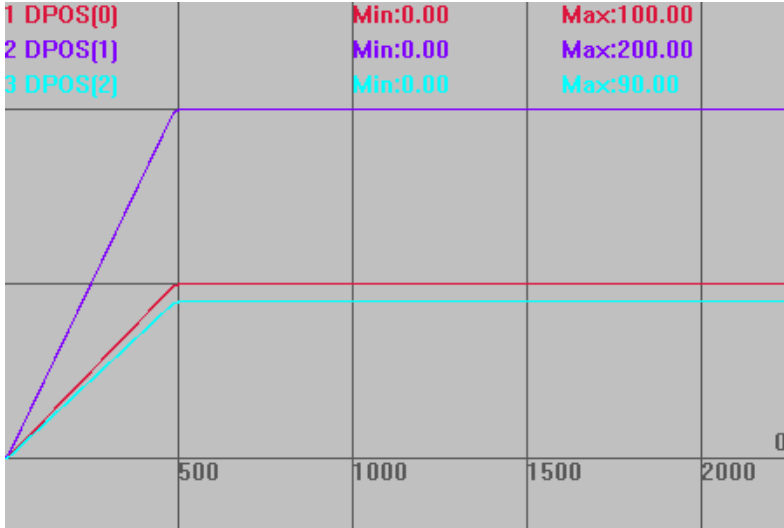
Type	Special Motion Instruction
Description	Spline interpolation, relative or absolute motion. Fill the spline points data into TABLE in advance. This instruction doesn't support SP function, continuous interpolation with self-defined speed can be set by instructions: BIT8 of CONNER_MODE.
Syntax	MOVESPLINE (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) & TABLE(1) will store 2nd control point, relative to start point. TABLE(10,200,-100) 'TABLE(10) & TABLE(11) will store 3rd control point, relative to start point. TABLE(20,300,0) 'TABLE(20) & TABLE(21) will store 4th 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) & TABLE(1) will store 2nd control point, relative to start point. TABLE(10,200,-100,200) 'TABLE(10) & TABLE(11) will store 3rd control point, relative to start point. TABLE(20,300,0,0) 'TABLE(20) & TABLE(21) will store 4th 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

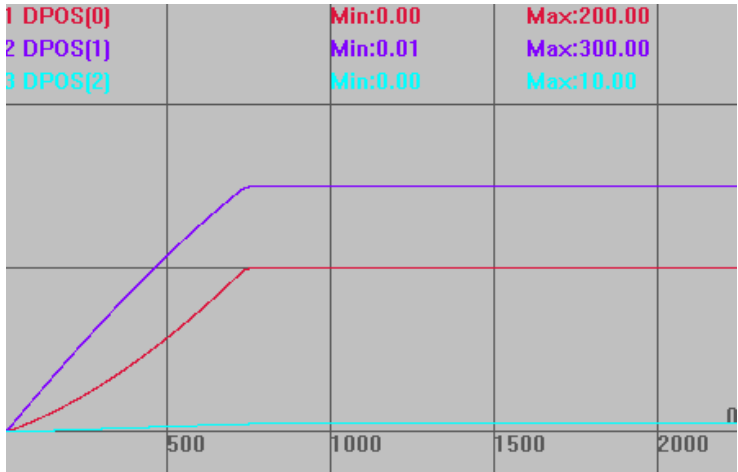
MOVE_TURNABS -- Rotary Table Interpolation

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 <i>ZMOTION Robotic Instructions Reference</i> for reference, the related frame is frame11/17.
Syntax	MOVE_TURNABS(tablenum,position1[,position2[,position3[, position4...]]]) ● tablenum: Table No. where saves rotating parameters.

	● position1: coordinate of the first axis ● position2: coordinate of next axis
Controller	General
Example	<pre> 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 </pre> 
Instructions	MCIRC_TURNABS

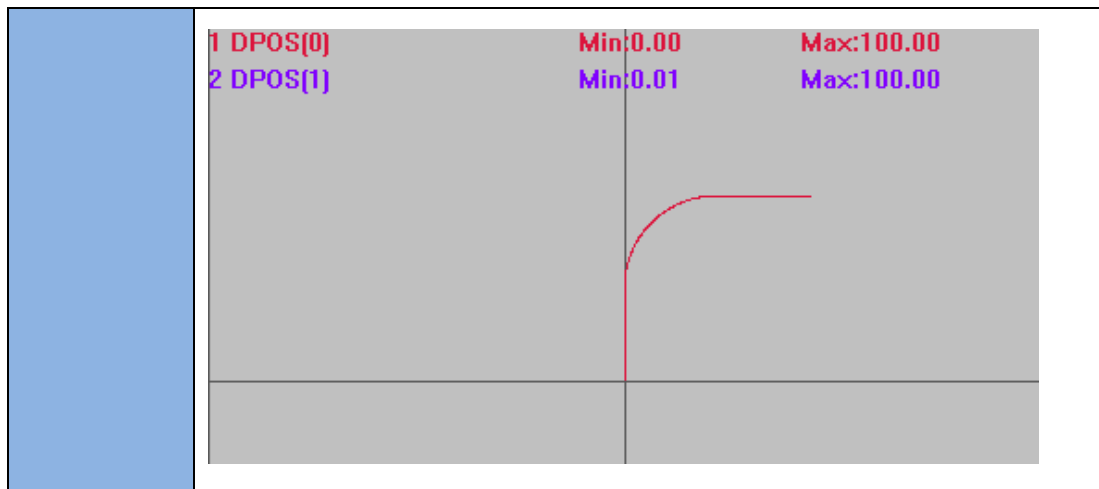
MCIRC_TURNABS -- Rotary Table Circular Interpolation

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.
Syntax	MCIRC_TURNABS(tablenum, refpos1, refpos2, mode, end1, end2 [, dis3, dis4, dis5]) ● tablenum: Table No. where 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 ● end2: 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

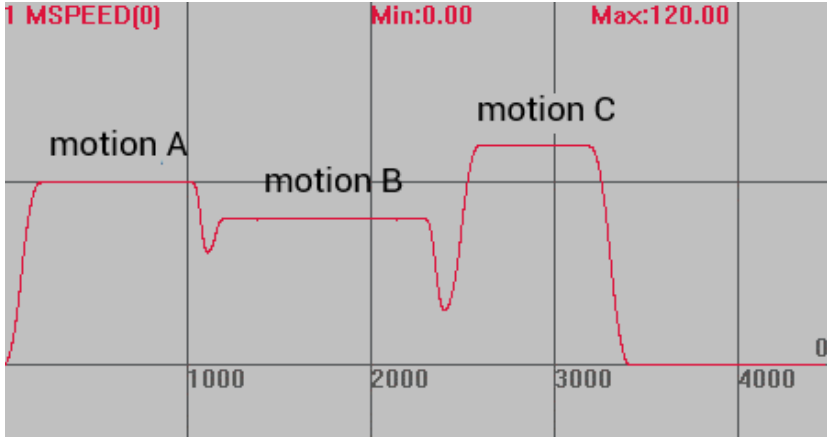
MOVESMOOTH -- Fillet

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.
Syntax	MOVESMOOTH (end1, end2, end3, next1, next2, next3, radius) ● 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	<pre> 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 </pre>



MOVESP -- SP Motion

Type	Multi-Axis Motion Instruction	
Description	It is used to do linear interpolation motion, making it move one relative distance, using SP to speed the limit. In the continuous interpolation motion that is set customized speed, it can use SP suffix commands, please check *SP command for details.	
Syntax	MOVESP(distance1 [,distance2 [,distance3 [,distance4...]]]) - distance 1: the first axis motion distance - distance 2: the next axis motion distance	
Controller	General	
Example	<pre> BASE(0) DPOS=0 ATYPE=1 UNITS=100 ACCEL=1000 DECEL=1000 SRAMP=100 MERGE=ON SPEED=100 FORCE_SPEED=80 STARTMOVE_SPEED=60 ENDMOVE_SPEED=30 TRIGGER MOVE(100) MOVESP(100) FORCE_SPEED=120 ENDMOVE_SPEED=30 MOVESP(100) </pre> Speed Path MSPEED(0) vertical scale 100 'open continuous interpolation. 'motion speed is 100 'limit speed is 80 'start speed is 60 'end speed is 30 'Trigger the oscilloscope automatically 'motion A, no SP limit. 'motion B, use SP limit. 'speed limit is 120 'end speed limit is 30 'motion C, use SP limit.	

	 When there is no SP limit, the motion speed is SPEED. When SP command is used, the motion speed is FORCE_SPEED. And when both STARTMOVE_SPEED and ENDMOVE_SPEED are set, STARTMOVE_SPEED will take effect.
Instructions	<u>*SP</u>

*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.
Syntax	SP based instructions: MOVESP, MOVEABSSP, MOVECIRCSP, MOVECIRCASSP, MHELICALSP, MHELICALABSSP, MECLIPSESP, MECLIPSEABSSP, MSPHERICALSP. FORCE_SPEED, ENDMOVE_SPEED and STRATMOVE_SPEED will enter motion buffer.
Controller	General
Example	Example 1: please check MOVE_SP example. <pre> BASE(0) DPOS=0 ATYPE=1 UNITS=100 ACCEL=1000 DECEL=1000 SRAMP=100 MERGE=ON </pre> '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 is 30
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



When there is no SP limit, the motion speed is SPEED.

When SP command is used, the motion speed is FORCE_SPEED.

And when both STARTMOVE_SPEED and ENDMOVE_SPEED are set, STARTMOVE_SPEED will take effect.

Example 2

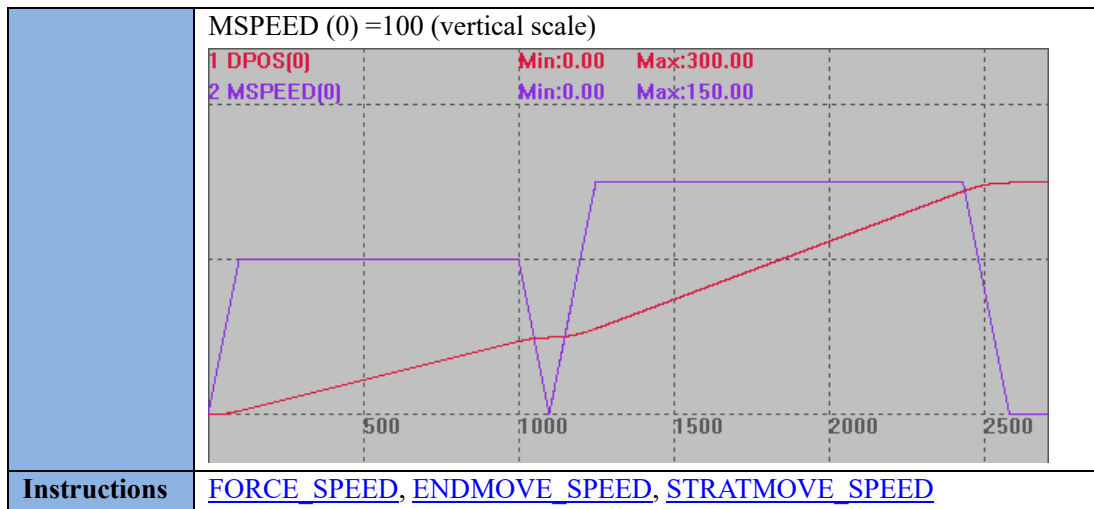
```

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

```

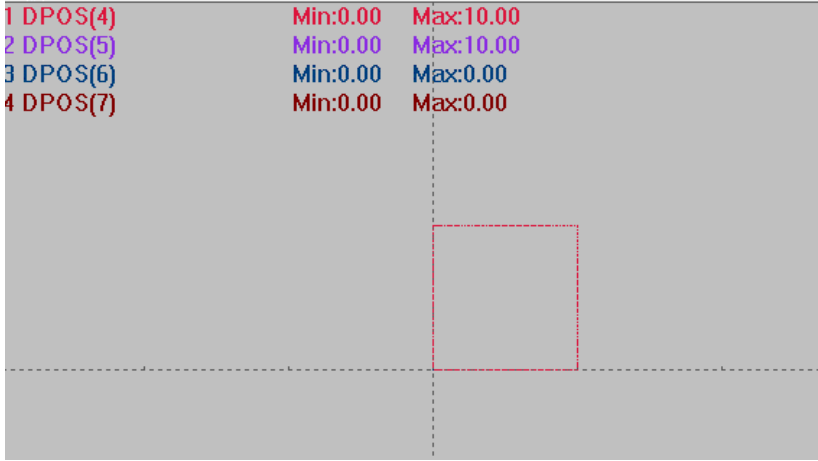
Motion trajectory:

DPOS (0) = 200 (vertical scale)



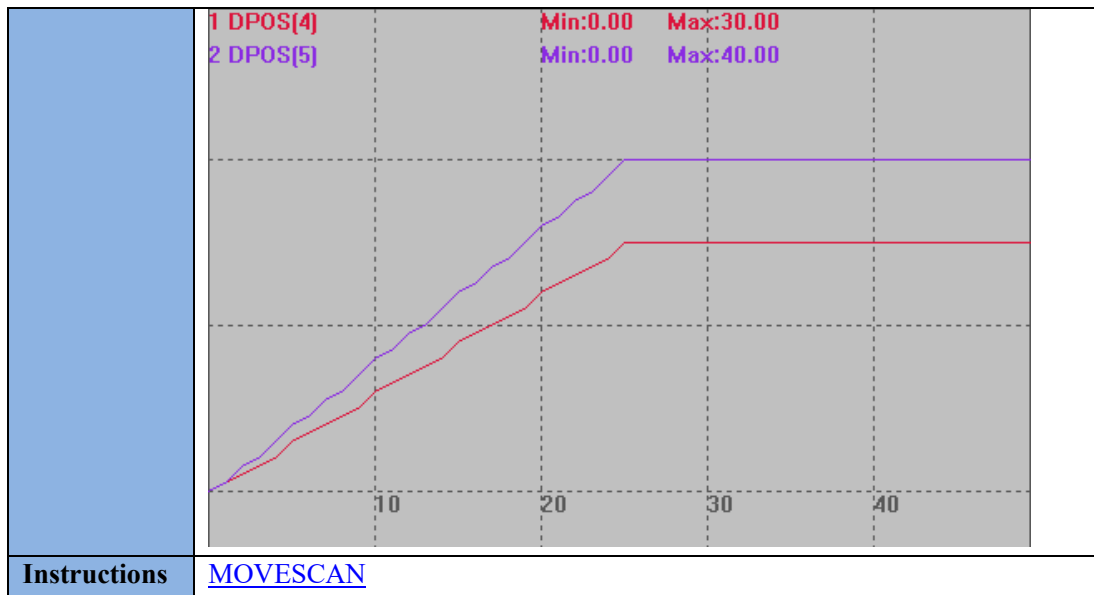
MOVESCAN -- Galvanometer (SCAN) Motion

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 delay. 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.
Syntax	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

	<pre> MOVE_OP(0,1) 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 </pre> Resultant trajectory under galvanometer axis XY mode: DPOS(4), vertical scale (Y scale): 10 DPOS(5), vertical scale (Y scale): 10  Example 2 <pre> 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 </pre>
Instructions	MOVE

MPULSCAN -- Galvanometer (SCAN) 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.
Syntax	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; Valid in firmware version above 20220225.
Example	<pre> 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 </pre>



7.3. Special Motion Instructions

MOVE_PAUSE -- Motion Pause

Type	Special Motion Instruction										
Description	Pause BASE axis motion. 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 the process. Some motions don't support pause, such as, VMOVE, synchronization motion instructions, etc.										
Syntax	➤ MOVE_PAUSE (mode) <table border="1"> <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 (ready for next motion) and MARK of two commands are different. 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>Force to pause, even pause under IDLE mode. Valid in firmware version above 20170513.</td></tr> <tr> <td>4</td><td>Pause assigned vector position. Valid in firmware version after version_build 241025.</td></tr> </table> ➤ MOVE_PAUSE (mode [,markbits]) - mode: please refer to above form, here is for mode 2; - markbits: to check whether MARK types are same or not when mode =	0 (default)	Pause the present motion.	1	Pause when the present motion finished completely.	2	Pause when present motion is finished completely (ready for next motion) and MARK of two commands are different. This mode can be used to suspend one motion which consist of multiple instructions when it is finished.	3	Force to pause, even pause under IDLE mode. Valid in firmware version above 20170513.	4	Pause assigned vector position. Valid in firmware version after version_build 241025.
0 (default)	Pause the present motion.										
1	Pause when the present motion finished completely.										
2	Pause when present motion is finished completely (ready for next motion) and MARK of two commands are different. This mode can be used to suspend one motion which consist of multiple instructions when it is finished.										
3	Force to pause, even pause under IDLE mode. Valid in firmware version above 20170513.										
4	Pause assigned vector position. Valid in firmware version after version_build 241025.										

	2, 32-bit integer & default value -1. ➤ MOVE_PAUSE (4, vectorpos) ● mode: please refer to above form, here is for mode 4; ● vectorpos: specify the pause position, don't be smaller than vector_moved, and don't be larger than vector_buffered. Notes: ! used under MERGE=ON; ! not recommend to use under separate mode or for MOVER & SPLINE commands, because the position may be wrong; ! don't call MOVE_PAUSE immediately after paused (4 is to set next paused position because MOVE_RESUME will clear pause).
Controller	MOVE_PAUSE (mode) – General; MOVE_PAUSE (mode [,markbits]) – ZMC4XX & version_build 220711; MOVE_PAUSE (4, vectorpos) – ZMC4XX & version_build 241025;
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 it will 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. WAIT IDLE ?DPOS(0) 'the present motion is finished. 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 at first, then pause until mark of next motion differs from the present motion. DELAY(3000) 'wait until motion pause. ?DPOS(0)

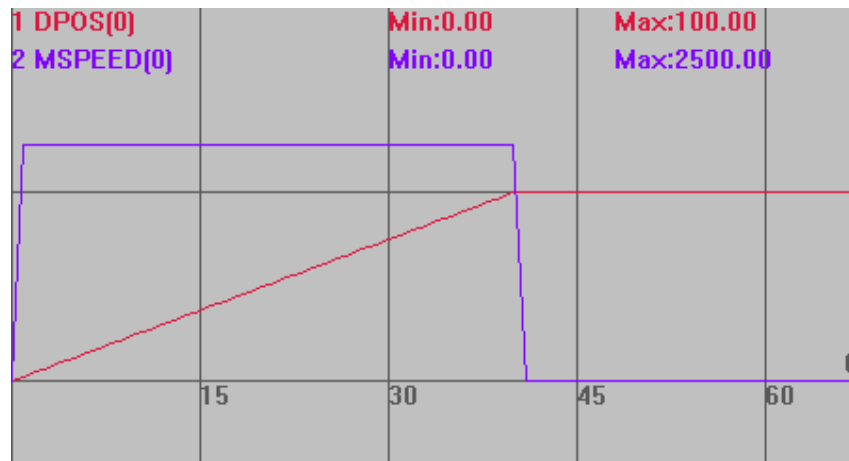
	'print result, -100 (present motion will not pause if the speed is too slow, the print result will over -100) 'now the current motion is finished, and the next motion MARK No. will be set as the same one for keep running. Then it will pause again when 3rd motion MARK No. is different. Example 4 mode 4 <pre> WHILE 1 MOVE_PAUSE (4, VECTOR_MOVED + 100) WAIT UNTIL AXISSTATUS AND \$800000 ? "PAUSED", VECTOR_MOVED MOVE_RESUME DELAY 1 'to ensure the axisstatus is correct, because AXISSTATUS is updated spacing one period. WEND </pre>
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.
Syntax	MOVE_RESUME
Controller	General
Example	<pre> 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 </pre>
Instructions	MOVE_PAUSE , AXISSTATUS

MOVE_PT -- Unit Distance

Type	Special Motion Instruction
Description	Drive the motor to reach one certain position in a period time. Usually, PC will calculate corresponding coordinates in every period, then transfer it to controller. BASE assigned axis can be used. Motion speed = (DIS/TICKS)*1000units/s Don't run a long distance in a very short time, the pulse frequency maybe too high, causing motor stall. It is better to divide the long distance into pieces. “multi-period speed auto-even” function is added by MOVE_PT.
Syntax	MOVE_PT (TICKS, DIS1,DIS2...) ● ticks: servo period numbers of time, time = system period * ticks ● dis1: motion distance when the controller SERVO_PERIOD is 1ms, TICKS = 1ms (for different SERVO_PERIOD, TICKS are different).
Controller	General
Example	Example 1: <pre> 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 </pre>

**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)

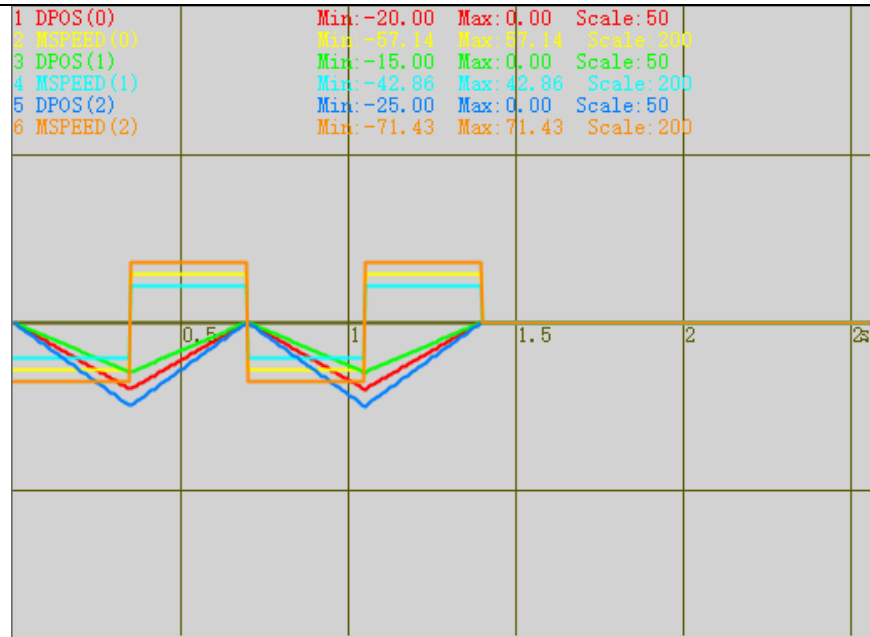
MSPEED(0) = 200 (vertical scale)

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|

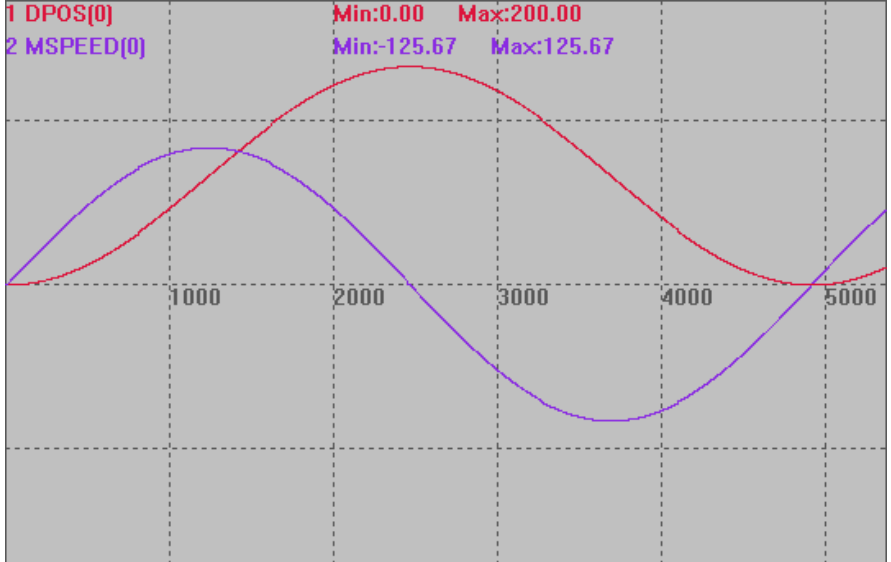
'x=A*COS(ωx+ψ)+C

'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

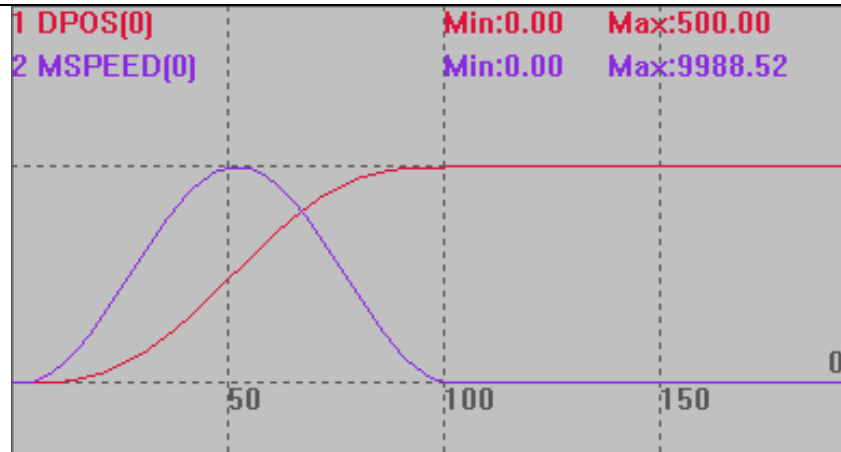
SPEEDval=100*2*PI*0.2*SIN(2*PI*0.2*timeadd+pi)

	<pre> ?"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 </pre> 
Instructions	MOVE_PTABS

MOVE_PTABS -- Unit Distance (Absolute)

Type	Special Motion Instruction
Description	Drive the motor to reach one certain position in a period time. Usually, PC will calculate corresponding coordinates in every period, then transfer it to controller. BASE assigned axis can be used. Motion speed = (DIS/TICKS)*1000units/s Don't run a long distance in a very short time, the pulse frequency maybe too high, causing motor stall. It is better to divide the long distance into pieces.
Syntax	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)

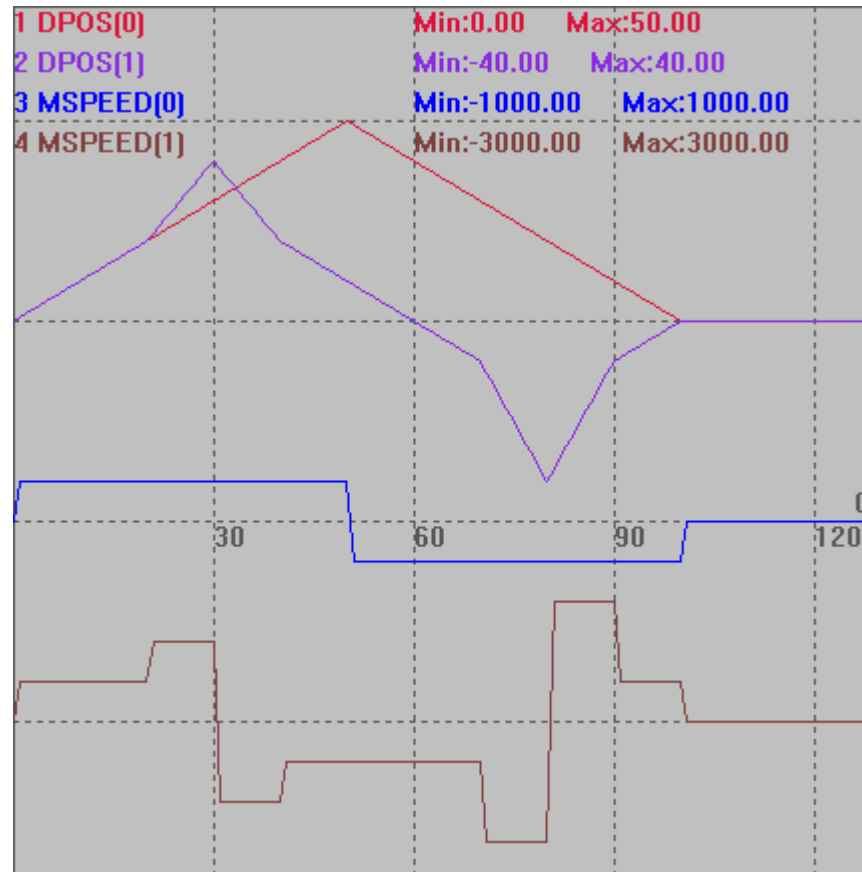
MOVE_PTABS (10,10,-10)

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)

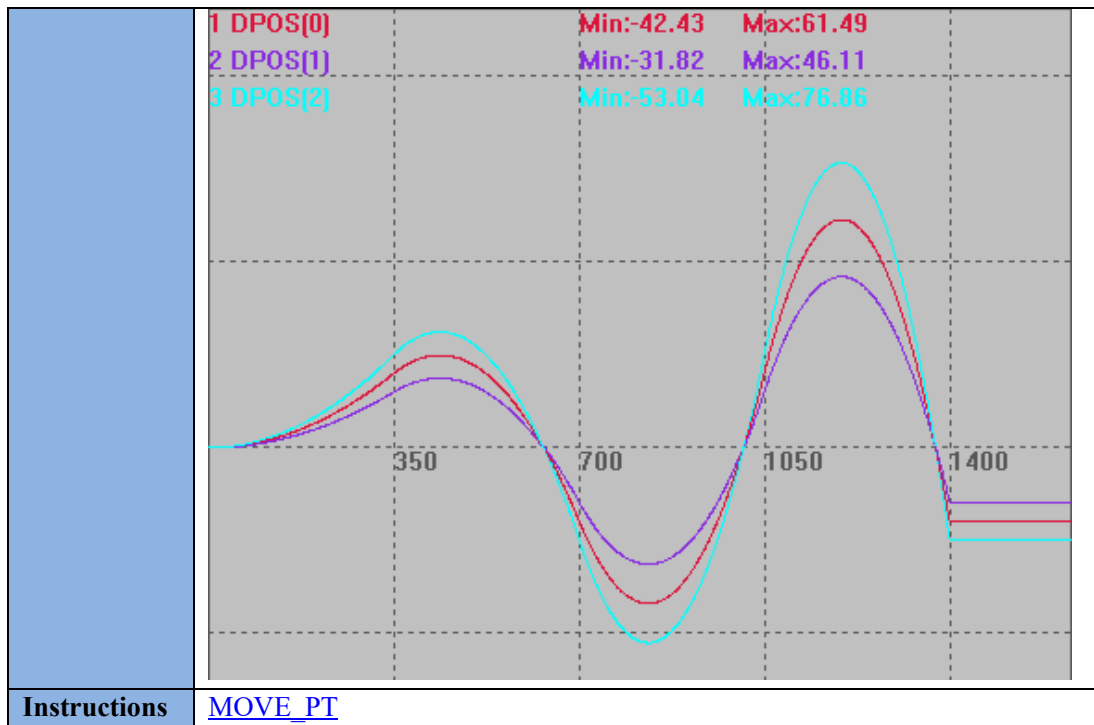
'in one period, axis 0, axis 1 and axis 2 run to position -20, -15, -25 separately

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)



MOVE_PVT -- Unit Distance (with Speed Planning)

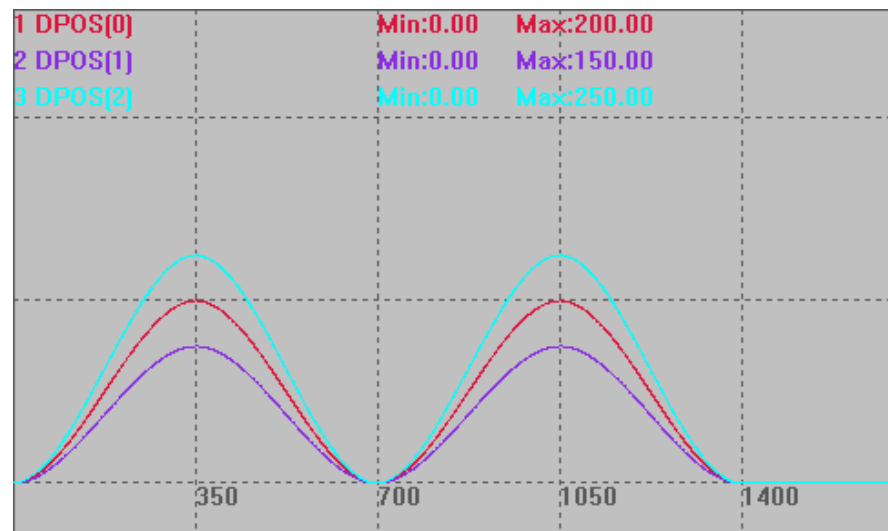
Type	Special Motion Instruction
Description	Set the distance of motor motion in a certain time, and it is with speed planning and can assign end speed. Speed in small distance will plan automatically according to former speed and end speed, as consecutive as possible. Usually, PC will calculate corresponding coordinates in every period, then transfer it to controller. BASE assigned axis can be used. Motion speed = (DIS/TICKS)*1000units/s Don't run a long distance in a very short time, the pulse frequency maybe too high, causing motor stall. It is better to divide the long distance into pieces.
Syntax	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 When the controller SERVO_PERIOD is 1000us, TICKS = 1ms (for different SERVO_PERIOD, TICKS are different).
Controller	General
Example	Example 1: BASE(0,1,2) UNITS=10,10,10,10


```

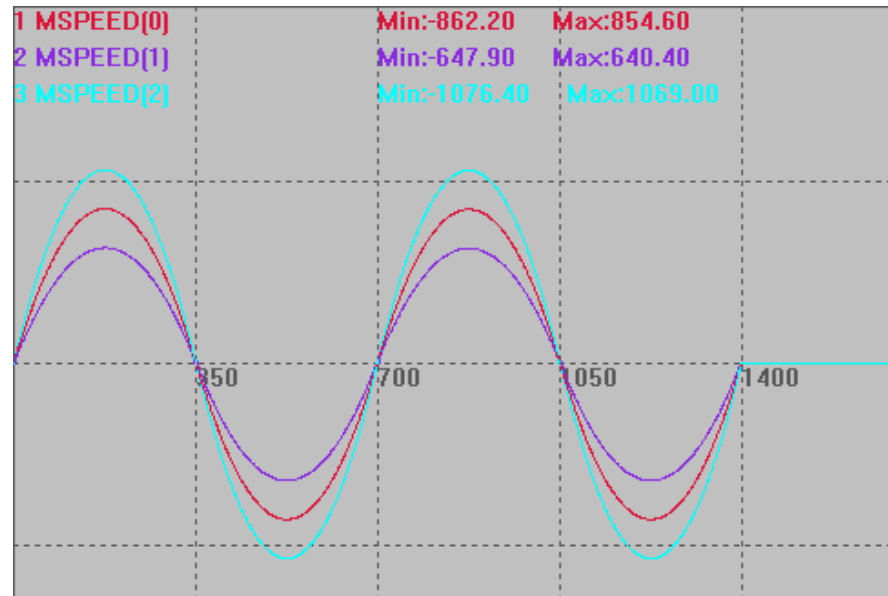
DPOS= 0,0,0
SPEED=10,10,10,10
ACCEL=1000,1000,1000,1000
DECEL=1000,1000,1000,1000
MERGE = ON
TRIGGER
MOVE_PVT(350,200,10,150,10,250,10)
'in one period, axis 0, axis 1 and axis 2 run 200, 150, 250 separately, end
speed is 10
MOVE_PVT(350,-200,10,-150,10,-250,10)
'in one period, axis 0, axis 1 and axis 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

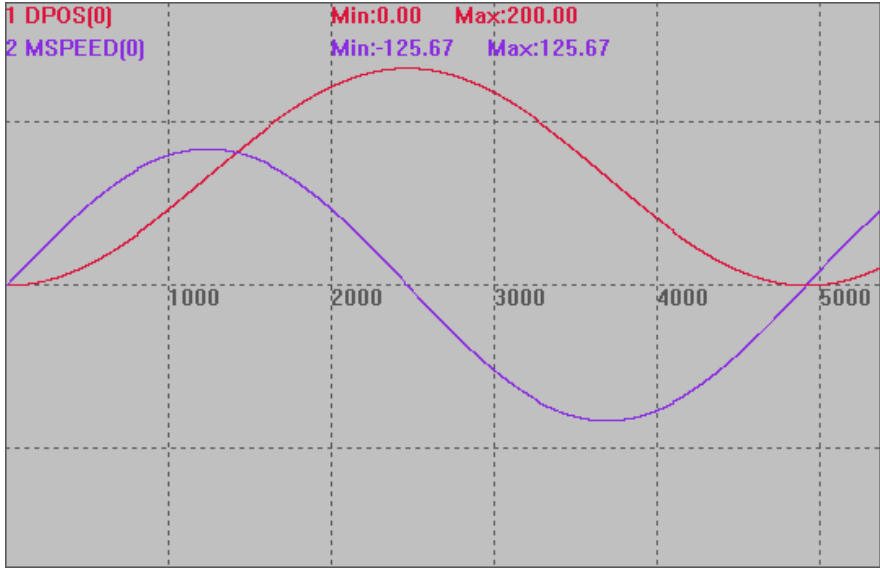
'period= $2\pi/|\omega|$

' $x=A*\cos(\omega x+\psi)+C$

'take the derivative to get the slope, which is the speed:

' $v=A*\omega*\sin(\omega x+\psi)$

val1=100*COS(2*PI*0.2*timeadd+pi)+100

	<pre> 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 </pre> 
Instructions	MOVE_PT

MOVE_PVTABS -- Unit Distance (Absolute & Speed Plan)

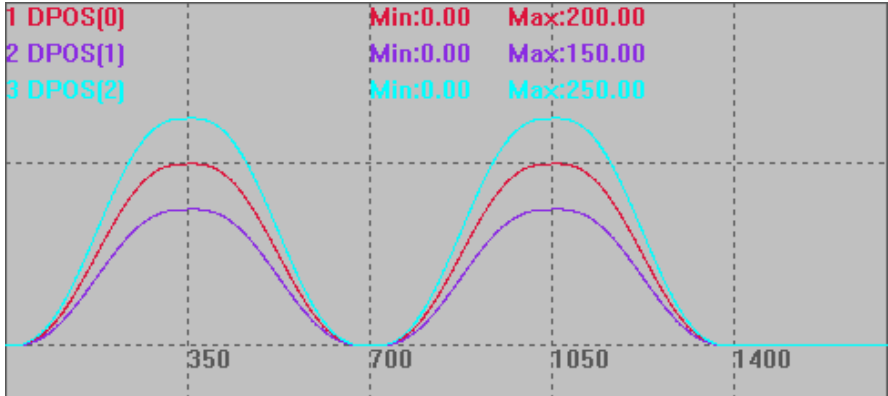
Type	Special Motion Instruction
Description	Set the distance of motor motion in a certain time, and it is with speed planning and can assign end speed. Speed in small distance will plan automatically according to former speed and end speed, as consecutive as possible. Usually, PC will calculate corresponding coordinates in every period, then transfer it to controller. BASE assigned axis can be used. Motion speed = (DIS/TICKS)*1000units/s Don't run a long distance in a very short time, the pulse frequency maybe too

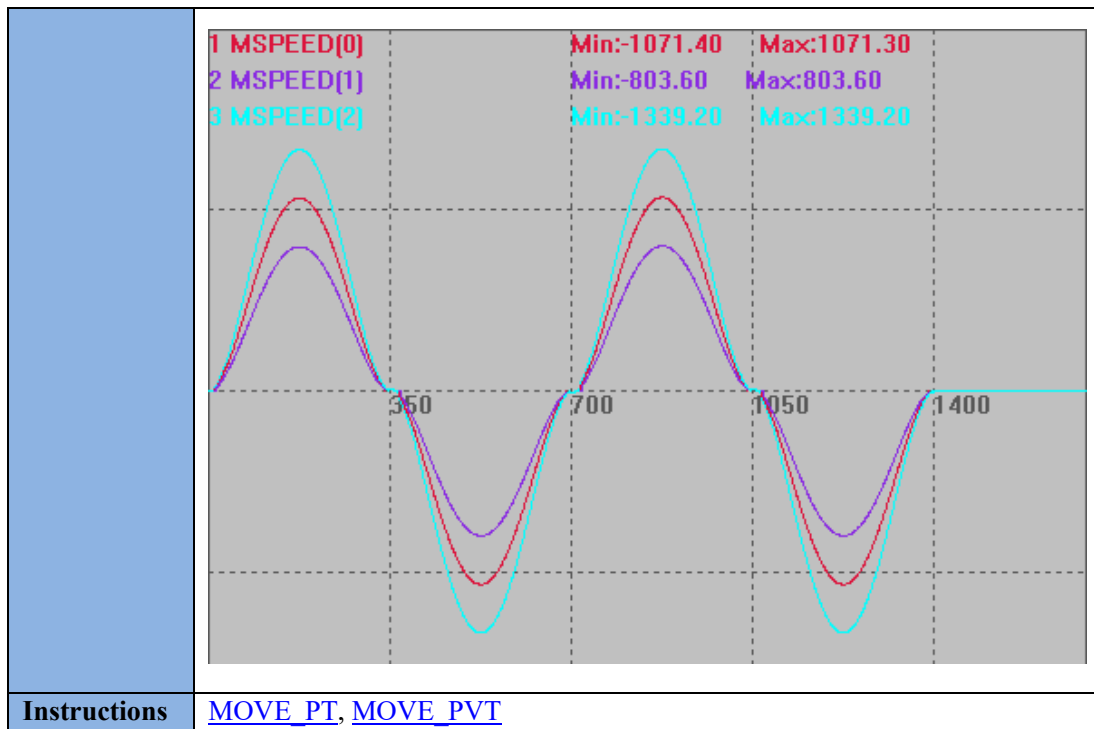
	high, causing motor stall. It is better to divide the long distance into pieces.
Syntax	MOVE_PVTABS (ticks, dis1, sp1, dis2, sp2...) ● ticks: servo period numbers of time ● dis1: motion absolute distance of the first axis ● sp1: the end speed when first axis moved ● dis2: motion absolute distance of the second axis ● sp2: end speed when second axis moved When the controller SERVO_PERIOD is 1000us, TICKS = 1ms (for different SERVO_PERIOD, TICKS are different).
Controller	General
Example	Example 1: <pre> 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 </pre>

	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 -- Unit Distance 2

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
Syntax	MOVE_PVTPP (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 When the controller SERVO_PERIOD is 1000us, TICKS = 1ms (for different SERVO_PERIOD, TICKS are different).

Controller	General
Example	Example 1: <pre> 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 </pre> DPOS(0), DPOS(1), DPOS(2) are 200 (vertical scale)  MSPEED(0), MSPEED(1) and MSPEED(2) are 1000 (vertical scale)



MOVE_PVTPPABS -- Unit Distance 2 (Absolute)

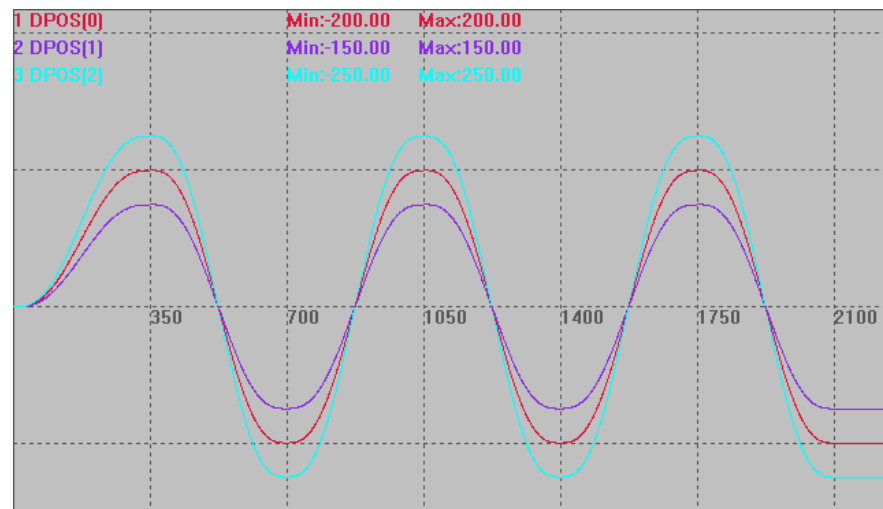
Type	Special Motion Instruction
Description	Same as MOVE_PVT, set the distance of motor motion in a certain time, and it is with speed planning. Accelerations of start and end moments can be sure as 0 through planning speed, and it can be used for point motions of assigned time. MOVE_PVT only can be 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
Syntax	MOVE_PVTPPABS (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 When the controller SERVO_PERIOD is 1000us, TICKS = 1ms (for different SERVO_PERIOD, TICKS are different).
Controller	General
Example	Example 1: BASE(0,1,2) UNITS=10,10,10,10

```

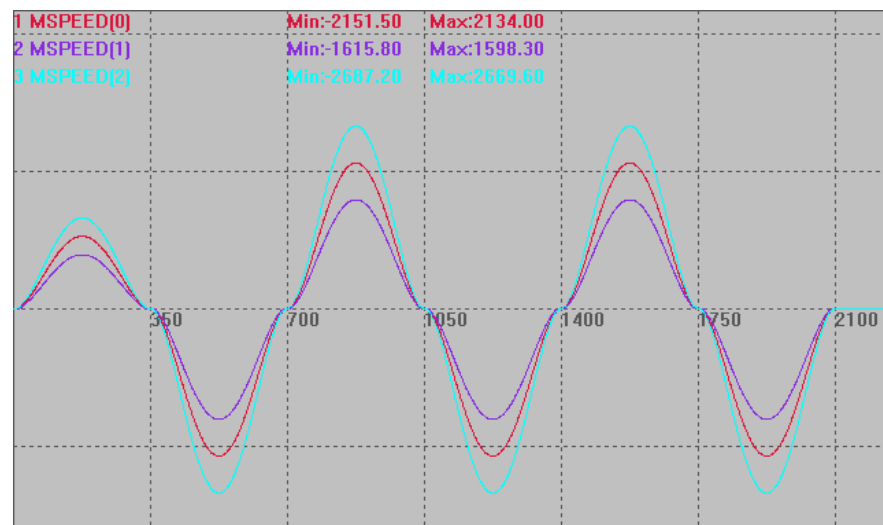
DPOS= 0,0,0,0
SPEED=10,10,10,10
ACCEL=1000,1000,1000,1000
DECEL=1000,1000,1000,1000
MERGE = ON
TRIGGER
MOVE_PVTPPABS(350,200,10,150,10,250,10)
'in one period, axis 0, axis 1 and axis 2 run 200, 150, 250 separately, end
speed is 10, acceleration decrease to be 0
MOVE_PVTPPABS(350,-200,-10,-150,10,-250,-10,10)
'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)



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 command dynamic modification, but it can use SPEED_RATIO to adjust speed, and VP_MODE configuration is valid.
Syntax	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 <pre> 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) </pre> 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

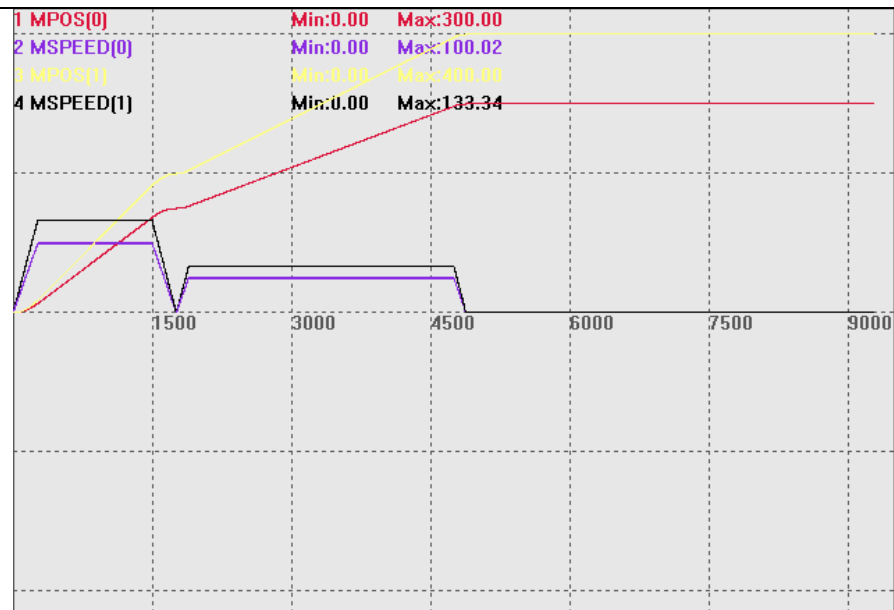
move_ptp(1,150,200)

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

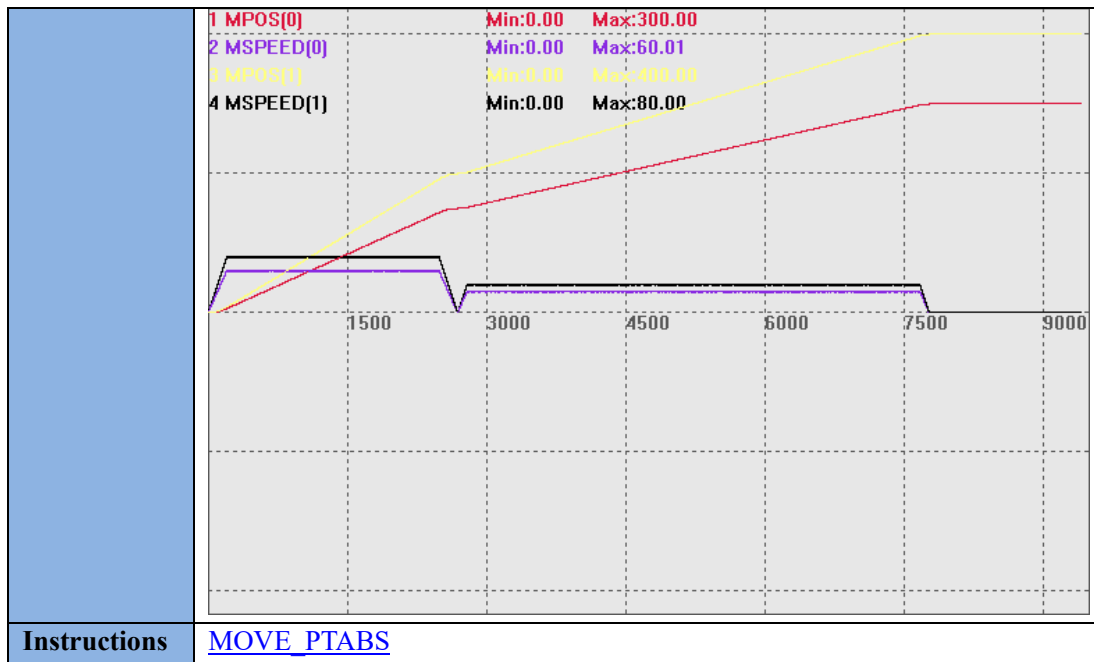
move_ptp(0,150,200) 'under mode 0, same effect as "MOVE" command

wait idle

speed_ratio = 0.5

move_ptp(0,150,200)

MPOS(0), MSPEED(0), MPOS(1), MSPEED(1), vertical scale (Y scale): 200



MOVE_PTPABS -- Point to Point (Absolute)

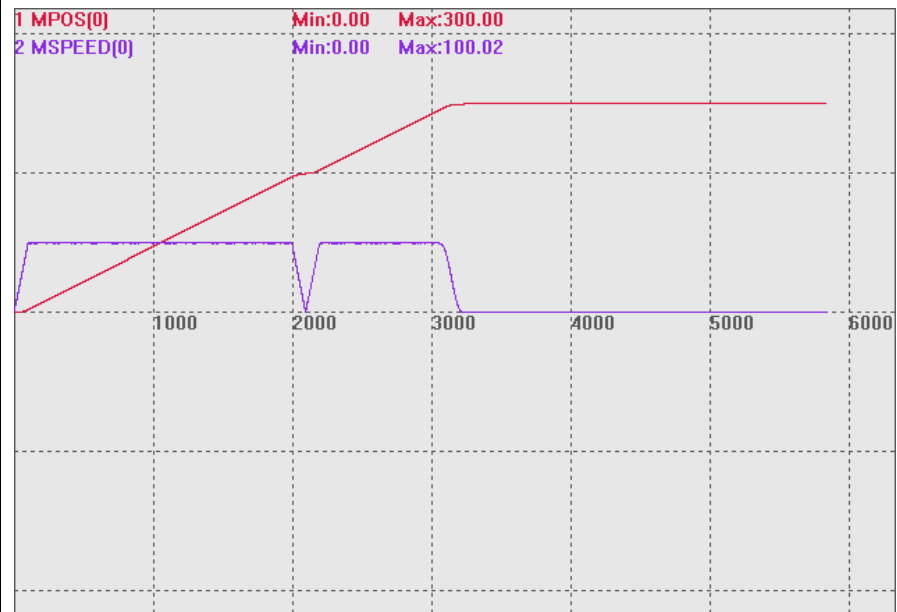
Type	Special Motion Instruction
Description	This is the linear interpolation motion command used for point-to-point motion. This command doesn't support speed ahead in continuous motion, and it uses each axis' SPEED, then speed parameter will enter motion buffer automatically. This command doesn't support command dynamic modification, but it can use SPEED_RATIO to adjust speed, and VP_MODE configuration is valid.
Syntax	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 absolute distance of first motion, unit: units, decimal - dis2: motion absolute distance of second motion, unit: units, decimal
Controller	Valid in ZMC4XX controllers with the version above “version_buid 230510”.
Example	Example 1: 'example of mode 1 single-axis and VP_MODE usage <pre> rapidstop(2) trigger speed_ratio = 1 base(0) mpos = 0 dpos = 0 atype = 1 </pre>

```

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

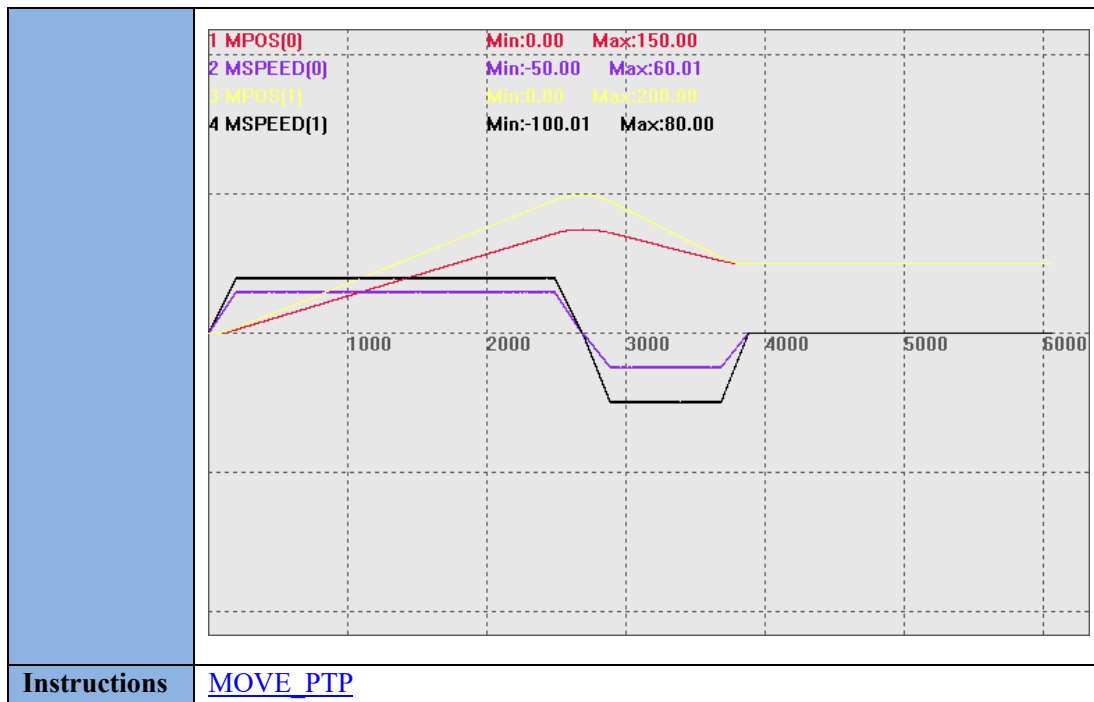
move_ptp(0,150,200) 'under mode 0, same effect as "MOVEABS" command

wait idle

speed_ratio = 0.5

move_ptpabs(0,100,100)

MPOS(0), MSPEED(0), MPOS(1), MSPEED(1), vertical scale (Y scale): 200



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, now the syntax is same as OP. ● Normal Mode The error is within one scan period, and all controllers support. ● High Precision Output Mode The error is within 1 microsecond. It is only for ZMC4XX controllers and above, the firmware should be after 20170421. <i>*notes of precision output mode*</i> 1) Not valid in all OP ports, only for the OP that supports HW hardware comparison output. 2) It is necessary to span one period between each precision output MOVE_OP, then it can take effect continuously, and in this gap, the new MOVE_OP will use normal mode automatically. If exceeds the span, new MOVE_OP also can take effect again. <u>That is, if there is no span time between continuous MOVE_OP commands, only the first one is valid.</u> 3) For some controllers, they can trigger several precision outputs at the same time, see <i>each manual</i> for details. For example, first 8 outputs of ZMC420SCAN all support precision output, and every output can use precision mode synchronously, but <u>when several precision OPs are used, only DPOS demand position can be compared (MPOS comparison is only valid in single OP) and MOVE_OP > 10 is not recommended.</u> 4) If your controller precision output OPs are isolate (please check parameter

	“max hwout” of “?*max” in RTSys output window, if there is “isolate”, which means these OPs are isolate, one OP is related to one axis for precision output, like: [max hwout:4, 32, isolate]), it still can achieve correct precision outputs when several axes MOVE_OP are opened. However, if these outputs are not isolate, and when multi-axis MOVE_OP is enabled, precision outputs maybe not triggered or triggered wrongly. 5) MOVE_OP precision function is used for the main axis of BASE (the first axis in BASE), that is, <u>in multi-axis interpolation, it can't ensure precision position output effect for the slave axis whose axis type ATYPE is different from BASE spindle ATYPE.</u> 6) Different precision parameters (like, open precision output, delay output, etc.) can be set (before calling MOVE_OP), then it can achieve “us” level control for laser power. 7) For ZMC4XX above controllers, and the firmware version after 20241029, MOVE_OP will not influence smoothing. 8) To check whether the encoder is arrived in precision output, please find the near position, that is, when one certain axis produces bigger effect, then you need to set this axis' micro stepping as bigger value, in this way, precision can be promoted.
Syntax	Syntax 1: MOVE_OP ([ionum],value) ionum: output No., which starts from 0 value: output status, when multiple outputs are operated, each port's states are shown bit by bit. Syntax 2: MOVE_OP (ionum1, ionum2,value[,mask]) ionum1: the first output channel to be operated ionum2: the last output channel to be operated value: output status, when multiple outputs are operated, each port's states are shown bit by bit. mask: set values by bit, and set which IOs to be operated, if it is not filled, all channels (from the first to the last) will 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

Make vertical offsets to view trajectory clearer:

DPOS(0) vertical scale 1000

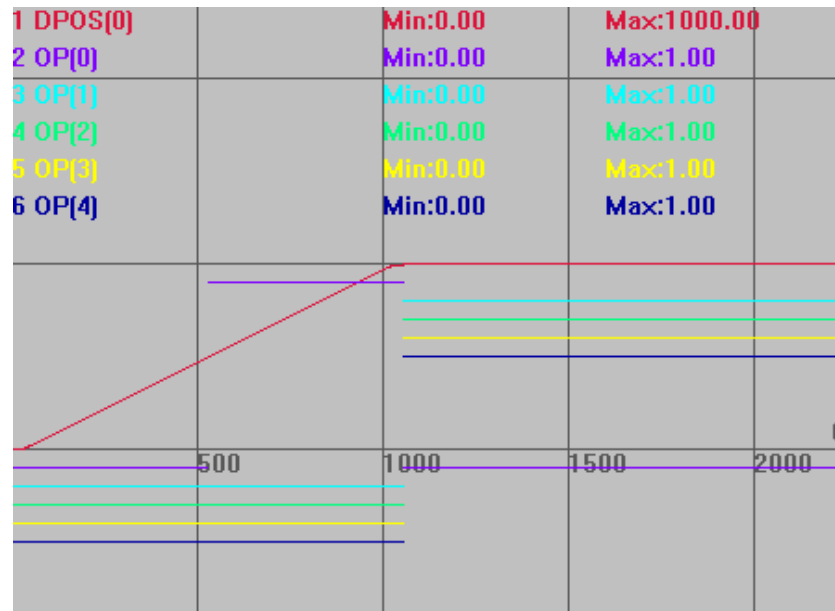
OP(0) vertical 1, offset -0.1

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)

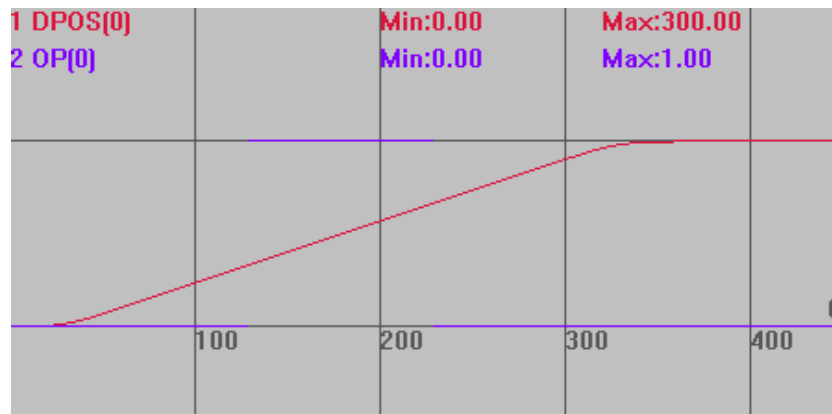
'precision takes effect

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, real encoder wiring is needed.

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)

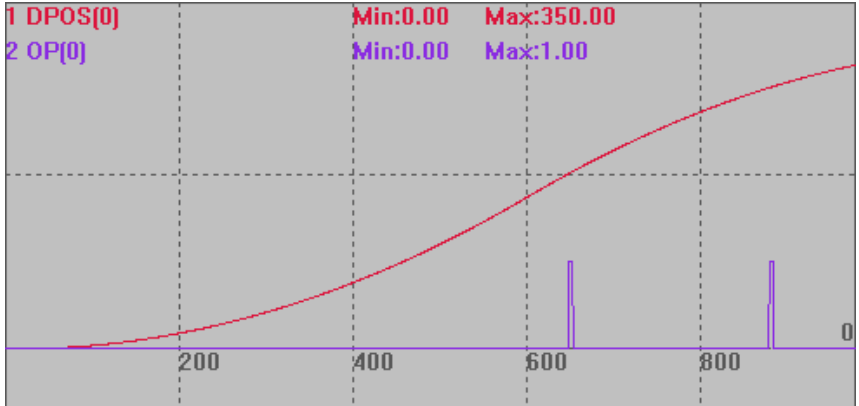
MOVE(50)

END

Path Curve:

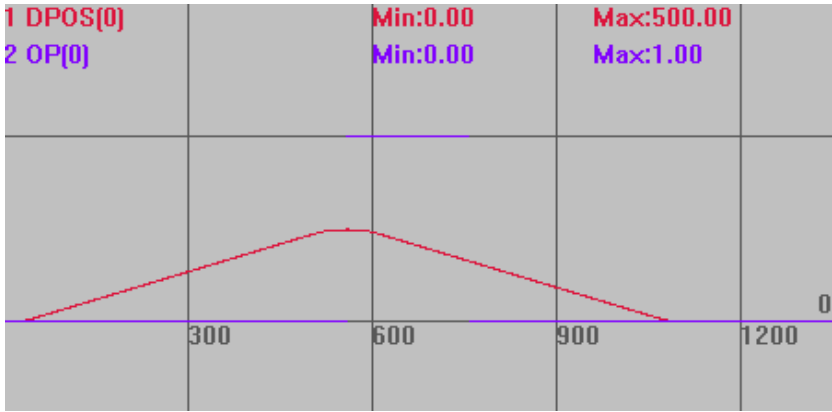
DPOS(0) vertical scale 200

OP(0) vertical scale 2

	
Instructions	<u>OP</u> , <u>MOVE_OP2</u> Precision Output Mode: <u>SYSTEM_ZSET</u> , <u>AXIS_ZSET</u>

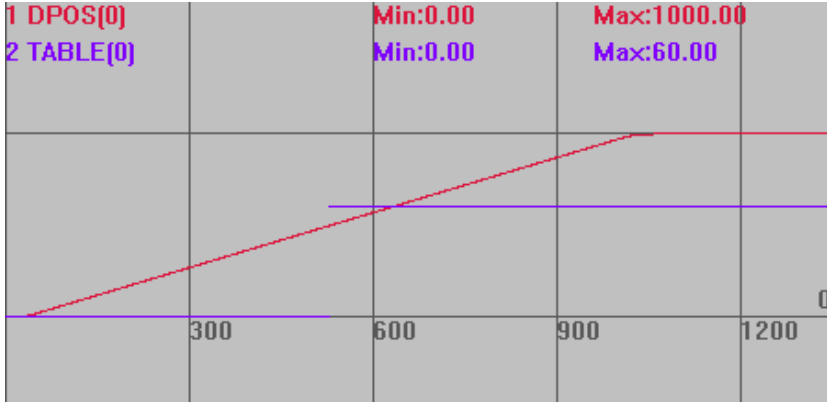
MOVE_OP2 -- Output in Buffer 2

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.
Syntax	MOVE_OP2(ionum,state,offtimems) ● ionum: output port No., starting from 0 ● state: output state. ● offtimems: the period after which signal inverts. (ms)
Controller	General
Example	<pre> 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 is finished. This instruction will not obstruct the execution of next instruction. MOVE(-500) Motion Path: DPOS(0) vertical scale 1000 </pre>

	OP(0) vertical scale 1 
Instructions	OP , MOVE_OP

MOVE_TABLE -- Table in Buffer

Type	Special Motion Instruction	
Description	Add one TABLE to motion buffer based on BASE axis motion. When this command is executing, there is no any motion, only TABLE is modified. MTYPE of this command is consistent with MOVE_OP command. For ZMC4XX controllers (above version_build 20241029), it will not affect smoothing effect via MOVE_TABLE.	
Syntax	MOVE_TABLE(tablenum, value) ● tablinum: TABLE No. ● value: the value to be modified.	
Controller	General	
Example	<pre> 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 </pre>	

	
Instructions	TABLE

MOVE_PARA -- Parameters in buffer

Type	Special Motion Instruction
Description	Modify motion buffer parameters based on BASE axis motion. When LOAD is executed in the buffer, only modifies parameters. Under this command, MTYPE and MOVE_OP are the same. Notes: ➤ MOVE_PARA command is only used to modify global parameters, not for parameters that will be brought into motion buffer, such as, FORCE_SPEED, CORNER_MODE, INTPEACTOR, CORNER_ACC. ➤ Under CORNER_MODE fractionated mode, ACCEL and DECEL will be entered into buffer, so MOVE_PARA can't be used to modify it. ➤ To ensure that the parameter values are correctly brought into the motion buffer, please modify by calling the parameters directly before the MOVE in the motion buffer. If you try to modify these parameters after the MOVE, it may cause randomness in the parameter values, affecting the motion.
Syntax	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 <pre> 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 </pre> Example 2: Single-Axis Speed Changing

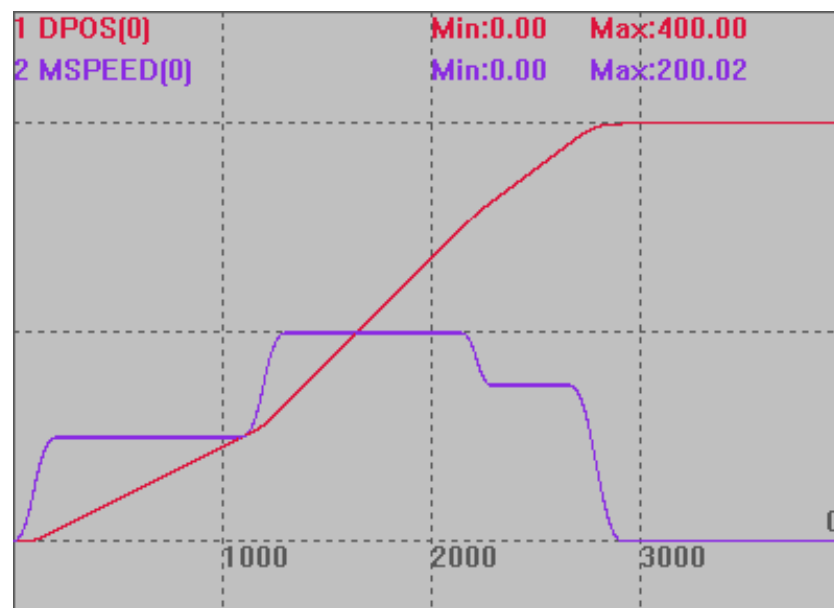
```

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

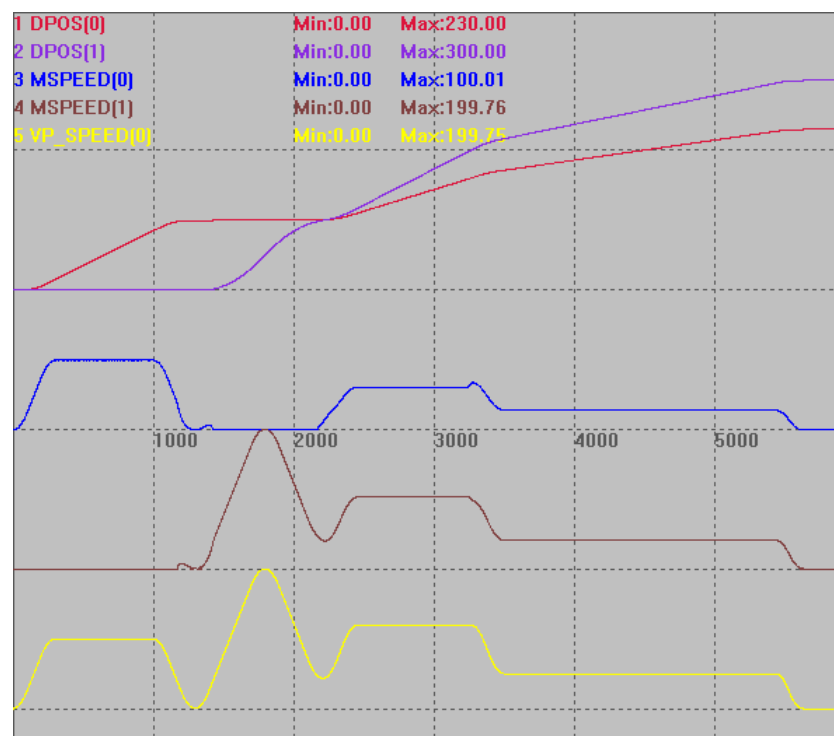
MOVE_PARA(SPEED,0,120)
MOVE(60,100)             'motion angle is 30.96, which is within 15~45,
                          'proportional deceleration

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. PWM is turned-off by setting duty cycle as 0 (not to set PWM frequency to 0), PWM frequency should be modified before PWM switch setting.
Syntax	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, turn off pwm. In one period, output valid electrical level at first, invalid signals will be output next. ● freq: the frequency, the default value is 1KHz, for hardware, up to 1MHz, for software, up to 2KHz.
Controller	With firmware version above 20170503
Routine	<pre> 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 </pre>
Instructions	PWM_DUTY , PWM_FREQ

MOVE_SYNMOVE -- Synchronous Axis Trigger in Buffer

Type	Special Motion Instruction
Description	Tigger other axes in motion buffer based on BASE, the present axis waits. MTYPE of this instruction is same as MYTPE value of MOVE_DELAY.
Syntax	MOVE_SYNMOVE(AXISNUM,DIS[,IFSP]) ● axisnum: the synchronous axis No.

	● dis: the relative motion distance ● ifsp: use SP function or not, default: No.
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_ASYNMOVE -- Synch Axis Trigger in Buffer 2

Type	Special Motion Instruction
Description	Tigger other axes in motion buffer based on BASE, the present axis doesn't wait for it. MTYPE of this instruction is same as MYTPE value of MOVE_DELAY.
Syntax	MOVE_ASYNMOVE(AXISNUM,DIS[,IFSP]) ● axisnum: the synchronous axis No. ● dis: the relative motion distance ● ifsp: use SP function or not, default: No.

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_ASYNMOVE(1,100,0) 'axis1 starts to move when axis 0 reaches 100. MOVE(100) 'axis 0 continues to move Motion Path: DPOS(0) vertical scale 200 DPOS(1) vertical scale 200
Instructions	MOVE_SYNMOVE

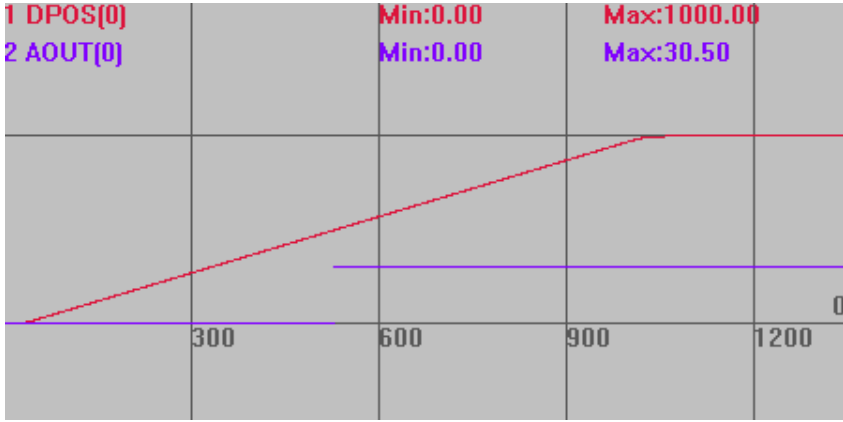
MOVE_TASK -- Start Task in Buffer

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.
Syntax	MOVE_TASK(tasknum, label) - tasknum: task No. - label: function name or label
Controller	General

Example	<pre> 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 </pre>
Instructions	RUNTASK , RUN

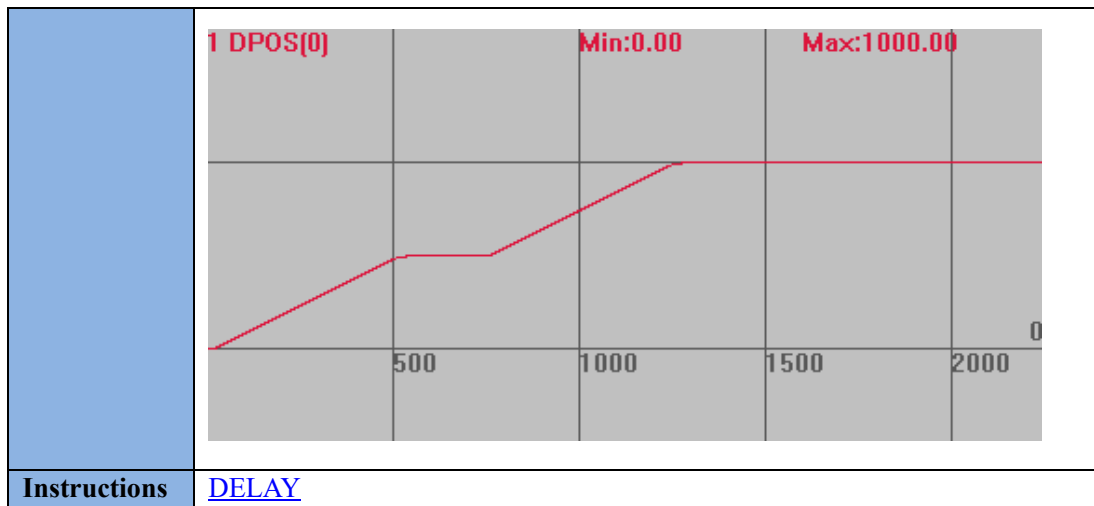
MOVE_AOUT -- Analog Signal in Buffer

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.	
Syntax	<pre>MOVE_AOUT(danum, value)</pre> - danum: DA No. - value: value to be modified.	
Controller	General, controller with DA channels.	
Example	<pre> 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. </pre>	

	Motion Path: DPOS(0) vertical scale 1000 AOUT(0) vertical scale 100 
Instructions	AOUT

MOVE_DELAY -- Delay in buffer

Type	Special Motion Instruction
Description	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.
Syntax	MOVE_DELAY(timems) / move_wa(timems) timems: delay time, ms as units.
Controller	General
Example	<pre> 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 </pre>



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.
Syntax	MOVE_WAIT(paraname, paranum, eq, value) ● paraname: choose parameter's name (parameters 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 BIT based parameters, like, IN, etc. 0 = Not recommended ● value: comparison value.
Controller	Valid in firmware version after 150802, or after XPLC160404.
Example	<pre> 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 </pre>

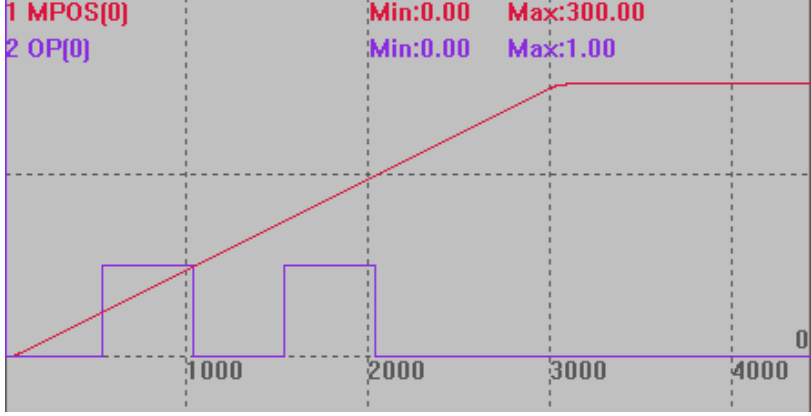
	IN(0) vertical scale 1			
Instructions	WAIT_UNTIL			

MOVE_CANCEL -- Stop Buffer

Type	Special Motion Instruction
Description	Add CANCEL in motion buffer
Syntax	MOVE_CANCEL(iaxis, imode) ● iaxis: the axis to operate ● imode: select CANCEL mode, same as CANCEL instruction
Controller	General
Example	MOVE_CANCEL(1,0) 'the command of stop axis1 is written in axis 0 buffer
Instruction	CANCEL

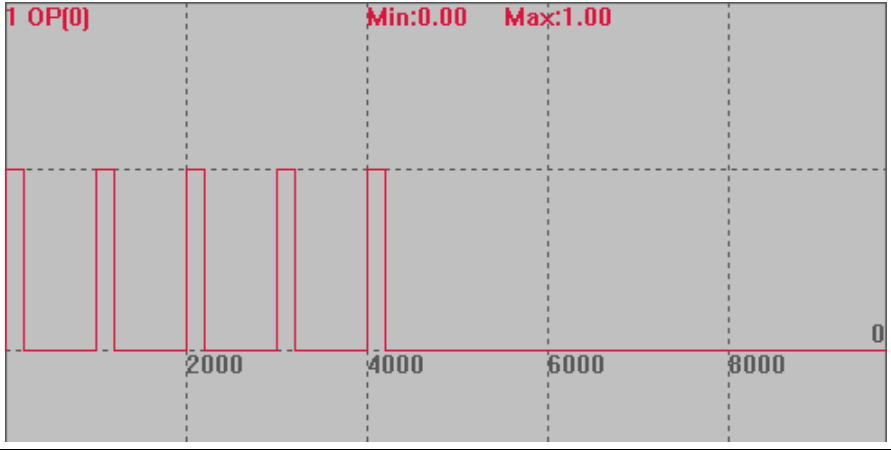
MOVE_HWPSWITCH2 – Hard Comparison Out in Buffer

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.
Syntax	MOVE_HWPSWITCH2(mode,[...]) ● mode: open different comparers, parameters to be filled also different. Refer to HW_PSWITCH2 command.
Controller	Controllers that support HW.
Example	BASE(0) UNITS=100 SPEED=100 ACCEL=1000 DECEL=1000 DPOS=0

	<pre> 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 </pre> Compare Output: Same output effect as HW_PSITCH2, but this example enters 3 commands into buffer. MPOS(200) vertical scale OP(0) vertical scale 
Instruction	HW_PSITCH2

MOVE_HWTIMER -- Hardware Timer in Buffer

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 affected.
Syntax	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 (controller should support HW/HW2 function)
Example	<pre> BASE(0) ATYPE=1 UNITS=100 </pre>

	<pre> 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 </pre> 
Instruction	HW_TIMER , HW_PSWITCH2

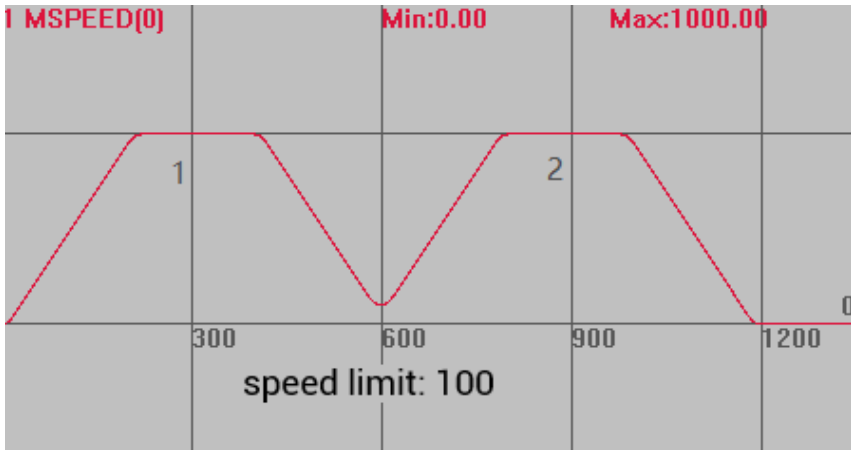
MOVE_ADDAX -- Motion Superposition

Type	Single Axis Motion Instruction
Description	Motion superposition: add motion of one axis to another axis; it supports BASE_MOVE, in this way, destaxis can be any adjusted. When using MOVE_ADDAX to realize superposition. the added value is not units but the pulses. <u>Conversion relationship:</u> distance of superimposing axis *units of superimposing axis /units of superimposed axis = distance of superimposed axis <u>For example:</u> If UNITS of axis A is 100, of Axis B is 50, make their superposition move 100. <i>Situation 1:</i> add motion of axis A to axis B, no axis A moves 100, then the axis B moves $100 \times 100 / 50 = 200$ <i>Situation 2:</i> add motion of axis B to axis A, now axis B moves 100, then the axis A moves $100 \times 50 / 100 = 50$. Motions can't be added to each other simultaneously between 2 axes, that is, when adding motion of axis A to axis B, B can't be added to axis A any more. 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 superimposing axis' acceleration and deceleration, and the ratio of the units of the two axes. MOVE_ADDAX has no effect when the axis MTYPE is FRAME or REFRAME.
Syntax	MOVE_ADDAX(srcaxis,[imode], [para]) - destaxis: superposing target axis No. - srcaxis: superposing source axis' No. - imode: superposition mode 0: default value, single-axis superposition, it is compatible with “previous superposition of direct pulses adding”. 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 / rationout 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 respectively by MOVE_ADDAX(-1, 3, 0) or MOVE_ADDAX(-1) AXIS (the superposed axis No.) 4: SCAN linked 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) MOVE_ADDAX(srcaxis, 4, srcaxis2) Use srcaxis to compensate destaxis, use srcaxis2 to compensate destaxis2. Note: two axes should be cancelled together, MOVE_ADDAX(-1, 4, -1) or

	MOVE_ADDAX(-1) AXIS (superposed axis No.) 5: SCAN linked 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) MOVE_ADDAX(srcaxis, 5, srcaxis2) srcaxis is inversely superposed at destaxis, srcaxis2 is inversely superposed at destaxis2. Note: two axes should be cancelled together, MOVE_ADDAX(-1, 5, -1) or MOVE_ADDAX(-1) AXIS (superposed axis No.) BASE_MOVE supported: BASE_MOVE = moveaxis BASE (detaxis) MOVE_ADDAX (srcaxis, [imode], [para]) BASE_MOVE = -1
Controller	Valid in ZMC4XX controllers with 20220708 firmware version or above.
Example	Example 1: For more mode description, see the ADDAX command BASE(0,1) 'select axis No. 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

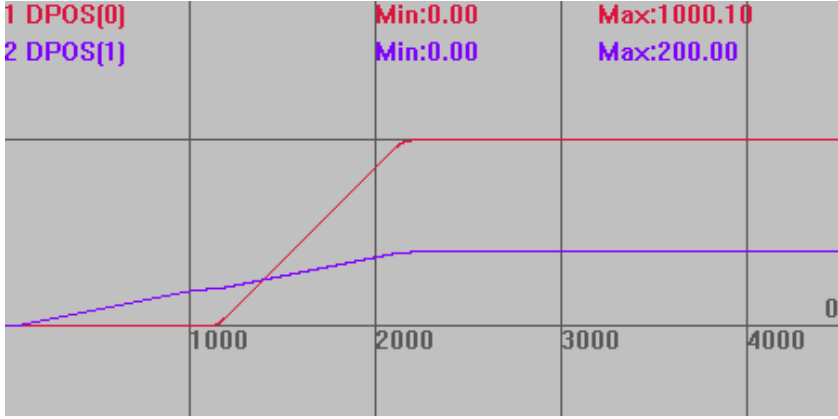
Type	Special Motion Instruction
Description	Add speed limit to the end of present motion, in order to force the axis to decelerate at the turning corner.
Syntax	MOVELIMIT(limitspeed) ● limitspeed: the limit speed value.
Controller	General
Example	<pre> BASE(0) UNITS=100 DPOS=0 SPEED=1000 'axis speed 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) </pre> 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

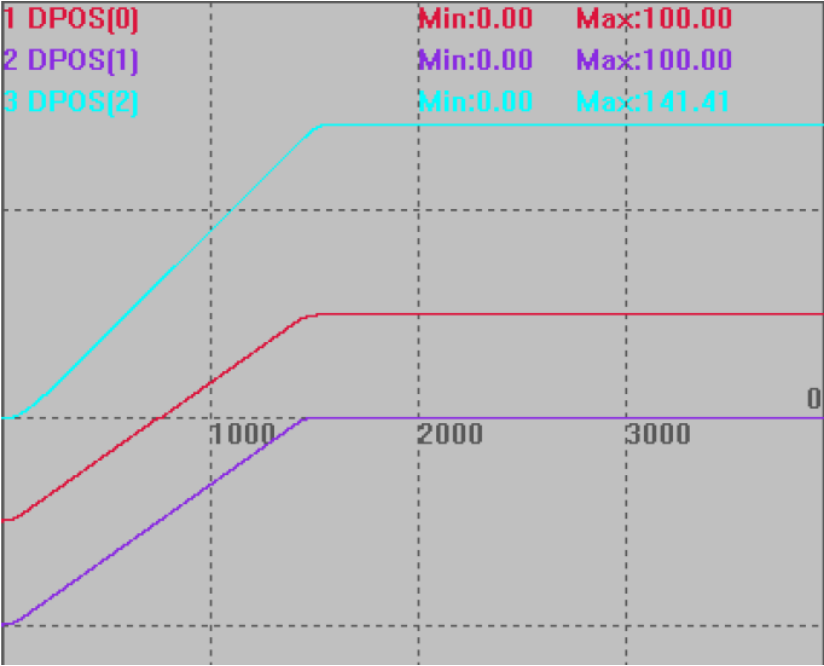
CONNECT -- Synchronization Motion

Type	Synchronization Motion Instruction
Description	Target position of present axis and measured position of driving_axis will be linked by electronic gearbox. Generally, it is used for handwheel. The link relationship is calculated in pulse amount, so please consider different axes' UNITS. And if you want to make it disconnect, CANCEL can do that. ■ Example: suppose UNITS of axis0 is 100, UNITS of linked axis1 is 10, the ratio is 1. ■ Result: after linking them by CONNECT, when axis0 moves 100 (s1=100), the axis1 will move 10 [s1*UNITS(0)*ratio/UNITS(1)]. Note: ratio can be changed dynamically by calling instruction.
Syntax	CONNECT (ratio, driving_axis) AXIS (connecting axis No.) ● ratio: the ratio of pulses, can be negative or positive. ● driving_axis: axis No. of linked axis, for handwheel, it is encoder axis.
Controller	General
Example	<pre> 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 MOVE(100) AXIS(1) </pre> 'trigger the oscilloscope automatically 'axis 1 moves 100, now axis 0 doesn't move

	WAIT IDLE(1) CONNECT(1,1) AXIS(0) 'axis0 is linked to axis1, ratio is 1. MOVE(100) AXIS(1) 'axis 1 moves 100, then axis 0 moves 1000. Motion Path DPOS(0) vertical scale 1000 DPOS(1) vertical scale 500 
Instructions	CLUTCH_RATE , CONNPATH

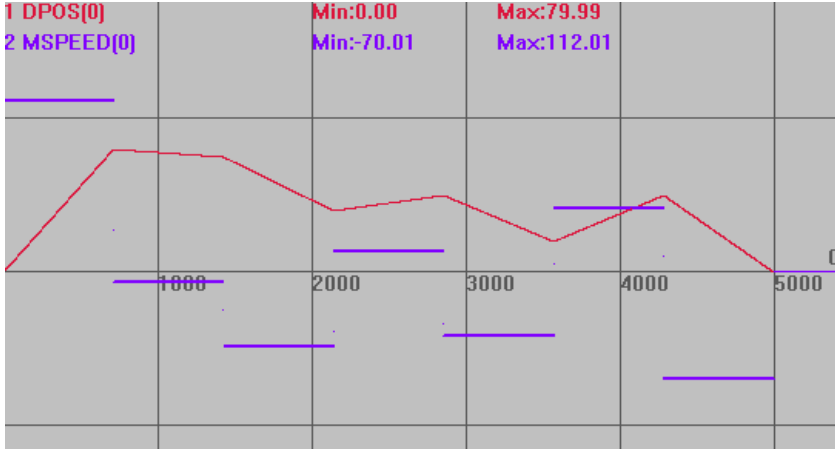
CONNPATH -- Synchronization Motion 2

Type	Synchronization Motion Instruction
Description	Target position of present axis and interpolation vector length of driving_axis will be linked by electronic gearbox. And it needs to connect to interpolation spindle, if it connects to slave axis, the effect will be same as CONNECT. The link relationship is calculated in pulse amount, so please consider different axes' UNITS. And if you want to make it disconnect, CANCEL can do that. Note: ratio can be changed dynamically by calling instruction.
Syntax	CONNPATH (ratio, driving_axis) AXIS (connecting axis No.) ● ratio: the ratio of pulses, can be negative or positive. ● driving_axis: axis No. of linked axis, for handwheel, it is encoder axis.
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

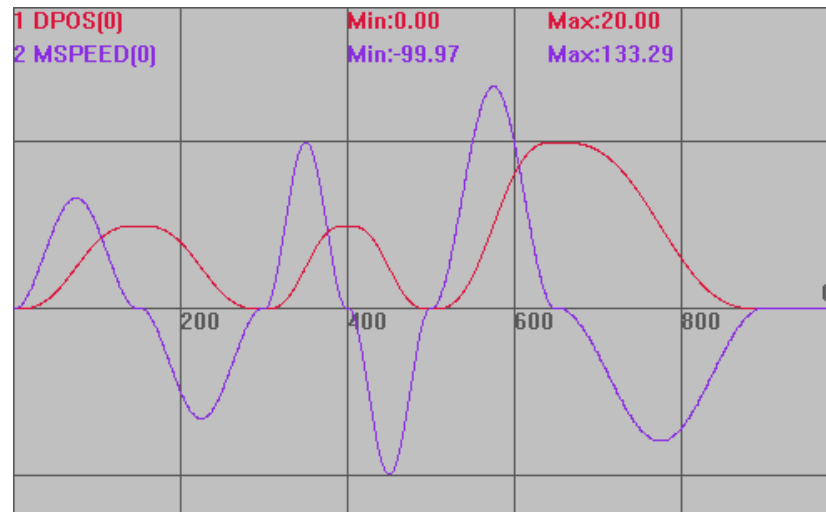
CAM -- Cam Table Motion

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.
Syntax	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. <i>total motion time=distance/Axis SPEED.</i>
Controller	General
Example	EXAMPLE 1: RAPIDSTOP(2) WAIT IDLE(0) 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) 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,SPEED*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 0-360° 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

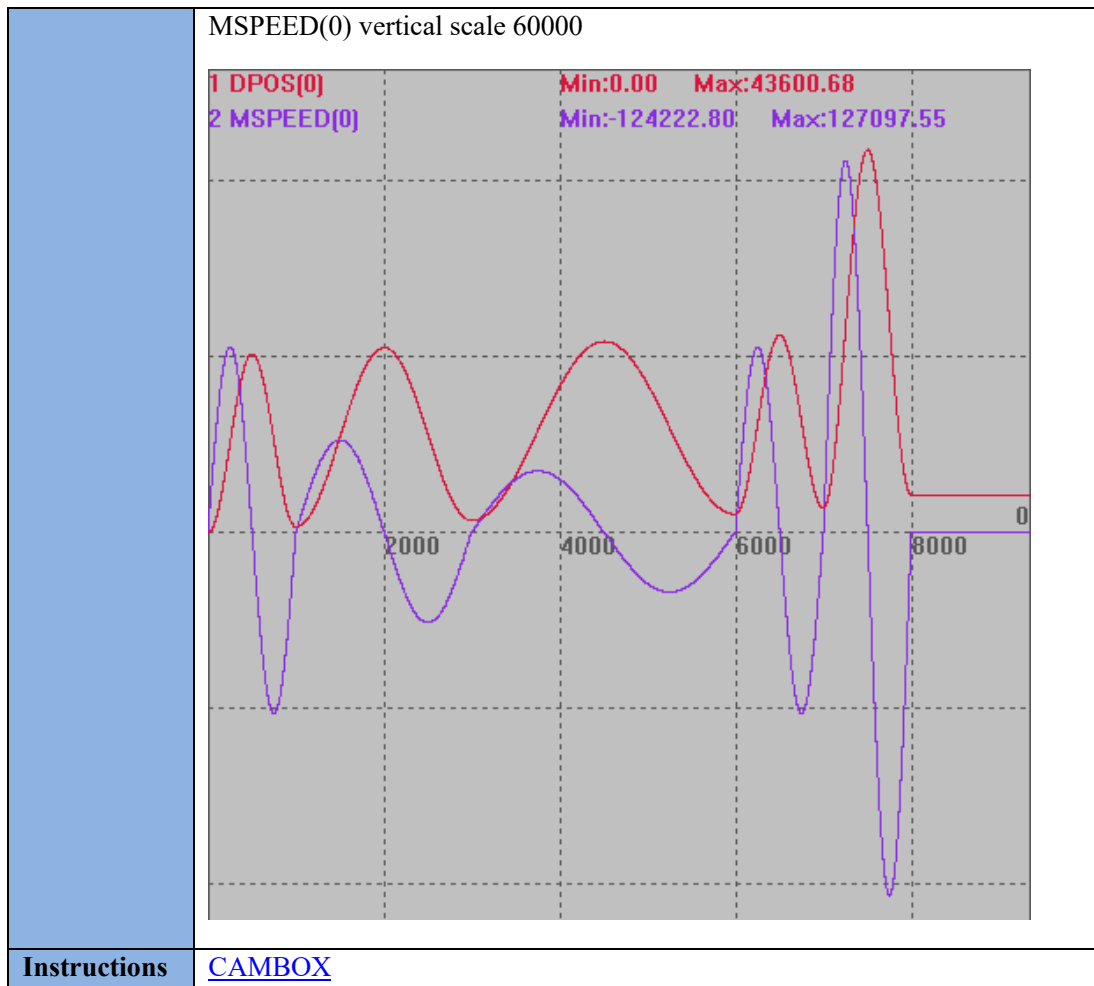
WAIT UNTIL REMAIN_BUFFER(1) > 0

'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



CAMBOX -- Following Cam Table Motion

Type	Synchronization Motion Instruction
Description	CAMBOX determines the axis motion according to TABLE data (data value means motion trajectory, which is related to motion starting point), and the following axis refers to the main axis. Note: one same TABLE data can be used and operated by two or more CAMBOX instructions. ■ The total motion time of the current axis is determined by motion distance and axis speed of reference axis, and the speed is matched automatically. ■ TABLE data is set manually, the first data is the index point (0 is recommended). ■ Table data * table multiplier = UNITS (pulses sent) ■ Please ensure the command distance parameter * units is the integer pulse, otherwise it will cause floats (minor errors).
Syntax	CAMBOX(start_point, end_point, table_multiplier, link_distance, link_axis[, link_options][, link_pos][, link_offpos]) ● start point: TABLE No. of start point, saving position of first point.

	● end point: TABLE No. of end point. ● table multiplier: position * this value, usually this value is set to the value that is the same as UNITS ● link_distance: motion distance of reference axis. ● link_axis: axis No. of reference axis. ● link_options: the link mode (connect to reference axis), there are several modes (of corresponding binary system).																				
	<table><tr><th>Bit</th><th>Meaning</th></tr><tr><td>bit 0</td><td>When the MARK signal event of the reference axis is triggered, the current axis starts coupled motion with the reference axis.</td></tr><tr><td>bit 1</td><td>When the reference axis moves to set absolute position, the current axis starts coupled motion with the reference axis.</td></tr><tr><td>bit 2</td><td>It will automatically and continuously repeat the bidirectional operation (it can be cancelled by REP_OPTION=1).</td></tr><tr><td>bit 3</td><td>Acceleration smoothing (S curve smoothing).</td></tr><tr><td>bit 4</td><td>Start from one certain middle position, then “power-off interrupt” is used together to resume “cam”.</td></tr><tr><td>bit 5</td><td>It is connected when the reference axis is in forward motion.</td></tr><tr><td>bit 8</td><td>When the MARKB signal event of the reference axis is triggered, the current axis starts coupled motion with the reference axis; the latch axis No. is reference axis No.; please use the latest firmware.</td></tr><tr><td>bit 9</td><td>When the MARKC signal event of the reference axis is triggered, the current axis starts coupled motion with the reference axis; the latch axis No. is reference axis No.; please use the latest firmware; this is valid in controllers that support 4-channel latching.</td></tr><tr><td>bit 10</td><td>When the MARKD signal event of the reference axis is triggered, the current axis starts coupled motion with the reference axis; the latch axis No. is reference axis No.; please use the latest firmware; this is valid in controllers that support 4-channel latching.</td></tr></table>	Bit	Meaning	bit 0	When the MARK signal event of the reference axis is triggered, the current axis starts coupled motion with the reference axis.	bit 1	When the reference axis moves to set absolute position, the current axis starts coupled motion with the reference axis.	bit 2	It will automatically and continuously repeat the bidirectional operation (it can be cancelled by REP_OPTION=1).	bit 3	Acceleration smoothing (S curve smoothing).	bit 4	Start from one certain middle position, then “power-off interrupt” is used together to resume “cam”.	bit 5	It is connected when the reference axis is in forward motion.	bit 8	When the MARKB signal event of the reference axis is triggered, the current axis starts coupled motion with the reference axis; the latch axis No. is reference axis No.; please use the latest firmware.	bit 9	When the MARKC signal event of the reference axis is triggered, the current axis starts coupled motion with the reference axis; the latch axis No. is reference axis No.; please use the latest firmware; this is valid in controllers that support 4-channel latching.	bit 10	When the MARKD signal event of the reference axis is triggered, the current axis starts coupled motion with the reference axis; the latch axis No. is reference axis No.; please use the latest firmware; this is valid in controllers that support 4-channel latching.
	Bit	Meaning																			
	bit 0	When the MARK signal event of the reference axis is triggered, the current axis starts coupled motion with the reference axis.																			
	bit 1	When the reference axis moves to set absolute position, the current axis starts coupled motion with the reference axis.																			
	bit 2	It will automatically and continuously repeat the bidirectional operation (it can be cancelled by REP_OPTION=1).																			
	bit 3	Acceleration smoothing (S curve smoothing).																			
	bit 4	Start from one certain middle position, then “power-off interrupt” is used together to resume “cam”.																			
	bit 5	It is connected when the reference axis is in forward motion.																			
	bit 8	When the MARKB signal event of the reference axis is triggered, the current axis starts coupled motion with the reference axis; the latch axis No. is reference axis No.; please use the latest firmware.																			
	bit 9	When the MARKC signal event of the reference axis is triggered, the current axis starts coupled motion with the reference axis; the latch axis No. is reference axis No.; please use the latest firmware; this is valid in controllers that support 4-channel latching.																			
	bit 10	When the MARKD signal event of the reference axis is triggered, the current axis starts coupled motion with the reference axis; the latch axis No. is reference axis No.; please use the latest firmware; this is valid in controllers that support 4-channel latching.																			
● link_pos: when link_options is set as 2, this is the absolute position of “connect & start”. ● link offpos: when bit4 of link_options is set as 1, this is the relative distance that has been executed by master axis.																					
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          'set how many segments, more segments – more stable
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

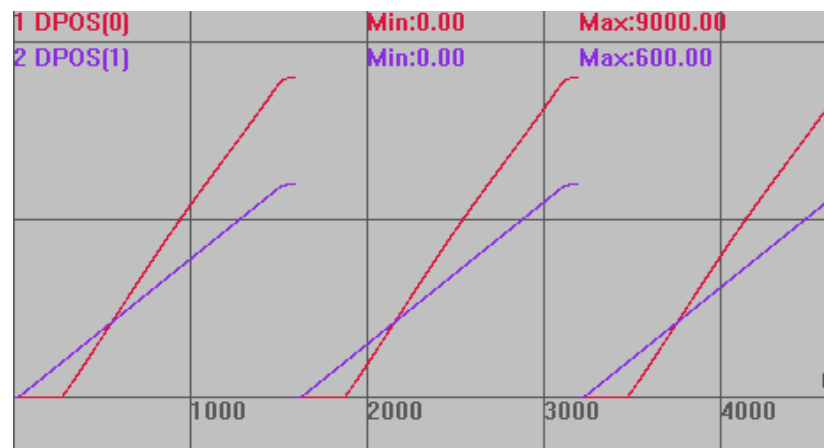
    IF IN (0) = ON THEN    'triggered when IN(0) is ON.
        DPOS= 0,0
        CAMBOX (0,360/stepdeg, 100, 500, 1,2,100) AXIS (0)
        'slave axis 0 is opened when reference axis 1 moves to position 100
        MOVE(600) AXIS(1)
        WAIT UNTIL IDLE AND IDLE    'wait until motion finishes.
        DELAY(100)
    ENDIF
WEND
END                    'stop present task.

```

Motion Path:

DPOS(0) vertical scale 5000

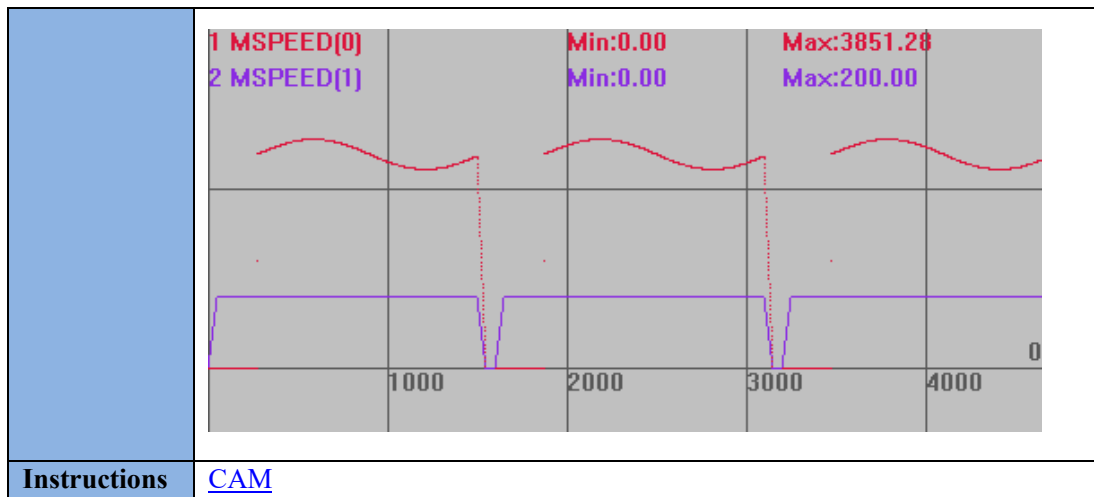
DPOS(1) vertical scale 500



Speed Curve:

MSPEED(0) vertical scale 3000

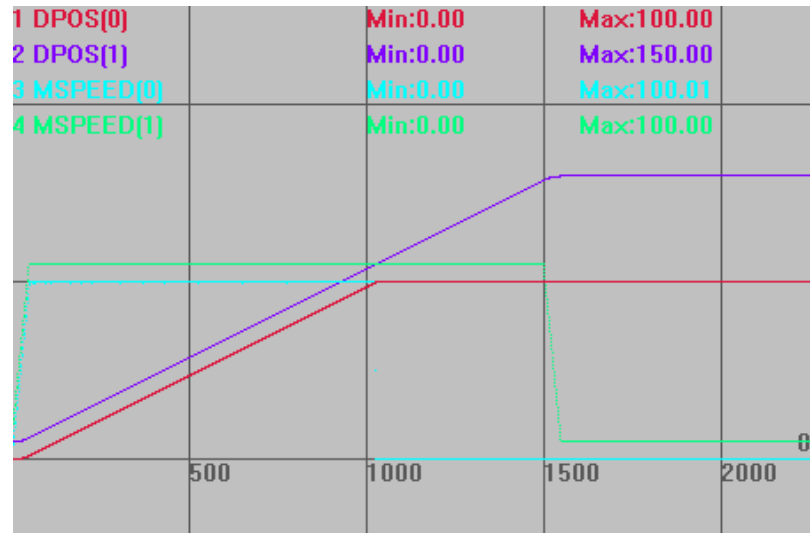
MSPEED(1) vertical scale 500



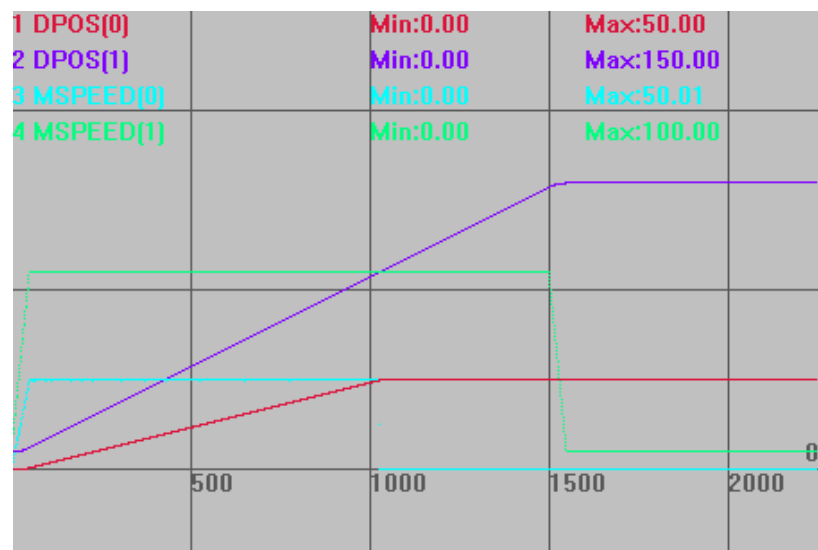
MOVELINK -- Auto Cam

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.									
Syntax	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 slave axis, units as unit. - link dec: absolute distance of reference axis during deceleration stage of slave axis, units as unit. <i>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.</i> - link axis: axis No. of reference axis. - link options: link mode, different binary bits stand for different meaning. <table><tr><th>Mode</th><th>Bit</th><th>Description</th></tr><tr><td>1</td><td>Bit 0</td><td>link starts when MARK signal of reference axis is triggered.</td></tr><tr><td>2</td><td>Bit 1</td><td>link starts when reference axis reaches a determined absolute position. (see link pos parameter description)</td></tr></table>	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)
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)								

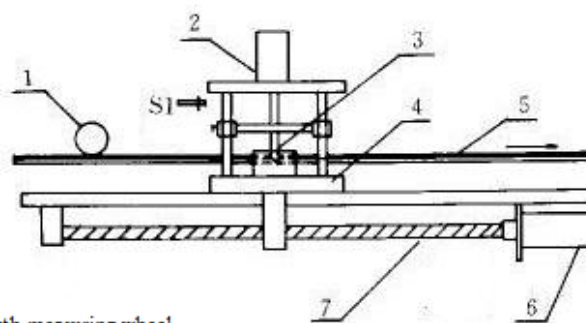
	4	Bit 2	MOVELINK will execute repeatedly, and it can be inverted. (this mode can be cancelled by setting the bit1 of REP_OPTION as 1.)
	8	Bit 3	curve acceleration or deceleration S mode, firmware version above 20170502 supports.
	16	Bit 4	start from a position in the middle, then use power failure interruption 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:



- 1: length measuring wheel
- 2: hydraulic cylinder
- 3: cutter
- 4: work platform
- 5: profile
- 6: servo motor
- 7: lead screw

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

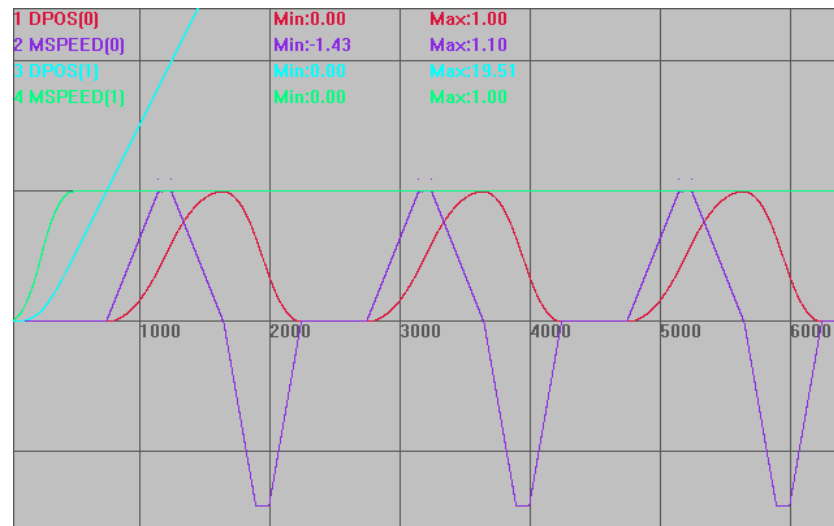
VMOVE(1)AXIS(1)      'the material keep feeding.
TRIGGER              'trigger the oscilloscope automatically

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.

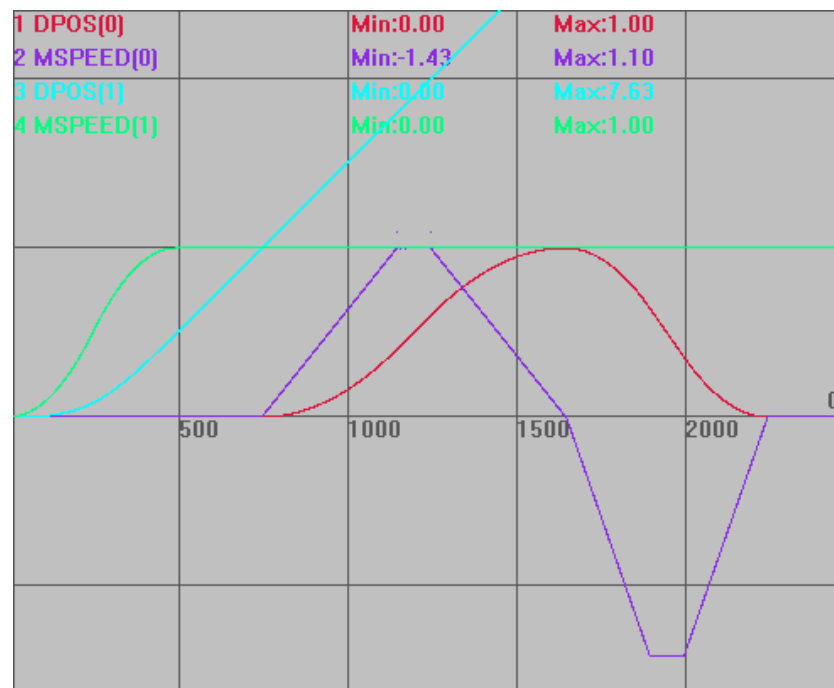
WEND

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): $1 + 0.8 + 0.2 + 0.8 + 1.2 = 4\text{m}$, 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

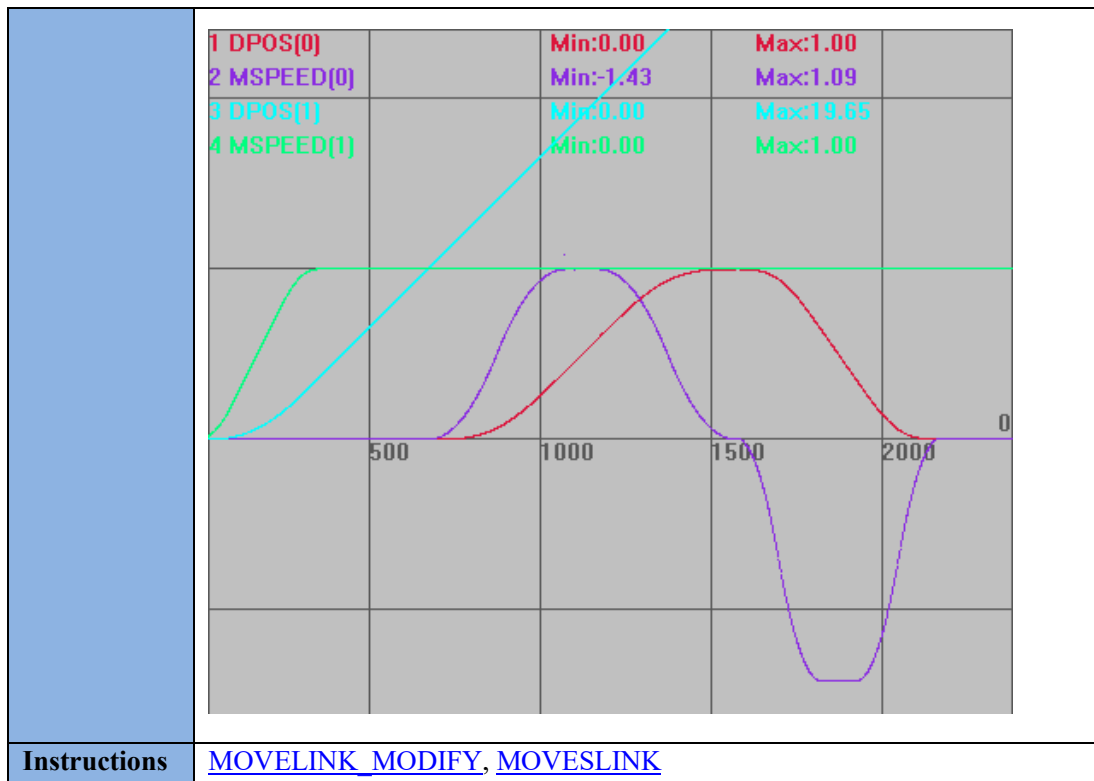
VMOVE(1)AXIS(1)      'the material keep feeding.
TRIGGER              'trigger the oscilloscope automatically

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

```



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.
Syntax	MOVELINK (distance, link dist, start sp, end sp, link axis[, link options] [, link pos][, link offpos]) behind 3 are optional parameters, when they are not set, comma must be added. ● 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 & slave axis when starting, units/units as unit. Negative value means the slave axis moves inversely. ● 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

VMOVE(1) AXIS(1)

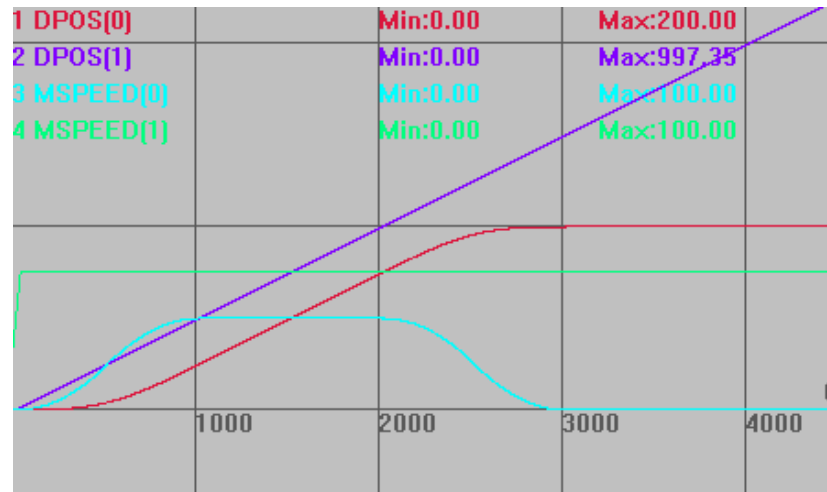
Interpolation path and speed curve:

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	MOVELINK

MOVELINK_MODIFY -- Sync 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.
Syntax	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)

```

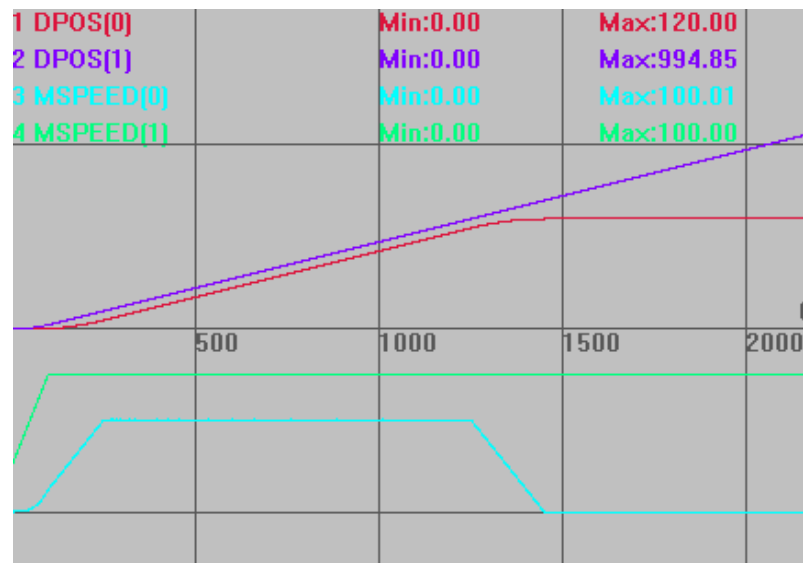
Motion Path and Speed Curve:

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

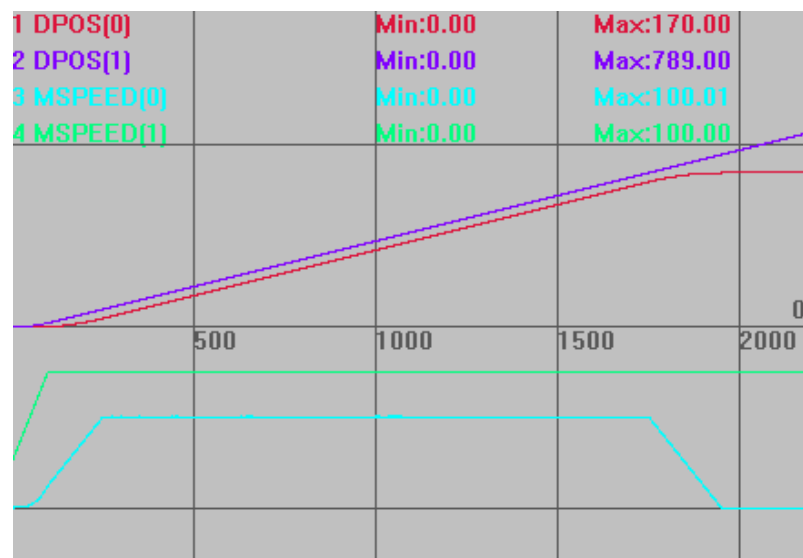


Add link distance, others same as former.

```

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

```



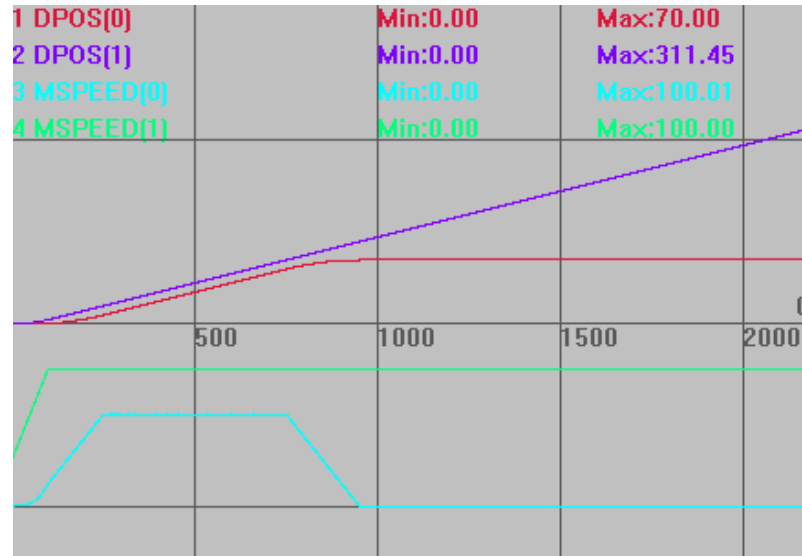
Decrease link distance, others same as former.

MOVELINK(10,20,20,0,1) 'acceleration stage of station

MOVELINK(100,100,0,0,1) 'link distance of slave axis is 100

MOVELINK_MODIFY=-50 'modify the link distance as 100-50

MOVELINK(910,20,0,20,1) '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

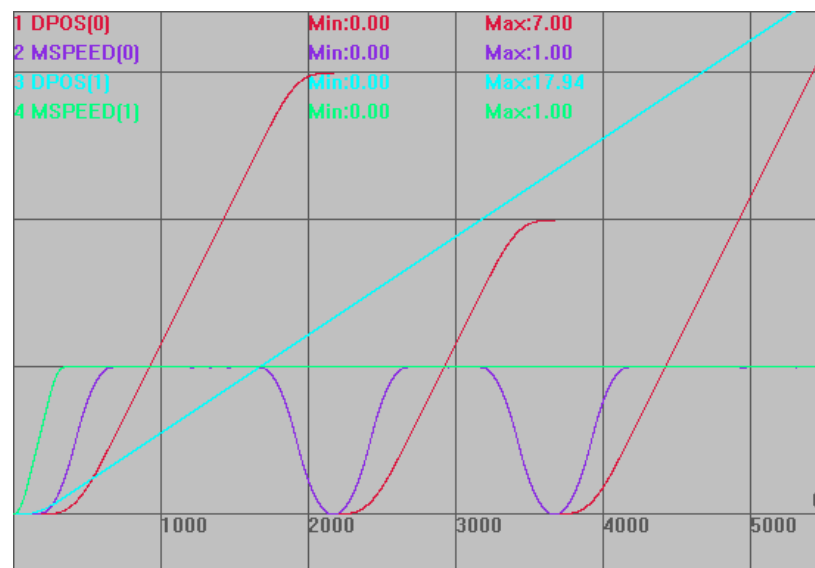
  WAIT IDLE(0)

  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

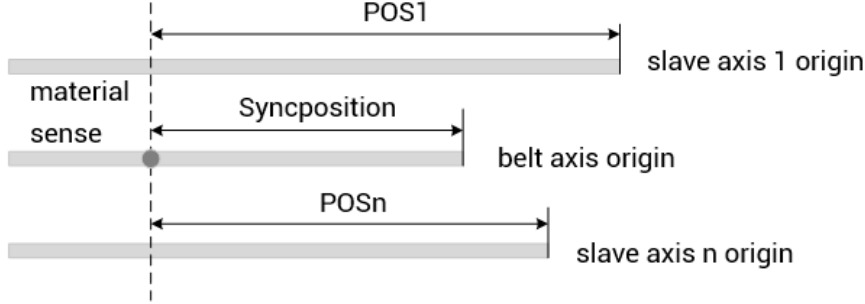
Motion Path and Speed Curve:
DPOS(0) vertical scale 1, no offset
DPOS(1) vertical scale 3, no offset
MSPEED(0) vertical scale 1, no offset
MSPEED(1) vertical scale 1, no offset

```

**Instructions**[MOVELINK](#)

MOVESYNC -- Synchronous Motion

Type	Motion Setting Instruction												
Description	Motion synchronization, follow with the belt axis (the object on the belt). This isn't interpolation motion, so it can't ensure linear path. ➤ The belt axis' length unit should be same as the following axis in BASE. ➤ When BASE axis following finishes, this instruction ends. ➤ MOVESYNC can be used continuously (ensure the speed is continuous), and MOVE_OP can be added in the middle (avoid the motion stops directly in high-speed following process when the motion ends). For the final MOVESYNC, please use mode -1. ➤ This belongs to CAM instruction; therefore, it doesn't support motion pause. And while using, please convert the angle to radian (00~2PI). ➤ Don't insert interpolation or joint motions when there are multiple movesync motions, otherwise it will cause speed interrupting or overshooting. However, you can use motion superposition "ADDAX" to change the motion trajectory.												
Syntax	MOVESYNC (mode, synctime, syncposition, syncaxis, pos1 [,pos2, pos3...]) <table border="1"> <thead> <tr> <th>Mode</th><th>Description</th></tr> </thead> <tbody> <tr> <td>-1</td><td> Synchronization mode ends. The motion has reached needed absolute position. Under this mode, if there are other MOVESYNC instructions next, it will not execute the motion, and "syncaxis" is invalid. </td></tr> <tr> <td>-2</td><td> Force it to end. When mode -2 is called, it will force to stop all MOVESYNC motions (that are in same buffer before this command) at assigned position. Also, if there are other MOVESYNC instructions next, it will not execute the motion, and "syncaxis" is invalid. </td></tr> <tr> <td>0</td><td>the first axis(x) of BASE follows with belt axis objects.</td></tr> <tr> <td>10</td><td>the second axis(y) of BASE follows with belt axis objects.</td></tr> <tr> <td>20</td><td>the third axis of BASE follows with belt axis objects.</td></tr> </tbody> </table> ● Mode: --mode = 0 + angle-- angle: belt rotation angle, angle means the forward angle between the belt axis DPOS and BASE axis. While using, please convert the angle to radian (0~2PI). Examples (in robotic arm coordinate system): a) BASE (X), mode=PI/4, which means the belt axis (with DPOS in forward direction) moves towards outside the coordinate system, then one 45 degrees angle is formed with axis X forward. b) BASE (X), mode=PI*5/4, which means the belt axis (with DPOS	Mode	Description	-1	Synchronization mode ends. The motion has reached needed absolute position. Under this mode, if there are other MOVESYNC instructions next, it will not execute the motion, and "syncaxis" is invalid.	-2	Force it to end. When mode -2 is called, it will force to stop all MOVESYNC motions (that are in same buffer before this command) at assigned position. Also, if there are other MOVESYNC instructions next, it will not execute the motion, and "syncaxis" is invalid.	0	the first axis(x) of BASE follows with belt axis objects.	10	the second axis(y) of BASE follows with belt axis objects.	20	the third axis of BASE follows with belt axis objects.
Mode	Description												
-1	Synchronization mode ends. The motion has reached needed absolute position. Under this mode, if there are other MOVESYNC instructions next, it will not execute the motion, and "syncaxis" is invalid.												
-2	Force it to end. When mode -2 is called, it will force to stop all MOVESYNC motions (that are in same buffer before this command) at assigned position. Also, if there are other MOVESYNC instructions next, it will not execute the motion, and "syncaxis" is invalid.												
0	the first axis(x) of BASE follows with belt axis objects.												
10	the second axis(y) of BASE follows with belt axis objects.												
20	the third axis of BASE follows with belt axis objects.												

	in forward direction) moves towards outside the coordinate system, then one 225 degrees angle is formed with axis X forward. c) Same, mode=PI/2, the belt is in the coordinate system Y forward. d) Same, mode=PI, the belt is in the coordinate system Y inverse. e) Same, mode=PI*1.75, the belt is in the direction of 315 degrees. <i>If you choose axis Y and belt axis' angle, please consider the forward angle between belt DPOS and axis Y.</i> mode=1000+tableindex The tracking ration of several axes are saved into TABLE in sequence, then 3D belt following motion can be achieved when the belt is tilted (but it is only valid in firmware version after 20180209). ● synctime: synchronization time (in ms). The motion will finish in assigned time, and when it finished, BASE axis follows with the belt axis and their speed are the same. If it is 0, which means the synchronization time can be estimated according to motion axis' speed, acceleration, but it maybe not accurate. ● syncposition: belt position when belt objects are sensed. It can make the belt axis coordinate in cycle, but please ensure there is no coordinate change or cycle between the parameter position and belt axis position when it is called. Therefore, don't use it near the coordinate 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 sensed. ● posn: the n axis' (BASE) absolute position when belt object is sensed. 
Controller	ECI241X, ECI282X, ECI382X, XPLC full series, ZMC3XX series and above.
Example	Example 1 <pre> GLOBAL tmp_val GLOBAL tmp_offPos GLOBAL tmp_AccPos tmp_flag=0 SPEED(0)=800 DPOS(0)=0 DPOS(1)=0 VMOVE(1)AXIS(0) wait UNTIL tmp_flag=1 </pre>

```

tmp_offPos=100
tmp_AccPos=800*0.4/2
tmp_val=Mpos(0)+tmp_offPos+tmp_AccPos
?tmp_val
TRIGGER
base(1)
MOVE_WAIT(mpos,0,1,tmp_val-tmp_AccPos)axis(1)
MOVESYNC(0,400,tmp_val,0,0)AXIS(1)    'acceleration
MOVESYNC(0,1000,tmp_val,0,0)AXIS(1)    'constant speed
MOVESYNC(-1,400,tmp_val,-1,800)AXIS(1) 'deceleration

```

End

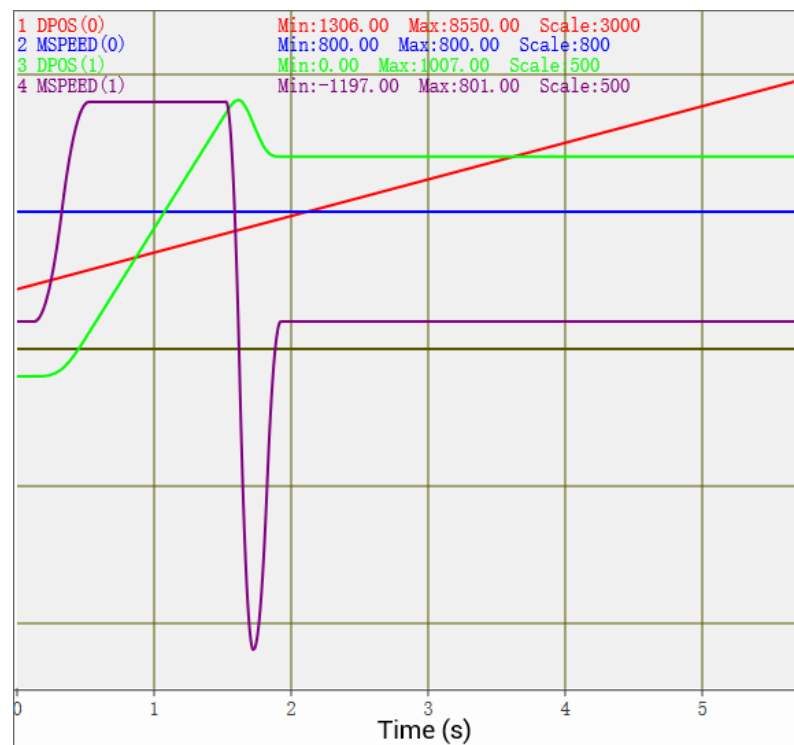
Effects (curves of motion path and speed)

DPOS(0), vertical scale 3000, no offset

DPOS(1), vertical scale 500, offset -100

MSPEED(0), vertical scale 500, no offset

MSPEED(1), vertical scale 500, offset 100



Example 2: belt gets the materials

```
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) 'synchronize with belt axis motion
MOVE_OP(1,1) 'down, if makes axis Z down, you also can
                        use MOVESYNC.

MOVE_OP(0,1) 'open the nozzle
MOVESYNC(0,1000,100,1,120) 'keep following 1s
MOVE_OP(1,0) 'up
MOVESYNC(-1, 0, 0, -1, 400) 'move to material placing position 400
MOVE_OP(1,1) 'down
MOVE_OP(0,0) 'close the nozzle
MOVE_DELAY(2) 'delay 2ms, don't add this kind of command
                        in MOVESYNC continuous motion

MOVE_OP(1,0) 'up
MOVEABS(0) 'back to origin
VMOVE(1) AXIS(1) 'belt axis motion

```

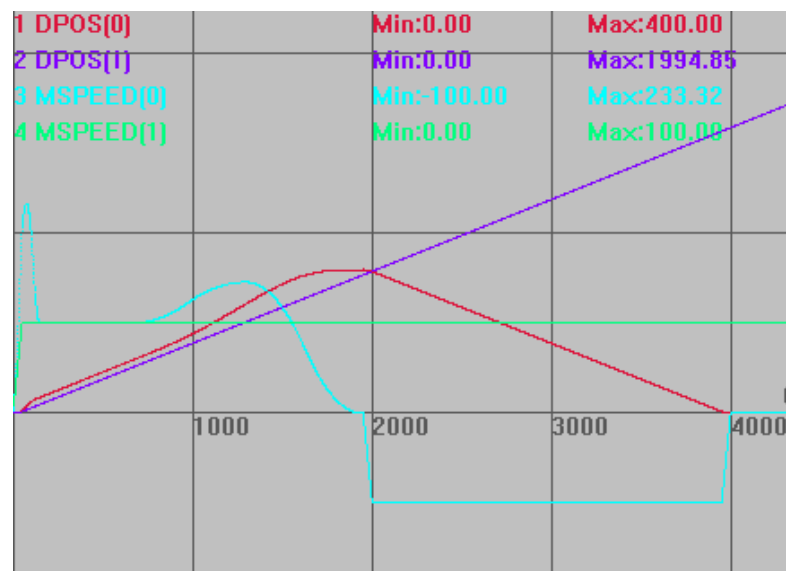
Motion path and speed curve:

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 3: Get 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 as different speeds
ACCEL=1000,1000,1000
DECEL=1000,1000,1000
TRIGGER
MOVESYNC(0, 0, 50, 1, 80 )   'synchronize with belt axis
MOVE_OP(0,1)                  'open the nozzle
MOVE_OP(1,1)
'down, if makes axis Z down, you also can use MOVESYNC.
MOVESYNC(0, 300, 50, 1, 80 ) 'keep following 2ms
MOVE_OP(1,0)                  'up
MOVESYNC(0, 0, 100, 2, 150) 'move to 2nd belt related position
MOVE_OP(1,1)                  'down
MOVE_OP(0,0)                  'close the nozzle
MOVESYNC(0,300, 100, 2, 150) 'synchronize 2ms
MOVE_OP(1,0)                  'up
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 4: SCARA MOVESYNC, please check [“SCARA Synchronization Motion”](#)

Example 4: 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 real axis
ADDAX(7) AXIS(1)
BASE(0, 1)
MOVESYNC(0, 0, 50, 2, 80,100) 'synchronize with the belt axis
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) 'in the process of superposition carving, absolute motion's
              instruction position will be wrong, make one delay to
  
```

avoid command calling.

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

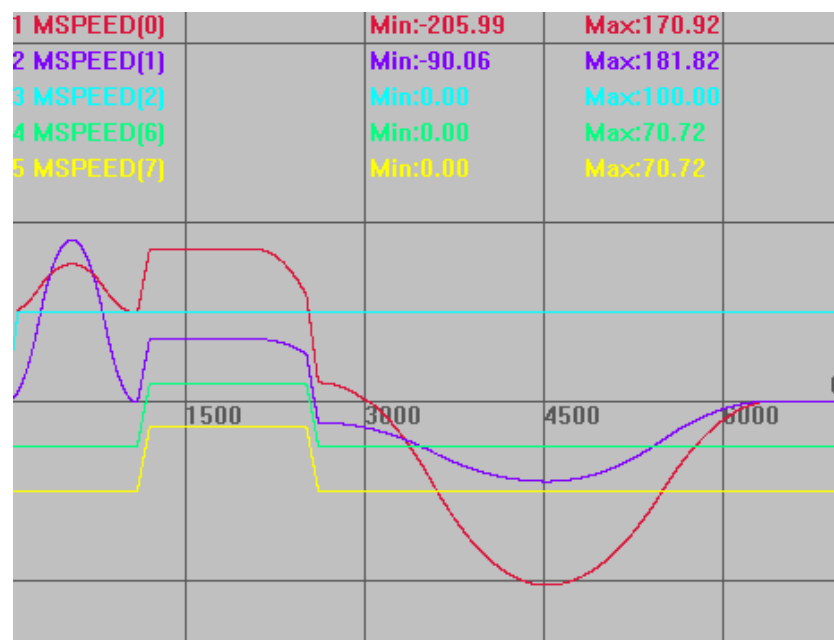
MSPEED(0) vertical scale 200, no offset

MSPEED(1) vertical scale 200, no offset

MSPEED(2) vertical scale 200, no offset

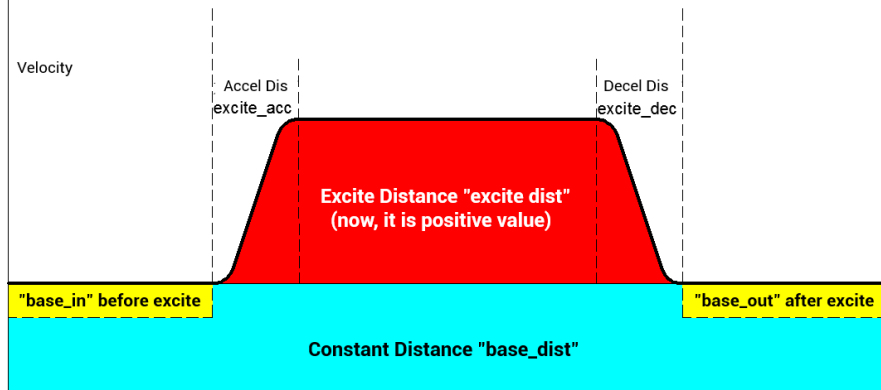
MSPEED(6) vertical scale 200, offset -50

MSPEED(7) vertical scale 200, offset -100



FLEXLINK -- Excite 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.

Syntax

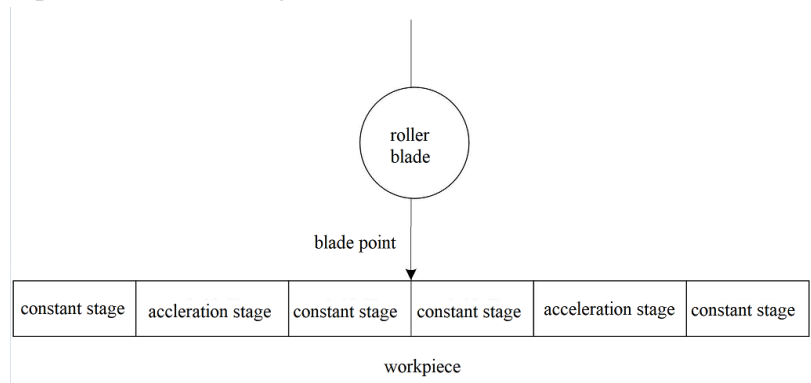
FLEXLINK(base_dist, excite_dist, link_dist, base_in, base_out, excite_acc, excite_dec, link_axis, link_options, [start_pos], [link_offpos])

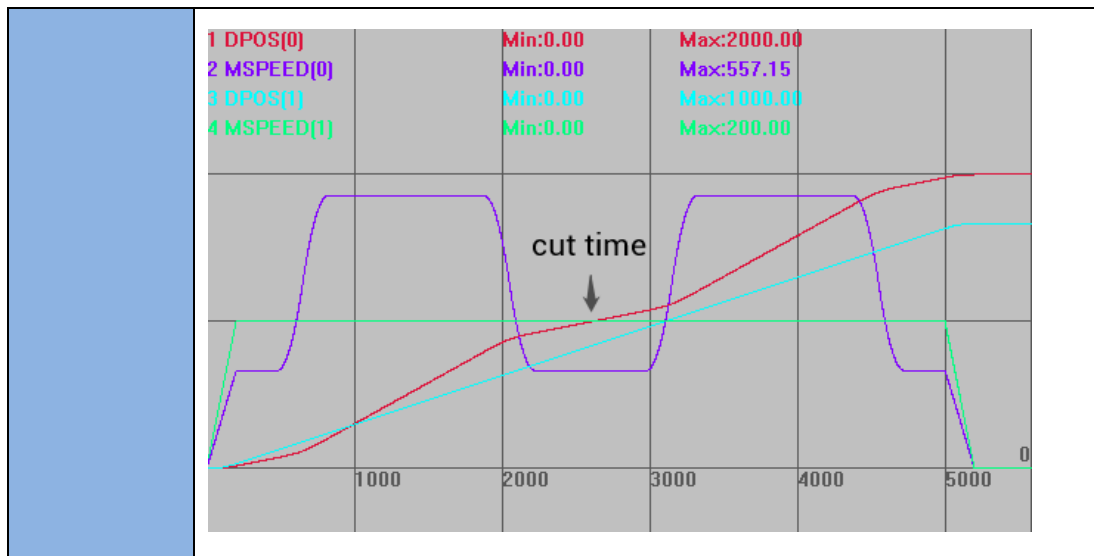
- base_dist: uniform motion distance of slave axis.
- excite_dist: excitation motion distance of slave axis, +: increase, -: decrease. **Total distance of slave axis = base_dist + excite_dist.**
- link_dist: distance of main axis during the whole link.
- base_in: percentage of base_dist that distance 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 axis 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  <pre> 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 </pre>



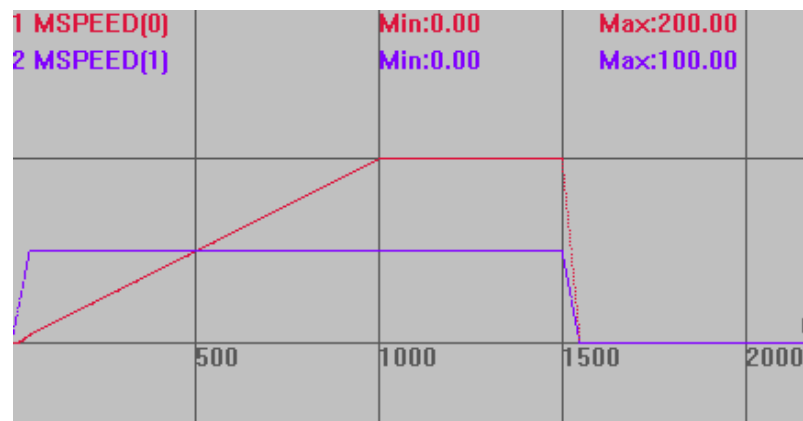
7.5. Motion Settings Instructions

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 ratio 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.)
Syntax	CLUTCH_RATE= value
Controller	General
Example	Example 1: <pre> 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 </pre>

MSPEED(0) vertical scale 200

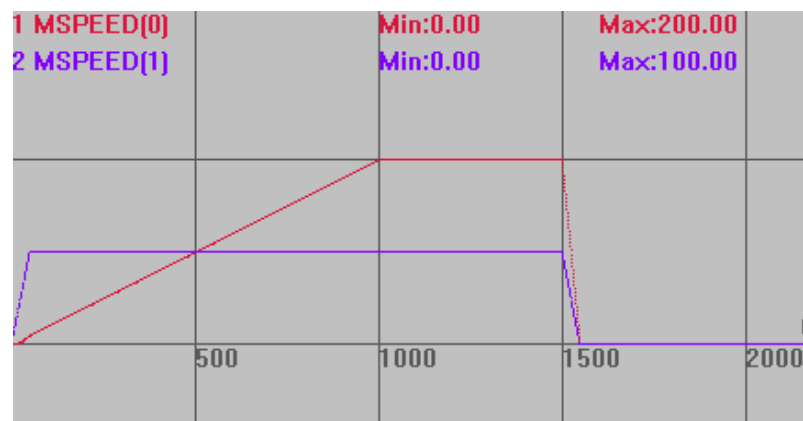
MSPEED(1) vertical scale 200



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



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)

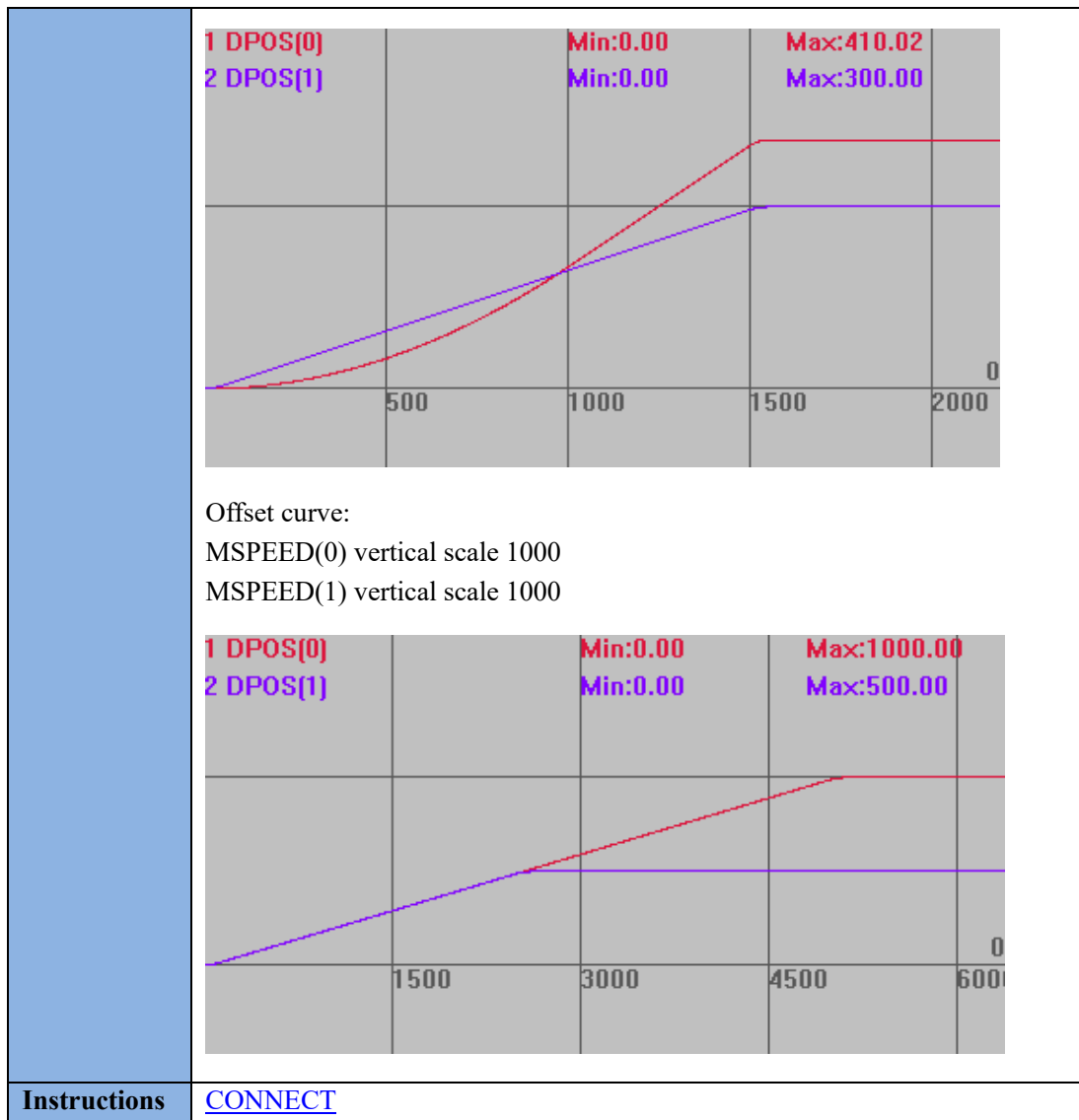
'link speed is 0.2s, link time is determined by slave axis speed / acceleration

VMOVE(500)AXIS(1)

Speed curve

MSPEED(0) vertical scale 100

MSPEED(1) vertical scale 100



ENCODER_RATIO -- Encoder Gear Ratio

Type	Motion Setting Instruction								
Description	Set the encoder IN gear ratio, the default value is (1,1). And set as one negative value to switch the direction. Note: if a negative gear ratio causes misalignment, swap the negative signs of the numerator and denominator.								
Syntax	ENCODER_RATIO(mpos_count, input_count[, mode]) - mpos_count: numerator, maximum is 65535 - input_count: denominator. maximum is 65535 <table border="1"> <thead> <tr> <th>Mode</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>AB 1X Mode</td></tr> <tr> <td>2</td><td>AB 2X Mode</td></tr> <tr> <td>3</td><td>AB 4X Mode</td></tr> </tbody> </table>	Mode	Description	1	AB 1X Mode	2	AB 2X Mode	3	AB 4X Mode
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.
Controller	Valid in firmware ZMC406 20170502 above.
Example	ENCODER_RATIO(4,1) 'input 4 times of encoder, which equals to ENCODER_RATIO (1,1,4) ENCODER_RATIO(1,-1) 'encoder IN -- switch the direction, which equals to ENCODER_RATIO (-1,1)
Instructions	PP_STEP , ENCODER

STEP_RATIO -- Motor Gear Ratio

Type	Motion Setting Instruction
Description	Set output gear ratio of stepper, default value is (1,1). Range: -2^{31} to $2^{31}-1$. The motor direction can be changed by setting minus value, but it is not recommended. For pulse motor, you can use INVERT_STEP. For bus moto, please modify in the drive. Don't modify the value frequently, it is better to change the pulse amount to realize the same effect. And if the position is not correct for negative gear ratio, try to swap the signs of the numerator and denominator.
Syntax	STEP_RATIO(output_count, input_count) ● output_count: numerator ● input_count: denominator Note: for numerator and denominator, only one can be set as minus.
Controller	General
Example	STEP_RATIO (16,1) 'output 16 times of the set pulse value. And multiplying the pulse equivalent by 16 can also achieve the same effect.

BACKLASH -- Backlash Compensation

Type	Motion Setting Instruction
Description	To set reverse compensation of axis, not valid in extended axis.
Syntax	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)

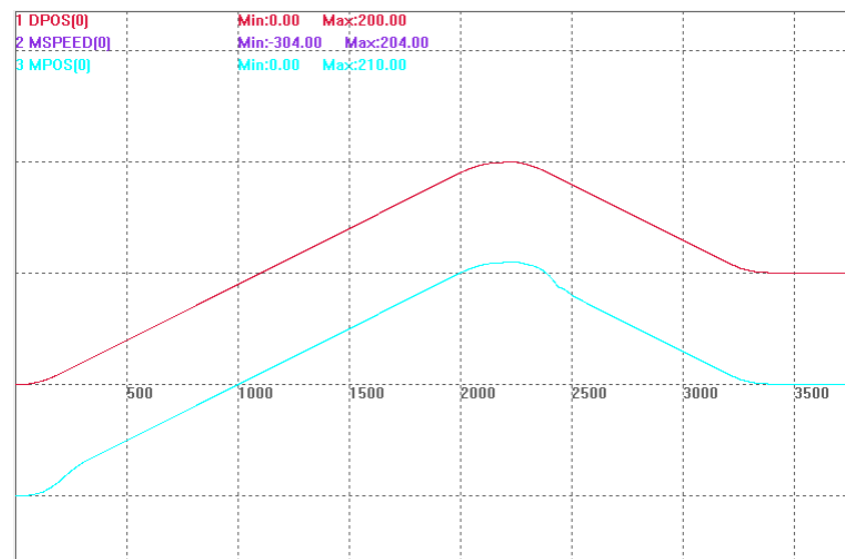
'10mm compensation

MOVE(200)

MOVE(-100)

'start to compensate when reverse

END



PITCHSET -- 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.
Syntax	PITCHSET(enable [,startpos, disone, maxpoint, tablenum] [,tablereverse]) ● 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: ABL position where saved the compensation point, it saves starting from next point of “startpos”, the unit is pulse ● tablereverse: it is used when the compensation values are different under forward and reverse – the TABLE position where saves pitch compensation Notes: a. It can modify compensation parameters dynamically. b. When the compensation is ON or OFF, adjust DPOS and MPOS, but don't make them correct by moving. c. The distance can't be negative, otherwise, the compensation becomes invalid. No matter the motor moving direction is, the direction that is set by pitset is always forward. d. No buffer for 1D compensation command, that is, it only can download one compensation command once time. e. Under reversely, if it is inconsistent with forward, using tablereverse parameter to set it.
Controller	General
Example	Singe-Axis Pitch Compensation: Example 1: <pre> 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 </pre>

PITCHSET(1,0,100,6,0) 'when MPOS=0, it starts to compensate 6 points, a space of 100

TRIGGER

MOVE(70)

MOVE(-700)

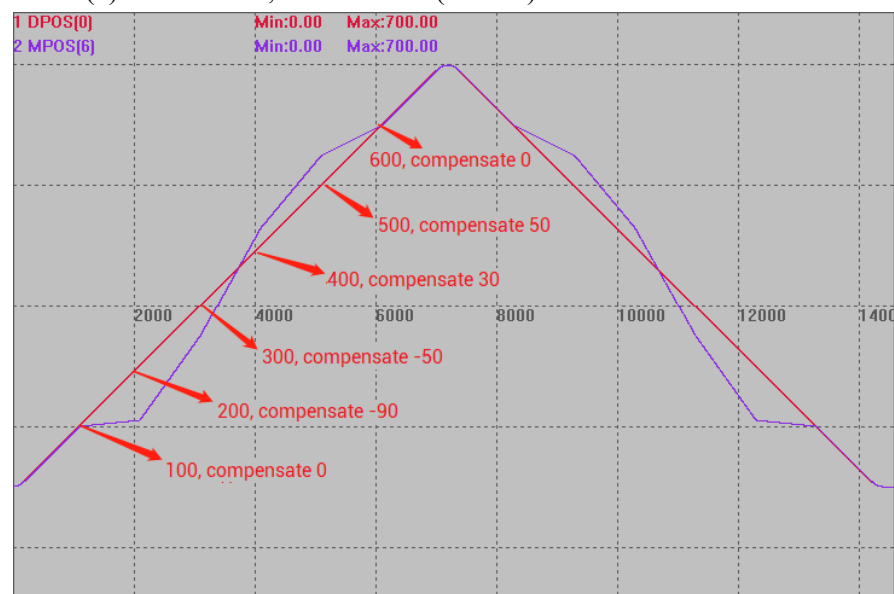
WAIT IDLE

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)
```

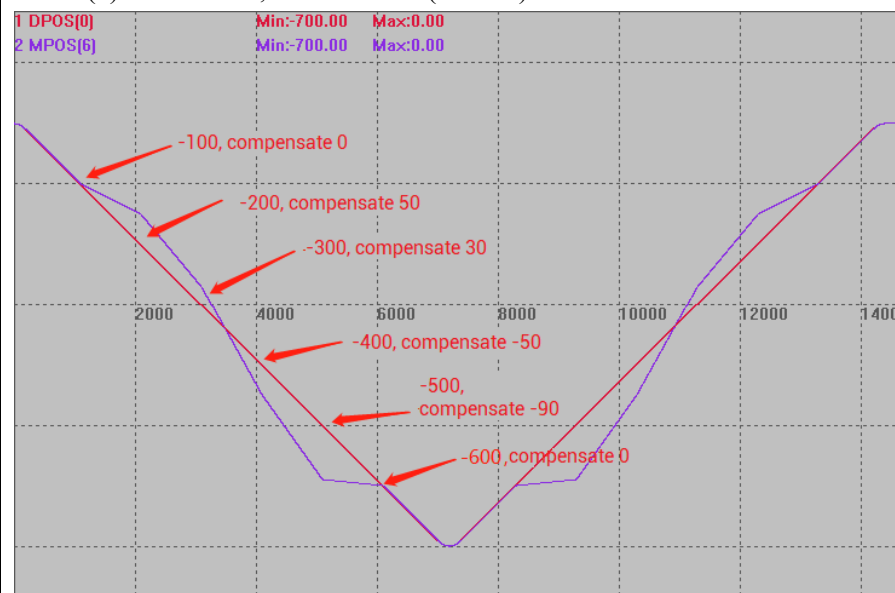
```
WAIT IDLE
```

```
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

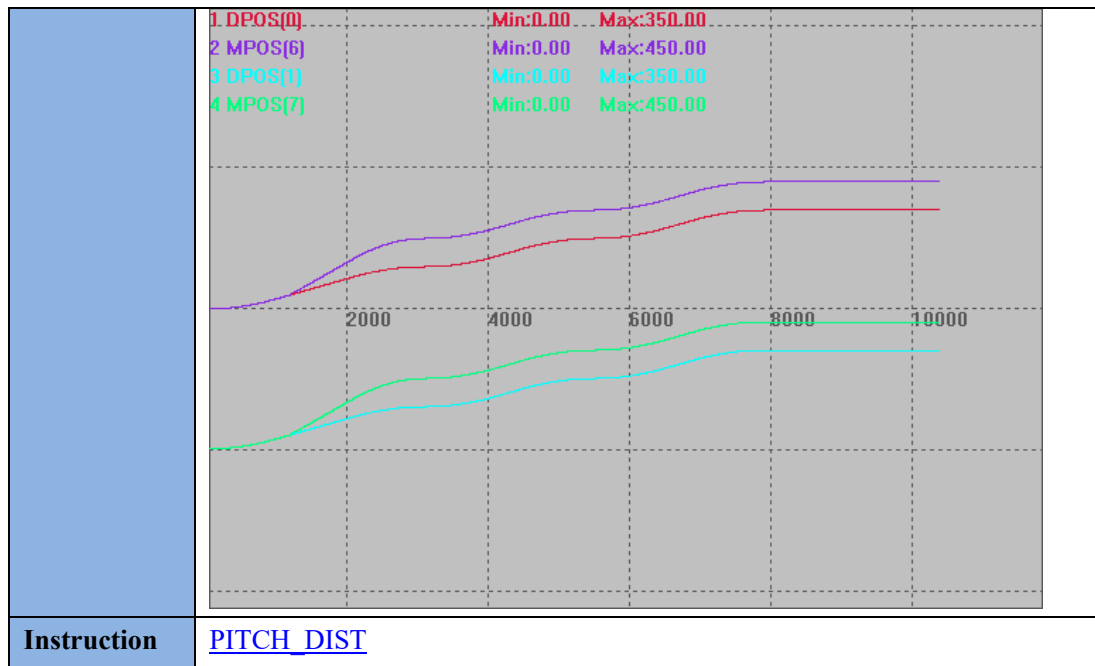
TABLE(0, 100*UNITS(0), 100*UNITS(0), 100*UNITS(0))
'TABLE saves pitch compensation value, the value is the number of pulses,
not the pitch compensation value.

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

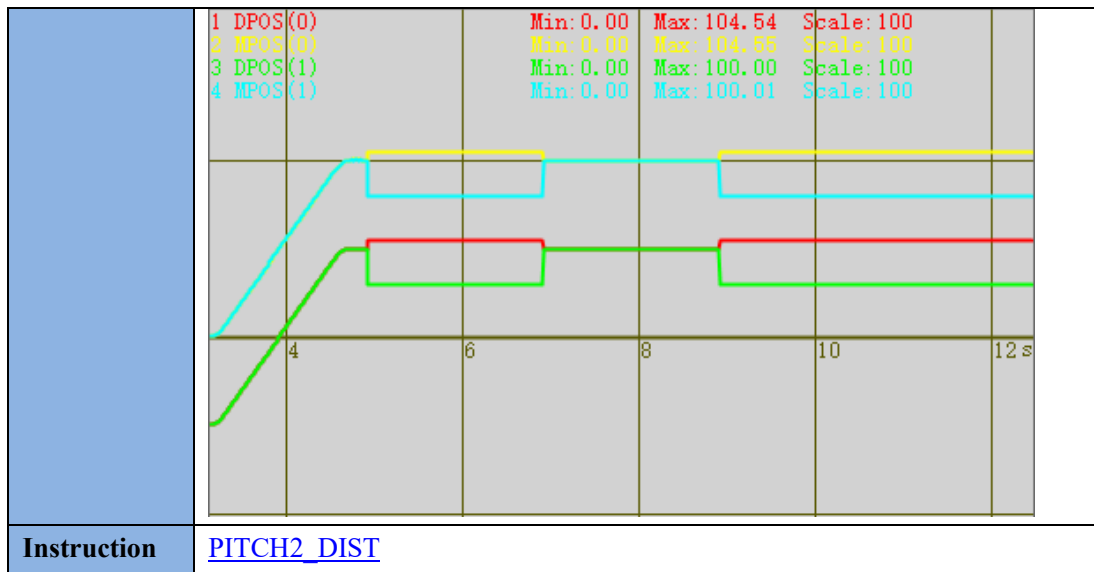
```



PITCH2SET -- 2D 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.
Syntax	PITCH2SET(enable, startposx, startposy, disonex, disoney, maxpointx, maxpointy, tableindex) - enable 0: close compensation, 1: XY compensation, 2: XYX compensation, 3: XYY compensation “BASE axis sequence of mode 2 must be X1, Y, X2, at this time, X2 compensation value refers to X1 compensation” “BASE axis sequence of mode 3 must be X, Y1, Y2, at this time, Y2 compensation value refers to Y1 compensation” - 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 starts from the next point of “startpos”, the unit is pulse. Each point saves 2 data, which are deviation values in X and Y (first row = X deviation, the second row = Y, maxpointx*maxpointy*2 TABLE position are occupied) It can modify compensation parameters dynamically. When the compensation is ON or OFF, please adjust DPOS and MPOS to make them correct. Notes: - Axis X must be smaller than axis Y.

	b. Please set ATYPE at first, then set compensation. While modifying ATYPE, remember to close compensation firstly. c. While setting compensation, axis should be IDLE stopped state. d. The compensation is calculated according to MPOS_ORI, when the axis is with feedback, MPOS feedback value must be correct.
Controller	Above ZMC4XX, and it must be with special firmware (VERSION_BUILD), which occupies CPU to the extent, so the standard firmware is not OK.
Example	<pre> BASE(0,1) ATYPE=5,5 UNITS=100,100 SPEED=100,100 ACCEL=500,500 DECEL=500,500 DPOS=0,0 MPOS=0,0 TABLE(0,10*UNITS(0),-20*UNITS(0),-20*UNITS(0),- 40*UNITS(0),0*UNITS(0),-0*UNITS(0)) delay(10) trigger PITCH2SET(1,0,0,90,90,2,2,0) MOVE(100,100) WAIT IDLE delay(100) delay(100) PITCH2SET(0) 'close compensation delay(2000) PITCH2SET(1,0,0,90,90,2,2,0) 'not run, open compensation, check MPOS delay(2000) PITCH2SET(0) To show better compensation effect, connect axis 0 / 1 pulse output to axis 6 / 7 encoder input, data are captured by SCOPE as below: DPOS (0) with -50 offset, vertical scale is 100 MPOS (0) without offset, vertical scale is 100 DPOS (1) with -50 offset, vertical scale is 100 MPOS (1) without offset, vertical scale is 100 </pre>



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.
Syntax	VAL=PITCH_DIST (axisnum) ● axisnum: axis No.
Controller	General
Instruction	PITCHSET

PITCH2_DIST -- 2D Pitch Compensation Distance

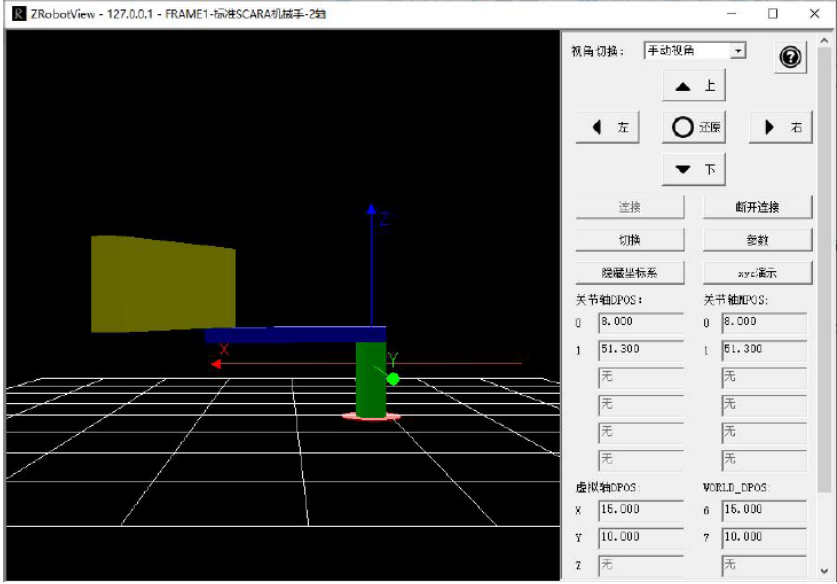
Type	Axis State
Description	Read distance value of current axis pitch compensation, the real MPOS returned value will minus the value.
Syntax	BASE (x, y) 'the former 2 axes are X and Y axis No. for 2D screw VAL=PITCH_DIST (axisnum) ● axisnum: axis No.
Controller	General
Instruction	PITCH2SET

7.6. Robot Instructions

CONNFRAME -- Robot Inverse Kinematics

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 CONNFRAME 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.
Syntax	CONNFRAME(frame, tablinum, 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 <i>Zmotion robotic arm instructions</i> 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	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.	

	<pre> WAIT LOADED BASE(6,7) MOVEABS(15,10) 'virtual axis sends motion instructions END </pre> Connect the simulation tool of robotic arm to view the running effect as shown below: 
Instructions	CONNREFRAME

CONNREFRAME -- Robot Forward Kinematics

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.
Syntax	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	<pre> 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. </pre>
Instructions	CONNFRAME

FRAME -- Robotic Arm Type

Type	Robotic Arm Instruction
Description	Choose robotic Type , see <i>Robotic Arm Manual</i> for reference.
Instructions	CONNFRAME

FRAME_STATUS -- Robot Axis Status

Type	Robotic Arm Instruction
Description	Indicate current robotic arm posture. When the status is not robotic arm, it returns -1. FRAME_TRANS2 instruction will use this posture. It is only for SCARA, Quasi-SCARA, and 6 DOF manipulators, and there are several postures. And for SCARA, left-hand status value is 0, right-hand status value is 1.
Syntax	VAR1 = FRAME_STATUS (AXIS)
Controller	General
Example	<pre> Input online instruction ?FRAME_STATUS, and print the current status. >>?FRAME_STATUS </pre>
Instructions	BASE

FRAME_TRANS2 -- Coordinates Transform (FWD / REV)

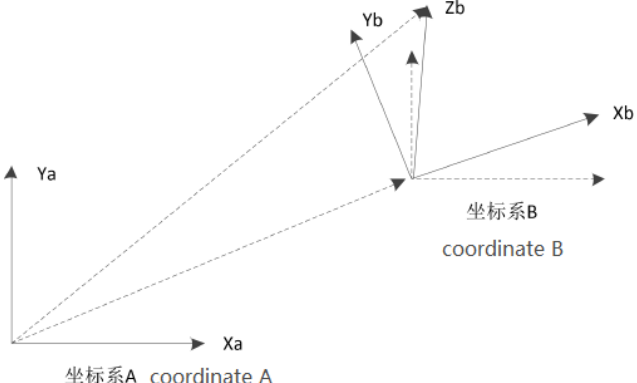
Type	Robotic Calculation Instruction																	
Description	Coordinate transformation function. It must be used when the forward or inverse is built. The axis No. of BASE must be correct when using the instruction. For inverse solution, base as virtual axis, for forward solution, base as joint axis. According to correct sequence, fill corresponding data. And data and number filled in should be the same as ?*frame.																	
Syntax	FRAME_TRANS2(tablein, tableout, dir) ● tablein: index No. of table array, that is, continuously start to save data from this index. Under forward mode, input joint coordinates, under inverse mode, input virtual axis coordinates, at last, plus the postures. ● tableout: under forward mode, output virtual coordinate and then plus posture, under inverse mode, 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>under inverse mode, transform the virtual axis coordinates to joint axis coordinates, without the posture, that is, it uses current posture automatically.</td></tr><tr><td>1</td><td>Forward</td><td>under forward mode, transform the joint axis coordinates to virtual axis coordinates, no posture output.</td></tr><tr><td>2</td><td>Inverse</td><td>under inverse mode, input the virtual axis coordinates, and add the posture at last.</td></tr><tr><td>3</td><td>Forward</td><td>under forward mode, output the virtual axis coordinates, and when output the last position, filling in the posture.</td></tr></table>			Mode	Type	Description	0	Inverse	under inverse mode, transform the virtual axis coordinates to joint axis coordinates, without the posture, that is, it uses current posture automatically.	1	Forward	under forward mode, transform the joint axis coordinates to virtual axis coordinates, no posture output.	2	Inverse	under inverse mode, input the virtual axis coordinates, and add the posture at last.	3	Forward	under forward mode, output the virtual axis coordinates, and when output the last position, filling in the posture.
Mode	Type	Description																
0	Inverse	under inverse mode, transform the virtual axis coordinates to joint axis coordinates, without the posture, that is, it uses current posture automatically.																
1	Forward	under forward mode, transform the joint axis coordinates to virtual axis coordinates, no posture output.																
2	Inverse	under inverse mode, input the virtual axis coordinates, and add the posture at last.																
3	Forward	under forward mode, output the virtual axis coordinates, and when output the last position, filling in the posture.																
Controller	General																	
Example	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:  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	Input X,Y, status	Command	Output Joint
Inverse	Virtual axes	table(100,20,0,0)	frame_trans2 (100,200,0)	table(200,0,0)
		table(100,10,10,1)		table(200,0,90)
		table(100,10,10,1)	frame_trans2 (100,200,2)	table(200,90,-90)
Status	BASE	Input joint coordinate	Command	Output X, Y, status
Forward	Joint axis	table(100,0,0)	frame_trans2 (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_trans2 (100,200,3)	table(200,10,10,1)

FRAME_ROTATE -- Workpiece Coordinates Transform

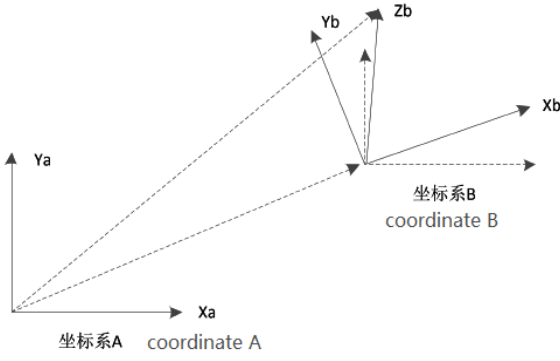
Type	Robot Calculation Instruction
Description	Used to translate and rotate workpiece coordinate system. At present, only FRAME6 can be rotated with posture together, for others that have XYZ virtual axes support XYZ 3 rotary axes, but the status can't be rotated. Virtual axis WORLD_DPOS means world coordinates won't change after rotation, for virtual axis DPOS, which means workpiece axis will change. And while using, the controller should be in mechanical connection. Notes: ➤ The axes of BASE can be either virtual axis or joint axis. And 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, then it will do coordinate calculation for mode 1 manipulator, that is, in BASE axis sequence.
Syntax	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 $X^{\wedge}$ ● RY: rotation angle of coordinate B towards $Y^{\wedge}$ ● RZ: rotation angle of coordinate B towards $Z^{\wedge}$  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	<u>FRAME_ROTATE2</u>

FRAME_ROTATE2 -- Coordinates Transform Calculation

Type	Robot Calculation Instruction						
Description	Manually calculate coordinate value after rotation. 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.						
Syntax	FRAME_ROTATE2(tablein, tableout, dir[, x,y,z[, rx,ry,rz]]) ret = FRAME_ROTATE2(tablein, tableout, dir[, x,y,z[, rx,ry,rz]]) ● tablein: before transform, TABLE where saves input coordinates. ● tableout: after transform, TABLE where saves output coordinates. ● dir: direction selection <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>From DPOS to WORLD_DPOS</td></tr> <tr> <td>1</td><td>From WORLD_DPOS to DPOS</td></tr> </tbody> </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^	Value	Description	0	From DPOS to WORLD_DPOS	1	From WORLD_DPOS to DPOS
Value	Description						
0	From DPOS to WORLD_DPOS						
1	From WORLD_DPOS to DPOS						

	● 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 
Controller	General
Example	Example1: FRAME=2, DETLA, rotates X axis at 90 degrees. <pre> 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) WAIT LOADED ret=FRAME_ROTATE2(100,200,1,0,0,0,PI/2,0,0) IF ret=-1 THEN ?"calculated values" ?"DPOS(6)=",TABLE(200) ?"DPOS(7)=",TABLE(201) ?"DPOS(8)=",TABLE(202) ?"compare values" ?"DPOS(6)compare",TABLE(200)-DPOS(6) ?"DPOS(7)compare",TABLE(201)-DPOS(7) ?"DPOS(8)compare",TABLE(202)-DPOS(8) </pre>

ENDIF

Output result:

Calculated Values

DPOS(6)=0

DPOS(7)=-58.1400

DPOS(8)=0.0000

Compare Values

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.
Syntax	var1=WORLD_DPOS(axis)
Controller	General
Instructions	Online instruction, print. >>?*WORLD_DPOS

MOVER_L/MOVER_LABS -- Joint 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.
Syntax	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 -- Joint Axis Plane Arc

Type	Motion Instruction								
Description	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.								
Syntax	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 border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <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> </tbody> </table>	Value	Description	0	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.	1	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.	2	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.
Value	Description								
0	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.								
1	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.								
2	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.								

	3	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.
	dis1-disn: the distance of spiral axis	
Controller	ZMC4XX series with firmware version 20170511 or above support.	
Routine	<pre> 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) </pre>	
Instructions	MOVER_L , MOVER_C3	

MOVER_C3/MOVER_C3ABS -- Joint Axis Space Arc

Type	Motion Instruction					
Description	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.					
Syntax	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 ● 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></table>		Value	Description	0	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.
Value	Description					
0	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.					

	1	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.
	2	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.
	3	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.
	● dis1- disn: distance of spiral axis	
Controller	Valid in ZMC4XX series with firmware version 20170511 or above.	
Routine	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)	
Instructions	MOVER_L , MOVER_C	

FRAME_CAL -- Parameters Correction

Type	Robot Calculation Instruction
Description	It corrects the present robotic arm parameter automatically according to coordinates and features of robotic arm teaching. When the robotic arm connection is built by the current origin position and manipulator parameters, steps are shown below: (1) use the joint-axis coordinates that are saved in Tablein to control the manipulator terminal point to do jog motion, arriving the correction point. (2) get the joint axis coordinate of correction point. (3) correct the deviation between the present origin position and theoretical

	origin position, then calculate joint axis coordinate of theoretical origin point. (4) correct robotic arm parameter values (correct some parameters), then calculate theoretical robotic parameter values. Notes: ➤ FRAME_CAL is only for calculation, if the return value is -1, which means succeeds in calculating, if it's 0, means it fails. For FRAME_CAL, the axes is BASE must be under FRAME state.
Syntax	FRAME_CAL(tablein,space,groups,tableaux, zeroout, [tableout2]) ● tablein: TABLE starting No. where saves joint coordinates, and each coordinate is saved in sequence, multiple points are separated by space. ● space: "table" element between every two points. ● groups: the number of points ● tableaux: table No. of auxiliary parameters, some of them are needed for FRAME. ● zeroout: TABLE No. where saves joint absolute coordinates when the theoretical origin is calculated. ● tableout2: the position where saves calculated robotic arm parameters. When it is not filled, directly save into original position.
Controller	General

MOVER2_J/MOVER2_JABS -- Robot Point Motion

Type	Robot Motion Instruction
Description	Move the manipulator terminal to each joint's target points through point to point (PTP) joint interpolation motion. Enter each joint axis' angle value, the unit is degree. This is valid in joint axes set in BASE. MOVER2_JABS – point to point motion with absolute position MOVER2_J -- point to point motion with incremental position <u>How to set speed:</u> please check "Speed & Accel of PTP Motion & Linear Motion Configurations".
Syntax	MOVER2_J/MOVER2_JABS(End_Joint1,End_Joint2,End_Joint3,End_Joint4,...) ● End_Joint1: the target degree value of the manipulator first joint axis. ● End_Joint2: the target degree value of the manipulator next joint axis. ●
Controller	Valid in ZMC4XX or above, and the firmware should be above 20170511.
Routine	BASE(6,7,8,9,10,11) 'select virtual axis No. CONNREFRAME(6,0,0,1,2,3,4,5) '6 DOF manipulator model, axis 0 ~ axis 5 are joint axes ?"forward mode"

	BASE(0,1,2,3,4,5) 'select 012345 joint axes FORCE_SPEED=0.9 'set the point to point (PTP) speed of below MOVER2_JABS as 0.9, that is, the 90% full speed. MOVER2_JABS(60,60,30,20,0,90) 'where the 6-axis manipulator runs (each joint-axis target position by PTP motion), moving to each joint-axis' angle value (60,60,30,20,0,90).
Instructions	MOVER2_P / MOVER2_PABS

MOVER2_P/MOVER2_PABS -- Robot Point Motion

Type	Robot Motion Instruction
Description	Move the manipulator terminal to each joint's target posture through point to point (PTP) joint interpolation motion. This is same as MOVER2_J, but it enters target point's posture. This is valid in virtual axes set in BASE.
Syntax	MOVER2_P/MOVER2_PABS(Fastatus,End_Pos1,End_Pos2,End_Pos3,End4,...) ● Fastatus: if it is 6 dof robot, it means the type value that determines the unique solution, the range is 0-7, if it is SCARA, then it will be left and right type, 0 = right, 1 = left, -1 = current status. ● End_Pos1: the target value of the manipulator first virtual axis (for example, axis X), the unit is mm. ● End_Pos2: the target value of the manipulator second virtual axis (for example, axis Y), the unit is mm. ● End_Pos3: the target value of the manipulator third virtual axis (for example, axis Z), the unit is mm. ● End4: if it is 6-axis robot, which means the target point posture's angle RX. If it is SCARA, which means the linear target point posture's axis RX rotary angle, the unit is degree. ● End5: for 6-axis robot, please fill in this, which means the linear target point posture's RY angle (degree). For SCARA, ignore it. ● End6: for 6-axis robot, please fill in this, which means the linear target point posture's RZ angle (degree). For SCARA, ignore it. <i>If there is End7, End8, End9, they are extended axes' target value. It can be set as rotation axes (degree) or movement axes (mm).</i>
Controller	Valid in ZMC4XX or above, and the firmware should be above 20170511.
Routine	BASE(6,7,8,9,10,11) 'select virtual axis No. CONNREFRAME(6,0,0,1,2,3,4,5) '6 DOF manipulator model, axis 0 ~ axis 5 are joint axes ?"forward mode" BASE(6,7,8,9,10,11)

	FORCE_SPEED=0.8 'set the point to point (PTP) speed of below MOVER2_PABS as 0.8, that is, the 80% full speed. MOVER2_PABS(3,374,0,630,0,90,0) 'the robot moves to target coordinates (374,0,630) through PTP joint interpolation, the target posture value is (0,90,0), the unique solution area when Fstatus=3. MOVER2_PABS(-1,189,-160,600,0,90,0) 'the robot moves to target coordinates (189,-160,600) through linear motion, the target posture value also is (0,90,0), and the unique solution area when Fstatus=3.
Instructions	MOVER2_J/MOVER2_JABS

MOVER2_L/MOVER2_LABS -- Robot Linear Motion

Type	Robot Motion Instruction
Description	Move the manipulator terminal to specified target point's posture through linear motion. This is valid in virtual axes set in BASE. <u>How to set speed</u>: please check "Speed & Accel of PTP Motion & Linear Motion Configurations". MOVER2_LABS = linear motion with absolute position MOVER2_L = linear motion with incremental position
Syntax	MOVER2_L/MOVER2_LABS(Fastatus,End_Pos1,End_Pos2,End_Pos3,End4,End5,...) ● Fastatus: if it is 6 dof robot, it means the type value that determines the unique solution, the range is 0-7, if it is SCARA, then it will be left and right type, 0 = right, 1 = left, -1 = current status. ● End_Pos1: the target value of the manipulator first virtual axis (for example, axis X), the unit is mm. ● End_Pos2: the target value of the manipulator second virtual axis (for example, axis Y), the unit is mm. ● End_Pos3: the target value of the manipulator third virtual axis (for example, axis Z), the unit is mm. ● End4: if it is 6-axis robot, which means the target point posture's angle RX. If it is SCARA, which means the linear target point posture's axis RX rotary angle, the unit is degree. ● End5: for 6-axis robot, please fill in this, which means the linear target point posture's RY angle (degree). For SCARA, ignore it. ● End6: for 6-axis robot, please fill in this, which means the linear target point posture's RZ angle (degree). For SCARA, ignore it. <i>If there is End7, End8, End9, they are extended axes' target value. It can be</i>

	<i>set as rotation axes (degree) or movement axes (mm).</i> Notes: ✓ MOVER2_L is for relative posture's incremental position. For frame102 type 2-axis parallel manipulator, it only has 2D coordinates, then you only need to set End_Pos1 and End_Pos2. ✓ How to set the linear speed and acceleration, please refer to below examples.
Controller	Valid in ZMC4XX or above, and the firmware should be above 20170511.
Routine	<pre> "define motor and manipulator parameters dim R1 'upper space radius dim r2 'bottom space radius dim L1 'upper rod length dim L2 'bottom rod length R1 = 40 'Delta manipulator structure related parameters r2 = 27 L1 = 120 L2 = 230 dim u_m1 'motor 1 pulses in one circle dim u_m2 'motor 2 pulses in one circle dim u_m3 'motor 3 pulses in one circle dim u_m4 'motor 4 pulses in one circle u_m1=480000 u_m2=480000 u_m3=480000 u_m4=640000 dim i_1 'joint 1 transmission ratio dim i_2 'joint 2 transmission ratio dim i_3 'joint 3 transmission ratio dim i_4 'joint 4 transmission ratio i_1=31 i_2=31 i_3=31 i_4=1 dim u_j1 'motor pulses of joint 1 in one circle dim u_j2 'motor pulses of joint 2 in one circle dim u_j3 'motor pulses of joint 3 in one circle dim u_j4 'motor pulses of terminal rotation in one circle u_j1=u_m1*i_1 u_j2=u_m2*i_2 u_j3=u_m3*i_3 u_j4=u_m4*i_4 </pre>


```

""""""""set joint axes""""""""
BASE(0,1,2,3) 'define real axes
'ATYPE = 1,1,1,1 'set as pulse axis
UNITS=u_j1/360,u_j2/360,u_j3/360,u_j4/360 'set units as pulses/degree
FS_LIMIT = 90,90, 150,720 'each joint axis' (+ & -) position limit value
RS_LIMIT = -90,-90, -150,-720
DPOS=0,0,0,0 'set joint axis position, modify it according to real needs
MPOS=0,0,0,0
SPEED=500,500,500,1500 'set each joint axis' max speed
ACCEL=10000,10000,10000,30000
DECEL=10000,10000,10000,30000
'set each joint axis' max acceleration and deceleration

""""""""set virtual axes""""""""
BASE(6,7,8,9)
ATYPE = 0,0,0,0 'set virtual axes
TABLE (0,R1,r2,L1,L2,u_j1,u_j2,u_j3, 0, 0, 0, u_j4)
UNITS = 1000,1000,1000,u_j4/360
'please set this pulse amount in advance, which can't be set while moving.
FS_LIMIT = 20000000,20000000, 20000000,20000000
RS_LIMIT = -20000000,-20000000, -20000000,-20000000
ACCEL=30000,30000,30000,30000
DECEL=30000,30000,30000,30000
FORCE_SPEED(6) = 3000
'set linear speed of linear trajectory, 6 is the virtual master axis, the speed
3000 is only valid in master axis.
FORCE_SPEED(9) = 1500
'set rotary speed of terminal rotation axis RX, 9 is the DELTA rotary axis.

BASE(6,7,8,9) 'select DELTA virtual axes
CONNREFRAME(14,0,0,1,2,3)
'axis 0/1/2/3 are joint axes, open forward mode connection
WAIT LOADED
WA(10)
?"Forward Mode"

SYSTEM_ZSET=0 'check sub-axis speed
TRIGGER

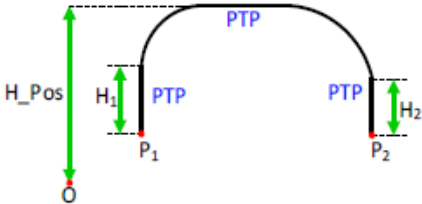
'Delta technology position
BASE(6,7,8,9)
MOVER2_LABS (-1,116.56,8.92117,-211,89.66428)
WAIT IDLE (0)

BASE(6,7,8,9) 'select virtual axes
CONNREFRAME(1,0,0,1,2,3) 'open forward mode connection

```

	WAIT LOADED 'waiting motion loaded, now it will automatically adjust virtual axis position ?"Forward Mode" BASE(6,7,9,8) SPEED_RATIO = 1.0 'speed ratio value is 1.0 1.0 ZSMOOTH=100 'set the smoothing coefficient as 100 MOVER2_LABS(1,300,200,-50,0) 'SCARA robot moves target position (300, 200, -50, 0) from current position by linear motion, it is left hand type MOVER2_LABS(-1,400,300,-100,0) 'SCARA robot moves target position (400, 300, -100, 0) from current position by linear motion, it is left hand type

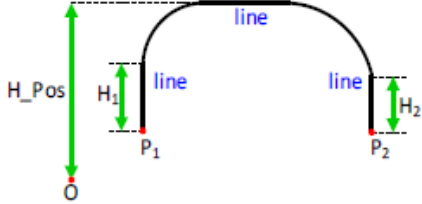
MOVER2_ARCHP/MOVER2_ARCHPABS -- Arch Jog

Type	Robot Motion Instruction (mainly for SCARA manipulator)
Description	Do arch point motion (hopping commands) – the manipulator terminal jumps to one certain height (arch type), then decreases to target position. Like the graphic showing below, the point arch is made by 3 PTP motions.  This is valid in virtual axes set in BASE. <u>How to set speed:</u> please check “Speed & Accel of PTP Motion & Linear Motion Configurations”.
Syntax	MOVER2_ARCHP (H_Pos, H1, H2, Fastatus, End_Pos1, End_Pos2, End_Pos3, End4,...) ● H_Pos: max height coordinates of virtual third axis hopping motion = axis 3 absolute position (like, axis Z), the unit is mm. ● H1: the lifting height (from starting point) of virtual third axis, mm. ● H2: the decreasing height (to target point) of virtual third axis, mm. ● Fastatus: if it is 6 dof robot, it means the type value that determines the unique solution, the range is 0-7, if it is SCARA, then it will be left and right type, 0 = right, 1 = left, -1 = current status. ● End_Pos1: the target value of the manipulator first virtual axis (for example, axis X), the unit is mm. ● End_Pos2: the target value of the manipulator second virtual axis (for example, axis Y), the unit is mm. ● End_Pos3: the target value of the manipulator third virtual axis (for

	example, axis Z), the unit is mm. ● End4: if it is 6-axis robot, which means the target point posture's angle RX. If it is SCARA, which means the linear target point posture's axis RX rotary angle, the unit is degree. ● End5: for 6-axis robot, please fill in this, which means the linear target point posture's RY angle (degree). For SCARA, ignore it. ● End6: for 6-axis robot, please fill in this, which means the linear target point posture's RZ angle (degree). For SCARA, ignore it. <i>If there is End7, End8, End9, they are extended axes' target value. It can be set as rotation axes (degree) or movement axes (mm).</i> Notes: ✓ The arch motion's two sides lifting height is confirmed by H1 and H2, which is not related to ZSMOOTH. And the arc motion's chamfer curve changes with each joint's max acceleration. ✓ For frame102 2-axis parallel manipulator, it only has 2D coordinates, that is, only End_Pos1 and End_Pos2 are set (H1 and H2 are set as 0, H_Pos and P1, P2 are with same height).
Controller	Valid in ZMC4XX or above, and the firmware should be above 20170511.
Routine	BASE(6,7,8,9) 'select virtual axes CONNREFRAME(1,0,0,1,2,3) 'open forward mode connection – SCARA forward mode is formed WAIT LOADED 'waiting motion loaded, now it will automatically adjust virtual axis position ?"Forward Mode" BASE(6,7,8,9) MOVER2_PABS(0,300,-100,-50,0) 'under right hand, run to target point 300,-100,-50 by PTP motion MOVER2_ARCHPABS(-25,10,20,0,300,-400,-50,0) 'under right hand, the max position of arch motion in axis Z direction is limited with -25mm, lifting 10mm from axis Z starting point, then, decreasing 20 to target position 300,-400,-50.

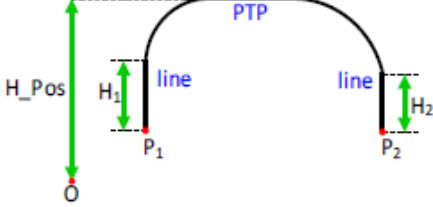
MOVER2_ARCHPL/MOVER2_ARCHPLABS -- Line Arch

Type	Robot Motion Instruction (mainly for DELTA, palletizing manipulators)
Description	Do arch linear motion (hopping commands) – the manipulator terminal jumps to one certain height (arch type), then decreases to target position. Like the graphic showing below, the linear arch is made by 3 linear trajectories.

	 This is valid in virtual axes set in BASE.
Syntax	MOVER2_ARCHPL (H_Pos, H1, H2, Fastatus, End_Pos1, End_Pos2, End_Pos3, End4,...) ● H_Pos: max height coordinates of virtual third axis hopping motion = axis 3 absolute position (like, axis Z), the unit is mm. ● H1: the lifting height (from starting point) of virtual third axis, mm. ● H2: the decreasing height (to target point) of virtual third axis, mm. ● Fastatus: if it is 6 dof robot, it means the type value that determines the unique solution, the range is 0-7, 1 = current status. ● End_Pos1: the target value of the manipulator first virtual axis (for example, axis X), the unit is mm. ● End_Pos2: the target value of the manipulator second virtual axis (for example, axis Y), the unit is mm. ● End_Pos3: the target value of the manipulator third virtual axis (for example, axis Z), the unit is mm. ● End4: if it is 6-axis robot, which means the target point posture's angle RX. If it is SCARA, which means the linear target point posture's axis RX rotary angle, the unit is degree. ● End5: for 6-axis robot, please fill in this, which means the linear target point posture's RY angle (degree). For SCARA, ignore it. ● End6: for 6-axis robot, please fill in this, which means the linear target point posture's RZ angle (degree). For SCARA, ignore it. <i>If there is End7, End8, End9, they are extended axes' target value. It can be set as rotation axes (degree) or movement axes (mm).</i> Notes: ✓ The arch motion's two sides lifting height is confirmed by H1 and H2, which is not related to ZSMOOTH. And the arc motion's chamfer curve changes with each joint's max acceleration. ✓ For frame102 2-axis parallel manipulator, it only has 2D coordinates, that is, only End_Pos1 and End_Pos2 are set (H1 and H2 are set as 0, H_Pos and P1, P2 are with same height).
Controller	Valid in ZMC4XX or above, and the firmware should be above 20170511.
Routine	<pre> BASE(6,7,8,9) 'select virtual axes CONNREFRAME(1,0,0,1,2,3) 'open forward mode connection WAIT LOADED 'waiting motion loaded, now it will automatically adjust virtual axis position ?"Forward Mode" BASE(6,7,8,9) </pre>

	MOVER2_PABS(0,300,-100,-50,0) 'under right hand, run to target point 300,-100,-50 by PTP motion MOVER2_ARCHPLABS(-25,10,20,0,300,-400,-50,0) 'under right hand, the max position of arch motion in axis Z direction is limited with -25mm, lifting 10mm from axis Z starting point, then, decreasing 20 to target position 300,-400,-50.
Instructions	MOVER2_ARCHP , MOVER2_ARCHP3

MOVER2_ARCHP3 / MOVER2_ARCHP3ABS -- 3D Arch

Type	Robot Motion Instruction (mainly for 6 DOF, palletizing manipulators)
Description	Do 3D arch motion (hopping commands) – the manipulator terminal jumps to one certain height (arch type), then decreases to target position. Like the graphic showing below, it is made by 1 linear motion + PTP + 1 linear motion.  This is valid in virtual axes set in BASE.
Syntax	MOVER2_ARCHP3 (H_Pos, H1, H2, Fastatus, End_Pos1, End_Pos2, End_Pos3, End4,...) ● H_Pos: max height coordinates of virtual third axis hopping motion = axis 3 absolute position (like, axis Z), the unit is mm. ● H1: the lifting height (from starting point) of virtual third axis, mm. ● H2: the decreasing height (to target point) of virtual third axis, mm. ● Fastatus: if it is 6 dof robot, it means the type value that determines the unique solution, the range is 0-7, 1 = current status. ● End_Pos1: the target value of the manipulator first virtual axis (for example, axis X), the unit is mm. ● End_Pos2: the target value of the manipulator second virtual axis (for example, axis Y), the unit is mm. ● End_Pos3: the target value of the manipulator third virtual axis (for example, axis Z), the unit is mm. ● End4: if it is 6-axis robot, which means the target point posture's angle RX. If it is SCARA, which means the linear target point posture's axis RX rotary angle, the unit is degree. ● End5: for 6-axis robot, please fill in this, which means the linear target point posture's RY angle (degree). For SCARA, ignore it. ● End6: for 6-axis robot, please fill in this, which means the linear target point posture's RZ angle (degree). For SCARA, ignore it. <i>If there is End7, End8, End9, they are extended axes' target value. It</i>

	<i>can be set as rotation axes (degree) or movement axes (mm).</i> Notes: ✓ The arch motion's two sides lifting height is confirmed by H1 and H2, which is not related to ZSMOOTH. And the arc motion's chamfer curve changes with each joint's max acceleration. ✓ For frame102 2-axis parallel manipulator, it only has 2D coordinates, that is, only End_Pos1 and End_Pos2 are set (H1 and H2 are set as 0, H_Pos and P1, P2 are with same height).
Controller	Valid in ZMC4XX or above, and the firmware should be above 20170511.
Routine	BASE(6,7,8,9) 'select virtual axes CONNREFRAME(1,0,0,1,2,3) 'open forward mode connection WAIT LOADED 'waiting motion loaded, now it will automatically adjust virtual axis position "Forward Mode" BASE(6,7,8,9) MOVER2_PABS(0,300,-100,-50,0) 'under right hand, run to target point 300,-100,-50 by PTP motion MOVER2_ARCHP3ABS(-25,10,20,0,300,-400,-50,0) 'under right hand, the max position of arch motion in axis Z direction is limited with -25mm, lifting 10mm from axis Z starting point, then, decreasing 20 to target position 300,-400,-50.
Instructions	MOVER2_ARCHP , MOVER2_ARCHPL

MOVER2_ARC / MOVER2_ARCABS -- Arc Interpolation

Type	Robot Motion Instruction
Description	Space circular motion, which includes plane arc. Enter arc's middle point and end point. This is valid in virtual axes set in BASE.
Syntax	MOVER2_ARC (Fastatus,midx,midy,midz,endx,endy,endz,enda,endb,encd) ● Fastatus: if it is SCARA, 0 = right, 1 = left. if it is 6 dof robot, it means the type value that determines the unique solution, the range is 0-7, 1 = current status. ● midx: axis X absolute position value of arc's middle point, unit: mm ● midy: axis Y absolute position value of arc's middle point, unit: mm ● midz: axis Z absolute position value of arc's middle point, unit: mm ● endx: axis X absolute position value of arc's target point, unit: mm ● endy: axis Y absolute position value of arc's target point, unit: mm ● endz: axis Z absolute position value of arc's target point, unit: mm ● enda: if it is 6-axis robot, which means the target point posture's angle RX. If it is SCARA, which means the arc target point posture's axis RX rotary angle, the unit is degree. ● endb: for 6-axis robot, please fill in this, which means the arc target

	point posture's RY angle (degree). For SCARA, ignore it. ● endc: for 6-axis robot, please fill in this, which means the arc target point posture's RZ angle (degree). For SCARA, ignore it.
Controller	Valid in ZMC4XX or above, and the firmware should be above 20170511.
Routine	BASE(6,7,8,9) 'select virtual axes CONNREFRAME(1,0,0,1,2,3) 'open forward mode connection WAIT LOADED 'waiting motion loaded, now it will automatically adjust virtual axis position ?"Forward Mode" BASE(6,7,8,9) MOVER2_ARC(0,50,50,-10,100,0,-10,0) 'under right hand, pass by middle point (50,50,-10), then do circular interpolation to target point (100,0,-10)

MOVER2_LIMIT -- Joint SPEED & ACCEL Limit Settings

Type	"Robot Joint Axis Limit" Related Command.
Description	To select each joint axis' max speed and acceleration modes while moving linear / circular motion. After setting new mode value, it becomes valid in following BASIC commands, the default value is 0.
Syntax	MOVER2_LIMIT (mainAxis)=mode ● mainAxis: each manipulator itself main joint axis No. (don't enter other axis No.) ● mode: mode 0: under low-speed, it will not be limited by joint axes. Specifically, when the trajectory speed FORCE_SPEED < 1000mm/s, or the speed ratio < 50%. 0 is the default value. 1: under high-speed & low-speed, the trajectory is limited by each joint-axis' max speed and acceleration.
Controller	General
Routine	BASE(6,7,8,9) 'select virtual axes CONNREFRAME(1,10001,0,1,2,3) 'open forward mode connection, the master axis No. is 0 WAIT LOADED 'waiting motion loaded, now it will automatically adjust virtual axis position ?"Forward Mode" MOVER2_LIMIT (0) = 1 'whole trajectory is limited by joint-axis BASE(6,7,8,9) FORCE_SPEED (6) = 500 'linear speed of linear motion is 500mm/s MOVER2_LABS (-1, 400, 300, -100, 0) 'run one linear motion to target position (400, 300, -100, 0)

ZSMOOTH -- Robot Smooth Coefficient

Type	Axis Parameter.
Description	When axes in BASE are for manipulator, this command is used to set the smoothing coefficient of robot path planning. And the bigger value, the smoothing chamfer between motion trajectories bigger. When it is used for MOVER2 command, the function of ZSMOOTH is different from MOVERABS, MOVER commands. ➤ If BASE are joint axes, then set joint main axis' ZSMOOTH value. ➤ If BASE are virtual axes, set virtual main axis's ZSMOOTH value.
Syntax	VAR1 = ZSMOOTH, ZSMOOTH = smoothdistance Range: 0-100 ● 0: default value, not to do path smoothing. ● 100: the robot running efficiency and trajectory smoothing is high.
Controller	Valid in ZMC4XX or above, and the firmware should be above 20170511.
Routine	BASE(6,7,8,9) 'select virtual axes CONNREFRAME(1,0,0,1,2,3) 'open forward mode connection BASE(6,7,8,9) ZSMOOTH=100.0 'take the value of ZSMOOTH(6) as 100, smoothing coefficient is set as 100. MOVER2_LABS(0,300,300) '100% smoothing MOVER2_LABS(0,350,400) ZSMOOTH=20.0 'take the value of ZSMOOTH(6) as 20 MOVER2_LABS(0,240,360) '20% smoothing MOVER2_LABS(0,-350,-400) BASE(0,1,2,3) ZSMOOTH=50.0 'take the value of ZSMOOTH(0) as 50 MOVER2_JABS(10,20,-50)

MOVER2_ZSMOFFSET -- Smooth Offset Time of Continuous Interpolation

Type	Command relates to manipulator smoothing.
Description	It is the time offset command for smoothing when doing continuous interpolation. Used to increase merged speed between motion commands, like below shown, promoting the efficiency. Principle: when the former motion starts to decelerate, and when the next motion starts to accelerate, the interpolation offset time is bigger, the deceleration is less / no deceleration.

Syntax	MOVER2_ZSMOFFSET(value) value: the value of smoothing time offset during continuous motion, ms. ● Set it as -1, which means internally calculate the best value. ● The unit is ms, the range is 0-1000ms. For example, MOVER2_ZSMOFFSET (60) 'set the smoothing offset time is 60ms ● Set it as 0, which means not to do time offset, but original configured ZSMOOTH value is also valid.
Controller	Valid in ZMC4XX or above, and the firmware should be above 20170511.
Routine	BASE(6,7,8,9) 'select virtual axes CONNREFRAME(1,0,0,1,2,3) 'open forward mode connection, SCARA WAIT LOADED 'waiting motion loaded, now it will automatically adjust virtual axis position ?"Forward Mode" BASE(6,7,8,9) ZSMOOTH=100.0 'smoothing coefficient is set as 100. MOVER2_PABS(-1,2201.2, 4.1,0,17.5) MOVER2_ZSMOFFSET(-1) 'internally calculate the best value to do smoothing time offset, which means the time offset of the interpolation between the former MOVER2_PABS command and the next MOVER2_PABS. MOVER2_PABS(-1,1137.2,941,200,68.9) MOVER2_ZSMOFFSET(0) 'the former MOVER2_PABS command and the next MOVER2_PABS don't do interpolation. MOVER2_LABS(-1,71.9,1876.3,-323.9,120.4) WAIT IDLE(0)

FORCE_SPEED -- Robot Motion Speed Settings

Type	Axis parameter.
Description	It is mainly for MOVER2 point motion (MOVER2_JABS, MOVER2_PABS, MOVER2_ARC, etc.), to set each motion command's speed separately. After setting, manipulator each joint axis' speed and acceleration will become smoother. ➤ For point motion (PTP) Set the point motion (PTP) multi-axis interpolation running speed for main

	axis No. in BASE by FORCE_SPEED command, the range is 0~0.1. When it is 1, it will run at 100% speed. ➤ For MOVER2_JABS (PTP) BASE are joint axes, then take joint axis main axis No. as the running speed of multi-axis interpolation. ➤ For MOVER2_PABS (PTP) BASE are virtual axes, then take virtual axis main axis No. as the running speed of multi-axis interpolation. <i>How to set speed:</i> please check "Speed & Accel of PTP Motion & Linear Motion Configurations".
Syntax	Set speed of point motions (PTP), like, MOVER2_JABS, MOVER2_PABS, the range is 0~1.0.
Controller	Valid in ZMC4XX or above, and the firmware should be above 20170511.
Routine	<pre> BASE(0,1,2,3,4,5) 'select joint axes CONNFRAME(6,0,6,7,8,9,10,11) '6 DOF robotic arm, axis 6 ~ axis 11 are virtual axes XYZABC ?"Forward Mode" BASE(0,1,2,3,4,5) FORCE_SPEED = 0.5 'set MOVER_JABS point motion speed as 0.5, which means 50% full speed MOVER_JABS(60,60,30,20,0,90) BASE(6,7,8,9,10,11) FORCE_SPEED=0.2 'set MOVER2_PABS point motion speed as 0.2, 20% full speed MOVER2_PABS(-1,374,0,630,0,90,0) 'run at speed of 20% FORCE_SPEED=0.4 'set its speed as 40% full speed MOVER2_PABS(-1,374,0,630,0,90,0) 'run at speed of 40% </pre>
Instructions	SPEED_RATIO

MOVER2 PTP & Path Motion SPEED & ACCEL Config

(1) Set Speed & Acceleration

Set manipulator speed and acceleration of PTP motion (point to point), trajectory motion (linear, circular).

The manipulator each joint axis' max speed SPEED correspond to the motor max speed, then each joint axis' max acceleration ACCEL correspond to motor set torque / others. Please note not modify SPEED and ACCEL any more after setting. Because they are the fixed properties of each joint. Therefore, each joint axis max SPEED value should be set in advance. If one certain axis SPEED is set as 0, which will report 1008 error.

If you want to modify each motion command's acceleration and speed, please check below form, different motion types are with different modification methods.

	Point Motion (MOVER2_P, MOVER2_J, Point Arch ARCHP)	Trajectory Motion (MOVER2_L, MOVER2_ARC, Linear Arch ARCHPL)	Trajectory Motion Terminal Rotation Axis RX / Posture Speed (Line, Arc, Linear Arch ARCHPL)
Set Speed	Set BASE joint main axis No. speed through FORCE_SPEED. The range 0-1.0. 1 = run at 100% speed. <u>For Example:</u> Axis 0~axis 6 are 6 DOF joint axes, 0 is the main axis, then FORCE_SPEED (0) = 0.6, which means it runs at the speed of 60% speed.	Set by FORCE_SPEED (mm/s). <u>For Example:</u> BASE(6,7,9,8) 'select virtual axes CONNREFRAME(1,0,57,58,60,59) 'open forward mode BASE(6,7,9,8) MOVER2_LABS(0,364, 0, 0, -90) FORCE_SPEED = 1000 MOVER2_LABS(0,524.8,160.8,0,10) FORCE_SPEED = 1500 MOVER2_LABS(0,524.8,-160.8,0,10)	Modify by current FORCE_SPEED. <u>Example 1:</u> Axis 7~axis12 are 6dof robot virtual axes. Axis 10 is the first axis (RX) of 6-joint posture, then use FORCE_SPEED to set terminal posture speed. <u>Example 2:</u> SCARA manipulator, CONNREFRAME (1, 30060, 57, 58, 60, 59), 60 is RX rotary axis, then, FORCE_SPEED (60) sets terminal posture speed (degree/s)
Set Accel	<u>Basic Codes:</u> FORCE_SPEED (0) = 0.4 (MOVER_JABS runs at 40% SPEED) MOVER_JABS (60, 60, 30, 20, 0, 90)	Modify by current ACCEL and DECEL directly. <u>For Example:</u> BASE(7,8,9,10,11,12) 'select virtual axes for 6DOF robot ACCEL(7)=value, DECEL (7)=value, this value is the axis 7 acceleration and deceleration. Unit: mm/s^2	Modify posture offset speed by current ACCEL and DECEL. <u>For Example:</u> BASE(7,8,9,10,11,12) 'select virtual axes for 6DOF robot ACCEL(10)=value, DECEL(10)=value, "value" is the terminal posture's acceleration and deceleration. Unit: mm/s^2

Notes:

 **For PTP Motion**

Set the speed through modifying joint main axis' FORCE_SPEED / SPEED_RATIO.

For Trajectory Motion (Line / Arc...):

- set the speed through modifying virtual main axis' FORCE_SPEED, if it uses terminal rotary axis RX / terminal posture (like, 6DOF), the speed is set by rotary axis' FORCE_SPEED independently. At the same time, if you don't want to set virtual axis' trajectory speed and acceleration, below are methods:
 - a. set joint main axis CORNER_MODE = 1<<8
 - b. set virtual main axis FORCE_SPEED = 0
 - c. set virtual main axis ACCEL = 0
- low & middle speed means FORCE_SPEED linear speed < 100mm/s, or the speed ratio SPEED_RATIO < 50%, which means it will not be limited by joint axis max speed and acceleration. If you want to limit the joint axes speed, using MOVER2_LIMIT command.

(2) Set Speed Ratio SPEED_RATIO

- If the speed ratio is set by SPEED_RATIO, it only sets joint main axis' SPEED_RATIO for all MOVER2 motion commands.

For example, if the main axis is 0, set SPEED_RATIO (0) = 0.9 to make the motion speed become 90% full speed.

- If you want to modify buffer each motion commands' SPEED_RATIO, using MOVE_PARA to call SPEED_RATIO.

For example:

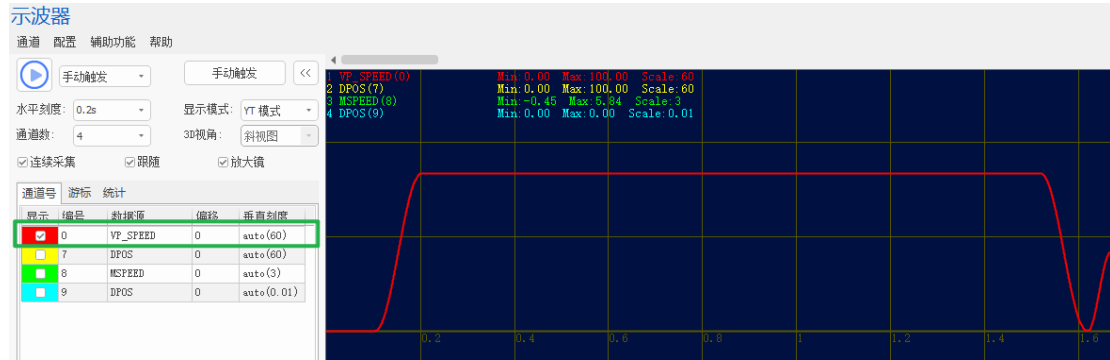
```

BASE(6,7,9,8)           'select virtual axes
CONNREFRAME(1,30060,57,58,60,59) 'forward connection, SCARA, main joint axis is 57
WAIT LOADED
?'Forward Mode'
WA(200)
BASE(6,7,8,9)
MOVE_PARA(SPEED_RATIO,57,0.5) 'set below MOVER2_PABS speed ratio ass 0.5
MOVER2_PABS(-1,300,450,-200)
MOVE_PARA(SPEED_RATIO,57,0.8) 'set below MOVER2_LABS speed ratio ass 0.8
MOVER2_LABS(-1,300,450,-200)
WAIT IDLE(57)
  
```

(3) Set Speed Curve Showing

- $\text{SYSTEM\_ZSET} = 1$ (default value), then the captured curve will show interpolation speed.

For example: the joint main axis is 0, it moves MOVER2_LABS motion, then VP_SPEED(0) in scope shows merged speed value of linear trajectory.



- If $\text{SYSTEM\_ZSET} = 0$, then the captured curve will show single-axis speed.

For example: the joint main axis is 0, $\text{SYSTEM\_ZSET} = 0$, it moves MOVER2_LABS motion, then VP_SPEED(0) in scope shows the single axis speed value of manipulator joint axis 57.

Chapter 8 Program Structure & Instruction

8.1. Program Symbols

' -- 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	Syntax Structure
Description	Make the label for user process, which can be used as SUB process without parameters.
Syntax	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 Definitions

CONST -- Constant Definitions

Type	Syntax Instructions
Description	Define a symbol to indicate constant value, avoid using value directly.
Syntax	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 -- Variables Definitions

Type	Syntax 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.
Syntax	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 -- Local Definitions

Type	Syntax 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.
Syntax	<pre>SUB subA() LOCAL localname 'localname local variable name ENDSUB</pre>
Controller	General
Example	<pre>SUB aaa() LOCAL v1 'define local variable V1 v1=100 END SUB</pre>
Instructions	DIM , GLOBAL

GLOBAL -- Global Definitions

Type	Syntax Instructions
Description	Define global variables, array. Define global SUB process. Global variables can be used in any process file of the whole project.
Syntax	Syntax1: GLOBAL VAR1 Syntax2: GLOBAL SUB SUB1() Syntax3: GLOBAL CONST CVARNAME = value ● 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: <pre>GLOBAL varname = 1</pre> 5. Array definition initialization: <pre>GLOBAL arrayname(size) = {1, 2, 3} GLOBAL arrayname(size) = "string"</pre> 6. Structure definition initialization: <pre>GLOBAL stname(size) as structname = {.item = 1, .item = {1, 2, 3} }</pre>

	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	DIM , LOCAL

8.3. Array Operations

DMINS -- Array Link List Insert

Type	Syntax Instructions
Description	Operations of array link list: when one element is inserted into one array, then the present element and all later elements will move backward one space. Be careful when operating one long size array, especially TABLE.
Syntax	Syntax 1: DMINS arrayname(pos, size) Syntax 2: DMINS arrayname(pos, [size]) ● arrayname: array name ● pos: array index ● size: how many to be modified. For syntax 1, when the “size” is larger than “pos”, it will alarm.
Controller	General
Example	<pre> DIM aa(6) 'define array aa FOR i=0 TO 4 'assign values: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. DMINS aa(1),3 'insert element 0, all behind elements will move backward 3 spaces aa(0) = 7 'assign value to aa(0) ?*aa 'print all array elements after insert operation. </pre>
Instructions	DMDEL , DMCPY

DMADD -- Array Increase (in Bulk)

Type	Syntax Instructions
Description	Add array elements value in batch. Don't modify over 500 elements once.
Syntax	DMADD ArrayName (pos, size, data) ● arrayname: array name ● pos: starting index ● size: element numbers to be modified. Don't exceed array size when adding pos. ● data: the value to be added
Controller	General
Example	<pre>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.</pre>
Instructions	DMINS , DMCPY

DMDEL -- Array Link List Delete

Type	Syntax 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.
Syntax	Syntax 1: DMDEL arrayname (pos, size) Syntax 2: DMDEL arrayname (pos) [,size] ● arrayname: array name ● pos: starting index ● data: the value to be added
Controller	General
Example	<pre>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.</pre>

Instructions	DMINS , DMCPY

DMCPY -- Array Copy

Type	Syntax Instructions
Description	Copy array, starting from array Src to array Des. Be careful when operating long size array, especially TABLE.
Syntax	DMCPY ArrayDes(startpos) , ArraySrc(startpos)[, size] ● arrayname: array name ● startpos: array startinfg index ● size: elements numbers to copy, it will reduce automatically if exceeds maximum value.
Controller	General
Example	<pre> 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 </pre>
Instructions	DMINS , DMDEL

DMSET -- Array Assign

Type	Syntax Instructions
Description	Assign array. Be careful when operating long size array, especially TABLE.
Syntax	DMSET arrayname(pos, size, data) ● pos: starting index ● size: length ● data: the array to be set
Controller	General
Example	<pre> 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 </pre>
Instructions	DMINS , DMDEL

DMCMP -- Array Comparison

Type	Syntax 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.
Syntax	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: ● arr1 > arr2 value = 1 ● arr1 = arr2 value = 0, in comparison range, element values equal ● arr1 < arr2 value = -1
Controller	General
Example	<pre> 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 </pre>
Instructions	DMINS , DMDEL

DMSEARCH -- Array Search

Type	Syntax 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't be searched, it will return -1. Please cautious to oversize arrays operation, especially array TABLE.

Syntax	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	<pre> DIM return, value DIM arr1(10) FOR i = 0 TO 9 Arr1(i) = i NEXT value = DMsearch(arr1,0,1,10,3) return = DMsearch(arr1,0,1,10,20) IF value = 3 AND return = -1 THEN ?"success" ELSE ?"fail" END IF </pre>
Instructions	DMINS , DMDEL

SIZEOFARRAY -- Array Space Get

Type	Syntax Instructions
Description	Get occupied array space.
Syntax	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
Controller	Valid in 5xx series controllers with firmware version above 20180327 Valid in 4xx series controllers of fast version with firmware version above 20190107.
Example	Example 1: the number of returned arrays <pre> 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 </pre>

	<pre>?SIZEOFARRAY(aa) 'print result : 12</pre> Example 2: the number of returned structural variables / arrays <pre>'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</pre>
Instructions	DMINS , DMDEL

8.4. Self-Defined Sub Function

SUB -- Self-Defined Subfunction SUB

Type	Syntax Instructions
Description	Users custom SUB process, GLOBAL description can be added before to define SUB process for global use.
Syntax	<pre>SUB label([para1] [,para2]...)</pre> ... <pre>END SUB</pre> ● label: process name, it can't be same as current key words. ● para1: transferred parameters when calling SUB, and it is used as LOCAL local variables automatically. ● para2: transferred parameters when calling SUB, and it is used as LOCAL local variables automatically. Below is supported in VPLC5XX series controllers / Linux controllers, and the firmware version after February 2022: <pre>SUB subname(BYREF paraname[(dimsize)] [AS structname])</pre> ● subname: sub name ● dimsized: the length of the array, must be defined as a constant

	● structname: the name of the structure type, it can transfer ZVOBJ through BYREF method. ✓ BYREF represents a quote, and currently, the calling side fills in corresponding type's variables or array. ✓ The default BYVAL means transfer by "copying", 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 method (assign array multiple elements). ✓ 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: <pre> 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 </pre> Example 2: valid in 5xx series controllers or above <pre> 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 </pre>

	<pre> ?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 </pre>
Instructions	SUB PARA , SUB_IFPARA

SUB_PARA -- SUB Parameters Transferring

Type	Syntax Instructions
Description	Choose input parameters of SUB.
Syntax	SUB_PARA(address) ● address: NO. of input parameters, starts from 0.
Controller	General
Example	<pre> 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 </pre>
Instructions	SUB , SUB_IFPARA

SUB_IFPARA -- SUB Input Parameters Judge

Type	Syntax Instructions
Description	Judge if SUB parameters were input.
Syntax	SUB_IFPARA(address) -1: already input, 0: -not input ● address: NO. of input parameters, starts from 0.
Controller	General
Example	<pre> 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 </pre>

	?0 ENDIF END SUB
Instructions	SUB , SUB PARA

GOSUB/CALL -- SUB Call

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.
Syntax	GOSUB/CALL label ● label: SUB name
Controller	General
Example	<pre> 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 </pre>

GOSUB -- Self-Defined Subfunction in G Code

Type	Syntax Instructions

Description	Users customize GSUB process. GLOBAL description can be added before to define global use GSUB process. When call GSUB, it will follow G code Syntax, no need to add ().	
Syntax	GSUB label([char1] [,char2]...) ... END SUB ● 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	<pre> G01 X100 Y100 Z100 U100 'call G01 END 'main process ends. GLOBAL GSUB G01(X, Y, Z, U) 'define GSUB process G01 ... END SUB </pre>	
Instructions	GSUB_PARA , GSUB_IFPARA	

GSUB_PARA -- GSUB Input Parameters

Type	Syntax Instructions
Description	Choose input parameters of GSUB.
Syntax	GSUB_PARA(char) ● char: input alphabet parameter when define GSUB
Controller	General
Example	<pre> 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 </pre>
Instructions	GSUB , GSUB_IFPARA

GSUB_IFPARA -- GSUB Input Parameters Judge

Type	Syntax Instructions
Description	Judge if GSUB parameters were input.
Syntax	GSUB_IFPARA(char) -1-already input 0-not input

	char: input alphabet parameter when define GSUB
Controller	General
Example	<pre> 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 </pre>
Instructions	GSUB , GSUB_PARA

END SUB -- Self-Defined Function End

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.
Syntax	RETURN
Controller	General
Example	<pre> CALL sub1 ?RETUEN 'result is 111 END 'main procedure ends SUB sub1() RETURN 111 'return 111 END SUB </pre>

XSUB -- Customized 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.
Syntax	Syntax is the same as SUB. When calling, it can use the Syntax of paraname = value.
Controller	Valid in VPLC5XX controllers and above.
Example	Example 1: <pre>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</pre> Example 2: <pre>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</pre>
Instructions	SUB , RSUB

RSUB -- Self-Defined 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.
Syntax	Syntax is the same as SUB. When calling, it can use the Syntax of paraname = value.
Controller	General
Example	<pre>subR ARR2=a2, VAR1 =a1, ARR3=pos4 RSUB subR(byref var1, byref arr2(100), byref arr3(2) as pos) ?SUB_IFPARA(0)</pre>

	?var1 ?arr2(1) ?arr3(1).a END SUB
Instructions	SUB , XSUB

8.5. Structural Definitions

STRUCTURE -- Structural Body Definitions

Type	Syntax instruction
Description	Definition of structural body.
Syntax	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	With firmware version above 5 xxx serials of 20180327 support. With firmware fast version above 4 xxx serials of 20190107 support.
Example	<pre>'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</pre>
Instructions	DIM , GLOBAL , UNION

UNION -- Union Definitions

Type	Syntax Command
Description	Define the union.
Syntax	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 It can use FLASH_WRITE, FLASH_READ commands 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 is same as above. It can array operation instructions to operate structural arrays. 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 can operate the last level arrays, can't assign structural array. DMSET structure variables. item array (index, size, data) DMADD: same as above.
Controller	Valid in VPLC5XX controllers whose firmware version is above 20180327. Valid in ZMC4XX controllers whose “fast” firmware version is above 20190109.
Example	<pre>'state the union AA GLOBAL UNION ClassAA DIM AA_val1 'member variables DIM BB_val1 'member variables DIM AA_array(4) 'member arrays END UNION 'create the union variables GLOBAL Class1 AS ClassAA Class1.AA_val1=123 ?Class1.AA_val1 'the result: 123</pre>

	Class1.BB_val1=456 ?Class1.BB_val1 ' the result: 456 ?Class1.AA_val1 ' the result: 456 END
Instructions	DIM , GLOBAL , STRUCTURE

8.6. Jump Instructions

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.
Syntax	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 -- Conditional Jump

Type	Procedure Structure
Description	When expression is true, then call label process.
Syntax	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 -- Conditional Jump 2

Type	Procedure Structure
Description	Condition jump, procedure jump when expression is true, called process will not be pushed onto stack.
Syntax	ON expression GOTO label ● expression: judgement condition ● label: jump to sub or label
Controller	General
Example	<pre> a=100 on a>10 goto label1 a=1000 END 'main procedure ends label1: PRINT a END 'can not return when use goto jump.</pre>

8.7. Conditional Judgement Instructions

IF -- Condition Judgement Structure

Type	Procedure Structure
Description	Condition Judgement, its structure same as standard BASIC Syntax.
Syntax	<pre> IF <condition1> THEN commands ELSEIF <condition2> THEN commands ELSE commands ENDIF Parameters condition1 condition condition2 condition</pre>
Controller	General
Example	<pre> Example one: DIM a 'define variable a=12 'assign value IF a>11 then 'judgement condition TRACE "the val a is bigger then 11" ELSEIF a<11 then TRACE "the val a is less then 11"</pre>

	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 Instructions

FOR -- “for” Cycle

Type	Procedure Structure
Description	“Loop”, it uses standard BASIC Syntax.
Syntax	FOR variable=start TO end [STEP increment] commands NEXT variable ● 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.
Syntax	WHILE condition ... WEND
Controller	General

Example	<pre> a=0 WHILE IN(4)=OFF 'exit cycle until input 4 is ON. a=a+1 PRINT a DELAY(1000) WEND </pre>

WEND -- While Cycle

Type	Procedure Structure
Description	See: WHILE
Controller	General

EXIT -- Exit Cycle

Type	Procedure Structure
Description	Exit the cycle; if the loop is nested multiple times, it will exit the loop one layer by one layer.
Syntax	EXIT FOR, EXIT WHILE
Controller	General
Example	<pre> LOCAL a FOR a=1 TO 100 'cycle from 1 to 100 PRINT a IF a> 20 THEN ?"times > 20, ready to exit "For" loop" EXIT FOR 'end the current loop ENDIF NEXT </pre>

REPEAT -- Conditional Cycle

Type	Procedure Structure
Description	Cycle sentence. Execute commands by cycle, exit cycle when condition is true.
Syntax	REPEAT commands UNTIL condition
Controller	General
Example	<pre> 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 </pre>

UNTIL -- Condition Structure

Type	Procedure Structure
Description	See: REPEAT, WAIT
Controller	General

8.9. Waiting (Execution) Instructions

DELAY -- Time Delay

Type	Syntax Instructions
Description	Set delay time, unit is ms. Other name: wa
Syntax	DELAY (delay time) ● delay time: how long (ms)
Controller	General
Example	DELAY(100) 'delay 100ms

WAIT UNTIL -- Time for Condition Met

Type	Procedure Structure
Description	Wait until condition is met.
Syntax	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.

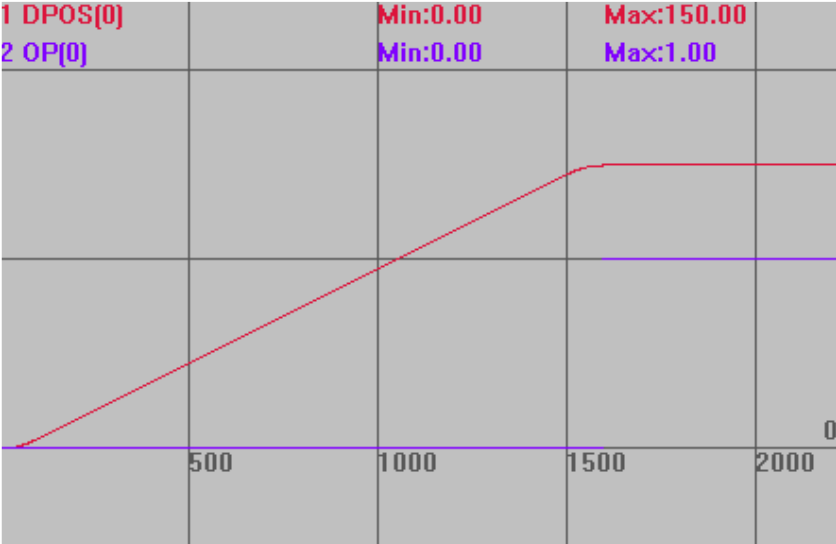
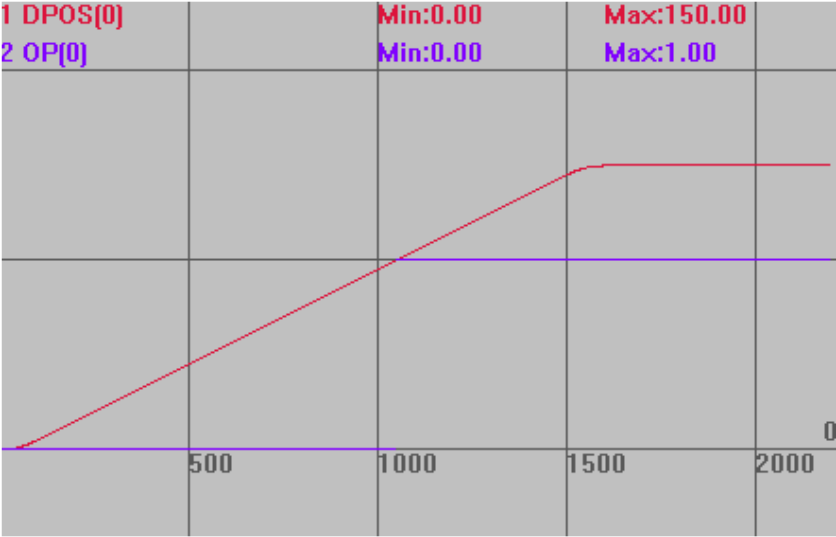
WAIT IDLE -- Wait Until Axes Stop

Type	Syntax 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 Syntax of axis parameters. Note: controller succeeds in sending motion that doesn't represent servo executed.
Syntax	WAIT IDLE
Controller	General
Example	Example 1: <pre>BASE(0,1) MOVE(100,100) WAIT IDLE 'wait until present interpolation motion end</pre> Example 2: <pre>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"</pre>

WAIT LOADED -- Wait Until Axes Buffer Clears

Type	Syntax 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 Syntax of axis parameters.
Syntax	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 1 DPOS[0] 2 OP[0] Min:0.00 Min:0.00 Max:150.00 Max:1.00  WAIT LOADED 'when use wait loaded, followed procedures will execute once buffer clears. OP(0,ON) 1 DPOS[0] 2 OP[0] Min:0.00 Min:0.00 Max:150.00 Max:1.00 

8.10. ZINDEX Pointer Instructions

ZINDEX_LABEL -- Pointer Index Build

Type	Syntax Instructions.
Description	Build pointer index, then it is convenient for behind to call pointer.
Syntax	Pointer = zindex_label (subname) subname: array or SUB name
Controller	General
Example	<pre> 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 </pre>
Instructions	ZINDEX_CALL , ZINDEX_ARRAY , ZINDEX_VAR

ZINDEX_CALL -- SUB Function Access

Type	Syntax Instructions.
Description	Call SUB function through index pointer.
Syntax	ZINDEX_CALL (zidnex) (subpara, ...) - zidnex: index pointer generated from ZINDEX_LABEL - subpara: sub parameters calling
Controller	General
Example	<pre> Pointer = ZINDEX_LABEL (sub1) 'build index pointer ZINDEX_CALL (Pointer) (2) 'call function SUB sub1 (a) PRINT a END SUB </pre>
Instructions	ZINDEX_LABEL

ZINDEX_ARRAY -- Array Access

Type	Syntax Instructions.
Description	Access array through index pointer.
Syntax	var = ZINDEX_ARRAY (Pointer) (index) - pointer: pointer index generated from ZINDEX_LABEL - index: array index
Controller	General
Example	<pre> DIM arr1(100) 'define array </pre>

	<pre>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</pre>
Instructions	ZINDEX_LABEL

ZINDEX_VAR -- Variables Access

Type	Syntax Instructions.
Description	Access variables through index pointer.
Syntax	ZINDEX_VAR (zindex) zindex: index pointer generated from ZINDEX_LABEL <pre>zindex= ZINDEX_LABEL(varname) ZINDEX_VAR(zindex)=value VAR2 = ZINDEX_VAR(zindex)</pre>
Controller	General
Example	<pre>global gTestVar global VarAdd1 VarAdd1=ZINDEX_LABEL(gTestVar) ZINDEX_VAR(VarAdd1)=10 ?ZINDEX_VAR(VarAdd1)</pre>
Instructions	ZINDEX_LABEL

ZINDEX_STRUCT -- Structure Access

Type	Syntax Instructions.
Description	Access structural variables or arrays through pointer after getting the pointer of structural variables.
Syntax	GLOBAL structarname(num) As structname <pre>zindex = ZINDEX_LABEL(structarname) ZINDEX_STRUCT(structname,zindex)(arrindex).item = var var = ZINDEX_STRUCT(structname,zindex)(arrindex).item</pre> - structarname: generated structural arrays, variables. - num: the number of generated structural arrays and variables elements. - zindex: generated index pointer through ZINDEX_LABEL - arrindex: structural array subscript - structname: structure name - item: structure member
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: <pre> 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 </pre> Example 2: <pre> 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 </pre>
Instructions	ZINDEX_LABEL

ZINDEX_MARK -- Pointer Mark No. Settings

Type	Syntax Instructions.
Description	Set mark No. for label, then it could record the label's array index.
Syntax	<pre> ZINDEX_MARK(labelname) = mark varmark = ZINDEX_MARK(labelname) </pre> ● labelname: the name; if different files have own definitions, the results called are different. ● mark: mark No.
Controller	Valid in the controller that supports ZV function / the series that is above 5.

Example	<pre> dim var1 dim arr1(10),MarkArr(10) dim ArrIndex,VarIndex arr1(0,1,2,3,4,5,6,7,8,9,10) VarIndex = zindex_label(var1) 'get the variable pointer and array pointer mark No. ArrIndex = zindex_label(arr1) zindex_mark("VarIndex") = 0 'set the variable pointer and array pointer mark No. zindex_mark("ArrIndex") = 1 markarr(zindex_mark("VarIndex")) = VarIndex 'save the pointer into assigned array subscript through obtained mark No. markarr(zindex_mark("ArrIndex")) = ArrIndex zindex_var(MarkArr(zindex_mark("VarIndex"))) = 777 'access variable and array that correspond to pointer array through mark No. ?zindex_var(MarkArr(0)) ?var1 ?zindex_array(MarkArr(zindex_mark("ArrIndex")))(9) ?zindex_array(MarkArr(1))(9) end </pre>
Instructions	<u>ZINDEX_LABEL</u>

Chapter 9 Task Instructions

RTBASIC supports real-time multi-task operations, and multi-task can be run in one file.

It starts to run the task from the first row through RUN.

And it can assign any SUB processes through RUNTASK.

9.1. Task Start & Stop

RUN -- File Task Start

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
Syntax	RUN "filename"[, tasknum] ● filename: file name, don't 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 -- SUB TASK Start

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.
Syntax	RUNTASK tasknum, label ● tasknum: Task No. ● label: self-defined SUB process (it can attach parameters) or label

Controller	General
Example	<pre> 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 </pre>
Instructions	RUN

END -- Task End

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 -- File Task Stop

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.
Syntax	STOP program name, [tasknum] ● program name: 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	<pre> RUN aaa, 1 'execute aaa.bas STOP aaa, 1 'stop task 1 </pre>
Instructions	STOPTASK , HALT

STOPTASK -- SUB Task Stop

Type	Task Instructions
Description	Force to stop task. Operate SUB and Label. Stop the tasks before restart tasks.
Syntax	STOPTASK [tasknum] ● tasknum: task NO., default value is present task NO.
Controller	General
Example	STOPTASK 2 'stop task 2
Instructions	STOP , HALT

HALT -- All Tasks Stop

Type	Task Instructions
Description	Stop all tasks. This instruction is only used to PC software calling. It will stop the whole process if it is used in BASIC, and the controller can't work.
Syntax	HALT
Controller	General
Example	HALT 'stop all tasks
Instructions	STOP , STOPTASK

PAUSE -- All Tasks Pause

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 calling. It will stop the whole process if it is used in BASIC, and the controller can't work. Task will continue when it resumes after pause.
Syntax	PAUSE
Controller	General
Example	PAUSE 'pause all tasks.
Instructions	PAUSETASK

PAUSETASK -- Assigned Tasks Pause

Type	Task Instructions
Description	Pause one specific task.

	Task will continue when it resumes after pause.
Syntax	PAUSETASK tasknum ● tasknum: task No., default value is present task No.
Controller	General
Example	PAUSETASK 1 'Pause task 1.
Instructions	RESUMETASK

RESUMETASK -- Assigned Task Resume

Type	Task Instructions
Description	Resume a specific task. Task will continue when it resumes after pause.
Syntax	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. File 3 (Z3P) Tasks Instructions

9.3. Task Parameters Instructions

Chapter 10 Operator & Math Instructions

RTBASIC supports all operational characters in standard BASIC Syntax, 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 Operational Instructions

10.2. Comparison Operational Instructions

10.3. Logical Operational Instructions

10.4. Trigonometry Instructions

10.5. Exponentiation Operational Instructions

10.6. Data Operational Instructions

10.7. Character String Operational Instructions

10.8. Constant Instructions

10.9. Advanced Operational Instructions

Chapter 11 Axis Parameter & Axis Status Instructions

Axis parameters modification syntax of example SPEED parameter:

SPEED = value – for axis in BASE

- **SPEED_AXIS** (axisnum) can be used to assign one specified axis, actually “axis” can be omitted, then it becomes “SPEED (axisnum) = value”.
- When there are several axes in BASE, **SPEED = value 1, value 2...** can be use to set corresponding axes’ speed of BASE.
- Axis parameters all can be read and written. While reading, “axis” also can be omitted, it becomes **VAR1 = SPEED(axisnum)**.
- Axis states only can be read generally, the value will change, except parameters that can be modified directly, like, MPOS, DPOS.

 **when in interpolation movement, take spindle axis’ parameters as the interpolation’s parameter. That is, when there are several axes in BASE, the first axis is the spindle.**

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.
Syntax	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 Selection

Type	Assistant instruction
Description	Modify a motion instruction or axis parameter temporarily to execute a defined axis. For axis parameters, AXIS can be omitted.
Syntax	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 Instructions

UNITS -- Pulse Amount

Type	Axis Parameters
Description	Pulse Amount, how many pulses sent every unit (the motor moves 1mm / the motor rotates 1°), 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 Related DPOS=3000, MPOS=3000 After Modification: UNITS=100 Related DPOS=300, MPOS=300
Syntax	For read: VAR1=UNITS / VR1=UNITS(axis No.) For written: UNITS=expression / UNITS (axis No.) = expression
Controller	General
Example	How to set Suppose Motor U needs 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 related UNITS value of 1° rotation, as below: $\text{UNITS} = U/360 = 3600/360 = 10, \quad \text{now MOVE}(1), \text{'motor runs } 1^\circ$ Set UNITS value of 1 mm movement, as below: $\text{UNITS} = U/P = 3600/2 = 1800, \quad \text{now MOVE}(1), \text{'guide screw runs } 1^\circ$ 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: $\text{UNITS} = U*i/P = 3600*2/2 = 3600$ How to Program Situation 1: UNITS is set by axis No. in sequence: BASE(0,1,2) 'choose axis 0,1,2. UNITS=10,100,1000 'UNITS of axis 0,1,2,3 is 10,100,1000 Situation 2: when the data is more than selected BASE axe numbers, please set it continuously. 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 Situation 3: when the data is less than selected BASE axe numbers, please set it in real. BASE(0,1,2) 'choose axis 0,1,2. UNITS=10,100 'UNITS of axis 0,1 is 10,100 Situation 4: set specified axis UNITS(2)=100 'set UNITS of axis 2 directly, no influence from BASE list.

ATYPE -- Axis Type

Type	Axis Parameters																																
Description	Set axis type, only axis available properties can be set. (please check axis type in hardware user manual / check it in RTSys.) It is better to set ATYPE before initialization. For ZCAN extended axes, please set AXIS ADDRESS at first, and call the motion instructions after delaying 2 ticks. Due to limit of field bus bandwidth, extended axes through ZCAN can't 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 RTSys.																																
Syntax	To Read: VAR1 = ATYPE or VR1 = ATYPE (axis No.) To Write: ATYPE = expression or ATYPE (axis No.) = expression <table border="1"> <thead> <tr> <th>ATYPE</th><th>Descriptions of Axis Type</th></tr> </thead> <tbody> <tr><td>0</td><td>Virtual-Axis</td></tr> <tr><td>1</td><td>Stepper / Servo of Pulse Direction</td></tr> <tr><td>2</td><td>Servo by Analog Signal Control</td></tr> <tr><td>3</td><td>Quadrature Encoder</td></tr> <tr><td>4</td><td>Pulse Dir OUT+ Quadrature Encoder IN</td></tr> <tr><td>5</td><td>Pulse Dir OUT + Pulse Dir Encoder IN</td></tr> <tr><td>6</td><td>Encoder of Pulse Dir</td></tr> <tr><td>7</td><td>ATYPE 1 + EZ Signal IN</td></tr> <tr><td>8</td><td>ATYPE 1 Expansion by ZCAN</td></tr> <tr><td>9</td><td>ATYPE 3 Expansion by ZCAN</td></tr> <tr><td>10</td><td>ATYPE 6 Expansion by ZCAN</td></tr> <tr><td>20</td><td>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-V22.</td></tr> <tr><td>21</td><td>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.</td></tr> <tr><td>22</td><td>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-V22.</td></tr> <tr><td>24</td><td>Remote Encoder Axis Used in ZHD500X handwheel, some controllers support.</td></tr> </tbody> </table>	ATYPE	Descriptions 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-V22.	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-V22.	24	Remote Encoder Axis Used in ZHD500X handwheel, some controllers support.
ATYPE	Descriptions 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-V22.																																
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-V22.																																
24	Remote Encoder Axis Used in ZHD500X handwheel, some controllers support.																																

	25	Define one encoder axis, coordinates are read from MODBUS / NODE_PDOBUF. 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_PDOBUF (slot, nodenum, index, subindex, type) ENCODER_ID: No. that saves data dictionary
	26	Customized 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 ", 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 Settings

Type	Axis Parameters												
Description	Set axes addresses for 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. Set AXIS_ADDRESS at first, then set ATYPE as extended ZCAN axes. And when modified, ATYPE must be reset. See example 1 for reference. <table border="1"> <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 ‘CAN expansion board’s local axis 0. AXIS_ADDRESS (axis No.) = (32*1) + CAN ID ‘CAN expansion board’s local axis 1. 2. Bus driver axis No. mapping, map connected drives correspondingly according to number 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. Set AXIS_ADDRESS at first, then set ATYPE as ECAT axis. And when modified, ATYPE must be reset. See example 2 for reference. <table border="1"> <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), and each</td></tr> </table>	Bit 1-4	CAN address DIP code, combination value is 0-15	Bit 5-6	CAN speed DIP code, different values have different speed.	Bit 7	Special function: Reserved	Bit 8	120ohm resistor DIP code, be connected when ON.	Bit 0-15	Drive No.+1, 0-Auto Assign	Bit 16-31	Slot No. (when there are multiple slots), and each
Bit 1-4	CAN address DIP code, combination value is 0-15												
Bit 5-6	CAN speed DIP code, different values have different speed.												
Bit 7	Special function: Reserved												
Bit 8	120ohm resistor DIP code, be connected when ON.												
Bit 0-15	Drive No.+1, 0-Auto Assign												
Bit 16-31	Slot No. (when there are multiple slots), and each												

slot address No. spaces 65535, such as, the slot 0 address is 0-65535, slot 1 address will be 65535-131071.

Rule:

$$\text{AXIS\_ADDRESS (Axis No.)} = (\text{Slot No.} \ll 16) + \text{Drive No.} + 1$$

3. Local pulse axis No. remapping (ZMC4XX series motion controllers support local pulse / encoder axis No. remapping, and the firmware must be 160608 or above)

While remapping, please set original local pulse axis type as virtual axis.

And when modified, ATYPE must be reset. See example 4 for reference.

Bit 0-15	Mapped local pulse axis No.
Bit 16-31	High 16-bit are set as 1 (under decimal system, high 16-bit = -1).

Rule:

BASE (axis No. to be remapped)

ATYPE=0 -- set axis type as 0, for low version, it will report errors if it is not 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. MotionRT control card's sub-card pulse-axis & encoder axis mapping.

While remapping, please AXIS_ADDRESS at first, then set ATYPE. And when modified, ATYPE must be reset.

Bit 0-15	Axis No. of sub-card + 1
Bit 16-31	Sub-card CARD No.

Rule:

BASE (axis No. to be remapped)

ATYPE=0 -- set axis type as 0, for low version, it will report errors if it is not set.

BASE (local pulse axis No. to be modified)

ATYPE=0 -- set axis type as 0

AXIS_ADDRESS (remapped axis No.) = (sub-card No. << 16) + sub-card corresponding physical axis No. + 1

BASE (axis No. to be remapped)

ATYPE = X – X is the required axis type value

5. Cancel axis mapping: AXIS_ADDRESS = 0

BASE (remapped axis No.)

	ATYPE (remapped axis No.) = 0 AXIS_ADDRESS = 0
Syntax	VAR1 = AXIS_ADDRESS, AXIS_ADDRESS = expression
Controller	General
Example	Example 1: extended axis by ZCAN 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 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 16, which corresponds to controller axis 0 interface.
	Example 5: galvanometer axis No. 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

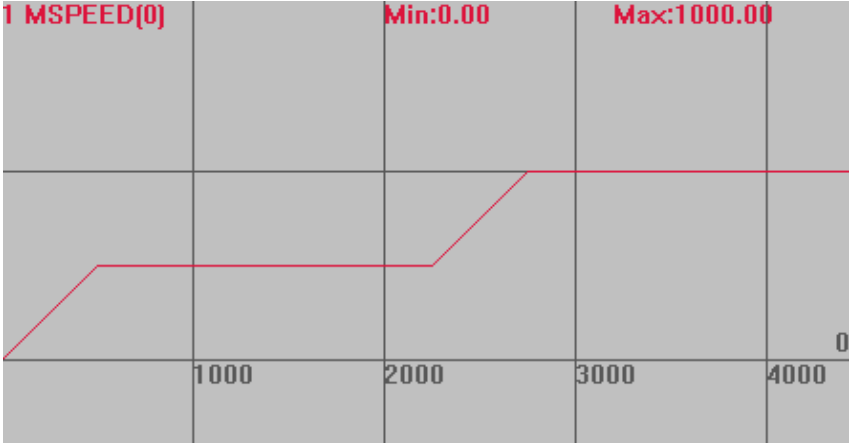
AXIS_ENABLE -- Axis Enable

Type	Axis Parameters
Description	Enable each axis. EtherCAT Bus axis should be configured, and WDOG=1 general enable must be set.
Syntax	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 Parameters Instructions

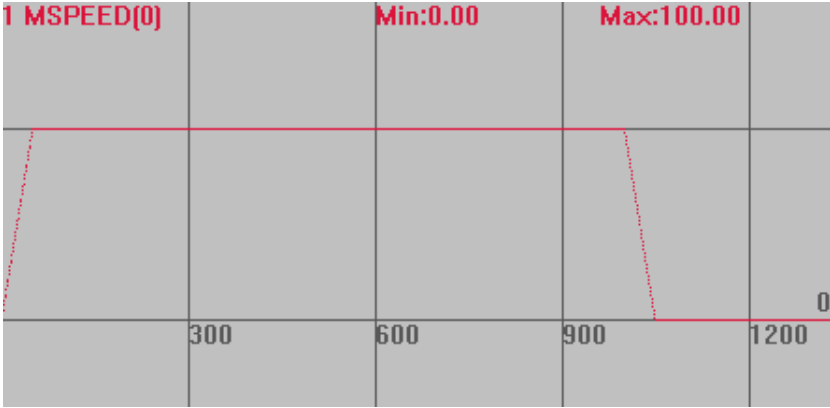
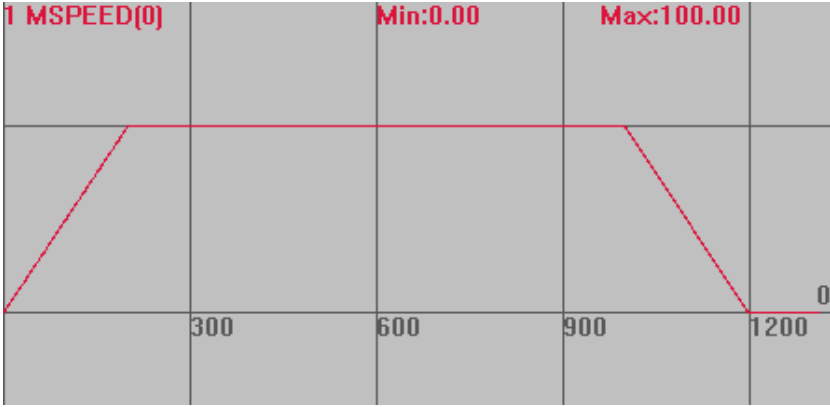
SPEED -- Motion Speed

Type	Axis Parameters
Description	Axis speed, unit is units/s. If the pulse amount is set as 1mm, SPEED is set as 100, that is, it will move 100mm each second. When multi-axis is in motion, SPEED as interpolation motion speed. After modification of SPEED, SPEED will take effect immediately, dynamic speed can be done in this way, but the moment of speed changing, it may cause speed jumping, which will also cause machine vibration, then SPEED_RATIO can be used to realize smoothing. When SP instruction is used: and if FORCE_SPEED is more than SPEED, SPEED will also take effect. (SPEED will not take effect in this situation in

	firmware version above 140716).
Syntax	To Read: VAR1 = SPEED or VAR1 = SPEED (axis No.) To Write: SPEED = expression or SPEED (axis No.) = 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

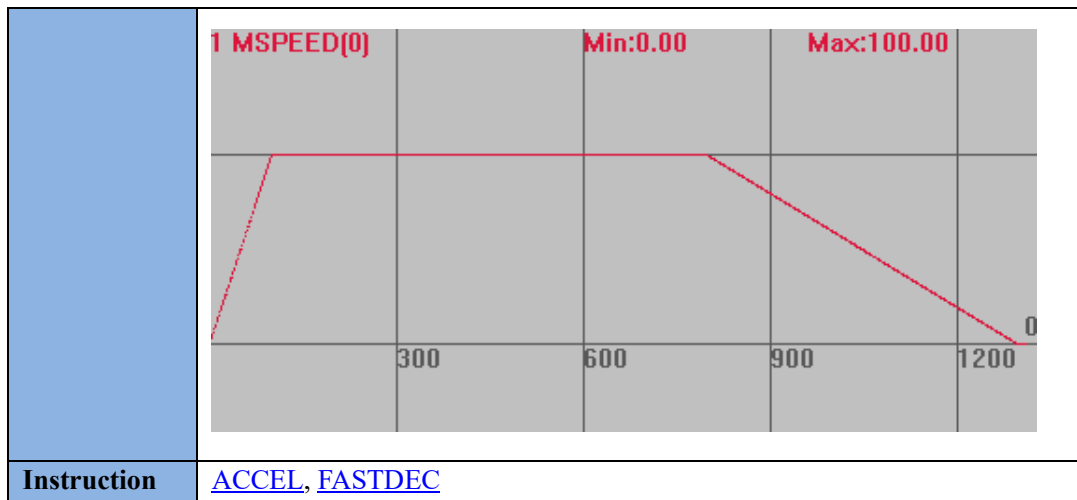
ACCEL -- Axis Acceleration

Type	Axis Parameters
Description	Axis acceleration, unit is units/s/s. If SPEED is set as 100, ACCEL is set as 1000, that is, when it accelerates to 100, the time is 100/1000 seconds. 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, otherwise, it will cause change of speed curve.
Syntax	To read: VAR1 = ACCEL or VAR1 = ACCEL (axis No.) To write: ACCEL = expression or ACCEL (axis No.) = expression
Controller	General

Example	Example 1 BASE(1,2,3,4) ACCEL=100, 100, 100, 100 ACCEL(2)=200 'select axis 'set acceleration of axis 1,2,3,4 'set acceleration of axis 2.
	Example 2 BASE(0) UNITS=100 DPOS=0 ACCEL=2000 SPEED=100 MOVE(100) 'pulse amount 'coordinate clears
	Speed curve of Acceleration Process MSPEED(0)=100(vertical scale) 
	Speed curve after acceleration and deceleration ACCEL=500 DECEL=500 
Instruction	DECEL, SPEED

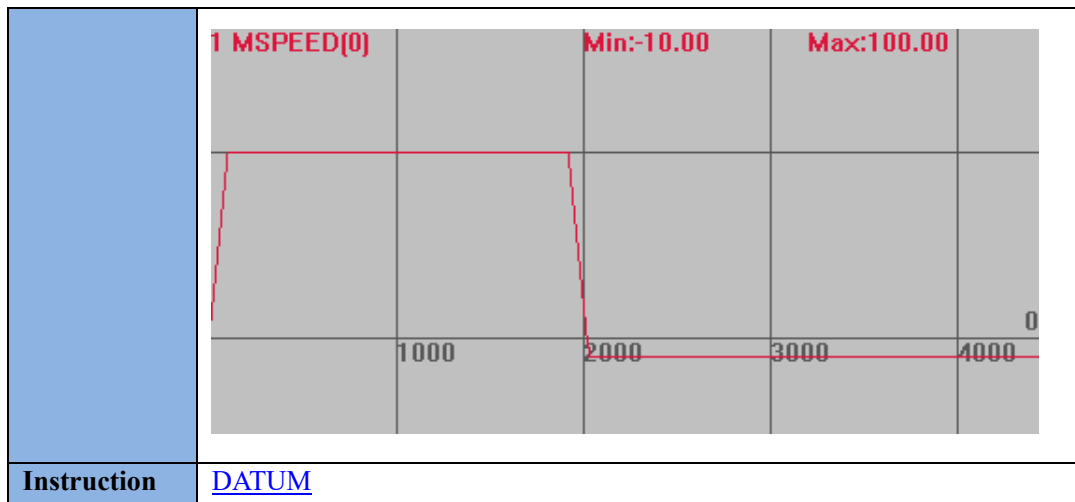
DECEL -- Axis Deceleration

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, otherwise, it will cause change of speed curve.
Syntax	To read: VAR1 = DECEL or VAR1 = DECEL (axis No.) To write: DECEL = expression or DECEL (axis No.) = 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



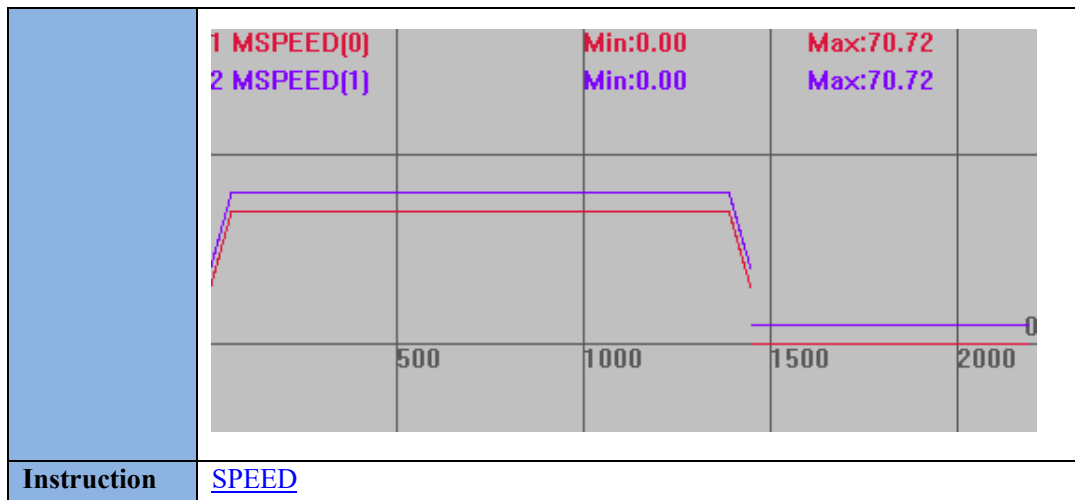
CREEP -- Creep Speed

Type	Axis Parameters
Description	Axis creep speed while homing, which is used to search origin point, unit is units/s.
Syntax	To read: VAR1 = CREEP or VAR1 = CREEP (axis No.) To write: CREEP = expression or CREEP (axis No.) = expression
Controller	General
Example	<pre> 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 </pre>



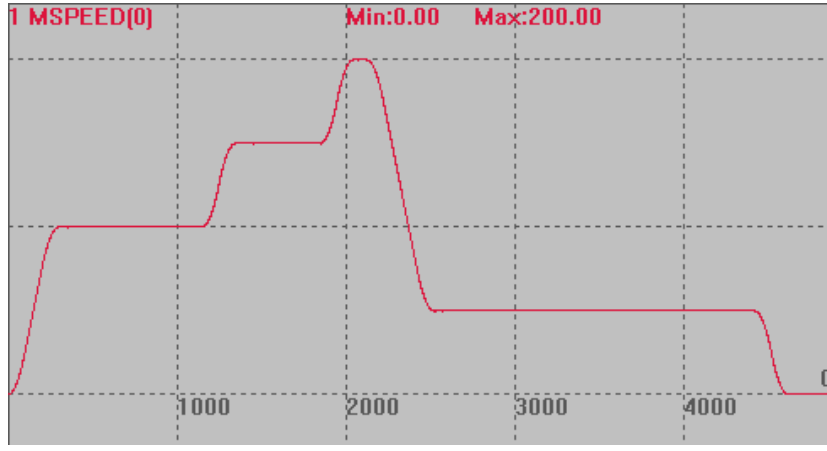
LSPEED -- Initial Speed

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.
Syntax	To read: VAR1 = LSPEED or VAR1 = LSPEED (axis No.) To write: LSPEED = expression or LSPEED (axis No.) = 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



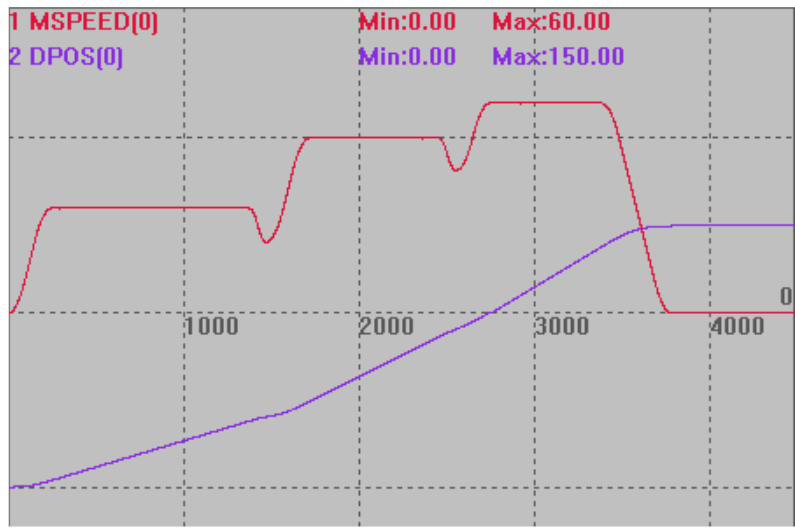
FORCE_SPEED -- SP Speed

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 > 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.
Syntax	To read: VAR1 = FORCE_SPEED or VAR1 = FORCE_SPEED (axis No.) To write: FORCE_SPEED = expression or FORCE_SPEED (axis No.) = expression
Controller	General
Example	<pre> 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 </pre>

	END
	Speed Curve: MSPEED(0)=100(vertical scale) 
Instruction	<u>*SP</u>

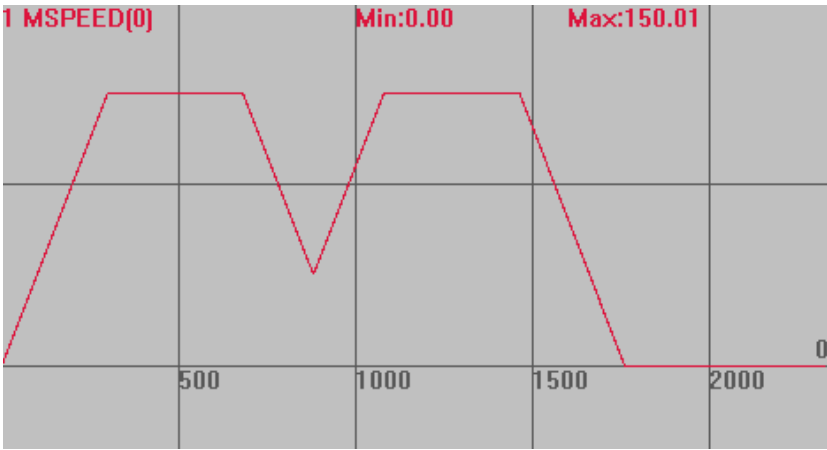
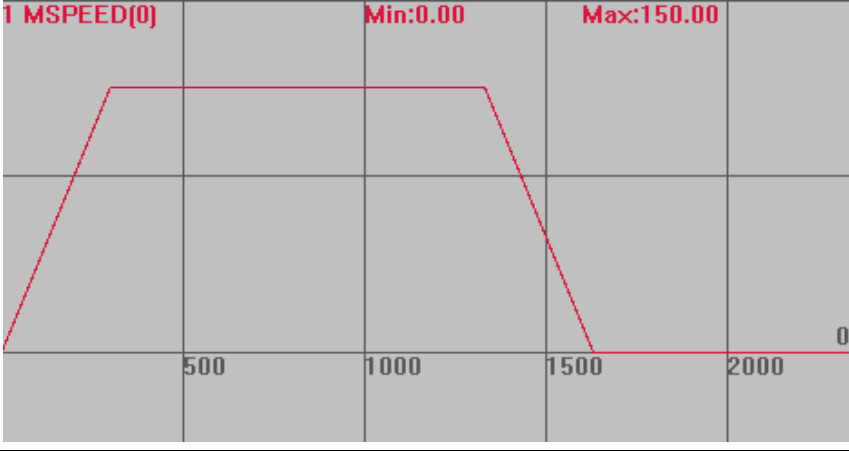
STARTMOVE_SPEED -- SP Starting Speed

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.
Syntax	To read: VAR1 = STARTMOVE_SPEED or VAR1 = STARTMOVE_SPEED (axis No.) To write: STARTMOVE_SPEED = expression or STARTMOVE_SPEED (axis No.) = expression
Controller	General
Example	<pre> 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 'first segment </pre>

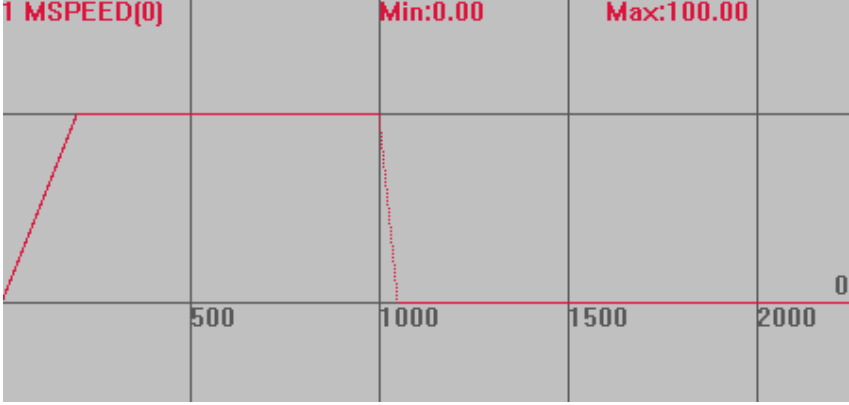
	FORCE_SPEED = 30	'the first segment speed is 30
	STARTMOVE_SPEED = 1000	'not set, default value is 1000
	ENDMOVE_SPEED = 1000	'not set, default value is 1000
	MOVESP(40)	
'second segment	FORCE_SPEED = 50	'the second segment speed is 50
	STARTMOVE_SPEED = 20	'the second segment's initial speed is 20
	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		
 1 MSPEED[0] Min:0.00 Max:60.00 2 DPOS[0] Min:0.00 Max:150.00		
Instructions	FORCE_SPEED , *SP , ENDMOVE_SPEED	

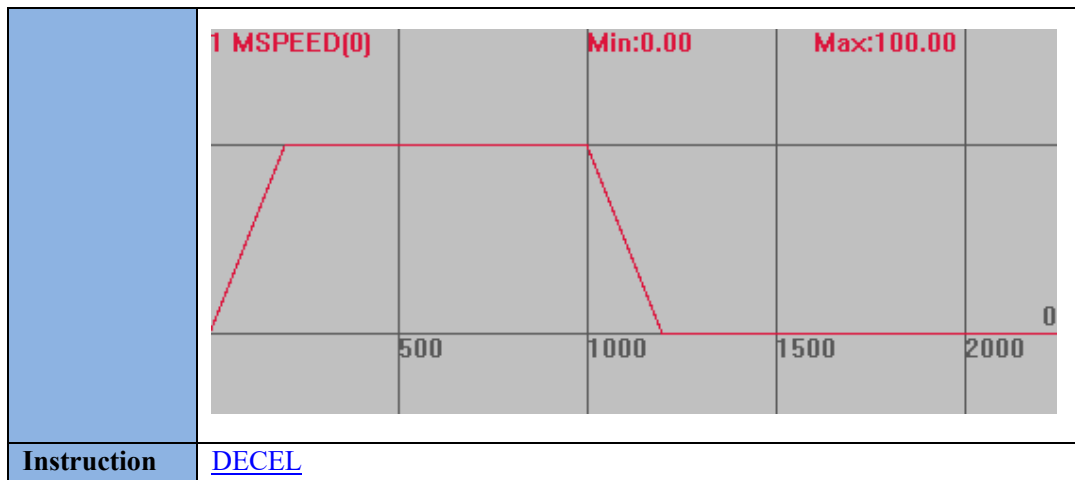
ENDMOVE_SPEED -- SP End Speed

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.
Syntax	To read: VAR1 = ENDMOVE_SPEED or VAR1 = EBDMOVE_SPEED (axis No.)

	To write: ENDMOVE_SPEED = expression or ENDMOVE_SPEED (axis No.) = 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.
Syntax	To read: VAR1 = FASTDEC or VAR1 = FASTDEC (axis No.) To write: FASTDEC = expression or FASTDEC (axis No.) = expression
Controller	General
Example	<pre> 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 </pre>  When FASTDEC=10, use DECEL to decelerate.

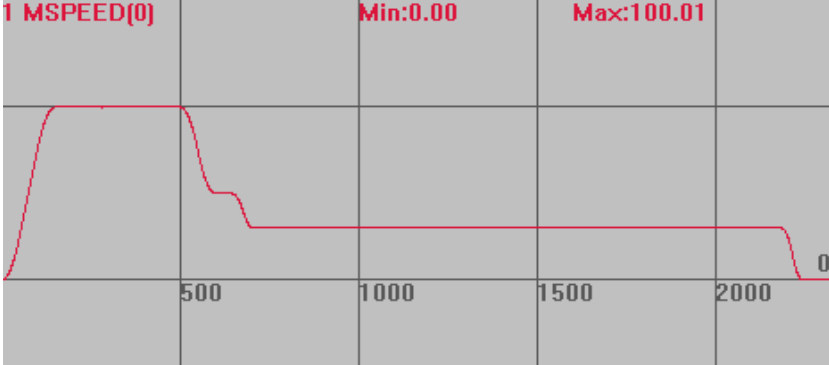


MSPEED -- Actual Speed Feedback

Type	Axis Status
Description	Measured speed feedback of axis, unit is units/s.
Syntax	VAR1 = MSPEED or VAR1 = MSPEED (axis No.)
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.
Syntax	SPEED_RATIO (axis number) = value ● value: ratio is 0-1 If not to assign axis No., it will use defined axis No. by BASE instruction by 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

	<pre> 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 </pre> Speed Curve VP_SPEED(0) vertical scale 100 
Instruction	FORCE_SPEED , SPEED

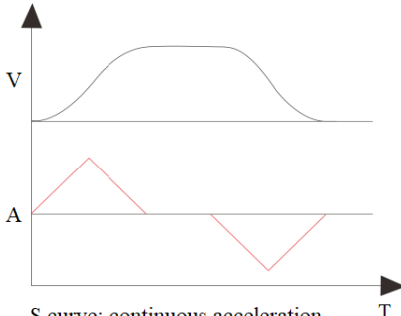
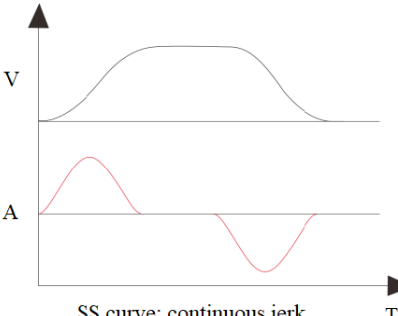
SRAMP -- Acceleration & Deceleration Curve Settings

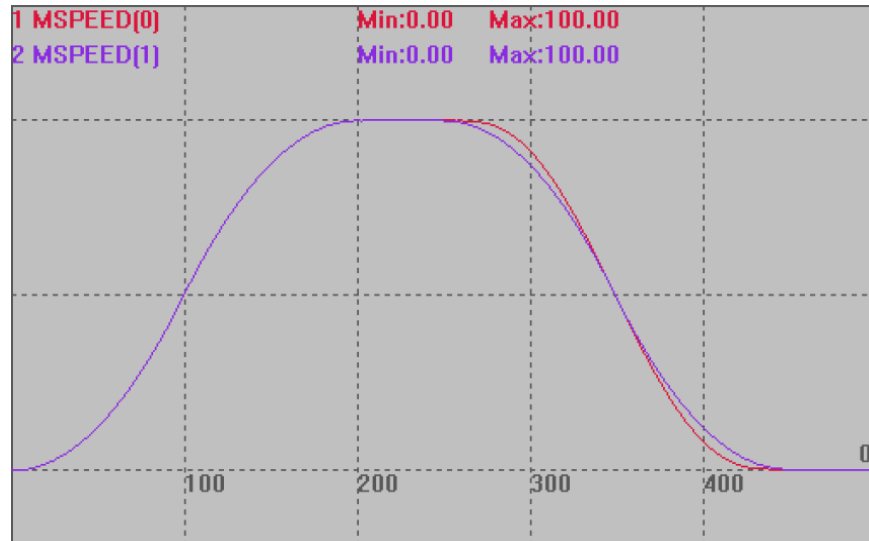
Type	Axis Parameters
Description	S curve setting of acceleration or deceleration process.
Syntax	VAR1=SRAMP, SRAMP=smoothms smoothms: 0-250ms, acceleration or deceleration process time will increase after setting.
Controller	General
Example	<pre> BASE(0) 'select axis 0 DPOS=0 UNITS=100 'pulse amount 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 </pre>

	MOVE(100) 'move 100units
	Speed Curve MSPEED(0) vertical scale 100
	
	When SRAMP=0
	
	Instruction ACCEL , DECEL

VP_MODE -- Acceleration & Deceleration Curve Types

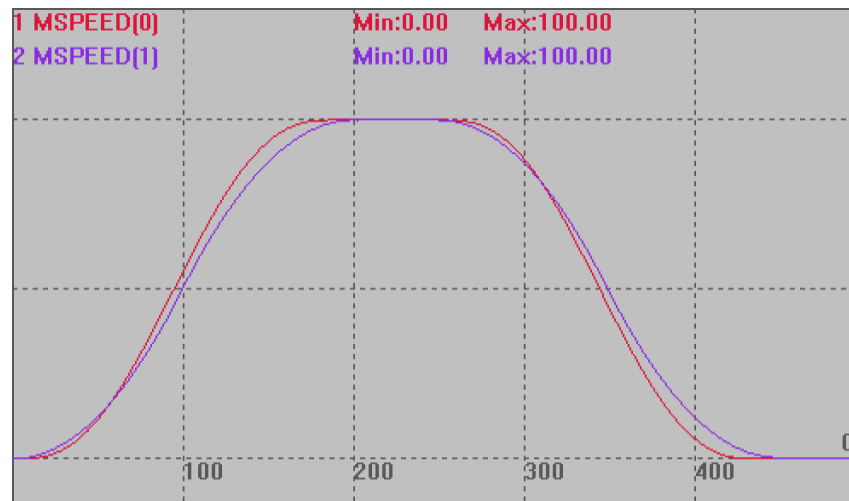
Type	Axis Parameters
Description	Acceleration and deceleration curve's type selection: 0: default value, use sramp to set S curve. 4: at the very beginning of motion, it uses the max acceleration, then acceleration will gradually decrease to 0 when achieving the highest speed. 6: new added SS curve, 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:  S curve: continuous acceleration  SS curve: continuous jerk
Syntax	VAR1=VP_MODE / VP_MODE(axis) = mode ● mode: select mode
Controller	General
Example	Example 1: mode 6 <pre> 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) </pre> 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**

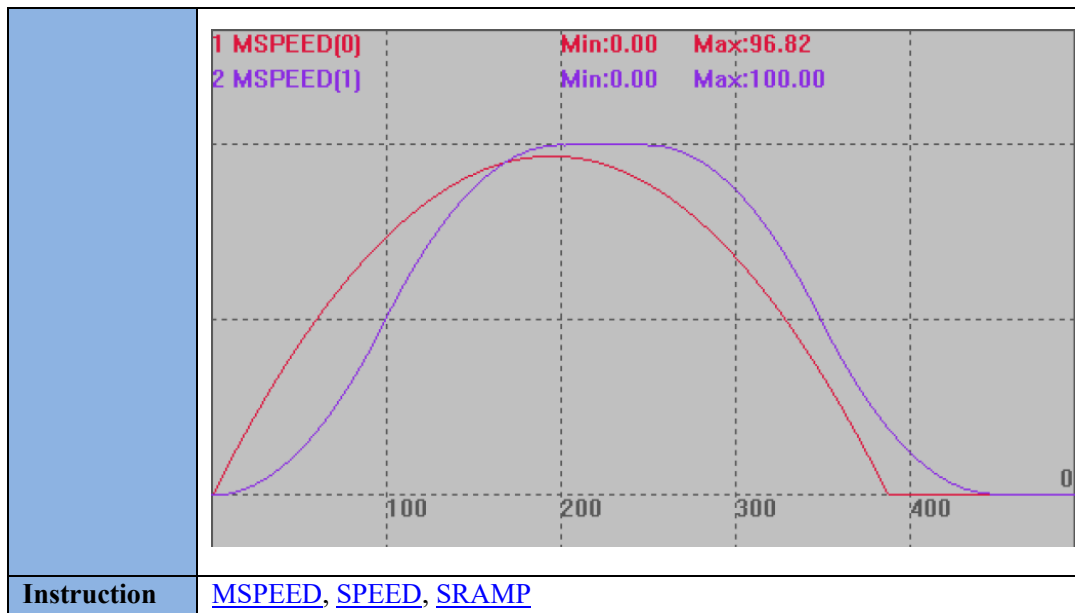
VP_MODE=7,0 'axis 0 with mode 7, axis 1 with mode 0

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.



VP_SPEED -- Current 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.
Syntax	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 Calculation

Type	Axis Parameters
Description	Axis participates speed calculation or not, default: participate (1). This parameter is only valid for any axis in linear interpolation or third axis in helical interpolation. Please restore it after moving, otherwise, it will cause wrong motion. For the slave axis whose interpolation speed is 0 doesn't participate in main axis vector calculation. 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.
Syntax	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.

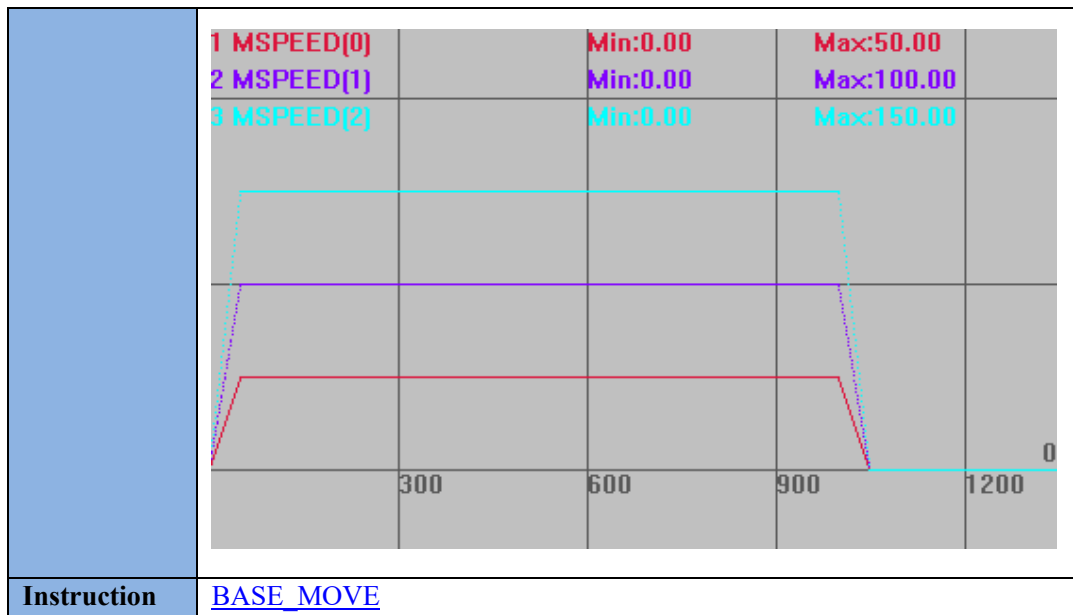


Example Three: only one axis participates speed calculation

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.



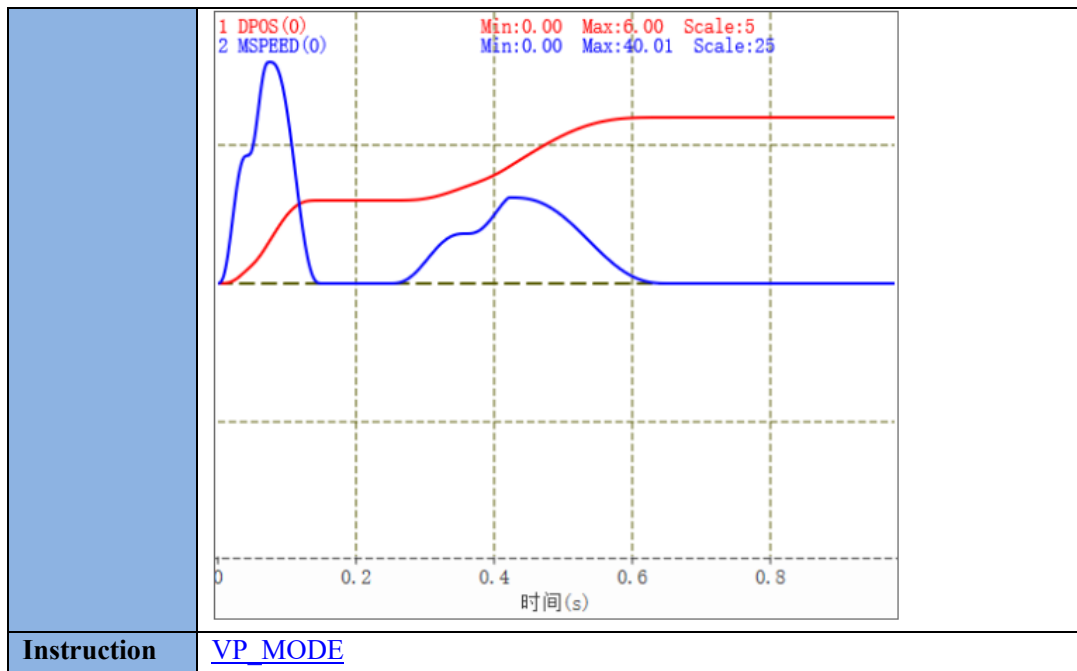
CORNER_ACCEL -- Corner Acceleration

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” (mode 64), each axis’ configured 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. It must enable small-round speed limit, then the corner acceleration can be valid.
Syntax	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

JERK -- Jerk

Type	Axis Parameter
Description	Axis jerk, the unit is units/s/s/s. The default value is 0 – not take effect. After setting the value, replace

	SRAMP. SRAMP is like “ACCEL*1000/JERK”. After JERK, under VP_MODE = 7 mode, it will automatically calculate to reduce the acceleration and deceleration, which can promote the vibration and jitter for short distance.
Syntax	To read: VAR1 = JERK / VAR1 = JERK (axis No.) To write: JERK = expression / JERK (axis No.) = expression
Controller	Valid in ZMC4XX controller; fast firmware.
Example	<pre> BASE(0,1) ATYPE = 1,1 DPOS = 0,0 MPOS = 0,0 UNITS = 1000,1000 SPEED = 100,100 ACCEL = 1000,1000 DECEL = 1000,1000 SRAMP = 100,100 MERGE = 1 VP_MODE=7 TRIGGER JERK = 0,0 MOVE(1) MOVE(1) MOVE(1) WAIT IDLE(0) JERK = 1000,1000 WA(100) MOVE(1) MOVE(1) MOVE(1) Corresponding Cruve: MSPEED (0): vertical scale 25 DPOS (0): vertical scale 5 </pre>



11.4. Axis Status Checking Instructions

MTYPE -- Current Motion Type

Type	Axis Status																						
Description	The motion command type of present motion in process. In terms of interpolation motion, slave axis always returns to master axis' motion command type.																						
Syntax	VAR1 = MTYPE <table border="1"> <thead> <tr> <th>MTYPE</th><th>Motion Type</th></tr> </thead> <tbody> <tr><td>0</td><td>IDLE (no motion)</td></tr> <tr><td>1</td><td>MOVE</td></tr> <tr><td>2</td><td>MOVEABS</td></tr> <tr><td>3</td><td>MHELICAL</td></tr> <tr><td>4</td><td>MOVECIRC</td></tr> <tr><td>5</td><td>MOVEMODIFY</td></tr> <tr><td>6</td><td>MOVESP</td></tr> <tr><td>7</td><td>MOVEABSSP</td></tr> <tr><td>8</td><td>MHELICALSP</td></tr> <tr><td>9</td><td>MOVECIRCSP</td></tr> </tbody> </table>	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
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, MOVE_PARA, MOVE_PWM, MOVE_ASYNMOVE, MOVE_AOUT
	29	MOVE_DELAY, MOVE_WAIT, MOVE_SYNMOVE
	31	MSPHERICAL, MSPHERICALSP
	32	MOVE_PT
	33	CONNFRAME
	34	CONNREFRAME
	35	MOVER_L / MOVER_LABS
	36	MOVER_C / MOVER_CABS
	37	MOVER_C3 / MOVER_C3ABS
	47	ZSMOOTH_MODE, ZSMOOTH_START, ZSMOOTH_END
	50	MOVER2_JABS
	51	MOVER2_PABS
	52	MOVER2_LABS
	53	MOVER2_ARCPABS

	54	MOVER2_ARCHPABS
	55	MOVER2_WAIT
Controller	General	
Example	<pre> WHILE 1 'loop IF MTYPE=0 THEN ?"no motion" ELSEIF MTYPE=1 THEN ?"Linear Interpolation" ELSEIF MTYPE=4 THEN ?"Circular Interpolation" ENDIF WEND </pre>	
Instruction	NTYPE , REMAIN_BUFFER	

NTYPE -- Next Motion Type

Type	Axis Status
Description	The next motion type of present motion instruction. In terms of interpolation motion, slave axis always returns to master axis.
Syntax	VAR1 = NTYPE
Controller	General
Example	<pre> 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 </pre>
Instruction	MTYPE

AXISSTATUS -- Axis Status

Type	Axis Status
Description	Check axis status. Show the value in decimal system; check the bit status as in binary system.

Syntax	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	1) Speed Protection. Axis Speed > MAX_SPEED, it will Alarm. 2) Superposed Axis Position Limit Exceeds / ALM Alarm is for Main Axis, If Spindle is Moving, It will Trigger “Stop”.	1048576	100000h
	21	Motion Buffer Error	2097152	200000h
	22	Alarm Signal Input	4194304	400000h
	23	Axis Paused	8388608	800000h
Controller	General			
Example	Example 1: 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 2: 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.			
Instruction	Example 3: 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.			
	<u>AXIS_STOPPREASON</u>			

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 returns IDLE value 0, in CONNREFRAME mode, virtual axis will returns IDLE value 0.
Syntax	VAR1 = IDLE
Controller	General
Example	Example One: <pre>IF IDLE(0) then 'if axis 0 stops BASE(1) MOVE (100) ENDIF</pre> Example Two: <pre>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</pre>
Instruction	LOADED , WAIT IDLE

ADDAX_AXIS -- Axis No. Add

Type	Axis Status
Description	Axis No. of added axis by instruction ADDAX, -1 means no axis was added.
Syntax	VAR1 = ADDAX_AXIS
Controller	General
Example	<pre>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.</pre>
Instruction	ADDAX

AXIS_STOPREASON -- Axes Stop Reason

Type	Axis Status
Description	Latch history reasons of axes stop.

Syntax	Write as 0, which means clear. Latch as per bit, same meaning as AXISSTATUS.
Controller	General; Valid in firmware above 20150731
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 to the reference axis No. of present link motion. Return -1 if there is no link.
Syntax	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 Instructions

CORNER_MODE -- Corner Speed Setting

Type	Axis Parameters	
Description	Corner deceleration mode configurations.	
Syntax	CORNER_MODE (axis No.) = mode	
	mode: different bits indicate different meanings, and bit can be used at the same time.	
	Bit	Value Description
	0	1 Reserved
	1	2 Automatic corner deceleration. ✧ ACCEL, DECEL are for acceleration and deceleration. ✧ This is used before MOVE commands. ✧ For deceleration angle, please check commands of DECEL_ANGLE and STOP_ANGLE. ✧ The corner deceleration speed refers to FORCE_SPEED, please set one reasonable speed.
	2	4 Reserved

	3	8	Auto speed limit for small arc, when radius is smaller than set value, there is speed limit, if bigger, no speed limit. ✧ This is used before MOVE commands. ✧ The speed limit radius is set by FULL_SP_RADIUS. ✧ The arc speed limit refers to FORCE_SPEED ✧ Real speed limit = $\text{FORCE\_SPEED} * \text{actual radius} / \text{FULL\_SP\_RADIUS}$ ✧ Multi-dimension also can be used.
	4	16	>stop angle, not do chamfering
	5	32	Auto chamfer setting. ✧ This is used before MOVE commands. ✧ Make “chamfering” for present MOVE motion and MOVE motion automatically, chamfer radius refers to ZSMOOTH. ✧ This chamfer is valid in all interpolation axes, the firmware should be above 20150701.
	6	64	Multi-axis interpolation separation speed, automatic corner deceleration. ✧ This is same as mode 2, the difference is the interpolation motion of mode 2 uses spindle SPEED, but for this mode, which 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 linear motion, doesn’t support circular and others.
	8	256	MOVER command uses SP mode.
	9	512	Reserved
	10	1024	Max speed limit, if the axis speed > MAX_SPEED, please reduce the speed; It only supports line and screw axis.
	11	2048	Intelligent corner deceleration + intelligent chamfer smoothing: ✧ Automatically calculate corner speed according to corner acceleration and corner accuracy; for the chamfer, it is smoothed in a new cross-segment way. ✧ BIT5 and BIT11 can be set at the same time; the small angle that doesn’t decelerate keeps using BIT5 chamfer function automatically; and for the overlay area of intelligent chamfer and BIT5 chamfer, it supports

		superposition; ✧ It can use together with ZSMOOTH_MODE smoothing; ✧ It supports “divide mode”, to increase the acceleration in different running directions; ✧ !!! it only supports: S curve of VP_MODE = 0; SS curve of VP_MODE = 7; ✧ !!! while replanning, SS curve maybe switched into S curve (make suitable SRAMP all the time). Relative Commands: ✓ REMAIN_MOVEVECT: UNITS distance within REMAIN_MOVETIME range, read-only parameter; ✓ USED_MOVETIME: executed time (MS); it can be read and written; it is floating number, please clear it before moving; ✓ CORNER_TOLERANCE: the allowable error for the corner, the unit is UNIT; it is used under BIT11, auto-calculating corner's speed and smoothing distance according to allowable errors; ✓ CORNER_ACCEL: set the corner acceleration, to calculate the corner's minimal speed reference; default is 0 (use ACCEL).
Controller	General	
Example	<i>To easily describe each bit's function, examples only show single-bit setting. That is, in real applications, multiple bits can be used together. For example, CONNER_MODE=2+8, it opens auto corner deceleration function and curve speed limit function.</i> Example 1: Corner Deceleration After opened corner deceleration, if SP motion commands (STARTMOVE_SPEED – SP starting speed / ENDMOVE_SPEED – SP end speed) are used, corner deceleration will become invalid, then the speed at the corner will move according to set SP speeds. <pre> 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 </pre>	

```

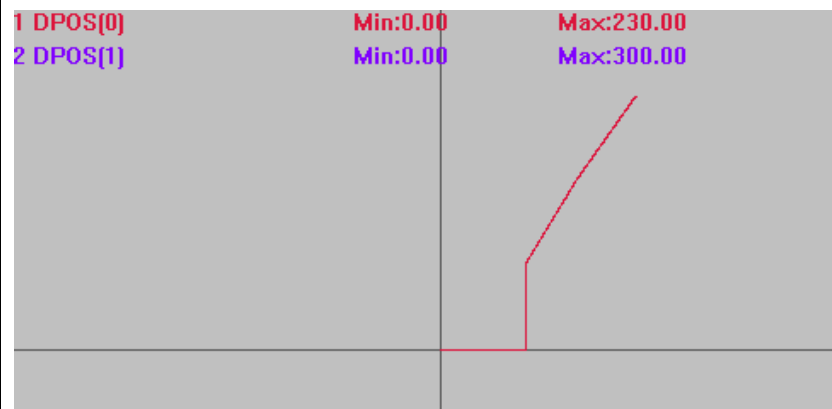
MOVE(100,0)
MOVE(0,100)           'motion angle is over 45°, full deceleration.
MOVE(60,100)
'motion angle is 30.96°, between 15°~45°, geometric deceleration
MOVE(70,100)          'motion angle is 4.03°, < 15°, no deceleration.

```

Path Curve

DPOS(0) vertical scale 200

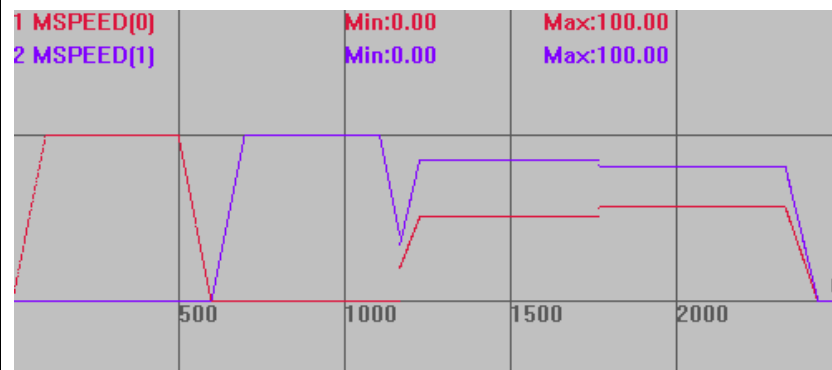
DPOS(1) vertical scale 200



Speed Curve:

MSPEED(0)=100(vertical scale)

MSPEED(1)=100(vertical scale)



Some precision errors may happen using the simulator, it is better to check with actual controllers connected.

Example 2: 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
```

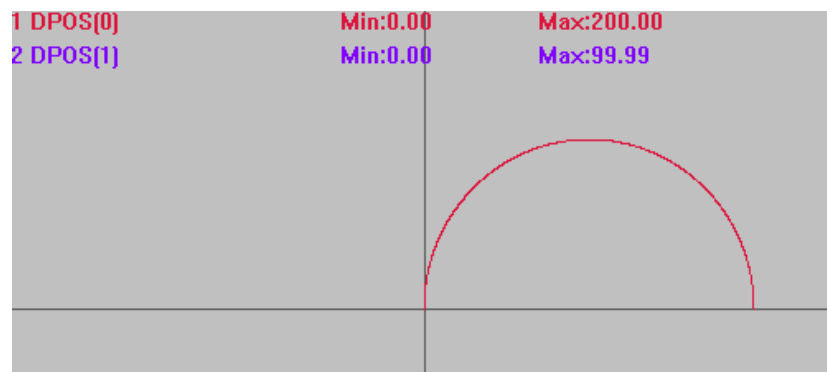
```
FORCE_SPEED=120         'limit speed of small circle
```


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.

Path Curve:

DPOS(0) vertical scale 100

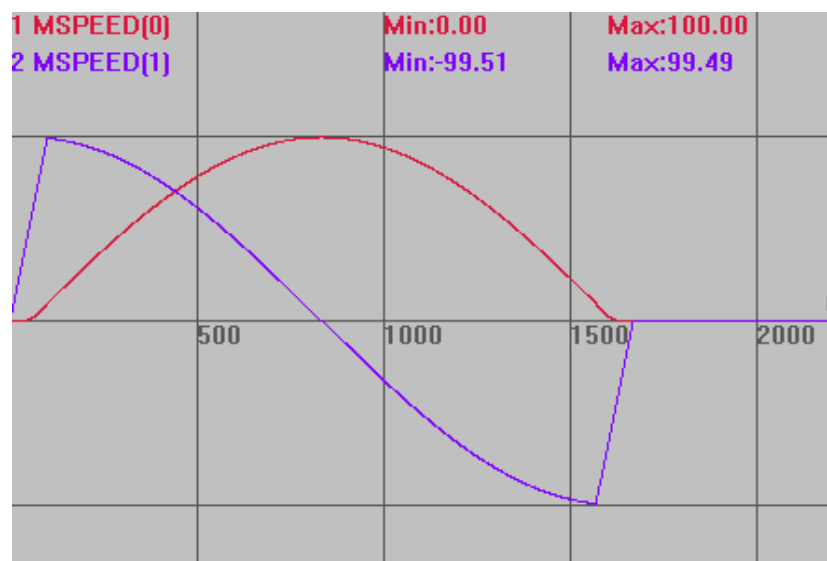
DPOS(1) vertical scale 100



Speed Curve:

MSPEED(0)=100(vertical scale)

MSPEED(1)=100(vertical scale)



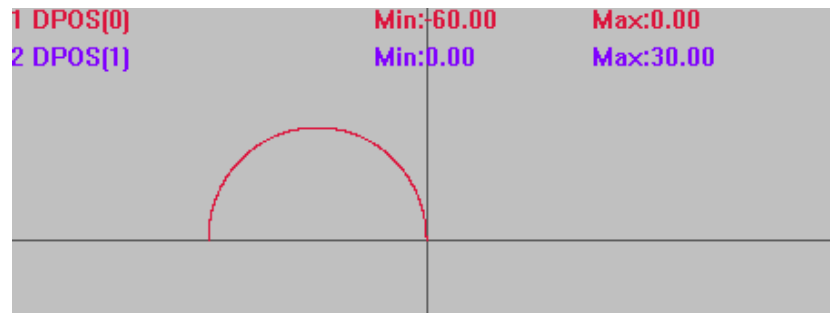
'when the radius is smaller than limit value, limit speed = FORCE_SPEED*
 actual radius / FULL_SP_RADIUS

MOVECIRC(-60,0,-30,0,0) 'now speed 60=120*30/60

Path Curve:

DPOS(0) vertical scale 100

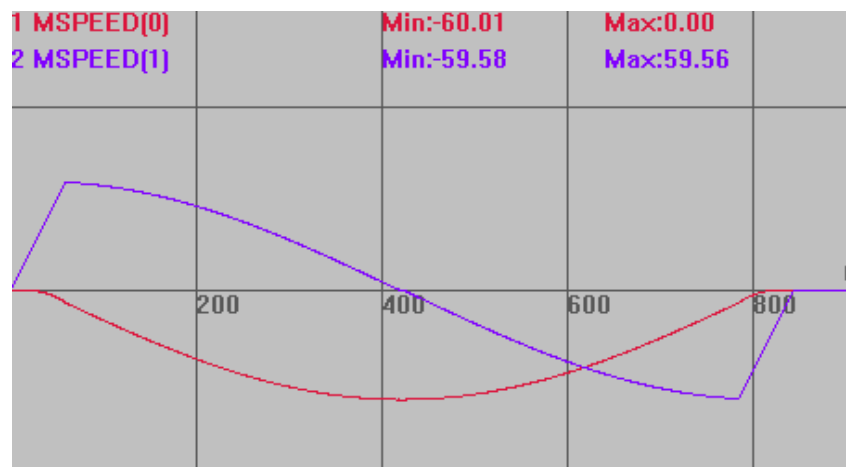
DPOS(1) vertical scale 100



Speed Curve:

MSPEED(0)=100(vertical scale)

MSPEED(1)=100(vertical scale)



Example 3: 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

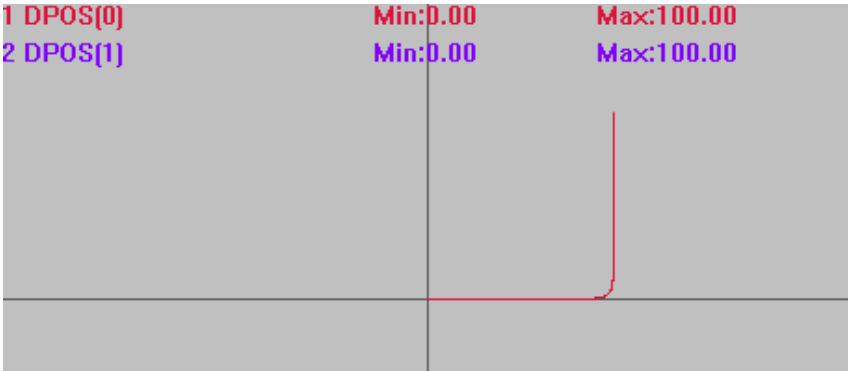
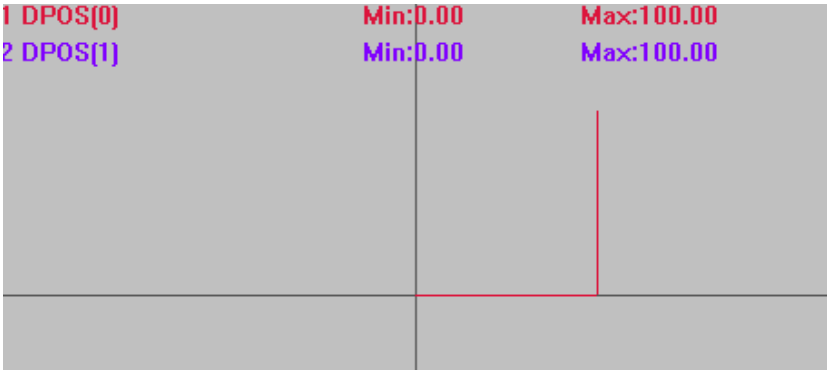
MOVE(100,0)

MOVE(0,100) 'chamfer between former two linear motions.

Interpolation Path with Chamfer.

DPOS(0) vertical scale 100

DPOS(1) vertical scale 100

	 Interpolation Path without Chamfer: DPOS(0) vertical scale 100 DPOS(1) vertical scale 100 
	Instruction MERGE , STOP_ANGLE , DECEL_ANGLE , FULL_SP_RADIUS , ZSMOOTH

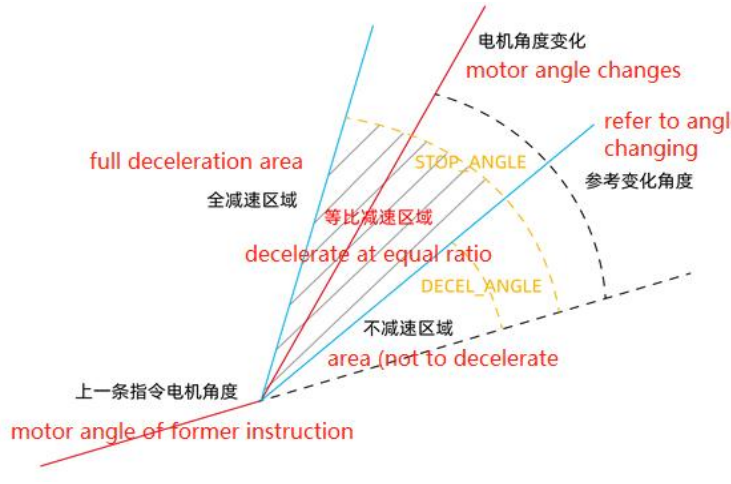
DECEL_ANGLE -- Corner Deceleration Start

Type	Axis Parameters
Description	Angle where deceleration starts, unit is radian. The corner deceleration speed refers to FORCE_SPEED, and please set proper FORCE_SPEED. Convert angle to radian: angle value * (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 the line links with the circle / arc, then the angle is calculated according to the arc tangent direction. DECEL_ANGLE is usually used with STOP_ANGLE together, when the real motion angle is between DECEL_ANGLE (upper limit) and STOP_ANGLE (lower limit), then deceleration will happen.
Syntax	VAR1 = DECEL_ANGLE, DECEL_ANGLE = expression
Controller	General
Example	Refer to CORNER_MODE example 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 , CORNER_MODE

STOP_ANGLE -- Corner Deceleration Stop

Type	Axis Parameters
Description	Angle where deceleration stops, unit is radian. The corner deceleration speed refers to FORCE_SPEED, and please set proper FORCE_SPEED. 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 the line links with the circle / arc, calculate angle according to the tangent direction of arc. Used with DECL_ANGLE together, when the real motion angle is between DECL_ANGLE (upper limit) and STOP_ANGLE (lower limit), then deceleration will happen.
Syntax	VAR1 = STOP_ANGLE, STOP_ANGLE= expression
Controller	General
Example	See example one in CORNER_MODE CORNER_MODE=2 DECL_ANGLE = 25 * (PI/180) STOP_ANGLE = 90 * (PI/180) FORCE_SPEED=SPEED 'FORCE_SPEED must be set
Instruction	DECL_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 > FULL_SP_RADIUS, motion speed will follow the value assigned by user procedure, or if < 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.
Syntax	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, using LSPEED value.
Syntax	VAR1 = SPLIMIT_RADIUS, SPLIMIT_RADIUS = expression
Controller	ZMC4XX series controller with firmware above 170518.
Instruction	CORNER_MODE , FULL_SP_RADIUS

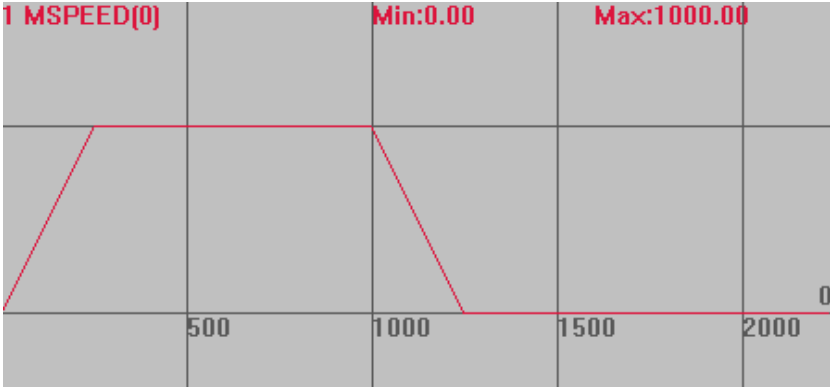
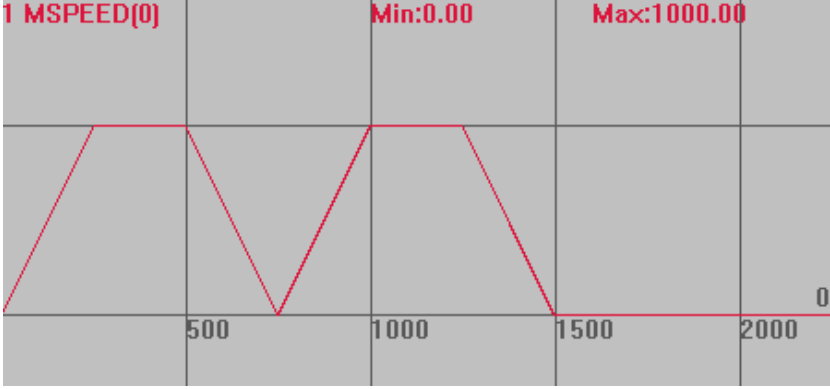
ZSMOOTH -- Chamfer Radius

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).
Syntax	VAR1 = ZSMOOTH, ZSMOOTH = smoothdistance
Controller	General
Example	See Example 3 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: ✚ MERGE is not set successfully. ✚ Controller is point-to-point model, which means it doesn't support continuous motion. Please contact with manufacturers. ✚ CORNER_MODE was set (corner deceleration). ✚ SP motion instructions are used, also ENDMOVE_SPEED and STARTMOVE_SPEED are set, then speed will use SP speeds. ✚ Main axes are switched between interpolations, and corresponding parameters are also changed. ✚ MOVE_DELAY is added between interpolations, even MOVE_DELAY is set as 0, it also will cause deceleration.
Syntax	MERGE = ON/OFF; MERGE = 1 / 0
Controller	General
Example	<pre> 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 VP_SPEED (0) vertical scale 2000 </pre>

	 When MERGE=OFF MSPEED(0) vertical scale 800 VP_SPEED (0) vertical scale 2000 
	Instruction CORNER_MODE, *SP

ZSMOOTH_MODE -- Speed Curve Smoothing Mode

Type	Axis Parameter.
Description	Smooth the speed curve by composing multiple segments as one. After smoothing, several small motion segments will become one segment, at this time, MTYPE is different from MARK. If you need to finish it, corresponding “end” command should be called, otherwise, the last small segments will be in waiting state. Notes: - The smoothing distance is up to 2 times chamfer distance. - The line segments whose angle exceeds the deceleration angle are not smooth.
Syntax	ZSMOOTH_MODE=mode Mode: 0 – not do smoothing, 1 – default mode
Controller	ZMC4XX controllers and above with firmware after 230908.
Example	ZSMOOTH_MODE=1
Instructions	ZSMOOTH_START , ZSMOOTH_END

ZSMOOTH_START -- Smoothing Mode (ON)

Type	Axis Parameter.
Description	Starting smoothing mode.
Syntax	ZSMOOTH_START([mode]) mode: set the smoothing mode, default is 1. Same as: ZSMOOTH_MODE = 1
Controller	ZMC4XX controllers and above with firmware after 230908.
Instructions	ZSMOOTH_MODE , ZSMOOTH_END

ZSMOOTH_END -- Smoothing Mode (OFF)

Type	Axis Parameter.
Description	Exit the smoothing mode.
Syntax	ZSMOOTH_END([mode]) mode: set the smoothing mode, default is 0. Same as: ZSMOOTH_MODE = 0
Controller	ZMC4XX controllers and above, and the firmware should be 230908 or later version.
Instructions	ZSMOOTH_MODE , ZSMOOTH_START

11.6. Motion Buffer Instructions

LOADED -- Empty Buffer

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.
Syntax	VAR1 = LOAED
Controller	General
Instruction	IDLE , WAIT LOADED

MOVES_BUFFERED -- Current Buffer Numbers

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.
Syntax	VAR1 = MOVES_BUFFERED
Controller	General
Example	Print MOVES_BUFFERED 'result: 0
Instruction	LOADED , LIMIT_BUFFERED , REMAIN_BUFFER

REMAIN_BUFFER -- Remain Buffers

Type	Special Axis Status
Description	Return remain motion buffers numbers. 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.
Syntax	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	<pre> 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 </pre>
Instruction	LOADED , LIMIT_BUFFERED

MOVE_MARK -- Motion Mark

Type	Axis Parameters
Description	MARK No. 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
Syntax	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 -- Motion Mark No.

Type	Axis Status
Description	Return MOVE_MARK value of present motion instruction in process.
Syntax	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 Buffers Limit

Type	System Parameters
Description	Limit motion buffer numbers, it can't exceed maximum buffer value of controller.
Syntax	LIMIT_BUFFERED=value, VAR1 = LIMIT_BUFFER
Controller	General
Example	Online print commands: <pre>>>?LIMIT_BUFFERED 4096</pre> Modify motion buffer number limit: <pre>>>LIMIT_BUFFERED=2000 >>?LIMIT_BUFFERED 2000</pre>
Instructions	REMAIN_BUFFER , MOVES_BUFFERED

11.7. Position Instructions

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.
Syntax	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
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
Syntax	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.
Syntax	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.
Syntax	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 -- Current Motion 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.
Syntax	VAR1 = ENDMOVE
Controller	General
Example	<pre> 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 </pre>
Instruction	DPOS , MPOS , ENDMOVE_BUFFER

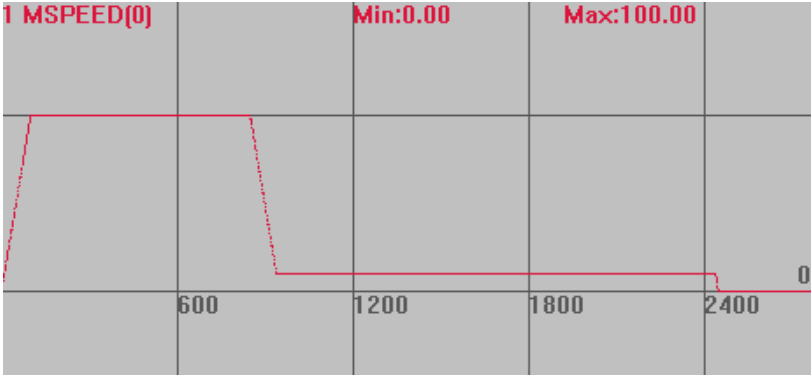
VECTOR_MOVED -- Current Motion Distance

Type	Axis Status
Description	Return motion distance of present axis, unit is units. This distance is vector distance in terms of multi 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.
Syntax	VAR1=VECTOR_MOVED, VECTOR_MOVED=0

Controller	General
Example	Example 1: single axis <pre>VECTOR_MOVED=0 'clear parameter MOVE(100) WAIT IDLE ? VECTOR_MOVED 'print motion distance of axis 0, result is 100.</pre> Example 2: multi-axis <pre>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</pre>
Instruction	ENDMOVE

REMAIN -- Current Motion Remain Distance

Type	Axis Status
Description	Return remain distance need to move, unit is units.
Syntax	VECTOR_BUFFERED
Controller	General
Example	<pre>BASE(0) 'select axis 0 DPOS=0 UNITS=100</pre>

	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.
Syntax	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 Position

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.
Syntax	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	<u>VECTOR_BUFFERED</u>

ENDMOVE_BUFFER -- End 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.
Syntax	VAR1 = ENDMOVE_BUFFER
Controller	General
Example	Example One <pre> BASE(0) SPEED = 10 DPOS = 0 MOVE(100) MOVE(200) PRINT ENDMOVE_BUFFER(0) 'print final absolute coordinate, result:300. </pre> Example Two: conversion between absolute and relative Use ENDMOVW_BUFFER and relative motion instructions together to realize absolute motion, such as MOVE, MSPHERICAL, etc. <pre> BASE(0) UNITS=100 SPEED=100 ACCEL=1000 DPOS=0 WHILE 1 MOVE(100- ENDMOVE_BUFFER(0))'move to position 100, then stop. WEND </pre> Example Three: returned value of cycle coordinate. <pre> 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 </pre>

Instruction	DPOS , MPOS , ENDMOVE

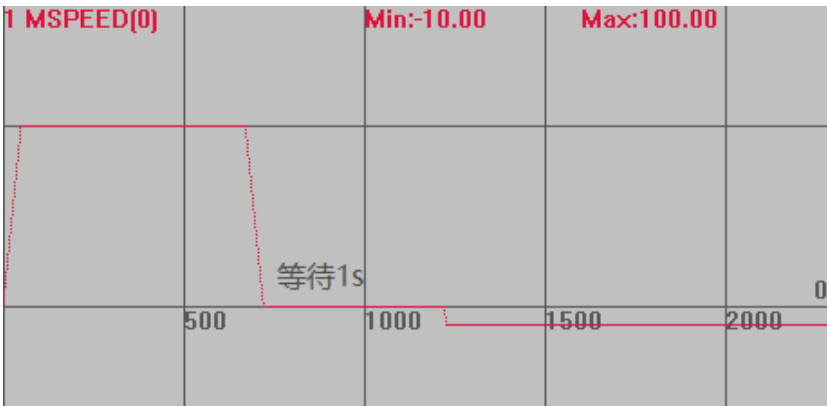
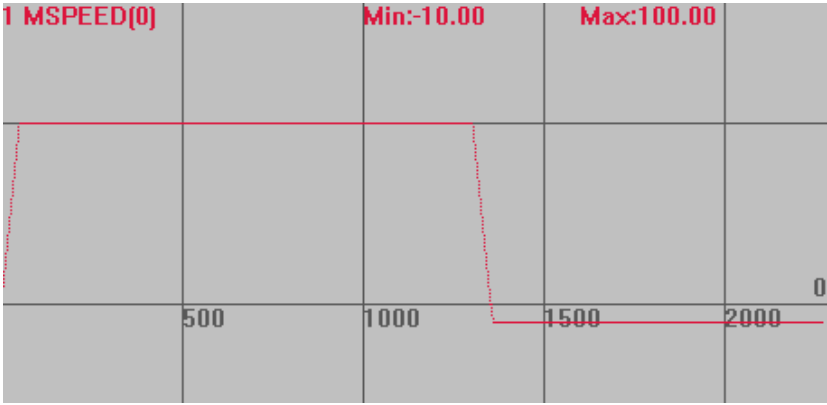
11.8. Origin Homing Instructions

DATUM_IN -- Origin Input Mapping

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)
Syntax	VAR1 = DATUM_IN, DATUM_IN = expression VAR1 = DATUM_IN (axisnum), DATUM_IN (axisnum) = 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 -- Homing Reverse-Find Delay

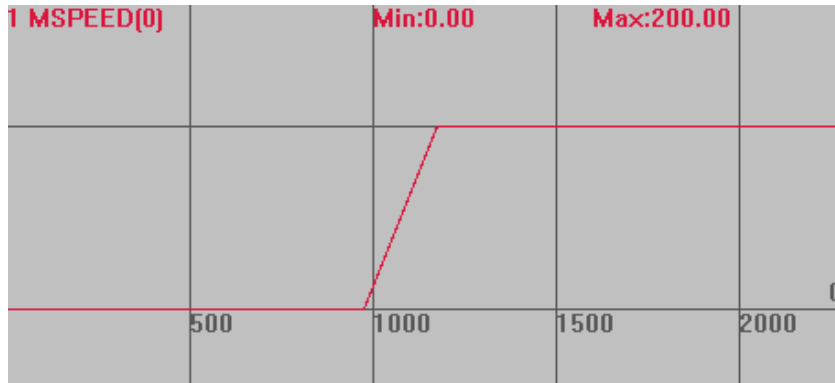
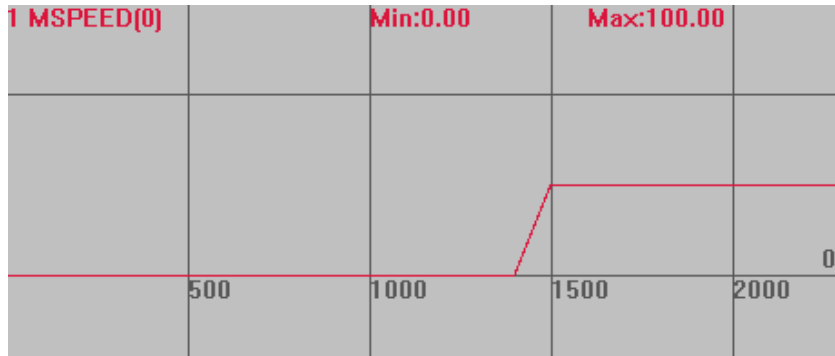
Type	Axis Parameters
Description	This parameter sets waiting time, units is millisecond. In terms of pulse servo drives, there needs time delay when back finding origin point. Default value of controller is 2ms.
Syntax	VAR1 = HOMEWAIT, HOMEWAIT = expression
Controller	General
Example	BASE(0) DPOS=0 'axis 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 Instructions

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)
Syntax	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)	
	 The graph shows the speed profile for MSPEED(0). The vertical axis represents speed, with a maximum value of 200.00 units/s. The horizontal axis represents position in units, with markers at 500, 1000, 1500, and 2000. The speed is 0 until 1000 units, then ramps linearly to 200 units/s at 1500 units, and remains constant at 200 units/s thereafter.	
	IN0:ON, IN1:ON, axis speed is SPEED=100	
	 The graph shows the speed profile for MSPEED(0) when both IN0 and IN1 are ON. The vertical axis represents speed, with a maximum value of 100.00 units/s. The horizontal axis represents position in units, with markers at 500, 1000, 1500, and 2000. The speed is 0 until 1000 units, then ramps linearly to 100 units/s at 1500 units, and remains constant at 100 units/s thereafter.	
Instruction	FWD_JOG , REV_JOG , SPEED , JOGSPEED	

FWD_JOG -- Positive JOG Input Mapping

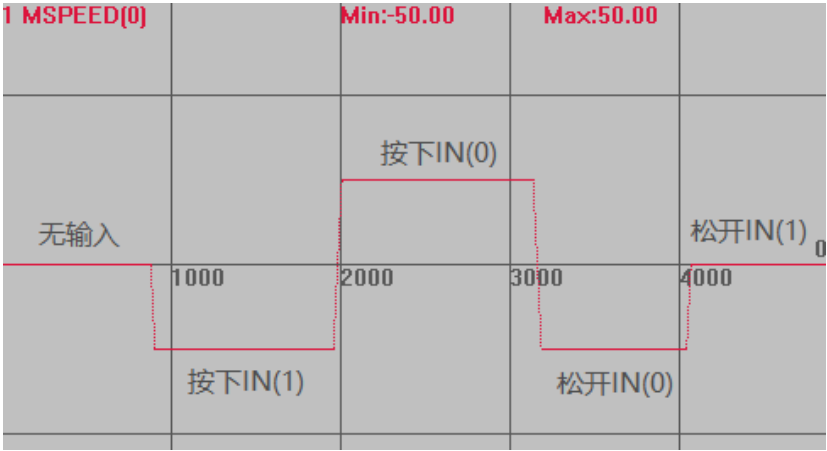
Type	Axis Parameters
Description	Input No. relates to positive JOG input, -1 means invalid. Input is valid when signal is off in ZMC controller, do use INVERT_IN to reverse the electric level. (except for ECI) When there is input signal, axis will move at speed of JOGSPEED in positive direction.
Syntax	VAR1 = FWD_JOG, FWD_JOG= expression VAR1 = FWD_JOG (axisnum), FWD_JOG (axisnum) = expression
Controller	General
Example	Example One <pre> 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 </pre> 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 1 MSPEED(0) Min:0.00 Max:100.00 2 IN[0] Min:0.00 Max:1.00 1000 1500 2000 2500
Instruction	REV_JOG , FAST_JOG , JOGSPEED

REV_JOG -- Negative JOG Input Mapping

Type	Axis Parameters
Description	Input No. 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
Syntax	VAR1 = REV_JOG, REV_JOG= expression
Controller	General
Example	See in Example FWD_JOG.
Instruction	FWD_JOG , FAST_JOG , JOGSPEED

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.
Syntax	JOGSPEED= value, VAR1=JOGSPEED
Controller	General
Example	Example One: <pre> 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 </pre>

	direction. Speed Curve: MSPPED(0) vertical scale 100 
Instruction	FWD_JOG , REV_JOG , FAST_JOG

FHOLD_IN -- Hold Input Mapping

Type	Axis Parameters
Description	Hold related input No., -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.
Syntax	VAR1 = FHOLD_IN, FHOLD_IN = expression VAR1 = FHOLD_IN (axisnum), FHOLD_IN (axisnum) = expression
Controller	General
Example	<pre> 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 </pre>

	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.
Syntax	VAR1 = FHSPEED, FHSPEED = expression
Controller	General
Example	See example in FHOLD_IN
Instruction	FHOLD_IN , SPEED

11.10. “Encoder” Instructions

ENCODER -- Encoder Original Value

Type	Axis Status
Description	Set / get the encoder (hardware register) original value. Inner parameters are only valid after setting correct ATYPE. If the drive encoder is multiturn, multiturn value is read.

	It only can be used for debugging (don't use in script codes, and different controllers support different bits, for example, ZMC432-V2 uses 32-bit, ZMC460N uses 16-bit).
Syntax	VAR1 = ENCODER (axis No.)
Controller	General
Example	?*ENCODER 'print each axis' encoder value, result:0 0 0 0 0 0 0 0 0 0
Instruction	MPOS , ENCODER_RATIO

ENCODER_STATUS -- Encoder Status

Type	Axis Status		
Description	Set / get encoder states: EA / EB / EZ.		
Syntax	VAR1 = ENCODER_STATUS		
	Bit	Value	Description
	0	1	EA Status
	1	2	EB Status
	2	4	EZ Status
Controller	General		
Example	?*ENCODER_STATUS 'print out all axes' encoder states		
Instruction	ATYPE , MPOS		

ENCODER_FILTER -- Encoder Filter

Type	Axis parameters
Description	Set inner encoder filter value, which is used to make blet encoder speed “uniform”, default value is 1, the range values = 1/cycles, ranging from 0.001~1. ● Default 1: no filter; ● 0.5: the filter of 2 periods; ● 0.25: the filter of 4 periods;
Syntax	ENCODER_FILTER = VALUE
Controller	Valid in VPLC5XX & ZMC4XX series above, the firmware should be above 170706.
Instruction	ENCODER_RATIO

PP_STEP -- Encoder Internal Proportion

Type	Axis Parameters
Description	When encoder value is input, it will multiply this parameter. Default value is 1.
Syntax	PP_STEP = VALUE

Controller	General
Instruction	ENCODER_RATIO

ENCODER_BITS -- Encoder Absolute Value

Type	Axis Parameters			
Description	Set SSI/BISS encoder absolute value.			
Syntax	ENCODER_BITS = VALUE			
	Encoder	Bit	Value	Function Description
	SSI/BISS	Bit0-5	0-32	The total bit of encoder communication: data bit + auxiliary state bit
		Bit6	64*(0≤n≤1)	Whether it is Gray code
		Bit8-10	256*(0≤n≤15)	Invalid bit, BISS = 8 (usually)
		Bit16-18	65536*(0≤n≤7)	Frequency division adjustment, default 0-2MHz; The calculation formular of frequency & n: Fre = 2M (n+1)
ATYPE = 48 'SSI absolute encoder				
ATYPE = 49 'BISS absolute encoder				
Before using this commands, axis mapping must be done.				
Controller	ZMC432-V2 & ZMC406-V2 controllers with Ssi firmware.			
Example	SSI Example:			
	BASE(n)			
	AXIS_ADDRESS = (-1<<16) + 4		'map the 4th physical axis position	
	ENCODER_BITS = 26		'26-bit absolute value	
	ATYPE = 48			
	BISS Example:			
BASE(n)				
AXIS_ADDRESS = (-1<<16) + 5		'map the 5th physical axis position		
ENCODER_BITS = 26				
'26-bit absolute value, it is with 8 state bits automatically				
ATYPE = 49				

ENCODER_ID -- Data Dictionary Storage No.

Type	Axis Parameters

Description	Set the data source of customized encoder axis.
Syntax	ENCODER_ID=(index<<16)+(subindex<<8)+bites bites: the bits of data type, such as, data is 32-bit, then write 32.
Controller	General
Example	Define one encoder axis: BASE(0) AXIS_ADDRESS = (0<<16) + nodenum ENCODER_ID= \$60640020 ATYPE=25

DRIVE_POSMIN -- Encoder Original Min Value

Type	Axis Parameters
Description	Set the minimal value of original value range transferred by encoder.
Syntax	DRIVE_POSMIN = VALUE Set it before modifying ATYPE. Make MPOS do coordinates loop, please 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_POSMAX -- Encoder Original Max Value

Type	Axis Parameters
Description	Set the maximal value of original value range transferred by encoder.
Syntax	DRIVE_POSMAX = VALUE Set before modifying ATYPE. Make MPOS do coordinates loop, please 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. Latch Instructions

REGIST -- Position Latch

Type	Position Latch Instruction						
Description	REGIST is used to latch the axis measured position. It can latch encoder axis and bus axis. Different controller models can latch different axis types. ZMC4XX controllers or above & H series controllers with the latest firmware support latching virtual axis & pulse axis. ➤ For EtherCAT controllers, they support drive latching, that is, use drive IO to achieve latching (configure DRIVE_PROFILE command), for specific modes, please check command details. ➤ For RTEX controllers, it only supports controller latching. 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. EtherCAT drive latching & controller latching can be used at the same time, but it should support 4-channel latching. For bus drive probe latching, please use MARK and MARKB channels. For controller local IN latching, please use MARKC, MARKD channels. For 6XX series and above, 8 latching channels are supported: MARK, MARKB, MARKC, MARKD, MARKE, MARKF, MARKG, MARKH. 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.						
Syntax	Syntax 1: REGIST(mode) ● mode: latch mode Rising or falling edge is determined by the inner status of controller (if sets as rising edge, latch will be triggered when external input port becomes OFF. Instead, if sets as falling edge, latch will be triggered when external input port becomes ON). Pulse axis type uses R0, R1 and Z latching generally; bus axis local IN uses R2, R3. <table border="1"> <thead> <tr> <th>value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Save absolute position in REG_POS when meets rising edge of Z pulse</td></tr> <tr> <td>2</td><td>Save absolute position in REG_POS when meets falling edge of</td></tr> </tbody> </table>	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
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.
	Mode 25- mode 32 are for 6XX and above controllers because they support 8-channel for latching.	
	25	Save absolute position in REG_POSE when meets rising edge of R4 signal.
	26	Save absolute position in REG_POSE when meets falling edge of R4 signal.
	27	Save absolute position in REG_POSF when meets rising edge of R5 signal.
	28	Save absolute position in REG_POSF when meets falling edge of R5 signal.
	29	Save absolute position in REG_POSG when meets rising edge of R6 signal.
	30	Save absolute position in REG_POSG when meets falling edge of R6 signal.
	31	Save absolute position in REG_POSH when meets rising edge of R7 signal.
	32	Save absolute position in REG_POSH when meets falling edge of R7 signal.

Syntax 2: REGIST(100+mode, tableindexn, numes)

- mode: latch mode
- tableindex: table position of saving continuously latched content. The first table element saves latched numbers, behind 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

2 channels are to do continuous latching separately, which can realize rising and falling edge of continuous latch.

ECI: with firmware version above 20150829.

4XX 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
	Mode 25- mode 32 are for 6XX and above controllers because they support 8-channel for latching.	
	25	Save absolute position in REG_POSE when meets rising edge of R4 signal.
	26	Save absolute position in REG_POSE when meets falling edge of R4 signal.
	27	Save absolute position in REG_POSF when meets rising edge of R5 signal.
	28	Save absolute position in REG_POSF when meets falling edge of R5 signal.
	29	Save absolute position in REG_POSG when meets rising edge of R6 signal.
	30	Save absolute position in REG_POSG when meets falling edge of R6 signal.
	31	Save absolute position in REG_POSH when meets rising edge of R7 signal.
	32	Save absolute position in REG_POSH when meets falling edge of R7 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 rising edge of R0 signal, and print <pre> 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 </pre> Example2: latch position of encoder axis 0 when meets rising edge of R1 signal, and print <pre> 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 </pre> Example3: latch position of EtherCAT bus axis 0 when meets rising edge of R2 signal, and print 'before latch, please do bus initialization enable, use R2R3 latch. <pre> 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 ELSEIF 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 </pre> 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 configure 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 UNTIL MARK    'probe 1
    ?"mode", imode, "latch position REG_POS", REG_POS
    DELAY(100)
ELSEIF imode = 14 OR imode = 15 THEN

```

	<pre> WAIT UNTIL MARKB 'wait until latch to be triggered ?"mode", imode, "latch position REG_POS", REG_POS DELAY(100) ELSEIF 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 ELSEIF MARKB THEN ?"mode", Imode, "latch position REG_POSB", REG_POSB WAIT UNTIL MARK ?"latch position REG_POSB", REG_POS ENDIF DELAY(100) ENDIF </pre>
Instructions	MARK , MARKB , REG_POS , REG_POSB

REG_INPUTS -- Latch Input Mapping

Type	Axis Parameter		
Description	Configure inputs of position latch signals: R0-R3, every 4 bits was mapped to one latch signal input.		
Syntax	VAR1 = REG_INPUTS		
	Bit	Latch signal	Input range (ZMC306E)
	Bit0-3	R0	0-3
	Bit4-7	R1	0-3
	Bit8-11	R2	0-3
	Bit12-15	R3	0-3
Inputs Range: different controller modes have different inputs ranges.			
Controller	Some ZMC3XX series and ZMC4XX series with latest firmware version support latch function.		
Example	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.		
Instruction	REGIST		

REGIST_DELAY -- Latch Delay

Type	Axis Parameter
Description	When using the local IN to latch the coordinates of the bus driver, the actual position of the bus driver may be delayed. That is, when the

	servo cycle parameter value is different from the default value, REGIST_DELAY must be adjusted.
Syntax	regist_delay=timems alias: regist_wa=timems timems: delay, the unit is “ms”
Controller	The controller has IN port that supports Latch.
Example	regist_delay(-1.01) ‘when SERVO_PERIOD is 1ms, it will compensate one servo period, and sensor input will delay “10us”. regist_delay(-0.515) ‘when SERVO_PERIOD is 0.5ms, it will compensate one servo period, and sensor input will delay “15us”.
Instruction	DELAY

MARK -- Latch Trigger

Type	Axis Status
Description	Check if 1st channel latched through returned value. When REGIST is executing, MARK is true, returned value is -1, once REGIST execution is finished, MARK becomes false, returned value is 0.
Syntax	VAR1 = MARK
Controller	General
Example	BASE(0) 'select axis 0 MOVE(100) 'move 100units REGIST(3) 'rising edge trigger, get signal R0. WAIT UNTIL MARK 'wait to be triggered.
Instruction	REG_POS , REGIST

MARKB -- Latch 2 Trigger

Type	Axis Status
Description	Check if 2nd channel latched through returned value. When REGIST is executing, MARKB is true, returned value is -1, once REGIST execution is finished, MARKB becomes false, returned value is 0.
Syntax	VAR1 = MARKB
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	Check if 3rd channel latched through returned value. When REGIST is executing, MARKC is true, returned value is -1, once REGIST execution is finished, MARKC becomes false, returned value is 0.
Syntax	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	Check if 4th channel latched through returned value. When REGIST is executing, MARKD is true, returned value is -1, once REGIST execution is finished, MARKD becomes false, returned value is 0.
Syntax	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

MARKE -- Latch 5 Trigger

Type	Axis Status
Description	Check if 5th channel latched through returned value. When REGIST is executing, MARKE is true, returned value is -1, once REGIST execution is finished, MARKE becomes false, returned value is 0.
Syntax	VAR1 = MARKE
Controller	General
Example	BASE(0) 'select axis 0 MOVE(100) 'move 100units REGIST(25) 'rising edge trigger, get signal R4 WAIT UNTIL MARKE 'wait to be triggered.
Instruction	REG_POSE , REGIST

MARKF -- Latch 6 Trigger

Type	Axis Status
Description	Check if 6th channel latched through returned value. When REGIST is executing, MARKF is true, returned value is -1, once REGIST execution is finished, MARKF becomes false, returned value is 0.
Syntax	VAR1 = MARKF
Controller	General
Example	BASE(0) 'select axis 0 MOVE(100) 'move 100units REGIST(27) 'rising edge trigger, get signal R5 WAIT UNTIL MARKF 'wait to be triggered.
Instruction	REG_POSE , REGIST

MARKG -- Latch 7 Trigger

Type	Axis Status
Description	Check if 7th channel latched through returned value. When REGIST is executing, MARKG is true, returned value is -1, once REGIST execution is finished, MARKG becomes false, returned value is 0.
Syntax	VAR1 = MARKG
Controller	General
Example	BASE(0) 'select axis 0 MOVE(100) 'move 100units REGIST(29) 'rising edge trigger, get signal R6 WAIT UNTIL MARKG 'wait to be triggered.
Instruction	REG_POSG , REGIST

MARKH -- Latch 8 Trigger

Type	Axis Status
Description	Check if 8th channel latched through returned value. When REGIST is executing, MARKH is true, returned value is -1, once REGIST execution is finished, MARKH becomes false, returned value is 0.
Syntax	VAR1 = MARKH
Controller	General
Example	BASE(0) 'select axis 0 MOVE(100) 'move 100units REGIST(31) 'rising edge trigger, get signal R7 WAIT UNTIL MARKH 'wait to be triggered.
Instruction	REG_POSH , REGIST

OPEN_WIN -- Latch Coordinate Range (ON)

Type	Axis Parameters
Description	The starting coordinates points when latch starts. Reserved
Syntax	OPEN_WIN = pos
Instruction	REGIST , CLOSE_WIN

CLOSE_WIN -- Latch Coordinate Range (OFF)

Type	Axis Parameters
Description	The end coordinates points when latch ends. Reserved
Syntax	CLOSE_WIN = pos
Instruction	REGIST , OPEN_WIN

REG_POS -- Position Latch

Type	Axis Status	
Description	Save the latched measured feedback position (MPOS) in the specified units. The latched position represents the encoder feedback position; it must be read and saved promptly, as REG_POS becomes invalid once the encoder coordinates continue to change. The latched position is subject to a specific range limit; if the encoder rotates beyond this range, REG_POS will increment or decrement by the corresponding range value based on the direction of motion. Depending on the controller model, this range is either 2^{16} or 2^{32} .	
Syntax	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 -- Position Latch 2

Type	Axis Status
Description	Save the measured feedback position (MPOS) of latching channel 2 in the specified units.

	The latched position represents the encoder feedback position; it must be read and saved promptly, as REG_POSB becomes invalid once the encoder coordinates continue to change. The latched position is subject to a specific range limit; if the encoder rotates beyond this range, REG_POSB will increment or decrement by the corresponding range value based on the direction of motion. Depending on the controller model, this range is either 2^{16} or 2^{32} .
Syntax	VAR1 = REG_POSB (axis No.)
Controller	General
Example	<pre> 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 </pre>
Instruction	REGIST , MARKB

REG_POSC -- Position Latch 3

Type	Axis Status
Description	Save the measured feedback position (MPOS) of latching channel 3 in the specified units. The latched position represents the encoder feedback position; it must be read and saved promptly, as REG_POSC becomes invalid once the encoder coordinates continue to change. The latched position is subject to a specific range limit; if the encoder rotates beyond this range, REG_POSC will increment or decrement by the corresponding range value based on the direction of motion. Depending on the controller model, this range is either 2^{16} or 2^{32}.
Syntax	VAR1 = REG_POSC (axis No.)
Controller	General
Example	<pre> 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 </pre>
Instruction	REGIST , MARKC

REG_POSD -- Position Latch 4

Type	Axis Status
Description	Save the measured feedback position (MPOS) of latching channel 4 in the specified units. The latched position represents the encoder feedback position; it must be read and saved promptly, as REG_POSD becomes invalid once the encoder coordinates continue to change. The latched position is subject to a specific

	range limit; if the encoder rotates beyond this range, REG_POSD will increment or decrement by the corresponding range value based on the direction of motion. Depending on the controller model, this range is either 2^{16} or 2^{32} .
Syntax	VAR1 = REG_POSD (axis No.)
Controller	General
Example	<pre> 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 </pre>
Instruction	REGIST , MARKD

REG_POSE -- Position Latch 5

Type	Axis Status
Description	Save the measured feedback position (MPOS) of latching channel 5 in the specified units. The latched position represents the encoder feedback position; it must be read and saved promptly, as REG_POSE becomes invalid once the encoder coordinates continue to change. The latched position is subject to a specific range limit; if the encoder rotates beyond this range, REG_POSE will increment or decrement by the corresponding range value based on the direction of motion. Depending on the controller model, this range is either 2^{16} or 2^{32}.
Syntax	VAR1 = REG_POSE (axis No.)
Controller	General
Example	<pre> MOVE(100) 'move 100units REGIST(20) 'trigger when meets rising edge signal WAIT UNTIL MARKE 'wait until the second latch happens. PRINT REG_POSE 'print latched position </pre>
Instruction	REGIST , MARKE

REG_POSF -- Position Latch 6

Type	Axis Status
Description	Save the measured feedback position (MPOS) of latching channel 6 in the specified units. The latched position represents the encoder feedback position; it must be read and saved promptly, as REG_POSF becomes invalid once the encoder coordinates continue to change. The latched position is subject to a specific range limit; if the encoder rotates beyond this range, REG_POSF will increment or decrement by the corresponding range value based on the direction of motion. Depending on the controller model, this range is either

	2^{16} or 2^{32} .	
Syntax	VAR1 = REG_POSF (axis No.)	
Controller	General	
Example	MOVE(100)	'move 100units
	REGIST(20)	'trigger when meets rising edge signal
	WAIT UNTIL MARKF	'wait until the second latch happens.
	PRINT REG_POSF	'print latched position
Instruction	REGIST , MARKF	

REG_POSG -- Position Latch 7

Type	Axis Status	
Description	Save the measured feedback position (MPOS) of latching channel 7 in the specified units. The latched position represents the encoder feedback position; it must be read and saved promptly, as REG_POSG becomes invalid once the encoder coordinates continue to change. The latched position is subject to a specific range limit; if the encoder rotates beyond this range, REG_POSG will increment or decrement by the corresponding range value based on the direction of motion. Depending on the controller model, this range is either 2^{16} or 2^{32}.	
Syntax	VAR1 = REG_POSG (axis No.)	
Controller	General	
Example	MOVE(100)	'move 100units
	REGIST(20)	'trigger when meets rising edge signal
	WAIT UNTIL MARKG	'wait until the second latch happens.
	PRINT REG_POSG	'print latched position
Instruction	REGIST , MARKG	

REG_POSH -- Position Latch 8

Type	Axis Status	
Description	Save the measured feedback position (MPOS) of latching channel 8 in the specified units. The latched position represents the encoder feedback position; it must be read and saved promptly, as REG_POSH becomes invalid once the encoder coordinates continue to change. The latched position is subject to a specific range limit; if the encoder rotates beyond this range, REG_POSH will increment or decrement by the corresponding range value based on the direction of motion. Depending on the controller model, this range is either 2^{16} or 2^{32}.	
Syntax	VAR1 = REG_POSH (axis No.)	
Controller	General	

Example	MOVE(100)	'move 100units
	REGIST(20)	'trigger when meets rising edge signal
	WAIT UNTIL MARKH	'wait until the second latch happens.
	PRINT REG_POSH	'print latched position
Instruction	REGIST , MARKH	

11.12. Position Limit Parameter Instructions

FS_LIMIT -- Soft Positive Position 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, don't 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't be as homing signal reference when using DATUM.																	
Syntax	VAR1 =FS_LIMIT, FS_LIMIT = expression VAR1 =FS_LIMIT(axisnum), FS_LIMIT = expression (axisnum)																	
Controller	General																	
Example	<table><tr><td>BASE(0)</td><td>'select axis 0</td></tr><tr><td>ATYPE=1</td><td>'axis type setting</td></tr><tr><td>UNITS=100</td><td>'pulse amount is 100</td></tr><tr><td>DPOS=0</td><td>'coordinate clears</td></tr><tr><td>SPEED=100</td><td>'speed is 100units/s</td></tr><tr><td>ACCEL=1000</td><td>'acceleration is 1000 units/s/s</td></tr><tr><td>FS_LIMIT=200</td><td>'set positive limit as 200units</td></tr><tr><td>MOVE(300)</td><td>'move 300units</td></tr></table> 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.		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
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																	
Instruction	RS_LIMIT , FWD_IN , REV_IN																	

RS_LIMIT -- Soft Negative Position 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, don't 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't be as homing signal reference when using DATUM.
Syntax	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 Input Mapping

Type	Axis Parameters
Description	Input No. related to positive position limit input (hard), -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)
Syntax	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	FS_LIMIT , REV_IN , FASTDEC

REV_IN -- Negative Limit Input Mapping

Type	Axis Parameters
Description	Input No. relates to negative position limit input (hard), -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)
Syntax	VAR1 = REV_IN, REV_IN = expression VAR1 =REV_IN(axisnum), REV_IN = expression (axisnum)
Controller	General
Example	See Example in FWD_IN
Instruction	RS_LIMIT , FWD_IN , FASTDEC

ALM_IN -- Alarm Input

Type	Axis Parameters
Description	Drive alarm input settings, -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.
Syntax	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.
Syntax	IN_POS = expression IN_POS (axisnum) = expression
Controller	General, valid in 4xx series controllers with the latest firmware version.
Example	<pre> 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 </pre>
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.
Syntax	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	<pre> BASE(0) 'select axis 0 ATYPE=1 UNITS=100 'set pulse amount as 100 DPOS=0 'clear coordinates as 0 </pre>

	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	<u>IN_POS_DIST</u> , <u>IN_POS_SPEED</u> , <u>IN_POS</u> , <u>INVERT_IN</u> ,

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.
Syntax	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

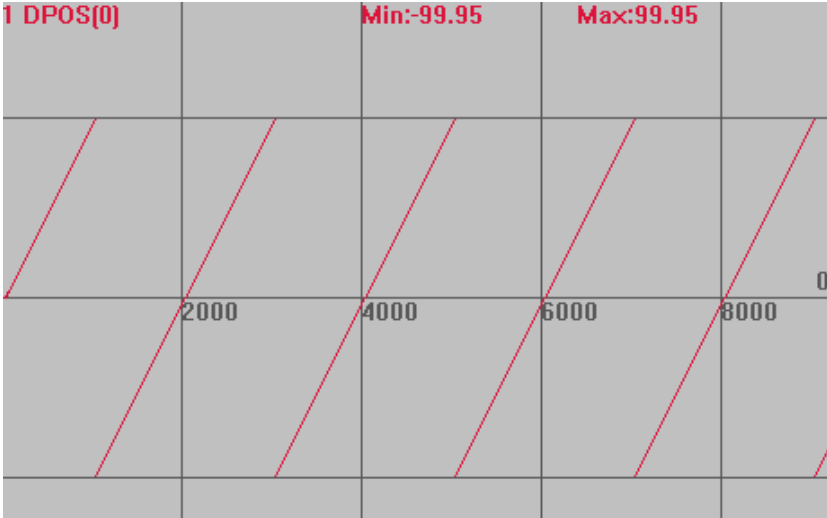
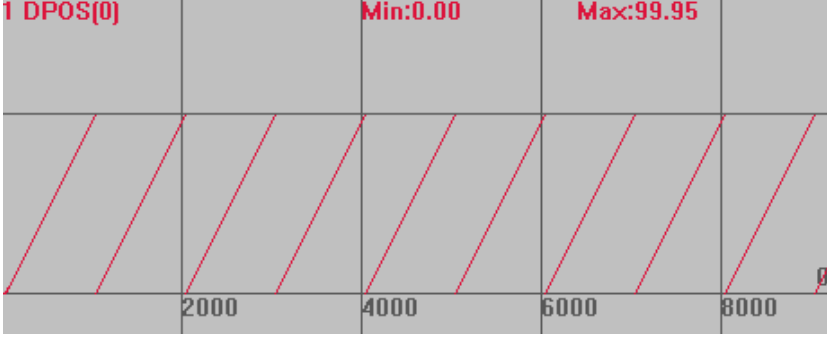
Type	Axis Parameters
Description	Configure on-position speed, MSPEED is less than this speed, and FE is less than IN_POS_DIST, it means it arrives the position. Parameters start from 0, when this parameter and on-position parameter are used together, on-position mark is controlled by on-position signal.
Syntax	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	<pre> 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 </pre>
Instruction	IN_POS_DIST , IN_POS , AXISINP_IN

11.14. Range Limit Parameter Instructions

REP_OPTION -- Coordinate Cycle Mode

Type	Axis Parameters

Description	Coordinate Repeat Set It can limit the coordinate cycle range of the cam spindle to realize continuous operation of multiple cam curves. ➤ Under absolute mode, if the target position is within coordinate cycle range, run correctly; otherwise, run abnormally. ➤ No influence on relative motion. ➤ If REP_DIST > max. valid value, it will reset REP_OPTION Bit4 as 1.																
Syntax	VAR1 = REP_OPTION, REP_OPTION = opt opt, different bits indicate different meanings. <table border="1"> <thead> <tr> <th>Bit</th><th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>1</td><td>0: ranges from - REP_DIST to + REP_DIST. 1: ranges from 0 to + REP_DIST.</td></tr> <tr> <td>1</td><td>2</td><td>1: the repeated motion between CAMBOX and MOVELIK is forbidden, 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> </tbody> </table>		Bit	Value	Description	0	1	0: ranges from - REP_DIST to + REP_DIST. 1: ranges from 0 to + REP_DIST.	1	2	1: the repeated motion between CAMBOX and MOVELIK is forbidden, once it activates, value recovers to 0.	2	4	reserved	4	16	1: don't use REP_DIST. 0: use REP_DIST.
Bit	Value	Description															
0	1	0: ranges from - REP_DIST to + REP_DIST. 1: ranges from 0 to + REP_DIST.															
1	2	1: the repeated motion between CAMBOX and MOVELIK is forbidden, once it activates, value recovers to 0.															
2	4	reserved															
4	16	1: don't use REP_DIST. 0: use REP_DIST.															
Controller	General																
Example	BASE(0) 'select axis 0 ATYPE=1 UNITS=100 'pulse amount is 100 DPOS=0 'clear coordinates as 0 SPEED=100 'speed is 100units/s ACCEL=1000 'acceleration is 1000units/s/s DECEL=1000 'deceleration is 1000units/s/s REP_DIST=360 'set coordinate cycle range REP_OPTION=0 'set cycle mode TRIGGER 'trigger oscilloscope automatically VMOVE(1) 'continuous motion Coordinate Curves: DPOS(0): vertical scale 200; horizontal scale (x scale) 3s.																

	
	When REP_OPTION=1, DPOS(0): vertical scale 100; horizontal scale (x scale) 3s.
	
Instruction	CAMBOX , MOVELINK , REP_DIST

REP_DIST -- Coordinate Cycle Position

Type	Axis Parameters
Description	Automatically cycle axis DPOS and MPOS coordinates according to REP_OPTION.
Syntax	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, this value = DPOS - MPOS.
Syntax	To write: VAR1 = FE (axis No.)

	To read: WAIT UNTIL ABS(FE(axis No.)) <= var Description: FE in WAIT UNTIL ABS(FE(axis No.)) <= var can be positive or negative, so it uses ABS absolute value. Generally, it is used after WAIT IDLE (axis No.).
Controller	General
Instruction	MPOS , DPOS

FE_RANGE -- Follow-up Error when Alarm

Type	Axis Parameters
Description	Follow-up error when alarm happens.
Syntax	VAR1 = FE_RANGE, FE_RANGE = expression Note: use FE_RANGE AXIS (axis) to set assigned axis.
Example	Refer to SERVO .
Instruction	FE , SERVO , P_GAIN , D_GAIN , I_GAIN , OV_GAIN , VFF_GAIN , FE_LIMIT

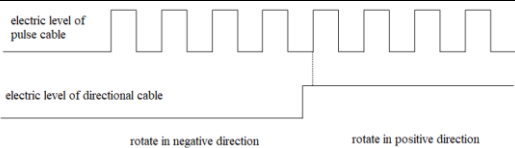
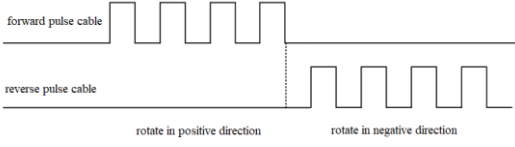
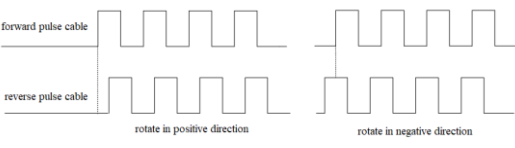
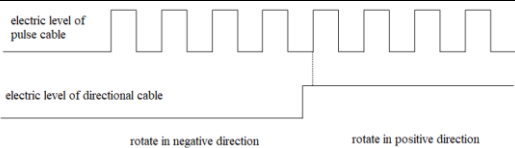
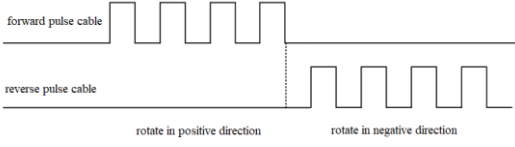
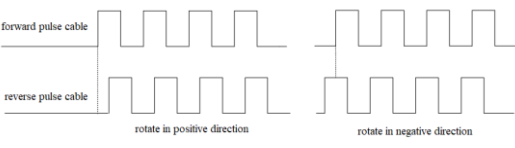
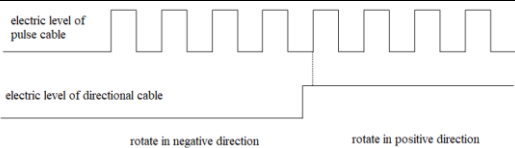
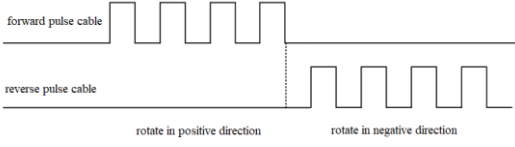
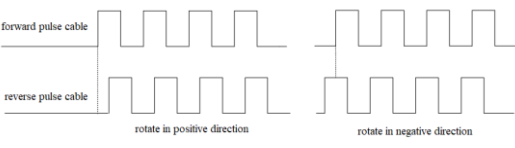
FE_LIMIT -- Maximum Follow-Up Error

Type	Axis Parameters
Description	Allowed maximum follow-up error, default: 3 When follow error exceeds, it will produce real-time error, and the enable relay (WDOG) will be cleared, which means it will prevent other generators from running. This limit usually is used in state of “protection”, such as, when the machine is locked, encoder feedback is lost, etc. It will report alarms when timeout, 2h/100h/102h are reported usually for AXISSTATUS.
Syntax	VAR1 = FE_LIMIT, FE_LIMIT= expression Note: use FE_LIMIT AXIS (axis) to set assigned axis.
Example	Refer to SERVO .
Instruction	FE , SERVO , P_GAIN , D_GAIN , I_GAIN , OV_GAIN , VFF_GAIN , FE_RANGE

11.15. Advanced Setting Instructions

INVERT_STEP -- Pulse Mode Settings

Type	Axis Parameters
Description	Servo/Step pulse output mode setting.

	There are 3 modes: pulse directional, double pulse, 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.																																																		
Syntax	INVERT_STEP = mode ● 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. <table><tr><th rowspan="2">Mode value</th><th rowspan="2">Description</th><th colspan="2">Panasonic setting</th><th>Mitsubishi setting</th></tr><tr><th>Pr0.06</th><th>Pr0.07</th><th>PA13</th></tr><tr><td>0</td><td>Pulse/direction(pulse positive logic) (positive)</td><td>0</td><td>3</td><td>××01h</td></tr><tr><td>1</td><td>Pulse/direction(pulse negative logic) (positive)</td><td>/</td><td>/</td><td>××11h</td></tr><tr><td>2</td><td>Pulse/direction(pulse positive logic) (negative)</td><td>1</td><td>3</td><td>××01h</td></tr><tr><td>3</td><td>Pulse/direction(pulse negative logic) (negative)</td><td>/</td><td>/</td><td>××11h</td></tr><tr><td>4</td><td>Double pulse (direction negative logic) (positive)</td><td>/</td><td>/</td><td>××10h</td></tr><tr><td>5</td><td>Double pulse (direction negative logic) (negative)</td><td>/</td><td>/</td><td>××10h</td></tr></table>	Mode value	Description	Reference (positive logical mode0)	0-3	pulse direction. Pulse line + direction line.		4-7	double pulse (or CW/CCW), positive pulse line + negative pulse line.		8-9	AB output, quadrature pulse (some controllers are customized)		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
Mode value	Description	Reference (positive logical mode0)																																																	
0-3	pulse direction. Pulse line + direction line.																																																		
4-7	double pulse (or CW/CCW), positive pulse line + negative pulse line.																																																		
8-9	AB output, quadrature pulse (some controllers are customized)																																																		
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.
Syntax	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 -- Precision OP

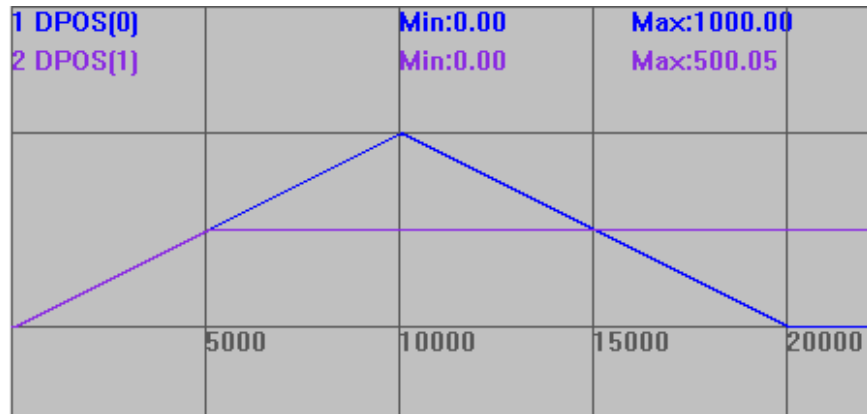
Type	Axis Parameters
Description	To set precision output function of MOVE_OP, which is used for the main axis of axes group. When SYSTEM_ZSET is modified, AXIS_ZSET of present BASE axes will also be modified. In order to fit old procedure, usually it is not recommended to use SYSTEM_ZSET instruction. 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 bit 6: set OP on the superposed axis, after enabled, and after position limit / alarm of superposed slave axis is triggered, the main axis also will stop, and main axis AXISSTATUS BIT20 will be triggered at the same time. Set whether to enable this function. 1 – ON, 0 – OFF.
Syntax	To read: VALUE=AXIS_ZSET To write: AXIS_ZSET=VALUE
Controller	Firmware version above 20170517
Example	Example 1: Open Precision Output <pre> 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 </pre> 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. <pre> 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) </pre>

	<pre> 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. </pre>
Instruction	SYSTEM_ZSET , MOVE_OP

AXIS_MODE -- “connect” Motion Hold

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 DPOS that is to track main axis. Involved instructions: CAMBOX, CONNECT, MOVELINK, MOVESLINK, MOVESYNC, HW_PSWITCH2. BIT8 = 0 (default), = 1, and when CLUTCH_RATE = 0, the tracking respond is faster; when the speed exceeds the target speed, and when the acceleration exceeds 2 times of target acceleration, the tracking lag will be small. Valid in ZMC4XX series whose firmware version version_build: 240226. Note: when the hard position limit is triggered while moving, and when the position limit direction is consistent with motion direction, it will lead the synchronization breaks. Valid in controllers with firmware version 20170616 and above.
Syntax	VAR1 = AXIS_MODE, AXIS_MODE = expression
Example	Example 1: not set AXIS_MODE parameters <pre> 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 </pre>



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

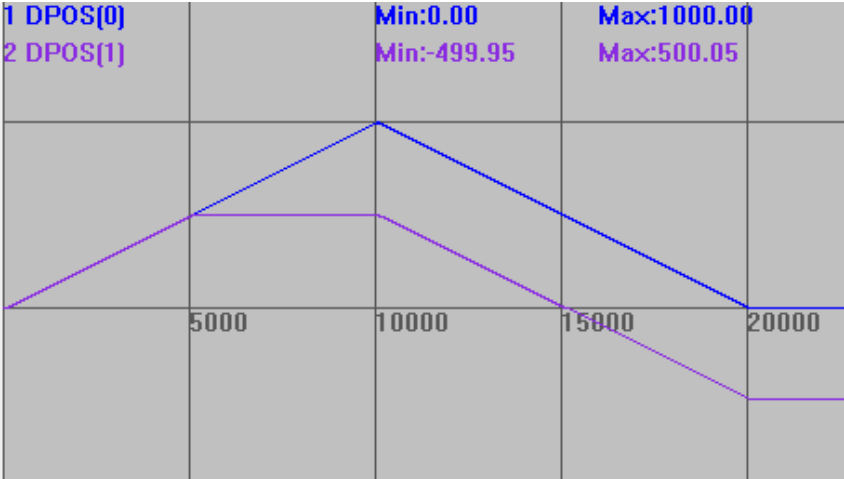
MOVE(1000) AXIS(0)

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 the 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. Please note this function (minus value) is only valid when the real servo drive is used, because “real servo drive” supports lag, then axis lag could do “output in advance”. For verification, recommend to use “latch”; the command will be influenced by axis FE, you could set ATYPE as 0 / 1 to test the delay function (no axis FE affect); same, the “in advance” function is achieved by servo lag, then, please use real drive with encoder feedback to verify (bus axis is also OK). MOVEOP_DELAY will not be cleared manually, if it is not cleared, it will work with MOVEOP_DIST and MOVEOP_AHEADMS together.
Syntax	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 Glue (in Advance)

Type	Axis parameters
Description	Output one certain distance in advance: ➤ 0: default value, no any effect. ➤ +: an assigned vector distance will be output in advance. ➤ -: an assigned vector distance will be delayed. <i>“no need to open precision output, it still can take effect”</i> Notes: ● This is only valid when one single OP is operated by MOVE_OP, and 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 be cleared automatically after MOVEOP_ADIST entering buffer to avoid affecting other MOVE_OP. ● Controllers that support HW functions can open HW precision output through AXIS_ZSET (same effect as MOVE_OP), but it only supports MOVE interpolation motions and MOVESCAN motions, it doesn't support cam, MOVE_PT, etc. And cross-small segments output in advance of MOVE can be achieved.
Syntax	MOVEOP_ADIST= distance ● distance: how far for “early / delay to closes the glue”
Controller	4xx series controllers with fast firmware version above 190201.
Example	<pre> 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) </pre>

	DPOS(0) 50 (vertical scale) OP(0) 5 (vertical scale)
Instruction	MOVE_OP , MOVEOP_DELAY , AXIS_ZEST

MOVEOP_AHEADMS -- Time of Early Closing Glue

Type	Axis parameters
Description	The time of early closing the glue (ms). Default 0 = invalid. Positive value = early, negative value = delay. Notes: ➤ Not recommend to use for delay to close, because it will automatically clear after MOVE_OP, the same as MOVEOP_ADIST. If you want it to delay, only average motion is valid, recommend use MOVEOP_DELAY. ➤ The effect is better when the output in constant motion / in deceleration, otherwise, it may be not accurate. ➤ Better not make the early time > deceleration time. ➤ Don't set it and MOVEOP_ADIST at the same time.
Syntax	MOVEOP_AHEADMS = timesms ● timesms: time, the unit is ms
Controller	4xx series controllers with fast firmware version above 190201.
Example	<pre> BASE(0) ATYPE=1 UNITS=100 DPOS=0 MPOS=0 SPEED=100 ACCEL=1000 DECEL=1000 MERGE=1 AXIS_ZSET(0) = 19 SRAMP=100 DELAY(500) TRIGGER MOVE(50) </pre>

	<pre> MOVEOP_AHEADMS = -10 MOVE_OP(0,1) MOVE(50) MOVE(50) MOVEOP_AHEADMS = 10 'the unit is ms MOVE_OP(0,0) MOVE_DELAY(1000) MOVE(-50) MOVEOP_AHEADMS = -10 MOVE_OP(0,1) MOVE(-50) MOVE(-50) MOVEOP_AHEADMS = 10 'the unit is ms MOVE_OP(0,0) MOVE_DELAY(1000) Motion Curve: MSPEED(0) 100 (vertical scale) DPOS(0) 100 (vertical scale) OP(0) 2 (vertical scale) </pre>
Instruction	MOVE_OP , MOVEOP_DELAY , AXIS_ZEST , MOVEOP_ADIST

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.
Syntax	VAR1 = DAC, DAC = expression

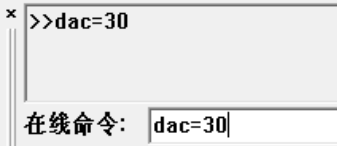
Controller	With EtherCAT port or Rtex port, firmware above 2017 support.																																	
Example	Example 1: Rtex speed control																																	
	Please use Rtex initialization procedure at first, and set ATYPE=51. Then do RTSys online instruction, as follow:																																	
	× >>dac=10 在线命令: dac=10																																	
	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)																																	
	<table><tr><td>Pr7.25</td><td>unit</td></tr><tr><td>0</td><td>[r/min]</td></tr><tr><td>1</td><td>[instruction unit/s]</td></tr></table>	Pr7.25	unit	0	[r/min]	1	[instruction unit/s]																											
	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																																		
× >>drive_read(7*256+25) 0 在线命令: drive_read(7*256+25)																																		
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>val</td><td>SL_SW</td></tr><tr><td>ue</td><td>0 1</td></tr><tr><td>0</td><td>Pr3.21</td></tr></table></td><td>Set speed limit when turn to torque control. In torque</td><td>WhenPr3.17(speed limit selection)=1, set SL SW speed</td></tr></table>	Type	3	3	3	No.	17	21	22	Property	B	B	B	Parameter name	speed limit selection	Speed limit value is 1	Speed limit value is 2	Set range	0-1	0-20000	0-20000	Unit	/	r/min	r/min	Function	<table><tr><td>val</td><td>SL_SW</td></tr><tr><td>ue</td><td>0 1</td></tr><tr><td>0</td><td>Pr3.21</td></tr></table>	val	SL_SW	ue	0 1	0	Pr3.21	Set speed limit when turn to torque control. In torque	WhenPr3.17(speed limit selection)=1, set SL SW speed
Type	3	3	3																															
No.	17	21	22																															
Property	B	B	B																															
Parameter name	speed limit selection	Speed limit value is 1	Speed limit value is 2																															
Set range	0-1	0-20000	0-20000																															
Unit	/	r/min	r/min																															
Function	<table><tr><td>val</td><td>SL_SW</td></tr><tr><td>ue</td><td>0 1</td></tr><tr><td>0</td><td>Pr3.21</td></tr></table>	val	SL_SW	ue	0 1	0	Pr3.21	Set speed limit when turn to torque control. In torque	WhenPr3.17(speed limit selection)=1, set SL SW speed																									
val	SL_SW																																	
ue	0 1																																	
0	Pr3.21																																	

		1 Pr3 .21 Pr3 .22	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 level),Pr9.10(maximum over speed level)	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 level), Pr9.10(maximum over speed level).

Set speed limit value selection mode if turn to torque control.

Then use [Rtex initialization procedure](#), and set ATYPE=52.

Next, can send dac online instruction in RTSys, 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)

```

	<pre> 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 </pre> 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 .
Syntax	VAR1 = ERRORMASK, ERRORMASK = expression
Controller	General
Example	<pre> 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. </pre>
Instruction	AXISSTATUS , WDOG

ZSCAN_CORRECT -- Galvanometer Correction

Type	Axis parameters
Description	Correct galvanometer axis parameters.
Syntax	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 coordinate start to save in the 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 protocol coordinate is from 32768 to 32767)
Controller	Valid in controllers with galvanometer axis
Example	<pre>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)</pre>

11.16. PID 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.
Syntax	VAR1=D_GAIN, VAR1=D_GAIN Valid: set assigned axis through D_GAIN AXIS (axis) method
Example	Refer to SERVO
Instruction	SERVO , P_GAIN , I_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.
Syntax	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.
Syntax	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 , I_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.

Syntax	VAR1=P_GAIN, P_GAIN = expression Valid: set assigned axis through P_GAIN AXIS (axis) method
Example	Refer to SERVO
Instruction	SERVO , OV_GAIN , D_GAIN , I_GAIN , VFF_GAIN , FE_LIMIT , FE_RANGE

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.
Syntax	VAR1=VFF_GAIN, VFF_GAIN = expression Valid: set assigned axis through VFF_GAIN AXIS (axis) method
Example	Refer to SERVO
Instruction	SERVO , OV_GAIN , D_GAIN , I_GAIN , VFF_GAIN , FE_LIMIT , FE_RANGE , AFF_GAIN , P_GAIN

AFF_GAIN -- Acceleration Feedforward Gain

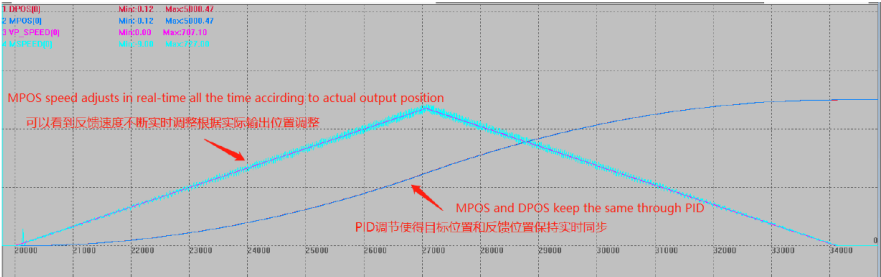
Type	Axis Parameters
Description	Feedforward gain feedbacked by acceleration, don't support non-bus servo. 60B2 pdo = (pulse speed change of axis in each period) * AFF_GAIN
Syntax	VAR1=AFF_GAIN, AFF_GAIN = expression Valid: use AFF_GAIN AXIS(axis) to set assigned axis.
Example	Refer to SERVO
Instruction	SERVO , OV_GAIN , D_GAIN , I_GAIN , VFF_GAIN , FE_LIMIT , FE_RANGE , P_GAIN

SERVO -- Closed-Loop Switch

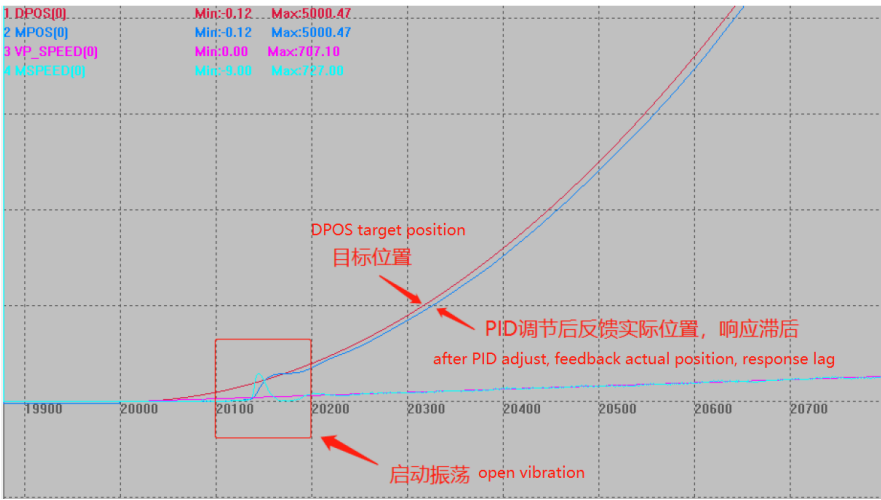
Type	Axis Parameters
Description	Close loop switch setting Value range: ON/OFF, default is OFF.

	It is used as closed-loop control system together with P_GAIN, D_GAIN, I_GAIN, OV_GAIN, VFF_GAIN commands, attention analog servo needs to be used. Attentions: 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.
Syntax	VAR1 = SERVO, SERVO = ON/OFF
Controller	General
Example	<pre> 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 </pre>

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 [OV_GAIN](#), [D_GAIN](#), [I_GAIN](#), [VFF_GAIN](#), [FE_LIMIT](#), [FE_RANGE](#) , [P_GAIN](#)

Chapter 12 IO Instructions

12.1. Input Instructions

IN -- Digital 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.
Syntax	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_AIO instruction, the value can only be a multiple of 8. Please check user manual for details. Attention: AIO mapped number should be more than AIO max No. of controller itself, and can't superpose with controller number.
Syntax	Var=AIN(channel) channel: analog input channels:0-127
Controller	General

Example	$a = \text{AIN}(1)$ 'read AD value of channel 1. $a = \text{AIN}(1) * 10 / 4096$ 'voltage value of channel 1. $a = ((\text{AIN}(1) - 32767) * 10 / 65535) * 2$ 'the voltage value of channel 1 when the range is between -10V to 10V.
Instructions	AOUT

ZSIMU_IN -- Digital Input Simulation

Type	Simulator specialized instructions.
Description	Simulate input of IN.
Syntax	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 Input Simulation

Type	Simulator specialized instructions
Description	Simulate analog input of IN.
Syntax	ZSIMU_AIN(ionum, value)
Controller	General
Example	ZSIMU_AIN(0,1024) 'analog input 0
Instructions	AIN

ZSIMU_ENCODER -- Encoder Input Simulation

Type	Simulator specialized instructions
Description	Simulate input of encoder.
Syntax	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 -- Input Invert

Type	Special instructions
Description	Reverse inputs status, it can be checked if inputs were reversed.
Syntax	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 -- Inputs Changes Scan

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.
Syntax	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	<pre> 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) ELSEIF IN_EVENT(0) < 0 THEN 'trigger falling edge of IN0 PRINT "IN0 DOWN", IN_BUFF(0) ENDIF ENDIF WEND </pre>
Instructions	IN_EVENT , SCAN_EVENT , IN_BUFF

IN_EVENT -- Input Changes Read

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.
Syntax	VAR1 = IN_EVENT(IONUM)
Controller	General
Example	See IN_SCAN
Instructions	IN_SCAN , SCAN_EVENT

SCAN_EVENT -- Changes Check

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.
Syntax	ret = SCAN_EVENT (expression) ● expression: any valid expression, result will become BOOL Type.
Controller	General
Example	Example One: Scan inputs signals <pre> WHILE 1 IF SCAN_EVENT(IN(0))>0 THEN 'trigger rising edge of IN0 PRINT "IN0 ON" ELSEIF SCAN_EVENT(IN(0))<0 THEN 'trigger falling edge of IN0 PRINT "IN0 OFF" ENDIF WEND </pre> Example Two: Scan register, variables <pre> WHILE 1 IF SCAN_EVENT(TABLE(0))>0 THEN 'trigger rising edge of TABLE0 PRINT "TABLE0 ON" ELSEIF SCAN_EVENT(TABLE (0))<0 THEN 'trigger falling edge of TABLE0 PRINT "TABLE0 OFF" ENDIF WEND </pre> Operate table(0) online, and print results.

Instructions	IN_SCAN , IN_EVENT

IN_BUFF -- Buffer Input Read

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.
Syntax	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, the longer filtering time, value is:2-9, default is 2.
Syntax	VAR1 = INFILTER, INFILTER= expression
Controller	General
Example	INFILTER= 5 'increase the filtering time when there is terrible interruption.

IN_MSFILTER -- IN Filter Settings

Type	Special Command
Description	Set the filter for one single input.
Syntax	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

LOCALIO_OFFSET -- Local IO Offset

Type	XPLC300 specialized command.
Description	This is used for offset XPLC300 controller's local IO address.
Syntax	LOCALIO_OFFSET=value value: IO starting address, the default value is 0 Note: it only can be a multiple of 8.
Controller	XPLC300
Example	LOCALIO_OFFSET=8 'offset local IO address to 8

12.2. Output Instructions

OP -- Output Ports

Type	Input and output instructions and functions
Description	Output signals or read outputs status When it is used in expression, it is regarded as function Syntax automatically. IO channel No. 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 No. should exceed IO max No. of controller itself, and it can't superpose with controller No.. Maximum operation output port No. is 32.
Syntax	OP([ionum],value) or OP(ionum1, ionum2,value[,mask]) OP([firstnum],[finalnum]) - ionum: output No. starting 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 channels both are operated when it is not filled. - firstnum: output No., starting from 0. - finalnum: output No., starting from 0, it reads one single output status if this parameter is not filled.
Controller	General
Example	Example 1: single operation

	<pre>'invert output 0 IF OP (0) = ON THEN OP (0,OFF) ELSE OP (0,ON) ENDIF</pre> Example 2: regional operation <pre>OP(0,7,\$FF) 'bit0-bit7 full open DELAY(1000) OP(0,7,0)</pre> <pre>OP(8,15,\$FF) 'bit8-bit15 full open DELAY(1000) OP(8,15,0)</pre> <pre>OP(0,15,\$FFFF) 'bit0-bit15 full open DELAY(1000) OP(0,15,0)</pre> <pre>OP(0,31,\$FFFFFFF) 'bit0-bit31 full open DELAY(1000) OP(0,31,0)</pre>
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.
Syntax	AOUT(channel) = value channel: analog output channels 0-63 DA channel No. is related to dial-up switch configuration on expansion module, the starting value is (4+dial-up value*4).
Controller	General
Example	<pre>AOUT(1) = 0 'close output DA channel 1. AOUT(1) = 4095 'DA1 output voltage is 10V.</pre>
Instructions	AIN

READ_OP -- Output Read

Type	Input and output functions
Description	Read outputs status. Same as OP, output as per bits in terms of multi-output operation.
Syntax	READ_OP ([firstnum],[finalnum]) ● firstnum: first output No., starting from 0. ● finalnum: last output No., starting from 0, it reads one 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 -- EXIO Interface Configuration

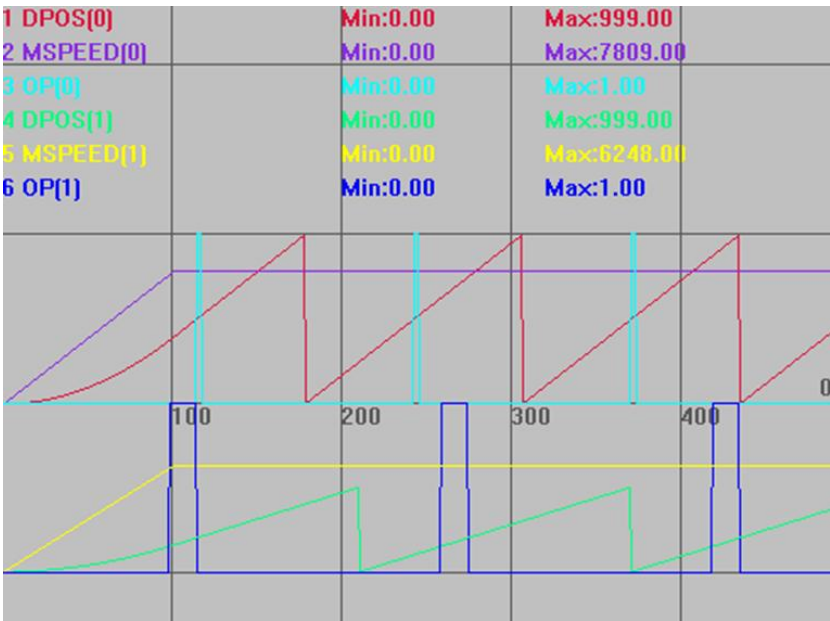
Type	Input and output functions
Description	Assign inputs and outputs of EXIO expansion interface as per bit, and it needs to be used together with customized adapter board. Use Fiber adapter board IO configuration instruction “EXIO_DIR(0, \$8FFFF), YAG adapter board IO configuration instruction “EXIO_DIR(0, \$FCBFE) and SPI adapter board IO configuration instruction “EXIO_DIR(0, \$FFFFA).
Syntax	Command Syntax: Exio_Dir(isel, idirbit) Function Syntax: Exio_Dir(isel) ● isel: Exio selection, fixed value 0 ● 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 Output (Soft)

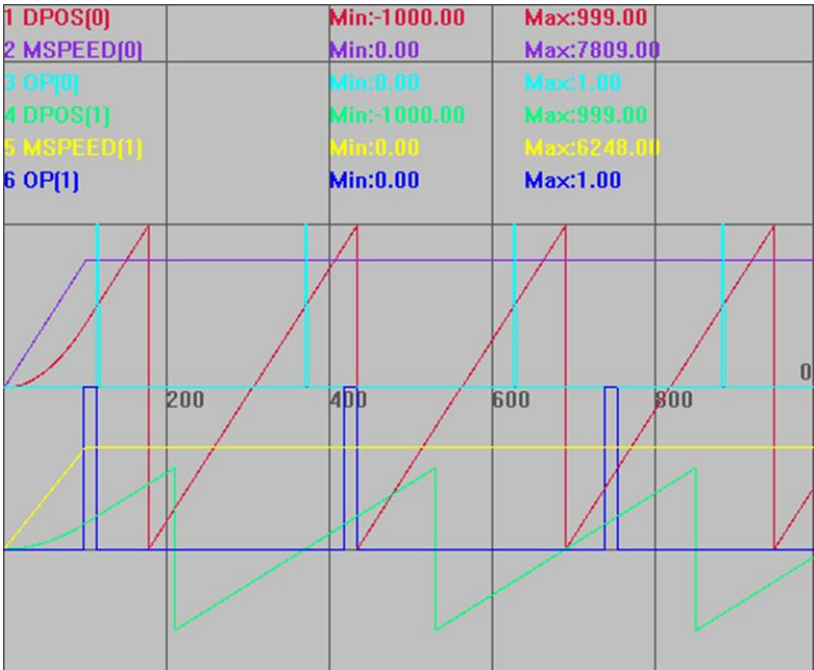
Type	Inputs and Outputs Instructions

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.
Syntax	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	<pre> 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 </pre>



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 -- Position Comparison Output (Hard)

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. When each compare point is triggered, current OUT port's electrical level will be inverted. HW busffers are 1024, totally 1024 HW instructions can be called continuously. After HW instruction is called, it won't be affected by followed coordinates. And please note HW coordinates in TABLE must be correct, therefore, it is better to modify coordinate by manual, and try to avoid conflict between HW instruction and coordinate cyclic auto-change. “auto-change coordinates in cycle“ is not controlled by the program, no way to determine it is before HW or after HW, which means TABLE coordinates also can't be confirmed. This instruction only supports pulse axis hardware position comparison output, for EtherCAT / RTEX controllers, please use HW_PSWITCH2. For pulse type motor, when ATYPE=4, it is the MPOS. Default ATYPE=1/7, it's DPOS. Firmware 240111 ZMC432 adds “HW Comparison Acceleration”. You can make it compare and output 32 times in one cycle, and when the period is 4K (250u), the comparison speed can be up to 128kHz. When the period is below 8K (125u), speed can achieve 256kHz.
Syntax	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. where saves first comparison coordinate. ● tableend : TABLE No. where saves last comparison coordinate. If there are some points that are comparing, mode must be set as 2, because HW_PSWITCH(2) can stop comparing, delete unfinished points, otherwise, it will cause abnormal work of output channels. ➤ Vector Comparison Descriptions (for ZMC432 special firmware):

	a) OUT0 (OP0) is used for vector comparison, front 6 physical encoder axes are supported. b) Axis type “ATYPE” must be configured as the one that has local encoder, because MPOS (feedback) must respond fast enough. If not, please use HW_PSWITCH (2). c) The initial OP state must be correct, because the OP state will invert after every trigger. If it is wrong, following states all may be wrong. d) Before moving, recommend to clear data. e) Don’t use coordinates loop, or don’t modify the coordinates while moving. ➤ Vector Comparison Syntax: 1) Mode = 5 – HW_PSWITCH(5, vectstart, repes, cycledis, ondis) Vectstart: comparison point’s vector coordinates. Repes: repeat cycle, compare 2 times in one cycle, it will output valid state at first, then outputs invalid state. Cycledis: cycle distance, output Opstate every this distance, after Ondis, it will restore as invalid state. Ondis: output valid state’s distance, “(cycledis - ondis)” is the invalid state’s distance 2) Mode = 6 – HW_PSWITCH(6, vectstart, repes, cycledis, ondis) Vectstart: comparison point’s vector coordinates. Repes: repeat cycle, compare once in one cycle. Cycledis: cycle distance, output once every this distance. 3) Mode = 7 – HW_PSWITCH(7, tablestart, tableend) tablestart: TABLE No. where saves the first comparison point’s vector coordinates. tableend: TABLE No. where saves the last comparison point’s vector coordinates. 4) Mode = 25 – HW_PSWITCH(25, axes, num, tablestart) axes: how many axes compared num: how many points compared tablestart: TABLE No. where saves the first comparison point’s MPOS coordinates, each point occupies axes TABLE, saving each axis’ MPOS in sequence. ➤ Through this method, the comparison point coordinates are input directly, then it will confirm the comparison direction according to the coordinate relation. And for the motion whose direction has different directions, please try to write several HW_PSWITCH commands.
Controller	ZMC4XX series or above, with firmware version above 20170704. ZMC420SCAN doesn’t support HW_PSWITCH. Vector comparison is only valid in ZMC432 controller’s special firmware.

Example

Example: ZMC432, firmware: 20170709 (invalid under simulation)

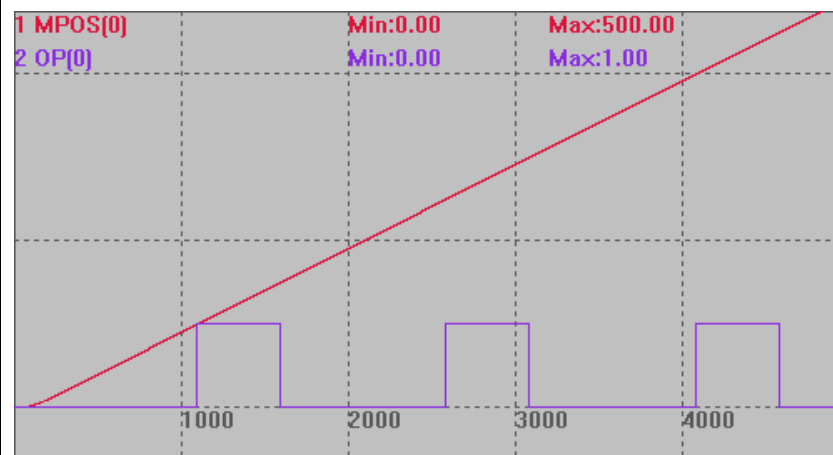
```

BASE(0)          'select axis 0 to output OP(0) by default
ATYPE=4          'encoder position is used as comparison output reference
                  point, if without 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)



Example of Vector Comparison (ZMC432 special firmware):

- **Mode 5**

```

BASE(0)
ATYPE = 1
DPOS= 0
MPOS= 0
SPEED = 1000
ACCEL= 1000
DECEL= 1000
VECTOR_MOVED = 0

```

```

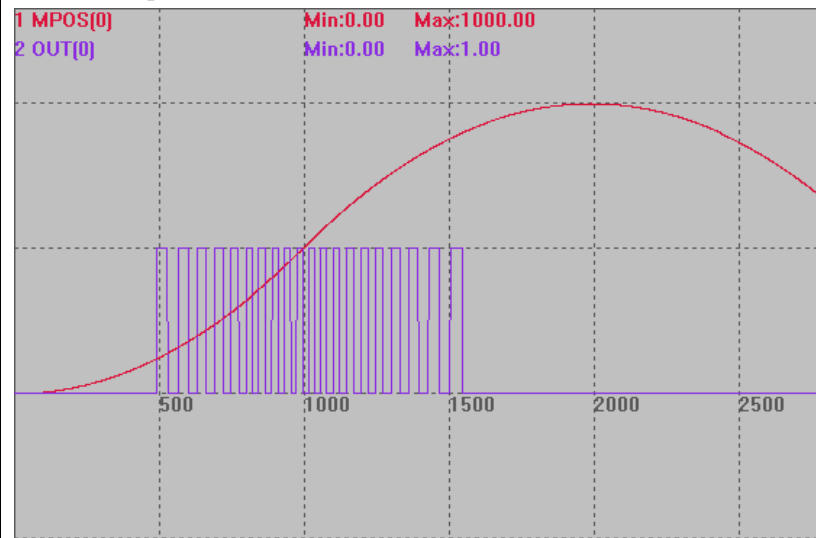
HW_PSWITCH(2)
HW_PSWITCH(5,100,20,40,20)
TRIGGER
MOVEABS(1000)
WAIT UNTIL IDLE(0)
MOVEABS(0)
WAIT UNTIL IDLE(0)

```

--effect showing--

MPOS (0) is captured, vertical scale is 500

OP (0) is captured, vertical scale is 1



● Mode 6

```

BASE(0)
ATYPE = 1
DPOS= 0
MPOS= 0
SPEED = 1000
ACCEL= 1000
DECEL= 1000
VECTOR_MOVED = 0

```

```

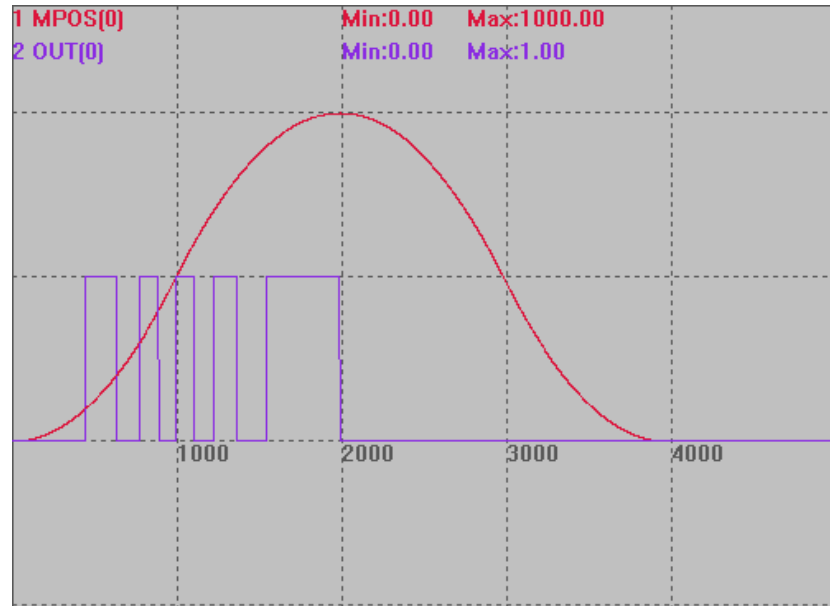
HW_PSWITCH(2)
HW_PSWITCH(6,100,10,100)
TRIGGER
MOVEABS(1000)
WAIT UNTIL IDLE(0)
MOVEABS(0)
WAIT UNTIL IDLE(0)

```

--effect showing--

MPOS (0) is captured, vertical scale is 500

OP (0) is captured, vertical scale is 1



● Mode 7

BASE(0)

ATYPE = 1

DPOS= 0

MPOS= 0

SPEED = 1000

ACCEL= 1000

DECEL= 1000

VECTOR_MOVED = 0

TABLE(100,50,100,150,200,250,300,350,400,450,500,550,600,650,700,750,800,850,900,950)

HW_PSWITCH(2)

HW_PSWITCH(7,100,117)

TRIGGER

MOVEABS(1000)

WAIT UNTIL IDLE(0)

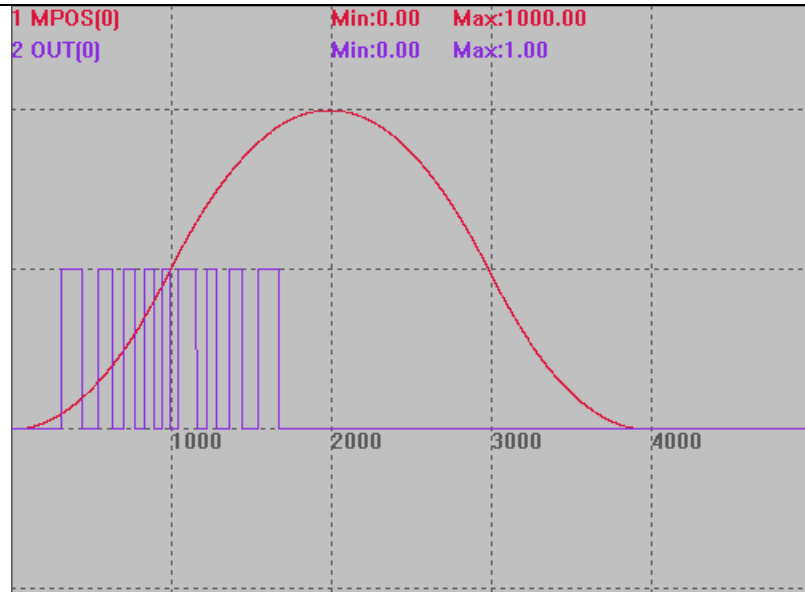
MOVEABS(0)

WAIT UNTIL IDLE(0)

--effect showing--

MPOS (0) is captured, vertical scale is 500

OP (0) is captured, vertical scale is 1



● **Mode 25**

BASE(0,1)

ATYPE = 5,5

DPOS= 0,0

MPOS= 0,0

SPEED = 100,100

ACCEL= 1000,1000

DECEL= 1000,1000

VECTOR_MOVED = 0

TRIGGER

for i=0 to 22

 table(i*2) = i

 table(i*2 + 1) = i

 table(100+i*2) = (22-i)

 table(100+i*2 + 1) = (22-i)

next

HW_TIMER(2, 2000, 1000, 1, 0, 0)

HW_PSWITCH(25, 2, 22, 0)

HW_PSWITCH(25, 2, 22, 100)

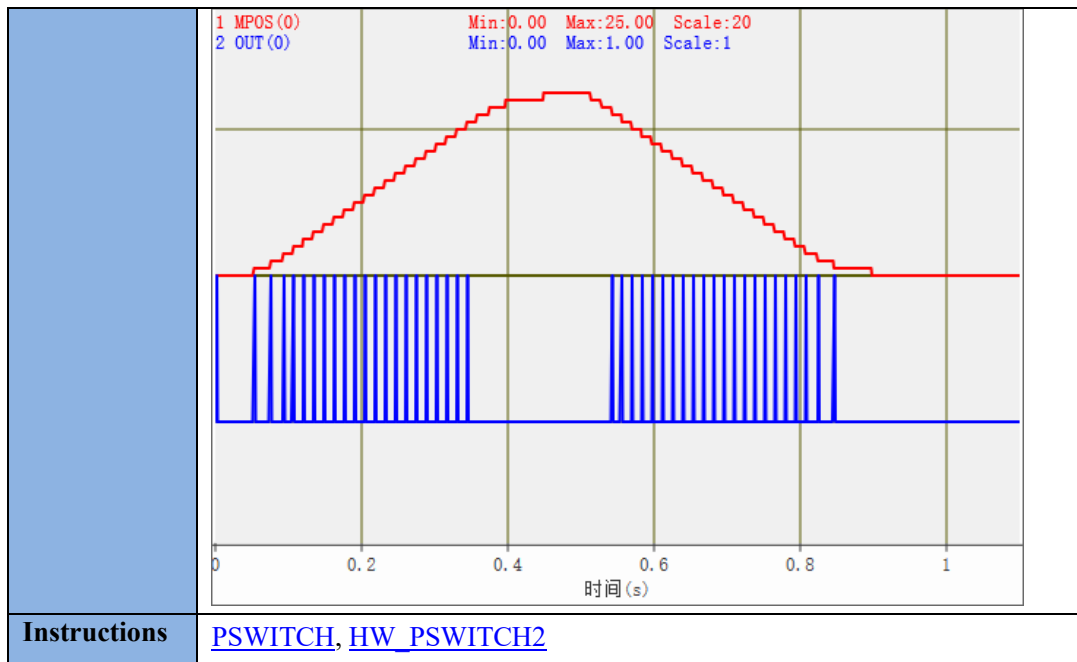
MOVE(25, 25)

MOVE(-25,-25)

--effect showing--

MPOS (0) is captured, vertical scale is 20

OUT (0) is captured, vertical scale is 1



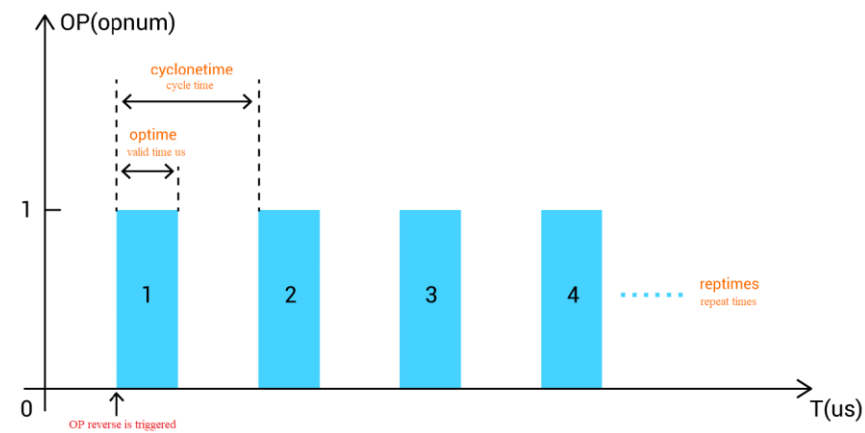
HW_TIMER -- Hardware Timing

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.
Syntax	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.

HW_TIMER



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.

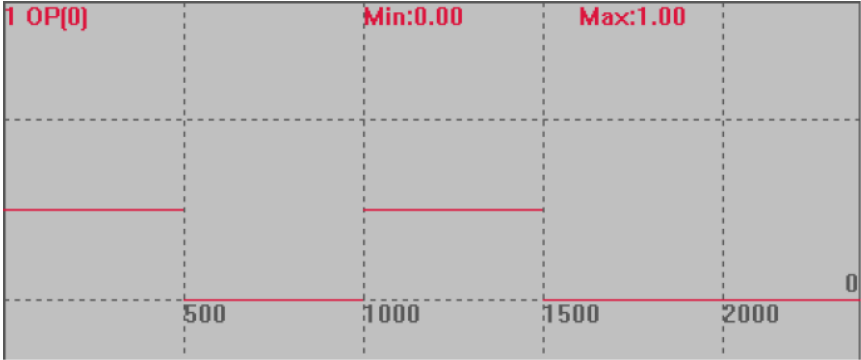
Galvanometer controllers with firmware version above 20170710 add **mode 1**. Mode 1 supports modifying timer parameters dynamically after again sending, but mode 1 can't be used to turn on hardware timer, and it takes effects when comparison in mode 2 is not completed. Mode 2 can't be turned on repeatedly.

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)

trigremaintime: 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.

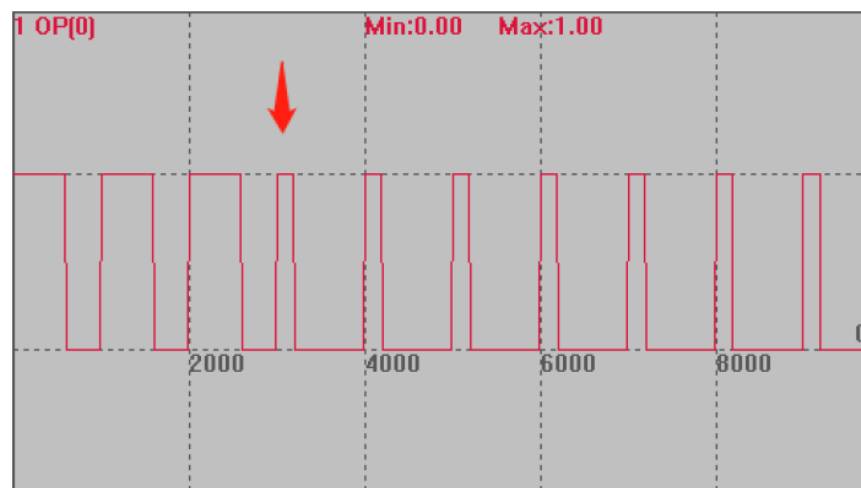

```

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.

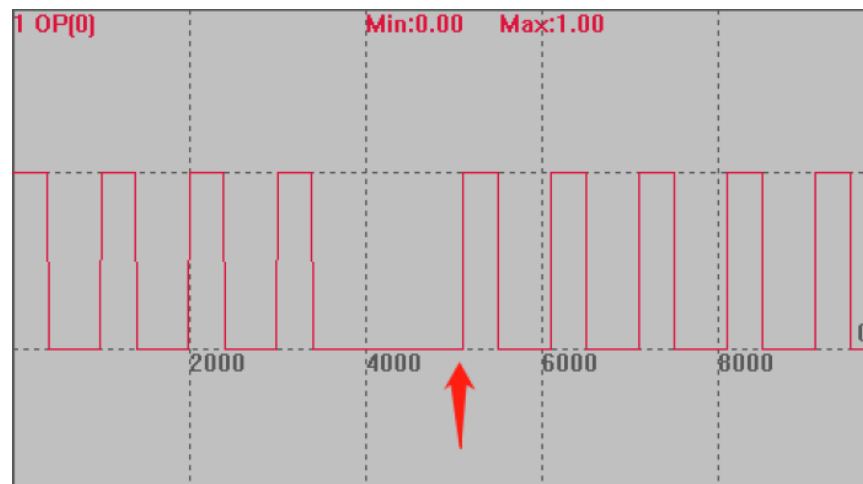
OP(0, ON)

```

```
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.



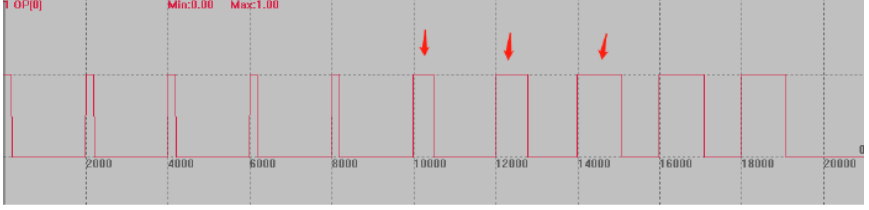
Example 4:

```
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.

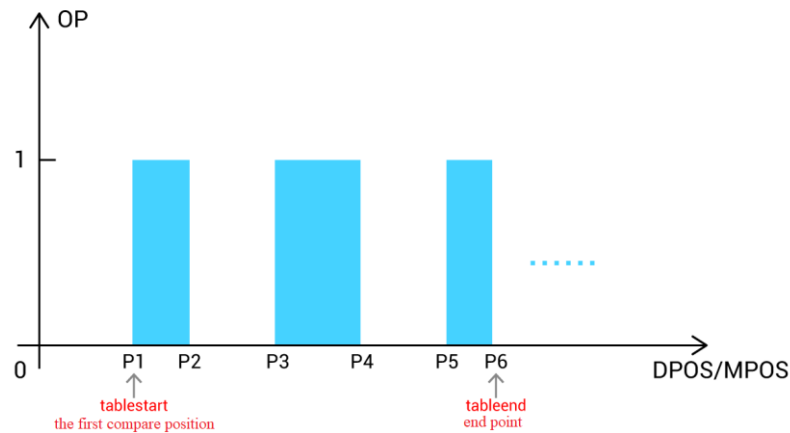
	
Instructions	HW_PSWITCH2

HW_PSWITCH2 -- Bus Position Comparison Output (Hard)

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.
Syntax	Command Syntax: HW_PSWITCH2(mode, [...]) Function Syntax: 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 comparison position ● tablestart: TABLE number that saves the first absolute comparison coordinate ● tableend: 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.

HW_PSWITCH2
mode=1



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.

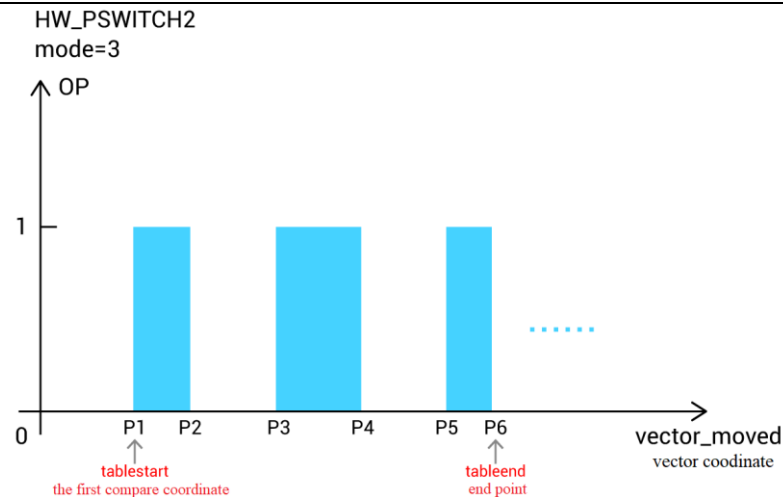
In vector compare mode, comparison is between VECTOR_MOVED and set positon, it is recommended to set an initial value of VECTOR_MOVED.

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
- tableend: 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.

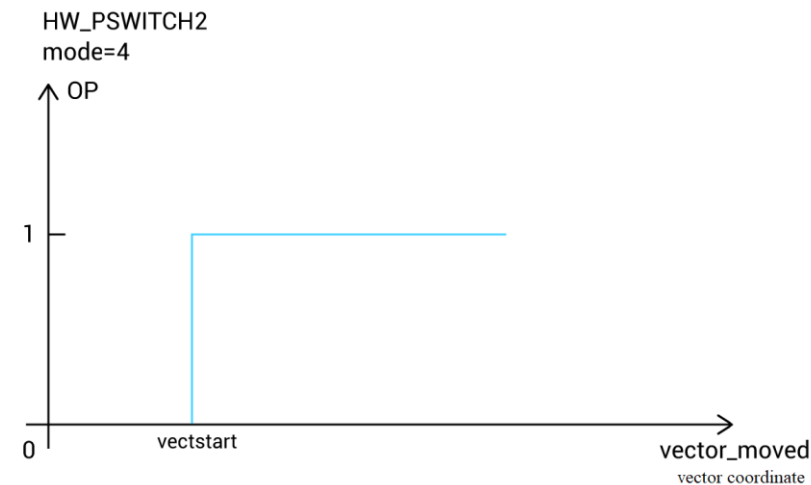


Mode=4:vector compare mode, single comparison position.

HW_PSWITCH2(4,opnum,opstate,vectstart)

- mode: 4-start comparer
- opnum: relevant outputs
- opstate: output status of the first comparison position
- vectstart: absolute coordinate of comparison position

Description: achieve one compare vector position set by sommand, OP reverse, compare ends.



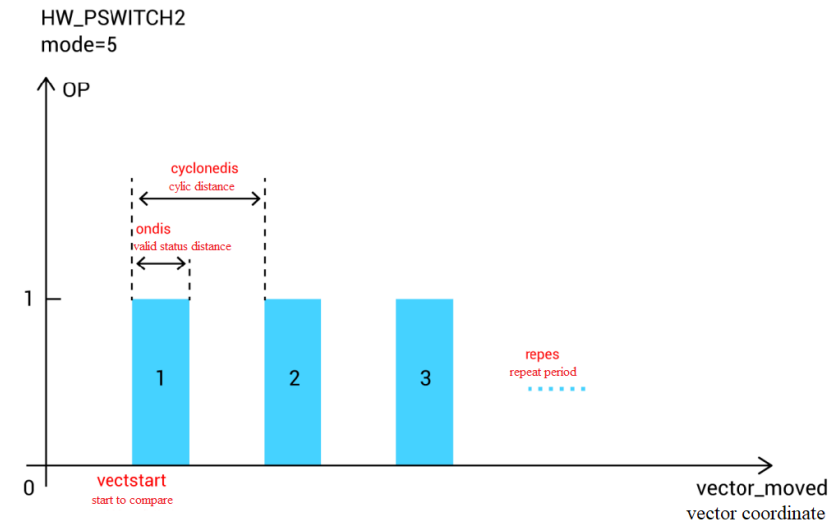
Mode=5: Vector compare mode, cycle pulse mode.

HW_PSWITCH2(5,opnum,opstate,vectstart,repes,cycledis,ondis)

- mode: 5-start comparer
- opnum: relevant outputs
- opstate: output status of the first comparison 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 cyclonedis 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.

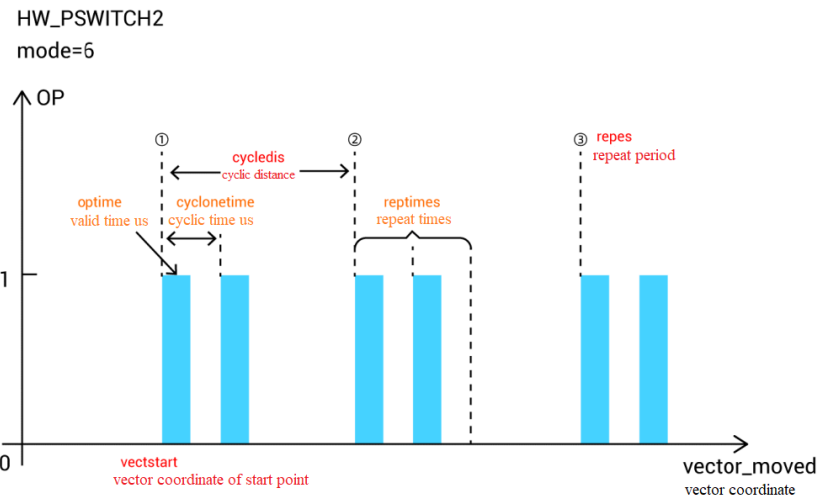


Mode=6 :vector compare mode, cycle mode, it is used together with HW_TIMER

HW_PSWITCH2(6,opnum,opstate,vectstart,repes,cycledis)

- mode: 6-start comparer
- opnum: relevant outputs
- opstate: output status of the first comparison 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.



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 together with HW_TIMER.
- Opnum: relevant outputs
- 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

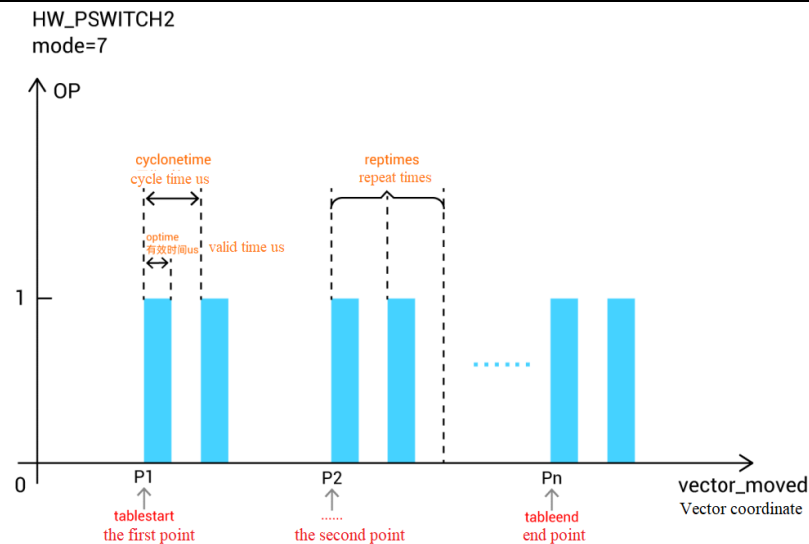
[when used together with hwtimer, it can adjust hwtimer parameters dynamically]

Optimeus: adjust HW_TIMER valid time dynamically.

Optimes: adjust HW_TIMER pulse amount dynamically, 0-not output

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 TABLE 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.



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, hwcycetimeus])

HW_PSWITCH2(35, opnum, opstate, maxerr, num, tablepos)

HW_PSWITCH2(36, opnum, opstate, maxerr, num, tablepos, [ophwtimeus, ophwtimes, hwcycetimeus])

- 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
- hwcycetimeus: pulse period

Note: under this mode, deviation parameter 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 Specific Usage Range :

HW_PSWITCH2 – Mode 1 & HW_TIMER: span one position, trigger once.

HW_PSWITCH2 – Mode 8 & HW_TIMER: every position, trigger once.

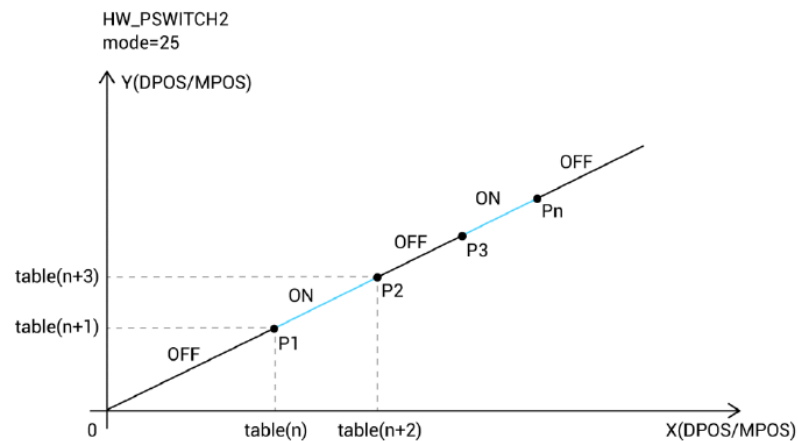
Pulse axis, bus axis, and virtual axis are OK.

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.

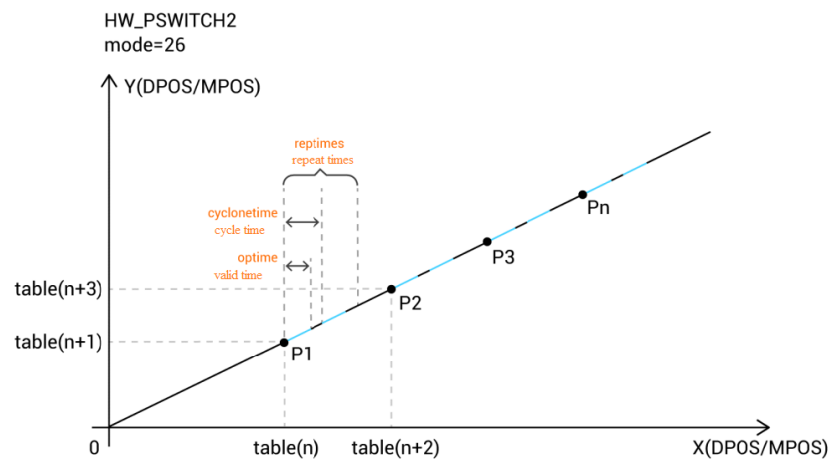
Below is the example, blue segment means OP is ON, all kinds of common used interpolation motions support comparison, please pay attention the coordinate of compare point must be accurate, otherwise, following comparison positions will be affected.



Mode = 26: 2D compare, reused together with HW_TIMER.

HW_PSWITCH2(26, opnum, opstate, maxerr, num, tablepos, [ophvertimeus, ophwtimes, hwcycetimeus])

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 similar with mode 7 and mode 36.

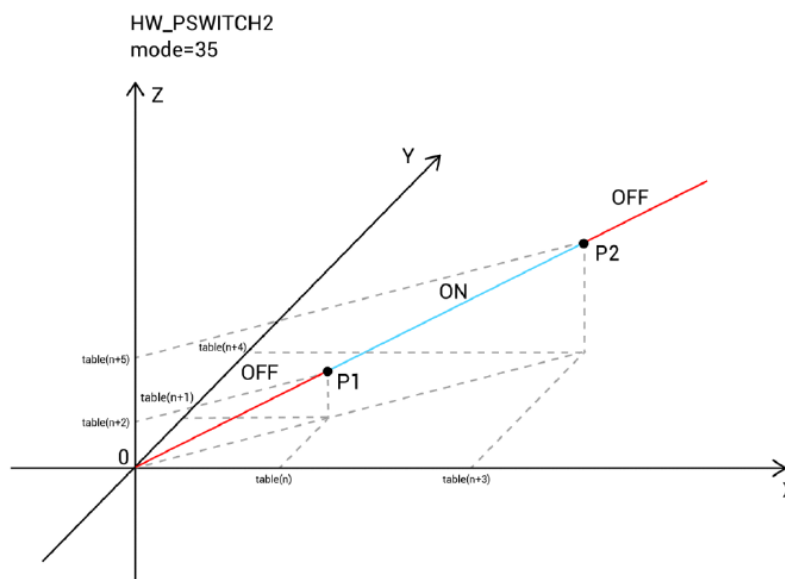


Mode = 35: 3D compare

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.

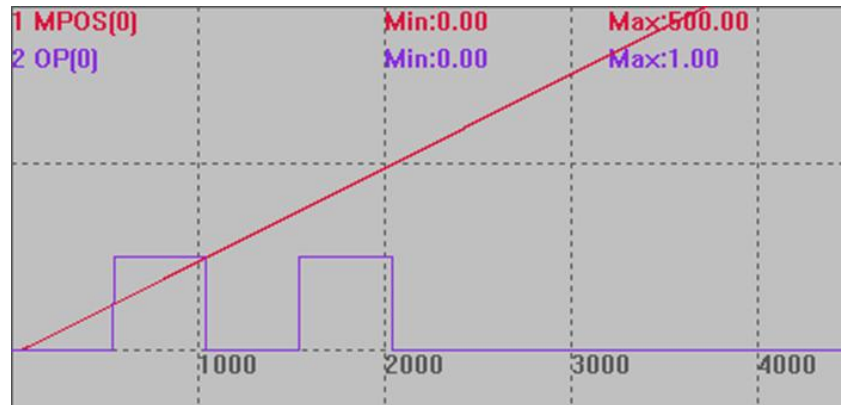


Mode = 36, 3D compare, reused together with HW_TIMER.

HW_PSWITCH2(36, opnum, opstate, maxerr, num, tablepos, [ophvertimeus, ophwtimes, hwcycltimeus])

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

	and mode 26.
Controller	Some ZMC3XX, above ZMC4XX Series and ZMC4XX series with firmware version above 20170704 supports.
Example	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)



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.

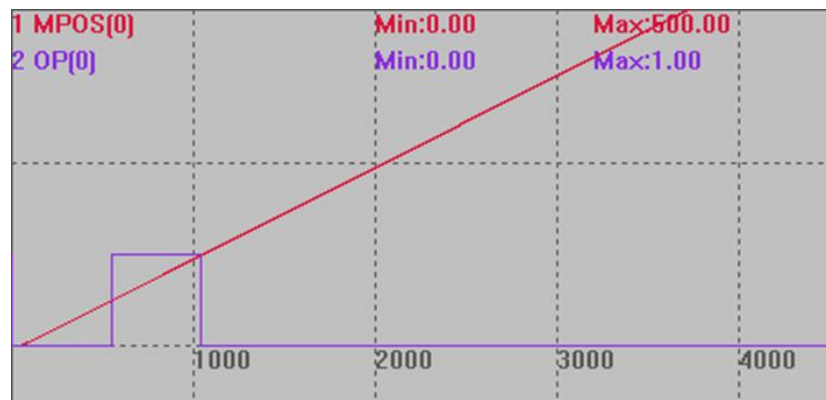
TRIGGER 'trigger oscilloscope

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)

```

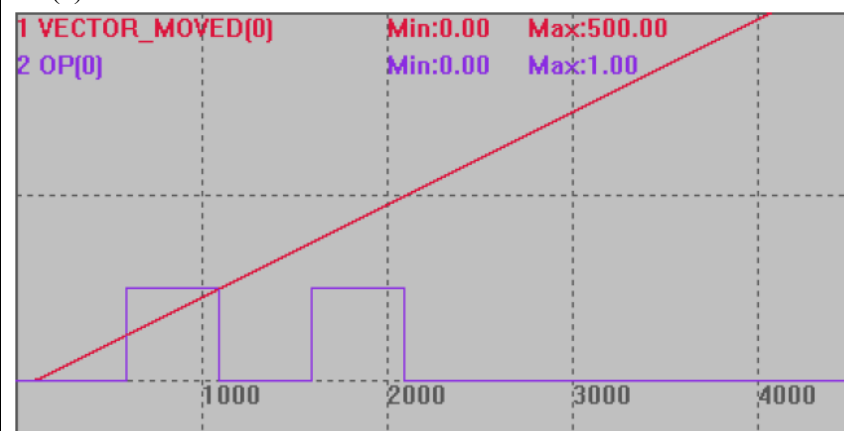
WAIT IDLE(0)
WAIT IDLE(1)
BASE(0,1)
ATYPE=1,1
SPEED=100,100
ACCEL=1000,1000
DECEL=1000,1000
SRAMP=100,100
DPOS=0,0
MPOS=0,0
OP(0,OFF)
TABLE(0,50,100,150,200) 'compare point coordinate configuration
VECTOR_MOVED=0           'set vector start position
HW_PSWITCH(2)           'stop and delete comparison 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

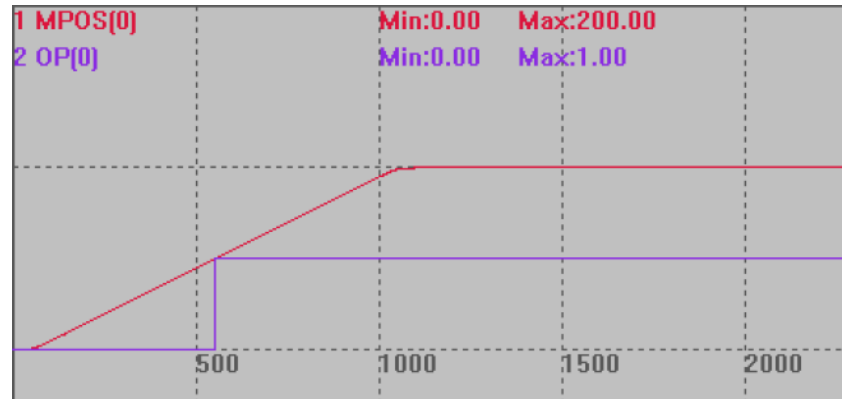
```

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.

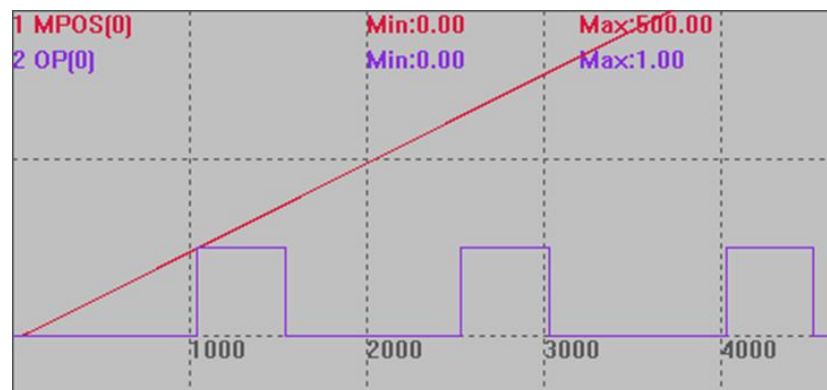
BASE(0)
DPOS=0
MPOS=0
OP(0,OFF)
VECTOR_MOVED=0 'set start position of vector as 0 for observing
HW_PSWITCH2(2) 'stop and delete comparison that not finished.
HW_PSWITCH2(5,0,1,100,3,150,50) 'start to compare at 100, all 3 times
cycle distance is 150, output valid
distance is 50

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

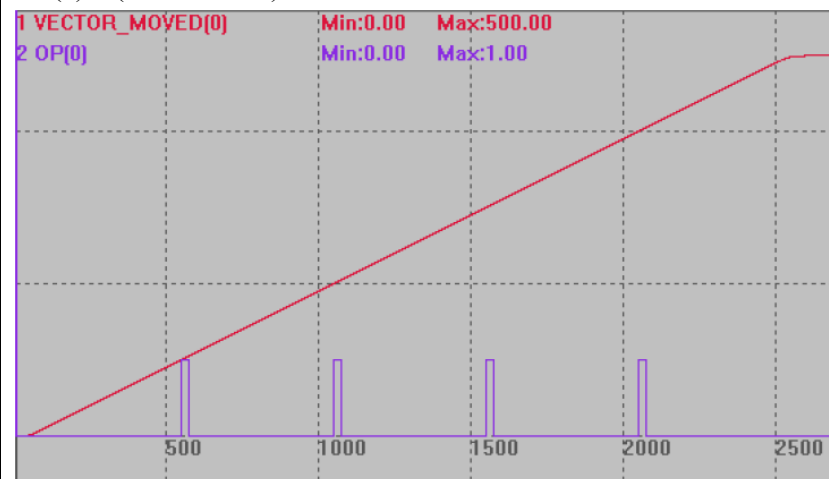
MOVE(500)

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:

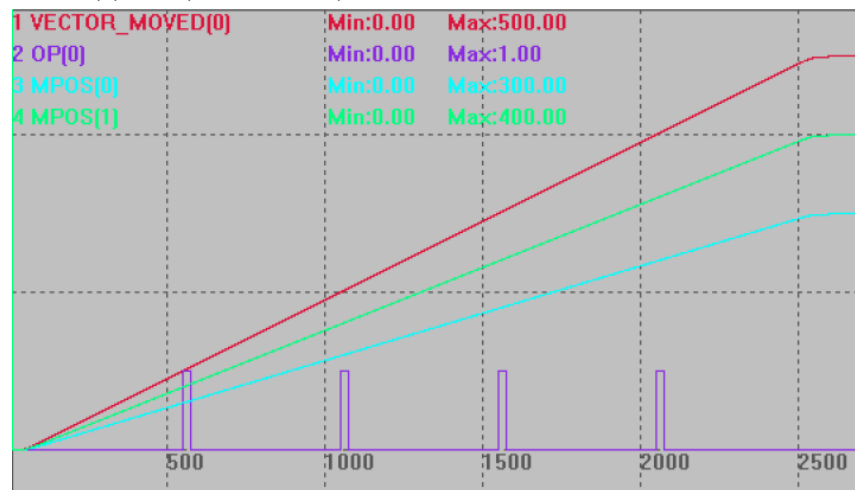
Coordinates are resultant vector position of two axes, comparison starts at position 100, compare 4 times. Turn to OFF 500ms later after output is ON.

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

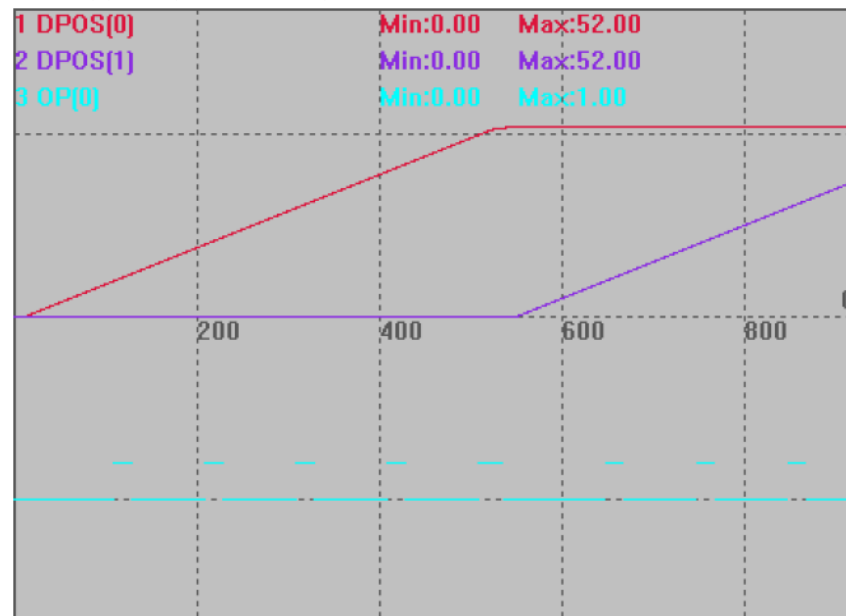
END

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)

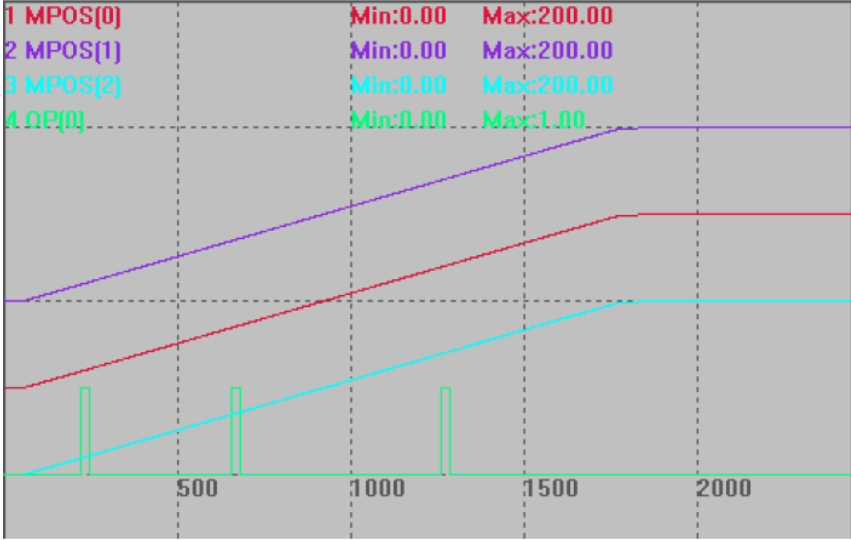
TABLE(0, 20,20,20, 40,40,40, 70,70,70, 100,100,100, 140,140,140, 180,180,180)

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_HWPSIWTC2, PSWITCH, HW_PSWITCH, MOVEOP_DELAY, REG_INPUTS

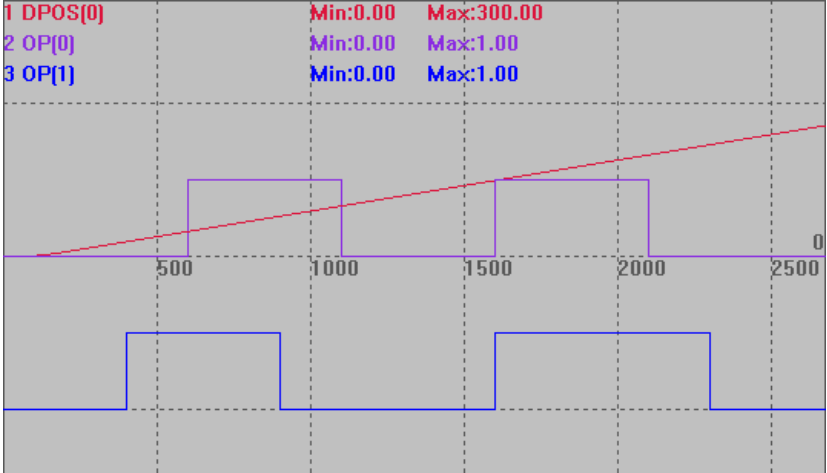
HW_MINTIME -- HW Min Time Space

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.

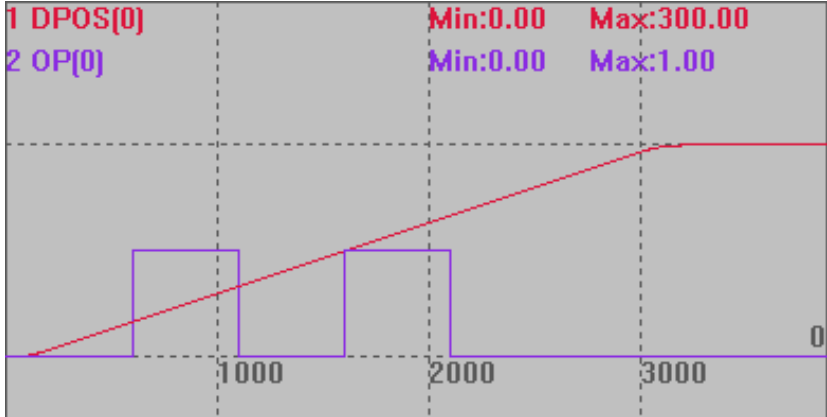
	XPCIE1032H (version after RT0925), trigger description of PSO mode 6 Ts: real feedback position space time in hardware position comparison Tcyc: trigger period time Top: trigger valid period time Tmin: the minimal space time between two triggers Command time parameter priority: HW_MINTIME>TIME, HW_WITCHPS2>HW_TIME
Syntax	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 -- PS2 Axis No. Settings

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.
Syntax	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.) <pre> 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 </pre> 
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.
Syntax	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) <pre> 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 </pre> 
Instruction	HW_PSWITCH2

12.4. PWM Control Instructions

PWM_FREQ -- PWM Frequency

Type	PWM control functions
Description	PWM frequency setting or reading. PWM is turned-off by setting duty cycle as 0 (not to set PWM frequency to 0), PWM frequency should be modified before PWM switch setting.
Syntax	PWM_FREQ (index, freq) or PWM_FREQ (index)=freq ● index: PWM output No., starting 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 -- PWM Duty Cycle

Type	PWM control functions
Description	PWM duty cycle setting or reading. PWM is turned-off by setting duty cycle as 0 (not to set PWM frequency to 0), PWM frequency should be modified before PWM switch setting. Duty cycle means the ratio of valid electric level to whole period. In one period, output valid electric level at first, then output invalid electric level. duty cycle is 0.5  decrease duty cycle  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.
Syntax	PWM_DUTY(index, duty) or PWM_DUTY(index)=duty ● index: PWM output No., starting 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.
Syntax	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 13 Communication Instructions

13.1. Serial Communication Instructions

SETCOM -- Serial Port Configuration

Type	System Instructions																						
Description	Configure serial port. When the controller repowered-on, SETCOM parameters will become default values, please set SETCOM at the beginning of program. ZMC00X series don't support MODBUS communication as master. Generally, delay is needed for switching to RS485 station No.																						
Syntax	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 border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <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> </tbody> </table> ● port: serial PORT No.: 0-1, please refer to PORT. ● mode: protocol <table border="1"> <thead> <tr> <th>value</th><th>Description</th></tr> </thead> <tbody> <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> </tbody> </table> ● variable: choose register type, 0-VR, 1-TABLE, 2-MODBUS <table border="1"> <thead> <tr> <th>value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>VR, one VR is mapped to one MODBUS_REG in this situation. VR is 32-bit float type, REG is 16-bit integer type. So when transferring VR data to REG type, the decimal part may be lost. And if the VR data exceeds positive or minus 15 bits, then the REG data will be changed.</td></tr> </tbody> </table>	Value	Description	0 (default)	No verification	1	Verified while odd	2	Verified while even	value	Description	0	RAW data mode, no protocol, at this time, use GET, PRITNT # to transfer data.	4 (default)	MODBUS slave (16 bits integer)	14	MODBUS master (16 bits integer)	15	Direct command mode, it can input character string directly through serial port (use line feed to make an end)	value	Description	0	VR, one VR is mapped to one MODBUS_REG in this situation. VR is 32-bit float type, REG is 16-bit integer type. So when transferring VR data to REG type, the decimal part may be lost. And if the VR data exceeds positive or minus 15 bits, then the REG data will be changed.
Value	Description																						
0 (default)	No verification																						
1	Verified while odd																						
2	Verified while even																						
value	Description																						
0	RAW data mode, no protocol, at this time, use GET, PRITNT # to transfer data.																						
4 (default)	MODBUS slave (16 bits integer)																						
14	MODBUS master (16 bits integer)																						
15	Direct command mode, it can input character string directly through serial port (use line feed to make an end)																						
value	Description																						
0	VR, one VR is mapped to one MODBUS_REG in this situation. VR is 32-bit float type, REG is 16-bit integer type. So when transferring VR data to REG type, the decimal part may be lost. And if the VR data exceeds positive or minus 15 bits, then the REG data will be changed.																						

			No loss will happen if REG is transmitted to VR.
	1		TABLE, one table is mapped to MODBUS_REG. (it is not recommended). TABLE is 32-bit float type, REG is 16-bit integer type. So when transferring TABLE data to REG type, the decimal part may be lost. And if the TABLE data exceeds positive or minus 15 bits, then the REG data will be changed. No loss will happen if REG is transmitted to TABLE.
	2 (default)		System MODBUS register, VR and MODBUS both belong to 2 independent registers.
	3		VR_INT mode, one VR_INT is mapped to two MODBUS_REG in this situation.
● timeout: this parameter takes effect when MODBUS slave station. Timeout means the longest delay time between frames (millisecond ms), for old firmware, the default 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 VR or TABLE is set, general output ports will be mapped to MODBUS_BIT(0), and general input ports will be mapped to MODBUS_BIT(1000), it is not recommended to use MODBUS_BIT as HMI button in this situation. ⚠ When VR or TABLE is set, outputs and inputs that are not used will connect together.			
Controller	General		
Example	Example 1: mode0, RAW mode <pre> 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 channel 0 in char1 PRINT char1 'print the character received from channel 0 in ASCII code. PUTCHAR #0, char1 'send the received character back. WEND </pre> In Chapter 23 Simple Routines -- Panasonic A6 servo encoder reading. Example 2: MODBUS communication configuration <pre> SETCOM(38400,8,1,0,0,4,2) 'set serial port 0 as MODBUS slave, baudrate is 38400. </pre>		

	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 3: 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 4: register mode 0 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) as 100.345 ?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 5: 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) as 100.345 ?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.
Syntax	ADDRESS = value
Controller	General
Example	Print ADDRESS 'print protocol station NO. Result: 1
Instructions	SETCOM , PORT , PROTOCOL

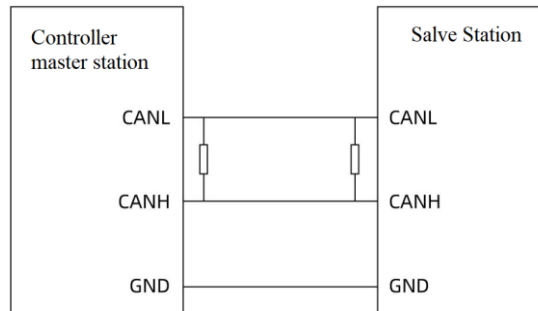
COM_UNUSED -- Serial Port Settings

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 RTBASIC Bit1: serial port 2, value as 1 means not use RTBASIC
Syntax	COM_UNUSED = value
Controller	Above ZMC5XX series controllers
Example	COM_UNUSED=1 'RTBASIC doesn't use RS232 serial port
Instruction	SETCOM , PORT , PROTOCOL

13.2. CAN Communication Instructions

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.

Syntax

CAN(channel, function, tabenum)

- channel: CAN channel, 0-first channel, -1-default channel.
- function: function No.

Value	Description
6	receive data: no data, identifier<0
7	send data
16 (please upgrade firmware)	receive extended data: no data, identifier<0
17 (please upgrade firmware)	send extended data, for normal data, please use 7.

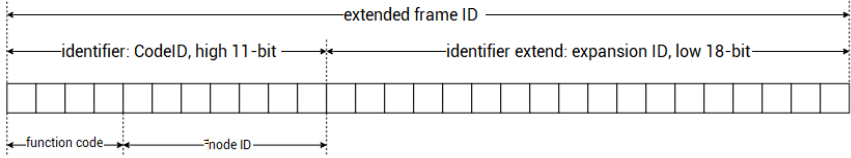
- tabenum: TABLE position where saves data.

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: <pre>'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)</pre> Example two: <pre>'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)</pre> Example three: CAN mode 17 (avoid overflow), independent task ON When no device is connected, CAN17 will jump automatically, it will not appear blocking and waiting. <pre>CANIO_ADDRESS = 32 'master station GLOBAL FLAG_SEND_NUM 'sending times FLAG_SEND_NUM = 1000 WHILE 1 ?FLAG_SEND_NUM</pre>

	<pre> 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 </pre>
Instructions	CANIO_ADDRESS , CANIO_STATUS , CANIO_ENABLE

CANIO_ADDRESS -- CAN Communication Settings

Type	System Parameters										
Description	Set CANID and CAN Speed for controller CAN. Can speed of IO expansion should be set through DIP switch attached, and the setting value will be saved into FLASH, it is valid after restarting. 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 : <table border="1"> <thead> <tr> <th>Value in upper 8 bits</th><th>CAN baud rate</th></tr> </thead> <tbody> <tr> <td>0</td><td>500KBPS(default value)</td></tr> <tr> <td>1</td><td>250KBPS</td></tr> <tr> <td>2</td><td>125KBPS</td></tr> <tr> <td>3</td><td>1MBPS</td></tr> </tbody> </table> Notes: 1. Don't configure multiple master controllers in one communication net. 2. Setting of CANID and CANSEED will be effective only after restarting.	Value in upper 8 bits	CAN baud rate	0	500KBPS(default value)	1	250KBPS	2	125KBPS	3	1MBPS
Value in upper 8 bits	CAN baud rate										
0	500KBPS(default value)										
1	250KBPS										
2	125KBPS										
3	1MBPS										
Syntax	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 to IO expansion. CANIO_ADDRESS=1+256 'set CANID=1 as slave, 250KBPS, used to do as ZCAN slave station CANIO_ADDRESS=1+512 'set CANID=1 as slave, 125KBPS, used to do as ZCAN slave station CANIO_ADDRESS=1+768 'set CANID=1 as slave, 1MBPS, used to do as ZCAN slave station CANIO_ADDRESS=3 'set CANID=3 as slave, 500KBPS, used to do as ZCAN slave station CANIO_ADDRESS=8+256 'set CANID=8 as slave, 250KBPS, used to do as ZCAN slave station CANIO_ADDRESS=16+512 'set CANID=16 as slave, 125KBPS, used to do as ZCAN slave station CANIO_ADDRESS=31+768 'set CANID=31 as slave, 1MBPS, used to do as ZCAN slave station See Chapter 23 CAN communication examples .
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) – enable.
Syntax	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, returned values are 1 (ON), 0 (OFF).
Syntax	CANIO_STATUS(cardnum) ● cardnum: this module's DIP switch ID
Controller	Valid in controllers with firmware above 20140325.
Example	Example 1:

	<pre>?*CANIO_STATUS 'print status of all IO expansion modules. Example 2: If CANIO_STATUS(1) =0 THEN 'Judge the connection status of IO module PRINT "IO Expansion 1 is not connected well" ENDIF</pre>
Instructions	CAN , CANIO_ENABLE , CANIO_ADDRESS , CANIO_INFO

CANIO_INFO -- CAN Expansion Information

Type	System status																																				
Description	Read current IO board information, then return to parameter value.																																				
Syntax	CANIO_INFO(canid, isel [, moduleid]) ● canid: expansion module dial code ID ● isel: parameters that are read <table border="1"> <tr><td>0</td><td>Zmotion</td></tr> <tr><td>1</td><td>DEVICE, device No.</td></tr> <tr><td>2</td><td></td></tr> <tr><td>3</td><td></td></tr> <tr><td>4</td><td></td></tr> <tr><td>IO numbers</td><td>Description</td></tr> <tr><td>10</td><td>the number of IN (all)</td></tr> <tr><td>11</td><td>The number of OP (outputs)</td></tr> <tr><td>12</td><td>The number of AIN (analog inputs)</td></tr> <tr><td>13</td><td>The number of AOUT (analog outputs)</td></tr> <tr><td>Add</td><td></td></tr> <tr><td>16</td><td>the number of modules</td></tr> <tr><td>17</td><td>Type No. of sub module, and it must be with moduleid.</td></tr> <tr><td>Reserve</td><td></td></tr> <tr><td>20</td><td>Submodule input</td></tr> <tr><td>21</td><td>Submodule output</td></tr> <tr><td>22</td><td>Submodule AIN</td></tr> <tr><td>23</td><td>Submodule AOUT</td></tr> </table> For example: the address of the first one expansion submodule connected is 0, the second expansion submodule's address is 1, and so on. [how to modify analog AD/DA range 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.	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
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																																				

	Range Type				
	Type No.	Module Type	Type No.	Module Type	Range
	2	4AD Module (IN Module)	10	4DA Module (OP Module)	0-10V
	3		11		-10-10V
	4		12		4-20mA
	5		13		0-20mA
	6		14		0-5V
	7		15		-5-5V
	20	8AD Module (IN Module)	21	8DA Module (OP Module)	0~10V
	Type No.	Module Type			Range
	22	8AD8DA Module (IN & OP Module)			0~10V
Controller	ZMC4XX series controller with firmware version above 170715.				
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_ADDRESS , CANIO_STATUS				

13.3. Self-Defined Communication Instructions

GET# -- String Read

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.
Syntax	Syntax1: GET #PORT, VARIABLE Syntax2: GET #PORT, ARRAY[(startindex)] [,maxchares] Syntax3: charesget = GET #PORT, VARIABLE Syntax4: 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 Syntax1/2: if no data was got, channel will jam, usually used in multi-task; Syntax3/4: it will return byte numbers that is read; Versions below 20150522 only support Syntax 1; For customized Ethernet communication: syntax 3 / 4 recommended For serial port RAW: syntax 1 / 2 recommended; UDP receiving must use Syntax 4, using the 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. Under the UDP_SERVER mode, one packet is received, PORT_TARGET will be the sender automatically, so it can receive multi-salve data.
Controller	General
Example	Example One: <pre> DIM VAR1 SETCOM(38400,8,1,0,0,0) 'open RAM mode. GET #0, VAR1 'get data from channel 0. PRINT VAR1 'print data </pre> Example Two: <pre> 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 </pre>
Instructions	PORT , SETCOM , PRINT#

OPEN# -- Self-Defined Ethernet Communication ON

Type	System Instruction
Description	Open Self-defined Ethernet Communication.
Syntax	OPEN #PORT, "mode",portnum [,ipaddress] ● port: communication channel, refer to PORT description, select self-defined Ethernet channel. ● mode: - TCP_CLIENT -- TCP slave station - TCP_SERVER -- TCP master station - UDP_CLIENT -- UDP slave station - UDP_SERVER -- UDP master station - TCP_EXCUTE -- TCP control directly.

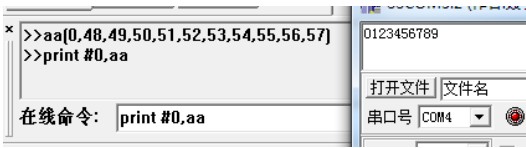
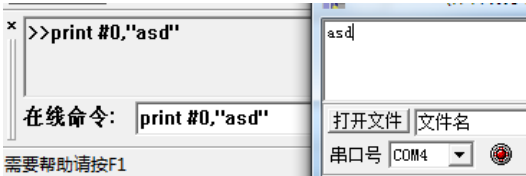
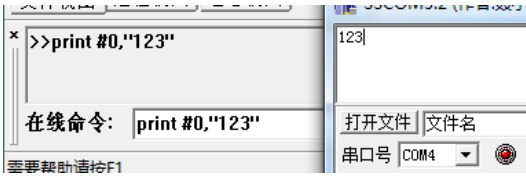
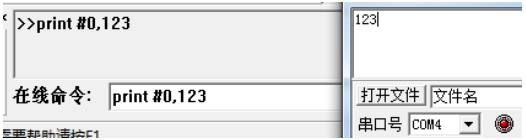
	● portnum: TCP port No. or UDP port No., the master station is local port No., slave is the other side port No. ● ipaddress: the other side IP address, character string, it should be filled when as slave TCP_EXCUTE means TCP control, that is, TCP sends BASIC commands, and it must be with \n end character. OPEN#port, "TCP_EXCUTE", portnum is to set "monitor" for server side. And this remote port No. can't be set as 502. For other port No., it only makes two sides consistent. UDP_SERVER must receive the other side data at first, then send back data. (except use PORT_TARGET to define the other forcibly). Communicate 3s at least, otherwise the communication may break. 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; valid in latest firmware version.
Example	Example 1 <pre>OPEN#11, "TCP_SERVER", 10 'set as master OPEN#10, "TCP_CLIENT", 10, "192.168.1.112" 'set as slave</pre> Example 2 <pre>OPEN#10, "UDP_SERVER", 1000 'UDP master OPEN#11, "UDP_CLIENT", 60000, "192.168.0.120" 'UDP slave</pre> Example 3 <pre>OPEN#3, "UDP_EXECUTE", 500</pre> 'direct TCP control, sent online command character string through TCP client side, the controller side is the server for TCP. See "self-defined Ethernet Communication" for details.
Instructions	PORT , PORT_TARGET , SETCOM , PRINT#

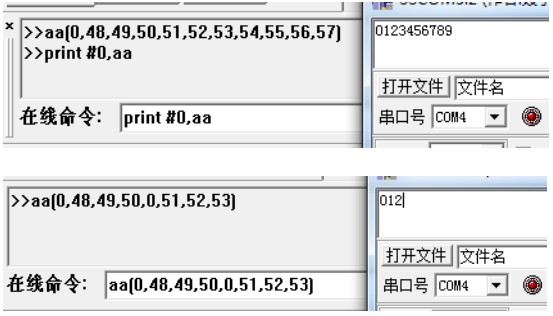
CLOSE # -- Self-Defined Ethernet Communication OFF

Type	System Instruction
Description	Close Self-defined Ethernet Communication.
Syntax	CLOSE # ecustomnum ● ecustomnum: customized net port channel No.

	CLOSE #PORT port: communication channel, close OPEN channels
Controller	General; Valid in latest firmware version.
Example	CLOSE #10 'close channel 10
Instructions	OPEN# , PORT , PORT_TRAGET , SETCOM , PRINT#

PRINT# -- Character String Output

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 it is called every time. PRINT # will send character string directly (" will be deleted itself), no need to do ASCII code switching, 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:

	
Syntax	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# -- Character Output

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 every 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:

	
Syntax	PUTCHAR #PORT, Character - port: channel No. PUTCHAR #PORT, ARRAY(INDEX, NUMES) - port: channel No. - index: output start position. - numes: output byte number, binary type.
Controller	General
Example	<pre> DIM VAR1, ARRAY1(10) 'define array VAR1 = \$FE 'assign value to VAR1 ARRAY1 = "ABCDEFGHJIJ" '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. </pre>
Instructions	GET# , PORT , SETCOM , PRINT#

PORT_TARGET -- IP & Port No. Configuration

Type	Character String Instruction
Description	Configure IP address and port No. of the other side.
Syntax	PORT_TARGET(port)="ipaddress:portnum or PORT_TARGET(port)="ipaddress" - Port: channel No. - Ipaddress: the other IP address - Portnum: port No. Returned Syntax: VAR=PORT_TARGET(port)) return IP address character string.
Controller	ZMC4XX series controller with firmware version above 20170628; XPLC series controller version above 20170702.
Example	<pre> PORT_TARGET="192.168.0.12:502" PORT_TARGET(port)="192.168.0.12" ?PORT_TARGET(port) DIM IPSTRING(100) </pre>

	IPSTRING = PORT_TARGET(port) ?IPSTRING
Instruction	OPEN# , CANIO_ADDRESS

13.4. Print & Output Instructions

PRINT -- Information Print

Type	Print Output Function
Description	Print information in output window of RTSys. Additional Name:? Print parameters value through format: *parameters name; Print special character string through format: * character string.
Syntax	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 ?*OPEN: print all OPEN port information
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
Instructions	TRACE

ERRSWITCH -- Information Output Settings

Type	System Parameters
Description	Debugging output switch, control output of TRACE, WARN and ERROR, judge whether debug output commands actually output the information.
Syntax	ERRSWITCH=switch - switch: output information switch 0-no one output 1-output ERROR only 2-output both WARN and ERROR 3-output all instructions (TRACE, WARN and ERROR) 4- output all instructions (TRACE, WARN and ERROR), and monitor motion commands. This is only valid in VPLC5XX controllers.
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 -- Information Print 2

Type	Print Output Function
Description	Print information in output window of RTSys.

	Controlled by ERRSWITCH setting.
Syntax	Same as PRINT
Controller	General
Example	ERRSWITCH = 3 'all trace functions are enabled DPOS(0) =0 TRACE "DPOS(0) =" DPOS(0) Print result 0
Instructions	ERRSWITCH , TRACE , WARN , ERROR

WARN -- Alarm Information

Type	Print Output Function
Description	Print alarm information in output window of RTSys automatically. Manual print is also available. Controlled by ERRSWITCH setting.
Syntax	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 , TRACE , PRINT , ERROR

ERROR -- Error Information

Type	Print Output Function
Description	Print error information in output window of RTSys automatically. Manual print is also available. Controlled by ERRSWITCH setting.
Syntax	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 , TRACE , PRINT , WARN

13.5. Channel Parameter Instructions

PORT -- Channel No.

Type	Channel Correction Subsidiary Instruction
Description	When accesses channel parameters, it can specify needed port No. For other commands, please check them while choosing the channel.

Syntax	PORT (portnum) portnum: channel No. Different controllers support different channel numbers, check it through instruction “?*port”. See example one. ✓ ZMC00x series <table border="1"> <tr> <th>Channel No.</th><th>protocol</th></tr> <tr> <td>0</td><td>Serial port A</td></tr> <tr> <td>1</td><td>Serial port B</td></tr> </table> ✓ ZMC1-2xx series, 2 ethernets can be connected at the same time, the port No. is 502, also it supports connecting to HMI through MODBUS_TCP. <table border="1"> <tr> <th>Channel No.</th><th>protocol</th></tr> <tr> <td>0</td><td>RS232 serial port</td></tr> <tr> <td>1</td><td>RS485 serial port</td></tr> <tr> <td>2</td><td>Ethernet, link 1. port NO.:502</td></tr> <tr> <td>3</td><td>Ethernet, link 2. port NO.:502</td></tr> <tr> <td>10</td><td>Self-defined net communication channel 1</td></tr> <tr> <td>11</td><td>Self-defined net communication channel 2</td></tr> </table> ✓ ZMC3xx series, with 3 serial ports. <table border="1"> <tr> <th>Channel No.</th><th>protocol</th></tr> <tr> <td>0</td><td>RS232 serial port</td></tr> <tr> <td>1</td><td>RS485 serial port</td></tr> <tr> <td>2</td><td>RS422 serial port</td></tr> <tr> <td>3</td><td>Ethernet, link 1.</td></tr> <tr> <td>4</td><td>Ethernet, link 2.</td></tr> <tr> <td>10</td><td>Self-defined net communication channel 1</td></tr> <tr> <td>11</td><td>Self-defined net communication channel 2</td></tr> </table> ✓ ZMC4xx series, with 2 serial ports. <table border="1"> <tr> <th>Channel NO.</th><th>protocol</th></tr> <tr> <td>0</td><td>RS232 serial port</td></tr> <tr> <td>1</td><td>RS485 serial port</td></tr> <tr> <td>2</td><td>Ethernet, link 1.</td></tr> <tr> <td>3</td><td>Ethernet, link 2.</td></tr> <tr> <td>10</td><td>Self-defined net communication channel 1</td></tr> <tr> <td>11</td><td>Self-defined net communication channel 2</td></tr> <tr> <td>20</td><td>Channel for link between controllers, don't support PORT_STATUS</td></tr> </table> ✓ ECI series, only 1 serial port. <table border="1"> <tr> <th>Channel NO.</th><th>protocol</th></tr> </table>	Channel No.	protocol	0	Serial port A	1	Serial port B	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	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	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	Channel NO.	protocol
Channel No.	protocol																																																						
0	Serial port A																																																						
1	Serial port B																																																						
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																																																						
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																																																						
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																																																						
Channel NO.	protocol																																																						

	<table> <tr> <td>0</td><td>RS232 serial port</td></tr> <tr> <td>1</td><td>Ethernet, link 1.</td></tr> <tr> <td>2</td><td>Ethernet, link 2.</td></tr> </table>	0	RS232 serial port	1	Ethernet, link 1.	2	Ethernet, link 2.
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, CONNECT: channels for controller interconnection.						
Instructions	FILE_PORT , PORT_STATUS , PROTOCOL						

PORT_STATUS -- Channel Status

Type	Channel Parameters, only for read.						
Description	Return to present channel status, for serial port, 1 is returned always. When MODBUS_TCP connection is built, PORT_STATUS will become 1. If it is used as slave, controller will search valid ports as MODBUS_TCP master automatically, but it can't know which PORT is used now. This function is valid in controller with new firmware.						
Syntax	VAR1 = PORT_STATUS(port) port: channel No. <table> <tr> <th>Value</th><th>Meaning</th></tr> <tr> <td>0</td><td>No connection</td></tr> <tr> <td>1</td><td>Connected</td></tr> </table>	Value	Meaning	0	No connection	1	Connected
Value	Meaning						
0	No connection						
1	Connected						
Controller	General						

Example	?PORT_STATUS(0) 'return channel status of serial 232, result is 1
Instructions	PORT , SETCOM , ADDRESS

PORT_MODE -- Channel Mode

Type	Channel Parameters
Description	PORT Mode Configurations, it can add PORT type dynamically.
Syntax	Command Syntax: PORT_MODE(portnum, mode [, "comname"]) - portnum: PORT No. - mode: the mode to be set must be an available PORT (ZPORT_MODE_NOTUSED). And it is recommended to set it at the beginning 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 customized c <i>“The current mode setting can be checked by ?*PORT or PORT_MODE”.</i> - 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 Syntax: mode= PORT_MODE(portnum) - portnum: PORT No. - return value: ZPORT_MODE_NOTEXIST = -1, // exceeds the max PORT No. ZPORT_MODE_NOTUSED = 0, // not used, available 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 customized ZPORT_MODE_ETHICONNET = 12, // interconnection channel
Controller	Valid in 5XX series controllers with firmware version above 20200302.
Example	Set customized 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 -- Current Channel File No.

Type	Channel Parameters
Description	Return or set default file No. for current channel. Access to module variables or array of relevant file after setting. This parameter will be called in RTSys automatically, no need to modify it in RTSys.
Syntax	VAR1 = FILE_PORT, FILE_PORT = filenum
Controller	General
Example	>>FILE_PORT = 0 'FILE_PORT parameters of present channel
Instructions	PORT

PROTOCOL -- Channel Communication Protocol

Type	Channel parameter, only for read										
Description	Return communication protocol in present channel.										
Syntax	VAR1 = PROTOCOL(port) ● port: channel No. Returned Value <table border="1"> <thead> <tr> <th>value</th><th>protocol</th></tr> </thead> <tbody> <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> </tbody> </table>	value	protocol	0	RAW data format, no protocol.	3	MODBUS protocol, controller as slave. (default)	14	MODBUS protocol, controller as master.	15	Direct command mode.
value	protocol										
0	RAW data format, no protocol.										
3	MODBUS protocol, controller as slave. (default)										
14	MODBUS protocol, controller as master.										
15	Direct command mode.										
Controller	General										
Instructions	PORT , SETCOM , ADDRESS										

ETH_MODE -- EtherNET Port Mode Settings

Type	Channel parameter.
Description	Net port mode configuration, array type, and each net port is set independently.
Syntax	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	Valid in controllers whose firmware is above 20170720 and with ETH.
Instructions	PORT

SEND_AUTOUP -- Active Report

Type	Channel parameter.
Description	Set the content that is reported actively by net port.
Syntax	SEND_AUTOUP (port, typecode, string) - port: PORT No. 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.
Syntax	SEND_AUTOUP2 (port, typecode, tableindex, length) - port: PORT No. 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: TABLE starting No. - 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 -- Active Reporting Port

Type	Channel parameter.
Description	Check PORT that supports active reporting. Check PORT ports of eth type by entering “?*port”, 0 – unsupported, 1 – support. Recommend to print with index.
Syntax	VALUE = IFAUTOUP_PORT
Controller	General
Example	Check all Ethernet channels. ?*IFAUTOUP_PORT(3)
Instructions	PORT

PORT_KEEPAIVE -- Timeout Detection

Type	Character string command.
Description	Detection of PORT timeout, specifically, detect the time of communication disconnection, after the time limit is exceeded, the communication is completely disconnected and no longer automatically reconnected. Applicable to custom ethernet communication.
Syntax	PORT_KEEPAIVE(iportnum) = value
Controller	ZMC4XX controllers that are with 230315 firmware and above.
Example	OPEN#11,"TCP_SERVER",60000,"192.168.0.11" ?PORT_KEEPAIVE(11) 'print custom ethernet channel 11 timeout time PORT_KEEPAIVE(11) = 5000 'set reference time, when it is out of connection completely, after 5000ms, it will not automatically resume. ?PORT_KEEPAIVE(11)
Instructions	PORT

PORT_485DELAY -- Delay Time of Receive & Send

Type	PORT parameter
Description	While setting the 485 port, the delay time between receiving data to sending data, the unit is us. For some slave devices, maybe they can't switch the direction rapidly, then causing the controller sends too fast to receive data.
Syntax	PORT_485DELAY(iportnum) = value
Controller	General
Instructions	PORT

13.6. MODBUS Communication Instructions

MODBUS_BIT -- Bit Register

Type	MODBUS Bit Register																
Description	Modify or read BIT register, Boolean type, which is called 0x register in HMI. Note: for special Modbus 0x registers, IO status can be read and set directly from the touch screen (original IO status is read, no influence from INVERT_IN). <table border="1"> <thead> <tr> <th>MODBUS_BIT Address</th><th>Descriptions</th></tr> </thead> <tbody> <tr> <td>0~7999</td><td>Customized</td></tr> <tr> <td>8000~8099</td><td>Special M register for PLC programming</td></tr> <tr> <td>8100~8199</td><td>IDLE mark for axis 0-99</td></tr> <tr> <td>8200~8299</td><td>BUFFER remain mark for axis 0-99</td></tr> <tr> <td>10000~14095</td><td>IN</td></tr> <tr> <td>20000~24095</td><td>OUT</td></tr> <tr> <td>30000~34095</td><td>Special S register for PLC programming</td></tr> </tbody> </table>	MODBUS_BIT Address	Descriptions	0~7999	Customized	8000~8099	Special M register for PLC programming	8100~8199	IDLE mark for axis 0-99	8200~8299	BUFFER remain mark for axis 0-99	10000~14095	IN	20000~24095	OUT	30000~34095	Special S register for PLC programming
MODBUS_BIT Address	Descriptions																
0~7999	Customized																
8000~8099	Special M register for PLC programming																
8100~8199	IDLE mark for axis 0-99																
8200~8299	BUFFER remain mark for axis 0-99																
10000~14095	IN																
20000~24095	OUT																
30000~34095	Special S register for PLC programming																
Syntax	MODBUS_BIT(first,[last]) = value ● first: bit register No., starting from 0. ● last: bit register No., starting from 0.																
Controller	General																
Example	DIM VAR MODBUS_BIT(100)=1 'assign bit100 as 1. VAR = MODBUS_BIT(100) 'assign bit100 to variable VAR.																

MODBUS_IEEE -- Word Register (32-Bit Floating)

Type	MODBUS Word Register				
Description	Modify or read word register, 32 bits float, which is called 4x register in HMI. Zmotion motion controller contains MODBUS word register, which will take 2 register addresses. For the relation between registers, please check "MODBUS Register". The controller uses MODBUS word register by default, if needs to modify it, please set the seventh parameter of SETCOM. Note: for special Modbus 4x registers, IO status can be read directly from the touch screen. <table border="1"> <thead> <tr> <th>Register (zero based)</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>10000-</td><td>read DPOS, each axis takes 2 registers.</td></tr> </tbody> </table>	Register (zero based)	Meaning	10000-	read DPOS, each axis takes 2 registers.
Register (zero based)	Meaning				
10000-	read DPOS, each axis takes 2 registers.				

	11000-	read MPOS, each axis takes 2 registers.
	12000-	read VP_SPEED, each axis takes 2 registers.
Syntax	MODBUS_IEEE (regnum) = value ● regnum: register No., starting 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-Bit 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 MODBUS word register, which will take 2 register addresses. For the relation between registers, please check "MODBUS Register". The controller uses MODBUS word register by default, if needs to modify it, please set the seventh parameter of SETCOM.
Syntax	MODBUS_LONG(regnum) = value ● regnum: register No., starting 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-Bit 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 MODBUS word register; For the relation between registers, please check "MODBUS Register". The controller uses MODBUS word register by default, if needs to modify it, please set the seventh parameter of SETCOM. Note: for special Modbus 4x registers, IO status can be read directly from the touch screen. <table border="1"> <thead> <tr> <th>Register (zero based)</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>13000-</td><td>DA, analog output</td></tr> </tbody> </table>	Register (zero based)	Meaning	13000-	DA, analog output
Register (zero based)	Meaning				
13000-	DA, analog output				

	14000-	AD, analog input
Syntax	MODBUS_REG(regnum) = value regnum: register No., starting 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_LONG , MODBUS_IEEE , 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. For the relation between registers, please check " MODBUS Register ".
Syntax	MODBUS_STRING(index, chares) - index: MODBUS register No., starting from 0, register No. will differ as per controller type. - chares: total character string numbers to be 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_LONG , MODBUS_IEEE , MODBUS_REG

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 make the present task overflow, no influence on other tasks. When using 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.
Syntax	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 variables can be modified through SEND.
Controller	General
Example	Example one: multi-master to multi-slave. <pre> 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 slave 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 slave 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 slave 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, </pre>

	<pre> 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 slave station into local station. WAIT UNTIL MODBUSM_STATE <> 1 'wait until message ends. ENDIF ?'channel1 status=", MODBUSM_STATE 'print communication status. ENDIF WEND </pre>
Instructions	PORT , SETCOM , ADDRESS , 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. <pre> >>?*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. </pre> As MODBUS_TCP Master: Choose one PORT of ETH mode as MODBUS_TCP communication channel (the first and last ETH ports are not recommended). If the chosen port is used as slave station, then the controller will choose a valid ETH port as MODBUS_TCP master automatically. As Interconnection Channel Between Controllers: Choose the PORT of CONNECT mode as controller interconnection channel. Note: don't write and read multiple MODBUS slave stations at the same moment, especially when there is multi-task, please operate independently.
Syntax	MODBUSM_DES2 (id, port, "desipaddress", [timer], [resendset], [destport502]) ● id: slave side controller MODBUS ID, default value is 1.

	● port: support two modes, <code>?*PORT</code> to confirm channel No. and mode. Under ETH, as MODBUS_TCP master channel; Under CONNECT, as connection channel between controllers ● desipaddress: IP address of slave controller, it is character string. ● timer: message timeout 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. Please note, if resends MODBUSM, slave controller may receive message twice, and scan the register twice. If resends SEND message, there is no influence on controllers. And significant symbolic variables can be modified through SEND.
Controller	Valid in 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 <code>"?*port"</code> <pre> 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 </pre> 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 <code>"?*port"</code>. <pre> 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 </pre>

	<pre> MODBUSM_REGSET(0,10,0) 'resend if there is error. MODBUSM_REGGET(20,10,20) 'copy slave 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 </pre>
Instructions	PORT , ADDRESS

MODBUSM_STATE -- MODBUS Communication Status

Type	Communication Status										
Description	MODBUS communication status of master station. <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Normal</td></tr> <tr> <td>1</td><td>Waiting for response</td></tr> <tr> <td>2</td><td>Waiting time out</td></tr> <tr> <td>3</td><td>Response error</td></tr> </tbody> </table>	Value	Description	0	Normal	1	Waiting for response	2	Waiting time out	3	Response error
Value	Description										
0	Normal										
1	Waiting for response										
2	Waiting time out										
3	Response error										
Syntax	<pre>VAR1 = MODBUSM_STATE</pre> Present modbus communication status of master station.										
Controller	General										
Example	<pre> 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 </pre>										
Instructions	PORT , SETCOM , ADDRESS										

MODBUSM_REGSET -- Slave Modbus Value Write

Type	Communication Instructions

Description	Assign local save Modbus value to slave station. Relevant standard protocol function code is 06 or 16: write save register.
Syntax	MODBUSM_REGSET (startreg, num, local_reg) ● startreg: modbus 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	PROTOCOL , PORT , SETCOM , ADDRESS , MODBUSM_REGGET

MODBUSM REGGET -- Slave Modbus Value Read

Type	Communication Instructions
Description	Assign save Modbus value of slave station to local station. Relevant standard protocol function code is 03, read save register.
Syntax	MODBUSM_REGGET (startreg, num, local_reg) ● startreg: modbus No. of slave station, starting 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

	<pre> 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 </pre>
Instructions	PROTOCOL , PORT , SETCOM , ADDRESS , MODBUSM_REGSET

MODBUSM_3XGET -- Input Register Read

Type	Communication Instructions
Description	Assign input Modbus value of slave station to local station. Relevant standard protocol function code is 04, read input register.
Syntax	MODBUSM_3XGET (startreg, num, local_reg) ● startreg: modbus No. of slave station, starting 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	PROTOCOL , PORT , SETCOM , ADDRESS

MODBUSM_BITSET -- Slave Coil Write

Type	Communication Instructions
Description	Assign local MODBUS bit register value to slave station. Relevant standard protocol function code is 05 or 15, write coil.
Syntax	MODBUSM_BITSET (startreg, num, local_reg) ● startreg: modbus No. of slave station, starting 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	PROTOCOL , PORT , SETCOM , ADDRESS , MODBUSM_BITGET

MODBUSM_BITGET -- Slave Coil Read

Type	Communication Instructions
Description	Assign MODBUS bit register value of slave station to local station. Relevant standard protocol function code is 01: read coil.
Syntax	MODBUSM_BITGET (startreg, num, local_reg) ● startreg : modbus No. of slave station, starting 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	PROTOCOL , PORT , SETCOM , ADDRESS , MODBUSM_BITSET

MODBUSM_1XGET -- Isolated Inputs Read

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.
Syntax	MODBUSM_1XGET (startreg, num, local_reg) ● startreg : modbus No. of slave station, starting 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	PROTOCOL , PORT , SETCOM , ADDRESS

13.7. Direct Command Instructions between Controllers

SEND_RESULT -- “send” Result Read

Type	Communication instruction
Description	Read “send” instruction result. Return value: 0-succeed, others: errors, including error code. 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 variables can be modified through SEND.

Syntax	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.
Syntax	SEND_CMD(cmdstring, selectable parameter list) ● cmdstring: command character string ● selectable parameter list: numbers can be changed, no need to add the bracket when without parameters
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 (with Axis No.)

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.
Syntax	SEND_CMDAXIS (cmdstring, iaxis, selectable parameter list) ● cmdstring: command character string ● iaxis: the axis numebr ● selectable parameter list: numbers can be changed, no need to add bracket when without parameters.
Controller	Valid in ZMC4XX series controller, version 20170618 support.
Example	SEND_CMDAXIS("MOVE",IAXIS,DIS1)

instruction	SEND_RESULT , SEND_CMD

SEND_ASSIGN -- “send” Command (with Value)

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.	
Syntax	SEND_ASSIGN(cmdstring,value, selectable parameter list) ● cmdstring: command character string ● value: assigned content ● selectable parameter list: numbers can be changed, no need to add bracket when without parameters.	
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 (to TABLE)

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. 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.	
Syntax	SEND_QUERY(cmdstring, selectable parameter list) ● cmdstring: command character string ● selectable parameter list: numbers can be changed, no need to add bracket when without parameters.	
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	<u>SEND_RESULT</u> , <u>SEND_QUERTSET</u>

SEND QUERTSET -- “send” Command (to TABLE)

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.
Syntax	SEND_QUERYSET (dtindex, dtnumes) ● dtindex: TABLE No. Where saves received contents. ● 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. Sending Instructions between Controller Files

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.
Syntax	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_FLASH -- 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.
Syntax	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 -- USB Drive Data Copy

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.
Syntax	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 -- flash Data Copy

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.
Syntax	SEND_IFLASH (dir,id,flashid) - dir: 1-copy USB Drive to controller FLASH, 0-copy controller FLASH to USB Drive - id: FLASH No. of main controller - 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_PERCENT -- Instruction Process Check

Type	Communication instruction
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.
Syntax	percent = SEND_PERCENT ()
Controller	Valid in ZMC4XX series controller, version 20170618 support.
Instruction	SEND_ZAR , SEND_FLASH , SEND_FILE

SEND_CONTROL -- Controller Type Check

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.
Syntax	id = ZAR_CONTROL ("ufilename") ufilename: file name of USB Drive, ZAR file
Controller	Valid in ZMC4XX series controller, version 20170618 support.

Example	<pre>id = ZAR_CONTROL("1.ZAR") IF(id/3000=3) Then 'ZHD series PRINT "zhd program" ENDIF</pre>
Instruction	SEND_ZAR , CONTROL

Chapter 14 System Instructions

All parameters and instructions of date and time can't be modified after locked by 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 that is to be downloaded, if the password doesn't match, then ZAR can't be loaded into controller. APP_PASS can't be modified after LOCK. APP_PASS is encrypted through irreversible algorithm, once forgotten, it can't recover any more.
Syntax	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 -- Controller Lock

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.
Syntax	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 -- Controller Unlock

Type	System Instruction
Description	Unlock Controller. LOCK password is encrypted through Irreversible algorithm, once forgotten, it can not recover any more.
Syntax	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 to the number of days since 1/1/2000. Date can't be modified under simulator.
Syntax	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	DATES , RTC_DATE

DATES -- System Date 2

Type	String Functions
Description	String function, return to date set by DATE in format: DD:MM:YYYY.
Syntax	DATES
Controller	General
Example	DATE=27:2:13 Online command input >>PRINT DATES 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.
Syntax	VAR=DAY, DAY=expression
Controller	General
Example	Example 1: DAY = 3 Example 2: Online command input >>PRINT DAY Output: 3
Instructions	DAYS

DAYS -- System Week 2

Type	String Functions
Description	String Functions, return week set by DAY. DAYS will not change as per DATE or RTC_DATE, they are independent to each other.
Syntax	DAYS
Controller	General
Example	Online command input >>PRINT DAYS Output: Wednesday
Instructions	DAY

RTC_DATE -- System Date 3

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.
Syntax	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 , DATES

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.
Syntax	TIME=hh:mm:ss
Controller	General
Example	Example One: TIME=11:14:40 Example two: Online command input >>PRINT TIME Output: 40541
Instructions	TIMES , RTC_TIME

TIMES -- System Time 2

Type	String Functions
Description	String Functions, return present time in format of 24-hour, hh:mm:ss
Syntax	TIMES
Controller	General

Example	Online command input >>PRINT TIMES\$ 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.
Syntax	RTC_TIME = hhmmss
Controller	General
Example	Online command input >>RTC_TIME=113706 >>PRINT RTC_TIME Output: 113706
Instructions	TIME , TIMES

14.3. Axis System Parameter Instructions

WDOG -- Total Axes Enable

Type	System Parameters
Description	Enable all axes. When using EtherCAT fieldbus, WDOG=1.
Syntax	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.
Syntax	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 No.

Type	System Status
Description	The first bus axis No. that appears error. If return -1 -- no error axes.
Syntax	Var=ERROR_AXIS
Controller	General (bus axis controllers)
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	The list includes all error bus axes. Each bit means one axis, bit0-n indicates axis0-n.
Syntax	Var=MOTION_ERROR
Controller	Bus controllers
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.
Syntax	ERROR_SET (outputs,Modbus register address[,errortime]) errortime: set one SUB process for temporary management, this function will be called when there is pause due to Syntax errors. Don't use instructions that may cause block, such as WAIT, it should be simple. If Syntax 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.
Syntax	RADIUS_ERRSET = mode - mode: 0- default value, no need to check - the radius of start point and end point are different (over 2 pulses), it will correct automatically. - It is inconsistent, return 1006 error.
Controller	General; Valid after 2022.01.11.
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 & 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 to 32-bit integer when reading, see example one. ➤ Modification will take effect immediately, the connection will break after modifying, please reconnect. ➤ Connect the controller to single net, please ensure them in same network segment. ➤ Connect the controller to multi-net, please set as different net segment correspondingly. ➤ When multiple controllers are used, please restart them after modifying IP address.
Syntax	IP_ADDRESS = dot.dot.dot.dot
Controller	General
Example	Online command input <pre>>>IP_ADDRESS=192.168.0.26 >>PRINT IP_ADDRESS</pre> Output :436250816 Conversion process details as follow: Convert four segments to binary type. <pre>192--1100 0000 168--1010 1000 0--0000 0000 26--0001 1010</pre> Reassociation of binary data <pre>26 0 168 192 0001 1010 0000 0000 1010 1000 1100 0000</pre> Convert to decimal data <pre>436250816</pre>
Instructions	IP_GATEWAY , IP_NETMASK

IP_ADDRESS2 -- IP Address 2

Type	System Parameters, which are saved into FLASH automatically

Description	Set the second ethernet IP address for VPLC5XX controllers. ➤ Valid in ZMC5XX series controller, default 2nd Ethernet IP address is 192.168.1.11, return to 32-bit integer when reading. ➤ Modification will take effect immediately, the connection will break after modifying, please reconnect. ➤ Connect the controller to single net, please ensure them in same network segment. ➤ Connect the controller to multi-net, please set as different net segment correspondingly. ➤ When multiple controllers are used, please restart them after modifying IP address.
Syntax	IP_ADDRESS2 = dot.dot.dot.dot
Controller	Valid in ZMC5XX series controllers and above.
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 to 32 bits integer when reading.
Syntax	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 reading.
Syntax	IP_NETMASK=dot.dot.dot.dot
Controller	General
Example	Online command input >>IP_NETMASK =255.255.252.0

	<pre>>>PRINT IP_NETMASK</pre> Output :16580607 Conversion process see IP_ADDRESS for reference.
Instructions	IP_GATEWAY , IP_ADDRESS

IP_IFDHCP -- IP Address Auto-Get

Type	System parameters
Description	Whether to use DHCP, not to use by default. Get IP address automatically. After it is set, RTSys fixed IP is useless, it only can be gained automatically. And please restart it after modification.
Syntax	IP_IFDHCP=1-0 1-get IP automatically 0-use fixed IP
Controller	General
Example	Online command input <pre>>>IP_IFDHCP =1</pre>
Instruction	IP_ADDRESS

IP_IFDHCP2 -- IP Address Auto-Get 2

Type	System parameters
Description	Whether to use DHCP for 2nd Ethernet of VPLC5XX series controllers, not to use by default. Get IP address automatically. After it is set, RTSys fixed IP is useless, it only can be gained automatically. And please restart it after modification.
Syntax	IP_IFDHCP2=1/0 1-get IP automatically 0-use fixed IP
Controller	Valid in ZMC5XX series controllers and above.
Example	Online command input <pre>>>IP_IFDHCP2 =1</pre>
Instruction	IP_ADDRESS

14.5. Controller Information Instructions

VERSION_FPGA -- System FPGA Version

Type	System Status
Description	Get system FPGA version No.
Syntax	VAR1=VERSION_FPGA Note: you also can check it through “State the controller” – SoftVersion”.
Controller	General
Example	?*VERSION_FPGA 'enter it in “output” window to print out FPGA version

	Result: 240104

VERSION_BUILD -- System Firmware Original Date

Type	System Status
Description	Get the date when creates the system firmware.
Syntax	VAR1=VERSION_BUILD Note: you also can check it through “State the controller” – SoftVersion”.
Controller	General
Example	?*VERSION_BUILD 'enter it in “output” window to print out firmware date Result: 20240111

VERSION_DATE -- System Firmware Version

Type	System Status
Description	Get system firmware version. Note: you also can check it through “State the controller” – SoftVersion”.
Syntax	VAR1=VERSION_DATE
Controller	General
Example	?*VERSION_DATE 'enter it in “output” window to print out firmware version Result: 20190305

VERSION -- System Software Version

Type	System Status
Description	System software version No..
Syntax	VAR1 = VERSION Note: you also can check it through “State the controller” – SoftVersion”.
Controller	General
Example	?*VERSION 'enter it in “output” window to print out software version Result: 4.9300

ID_HARDWARE -- Controller Hardware Model

Type	Read-only system parameter.
Description	Get controller hardware type.
Syntax	Val=ID_HARDWARE

	Note: you also can check it through “ State the controller ” – SoftVersion ”.
Controller	General
Example	?*ID_HARDWARE / >>PRINT ID_HARDWARE 'enter it in “output” window to print out the controller model Result: 432
Instructions	CONTROL

CONTROL -- Controller Software Model

Type	Read-only system parameter.
Description	Get controller software model.
Syntax	Val=CONTROL Note: you also can check it through “ State the controller ” – SoftVersion ”.
Controller	General
Example	>>PRINT CONTROL 'enter it in “output” window to print out the controller soft model result: 464
Instructions	ID_HARDWARE

SYSTEM_ZSET -- Controller Settings

Type	System Parameters
Description	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 can be used after bus opened.

	Check whether it is used through ?*ETHERCAT . >>?*ethercat Slot:0 contain 1 nodes. dc:ecat-sensitive .Lostcount:0-0
Syntax	To read: value=SYSTEM_ZSET To write: SYSTEM_ZSET=value
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.
Syntax	LEDOUT (num,state) ● num: indicator light No., 1-RUN, 2-ALM ● state: status, 0-OFF, 1-ON
Controller	General
Example	LEDOUT(1,0) 'turn off RUN indicator LEDOUT (2,0) 'turn off ALM indicator

SERIAL_NUMBER -- Controller Unique ID

Type	Read-only system parameter.
Description	Get the unique ID of controller. It is a unique serial No., generated ZAR file also can be bound with this ID, then this ZAR can only be used in this controller.
Syntax	Var=SERIAL_NUMBER
Controller	General
Example	Example 1 PRINT SERIAL_NUMBER 'print controller ID Print result: 191201941 Example 2: 9-bit ID of controller is stored into VR, due to ZMC below 4xx series VR is single-precision float type, which means it only has 8 valid value, 2 VR spaces can be used. ?SERIAL_NUMBER GLOBAL giA,giB giA = SERIAL_NUMBER MOD 100000 'get remainder VR(0)=giA giB = SERIAL_NUMBER \ 100000 'exact division

	<pre> VR(1)=giB PRINT giA,VR(0) PRINT giB,VR(1) Print result: 191201941 1941 1941 1912 1912 </pre>

SERVO_PERIOD -- System Period

Type	System Parameters
Description	System period. Default value: 1000 microseconds; to modify it after correct firmware is set. Valid Firmware Version (support period configurations): ➤ ZMC 4XX (standard firmware version): 20170713; ➤ ZMC4XX (fast firmware version): 20190106; ➤ VPLC 5XX (firmware version): 20180307; Note: the period that is modified must be within its system period, and after modifying, please restart it. Please set according to below guidance for RTEX drives: ➤ controller period is 500us, set drive P7.20 as 3, set drive P7.21 as 1; ➤ controller period is 1000us, set drive P7.20 as 6, set drive P7.21 as 1; See the example.
Syntax	value = SERVO_PERIOD
Controller	General
Example	Enter command in “output” window to print out controller communication period and RTEX drive setting. <pre> >>PRINT SERVO_PERIOD 'print servo period, result is 1000 >>DRIVE_READ(7*256+20)AXIS(0) 'print Rtex axis 0 drive period setting >>DRIVE_READ(7*256+21)AXIS(0) 'print Rtex axis 0 drive period ratio value setting </pre> Modification Examples: SERVO_PERIOD=500 SERVO_PERIOD=1000
Instructions	SERVO

SYS_ZFEATURE -- System Specification

Type	System parameters

Description	Get system maximum specification.
Syntax	num = SYS_ZFEATURE (code) ● num: returned value ● code: get data type 0- maximum virtual axis amounts 1- how many motors the controller supports 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 included. 8- the number of bus slot 9- the number of FILE 3 (Z3P) 10- the number of serial-ports 11- the number of ethernet ports 12- the number of customized ethernets, 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 registers 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, (include 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 (it is added in 230814)
- 40- ZV max latches
- 41- ZV max tasks recommended
- 42- ZV latch max byte size

- 50- whether supports NC function
- 51- whether supports CANOPEN
- 52- whether the firmware supports robotic arm
- 53- the number of ECAT bus slots
- 54- the number of RTEK 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 (the unit is kbytes)
- 63- Nandflash space (the unit is kbytes)
- 64- Nandflash remaining space (the unit is kbytes)

- 66- platform compiler type
- 67- the bits of pulse
- 68- the bits of TABLE
- 69- the platform pointer length
- 70- whether it supports C language
- 71- whether it supports Anti-Collision Function
- 72- whether it supports DDZ (multi-mover) function

- 74- the max numbers of inside drives
- 75- the real numbers of inside drives
- 76- drive max parameters specification
- 77- the max channels of oscilloscope

- 99- controller function specifications, BIT, 2411 is valid
 - Bit2: cam
 - Bit3: linear interpolation
 - Bit4: circular interpolation
 - Bit5: continuous interpolation
 - Bit10: basic robotic arm
 - Bit11: complex robotic arm
 - Bit12: multi-mover
 - Bit13: CNC
 - Bit14: vision

	101- whether ZMIO is integrated 102- max ZMIO inputs 103- max ZMIO outputs 104- max ZMIO ADs 105- max ZMIO DAs
Controller	Valid in ECI controller with firmware version above 150830 and ZMC4xx series controller with firmware version above 170530.
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. Special inputs include origin, position limit, etc.
Syntax	SYS_IOSET = value ● 0: default mode, it selects according to controller types; ● 1: ZMC mode, normal-OFF (default); ● 2: ECI mode, normal-ON (default);
Controller	General
Example	SYS_IOSET = 1

LASER_SET -- Energy Parallel Output Switch

Type	System parameters
Description	Set the energy parallel output port output commands: AOUT / OP.
Syntax	LASER_SET(isel, value) ● isel: 1 – set whether to use AOUT for energy parallel port; ● value: 1 – enable AOUT output
Controller	VPLC504SCAN
Example	LASER_SET(1,1) ‘use AOUT to set laser energy, now the 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, it will be saved into FLASH, and please restart it. This function is added after version_build 20240719.
Syntax	value = ZML_DEFSHIFT
Controller	General

Example	Online command print >>PRINT ZML_DEFSHIFT

14.6. Log Instructions

Get real time error log – save normal error messages through real-time FIFO, but for some errors that are generated cyclically, they will not be saved.

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).
Syntax	VAL = RTLOG_COUNT()
Controller	Valid in controllers above ZMC4XX series and with firmware above 220907.
Example	?RTLOG_COUNT
Instructions	RTLOG_CLEAR

RTLOG_CLEAR -- Current Logs Clear

Type	Log Instructions
Description	Clear current recorded logs.
Syntax	RTLOG_CLEAR ([number]) number: the number of logs to be cleared, which starts to clear from the first recorded log. The default is -1 (clear all).
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 -- Log Error Message Add

Type	Log Instructions
Description	For application level, system logs can be used to record error messages.
Syntax	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 -- Log Error No. Get

Type	Log Instructions
Description	Get error No. of log.
Syntax	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	<pre>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</pre>
Instructions	RTLOG_CLEAR

RTLOG_TIME\$ -- Log Error Time Get

Type	Log Instructions
Description	Get error time of log.
Syntax	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	<pre>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</pre>
Instructions	RTLOG_CODE

RTLOG_INFO -- Log Error Message Get

Type	Character String Functions
Description	Get error information of logs.
Syntax	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	<pre>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</pre>
Instructions	RTLOG_INFO2

RTLOG_INFO2 -- Log Error Message Get 2

Type	Character String Functions
Description	Get error information of logs, including error No. and error time.
Syntax	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	<pre>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</pre>
Instructions	RTLOG_INFO

?* RTLOG -- Current Recorded Logs Get

Type	Log Instructions
Description	Rapidly print out recent 10 logs.
Syntax	?*RTLOG
Controller	Valid for ZMC4XX controllers and above, also the firmware should be

	above 220907.
Example	<pre> 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 </pre>
Instructions	RTLOG_COUNT

14.7. TABLE Array Instructions

TABLE -- System Default Array

Type	System Array
Description	System default global array can be accessed by all programs. “Data” into TABLE: buffer area, cam data list, screw compensation list, robotic arm parameters, etc.
Syntax	TABLE(index) = value, VAR1 = TABLE(index), TABLE(index [, value1..]) ● index: index No.
Controller	General
Example	TABLE(0)=10 'assign 10 to table(0) TABLE(10,100,200,300) 'table(10) = 100, table(11) = 200, table(12) = 300.
Instructions	TSIZE

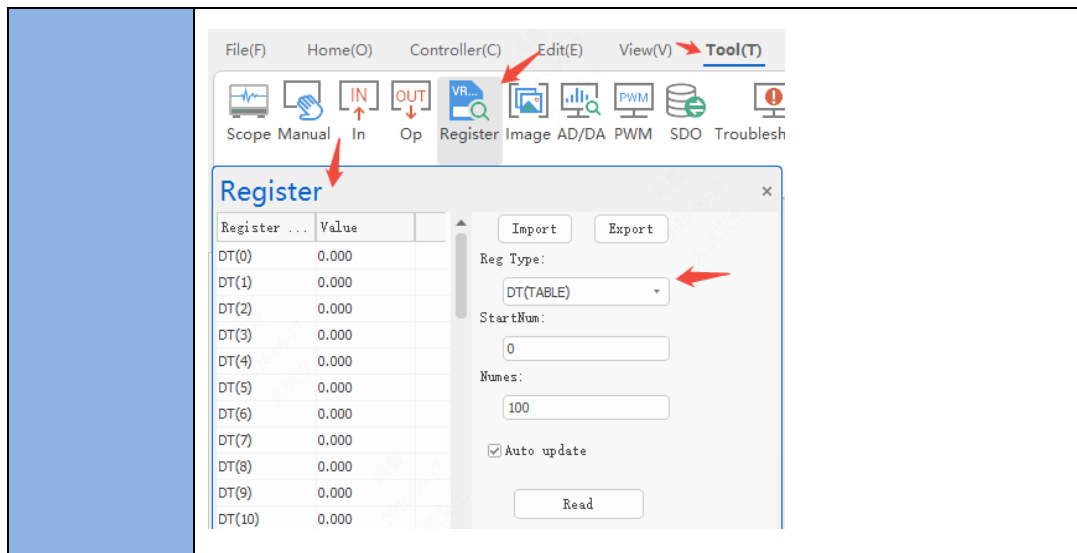
TSIZE -- Table Size

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.
Syntax	Var=TSIZE TSIZE=Value
Controller	General
Example	Read: PRINT TSIZE 'print table size of controller Write: TSIZE=10000 'set table size, don't exceed max size of controller table.
Instructions	TABLE

TABLESTRING -- Table Print (in Character String)

Type	Character string function
Description	Print Table data in character string format. Data will be converted automatically -- printed data is ASCII Code.
Syntax	TABLESTRING(index, length) ● index: initial address of print data ● length: data length to print
Controller	General
Example	Example 1 TABLESTRING(0,5)= "abc" 'save the 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, convert into ASCII Code 'print result: D:\ In RTSys, TABLE each position data can be checked through "view - register" window.



14.8. Oscilloscope Instructions

TRIGGER -- Oscilloscope Trigger

Type	System Instruction
Description	Start to execute data sampling through oscilloscope. When oscilloscope opens continuous acquisition, don't call TRIGGER instruction in Basic, it is recommended to trigger manually.
Syntax	TRIGGER
Controller	Valid in firmware version after 150723.
Example	Example 1 : TRIGGER 'the oscilloscope starts to sample the speed of each motion, then save them in TABLE (storage range: scope – function configuration). MOVE(10000) Example 2: TRIGGER (1) 'trigger continuous acquisition
Instructions	SCOPE

SCOPE -- Data Acquisition

Type	System Instruction
Description	Data acquisition (save into TABLE), max 8 types of data can be sampled at the same time. Use TRIGGER to open automatic sampling, sampling time = sampled period * sampled numbers. It can't use SCOPE and AIN at the same time.

Syntax	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 where 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)) 'capture DPOS and MSPEED every 10ms and store them into TABLE 0~1000, 0~499 for DPOS, 500~1000 for MSPPED, the total sampling time is 1000/2*10=5s TRIGGER 'start sampling MOVE(10000)
Instructions	TRIGGER , SCOPE_POS

SCOPE_POS -- Captured Points

Type	System Instruction
Description	It is only read -- get the number of points sampled by SCOPE currently.
Syntax	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. VR Instructions

CLEAR -- VR Clear

Type	System Instruction
Description	Clear all VR data.
Syntax	CLEAR()
Controller	General
Example	CLEAR() 'clear all data in VR
Instructions	VR

VR -- Power Failure Storage (Float)

Type	System Instruction
Description	Power failure saving type register, 32-bit float. Register numbers differ in different controller models Used to saved float type data, use same space with VR_INT and VR_STRIN.
Syntax	VR(index) = value , VAR1 = VR(index)
Controller	General, but ECI doesn't support power-off storage.
Example	VR(0) = 10 aaa = VR(0) ?aaa Print result: 10.5800
Instructions	VR_INT , VRSTRING

VR_INT -- Power Failure Storage (Integer)

Type	System Instruction
Description	Power failure saving type register, 32-bit integer. Register numbers differ in different controller models Used to saved integer type data, use same space with VR and VR_STRIN.
Syntax	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 -- Power Failure Storage (String)

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.
Syntax	VRSTRING (index[, chares]) ● index: start VR No., starting from 0. ● chares: total character numbers to be read
Controller	General
Example	Online command input <pre>>> VRSTRING (0, 8) = "abc" 'save string.</pre> <pre>>>PRINT VRSTRING (0, 8)</pre> Output: abc
Instructions	VR , VR_INT

Chapter 15 Storage Instructions

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 -- USB File Operations

Type	Files Instructions																						
Description	Upload and search files in controller or in USB drive. Choose functions as per character 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 to use USB3.0 for 4xx controllers or above. VPLC 5 series are Linux systems, the filename (to be read) is case-sensitive, that is, the name must be capital.																						
Syntax	value = FILE "function" ,... <table border="1"> <tr> <td>"LOAD_ZAR"</td><td colspan="2"> FILE"LOAD_ZAR", "filename" Load the upgrade ZAR file from USB drive. ● filename: program file name If upgrade fails, it will return 0 and output reasons through WARN. If it succeeds, ZAR file will be started automatically, which means return value of TRUE is useless.  the breakpoint in original debugging state will be cleared after upgrade is finished. </td></tr> <tr> <td>"LOAD_TCF"</td><td colspan="2"> FILE"LOAD_TCF", "filename", tableindex, maxsize Read unique TCP file. ● filename: file name. ● tableindex: start TABLE No. where saves data. ● maxsize: total TABLE index to save data. <table border="1"> <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>Point Type</td><td>First Point</td></tr> <tr> <td>TABLE101</td><td>X coordinate</td><td></td></tr> </table> </td></tr> </table>		"LOAD_ZAR"	FILE"LOAD_ZAR", "filename" Load the upgrade ZAR file from USB drive. ● filename: program file name If upgrade fails, it will return 0 and output reasons through WARN. If it succeeds, ZAR file will be started automatically, which means return value of TRUE is useless.  the breakpoint in original debugging state will be cleared after upgrade is finished.		"LOAD_TCF"	FILE"LOAD_TCF", "filename", tableindex, maxsize Read unique TCP file. ● filename: file name. ● tableindex: start TABLE No. where saves data. ● maxsize: total TABLE index to save data. <table border="1"> <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>Point Type</td><td>First Point</td></tr> <tr> <td>TABLE101</td><td>X coordinate</td><td></td></tr> </table>		TABLE0	Mark Point		TABLE1	Total Points		TABLE2	Glue Head No.		TABLE100	Point Type	First Point	TABLE101	X coordinate	
"LOAD_ZAR"	FILE"LOAD_ZAR", "filename" Load the upgrade ZAR file from USB drive. ● filename: program file name If upgrade fails, it will return 0 and output reasons through WARN. If it succeeds, ZAR file will be started automatically, which means return value of TRUE is useless.  the breakpoint in original debugging state will be cleared after upgrade is finished.																						
"LOAD_TCF"	FILE"LOAD_TCF", "filename", tableindex, maxsize Read unique TCP file. ● filename: file name. ● tableindex: start TABLE No. where saves data. ● maxsize: total TABLE index to save data. <table border="1"> <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>Point Type</td><td>First Point</td></tr> <tr> <td>TABLE101</td><td>X coordinate</td><td></td></tr> </table>		TABLE0	Mark Point		TABLE1	Total Points		TABLE2	Glue Head No.		TABLE100	Point Type	First Point	TABLE101	X coordinate							
TABLE0	Mark Point																						
TABLE1	Total Points																						
TABLE2	Glue Head No.																						
TABLE100	Point Type	First Point																					
TABLE101	X coordinate																						

		TABLE102	Y coordinate									
		TABLE103	Z coordinate									
		TABLE104	Reserved									
		TABLE105	Point Type	Second Point								
		TABLE106	X coordinate									
		TABLE107	Y coordinate									
		TABLE108	Z coordinate									
		TABLE109	Reserved									
	"LOAD_BYTE"	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: TABLE starting index. ● maxsize: Total TABLE index to save data. ● offset: file bytes offset when it starts to read. <table><tr><td>TABLE0</td><td>Total Bytes</td></tr><tr><td>TABLE1</td><td>1st byte (was read)</td></tr><tr><td>TABLE2</td><td>2nd bytes</td></tr><tr><td>TABLEn</td><td>The Nth byte</td></tr></table>			TABLE0	Total Bytes	TABLE1	1 st byte (was read)	TABLE2	2 nd bytes	TABLEn	The Nth byte
	TABLE0	Total Bytes										
TABLE1	1 st byte (was read)											
TABLE2	2 nd bytes											
TABLEn	The Nth byte											
"FIND_FIRST"	FILE "FIND_FIRST", type, modbus [,dir] Search files in USB drive. Add selectable parameters, dir, character string parameter. ● type:1-file/2-folder/" .extend" file suffix name. ● modbus: modbus_string(modbus) saves the found result (recommended), vr also can be used. ● dir: specify the search path, in character string expression format. When not assign, it will search from root directory of U disk by fault.											
"FIND_NEXT"	FILE "FIND_NEXT", modbus Search the next file in USB drive. ● modbus: modbus_string(modbus) saves the found result (recommended), vr also can be used.											
"FLASH_FIRST"	FILE "FLASH_FIRST", type, modbus Search FLASH file, only BIN, Z3P formats supported. ● type:1-file/2-folder/" .extend" file suffix name. ● modbus: modbus_string(modbus) saves the found result (recommended), vr also can be used.											
"FLASH_NEXT"	FILE "FLASH_NEXT", modbus Search the next flash file. ● modbus: modbus_string(modbus) saves the found result (recommended), vr also can be used.											

	"FLASH_DEL"	FILE "FLASH_DEL", "fileorddir" Delete selected file. ● file: full name of file, and with extension name. <i>It also can delete the folder (with / without symbol drive A, C)</i> Drive C: FLASH (default -- without drive symbol) Drive A: 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, for NAND flash type, CSV is valid. ● filename: full name of file, with extension name.
	"COPY_FROM"	FILE "COPY_FROM", "FLASH file name"[, "file name in USB drive"] <i>Copy flash file to USB drive, support BIN and Z3P file.</i> <i>Rule of FLASH file name: SD block No. BIN</i> 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" <i>It also can copy the folder (with / without symbol drive A, C)</i> Drive C: FLASH (default -- without drive symbol) Drive A: 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 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 ENDIF </pre>	

END

Example 2: find zar procedure

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)

Example 3: FLASH and USB drive data copy each other

```
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
```

Example 4: read/delete USB drive file

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
 00.txt file content: ZMOTION

Read result: the first position saves the number of characters, followings
 save character data in sequence.

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

Example 5:

FILE "FIND_FIRST", 1, 0, "C:\DIR1" 'search DIR1 content of U drive.

U_STATE -- U Disk 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 be loaded automatically, please note it takes effect only when no any program is downloaded into factory controller, which means it is still factory status (valid in 4XX series controllers with 20170423 firmware version or above). If there is running program, please load it 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.
Syntax	Val=U_STATE
Controller	Controllers with USB interface
Example	<pre>?U_STATE 'print USB status If U_STATE = TRUE TEHN 'USB Drive has been inserted. U_READ 1, VAR, ARRAY1, ARRAY2(1) 'read data in USB Drive. ENDIF</pre>
Instructions	U_READ , U_WRITE

U_READ -- U Disk Read (float)

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 takes one float. You could use PC to make file ready at 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.
Syntax	U_READ sect_num,[varname][,arrayname][,arrayname(a)][,arrayname(a,length)] ● sect_num: file No., 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	<pre> 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 </pre>
Instructions	U_WRITE , U_READ2 , U_STATE

U_READDBL -- U Disk Read (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 takes one float. You could use PC to make file ready at 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.
Syntax	<pre> U_READDBL sect_num, [,varname] [,arrayname] [,arrayname(a)] [,arrayname(a,length)] </pre> - sect_num: file No., 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	<pre> 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 </pre>
Instructions	U_WRITE , U_READ2 , U_STATE

U_READ2 -- U Disk Read 2

Type	Storage Instructions
Description	Read data from external memorizer (USB DIRVE) to variable or array,

	and it supports setting the start position when reading file. 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 takes one float. You could use PC to make file ready at 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.
Syntax	U_READ sect_num, star_num [,varname] [,arrayname] [,arrayname(a)] [,arrayname(a,length)] - sect_num: file No., 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	<pre> 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 </pre>
Instructions	U_WRITE , U_READ , U_STATE

U_READ2DBL -- U Disk Read 2 (double)

Type	Storage Instructions
Description	Read data from external memorizer (USB DIRVE) to variable or array, and it supports setting the start position when reading file. 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 takes one float. You could use PC to make file ready at 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.
Syntax	U_READ2DBL sect_num, star_num [,varname] [,arrayname] [,arrayname(a)] [,arrayname(a,length)] sect_num: file No., related to SD 【filenum】 .BIN.

	● star_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 and version above 4xx series, valid in firmware above 20190128.
Example	<pre> 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 </pre>
Instructions	U_WRITE , U_READ , U_STATE

U_READDSB -- DSB File Read

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 controllers below 4xx series, and for 4xx series and above support USB3.0.																																																				
Syntax	U_READDSB "filename", tableindex, maxsize [, flag] ● filename: file name ● tableIndex: TABLE starting position for storage ● maxsize: the total number of TABLE size for storage ● flag: reserved. <table border="1"> <thead> <tr> <th>Number</th><th>Contents</th><th></th></tr> </thead> <tbody> <tr> <td>0</td><td>Mark bit: the number of variables in one line. (reserved)</td><td></td></tr> <tr> <td>1</td><td>Point & Path numbers</td><td></td></tr> <tr> <td>2--29</td><td>File name, 28 characters</td><td></td></tr> <tr> <td>30</td><td>times of colors changing</td><td></td></tr> <tr> <td>31</td><td>+X:</td><td></td></tr> <tr> <td>32</td><td>-X:</td><td></td></tr> <tr> <td>33</td><td>+Y:</td><td></td></tr> <tr> <td>34</td><td>-Y:</td><td></td></tr> <tr> <td>35</td><td>AX: +</td><td></td></tr> <tr> <td>36</td><td>AY: -</td><td></td></tr> <tr> <td>37</td><td>MX: +</td><td></td></tr> <tr> <td>38</td><td>MY: +</td><td></td></tr> <tr> <td>39</td><td>PD:</td><td></td></tr> <tr> <td>40</td><td>L_limit</td><td>coordinate of left limit</td></tr> <tr> <td>41</td><td>R_limit</td><td>coordinate of right limit</td></tr> <tr> <td>42</td><td>U_limit</td><td>coordinate of up limit</td></tr> </tbody> </table>		Number	Contents		0	Mark bit: the number of variables in one line. (reserved)		1	Point & Path numbers		2--29	File name, 28 characters		30	times of colors changing		31	+X:		32	-X:		33	+Y:		34	-Y:		35	AX: +		36	AY: -		37	MX: +		38	MY: +		39	PD:		40	L_limit	coordinate of left limit	41	R_limit	coordinate of right limit	42	U_limit	coordinate of up limit
Number	Contents																																																				
0	Mark bit: the number of variables in one line. (reserved)																																																				
1	Point & Path numbers																																																				
2--29	File name, 28 characters																																																				
30	times of colors changing																																																				
31	+X:																																																				
32	-X:																																																				
33	+Y:																																																				
34	-Y:																																																				
35	AX: +																																																				
36	AY: -																																																				
37	MX: +																																																				
38	MY: +																																																				
39	PD:																																																				
40	L_limit	coordinate of left limit																																																			
41	R_limit	coordinate of right limit																																																			
42	U_limit	coordinate of up limit																																																			

	43	D_limit	coordinate of lower limit
	99	Total lines have been read	
	100	Line Type	First point
	101	parameter1: X relative distance	
	102	parameter2: Y relative distance	
	103	parameter3: X coordinate relates to initial point	
	103	parameter4: Y coordinate relates to initial point	
	105	Line Type	Second point
	106	parameter1: X relative distance	
	107	parameter2: Y relative distance	
Controller	Controllers with USB interface		
Instructions	U_WRITE , U_READ , U_STATE		

U_WRITE -- U Disk Write (float)

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 takes one float. You could use PC to make file ready at 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.
Syntax	U_WRITE sect_num,[varname][,arrayname][,arrayname(a)][,arrayname(a,length)] ● sect_num: file No., 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 numbers to be operated
Controller	Controllers with USB interface
Example	<pre>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</pre>
Instructions	U_READ , U_STATE

U_WRITEDBL -- U Disk Write (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_WRTEDBL 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 takes one float. You could use PC to make file ready at 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.
Syntax	U_WRITEDBL sect_num, [,varname] [,arrayname] [,arrayname(a)] [,arrayname(a,length)] ● sect_num: file No., 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 numbers to be operated
Controller	Controllers with USB interface and version above 4xx series, valid in firmware above 20190128.
Example	<pre> 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 </pre>
Instructions	U_WRITE , U_READ , U_STATE

U_WRITE2 -- U Disk Write 2

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 memory interface. File is saved as 32 bits ieee float type in sequence, one variable or one array element takes one float. You could use PC to make file ready at first, then use U_READ to read. It is recommended to use USB2.0 for controllres below ZMC4xx series, and for 4xx series and above support USB3.0.

Syntax	U_WRITE 2 sect_num,start_num[,varname][[,arrayname][[,arrayname(a)][[,arrayname(a,length)]] - sect_num: file No., related to SD 【filenum】 .BIN. - start_num: the file reading starting position - varname: variable name - arrayname: array name, can be TABLE, VR. - a: index to operate in array - length: array elements numbers to be operated
Controller	Controllers with USB interface
Example	If U_STATE = TRUE THEN 'USB drive was already inserted. U_WRITE2 0, TABLE(0,10) 'write TBALE 0-10 data into USB SD0 ENDIF
Instructions	U_READ , U_STATE

STICK_READ -- U Disk Read (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.						
Syntax	value = STICK_READ (filenum, table_start [,format]) - filenum: file name, relates to SD 【filenum】 - table_start: TABLE starting position - format: file format that will get from external memorizer. <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0(default)</td><td>Float format, .BIN</td></tr> <tr> <td>1</td><td>Text format, .CSV</td></tr> </tbody> </table>	Value	Description	0(default)	Float format, .BIN	1	Text format, .CSV
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 -- U Disk Write (from Table)

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 controllers below 4xx series, and for 4xx series and above support USB3.0.						
Syntax	value = STICK_WRITE(filename, table_start [,length [,format]]) ● filename: file name, relates to SD 【filename】 , max is 9999 ● table_start: TABLE starting position ● length: TABLE data numbers to be operated, default value is 128. ● format: file format that will be written into memorizer. <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0(default)</td><td>Float format, .BIN</td></tr> <tr> <td>1</td><td>Text format, .CSV</td></tr> </tbody> </table>	Value	Description	0(default)	Float format, .BIN	1	Text format, .CSV
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 -- U Disk Read (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 controllers below 4xx series, and for 4xx series and above support USB3.0.						
Syntax	value = STICK_READVR (filename, vr_start [,format]) ● filename: file name, relates to SD 【filename】 ● table_start: VR starting No. ● format: file format that will get from external memorizer. <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0(default)</td><td>Float format, .BIN</td></tr> <tr> <td>1</td><td>Text format, .CSV</td></tr> </tbody> </table>	Value	Description	0(default)	Float format, .BIN	1	Text format, .CSV
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						

DISK_WRITEVR -- U Disk Write (from VR)

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 controllers below 4xx series, and for 4xx series and above support USB3.0.						
Syntax	value = STICK_READVR (filenum, vr_start [,format]) ● filenum: file name, relates to SD [filenum] ● table_start: VR starting position ● length: VR data number to operate, default value is 128. ● forma: file format that will be written into external memorizer. <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0(default)</td><td>Float format, .BIN</td></tr> <tr> <td>1</td><td>Text format, .CSV</td></tr> </tbody> </table>	Value	Description	0(default)	Float format, .BIN	1	Text format, .CSV
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						

DISK_CONFIG -- ZHD HMI “U Drive”

Type	Command of “operation”
Description	Operate teach pendant’s U drive in the motion controller side. While switching the disk, it will automatically close all opened files of original drive.
Syntax	DISK_CONFIG disknum, "NONE" 'turn off the disk DISK_CONFIG disknum, "DEFAULT" 'restore default configuration DISK_CONFIG disknum, "LCD", iconnectport 'map as current LCD disk, it must support ZHD firmware. ● Disknum: the controller disk No., 0 – disk A, 1 – disk B. ● LCD: HMI No. ● Iconnectport: use interconnection PORT to do communication, recommend use channel 21, because it will not conflict with traditional channel 20. “?*PORT” can check PORT list. Current disk information can be checked by “?*DISK”.
Controller	VPLC5XX, the firmware should be version_build250518 and above.

15.2. FLASH Instructions

FLASH_WRITE -- Flash Write (float)

Type	Storage Instructions
Description	Store variables or arrays, single element or some elements in array are saved into flash, support power failure storage. Data is saved into Flash in sequence, then the sequence of reading should be same as sequence of writing. Storage times are limited in Flash, don't make cycle operations randomly. Don't operate FLASH while moving, or it will affect motion execution.
Syntax	FLASH_WRITE sect_num [, varname] [, arrayname] [, arrayname(a)] [, arrayname(a,length)] - sect_num: FLASH block No. (depend on controller type and model) - varname: the variable' name - arrayname: the array' name, TABLE, VR, MODBUS - a: array index that is to be operated - length: how many array elements to be operated
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 uses "float" precision, for 32 bits integer data, it should use 2 MODBUS_REG to store MODBUS_LONG. MODBUS_LONG(1)=123456 'use MODBUS_REG(1) and MODBUS_REG(2) FLASH_WRITE 1, MODBUS_REG(1,2) 'select 2 elements from MODBUS_REG (1), write them into FLASH block (same 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 -- Flash Write (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. Data is saved into Flash in sequence, then the sequence of reading should be same as sequence of writing. Storage times are limited in Flash, don't make cycle operations randomly. Don't operate FLASH while moving, or it will affect motion execution.
Syntax	FLASH_WRITEDBL sect_num [, varname] [, arrayname] [, arrayname(a)] [, arrayname(a,length)] - sect_num: FLASH block No. (depend on controller type and model) - varname: the variable' name - arrayname: the array' name, TABLE, VR, MODBUS - a: array index that is to be operated - length: how many array elements to be operated
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_WRITE2 -- Flash Write 2

Type	Storage Instructions
Description	Store variables or arrays, single element or some elements in array are saved into flash, support power failure storage. Data is saved into Flash in sequence, then the sequence of reading should be same as sequence of writing. Storage times are limited in Flash, don't make cycle operations randomly. Don't operate FLASH while moving, or it will affect motion execution. ➤ The data must be written once completely at first. ➤ The starting position can be calculated by SIZEOFARRAY.
Syntax	FLASH_WRITE2 sect_num start_num [, varname] [, arrayname] [, arrayname(a)] [, arrayname(a,length)] - sect_num: FLASH block No. (depend on controller type and model) - start_num : the position where starts to write - varname: the variable' name - arrayname: the array' name, TABLE, VR - a: array index that is to be operated - length: how many array elements to be operated

Controller	General
Example	FLASH_WRITE2 1, TABLE(0,100) 'write into FLASH FLASH_WRITE2 1, 10, TABLE(10,10) 'write middle part data into FLASH
Instructions	FLASHVR , FLASH_READ

FLASH_READ -- Flash Read

Type	Storage Instructions
Description	Read data from internal Flash to variable or array. Data is saved into Flash in sequence, then the sequence of reading should be same as sequence of writing. When the unwritten FLASH is read, it will show message (but no influence): Warn file: "BASIC1.BAS" line:5 task:0, File:C\SD10.BIN open error, not load. Don't operate FLASH while moving, or it will affect motion execution. ➤ For Linux controller, please write correct bin file suffix name, it must be "BIN", can't be "bin". But for ZMC controllers, both are OK.
Syntax	FLASH_READ sect_num start_num [, varname] [, arrayname] [, arrayname(a)] [, arrayname(a,length)] ● sect_num: FLASH block No. (depend on controller type and model) ● varname: the variable' name ● arrayname: the array' name, TABLE, VR ● a: array index that is to be operated ● length: how many array elements to be operated
Controller	General
Example	FLASH_READ 1, VAR, ARRAY1, ARRAY2(1) 'read FLASH 1 data, and save into VAR, ARRAY1, ARRAY2(1) in order.
Instructions	FLASH_READDBL , FLASHVR , FLASH_WRITE

FLASH_READDBL -- Flash Read (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. Data is saved into Flash in sequence, then the sequence of reading should be same as sequence of writing. When the unwritten FLASH is read, it will show message (but no influence): Warn file: "BASIC1.BAS" line:5 task:0, File:C\SD10.BIN open error, not load. Don't operate FLASH while moving, or it will affect motion execution.

Syntax	FLASH_READDBL sect_num start_num [, varname] [, arrayname] [, arrayname(a)] [, arrayname(a,length)] ● sect_num: FLASH block No. (depend on controller type and model) ● varname: the variable' name ● arrayname: the array' name, TABLE, VR ● a: array index that is to be operated ● length: how many array elements to be operated
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	FLASH_READ , FLASHVR , FLASH_WRITE

LASH_READ2 -- Flash Read 2

Type	Storage Instructions
Description	Read data from internal Flash to variable or array. Data is saved into Flash in sequence, then the sequence of reading should be same as sequence of writing. When the unwritten FLASH is read, it will show message (but no influence): Warn file: "BASIC1.BAS" line:5 task:0, File:C\SD10.BIN open error, not load. Don't operate FLASH while moving, or it will affect motion execution.
Syntax	FLASH_READ2 sect_num start_num [, varname] [, arrayname] [, arrayname(a)] [, arrayname(a,length)] ● sect_num: FLASH block No. (depend on controller type and model) ● start_num : the position where starts to read ● varname: the variable' name ● arrayname: the array' name, TABLE, VR ● a: array index that is to be operated ● length: how many array elements to be operated
Controller	General
Example	<pre> 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) </pre>

	<pre>'read data starting from table(2) in sequence, put 5 data to table(4) - table (8) FOR i = 0 TO 11 ? "TABLE", i, TABLE(i) NEXT END</pre>
Instructions	FLASH_READ , FLASH_WRITE

FLASH_READ2DBL -- Flash Read 2 (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_READ2DBL saves double type with 64-bit. Data is saved into Flash in sequence, then the sequence of reading should be same as sequence of writing. When the unwritten FLASH is read, it will show message (but no influence): Warn file: "BASIC1.BAS" line:5 task:0, File:C\SD10.BIN open error, not load. Don't operate FLASH while moving, or it will affect motion execution.
Syntax	<pre>FLASH_READ2DBL sect_num start_num [, varname] [, arrayname] [, arrayname(a)] [, arrayname(a,length)]</pre> ● sect_num: FLASH block No. (depend on controller type and model) ● start_num : the position where starts to read ● varname: the variable' name ● arrayname: the array' name, TABLE, VR ● a: array index that is to be operated ● length: how many array elements to be operated
Controller	Valid in ZMC4XX series controllers with firmware 20190128 or above.
Example	<pre>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, put 5 data to table(4) - table (8) FOR i = 0 TO 11 ? "TABLE", i, TABLE(i)</pre>

	NEXT END
Instructions	FLASH_READDBL , FLASH_READ2 , FLASH_WRITE

FLASHVR -- RAM Data Copy

Type	Storage Instructions				
Description	Copy data from RAM to FLASH. ➤ ZMC00X (except 005) & ECI Series: TABLE data are saved into the last FLASH block, and FLASH_WRITE & FLASH_READ also can be used to operate this block, don't make conflict. ➤ Other Series TABLE data are saved in an independent area, which can't be operated. Don't operate FLASH while moving, or it will affect motion execution.				
Syntax	FLASHVR (function) function: select function <table border="1"> <tr> <td>-1</td><td>Save all TABLE into FLASH. and read data to TABLE automatically when powered on.</td></tr> <tr> <td>-2</td><td>Cancel function: auto-read data to TABLE when power on.</td></tr> </table>	-1	Save all TABLE into FLASH. and read data to TABLE automatically when powered on.	-2	Cancel function: auto-read data to TABLE when power on.
-1	Save all TABLE into FLASH. and read data to TABLE automatically when powered on.				
-2	Cancel function: auto-read data to TABLE when power on.				
Controller	General				
Example	FLASHVR (-1) 'save data from table into FLASH, read flash data to table when power on.				
Instructions	FLASH_WRITE				

FLASH_SECTSIZE -- Variable Numbers in Flash

Type	System Status Functions
Description	Read how many variables that can be stored into one FLASH block. Depend on controller models.
Syntax	value = FLASH_SECTSIZE
Controller	General
Example	? FLASH_SECTSIZE print result 20480 'ZMC1xx series 1024 'ZMC00x series ...
Instructions	FLASH_SECTES

FLASH_SECTES -- Flash Block Numbers

Type	System Status Functions
Description	Read the total numbers of FLASH inside the controller. Depend on controller models. For ZMC00X series, FLASHVR will use the last block to avoid conflict.
Syntax	value = FLASH_SECTES
Controller	General
Example	<pre>?FLASH_SECTES 'print out flash block numbers print result 128 'ZMC005</pre>
Instructions	FLASH_SECTSIZE

15.3. File Storage Instructions

FILE_ZOPEN -- File Open

Type	Storage Instructions
Description	According to the mode assigned by “mode” parameter, open one file, then return file handle.
Syntax	Function Syntax: 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 customize 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 controllers with firmware version above 20170601.
Example	Example 1: <pre>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</pre>

	<pre>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)</pre>
Instructions	FILE_ZCLOSE

FILE_ZCLOSE -- File Close

Type	Storage Instructions
Description	Close file handle “handle” indicated file, if the handle is invalid, warn will output.
Syntax	Command Syntax: FILE_ZCLOSE (handle) ● handle: returned file handle of function “FILE_ZOPEN”.
Controller	Valid in ZMC4XX controllers with firmware version above 20170601.
Example	<pre>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)</pre>
Instructions	FILE_ZOPEN

FILE_ZWRITES -- File Write (Character String)

Type	Storage Instructions
Description	Write character string “string” into file handle “handle” assigned file, and the function returns to the number of wrote characters.
Syntax	Command Syntax: FILE_ZWRITES (handle, string) Function Syntax: num = FILE_ZWRITES (handle, string) ● handle: returned file handle of function “FILE_ZOPEN”. ● string: normal character string to be written

Controller	Valid in ZMC4XX controllers with firmware version above 20170601.
Example	<pre> 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) </pre>
Instructions	FILE_ZOPEN , FILE_ZCLOSE , FILE_ZSEEK , FILE_ZREAD .

FILE_ZWRITE -- File Write (Character)

Type	Storage Instructions
Description	Write “count” characters 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 writing in. In addition, there is the limit for character writing number, if it exceeds, write the max number into.
Syntax	Command Syntax: FILE_ZWRITE (handle, tableindex, count) Function Syntax: num = FILE_ZWRITE (handle, tableindex, count) ● handle: returned file handle of function “FILE_ZOPEN”. ● tableindex: TABLE index, put the character that is to be written to starting table of tableindex, each character is saved into each TABLE position. ● count: the number of characters that is to be written.
Controller	Valid in ZMC4XX controllers with firmware version above 20170601.
Example	<pre> 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) </pre>
Instructions	FILE_ZOPEN , FILE_ZCLOSE

FILE_ZREAD -- File Read (Character)

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 to the number of reading characters, also, there is the limit for reading number, if it exceeds, according to the max number.
Syntax	Command Syntax: FILE_ZREAD (handle, tableindex, maxchars) Function Syntax: 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 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 Read

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. It starts to read 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 to the number of characters read. The number of reads has a maximum limit.
Syntax	Command Syntax: FILE_ZREAD LINE (handle, tableindex, maxchars) Function Syntax: 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 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 <pre>num = FILE_ZREADLINE(handle, 0, 1000)</pre> 'has reached the end of the line, return 0 <pre>?num, table(0) '0 0</pre> <pre>?FILE_ZTELLLINE(handle)</pre> 'the line number read above remains <pre>FILE_ZSEEKLINE(handle, -1, 2)</pre> 'the beginning of the penultimate line <pre>num = FILE_ZREADLINE(handle, 0, 1000)</pre> 'read the penultimate line, locate to the beginning of the penultimate line <pre>FILE_ZSEEKLINE(handle, -1, 1)</pre> 'the beginning of the previous line, here is the beginning of the penultimate line <pre>FILE_ZSEEKLINE(handle, 17, 0)</pre> 'locate to line number 17 (starting from 0), which is the beginning of line 18 <pre>FILE_ZSEEKLINE(handle, 0, 1)</pre> 'move to the beginning of this line <pre>num = FILE_ZREADLINE(handle, 0, 1000)</pre> <pre>FILE_ZCLOSE(handle)</pre>
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.
Syntax	Command Syntax: 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 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.
Syntax	Command Syntax: 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 controllers with firmware version above 20170601.
Example	Refer to FILE_ZREADLINE routine.
Instructions	FILE_ZREADLINE , FILE_ZTELLLINE

FILE_ZTELL -- File Read & Write Position

Type	Storage Instructions
Description	Return the current reading and writing position of “handle” file, which starts from 0.
Syntax	Function Syntax: pos = FILE_ZTELL (handle) ● handle: the file handle returned by the function FILE_ZOPEN
Controller	Valid in ZMC4XX controllers with firmware version above 20170601.

Example	<pre> 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) </pre>
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.
Syntax	Function Syntax: pos = FILE_ZTELL (handle) ● handle: the file handle returned by the function FILE_ZOPEN
Controller	Valid in ZMC4XX controllers with firmware version above 20170601.
Example	Refer to FILE_ZREADLINE routine.
Instructions	FILE_ZREADLINE , FILE_ZSEEKLINE

Chapter 16 Interrupt Instructions

16.1. Three Interrupt Instructions

INT_ENABLE -- Interrupt Main Switch

Type	System Parameters												
Description	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 it deals with too much functions, especially with longer code, it will cause all interruption responses become slower, even cause interruption overflow, then influence other interruption execution. Solutions: (1) Decrease interrupts, because many applications can be handled in cycle. (2) For an extreme long function, call a single task to deal with the complicated interruption task, avoid blocking other interruption responses.												
Syntax	INT_ENABLE=switch <table><tr><th>Value</th><th>Description</th></tr><tr><td>0 (Default)</td><td>OFF</td></tr><tr><td>1</td><td>ON</td></tr></table>	Value	Description	0 (Default)	OFF	1	ON						
Value	Description												
0 (Default)	OFF												
1	ON												
Controller	General												
Example	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. <pre>1 INT_ENABLE=1 ' 开启中断 2 TIMER_START(0,1000) ' 定时器0开启, 1000ms后执行一次 3 TIMER_START(1,1100) ' 定时器1开启, 1100ms后执行一次 4 5 END 6 7 GLOBAL SUB ONTIMER0() ' 中断处理函数 8 DELAY 1000 ' 假设大量的堵塞性代码 9 WAIT UNTIL IN(0) <> 0 ' 第一个中断 10 11 END SUB 12 13 GLOBAL SUB ONTIMER1() ' 中断处理函数 14 ' 第二个中断 15 END SUB</pre> <table><tr><th>任务</th><th>状态</th><th>文件和行号</th></tr><tr><td>0</td><td>Stopped</td><td>TICKS.BAS,line:7</td></tr><tr><td>38</td><td>Running</td><td>TICKS.BAS,line:9</td></tr></table> 中断任务 <table><tr><th>栈</th><th>过程</th><th>文件和行号</th></tr></table>	任务	状态	文件和行号	0	Stopped	TICKS.BAS,line:7	38	Running	TICKS.BAS,line:9	栈	过程	文件和行号
任务	状态	文件和行号											
0	Stopped	TICKS.BAS,line:7											
38	Running	TICKS.BAS,line:9											
栈	过程	文件和行号											
Correct demonstration:	When there needs to deal with a lot of codes, build a task in the interrupt, as follow task 3, execute below procedures, print “the second interrupt”, timer												

interrupt 0 blocks, this has no influence on timer interrupt 1.

```

1  INT_ENABLE=1          '开启中断
2  TIMER_START(0,1000)    '定时器0开启, 1000ms后执行一次
3  TIMER_START(1,1100)    '定时器1开启, 1100ms后执行一次
4
5  END
6
7  GLOBAL SUB ONTIMER0() '中断处理函数
8  '创建一个新任务来处理自己的复杂任务
9  '就不会堵塞其他中断的响应速度
10
11  RUNTASK 3, MyIntHandler()
12  END SUB
13
14  GLOBAL SUB MyIntHandler()
15  DELAY 1000 '假设大量的堵塞性代码
16  WAIT UNTIL IN(0) <> 0
17  ?"第一个中断"
18  END SUB
19
20  GLOBAL SUB ONTIMER1() '中断处理函数
21  ?"第二个中断"
22  END SUB
23

```

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

```

Instructions [ONPOWEROFF](#), [ONTIMERn](#)

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.
Syntax	<pre>GLOBAL ONPOWEROFF()</pre> ...

	END SUB
Controller	General
Example	<pre> 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 </pre>
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. <i>It is valid when the firmware that supports PLC function.</i> <i>Interrupt Inputs: IN0-IN31. For ZMC4XX series, IN0-IN63.</i>
Syntax	<pre> GLOBAL SUB INT_ONn() n is input No. ... END SUB </pre>
Controller	Controller with firmware that supports PLC
Example	<pre> INT_ENABLE=1 'open interrupt END GLOBAL SUB INT_ON0() 'interrupt procedure print "triggered when meets rising edge of IN0" END SUB </pre>
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. <i>It is valid when the firmware that supports PLC function.</i> <i>Interrupt Inputs: IN0-IN31. For ZMC4XX series, IN0-IN63.</i>
Syntax	<pre> GLOBAL SUB INT_OFFn() n is input NO. ... END SUB </pre>
Controller	Controller with firmware that supports PLC

Example	<pre> INT_ENABLE=1 'open interruption END GLOBAL SUB INT_OFF0() 'interrupt procedure print "triggered when meets falling edge of IN0" END SUB </pre>
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 RTSys software, remote command send “?*max” see max_timer.
Syntax	<pre> GLOBAL SUB ONTIMERn() n is timer No. ... END SUB </pre>
Controller	General
Example	<pre> 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 </pre>
Instructions	INT_ENABLE , TIMER_START

INT_CYCLE -- Interrupt Period Execute

Type	System parameters
Description	BASIC Interrupt period execution, each SERVO_PERIOD executes once.
Syntax	Command Syntax: INT_CYCLE(function, taskid [, subname]) ● 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 Syntax: var = INT_CYCLE(function, taskid) ● 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	Valid in ZMC4XX series controller and version above 20170630.
Example	<pre> DIM times INT_CYCLE(1,1,intisr) END GLOBAL SUB intisr() times=times+1 MOVE_PT(1,1) 'run in every period END SUB </pre>
Instruction	INT_ENABLE

16.2. Timer Instructions

TIMER_IFEND -- Timer Status

Type	System Functions
Description	Check if timer ends through the returned value.
Syntax	value=TIMER_IFEND (timernum) Returned value is 0: timer is in process, timer interrupt is not executed. Returned value is 1: timer ends, timer interrupt starts to execute. For PLC firmware, print 0 before opening the timer; For non-PLC firmware, print 1 before opening the timer.
Controller	General
Example	<pre> 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. </pre>
Instructions	ONTIMERn , TIMER_START

TIMER_START -- Timer ON

Type	System Instruction
Description	Start system timer, only execute 1 time.
Syntax	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 <code>TIMER_IFEND</code>
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.
Syntax	value = <code>TIMER_COUNT</code> (timernum) timernum: timer No., 0 – the number of timers reduce 1
Controller	General
Example	<pre> 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 </pre>
Instructions	ONTIMERn , TIMER_START

TIMER_STOP -- Timer OFF

Type	System Instruction
Description	Force to stop system timer.
Syntax	<code>TIMER_STOP</code> (timernum) timernum: No., 0-(timer number-1)
Controller	General
Example	<pre> 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 </pre>
Instructions	ONTIMERn , TIMER_IFEND

Chapter 17 Fieldbus Instructions

17.1. No. Descriptions

Slot No.

Slot No. means the interface No. 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, starting from 0, increase as connection order.

Drive No. is different from Device No., if 3 devices are connected to controller, the first 2 are IO expansion modules, the last is drive, then now the device No. of drive node=2, the drive No. is 0.

17.2. Basic Instructions

SLOT_SCAN -- Bus Scan

Type	Field Bus Instructions
Description	Scan Field Bus.

	Use RETURN to check if scanning is done successfully, if it succeeds, the returned value is -1, otherwise, it is 0. In terms of Rtex, it will report errors if it 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 scanning, use NODE to read information of connected devices, and use DRIVE instructions to configure.
Syntax	SLOT_SCAN (slot) slot: Slot No. of EtherCAT or RTEX, 0-default
Controller	Controllers with EtherCAT or Rtex.
Example	<pre> 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 </pre>
Instructions	SLOT_START , SLOT_STOP

SLOT_START -- Field Bus ON

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.
Syntax	SLOT_START (slot [,opstate]) - slot: slot No. of EtherCAT or RTEX, 0-default. - opstate: EtherCAT status, 4-SAFEOP, 8-OP(default). NODE_PDOBUFF can be modified after started, it can start 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	<pre> 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. </pre>

	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_PDOBUF

SLOT_STOP -- Field Bus OFF

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.
Syntax	SLOT_STOP (slot) ● slot: slot No. of EtherCAT or RTEX, 0-default.
Controller	Controllers with EtherCAT or Rtex.
Example	<pre> aa: SLOT_SCAN(0) 'scan slot 0. IF RETURN THEN 'start axis configuration if scan succeeds 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. 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 </pre>
Instructions	SLOT_START , SLOT_SCAN

SLOT_INFO -- Bus Information Get

Type	EtherCAT Command
Description	Read bus information It reads after scanning the bus.

Syntax	Read-Only: var = SLOT_INFO (slot, sel)	
	- slot: slot No., default is 0 - sel: information No.	
	Value	Description
	0	slot type: SLOT_TYPE_NULL = 0, // invalid SLOT_TYPE_ECAT = 1, // ECAT SLOT_TYPE_RTEX = 4, // RTEX SLOT_TYPE_HLINK = 5, // Huawei HLINK
	4	Read AL state: 1--Init—initialization state 2--pre-operational—pre-operation state 4--safe-operational—safe operation state 8--operational—running state (enable)
	5	Read whether disconnect is detected, it is used for redundancy mode, 0529 adds. 0—normal, 1--break
	14	How many bytes of PDO sending
	15	How many bytes of PDO receiving
	16	How many devices in total, it must scan at first, then read
	17	How many motors in total, it must scan at first, then read
Controller	VERSION_BUILD: 240529 above version	
Instructions	SLOT_SLAVE	

?*SLOT -- Field Bus Port Print

Type	EtherCAT Subsidiary Instruction
Description	Check Field Bus Port No. and Type.
Syntax	?*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 -- EtherCAT Bus Status Print

Type	EtherCAT Subsidiary Instruction
Description	Use this instruction while debugging to get status of devices connected. Only valid after a successful field bus scan.
Syntax	?*ETHERCAT
Controller	Controllers with EtherCAT

Example	?*ETHERCAT Result as follow: <pre>Slot:0 contain 1 nodes. Lostcount:0-0. Node:0 status:1 manid:7595h productid:0h axes:1 Alstate:8 Node_profile:0. BindAxis:0 Drive_profile:0 Controlword:fh drive_status:1237h Drive_mode:8h target:ffff067h encode:ffff068h.</pre> 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. Axes: 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 -- EtherCAT Information Check

Type	EtherCAT subsidiary instruction
Description	It can be used to check much information while debugging.
Syntax	ZTEST (30,10,nodeid) nodeid = device No., n--(n-1)
Controller	Controllers that have EtherCAT.
Example	Example 1: check present PDO and key data dictionary ZTEST(30,10,nodeid) nodeid = device No., 0--(n-1) <pre>>>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</pre>

```
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: check lost-message

```
>>ztest(30,12)
```

```
TestDriver_ecat para1:12,para2:0!
```

Slot:0 contain 1 nodes.

```
Lostcount:0-0.
```

The first data: the times of “not respond”

The second data: the times of “clock conflict”

Example 4: check if it supports 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.
```

Example 5: check PDO length

```
>>ztest(30,13)
```

Print all devices' PDO nodes.

Method 2 (valid in versions after 2310):

NODE_INFO(islot, inode, 15) check PDO bytes (sending)

NODE_INFO(islot, inode, 16) check PDO bytes (receiving)

Below is valid in version after “version_buil: 231124”

>>ztest(30,17)

Print and send PDO all contents

>>ztest(30,18)

Print and receive PDO all contents

Example 6: check multiple slot

>>ztest(32,13) 'the 2nd ECAT

>>ztest(33,13) 'the 3rd ECAT

>>ztest(34,13) 'the 4th ECAT

Example 7

ztest(30,80,0,times) set the number of continuous package loss

It also can check current configurations at the same time.

Followings are valid in firmware version after 200170612.

Example 8: check slave side (drive) package-lost numbers

For old version, you can use this command to check:

ztest (30,0,nodeid,\$300)

nodeid 0- max numbers -1

Example 9: check from slave register

ztest (30, 0, nodeid, regnum)

nodeid 0- max numbers -1

print “8 continuous reg (starting from regnum)”

Example 10: test “package sending & receiving”

Check whether the link is normal while scanning device 0.

ZTEST(60,7,0,1)

ZTEST(60,8,0,1)

Both 2 are OK, when the command is sent each time, the Ethernet led will shrink once.

Example 11: check real-time

ZTEST(61,0) reset before viewing

ZTEST(61,1) print after running one certain of time

TestDriver_int para1:1,para2:0!

inttest_fastenterticks:9307

inttest_slowestticks:34321

if it is smaller than 1/2 of scanning period, the real-time is better.

	<pre> inttest_maxprocessticks:143584 if it is smaller than 1/2 of scanning period, the real-time is better. inttest_curprocessticks:111792 Example 12: interrupt checking (valid in VPLC7XX series) m_pciisrtimes, m_isrtimes values should be consistent >>ZTEST(60,97) m_pciisrtimes 0x:3956106 m_isrtimes 0x:3956106 m_isrfirstelkmin -4987 m_isrfirstelkmax 4930 m_isrfirstelkminisr 72 m_isrfirstelkmaxisr 3125 m_isrfirstelkCur 186 m_isrLostIsres 0 m_isrLostIsres2 6413553 m_isrOverIsres 0 closeisrns 0 m_iTimerPeriod 10000 m_iCpuTimerPeriod 10000 m_iTimerPeriodns 1000000 m_iTimerPeriodAdjustMax 10000 m_iTimerPeriodAdjustMin 10000 </pre>
Instruction	PRINT

?*RTEX -- Rtex Status Print

Type	Rtex Subsidiary Instruction
Description	Use this instruction while debugging to get status of devices connected. Only valid after a successful field bus scan.
Syntax	?*RTEX
Controller	Controllers with Rtex interface.
Example	?*RTEX Result as follow: <pre> Slot:0 contain 1 nodes. Lostcount:0-0. Node:0 status:1 manid:616e6150h devicetype:31h axes:1 Alstate:1. BindAxis:0 Drive_profile:0 Controlword:80h drive_status:3c1h target:ffffe5ch encode:ffffe5ch. </pre> 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.

	Axes: 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

?*ZML -- ZML Information Print

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.
Syntax	?*ZML
Controller	Valid in ECAT controllers with firmware version above 20221021.
Example	>>?*zml Print result: Vender:41bh id:1ab0h ver:11h used:1.
Instruction	PRINT , ZML_INFO

17.3. SDO Operation Instructions

SDO_WRITE -- Data Dictionary Write

Type	Field Bus Instructions are only for EtherCAT.
Description	To do SDO writing by 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. Don't write and read SDO frequently.
Syntax	Syntax 1: 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., it is represented by one 2-bit hexadecimal

	system, for example, \$01, each sub No. means different functions. ● 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 17 double 18 byte list ● value: data value, set the value within the range of data type, for example, when the type = 2, the range is -128 ~ 127; when the type = 5, the range is 0~ 255 (255 -- 8-bit binary system 11111111); when the type = 9, value is TABLE register address, write needed values into TABLE register. Syntax 2: when type = 18, SDO_WRITE (slot, node, index, subindex, 18, tablenum, sizebytes) ● tablenum: TABLE position where the read data is stored ● sizebytes: length
Controller	Controllers With EtherCAT.
Example	<pre> 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. SDO_WRITE(0,0,\$6200,\$01,5,255) 'open all OUT0-OUT7 of EIO module SDO_WRITE(0,0,\$6200,\$01,5,0) 'close all OUT0-OUT7 of EIO module SDO_WRITE(0,0,\$6200,\$02,5,255) 'open all OUT8-OUT15 of EIO module ENDIF </pre>
Instructions	SDO_WRITE_AXIS , SDO_READ

SDO_WRITE_AXIS -- Data Dictionary Write (with Axis)

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.
Syntax	SDO_WRITE_AXIS (axis, index, subindex, type, value) - axis: axis No. - index: data dictionary No., add "\$" before the value to indicate hexadecimal, such as \$6060. - subindex: sub 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 - value: data value
Controller	Controllers with EtherCAT interface.
Example	Please use the followed sample before a EtherCAT device is connected successfully. <pre> 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 </pre>
Instructions	SDO_WRITE , SDO_READ_AXIS

SDO_READ -- Data Dictionary Read

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.
Syntax	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 where 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	<pre> 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 SDO_READ (0,0,\$6000, \$01,5,0) 'read EIO module IN0~IN7 state, save data into table(0) SDO_READ (0,0,\$6000, \$02,5,-1) 'read EIO module IN8~IN15 state, save data into table(0) ENDIF </pre>
Instructions	SDO_READ_AXIS , SDO_WRITE

SDO_READ_AXIS -- Data Dictionary Read (with Axis)

Type	Field Bus Instructions are 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.
Syntax	SDO_READ_AXIS (axis, index, subindex, type, tablenum) ● 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 8 float 9 string ● tablenum: TABLE where saves read data
Controller	Controllers with EtherCAT interface.
Example	Please use the followed sample before a EtherCAT device is connected successfully. <pre> 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 </pre>
Instructions	SDO_READ , SDO_WRITE_AXIS

SDO_READ2 -- Data Dictionary Read (to MODBUS)

Type	Field Bus Instructions are only for EtherCAT.
Description	Do SDO reading through node (device No.) and slot (bus 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 the data dictionary that can be read.
Syntax	SDO_READ2 (slot, node, index, subindex, type, modbusnum) ● 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 ● modbusnum: MODBUS where saves read data Directly input this command in “online command”, when modbusnum is -1, won't store, print directly. Note: through this command, the problem of precision loosing (caused by converting into float type with MODBUS reading method) can be avoided.
Controller	Controllers with EtherCAT interface.
Example	<pre> SLOT_SCAN(0) IF NODE_COUNT(0)>0 THEN SDO_READ (0,0,\$6060,0,2,0) 'read node 0 (device 0) control mode, save data into MODBUS_LONG (0) ?modbus_long(0) 'print the data ENDIF </pre>
Instructions	SDO_WRITE_AXIS , SDO_WRITE

17.4. Device Instructions

NODE_COUNT -- Connected Device Numbers

Type	Field Bus Instructions
Description	The total number of devices connected through Bus. Only valid after a successful field bus scan.
Syntax	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 1 <pre> SLOT_SCAN(0) ? NODE_COUNT(0) 'print devices No. connected in slot 0. </pre> Example 2

	SLOT_SCAN(0) IF NODE_COUNT (0) = 3 THEN 'set how many devices connected, then do axes mapping, axis type ATYPE configurations, etc. ENDIF
Instructions	NODE_INFO

NODE_STATUS -- Device Status

Type	Field Bus Instructions								
Description	Device status, which is valid after successful field bus scan. <table border="1"> <thead> <tr> <th>BIT</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>0</td><td>Indicates if node exists, 1-exist.</td></tr> <tr> <td>1</td><td>Communication status, 1-error.</td></tr> <tr> <td>2</td><td>Node Status, 1-error.</td></tr> </tbody> </table> When value = 1 → bit0=1, bit1 & bit2=0, device communication is normal. When value = 3 → bit0 & bit1=1, bit2=0, error in device communication.	BIT	Meaning	0	Indicates if node exists, 1-exist.	1	Communication status, 1-error.	2	Node Status, 1-error.
BIT	Meaning								
0	Indicates if node exists, 1-exist.								
1	Communication status, 1-error.								
2	Node Status, 1-error.								
Syntax	Only for read: var = NODE_STATUS (slot, node) - slot: slot No., 0-default - node: device No., starting 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 Numbers

Type	Field Bus Instructions
Description	Connected motors of each device. Only valid after successful field bus scan.
Syntax	Only for Read: var = NODE_AXIS_COUNT (slot, node) - slot: slot No., 0-default - node: device No., starting 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 motor numbers of device 0. ENDIF
Instructions	NODE_INFO

NODE_IO -- Device IO

Type	EtherCAT Field Bus Instructions
Description	Set device IO starting No., single device's IN starting No. and OP starting No. are the same. 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 other devices with IOs.
Syntax	To read: var = NODE_IO (slot, node) To write: NODE_IO(slot, node)=iobase ● slot: slot No., 0-default ● node: device No., starting from 0.
Controller	Controllers with EtherCAT or Rtex
Example	<pre> SLOT_SCAN(0) IF NODE_COUNT(0)>0 THEN NODE_IO(0,0)=32 'set IO starting No. as 32 in device 0. ?NODE_IO(0,0) 'print the IO start No. of device 0 ENDIF </pre>
Instructions	NODE_AIO

NODE_AIO -- Device Analog

Type	Field Bus Instructions						
Description	Set device AIO starting No., single device's AIN starting No. and AOUT starting No. are the same. Setting value should be times of 8. Only valid after successful field bus scan. Usually used in ZMIO expansion module for AIO configuration, also valid in other devices with AIOs.						
Syntax	To read: var=NODE_AIO (slot, node[,idir]) To write: NODE_AIO(slot, node[,idir])=Aiobase ● slot: slot No., 0-default. ● node: device No., starting from 0. ● idir: choose AD or DA <table border="1"> <tr> <td>0</td><td>Default; set both AIN & AOUT, but only read AIN.</td></tr> <tr> <td>3</td><td>AIN</td></tr> <tr> <td>4</td><td>AOUT</td></tr> </table>	0	Default; set both AIN & AOUT, but only read AIN.	3	AIN	4	AOUT
0	Default; set both AIN & AOUT, but only read AIN.						
3	AIN						
4	AOUT						
Controller	Controllers with EtherCAT or Rtex						
Example	<pre> SLOT_SCAN(0) IF NODE_COUNT(0)>0 THEN NODE_AIO(0,0,3)=3 'set AIN starting No. as 3 in device 0. ?NODE_AIO(0,0,3) 'print AIO start No. of device 0. </pre>						

	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.																																																		
Syntax	For reading: var=NODE_INFO (slot, node, sel, [,moduid]) ● slot: slot No., 0-default. ● node: device No., starting from 0. ● sel: information No. ● moduid: optional parameters, it is used when checking submodule information, parameter value is n-1 (n: the number of submodules). <table> <tr> <th>Value</th><th>Descriptions</th></tr> <tr><td>0</td><td>VENDER, manufaturer No.</td></tr> <tr><td>1</td><td>DEVICE, Device No.</td></tr> <tr><td>2</td><td>VERSION, Version</td></tr> <tr><td>3</td><td>ALIAS, Name to distinguish drive.</td></tr> <tr><td>4</td><td>Reserved</td></tr> <tr><td>5</td><td>Connection breaks, by BIT, valid in firmware 240531.</td></tr> <tr><td>6</td><td>Ethernet state, by BIT, valid in firmware 240531.</td></tr> <tr><td>10</td><td>the number of IN</td></tr> <tr><td>11</td><td>the number of OP (OUT)</td></tr> <tr><td>12</td><td>the number of AIN (analog inputs)</td></tr> <tr><td>13</td><td>the number of AOUT (analog outputs)</td></tr> <tr><td>14</td><td>the bytes of PDO sending, vaid in firmware 230823.</td></tr> <tr><td>15</td><td>the bytes of PDO receiving, vaid in firmware 230823.</td></tr> <tr><td>16</td><td>the number of submodules</td></tr> <tr><td>17</td><td>check submodule type, it needs moduid</td></tr> <tr><td>18</td><td>lag time of scanned device (ns)</td></tr> <tr><td>19</td><td>sync offset of EtherCAT (ns)</td></tr> <tr> <td colspan="2"> 20-27: it needs moduid, valid in firmware 241203. 20-23: it is only valid in zml devices. -1: wrong parameters returned. </td></tr> <tr><td>20</td><td>submodule IN numbers (plug in & out type submodule)</td></tr> <tr><td>21</td><td>submodule OP numbers (plug in & out type submodule)</td></tr> <tr><td>22</td><td>submodule AIN numbers (plug in & out type submodule)</td></tr> <tr><td>23</td><td>submodule AOUT numbers (plug in & out type submodule)</td></tr> <tr><td>24</td><td>PDO bytes sent by submodule</td></tr> <tr><td>25</td><td>PDO bytes received by submodule</td></tr> </table>	Value	Descriptions	0	VENDER, manufaturer No.	1	DEVICE, Device No.	2	VERSION, Version	3	ALIAS, Name to distinguish drive.	4	Reserved	5	Connection breaks, by BIT, valid in firmware 240531.	6	Ethernet state, by BIT, valid in firmware 240531.	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, vaid in firmware 230823.	15	the bytes of PDO receiving, vaid in firmware 230823.	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)	20-27: it needs moduid, valid in firmware 241203. 20-23: it is only valid in zml devices. -1: wrong parameters returned.		20	submodule IN numbers (plug in & out type submodule)	21	submodule OP numbers (plug in & out type submodule)	22	submodule AIN numbers (plug in & out type submodule)	23	submodule AOUT numbers (plug in & out type submodule)	24	PDO bytes sent by submodule	25	PDO bytes received by submodule
Value	Descriptions																																																		
0	VENDER, manufaturer No.																																																		
1	DEVICE, Device No.																																																		
2	VERSION, Version																																																		
3	ALIAS, Name to distinguish drive.																																																		
4	Reserved																																																		
5	Connection breaks, by BIT, valid in firmware 240531.																																																		
6	Ethernet state, by BIT, valid in firmware 240531.																																																		
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, vaid in firmware 230823.																																																		
15	the bytes of PDO receiving, vaid in firmware 230823.																																																		
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)																																																		
20-27: it needs moduid, valid in firmware 241203. 20-23: it is only valid in zml devices. -1: wrong parameters returned.																																																			
20	submodule IN numbers (plug in & out type submodule)																																																		
21	submodule OP numbers (plug in & out type submodule)																																																		
22	submodule AIN numbers (plug in & out type submodule)																																																		
23	submodule AOUT numbers (plug in & out type submodule)																																																		
24	PDO bytes sent by submodule																																																		
25	PDO bytes received by submodule																																																		

		26	PDO starting offset send by submodule					
		27	PDO starting offset received by submodule					
		30	device special parameters					
		0x300-0x307	package losing information of Esc					
	<i>If the range is configured via SDO during operation, the device needs to be scanned again with SLOT_SCAN(0) for the corresponding NODE_INFO (default, default, 17, module address) command value to be updated synchronously. Not scanning SLOT_SCAN(0) again will not affect normal operation. SDO can be used to read instead of NODE_INFO command to monitor range data in real time.</i> Writing mode is valid in ZMXC4XX series controllers with fast firmware version above 20190201. To Write: NODE_INFO (slot, node, sel) = value ● slot: slot No., 0-default. ● node: device No., starting from 0. ● sel: information No. <table><tr><th>Value</th><th>Description</th></tr><tr><td>3</td><td>ALIAS, alias for distinguish from drives, if the drive uses eeprom for storage, this can be used to modify.</td></tr><tr><td>19</td><td>Sync offset iof Ecat (ns), which can be modified after bus scanned and before scan open.</td></tr></table>			Value	Description	3	ALIAS, alias for distinguish from drives, if the drive uses eeprom for storage, this can be used to modify.	19
Value	Description							
3	ALIAS, alias for distinguish from drives, if the drive uses eeprom for storage, this can be used to modify.							
19	Sync offset iof Ecat (ns), which can be modified after bus scanned and before scan open.							
Controller	Controllers with EtherCAT or Rtex							
Example	<pre>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. ?NODE_INFO(0,0,20,0) 'read how many IN of plug in & out submodule 0 ENDIF</pre>							
Instructions	SLOT_SCAN							

NODE_PROFILE -- PDO Preset

Type	Field Bus Instructions
Description	Profile setting of field bus devices. Reserved, modification is not allowed after field bus starts. When NODE_PROFILE (slot, node) = -1 is written before SLOT_STATR, which means open controller customized PDO setting.
Syntax	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	Valid in special EtherCAT devices (PDO). Read or write PDO for devices except for IO and non-axis devices, such as power supply devices. For axis devices and IO devices, don't use this command because they can access PDO by axis parameters and IO commands. SLOT_START will read present PDO list automatically in advance and it can set current PDO values. Please modify it after opened. You can use SLOT_START to start, making it at SAFEOP, then set initialized PDO state, next open OP.																		
Syntax	Command Syntax: NODE_PDOBUFF (slot, node, index, subindex, type) Function Syntax: Buff=NODE_PDOBUFF (slot, node, index, subindex ,type) - slot: slot No., 0-default. - node: device No., starting from 0. - index: data dictionary No., add "\$" before the value to indicate hexadecimal, such as \$6060. - subindex: subsidiary No. - type: data type <table border="1"> <tbody> <tr><td>1</td><td>Boolean</td></tr> <tr><td>2</td><td>Integer 8</td></tr> <tr><td>3</td><td>Integer 16</td></tr> <tr><td>4</td><td>Integer 32</td></tr> <tr><td>5</td><td>Unsigned 8</td></tr> <tr><td>6</td><td>Unsigned 16</td></tr> <tr><td>7</td><td>Unsigned 32</td></tr> <tr><td>8</td><td>float</td></tr> <tr><td>9</td><td>string</td></tr> </tbody> </table>	1	Boolean	2	Integer 8	3	Integer 16	4	Integer 32	5	Unsigned 8	6	Unsigned 16	7	Unsigned 32	8	float	9	string
1	Boolean																		
2	Integer 8																		
3	Integer 16																		
4	Integer 32																		
5	Unsigned 8																		
6	Unsigned 16																		
7	Unsigned 32																		
8	float																		
9	string																		
Controller	Valid in ZMC4XX EtherCAT controllers, whose firmware version should be above 20170508.																		
Example	<pre>>>>NODE_PDOBUFF(0,0, \$6040, 0, 3) = 15 >>? NODE_PDOBUFF (0,0, \$6041, 0, 3)</pre>																		
Instructions	SDO_WRITE , SDO_READ																		

NODE_PDO_WRBUFF -- PDO Modify by Offset

Type	Field Bus Instructions.

Description	Modify PDO according to offset.
Syntax	NODE_PDO_WRBUFF (slot, node, offset, tableindex, size) ● slot: slot No., 0 – default ● node: device No., 0 – ● offset: PDO byte offset ● tableindex: TABLE starting No. where 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 -- PDO Read by Offset

Type	Field Bus Instructions.
Description	Read PDO according to offset.
Syntax	NODE_PDO_RDBUFF (slot, node, offset, tableindex, size) ● slot: slot No., 0 – default ● node: device No., 0 – ● offset: PDO byte offset ● tableindex: TABLE starting No. where saves data ● size: the number of data bytes to be read
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).
Syntax	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, valid in firmware 220418.

Example	>>NODE_REGWRITE (0, 0, address, bytes, value)
Instruction	NODE_REGREAD

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).
Syntax	● 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, valid in firmware above 2204183
Example	>>NODE_REGREAD(0, 0, address, bytes, modbusindex)
Instruction	NODE_REWRITEE

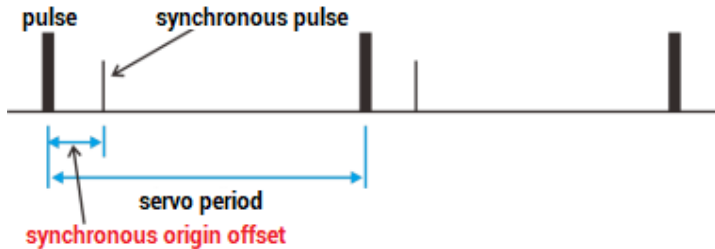
NODE_PRESET -- Device Preset

Type	EtherCAT Field Bus Instructions
Description	Preset field bus devices: please start the bus in advance even if there is no external device after preset. 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, field bus can't start. If no preset value for added devices, and no scan process to detect them, then field bus also can't start.
Syntax	Command Syntax1: NODE_PRESET (slot, node, manuid, productid) Command Syntax2: NODE_PRESET (slot, -1), clear all preset. Function Syntax1: VALUE=NODE_PRESET (slot,node), check if there is preset. Function Syntax2: VALUE=NODE_PRESET (slot), check the maximum number of preset. ● slot: slot No., 0-default. ● node: Device No., starting from 0.

	● manuid: Manufacturer ID, see NODE_INFO for reference. ● productid: Device ID serial NO., see NODE_INFO for reference.
Controller	Controllers with EtherCAT or Rtex; Valid in controllers with firmware version above 20160601.
Example	<pre> 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 </pre>
Instructions	NODE_STATUS

ZML_INFO -- Device XML Check

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.																								
Syntax	➤ Function Syntax: <pre>para = ZML_INFO(infosel, venderid, devid, [version,])</pre> ● infosel: operation No. <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>VENDER, manufacturer No.</td></tr> <tr> <td>1</td><td>DEVICE, device No.</td></tr> <tr> <td>2</td><td>VERSION, version</td></tr> <tr> <td>10</td><td>the number of fixed IN of device</td></tr> <tr> <td>11</td><td>the number of fixed OP of device</td></tr> <tr> <td>12</td><td>the number of fixed AIN of device</td></tr> <tr> <td>13</td><td>the number of fixed AOUT of device</td></tr> <tr> <td>19</td><td>Shift time configuration, the unit is ns</td></tr> <tr> <td>21</td><td>the waiting time of AL states 1-2 switching, in ms.</td></tr> <tr> <td>22</td><td>the waiting time of AL states 2-4 switching, in ms.</td></tr> <tr> <td>23</td><td>the waiting time of AL states 4-8 switching, in ms.</td></tr> </tbody> </table> ● venderid: manufacturer ID ● devid: device ID ● version: version No., optical ● 21-23: it is valid after firmware 230927. ➤ Function Syntax: <pre>para = ZML_INFO(infosel, venderid, devid, [version,])</pre>	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	21	the waiting time of AL states 1-2 switching, in ms.	22	the waiting time of AL states 2-4 switching, in ms.	23	the waiting time of AL states 4-8 switching, in ms.
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																								
21	the waiting time of AL states 1-2 switching, in ms.																								
22	the waiting time of AL states 2-4 switching, in ms.																								
23	the waiting time of AL states 4-8 switching, in ms.																								

	➤ command Syntax: ZML_INFO(infosel, venderid, devid, [version,]) = para Now, it only supports mode 19. Please modify it before bus scanning, the synchronous origin offset is modified, see below showing:  Notes: 1) several devices may share one XML file, at this time, they will be changed at the same time when modifying several devices. XML file can be checked through venderid, devid or ?*ETHERCAT after scanned. 2) After setting the sync origin offset, read NODE_INFO mode 19 to check whether the setting takes effect.
Controller	Controllers with EtherCAT
Example	<pre> >>?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) 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 </pre>
Instructions	SLOT_SCAN

17.5. Drive Instructions

DRIVE_MODE -- Drive Mode

Type	Axis Parameters
Description	Set the drive's control Mode, mapped data dictionary is 0x6060. Only valid after correct ATYPE settings. (set as 65/66/67)

Syntax	To Read: var=DRIVE_MODE (axis) To Write: DRIVE_MODE (axis)= value ● Axis: axis No.
Controller	Controllers with EtherCAT or Rtex
Example	<pre> SLOT_SCAN(0) ... 'axis enable process, refer to Chapter 23 simple routines 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 </pre>

DRIVE_PROFILE -- Drive PDO Settings

Type	EtherCAT axis parameters
Description	PDO Sending or Receiving setting of each axis. Only valid after correct ATYPE settings. (set as 65/66/67) Consult the manufacturer for more. EtherCAT: When DRIVE_PROFILE=-1, it indicates that the controller will follow the default PDO list in drive, only valid in controller with firmware above 20160601. If default PDO list doesn't 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, 0x606000
08},
//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, 0x606000
08},
//control word, target position, IO output, probe setting, torque limit, mode
{0x60410010, 0x60640020, 0x60770010, 0x60fd0020, 0x60b90010, 0x60ba00
20, 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 {0x60400010,0x60710010,0x60ff0020,0x607a0020,0x60b80010,0x607200 10,0x60600008}, //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 {0x60400010,0x60710010,0x60ff0020,0x607a0020,0x60b80010,0x607200 10, 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},

	<pre>//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 {0x60400010,0x60710010,0x607a0020,0x60b80010,0x60720010,0x606000</pre>

	<pre> 08} //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) </pre>
Syntax	To read: var= DRIVE_ PROFILE(axis) To write: DRIVE_ PROFILE(axis)= value axis: axis No.
Controller	Controllers with EtherCAT or Rtex
Example	<pre> 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 </pre>
Instructions	ATYPE , NODE_PROFILE

DRIVE_CW_MODE -- Drive Settings

Type	Axes Parameters
Description	Set drive parameters. Only valid after correct ATYPE settings. (set as 65/66/67) For some version, MODE can be operated directly. Note: don't modify control word of RTEX. 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.
Syntax	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, operate by bit. Only valid after correct ATYPE settings. (set as 65/66/67)																																																																									
	For EtherCAT based drive, the mapped data dictionary is 0x6040.																																																																									
	This parameter will change automatically to enable the drive according to WDOG / AXIS_ENABLE when EtherCAT controller’s ATYPE (axis type) is 65. Main bits setting as follow, see relevant drive manual for details.																																																																									
	<table><tr><th rowspan="3">Command</th><th colspan="5">bits of the controlword</th><th rowspan="3">PDS Transitions</th></tr><tr><th>bit 7</th><th>bit 3</th><th>bit 2</th><th>bit 1</th><th>bit 0</th></tr><tr><th>fault reset</th><th>enable operation</th><th>quick stop</th><th>enable voltage</th><th>switch on</th></tr><tr><td>Shutdown</td><td>0</td><td>–</td><td>1</td><td>1</td><td>0</td><td>2, 6, 8</td></tr><tr><td>Switch on</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>3</td></tr><tr><td>Switch on + Enable operation</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>3+4 (*1)</td></tr><tr><td>Enable operation</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>4, 16</td></tr><tr><td>Disable voltage</td><td>0</td><td>–</td><td>–</td><td>0</td><td>–</td><td>7, 9, 10, 12</td></tr><tr><td>Quick stop</td><td>0</td><td>–</td><td>0 (*2)</td><td>1</td><td>–</td><td>7, 10, 11</td></tr><tr><td>Disable operation</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>5</td></tr><tr><td>Fault reset</td><td></td><td>–</td><td>–</td><td>–</td><td>–</td><td>15</td></tr></table>	Command	bits of the controlword					PDS Transitions	bit 7	bit 3	bit 2	bit 1	bit 0	fault reset	enable operation	quick stop	enable voltage	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
Command	bits of the controlword					PDS Transitions																																																																				
	bit 7		bit 3	bit 2	bit 1		bit 0																																																																			
	fault reset	enable operation	quick stop	enable voltage	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. When manual-mode is needed, please set DRIVE_CW_MODE as 1 at first, don’t modify it randomly. And each bit meanings are shown as below, please check RTEX manual chapter 4-2-3 contents.																																																																									
	<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>Servo_On</td><td>0</td><td>0</td><td>Gain_SW</td><td>TL_SW</td><td>HM_Ctrl</td><td>0</td><td>0</td></tr><tr><td>bit15</td><td>bit14</td><td>bit13</td><td>bit12</td><td>bit11</td><td>bit10</td><td>bit9</td><td>bit8</td></tr><tr><td>Hard_Stop</td><td>Smooth_Stop</td><td>Pause</td><td>0</td><td>SL_SW</td><td>0</td><td>EX-OUT2</td><td>EX-OUT1</td></tr></table>	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																																									
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																																																																			
Syntax	To Read: var=DRIVE_CONTROLWORD(axis) To Write: DRIVE_CONTROLWORD(axis)=value ● axis: axis No.																																																																									

Controller	The controller that has EtherCAT or RTEX interface.
Example	Example 1: for EtherCAT <pre> SLOT_SCAN(0) IF NODE_COUNT(0)>0 THEN AXIS_ADRESS(0)=1 ATYPE(0)=65 'set EtherCAT ass position control mode DRIVE_PROFILE(0)=0 'set PDO as 0 DRIVE_CONTROLWORD(0)=128 'clear servo errors DELAY (100) DRIVE_CONTROLWORD(0)=6 'servo shutdown DELAY (100) DRIVE_CONTROLWORD(0)=15 'servo switch on ENDIF </pre> Example 2: for RTEX <pre> SLOT_SCAN(0) IF NODE_COUNT(0)>0 THEN AXIS_ADDRESS(0)=1 ATYPE(0)=50 'set RTEX as position control mode DRIVE_PROFILE(0)=1 'servo that is with IO DRIVE_CW_MODE=1 'set control data word by manual. DRIVE_CONTROLWORD(0)=128 'servo enable. ENDIF </pre>
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 Settings. (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.																												
	<table><tr><td>Status Word</td><td colspan="2">PDO state</td></tr><tr><td>xxxx xxxx x0xx 0000 b</td><td>not ready to switch on</td><td>initialization, not done</td></tr><tr><td>xxxx xxxx x1xx 0000 b</td><td>switch on disabled</td><td>initialization, done</td></tr><tr><td>xxxx xxxx x01x 0001 b</td><td>ready to switch on</td><td>main power is off</td></tr><tr><td>xxxx xxxx x01x 0011 b</td><td>switch on</td><td>servo enable off/servo prepared</td></tr><tr><td>xxxx xxxx x01x 0111 b</td><td>operation enabled</td><td>servo enable on</td></tr><tr><td>xxxx xxxx x00x 0111 b</td><td>quick stop active</td><td>stop rapidly</td></tr><tr><td>xxxx xxxx x0xx 1111 b</td><td>fault reaction active</td><td>abnormal (alarm) judge</td></tr><tr><td>xxxx xxxx x0xx 1000 b</td><td>fault</td><td>abnormal (alarm) status</td></tr></table>			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
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.)																																
	<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>Servo_Active</td><td>Servo_Ready</td><td>Alarm</td><td>Warning</td><td>Torque_Limited</td><td>Homing_Complete</td><td>In_Progress</td><td>In_Position</td></tr><tr><td>bit15</td><td>bit14</td><td>bit13</td><td>bit12</td><td>bit11</td><td>bit10</td><td>bit9</td><td>bit8</td></tr><tr><td>SI-MON5/E-STOP</td><td>SI-MON4/EX-SON</td><td>SI-MON3/E XT3/STOP</td><td>SI-MON2/E XT2/RET</td><td>SI-MON1/EXT1</td><td>HOME</td><td>POT/NOT</td><td>NOT/POT</td></tr></table>	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/E XT3/STOP	SI-MON2/E XT2/RET	SI-MON1/EXT1	HOME	POT/NOT	NOT/POT
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/E XT3/STOP	SI-MON2/E XT2/RET	SI-MON1/EXT1	HOME	POT/NOT	NOT/POT																										
Syntax	Read-Only: toq = DRIVE_STATUS (axis) axis: axis No.																																
Controller	Controllers with EtherCAT or Rtex																																
Example	<pre>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</pre>																																
Instructions	ATYPE , DRIVE_PROFILE																																

DRIVE_IO -- Drive IO Settings

Type	Axis Parameters																										
Description	Set drive IO start No., IN No. and OP No. are the same. It is valid when DRIVE_PROFILE supports reading drive IO.																										
Syntax	To read: var= DRIVE_IO (axis) To write: DRIVE_IO (axis)=value axis: Axis No. For inputs and outputs function & address of EtherCAT fieldbus servo, please see data dictionary 60FD, 60FE for reference. (attention: some manufacturers don't set standard protocol address, please see drive manual) Rtex Fieldbus Servo inputs and outputs function related controller address=DRIVE_IO+NO. <table border="1"> <thead> <tr> <th>DRIVE_IO and below No.</th><th>Servo function</th></tr> </thead> <tbody> <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> </tbody> </table>	DRIVE_IO and below No.	Servo function	Inputs		0	NOT/POT	1	POT/NOT	2	HOME	3	SI-MON1/EXT1	4	SI-MON2/EXT2	5	SI-MON3/EXT3	6	SI-MON4/EX-SON	7	SI-MON5/E-STOP	Outputs		0	EX-OUT1	1	EX-OUT2
DRIVE_IO and below No.	Servo function																										
Inputs																											
0	NOT/POT																										
1	POT/NOT																										
2	HOME																										
3	SI-MON1/EXT1																										
4	SI-MON2/EXT2																										
5	SI-MON3/EXT3																										
6	SI-MON4/EX-SON																										
7	SI-MON5/E-STOP																										
Outputs																											
0	EX-OUT1																										
1	EX-OUT2																										

Controller	Controllers with EtherCAT or Rtex
Example	<pre> SLOT_SCAN(0) ... 'axis enable process, please refer to Chapter 23 Simple Routines 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 </pre>
Instructions	NODE_IO , DRIVE_PROFILE

DRIVE_TORQUE -- Drive Torque

Type	EtherCAT axis status
Description	Current drive torque value. Only valid after correct ATYPE and DRIVE_PROFILE settings.
Syntax	Read-Only: var=DRIVE_TORQUE(axis) ● axis: axis No.
Controller	Controllers with EtherCAT or Rtex
Example	<pre> 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 </pre>
Instructions	ATYPE , DRIVE_PROFILE

DRIVE_TORQUEMAX -- Drive Max Torque Limit

Type	EtherCAT axis status
Description	The drive's current torque max limit. Only valid after correct ATYPE (65/66/67) and DRIVE_PROFILE setting.
Syntax	To read: var = DRIVE_TORQUEMAX(axis) To write: DRIVE_TORQUEMAX(axis) = value ● axis: axis No.
Controller	Controllers with EtherCAT or Rtex
Example	<pre> 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_TORQUEMAX (0) 'print torque of axis 0. ENDIF </pre>
Instructions	ATYPE , DRIVE_PROFILE , DRIVE_TORQUE

DRIVE_TORQUEMIN -- Drive Min Torque Limit

Type	EtherCAT axis status
Description	The drive's current torque minimal limit. Only valid after correct ATYPE (65/66/67) and DRIVE_PROFILE settings.
Syntax	To read: var = DRIVE_TORQUEMIN(axis) To write: DRIVE_TORQUEMINN(axis) = value ● axis: axis No.
Controller	Controllers with EtherCAT or Rtex
Example	<pre> 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_TORQUEMIN (0) 'print torque of axis 0. ENDIF </pre>
Instructions	ATYPE , DRIVE_PROFILE , DRIVE_TORQUE , DRIVE_TORQUEMAX

DRIVE_FE -- Drive Error

Type	Axis Status
Description	Read present error overrun of drive, corresponding object dictionary is 0x60F4. Only valid after correct ATYPE setting. (set as 65/66/67) It is valid for ZMC408SCAN-V22 when its ATYPE = 22 (with feedback position), at this time, return the deviation value of drive received position and feedback position.
Syntax	Read-Only: var=DRIVE_FE (axis) ● axis: axis No.
Controller	Controllers with EtherCAT or Rtex
Example	<pre> SLOT_SCAN(0) 'field bus scan. IF NODE_COUNT(0)>0 THEN AXIS_ADDRESS(0)=1 'map the first axis to axis 0 ATYPE(0)=65 'set axis 0 type as 65, position control mode. DRIVE_PROFILE(0)=0 'PDO period report config, by DRIVE_PROFILE ?DRIVE_FE (0) 'read the following error value of axis 0. ENDIF </pre>
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 Settings. (set as 65/66/67) Reserved.
Syntax	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 -- Alarm Clear

Type	Field Bus Instructions
Description	Operate present BASE axis, clear drive alarm. 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.
Syntax	BASE(axis No.) DRIVE_CLEAR(para) - para: 0 Clear present alarm 1 Clear history alarm 2 Clear external input alarm
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 -- Parameters Read

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.	
Syntax	DRIVE_READ(para [,vr_index])	
	● para:	
	Value	Descriptions
	1	parameter classification * 256 + parameter No. (Pr7.20=7*256+20)
	2	parameter = 130, read the latch status, BIT0 and BIT1 indicate status of 2 channels.
3	parameter = \$10000 + (ssid), read information of Rtex drive	

	system, string type data will be saved in VRSTRING.
4	parameter = \$20000 + (alarm code) + (\$1000*index), read alarm information.
5	parameter = \$30000 + (monitor code) + (\$1000*index), read monitor information.

- vr_index: save the read data in VR, if this parameter is omitted, print out it in output bar 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 time 2: 2 times
Pr7.91	Rtex communication period expansion	62500 ns 125000 ns 250000 ns 500000 ns 1000000 ns 2000000 ns

3.Syestem ID Commands

SSID	Definitions
\$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 Commands	
	Function code	Function
		Index
	\$000	Read now alarm / alarm records
		0: alarm code of this time 1: alarm code of former time 2: alarm code of former two times ... 14: alarm code of 14 times before
	\$001	Clear now alarm
		0: clear alarm of this time
	\$011	Clear all alarms
		0: clear alarm 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)

	<pre>?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</pre>
Instructions	DRIVE_WRITE , DRIVE_CLEAR

DRIVE_WRITE -- Parameters Write

Type	Field Bus Instructions, only valid in Rtex controller.		
Description	Operate Base axes, read drive parameters. Check RETURN value to judge if the operation is done successfully or not.		
Syntax	DRIVE_WRITE(para, value)		
	● para:		
	Value	Descriptions	
	1	parameter classification * 256 + parameter No. (Pr7.20=7*256+20)	
	2	special parameter = 128, current parameters are written into EEPROM (now value = 1)	
	3	special parameter = \$40000 + typecode + (set value * 256), se homing latch position function of drive	
	● value: parameter's 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)	
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
		3	Pr0.13		Pr5.22	
		4	Pr5.22	Pr0.13	Pr5.22	Pr5.25
	Pr5.22	The second torque limit		0~500%		
	Pr5.25	torque limit in positive		0~500%		
	Pr5.26	torque limit in negative		0~500%		
	Speed:					
	Pr3.12	Acceleration setting	time	0~10000ms (reach 1000.r/min)		
	Pr3.13	Deceleration setting	time	0~10000ms (reach 1000.r/min)		
	Pr3.14	S acceleration and deceleration setting			0~10000ms	
	Pr3.17	Speed limit selection: select speed limit value mode when torque controls		Set value	SL_SW	
					0	1
				[0]	Pr3.21	
				1	Pr3.21	Pr3.22
			When set as 1, select it as SL_SW value of RTEX communication instruction.			
	Pr3.21	Speed limit value 1		0~20000.r/min		
	Pr3.22	Speed limit value 2		0~20000.r/min		
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 time 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)							
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
	LATCH_SEL2				LATCH_SEL1			
	Latch 1/2 settings		Description					
	\$0		Z Phase signal trigger					
\$1		EXT1 logic rising edge						
\$2		EXT2 logic rising edge						
\$3		EXT3 logic rising edge						
\$9		EXT1 logic falling edge						
\$10		EXT2 logic falling edge						
\$11		EXT3 logic falling edge						
Controller	Controllers with Rtex							
Example	Only valid when field bus starts successfully Example one IF NODE_COUNT(0)>0 THEN BASE(0)							

Chapter 18 ZHD Teaching Box Parameters

18.1. Teaching Box Instructions

LCD_CONNECT -- LCD No. Settings

Type	Teaching Box Commands
Description	This command is used when controller and teaching box communicates successfully, and it is operated in teaching box side. Set the controller's LCD (HMI) No. that is to be connected, default is 0. When the controller supports multi-HMI, it can be set as other values. After set, please restart the ZHD teach pendant for reconnection.
Syntax	VAR1=LCD_CONNECT LCD_CONNECT=VAR1
Example	>> LCD_CONNECT=1 >> LCD_CONNECT=0

IP_CONNECT -- IP Connect

Type	Teaching Box Commands
Description	This command is used when controller and teaching box communicates successfully, and it is operated in teaching box side. Set the controller's IP that is to be connected. It will reconnect after each time modification.
Syntax	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 side. Specify teaching box's IP address, the default is 192.168.0.10.
Syntax	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 side. Specify teaching box's IP gateway, the default is 192.168.0.1.
Syntax	IP_GATEWAY = dot.dot.dot.dot
Example	>> IP_GATEWAY = 192.168.0.1
Instructions	IP_ADDRESS , IP_NETMASK

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 side. Specify teaching box's IP mask, the default is 255.255.0.0.
Syntax	IP_NETMASK = dot.dot.dot.dot
Example	>> IP_NETMASK = 255.255.0.0

18.2. Controller Instructions

LCD_LEDSTATE -- LED State Settings

Type	System Screen Parameters
Description	This command is used when controller and teaching box communicates successfully, and it is operated in teaching box side. Set LED state on teaching box, which is set by BIT, default is 1, and turn on the LED. It needs both firmware support of controller and teaching box.
Syntax	VAR = LCD_LEDSTATE (lcdnum) LCD_LEDSTATE (lcdnum) = VAR ● lcdnum: controller LCD (HMI) No., default is 0
Controller	Controller that supports RTHMI; Valid in VERSION_BUILD = 230801 or above.
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 teaching box side. Set the time (ms) that is used to process screen power-down problem. 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.
Syntax	LCD_WDOGTIME (lcdnum) = time ● lcdnum: controller LCD (HMI) No., default is 0 ● time: time, the unit is ms
Controller	Controller that supports RTHMI; Valid in 5XX series with firmware version 20180404 or above, and valid in 4XX series with firmware version 20170721 or above.
Example	LCD_WDOGTIME(0) = 100

Chapter 19 MotionRT Instructions

19.1. MotionRT Instructions

CARD_INFO -- Control Card Info Read & Write

Type	Control Card Commands																																																		
Description	Read / Write control card information.																																																		
Syntax	--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> <th>Value</th><th>Description</th></tr> <tr> <td>0</td><td>Sub-card totals, now, fill 0 for cardnum.</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 is automatically sorted by default when powered 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 encoder axes.</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 storage, 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> </table>	Value	Description	0	Sub-card totals, now, fill 0 for cardnum.	1	DEVICE, device No., hardid.	2	VERSION, version	3	Sub-card dial code, it is only valid in PCI.	4	Reserved	5	Unique No. of sub-card, the last PCI sub-card's unique No. is the unique No. of RT.	6	Reserved for special No.	7	Reserved	8	IO offset, it is automatically sorted by default when powered on.	9	AIO offset	10	The number of IN	11	The number of OP	12	The number of AIN	13	The number of AOUT	14	Reserved	15	The number of HW OP	16	The number of pulse axes	17	The number of encoder axes.	18	The number of ECAT, 0 / 1	19	The number of scan, 0 / 1	20	The number of 3D SCAN	21	Support power-down storage, the number of VR, usually XPCI doesn't bring with VR.	22	VR offset, sorting automatically	23	The number of PWM
Value	Description																																																		
0	Sub-card totals, now, fill 0 for cardnum.																																																		
1	DEVICE, device No., hardid.																																																		
2	VERSION, version																																																		
3	Sub-card dial code, it is only valid in PCI.																																																		
4	Reserved																																																		
5	Unique No. of sub-card, the last PCI sub-card's unique No. is the unique No. of RT.																																																		
6	Reserved for special No.																																																		
7	Reserved																																																		
8	IO offset, it is automatically sorted by default when powered on.																																																		
9	AIO offset																																																		
10	The number of IN																																																		
11	The number of OP																																																		
12	The number of AIN																																																		
13	The number of AOUT																																																		
14	Reserved																																																		
15	The number of HW OP																																																		
16	The number of pulse axes																																																		
17	The number of encoder axes.																																																		
18	The number of ECAT, 0 / 1																																																		
19	The number of scan, 0 / 1																																																		
20	The number of 3D SCAN																																																		
21	Support power-down storage, the number of VR, usually XPCI doesn't bring with VR.																																																		
22	VR offset, sorting automatically																																																		
23	The number of PWM																																																		

	24	PWM starting No. on sub-card
	--How to Write-- CARD_INFO (cardnum, sel) = value ● cardnum: sub card No., 0 – default ● sel: information No.	
	Value	Description
	8	IO offset, it is automatically sorted by default when powered on. Note: the value must be multiple of 8
	9	AIO offset, automatically sorted by default when powered on.
	Example GLOBAL value value=CARD_INFO (0,1) 'read control card device No. ?value end	

?*CARD -- Sub-Card Info Print

Type	Control Card Commands
Description	Print sub-card information (send command in RTSys).
Syntax	?*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
Example	?*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 M08 ZV HW

REG_CARD -- Latch Select

Type	Control Card Commands
Description	Select latch IN. When several sub-card support latching, they can be switched. Used together with REGINPUTS, REGIST. After calling REGIST, it can

	switch immediately. ● Latch Position: REG_POSE, REG_POSF, REG_POSG, REG_POSH ● Latch Channel: MARKE, MARKF, MARKG, MARKH (it can be expanded to 8 channels at most)
Syntax	REG_CARD = value Set REG_CARD = 1 to use axis specified latch. When other values are set -- it specifies the card No. for latching.
Controller	Valid in VPLC7XX controllers.
Example	➤ Example 1: for single-card <pre>reg_card(0) = -1 'set sub-card No. to latch axis 0, starting from 0 usually base(0) dpos(0)=0 mpos(0)=0 reg_inputs=\$0000'make R0-R3 all correspond to IN0, max 8 latch channels. atype = 1 vmove(1) regist(21) wait until markd ? reg_posd(0)</pre> ➤ Example 2: for multi-card, use card 1 IN to latch the axis of card 2 <pre>base(4) dpos(4)=0 mpos(4)=0 reg_inputs=\$0000'make R0-R3 all correspond to IN0, max 8 latch channels. atype = 1 vmove(1) regist(21) wait until markd ? reg_posd(4)</pre> ➤ Example 3: card 1 latches the axis 0 of card 1 (same as example 1) 'card 1 uses itself IN to latch card 1' axis, using new added channel. <pre>BASE(0) REG_CARD = -1 '-1 is the default, which means it use the current card IN to latch the axis REG_INPUTS = \$32103210 'map R0-R3 to card 1 IN0-3, map R4-R7 to card 1 IN0-3 ATYPE=1 DPOS(0) = 0 MPOS(0) = 0 REGIST(25) 'single-latch REGIST(27) REGIST(29) REGIST(31) VMOVE(1)</pre>

	WAIT UNTIL MARKE ? REG_POSE WAIT UNTIL MARKF ? REG_POSF WAIT UNTIL MARKG ? REG_POSG WAIT UNTIL MARKH ? REG_POSH

SLOT_SLAVE -- EtherCAT Redundancy Settings

Type	Special parameter
Description	Configure 2 ECAT channels as one channel, for supporting 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 correct, otherwise, initialization will fail, or appear wrong scanned devices numbers, you can check the configuration by “?SLOT_SLAVE(islot1).
Syntax	SLOT_SLAVE(islot1)=islot2 ● islot1: master bus, it must connect to the first device's IN ● islot2: slave bus, it 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
Instructions	SLOT_INFO , NODE_REGREAD

SLOT_PERIOD -- Multi-ECAT Bus Period Settings

Type	Slot parameter.
Description	When there are multiple EtherCAT, assign different periods for each ECAT. Notes: 1. It is set when bus stops. 2. It must be the integer times of SERVO_PERIOD. 3. This is not saved into FLASH. And the default power-on value is SERVO_PERIOD.
Syntax	SLOT_PERIOD (islot) = value ● islot1: bus slot No. ● value: bus period
controller	RT750, the controller of F series that supports multi-slot.
Example	SLOT_PERIOD (0) = 250
Instructions	SERVO_PERIOD

Chapter 20 Local Slave Port Instructions

ZMIO_CONFIG -- Analog Range & Channel State Set / Get

Type	Local slave level interface RS485 bus command for XPLC300 / ZMC432M.																																																
Description	Read or set extended sub-module's AD/DA switch states & range types.																																																
Syntax	To read: var = ZMIO_CONFIG (sel, moduleid) To write: ZMIO_CONFIG (sel, moduleid, value) ● sel: function No.; ● mduleid: extended submodule address, starting from 0; ● value: configure extended submodule channel value or range type --sel: <table><tr><th>sel</th><th>Description</th></tr><tr><td>1</td><td>Configure extended submodule AD/DA range type.</td></tr><tr><td>2</td><td>Configure extended submodule analog AD channel switch state.</td></tr></table> --range data values of “value” <table><tr><th>type</th><th>value</th><th>range</th><th>type</th><th>value</th><th>range</th></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><th>AD Channel</th><th>CH3</th><th>CH2</th><th>CH1</th><th>CH0</th></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 = CH2+CH1 = 6.	sel	Description	1	Configure extended submodule AD/DA range type.	2	Configure extended submodule analog AD channel switch state.	type	value	range	type	value	range	AD	2	0~10V	DA	10	0~10V	3	-10~10V	11	-10~10V	4	4~20mA	12	4~20mA	5	0~20mA	13	0~20mA	6	0~5V	14	0~5V	7	-5~5V	15	-5~5V	AD Channel	CH3	CH2	CH1	CH0	value	8	4	2	1
sel	Description																																																
1	Configure extended submodule AD/DA range type.																																																
2	Configure extended submodule analog AD channel switch state.																																																
type	value	range	type	value	range																																												
AD	2	0~10V	DA	10	0~10V																																												
	3	-10~10V		11	-10~10V																																												
	4	4~20mA		12	4~20mA																																												
	5	0~20mA		13	0~20mA																																												
	6	0~5V		14	0~5V																																												
	7	-5~5V		15	-5~5V																																												
AD Channel	CH3	CH2	CH1	CH0																																													
value	8	4	2	1																																													
Controller	XPLC300, ZMC432M																																																
Example	ZMIO_CONFIG (1, 0, 10) 'set 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																																																
Instruction	ZMIO_INFO																																																

ZMIO_INFO -- ZMIO Itself Expansion Check

Type	Local slave level interface RS485 bus command for XPLC300 / ZMC432M.														
Description	Used to check XPLC300 series controllers' ZMIO expansion.														
Syntax	Syntax 1: var = ZMIO_INFO (sel) Syntax 2: var = ZMIO_INFO (17, moduleid) - sel: function No. <table border="1"> <thead> <tr> <th>Function No.</th><th>Meaning</th></tr> </thead> <tbody> <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> <tr> <td>17</td><td>The type No. of the sub-module, it must be with "moduleid" behind</td></tr> </tbody> </table> - moduleid: the extended sub-module address, starting from 0 in sequence.	Function No.	Meaning	10	Max IN	11	Max OP	12	Max AIN	13	Max AOUT	16	The number of modules	17	The type No. of the sub-module, it must be with "moduleid" behind
Function No.	Meaning														
10	Max IN														
11	Max OP														
12	Max AIN														
13	Max AOUT														
16	The number of modules														
17	The type No. of the sub-module, it must be with "moduleid" behind														
Controller	XPLC300, ZMC432M														
Example	?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														
Instruction	ZMIO_CONFIG														

ZMIO_OFFSET -- ZMIO Extended IO Offset

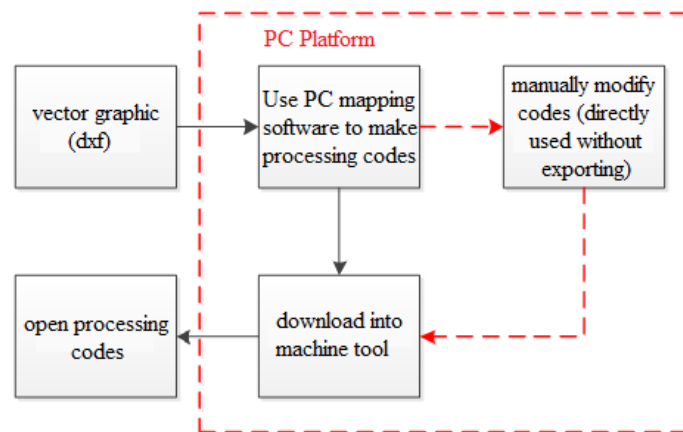
Type	The RS485 bus command of XPLC300/ZMC432M local slave interface.
Description	This is used for offset the controller itself ZMIO expanded IO address. Please note restart after ZMIO_OFFSET modified. If no need to connect to ZMIO310 sub-module, please set it as negative value or big value (over IN and OP starting No.), and it must be the times of 8. For example, ZMIO310_OFFSET = -8 (no submodule after XPLC300), ZMIO310_OFFSET = 560 (for XPLC300, its max starting range is 0-511), otherwise, it will report an error of 201.
Syntax	ZMIO310_OFFSET = value value: IO starting address, the default value is 32 Note: it can only be a multiple of 8.
Controller	XPLC300, ZMC432M.
Example	ZMIO_OFFSET = 48 'offset controller ZMIO expanded IO to 48.
Instruction	ZMAIO_OFFSET

ZMAIO_OFFSET -- ZMIO Extended AIO Offset

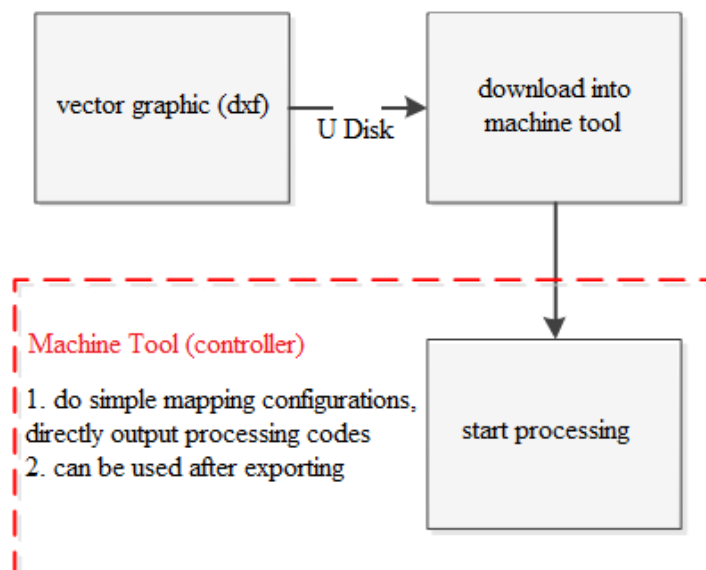
Type	The RS485 bus command of XPLC300/ZMC432M local slave interface.
Description	This is used for offset the controller itself ZMIO expanded AIO address. Please note restart after ZMAIO_OFFSET modified.
Syntax	ZMAIO310_OFFSET = value ● value: AIO starting address, the default value is 32
Controller	XPLC300, ZMC432M.
Example	ZMIO_OFFSET = 33 'offset controller ZMIO expanded AIO to 33.
Instruction	ZMIO_OFFSET

Chapter 21 CAD Controls & Instructions

Currently, for most machine tool machining, it obtains processed vector graphic (like, dxf), then generates corresponding processing codes through PC related mapping software, even modifies codes by PC, then downloads codes into machine tool controller for operation. It can be known PC is needed while using machine tool



If no need PC, integrating CAD mapping function into machine tool (controller) to reduce the cost and simplify the process. You only need to get processing vector graphic, then directly export processing codes in controller. Even not to export codes, processing the graphic (dxf) directly to complete actions.



✓ Property of “use after exporting”

The mapping function is integrated in the machine tool (controller), exported processing codes are

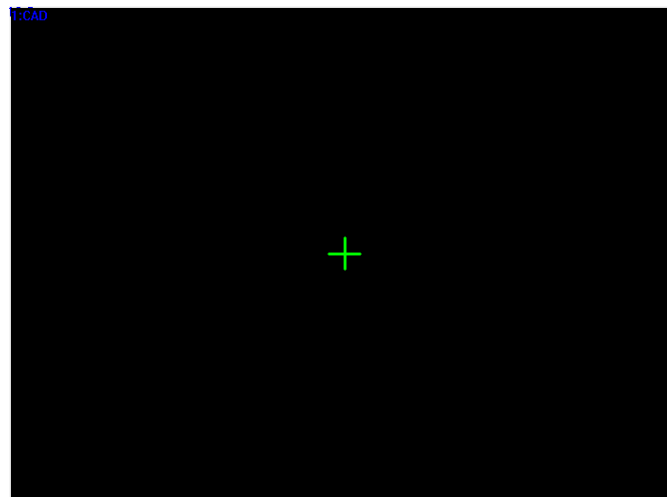
highly compatible with machine tool, that is, it can be used directly after exported, no need to modify again, convenient enough.

✓ Controller + Touch Screen

Due to the operation platform of **controller + touch screen**, then **simple operations** can **achieve normal CAD mapping** (remove the configurations that are complex and not general), at the same time, it can **reserve editable interfaces and required parameters configuration interfaces** (advanced functions) for you.

- **Note: this function is only valid in VPLC5XX controllers.**

21.1. CAD Controls Description



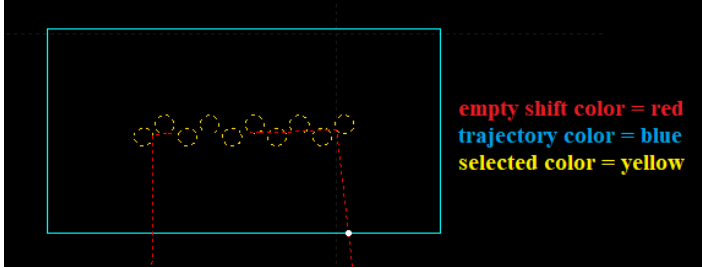
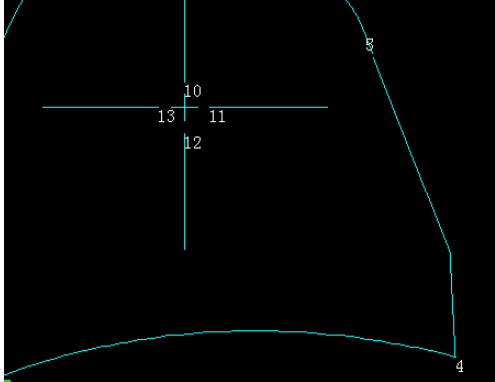
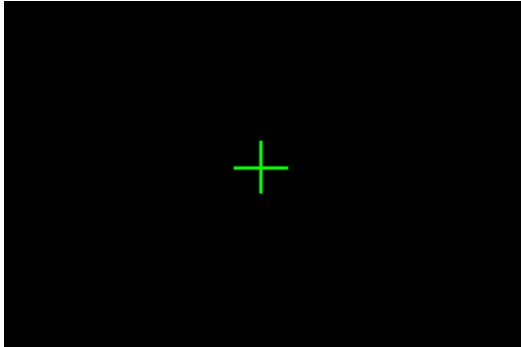
➤ Basic Properties

No., name, layer, starting position X, starting position Y, width, height, valid display, valid control (when it is TRUE, valid properties are device No., register type, register No.).

➤ Other Properties

Property	Description
Channel No.	Default is 0. The first version only provides one channel.
If edge	When it is TRUE, properties are edge color, edge width.
Background color	RGB (please be high comparison with other colors)
If use layer color	0-1 (false/true) When it is 0, which means false, the trajectory color property takes effect. When it is 1, which means true, use system inner layer color, ignore the trajectory color.



Trajectory color	RGB, default is the sky blue. When this property takes effect, all layers use trajectory color.
Empty shift color	RGB, default is the red. The connecting segment between 2 continuous trajectory segments use empty shift color.
Selected color	RGB, default is yellow. 
Format text (mark /flag)	Format text (only one state) The font color is the displaying color of continuous curve's starting point, and the sequence No., for the font, and the font size show sequence No.  Format text's alignment, background color, format properties are invalid, only font color, font size, font are effective.
Auxiliary color	RGB, the default color is green. Like, origin coordination axis, selection box, etc. 

21.2. CAD Basic Instructions

Through below basic commands, it can achieve G code CAD applications, or do some simple CAD mapping.

CAD_CHANNEL -- CAD Channel

Now version only has one CAD channel, even if you use several CAD controls, they are for one same CAD channel. That is, in one CAD channel, there are different visual windows of multiple CAD controls.

Type	CAD Commands
Description	Set / get the current CAD channel. After switching channel, all CAD commands are used to operate the channel that is switched.
Syntax	CAD_CHANNEL = channel channel = CAD_CHANNEL channel = current CAD channel, default is 0
Controller	Controller that supports RTHMI
Example	CAD_CHANNEL = 0 'switch to channel 0 ?CAD_CHANNEL '=' > 0

CAD_LOADFILE -- Graphic Load from File

Type	CAD Commands
Description	Set / get current loading graphic file, then show it in CAD control. Valid formats: dxf, plt, dst, ai, nc, cnc. Please note it doesn't support big file (<10M) loading for current version. And if the graphic file > 5M, the loading time will be long, even appear "disconnection", you need to connect again. Therefore, recommend <5M.
Syntax	CAD_LOADFILE (dxfname [,refdis[, layers]]) dxfname = CAD_LOADFFILE() - dxfname: dxf file name, be with the path - refdis: import complex curve's segment accuracy, [0.01~0.5] this segment accuracy determines whether it is segmented as segments automatically while importing complex curves. When "refdis" is default value / is 0, it won't segment. Otherwise, be segmented according to the value. - layers: select which layer to be loaded, bit0~bit31 are valid, up to 32 layers. It loads all layers by default (valid when loading G code file).
Controller	The controller that supports RTHMI function.
Example	Example 1: CAD_LOADFILE ("zheng.dxf") 'load "zheng.dxf" file ?CAD_LOADFILE () '=' > zheng.dxf Example 2: CAD_LOADFILE ("") 'loaded file doesn't exist, [CAD] content empty ?CAD_LOADFILE () '=' >

	Example 3: CAD_LOADFILE (“zhenng.dxf”, 0.05, 1) 'reference accuracy = 0.05, only layer 1 is loaded ?CAD_LOADFILE () '= > zheng.dxf

CAD_LOADTABLE -- Graphic Load from TABLE

Type	CAD Commands
Description	Load graphic data from TABLE register. Before importing, please adjust TABLE data according to TABLE data formats.
Syntax	CAD_LOADTABLE (tabid) tabid = CAD_LOADTABLE () ● tabid: table subscript
Controller	The controller that supports RTHMI function.
Example	CAD_LOADTABLE (10000) = 0 'load the graphic starting from TABLE (10000) ?CAD_LOADTABLE() '= > 10000

CAD_LOADBIN -- Graphic Load from BIN File

Type	CAD Commands
Description	Load graphic data from BIN file. For data format, you can refer to TABLE data format.
Syntax	CAD_LOADBIN (filename, type) filename = CAD_LOADBIN() ● filename: bin file name, be with the path ● type: BIN file data format 0 – float type, 4 bytes of each item 1 – double type, 8 bytes of each item
Controller	The controller that supports RTHMI function.
Example	CAD_LOADBIN (“ZHENG.BIN”, 0) 'load BIN file, the type is float

CAD_CLOSE -- CAD Graphic Loading OFF

Type	CAD Commands
Description	Close current loading CAD graphic file.
Syntax	CAD_CLOSE ()
Controller	The controller that supports RTHMI function.
Example	CAD_LOADFILE (“zhenng.dxf”) 'load “zheng.dxf” file CAD_CLOSE() 'close loading

CAD_TOCODE – Export Code File (NC / Z3P)

Type	CAD Commands
Description	Make current loading graphic export as code file. It can export NC file and Z3P file.
Syntax	CAD_TOCODE (filename, type, layers) ● filename: exported code file name xxx.z3p file outputs Z3P file of basic xxx.nc file outputs NC file of G code when filename = “”, it directly outputs to current file 3 (Z3P) file loading area use FILE3_CHANNEL command to assign the Z3P channel ● type: exported file type, 0 – Z3P (Basic), 1 – NC (G code) ● layers: exported layer, bit0~bit31 are valid, -1 means export all layers, for example, layers = 1+2+4+8, which means export layer 0, 1, 2, 3
Controller	The controller that supports RTHMI function.
Example	CAD_LOADFILE (“zheng.dxf”) 'load “zheng.dxf” file CAD_TOCODE (“zheng.nc”, 1, -1) 'export zheng.nc file 'output NC codes to file 3 (Z3P) channel 1 loading area, if the loading area is not enough, it will alarm. FILE3_CHANNEL=1 CAD_TOCODE(“”, 1, -1)

CAD_TABSIZE -- TABLE Space Occupy

Type	CAD Commands
Description	Get the TABLE size (table numbers) occupied by current loading vector graphic while exporting to TABLE. That is, check whether the TABLE space is enough for exported graphic.
Syntax	tabsize = CAD_TABSIZE (layers) ● layers: exported layer, bit0~bit31 are valid, -1 means export all layers. ● tabdize: required TABLE size when exporting graphic to TABLE
Controller	The controller that supports RTHMI function.
Example	CAD_LOADFILE (“zheng.dxf”) 'load “zheng.dxf” file Tabsize = CAD_TABSIZE (-1) 'get size of exported zheng.dxf graphic in TABLE

CAD_TOTABLE -- TABLE Data Export

Type	CAD Commands
Description	Output current loaded graphic file to TABLE according to standard format, you can read data from TABLE, then move.

	Please confirm whether there is enough TABLE space for saving DXF.
Syntax	CAD_TOTABLE (datatabid, layers) ● datatabid: table address, output table data (for table data storage format, please check TABLE Data Format Design). ● layers: exported layer, bit0~bit31 are valid, -1 means export all layers, for example, layers = 1+2+4+8, which means export layer 0, 1, 2, 3
Controller	The controller that supports RTHMI function.
Example	<pre> CAD_LOADFILE ("zhenng.dxf") 'load "zheng.dxf" file Tabsize = CAD_TABSIZE (-1) 'get size of exported zheng.dxf graphic in TABLE if tabsize < 300000 then 'can be exported if it is <300000 CAD_TOTABLE (20000, -1) 'export data after TABLE (20000) endif CAD_CLOSE() 'close graphic loading </pre>

CAD_TOBIN -- BIN File Export

Type	CAD Commands
Description	Output current loaded graphic file to BIN file according to standard format.
Syntax	CAD_TOBIN (filename, type, layers) ● filename: output BIN file name ● type: BIN file data type 0 – float type, 4 bytes of each item 1 – double type, 8 bytes of each item ● layers: exported layer, bit0~bit31 are valid, -1 means export all layers, for example, layers = 1+2+4+8, which means export layer 0, 1, 2, 3
Controller	The controller that supports RTHMI function.
Example	<pre> CAD_LOADFILE ("zhenng.dxf") 'load "zheng.dxf" file CAD_TOBIN (zheng.BIN, 0, -1) 'export to BIN file CAD_CLOSE() 'close graphic loading </pre>

CAD_TAB2CODE -- Code File (from TABLE Data)

Type	CAD Commands
Description	Export TABLE graphic trajectory as code file. Please confirm whether dxf trajectory in TABLE is normal, it can export as NC file and Z3P file.
Syntax	CAD_TAB2CODE (filename, type, tabid, layers) ● filename: exported code file name ● type: exported code type, 0 – Z3P (Basic), 1 – NC (G Code) ● tabid: table index ● layers: exported layer, bit0~bit31 are valid, -1 means export all layers, for example, layers = 1+2+4+8, which means export layer 0, 1, 2, 3

Controller	The controller that supports RTHMI function.
Example	'export the graphic that was saved into TABLE (3000) as G code CAD_TAB2CODE ("xxx.nc", 1, 3000, -1)

CAD_PARA -- Parameters Settings

Type	CAD Commands
Description	Set CAD processing parameters. When the parameter of bIfUseSel is default value / is 0, this command is for global, otherwise, for CAD selected part. Note: G code exporting and Basic exporting are different, for G code, no need to set axis No., empty shift speed, acceleration, deceleration, etc.
Syntax	CAD_PARA (para, setval [,bIfUseSel]) ● para: set parameter No., please see CAD Parameters List for details. ● setval: the value to be set ● bIfUseSel: reserved, whether this command is only for selected part or not, default 0 means global. setval: CAD_PARA (para [,bIfUseSel]) ● para: set parameter command, same as above ● bIfUseSel: reserved, whether this command is only for selected part or not, default 0 means global. ● setval: return to the latest configured parameter value
Controller	The controller that supports RTHMI function.
Example	CAD_PARA (12, 1) 'set parameter 12 (whether to invert) as 1 ?CAD_PARA (12) '=>1

CAD_DISPLAY -- Display Settings

Type	CAD Commands
Description	Set CAD displaying method.
Syntax	CAD_DISPLAY (mode) mode = CAD_DISPLAY () ● mode: default is 0 when power on. - Bit0: 1, whether to show not-closed graphic - Bit1: 2, whether to show empty shift - Bit2: 4, whether to show sequence - Bit3: 8, whether to show starting point - Bit4: 16, whether to show the direction - Bit5: 32, whether to show highlight for action point
Controller	The controller that supports HMI function.
Example	CAD_DISPLAY (2+8) 'show the empty shift + starting point

CAD_MOUSE -- Mouse Mode Settings

Type	CAD Commands
Description	Set CAD displaying method.
Syntax	CAD_MOUSE (mode) mode = CAD_MOUSE () - mode: mouse control mode, default is 0 when power on. 0: view panning 1: view zooming, move up -- zoom in, move down -- zoom out 2: trajectory selection, mouse clicked / selected area trajectory will be selected. 3: origin setting, mouse clicked position is set as origin Note: when mode = 2, there are 2 box selection methods to select trajectory: right-dragged selection box: Requires the graphic to be fully enclosed to be selected; left-dragged selection box: Selects graphics with any partial overlap.
Controller	The controller that supports RTHMI function.
Example	CAD_MOUSE (2) 'set mouse control, and select trajectory mode

CAD_AUTOVIEW -- Auto-View

Type	CAD Commands
Description	Automatically adapt and show the view, use this command to automatically show in center, and zoom in view range.
Syntax	CAD_AUTOVIEW (winId, controlId [,offX, offY [,fZoom [,plane [,RotX, RotY, RotZ]]]]) - winId, controlId: the window No. of control that is to be set, and control No. - offX, offY: view offset, default (0, 0), view is shown in center - fZoom: scaling ratio, default 1.0, it is shown in range below properties are reserved for 3D CAD graphic (now version doesn't support) - plane: showing plane, default is 0-XY, 1-YZ, 3-ZX - RotX, RotY, RotZ: the rotary angle around XYZ axes, 0-360, default 0
Controller	The controller that supports RTHMI function.
Example	'show all graphics completely in the center within auto view range, the CAD control is in window 10, No. is 1 CAD_AUTOVIEW (10, 1)

CAD_RANGE -- Trajectory Range Get

Type	CAD Commands

Description	Get current loaded / TABLE assigned graphic data range												
Syntax	CAD_RANGE (tabout [,tabin]) ● tabout: output results to TABLE (tabout) ~ TABLE (table+5) <table border="1"> <tr> <td>TABLE (tabout)</td><td>Graphic trajectory's min axis X position</td></tr> <tr> <td>TABLE (tabout+1)</td><td>Graphic trajectory's min axis Y position</td></tr> <tr> <td>TABLE (tabout+2)</td><td>Graphic trajectory's axis X width</td></tr> <tr> <td>TABLE (tabout+3)</td><td>Graphic trajectory's axis Y width</td></tr> <tr> <td>TABLE (tabout+4)</td><td>Graphic trajectory's min axis Z position, reserved</td></tr> <tr> <td>TABLE (tabout+5)</td><td>Graphic trajectory's axis Z height, reserved</td></tr> </table> ● tabin: TABLE graphic data starting position, the default is -1 Note: if the tabin is invalid table subscript, then take current CAD loaded graphic data as reference, otherwise, take TABLE graphic data as reference.	TABLE (tabout)	Graphic trajectory's min axis X position	TABLE (tabout+1)	Graphic trajectory's min axis Y position	TABLE (tabout+2)	Graphic trajectory's axis X width	TABLE (tabout+3)	Graphic trajectory's axis Y width	TABLE (tabout+4)	Graphic trajectory's min axis Z position, reserved	TABLE (tabout+5)	Graphic trajectory's axis Z height, reserved
TABLE (tabout)	Graphic trajectory's min axis X position												
TABLE (tabout+1)	Graphic trajectory's min axis Y position												
TABLE (tabout+2)	Graphic trajectory's axis X width												
TABLE (tabout+3)	Graphic trajectory's axis Y width												
TABLE (tabout+4)	Graphic trajectory's min axis Z position, reserved												
TABLE (tabout+5)	Graphic trajectory's axis Z height, reserved												
Controller	The controller that supports HMI function.												
Example	CAD_RANGE (0) 'get current loaded graphic range ?"left = ", table(0) ?"bottom = ", table(1) ?"width = ", table(3) ?"height = ", table(4)												

21.3. CAD “Custom” Control Extended Instructions

CAD_SETCOLOR -- CAD Drawing Color Set

Type	CAD Commands
Description	In “custom” control, draw current CAD loaded graphic. It only can be used in custom control's refresh/draw sub.
Syntax	CAD_SETCOLOR (type, color) ● type: color type 0- Trajectory color, default is -1, using inner layer color 1- Empty shift color, default is red RGB (255, 0, 0) 2- Selected color, default is yellow RGB (255, 215, 0) 3- Mark color (starting point / sequence), default is white RGB (255, 255, 255) ● color: set the color Note: only when the trajectory color is set as -1, it is valid (it can use inside layer color)
Controller	The controller that supports RTHMI function.
Example	CAD_SETCOLOR (1, RGB (255,0,0)) 'set the empty trajectory as red

CAD_DRAW -- CAD Graphic Draw

Type	CAD Commands
Description	In custom control, draw current CAD loaded graphic. It only can be used in custom control's draw sub. The display mode is controlled by CAD_DISPLAY command.
Syntax	CAD_DRAW (offsetX, offset, fZoom [,plane, RotX, RotY, RotZ]) - offset1, offset2: graphic offset, when = (0, 0), the graphic center is in custom control center position. - fZoom: scaling ratio, 1.0 means the graphic is shown fully within custom control range. The scaling range is 0.002 ~ 1000.0 below properties are reserved for 3D CAD graphic: - plane: showing plane, default is 0-XY, 1-YZ, 3-ZX - RotX, RotY, RotZ: the rotary angle around XYZ axes, 0-360, default 0
Controller	The controller that supports RTHMI function.
Example	CAD_DRAW (0, 0, 1, 0) 'show CAD graphic center in custom control

CAD_DRAWTAB -- CAD Graphics Draw (from TABLE)

Type	CAD Commands
Description	Draw graphics that are saved into TABLE in "custom" control. It only can be used in custom control's draw sub. The display modes are controlled by the instruction CAD_DISPLAY, but it doesn't mode bit0, bit5.
Syntax	CAD_DRAWTAB (tabid, offsetX, offsetY, fZoom [,plane, RotX, RotY, RotZ]) - tabid: table subscript, TABLE position where saves the graphic; - offset1, offset2: displayed graphic offset, when it is (0, 0), the graphic center is in custom control center position. - fZoom: scaling ratio, 1.0 means the graphic is shown fully within custom control range. below properties are reserved for 3D CAD graphic: - plane: showing plane, default is 0-XY, 1-YZ, 3-ZX - RotX, RotY, RotZ: the rotary angle around XYZ axes, 0-360, default 0
Controller	The controller that supports RTHMI function.
Example	'show TABLE graphic in "custom" control center position CAD_DRAWTAB (3000, 0, 0, 1.0)

CAD_TRANSPPOINT -- Convert to Pixel Coordinates

Type	CAD Commands
Description	Convert data coordinates to current custom control's pixel position (X, Y) It only can be used in custom control's draw sub.

Syntax	CAD_TRANSPOINT (tableout, x, y, z, offsetX, offset, fZoom [,tabin] [,plane, RotX, RotY, RotZ]) ● tableout: output the result to TABLE (tabout)=X, TABLE (tabout+1)=Y ● x,y,z: the coordinates to be converted, axis z is reserved for 3D ● offset1, offset2: graphic offset, when = (0, 0), the graphic center is in custom control center position. ● fZoom: scaling ratio, 1.0 means the graphic is shown fully within custom control range. ● tabin: it takes current loaded data as reference (by default), otherwise, take TABLE data as reference. below properties are reserved for 3D CAD graphic: ● plane: showing plane, default is 0-XY, 1-YZ, 3-ZX ● RotX, RotY, RotZ: the rotary angle around XYZ axes, 0-360, default 0
Controller	The controller that supports RTHMI function.
Example	CAD_DRAW (g_offsetX, g_offsetY, g_fZoom) 'draw current load graphic 'take current loading graphic as reference, get coordinates (10,10,0)'s pixel position in custom control CAD_TRANPOINT (0, 10, 10, 0, g_offsetX, g_offsetY, g_fZoom)

CAD_TRANSPOINT2 -- Convert to Graphic Coordinates

Type	CAD Commands
Description	Convert current custom control's pixel position (px, py) to data coordinates. It only can be used in custom control's refresh/draw sub.
Syntax	CAD_TRANSPOINT2 (tableout, px, py, centerX, centerY, fZoom [,tabin] [,plane, RotX, RotY, RotZ]) ● tableout: output the result to TABLE (tabout)=X, TABLE (tabout+1)=Y ● px,py: the pixel coordinates to be converted ● centerX, centerY: view center offset, when = (0, 0), the graphic center is in custom control center position. ● fZoom: scaling ratio, 1.0 means the graphic is shown fully within custom control range. ● tabin: TABLE graphic data starting position, the default is -1 (note: if the tabin is invalid table subscript, then use current CAD loading graphic data as reference, otherwise, use this TABLE graphic data as reference) below properties are reserved for 3D CAD graphic: ● plane: showing plane, default is 0-XY, 1-YZ, 3-ZX ● RotX, RotY, RotZ: the rotary angle around XYZ axes, 0-360, default 0
Controller	The controller that supports RTHMI function.
Example	CAD_DRAW (g_offsetX, g_offsetY, g_fZoom) 'draw current load graphic 'take current loading graphic as reference, get coordinates (10,10,0)'s pixel position in custom control CAD_TRANPOINT2 (0, 10, 10, g_offsetX, g_offsetY, g_fZoom)

CAD_SELECT -- Graphic Box Select

Type	CAD Commands
Description	Point select / box select CAD loading graphic with custom control. <i>It only can be used in custom control's refresh sub.</i>
Syntax	CAD_SELECT (ix1, iy1, ix2, iy2, offsetX, offset, fZoom [,plane, RotX, RotY, RotZ]) - ix1, iy1, ix2, iy2: selection box's 2 top coordinates (2 points make one rectangle) - offset1, offset2: graphic offset, when = (0, 0), the graphic center is in custom control center position. - fZoom: scaling ratio, 1.0 means the graphic is shown fully within custom control range. <i>below properties are reserved for 3D CAD graphic:</i> - plane: showing plane, default is 0-XY, 1-YZ, 3-ZX - RotX, RotY, RotZ: the rotary angle around XYZ axes, 0-360, default 0
Controller	The controller that supports HMI function.
Example	CAD_SELECT (10, 10, 90, 90, 0, 0, 1, 0) 'box select the area from (10, 10) to (90, 90)

21.4. CAD Advanced Instructions

For advanced commands, CAD mapping data can be modified. There are intelligent processing commands, which can automatically export corresponding data codes according to different industries and technologies.

CAD_MACHTYPE -- Machine Type

As for different non-standard devices, you only need to add one machine type according to special structure requirements. After that, no need other complex configurations. *Then, this command is used to set the machine type, after it is added, generating matched processing codes automatically while exporting codes according to the set machine type.*

Type	CAD Commands				
Description	Set machine type, CAD will output corresponding machine types' processing codes.				
Syntax	CAD_MACHTYPE (type [,strAxes [,axis[,axis2[,...]]]]) type: machine type <table border="1"> <tr> <td>0</td><td>General 3-axis machine tool (XYZ), default</td></tr> <tr> <td>1</td><td>Special 3-axis drilling machine (XYZ), it can filter the curve automatically, only export drilling related codes.</td></tr> </table>	0	General 3-axis machine tool (XYZ), default	1	Special 3-axis drilling machine (XYZ), it can filter the curve automatically, only export drilling related codes.
0	General 3-axis machine tool (XYZ), default				
1	Special 3-axis drilling machine (XYZ), it can filter the curve automatically, only export drilling related codes.				

	11	General 2-axis machine tool (XY)
	If you need other types, please contact us. ● strAxes: re-assign which axis name to be exported, it is valid when exporting G code. (recommend use XYZABCUVW, now A-Z supported) for example, when the type is 4-axis machine tool, specify strAxes = "XYZC", which means use linear axes XYZ, rotary axis C. ● Axis1~N: re-assign basic axis No., it is valid when exporting Basic, default is 0, 1, 2, ...	
Controller	The controller that supports RTHMI function.	
Example	'set as general 3-axis type, and make the coordinate axis name as XYZ, bind axis No.0, No.1, and No.2. CAD_MACHTYPE (0, "XYZ", 0, 1, 2)	

CAD_OPTIMIZE -- Trajectory Optimize

Type	CAD Commands																				
Description	Optimize dxf graphic trajectory. When bIfUseSel is default / is 0, this command is for global, otherwise, it is for CAD selected part.																				
Syntax	CAD_OPTIMIZE (option, optval [,bIfUseSel]) ● option: optimization items 0: remove extreme small graphic's max size (a*a) 1: whether to remove repeated lines, 0 / 1 2: merge connecting lines, the max distance it can be merged 3: segment the curve, enter the segment accuracy (0.01~0.5) 4: optimize empty shift, enter reference accuracy (0.01~0.5) 5: path sequence <table border="1"> <tr><td>0</td><td>Not sort</td></tr> <tr><td>1</td><td>Grid sort</td></tr> <tr><td>2</td><td>Sort by min empty shift locally</td></tr> <tr><td>3</td><td>Small graphic optimizes</td></tr> <tr><td>4</td><td>From inside to outside</td></tr> <tr><td>5</td><td>From left to right</td></tr> <tr><td>6</td><td>From right to left</td></tr> <tr><td>7</td><td>From up to bottom</td></tr> <tr><td>8</td><td>From bottom to up</td></tr> <tr><td>9</td><td>Sort inversely</td></tr> </table> 6: curve smooth (reserved), enter smooth accuracy 7: invert the path, input parameters are invalid ● optval: optimization value, assign it according to option ● bIfUseSel: whether to assign selected part, default 0 = ignore selected part, for global.	0	Not sort	1	Grid sort	2	Sort by min empty shift locally	3	Small graphic optimizes	4	From inside to outside	5	From left to right	6	From right to left	7	From up to bottom	8	From bottom to up	9	Sort inversely
0	Not sort																				
1	Grid sort																				
2	Sort by min empty shift locally																				
3	Small graphic optimizes																				
4	From inside to outside																				
5	From left to right																				
6	From right to left																				
7	From up to bottom																				
8	From bottom to up																				
9	Sort inversely																				
Controller	The controller that supports RTHMI function.																				
Example	CAD_OPTIMIZE (3, 0.05)																				

	'curve segmentation, the reference accuracy is 0.05

CAD_EDIT -- Custom Edit

Some technologies may not be exported while outputting codes automatically, then you can use this instruction to complete it manually.

Type	CAD Commands
Description	Edit output file, insert action character string at specified editing position. When CAD PARA(13) =1, it only can edit selected trajectory, otherwise, for global.
Syntax	CAD_EDIT (EditPos, action) ● EditPos: where you need to edit 0: file / selected area's starting position 1: file / selected area's end position 2: file / selected area's each continuous starting position 3: file / selected area's each continuous end position ● action: edit the inserted action (No., please create action data at first) Note: action No. starts from 1, 0 means empty action, that is, insert action 0 = delete action action = CAD_EDIT (EditPos, [,bIfUseSel]) ● EditPos: edit the position, same as above ● returned action: the latest inserted action (No.)
Controller	The controller that supports RTHMI function.
Example	CAD_PARA (13, 0) CAD_EDIT (0, 10) 'insert action 10 at the graphic's starting position

CAD_MOVE -- Trajectory Translate

Type	CAD Commands
Description	Move the CAD imported trajectory linearly in X / Y. When CAD PARA(13) =1, it only can edit selected trajectory, otherwise, it is for global.
Syntax	CAD_MOVE (x, y, [,z]) ● x,y: relative translation coordinates ● z: reserved for 3D CAD graphic
Controller	The controller that supports RTHMI function.
Example	CAD_MOVE(100, 0) 'right move 100

CAD_SCALE -- Trajectory Scaling

Type	CAD Commands
Description	Scale the CAD imported trajectory When CAD_PARA(13)=1 , it only can edit selected trajectory, otherwise, it is for global.
Syntax	CAD_SCALE (fscaleX, fscaleY, [,x, y[,z]]) ● fscaleX: scaling in X direction ● fscaleY: scaling in Y direction ● x,y: scaling center position, default is the graphic's center ● z: reserved for 3D CAD graphic
Controller	The controller that supports RTHMI function.
Example	CAD_SCALE(2.0, 2.0) 'scale in 2 times

CAD_ROTATE -- Trajectory Rotate

Type	CAD Commands
Description	Rotate the CAD imported trajectory When CAD_PARA(13)=1 , it only can edit selected trajectory, otherwise, for global.
Syntax	CAD_ROTATE (fRotAngle [,center1, center2 [,plane]]) ● fRotAngle: rotation angle, the radian value ● center1, center2: rotation center position, the default is graphic center below is reserved for 3D CAD graphic. ● plane: rotary plane, default: 0-XY, 1-YZ, 2-ZX
Controller	The controller that supports RTHMI function.
Example	CAD_ROTATE(90*PI/180) 'rotate 90 degrees around graphic center

CAD_DELETE -- Trajectory Delete

Type	CAD Commands
Description	Select the graphic trajectory, then delete it. When CAD_PARA(13)=1 , it only can edit selected trajectory, otherwise, it is for global.
Syntax	CAD_DELETE
Controller	The controller that supports RTHMI function.
Example	CAD_PARA (13, 1) CAD_DELETE() 'delete selected trajectory

CAD_ACTION -- Action Data Get / Set

Type	CAD Commands

Description	Get / modify the action data of specified action No. The action No. starts from 1.
Syntax	CAD_ACTION (action, strAct) ● action: specify action No. ● strAct: write the action data that is to be written (character string, it supports multiple row) strAct = CAD_ACTION (action) ● action: specify action No. ● returned strAct: action data (character string)
Controller	The controller that supports RTHMI function.
Example	CAD_ACTION (10, "action 10") 'modify action 10's action ?CAD_ACTION (10) 'print action 10's action

CAD_ACTIONADD -- Action Add

Type	CAD Commands
Description	Add action data, when operated successfully, return action No., if not, return -1.
Syntax	action = CAD_ACTIONADD (strAct) ● strAct: action data (character string, it supports multiple row) ● returned action: action data
Controller	The controller that supports RTHMI function.
Example	action=CAD_ACTIONADD ("M7; open water") 'add "open water" action, the No. is action

CAD_ACTIONDEL -- Action Data Delete

Type	CAD Commands
Description	Delete the action data of specified action No.
Syntax	CAD_ACTIONDEL([action]) ● action: specify action No., default: delete all
Controller	The controller that supports RTHMI function.
Example	CAD_ACTIONDEL (10) 'delete action 10

CAD_UNDO -- Last Step Undo

Type	CAD Commands
Description	Undo the last editing operation.
Syntax	CAD_UNDO ()
Controller	The controller that supports RTHMI function.
Example	/

CAD_REDO -- Last Step Redo

Type	CAD Commands
Description	Redo the last editing operation.
Syntax	CAD_UNDO ()
Controller	The controller that supports RTHMI function.
Example	/

CAD_SORT -- Sort Manual

Type	CAD Commands												
Description	Manually sort current selected graphics. Note: it only can manually sort selected, it doesn't support undo and redo.												
Syntax	CAD_SORT (mode [,seqnum]) - mode: sorting mode <table border="1"> <tr> <td>0</td><td>Manually set selected graphic's sequence. Note: it only supports single selection, when there are several graphics selected, only the front one will be operated.</td></tr> <tr> <td>1</td><td>Make the selected graphic move forward one time</td></tr> <tr> <td>2</td><td>Make the selected graphic move back one time</td></tr> <tr> <td>3</td><td>Make the selected graphic move to be the first one</td></tr> <tr> <td>4</td><td>Make the selected graphic move to be the last one</td></tr> <tr> <td>5</td><td>Make the selected graphic inversely, it is valid when in multi-selection.</td></tr> </table> - seqnum: selectable, it is valid when mode = 0, set the sequence No. of selected graphic.	0	Manually set selected graphic's sequence. Note: it only supports single selection, when there are several graphics selected, only the front one will be operated.	1	Make the selected graphic move forward one time	2	Make the selected graphic move back one time	3	Make the selected graphic move to be the first one	4	Make the selected graphic move to be the last one	5	Make the selected graphic inversely, it is valid when in multi-selection.
0	Manually set selected graphic's sequence. Note: it only supports single selection, when there are several graphics selected, only the front one will be operated.												
1	Make the selected graphic move forward one time												
2	Make the selected graphic move back one time												
3	Make the selected graphic move to be the first one												
4	Make the selected graphic move to be the last one												
5	Make the selected graphic inversely, it is valid when in multi-selection.												
Controller	The controller that supports HMI function.												
Example	CAD_SORT (3) 'move the selected graphic in the front												

21.5. CAD Parameters List

General Parameters

Parameter No.	Description	Range	Initial Value
1	Motion mode: 0-absolute motion, 1-relative motion	[0, 1]	0
2	Machine type: ➤ 0: general 3-axis machine tool (XYZ) ➤ 1: special 3-axis machine tool, filter	-	0

	the curve, just do drilling ➤ 11: general 2-axis machine tool (XY) ➤ 12: general 2-axis machine tool (ZX), reserve ➤ 101: special 4-axis machine tool, tool tip automatically follows with cutting direction, reserved		
3	Processing speed, the speed of cutting graphic. ➤ unit of exporting G code: mm/min ➤ unit of exporting Basic: units/s	[0, 999999]	200.0
4	Processing height	[-999999, 999999]	1.0
6	Lifting height		10.0
5	Whether to lift up while processing	[0, 1]	0
7	Whether to back to machine origin before & after processing	[0, 1]	0
8	Whether convert the arc to small segment	[0, 1]	0
9	The curve segment reference accuracy, 0.01 ~ 0.5	[0.01, 0.5]	0.05
10	Processing plane, 0-XY, 1-YZ, 2-ZX. Reserved	[0, 2]	0
11	Processing plane radius compensation value, be calculated to exported codes. ➤ 0: not compensate ➤ >0: right compensation ➤ <0: left compensation Reserved	[-999999, 999999]	0
12	Allow invert for sorting optimization?	[0, 1]	0
13	Whether the editing function is only for selected part. ➤ Valid commands: CAD_MOVE, CAD_SCALE, CAD_ROTATE, CAD_DELETE, CAD_EDIT	[0, 1]	0
14	Whether only the trajectory part is saved while exporting TABLE, BIN files.	[0, 1]	0
20	Action No. before processing	[0, 999999]	0
21	Action No. after processing	[0, 999999]	0
50	Drilling technology, R point height (Z pos)	[-999999, 999999]	5.0
51	Drilling technology, drilling depth (Z pos)		0.0
52	Drilling speed (drill down)	[0, 999999]	200.0
53	Speed of drilling up, 0 means it moves up at empty shift speed	[0, 999999]	0
54	Action No. of hole bottom – the action to drill down to the bottom	[0, 999999]	0

G Code Parameters

Parameter No.	Description	Range	Initial Value
101	Whether to export row No.	[0, 1]	0
102	Used tool No., reserved	[0, 999999]	0
103	Tool radius compensation No., reserved	[0, 999999]	0
104	Tool length compensation No., reserved	[0, 999999]	0
105	Tool radius compensation direction: 1: right compensation (G42) -1: left compensation (G41) 0: not compensate Reserved	[-1, 1]	0
106	Tool length compensation direction: 1: forward compensation (G43) -1: reverse compensation (G44) 0: not compensate Reserved	[-1, 1]	0
120~136	Axis name	['A' ~ 'Z']	XYZ

Basic Parameters

Parameter No.	Description	Range	Initial Value
201	Processing acceleration	[-999999, 999999]	1000.0
202	Processing deceleration		
203	Empty shift speed		
204	Empty shift acceleration		
205	Empty shift deceleration		
220~236	Controller axis No.	[0~127]	0~2

21.6. TABLE Data Format

Export processing graphic data into TABLE assigned position according to set format, in this way, BASIC can read and process directly.

Data Format Designing

Tabid is the TABLE subscript that exports TABLE data, the formats are shown below:

tabid	tabid+1	tabid+2	tabid+3~	...	...	...	...
TableSize	DataType	ParaSize	Para1~ParaN	...	DataType	ParaSize	Para1~ParaN

The first table value is TableSize, that is, the size of Table data occupied

The second table value starts storing motion data, each motion data's format is: data type (DataType) + parameter numbers (ParaSize) + all parameter values (Para1~ParaN). And it can infer the next motion data address according to current motion data, until all are read (infer by Tablesize). For example, TableSize = 10

10	0	3	para1	para2	para3	1	2	para1	para2

Data Type Definition

DataType (integer type) includes mark type, motion type, and parameter configuration types, etc.

(1) Mark Type

0~99 are mark types, which are mainly used to do auxiliary, you can ignore them.

Also, you can specify action No. according to mark types, while analyzing table, you insert needed special processing.

DataType	Type Description	Parameter Description
0	The mark of ending	ParaSize: the number of parameters is 0. It means TABLE data ends. While reading TABLE data, TableSize can be used to judge whether it ends, also end mark (0) can do that.
1	The mark of the graphic's starting position	ParaSize: the number of parameters is N (0~1) When N=0, no parameter, no specified action data When N>0, there is specified action data (reserved) para1: specify action No. (reserved)
2	The mark of the graphic's end position	ParaSize: the number of parameters is N (0~1) para1: specify action No. (reserved)
3	The mark of each continuous segment's starting position	ParaSize: the number of parameters is N (0~1) para1: specify action No. (reserved)
4	The mark of each continuous segment's end position	ParaSize: the number of parameters is N (0~1) para1: specify action No. (reserved)
5	The mark of drilling bottom position	ParaSize: the number of parameters is N (0~1) para1: specify action No.
20	Layer No.	ParaSize: the number of parameters is 1 para1: graphic layer position (1~32)
21	Graphic trajectory range	ParaSize: the number of parameters is N (4~6) para1: min X coordinate para2: min Y coordinate para3: X direction width

		para4: Y direction width para5: min Z coordinate (reserved) para6: Z direction height (reserved)

(2) Motion Type

100~999 are motion types, generally, exported data are mainly linear segment, circular segment.

Data Type	Type Description	Parameter Description
100	Empty shift motion	ParaSize: the number of parameters is N para1~paraN: linear motion data Reference: MOVE(para1) AXIS(0) MOVE(para2) AXIS(1) MOVE(para3) AXIS(3)
101	Linear motion	ParaSize: the number of parameters is N para1~paraN: linear motion data Reference: MOVE (para1, para2, ..., paraN) MOVEABS (para1, para2, ..., paraN)
102	Circular motion	ParaSize: the number of parameters is N (5~6) para1, para2: end point coordinates para3, para4: circle center coordinates para5: 0-anticlockwise, 1-clockwise para6: the third axis' motion distance, it is needed when there is spiral. Reference: select different commands according to how many parameters MOVECIRC (para1, para2, para3, para4, para5) MHELICAL (para1, para2, para3, para4, para5, para6, 1) MOVECIRCABS (para1, para2, para3, para4, para5) MHELICALABS (para1, para2, para3, para4, para5, para6, 1)
103	Ellipse motion	ParaSize: the number of parameters is N (7~8) para1, para2: ellipse end point coordinates para3, para4: ellipse circle center coordinates para5: 0-anticlockwise, 1-clockwise para6, para7: ellipse radius of the first and second axis para8: the third axis' motion distance, it is needed when there is spiral. Reference: select different commands according to

		how many parameters MECLIPSE (para1, para2, para3, para4, para5, para7) MECLIPSE(para1, para2, para3, para4, para5, para7, para8) MECLIPSEABS (para1, para2, para3, para4, para5, para7) MECLIPSEABS(para1, para2, para3, para4, para5, para7, para8)
104	Motion delay	ParaSize: the number of parameters is N (1) para1: motion delay time ms Reference: MOVE_DELAY (para1)
200	Back to machine origin	ParaSize: the number of parameters is N (0)
201	Tool height positioning	ParaSize: the number of parameters is N (1) para1: axis Z height Reference: MOVEABS(Para1) AXIS(2)

For other needed motion types, which can be extended. Including some special machine types' switch actions.

(3) Motion Parameter Types

>1000 are parameter types, which are mainly running parameters that control motion data. Generally, you need to use MOVE_XXX commands to set buffer parameters.

Data Type	Type Description	Parameter Description
1000	Set relative / absolute mode	ParaSize: the number of parameters is 1 para1: 0 – absolute mode, 1 – relative mode
1001	Set speed	ParaSize: the number of parameters is 1-2 para1: speed value para2: axis No., default is all axes / main axis Reference: MOVE_PARA(SPEED, para2, para1) Note: Axis No. starts from 0 automatically, which is not related to controller axis No., please do mapping.
1002	Set acceleration	ParaSize: the number of parameters is 1-2 para1: speed value para2: axis No., default is all axes / main axis Reference:

		MOVE_PARA(ACCEL, para2, para1)
1003	Set deceleration	ParaSize: the number of parameters is 1-2 para1: speed value para2: axis No. Reference: MOVE_PARA(DECCEL, para2, para1)
1004	...	...

For other needed parameter types, which can be extended.

21.7. Machine Type Technology

General 3-Axis Machine Tool

Generally, the default machine tool is set as XYZ 3-axis type. The CAD graphic trajectory is the motion path. The CAD graphic provides XY trajectory, the axis Z is only for lifting the tool and cutting, specifically, moving between lift height and working height. When current continuous curve cutting completed, you can set one empty motion (arrive next cutting position), then lift up the tool before empty motion, when arrived, do cutting.

General 2-Axis Machine Tool

Generally, set it as XY 2-axis type. The CAD graphic trajectory is the motion path. There is no axis Z's tool lifting height and working height.

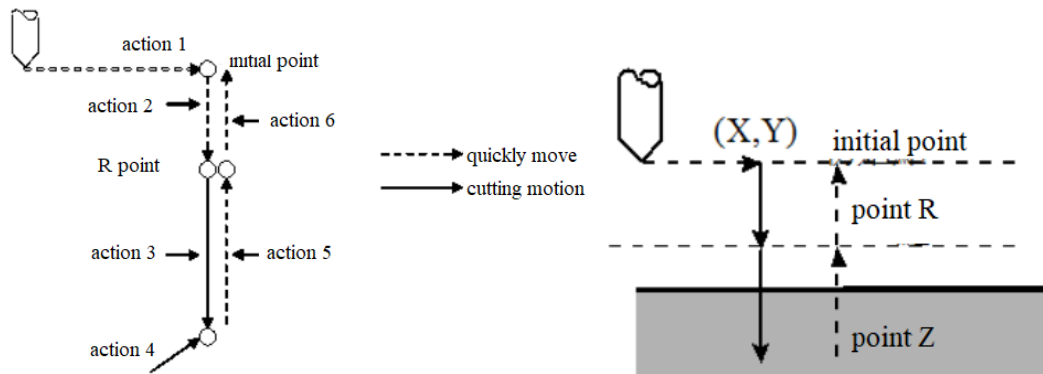
Special 3-Axis Drilling Machine Tool

After setting the special 3-axis drilling type, it will automatically filter non-circle trajectories of CAD graphic while exporting codes. That is, it only keeps the circle trajectory as drilling path. Among several circle trajectories, move it with tool lifting empty shift method. The drilling position is the circle center.

Drilling Processes (6 continuous actions):

- a) Quick move to axis X, axis Y positioning point (drilling point)
- b) Quick move to point R (R is controlled by parameter **Pr50**)

- c) Drilling (drill down, the drilling down speed is controlled by parameter **Pr52**, the depth is controlled by parameter **Pr51**)
- d) Act at hole bottom position (optional, controlled by parameter **Pr54**)
- e) Leave point R (leaving hole, the speed is controlled by parameter **Pr53**)
- f) Quick move to the initial point (optional, controlled by parameter **Pr5**)



Chapter 22 DT Motion Instructions

To support G code's variables parameters, DT motion commands are added. Use DT commands to call parameter that are in TABLE.

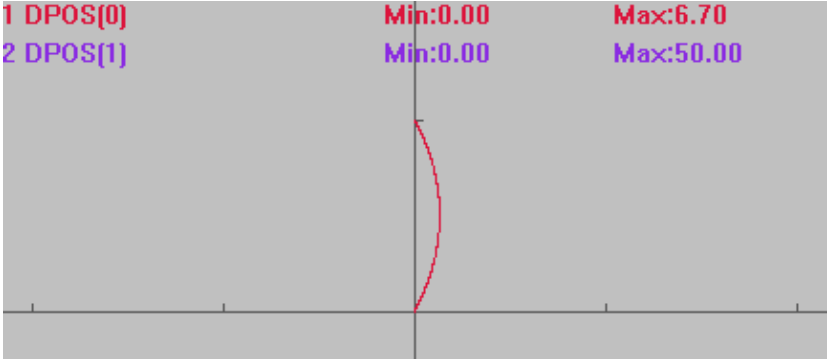
ABS suffix means absolute motion, without ABS means relative motion.

MOVEDTSP / MOVEDTABSSP -- DT Linear Motion

Type	DT motion command.
Description	Save axis No. and motion distance into TABLE separately. Then, do linear motion by TABLE list. Use bit to select axis No. of TABLE.
Syntax	MOVEDTSP (max axes, bit, axis DT list starting No., distance DT list starting No.)
Controller	Valid in ZMC4XX controllers and above.
Example	<pre>TABLE (10, 4, 5, 6) 'save axis 4, 5, 6 into TABLE(10) TABLE (20, 100, 50, -10) 'save three axes' distance into TABLE(20) WHILE 1 IF SCAN_EVENT (IN(0))>0 THEN MOVEDTSP (3, 5, 10, 20) 'move TABLE distance of each time MOVEDTABSSP (3, 5, 10, 20) 'move to position of TABLE 'bit is 5, converting to binary "0101", it only selects axis 4, axis 6. ENDIF WEND</pre>

MOVECIRCDTSP / MOVECIRCDTABSSP -- DT Arc

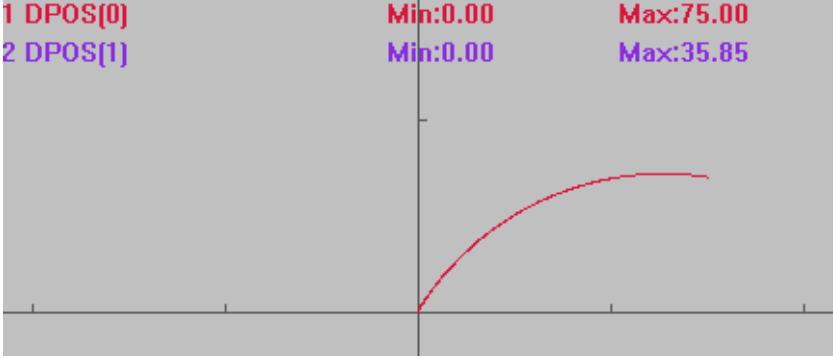
Type	DT motion command.
Description	Save axis No. and motion distance into TABLE separately. Then, do circular motion by TABLE list. Use bit to select axis No. of TABLE. According to current point (starting point) and middle point position, and radius to draw the circle, the circle center is calculated automatically. When the line distance between end and starting points is larger than radius, taking these two points to draw circle, then the radius is half of connecting distance, center the middle of connecting line.
Syntax	MOVECIRCDTSP (max axes, bit, axis DT list starting No., end point DT list starting No., radius, clockwise 0 / anticlockwise 1)
Controller	Valid in ZMC4XX controllers and above.
Example	<pre>BASE (0, 1, 2) ATYPE = 1, 1, 1</pre>

	<pre> DPOS = 0, 0, 0 TABLE (10, 0, 1) 'save axis 0, 1 into TABLE(10) TABLE (20, 0, 50) 'only one end point (X, Y) is put TRIGGER WHILE 1 IF SCAN_EVENT (IN(0))>0 THEN MOVECIRCDTABSSP (6, 3, 10, 20, 50, 1) MOVECIRCDTSP (6, 3, 10, 20, 50, 1) 'move from starting point, pass (0, 50), the radius is 50, draw the arc in anticlockwise direction. ENDIF WEND </pre>
	

MOVECIRC2DTSP / MOVECIRC2DTABSSP -- DT 3-Point

Arc

Type	DT motion command.
Description	Save axis No. and motion distance into TABLE separately. Then, do circular motion by TABLE list. Use bit to select axis No. of TABLE. <u>Draw the arc according to the starting point, reference point, end point.</u>
Syntax	MOVECIRC2DTSP (max axes, bit, axis DT list, end point DT list, reference point DT list, mode) mode: ➤ < 0: the reference point is front of current point ➤ = 0: the reference point is in the middle ➤ > 0: the reference point is behind of current end point
Controller	Valid in ZMC4XX controllers and above.
Example	<pre> BASE (0, 1) ATYPE = 1, 1 DPOS = 0, 0 TABLE (10, 0, 1) 'save axis 0, 1 into TABLE(10) TABLE (20, 50, 10) 'save end point coordinates into TABLE(20) TABLE (30, 25, 25) 'save reference point coordinates into TABLE(30) TRIGGER WHILE 1 </pre>

	<pre> IF SCAN_EVENT (IN(0))>0 THEN MOVECIRC2DTSP (3, 3, 10, 20, 30, 0) 'when mode = 0, the trajectory pass the reference point, then run to the position of reference point + end point ENDIF WEND </pre>
	

MSPHERICALDTSP / MSPHERICALDTABSSP -- DT Space Arc

Type	DT motion command.
Description	Save axis No. and motion distance into TABLE separately. Then, do space circular motion by TABLE list. Use bit to select axis No. of TABLE. <u>Draw the arc according to the starting point, reference point, end point.</u>
Syntax	MSPHERICALDTSP (max axes, bit, axis DT list, end point DT list, reference point DT list, mode) mode: assign the second point's position ➤ -1: assign the second point in front of current point, just for reference ➤ 0: assign the second point in the middle, run to reference point ➤ 1: assign the second point in behind of current point and end point
Controller	Valid in ZMC4XX controllers and above.
Example	<pre> BASE (0, 1, 2) ATYPE = 1, 1, 1 DPOS = 0, 0, 0 TABLE (10, 0, 1, 2) 'save axis 0, 1, 2 into TABLE(10) TABLE (20, 0, 0, 100) 'save end point coordinates into TABLE(20) TABLE (30, 30, 40, 50) 'save reference point coordinates into TABLE(30) WHILE 1 IF SCAN_EVENT (IN(0))>0 THEN MSPHERICALDTSP (3, 7, 10, 20, 30, 0) 'when mode = 0, it starts from current point, pass the reference point, then run to the position of reference point + end point ENDIF WEND </pre>

Chapter 23 Simple Routines

23.1. 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.
  ELSEIF IN(1) = ON AND IN(0) = OFF THEN
    CONNECT(10, AXISHAND) 'link to axis 0,ratio is 10
  ELSEIF 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.

  ELSEIF 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](#) instruction.

Position Comparison Output

Please refer to Chapter 12 “[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 7 “[6 FOD Robotic Arm](#)”, there are hardware position comparison output and software position hardware comparison.

Robotic Arm by MOVESYNC Instruction

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=1 then
    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_Time,
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
G_X)
    BASE(8)
MOVESYNC(convAngleY,CONVSYNCGO_Time,CONVREG_POS,CONVREG_Axis,convREG
G_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

Encoder Read

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 codes

```

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

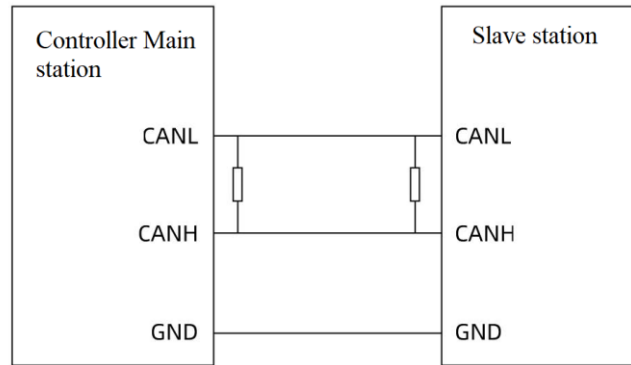
    IF coor_rel THEN
        MOVE(sub_para(0),sub_para(1),sub_para(2),sub_para(3))
    ELSE
        local xdis, ydis, zdis, udis
        IF sub_ifpara(0) THEN
            xdis = sub_para(0)
        ELSE
            xdis = ENDMOVE_BUFFER(0)
        ENDIF
        IF sub_ifpara(1) THEN
            ydis = sub_para(1)
        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

```

23.2. 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

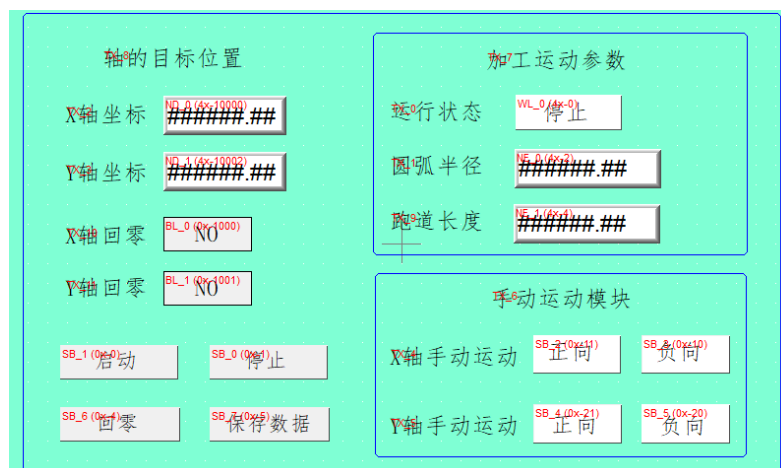
WHILE 1
    WAIT UNTIL if_send = 1 'wait until data comes.
    TABLE(0,32,1,$ff)      'send 1 byte to controller which can-cob-id is 32,it is FF.
    CAN(0,7,0)              '0-CAN channel,7-send data,0-start table address of data
    if_send = 0
WEND
END

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_state = 1                                    'run task: draw arc + runway
    MODBUS_REG(0) = run_state                        'enter 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, indicates axis 0 has done homing motion

WAIT IDLE(1)
MODBUS_BIT(1001)=1         'set sign, indicates 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,"aaaaaaaaa"
        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
```

23.3. Bus Initialization

EtherCAT Initialization

Requirements: EtherCAT motion controller and EtherCAT servo drives, use latest RTSys version.

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 24 Errors Debugging

Due to wrong wiring, procedure logic problems, instructions errors, etc., motor may 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 RTSys to debug, following functions should be known: manual motion debugging, interrupt debugging, oscilloscope collection, register check, remote commands, procedure information print, fast IO test.

24.1. Common Problems List

Problem	Debugging Solutions
Motor doesn't run	Debugging: Manual Motion
Register value on HMI is not correct	Debugging: Register Reading
Not run as expected in procedure	Debugging: Breakpoint; Debugging: Program Information Print
inputs or outputs don't work	Debugging: Fast-IO Detection
Machine shakes too much	Debugging: Oscilloscope Sampling

Problems 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 automatically. 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 Syntax 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:

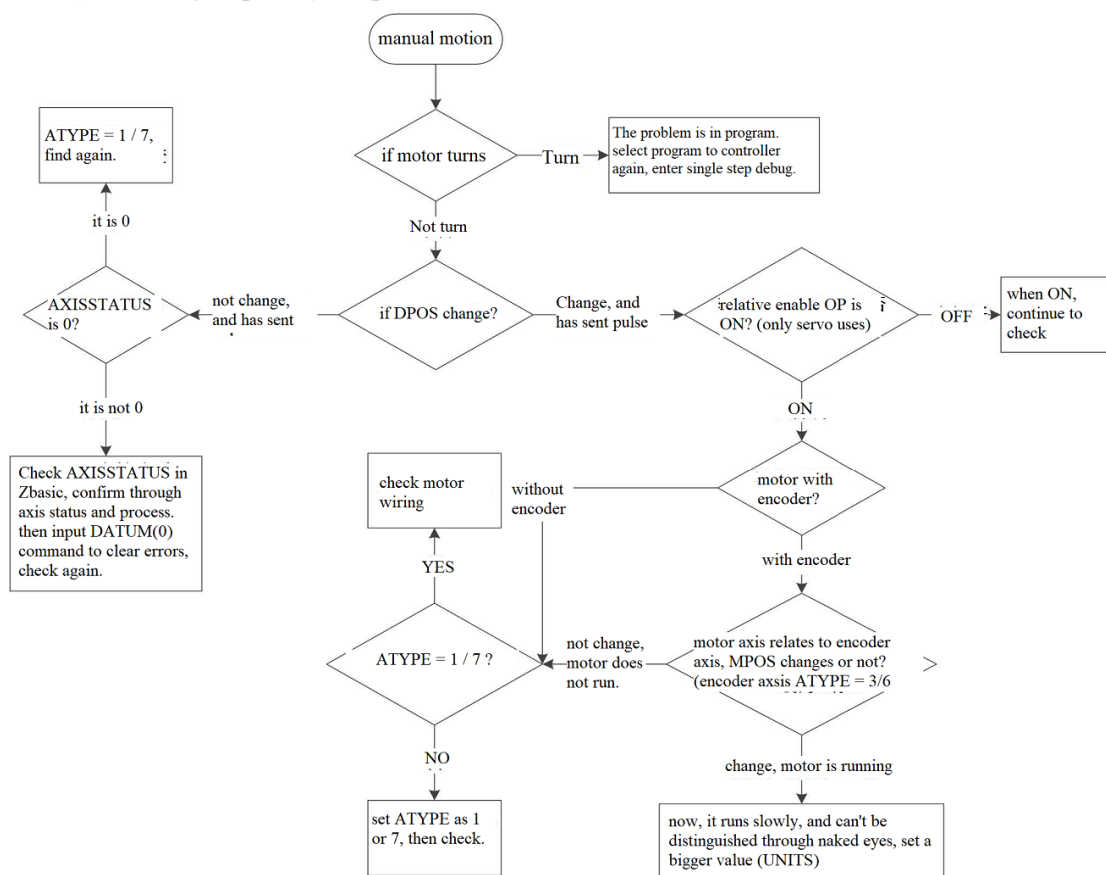
1) open RTSys software.

2) close all other software and procedures linked with the controller, except ZDevelop, to avoid these external elements make influence on operation.

3) use RTSys to download an empty program to controller, for avoid making influence by internal elements.

4) open RTSys, “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.

24.2. Solutions

Debugging: Manual Motion

Use “Manual” motion to check if there are wiring problems.

Close all related software except RTSys, and use RTSys 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.

Axis	ATYPE	UNITS	ACCEL	DECEL	SPEED	DPOS	Left/Move	Right/Move	DistanceAbsolute	Move	MPPOS	IDLE	AXISSTATUS
0	0	1.000	10000.0	0.000	1000.00	0.000	-	+		☐ Move	0.000	-1	0h
1	0	1.000	10000.0	0.000	1000.00	0.000	-	+		☐ Move	0.000	-1	0h
2	0	1.000	10000.0	0.000	1000.00	0.000	-	+		☐ Move	0.000	-1	0h
3	0	1.000	10000.0	0.000	1000.00	0.000	-	+		☐ Move	0.000	-1	0h
4	0	1.000	10000.0	0.000	1000.00	0.000	-	+		☐ Move	0.000	-1	0h
5	0	1.000	10000.0	0.000	1000.00	0.000	-	+		☐ Move	0.000	-1	0h

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.

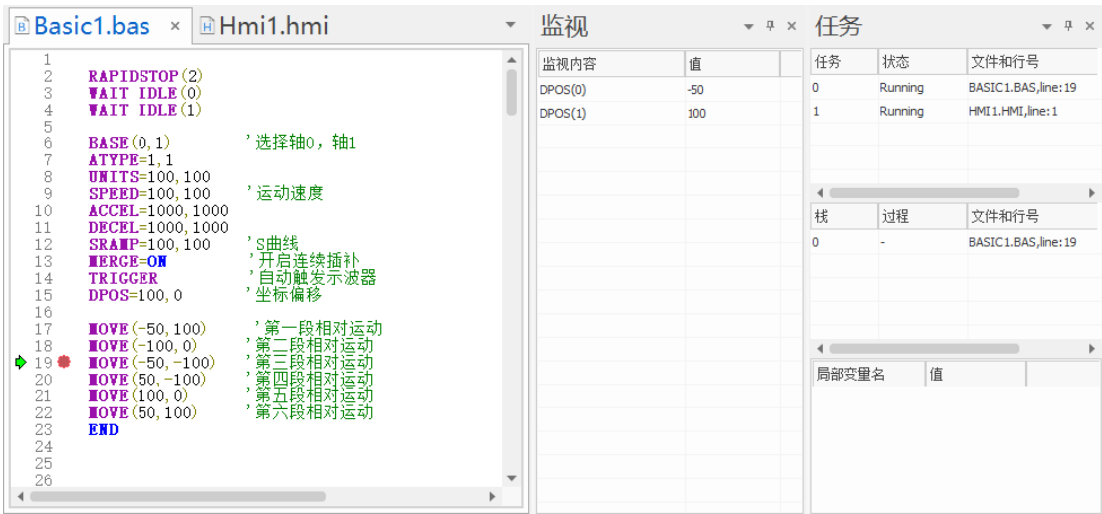
5. For controller that is supplied by dual-power, please check whether IO 24V is wired

and supplied normally.

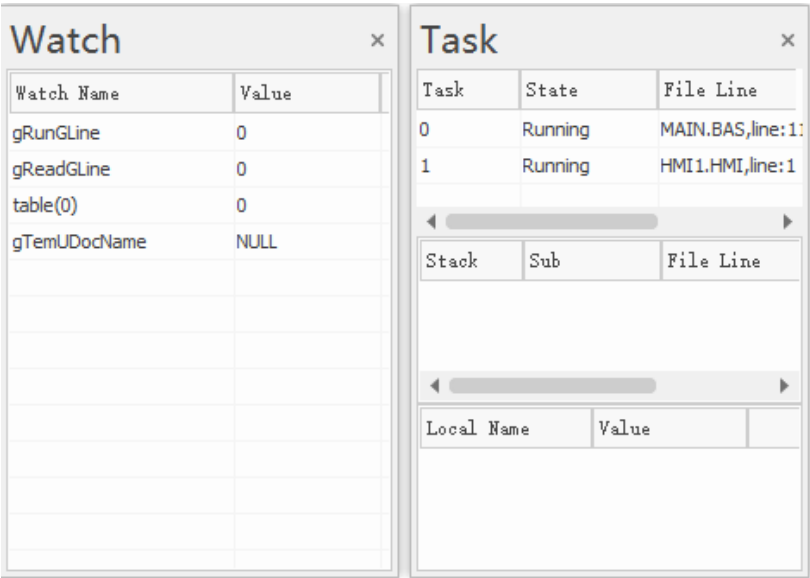
Debugging: Breakpoint

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 RTSys, connect controller well, then click debug-start/stop debug to enter debugging mode.



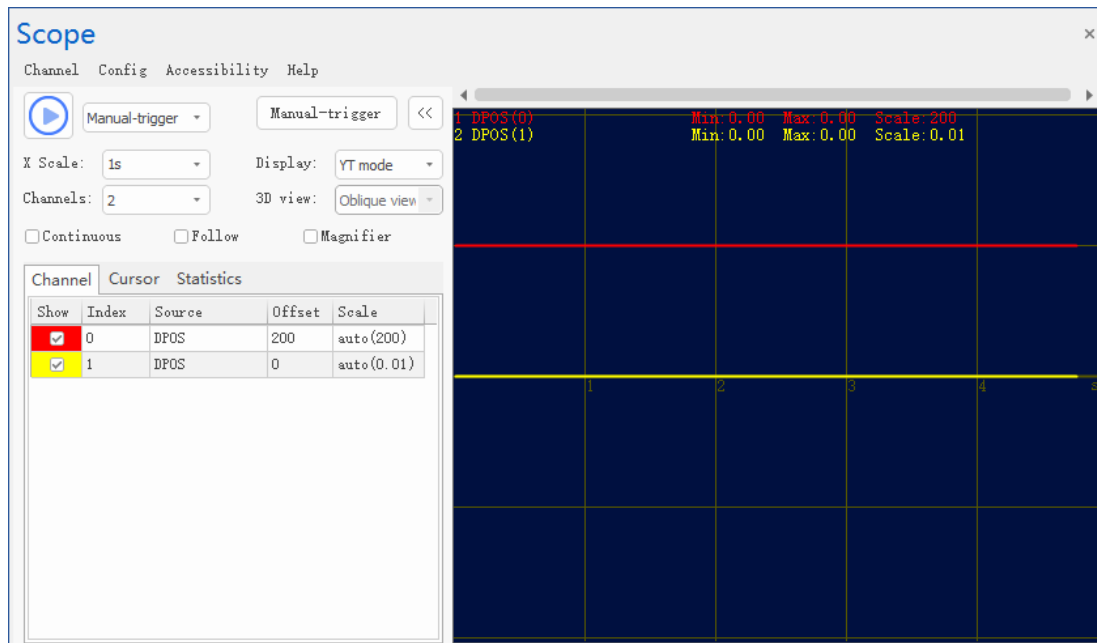
In debug mode, task running process, monitoring item, sub process, local variables in subsidiary functions all can be checked.



Debugging: Oscilloscope Sampling

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.

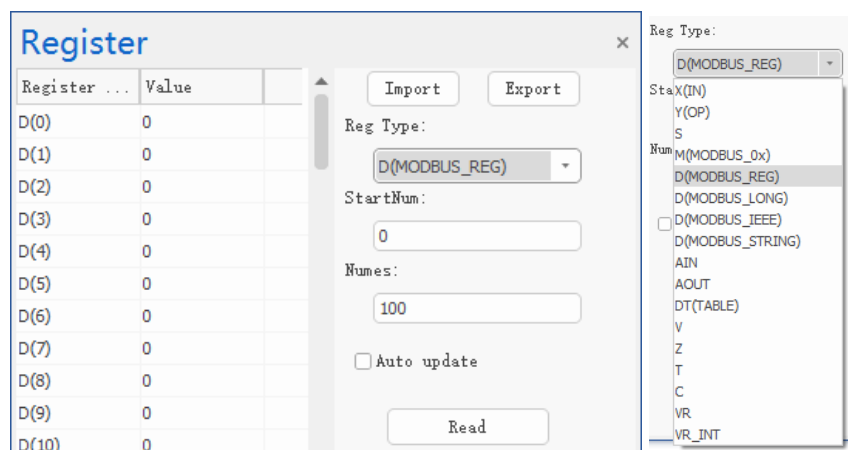


XYmode is used to check two-dimensional trajectory, source of first 2 channels should be set as DPOS or MPOS.

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

Debugging: Register Reading

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.

Debugging: Remote Commands Enter

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.

Debugging: Program Information Print

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).

1		16	 while 1
2	dim ccv(20), aa, bb	17	?bb
3	ccv="ad"	18	move(bb)
4	aa=100	19	wend
5	bb=300	20	end
6		21	
7	base(0, 1, 2)	22	
8	atype=1, 1, 1	23	 sub aaa(num)
9	units=200, 200, 200	24	while 1
10	speed=100, 100, 100	25	?aa
11	accel=1000, 1000, 1000	26	move(aa)
12		27	wend
13	stoptask 1	28	end sub
14	runtask 1, aaa	29	

Debugging: Fast-IO Detection

Connect RTSys to controller, click view-input/output, link output and input one by one (EGND must be linked) at the same time, then operate output, and check output status in RTSys.

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.

Debugging: Axis Parameters Status Reading

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			
101	Wrong path		Recommend to use the path that is with disk symbol.
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 zip project or jar 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		Check if MotionRT7 drive version is too old, the new card can't use old drive.
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 to repair. 2. For XPCIE, please check wiring cable,

			make sure it is good.
264	RAM Error		1. Make sure internet stable. 2. Hardware error, contact with us to repair.
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		Contact with us to repair.
268	Ethernet Hardware Error		
Software Error			
271	Backup Power Error		Contact with us to repair.
272	Sub-Card Not Exist		
273	ID File Lost		Contact with us to repair.
274	System File Lost		Contact with us to repair.
275	No Master Control, Appear in Sub-Card		
276	Program File Verify Error		1. For ZMC0XX, see if there is the ROM file. 2. Check if the ZAR file is correct.
277	Program File Error		1. See whether it lacks of 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	Program Abnormal	The problem may appear in .c file, the syntax error.	1. Check your edited C language program. 2. If bottom layer C language error, please contact with us to update firmware.
288	287 Not Solve, Alarm 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 "Pause" / Others		
1011	In IDLE, Can't 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 “OK” FRAME Correction Data		
1020	Too Less FRAME Auxiliary Para		
1021	Too Small Span Between FRAME Correction Data, < Joint-Axis Numbers		
1022	FRAME IN Coordinate Error		
1023	Don’t Force to Modify Coordinates under FRAME.		
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 unit of the angle.	
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		
1037	Don’t Operate Robot when Moving	IDLE of Joint spindle and virtual spindle are 0 when using frame_rotate.	
1051	MOVER2_ARC Coordinate Error, like, 3-Point is collinear.		
1052	Now Point is in Abnormal Area, Don’t Run Trajectory Command.		
1053	Now Trajectory Command Passes Abnormal Area, Don’t Run Trajectory Command.		
1054	The Motion’s Target is in Abnormal Area, Don’t Run Trajectory Command.		
1055	Smooth Coefficient ZSMOOTH Para Out of Range [0, 100].		

1056	Arch Motion Para Error		
2000	RTBASIC Module Offset, Para in Module Error		
2001-2020	Internal Error of 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 of 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 / Chinese is not noted.	1. See if there are unsupported symbols (especially in function definition, 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 characters.
2033	Unknown Name in Expression	Undefined variable /	1. See if commands are

		array	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 1 st 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 Command, Can't Read		1. Cancel related 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 it 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 the same line, pls add one operator.	
2070	CALL must Call SUB		1. Correct the content 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 Started) is Running		1. Check if there is conflict on executing task No.
2076	Multi-Para, Please Use Comma		1. Check para usage format, and correct it.
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 the error when defined constant data is assigned again.	1. Correct CONST usage.
2082	Command Sending Online by PC Not Allowed		1. The command can't be sent out in "output" window.
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		
2140	SUB Called Para Repeat		SUB called special robotic arm syntax, check the syntax usage.
2150	Function Return Not Immediate Done		
2151	Function Can't Immediate Return Because Now Expression Doesn't Support		
2152	Dynamic Stake is Overflow		
5165	SO File Platform Error		SO file compile platform should be consistent with the controller.
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		Correct command usage.
3302	2 Parallel Lines, No Intersection Point		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 Transform Error		
3421	MODBUS Slave Returns Unsupported Function Code		
3422	MODBUS Slave Return Address Space Error		Master and slave addresses are not matched, please read & write register addresses that are not exist, then correct the address.
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	1 PLC Main Task Only		
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	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	Not OUT Type in the Rightmost		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 are 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	Correct it.

		are set same component No.	
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 > controller allowed	You can check x and y 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 executing.	Please use CAD control to import corresponding graphics, then do other 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 a 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			
7001	File Execution End		
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	Main 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	Para 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

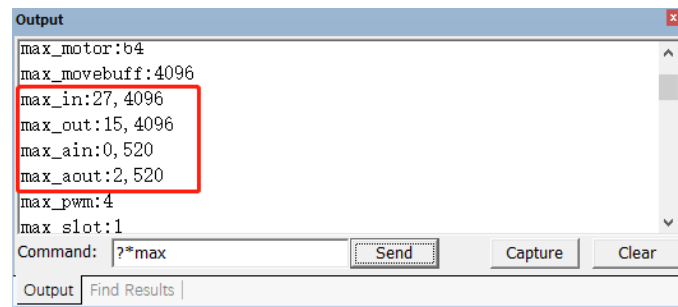
Module expansion is used to expanse pulse-axis, digital inputs & outputs and analog inputs & outputs when there is no enough axis resource and IO resource on controller. Pulse-axis extension is only valid in expansion module with pulse interface, which means bus axis can't be expanded.

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:

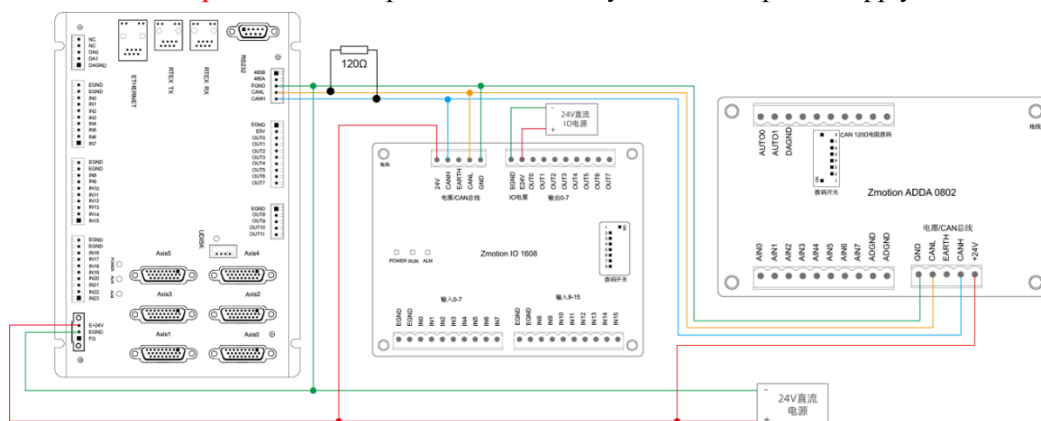
Stitch NO.	Name	Description
1	GND	Internal power position
2	CANL	CAN differential data-
3	EARTH/SHIELD	Shield layer
4	CANH	CAN differential data+
5	+24V	Internal power 24V input

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.



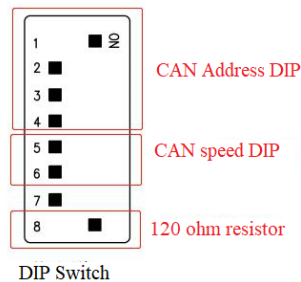
Resource Mapping

ZCAN expansion module resource becomes 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 Mapping

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

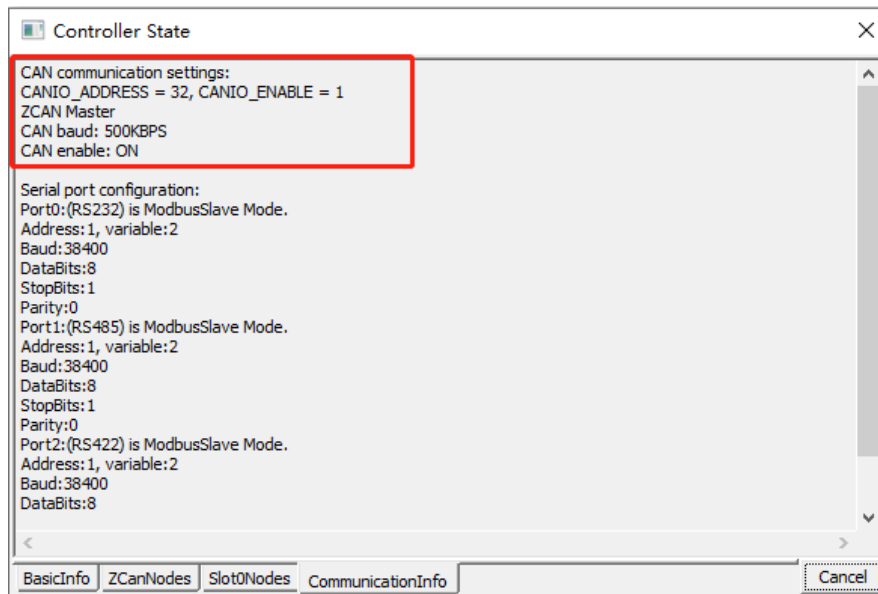
Dial 5-6 choose CAN fieldbus communication speed, speed combination value=dial 6×2+dial 5×1, combination value is from 0 to 3, relative speed as follow:

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)

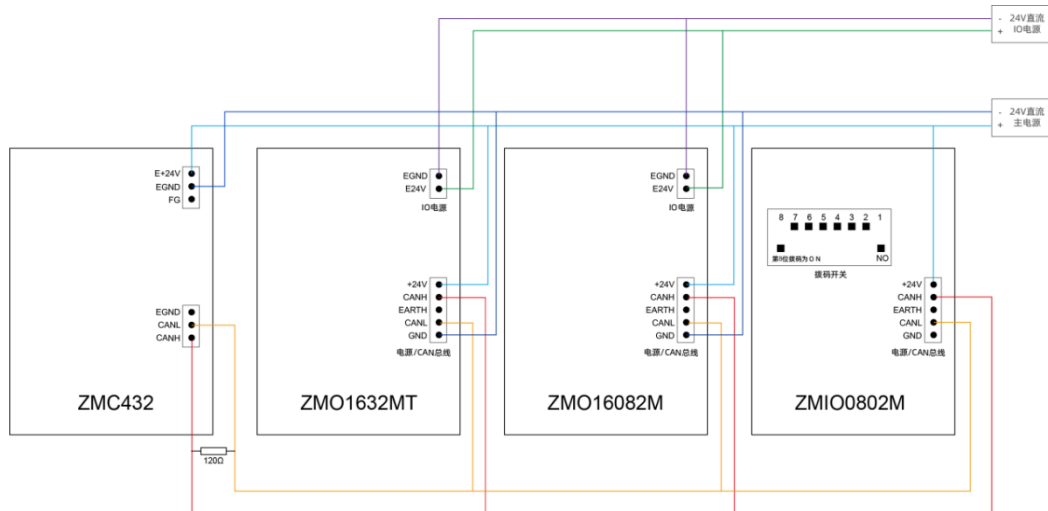
For example, controller has 28 IN and 16 OP itself, which means start address of the first expansion module should exceed 28, and address dial should be set as combination value 1 according to IO map rule(binary combination value is 0001, relevant dial 1-4 from right to left, dial 1 as ON, dial others 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.

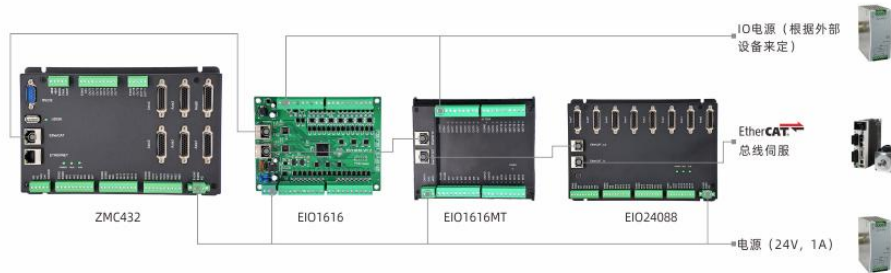
控制器状态							
CanID	硬件ID	轴数	输入	输出	AD	DA	
Local	432-0(ZMC432)	32	30(0-29)	18(0-17)	0	2(0-1)	
1	48(ZIO 1632)	0	16(32-47)	32(32-63)	0	0	
3	26(ZIO 16082)	2	16(64-79)	8(64-71)	0	0	
4	10(ZAIO0802)	0	0	0	8(40-47)	2(20-21)	
基本信息 ZCan节点 槽位0节点 通讯配置 取消							

Axis Mapping

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:

$\text{AXIS\_ADDRESS}(\text{axis NO.}) = (32 * 0) + \text{ID}$ 'local axis port 0 of expansion module



Resource Mapping

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 linking sequence with controller, and start from 0.

IO Mapping

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.

Syntax:

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-default

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 Mapping

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` '1st ECAT drive, drive NO. is 0, binding with axis 0

`AXIS_ADDRESS (0)=(0<<16)+0+1` '2nd ECAT drive, drive NO. is 1, binding with axis 2

`AXIS_ADDRESS (0)=(0<<16)+0+1` '3rd 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 & HMI Communication

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:

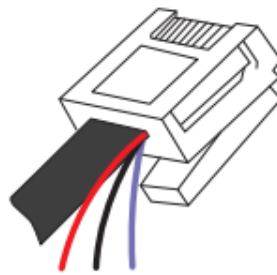
1. Download program written in ZDevelop into controller.
2. HMI program written by relevant programming software is downloaded and saved in HMI.
3. After program is downloaded, choose serial port or net port link with HMI, and controller run offline.

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.

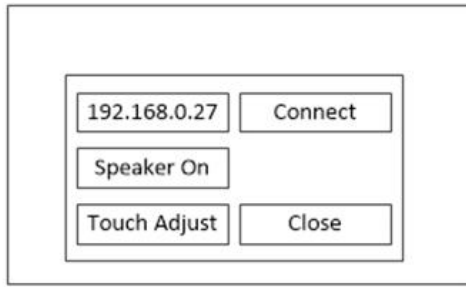
Main power of HMI and controller can be the same.



Steps for connecting HMI to controller:

1. use RTSys software to write HMI program, then connect to controller, download the program into ROM for storage when power off, next, disconnect controller and RTSys and power on the HMI.

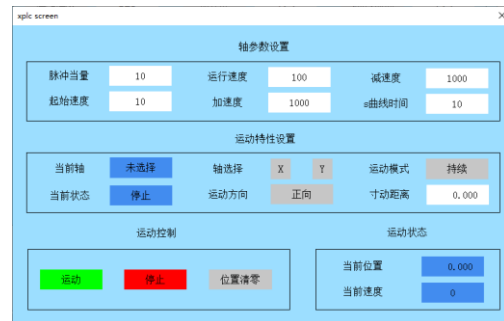
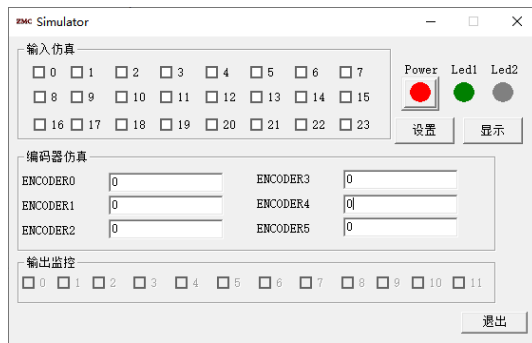
2. connect the ZHD400X directly to the network port of the controller using the provided cable, and then click on the four corners of the screen in the order of drawing a Z, 2 times in a row, wake up the screen, and a setting window will pop up to perform touch correction, controller IP modification, 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.

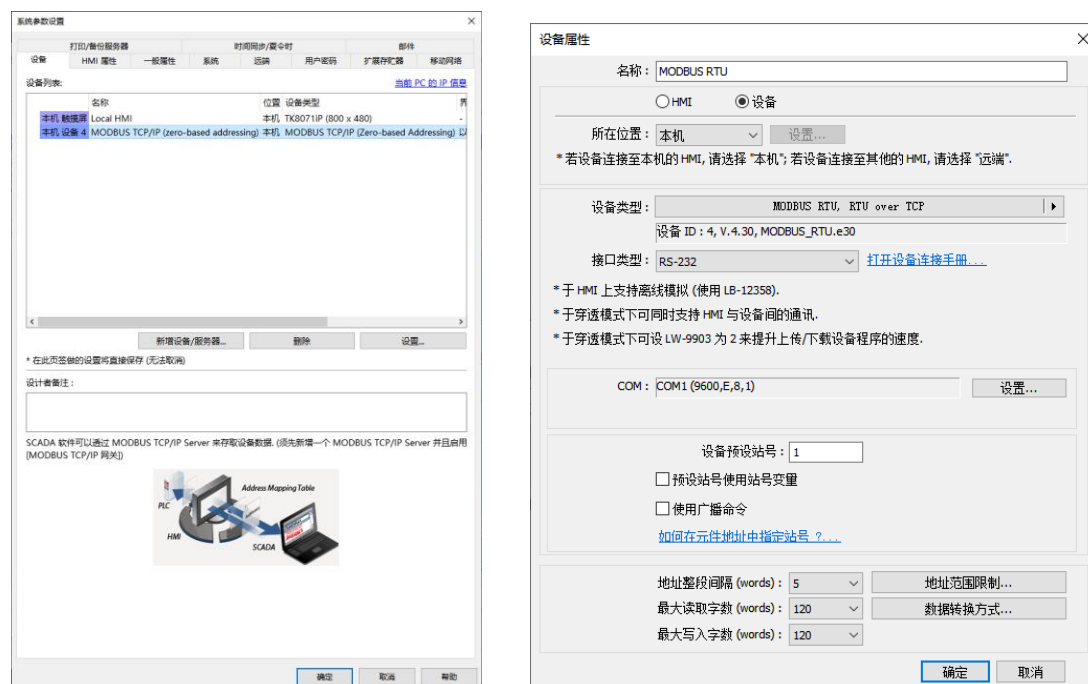
2. Download HMI program

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.

通讯端口设置

通讯端口: COM 2

波特率: 9600

数据位: 8 Bits

校验: Even

停止位: 1 Bit

超时 (秒): 1.0

通讯延时 (毫秒): 0

命令重送次数: 0

* OS 20120920 或更新版本支持 14400 波特率.

确定 取消

If connects through net port:

Device type: select mode MODBUS RTU (Zero-BASEd Addressing), interface type will be 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.

IP 地址设置

IP 地址: 192 . 168 . 0 . 11

端口号: 502

超时 (秒): 1.0

通讯延时 (毫秒): 0

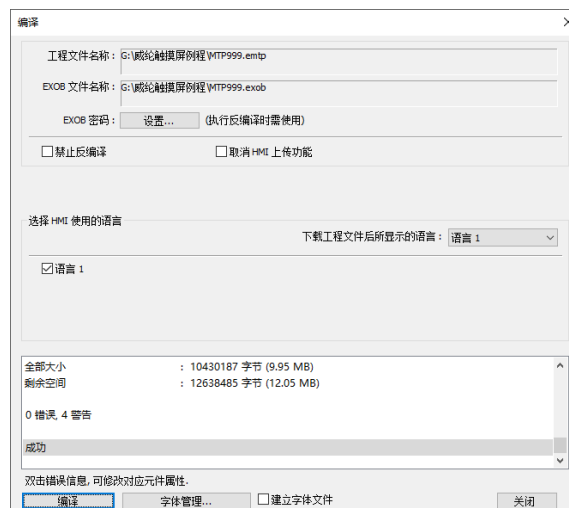
命令重送次数: 0

确定 取消

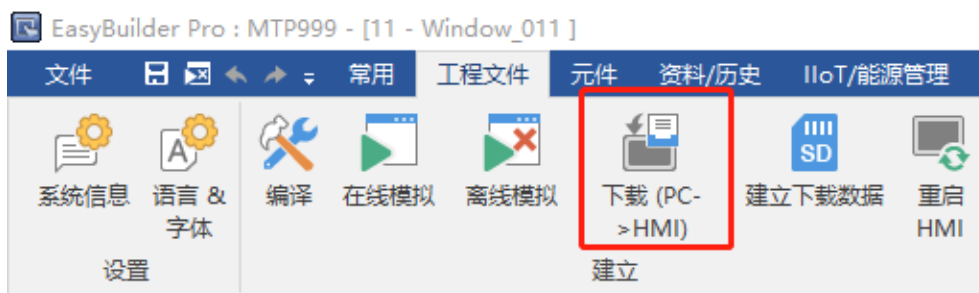
After system parameters are set, compile written configuration program, click “编译”, open the window.



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.

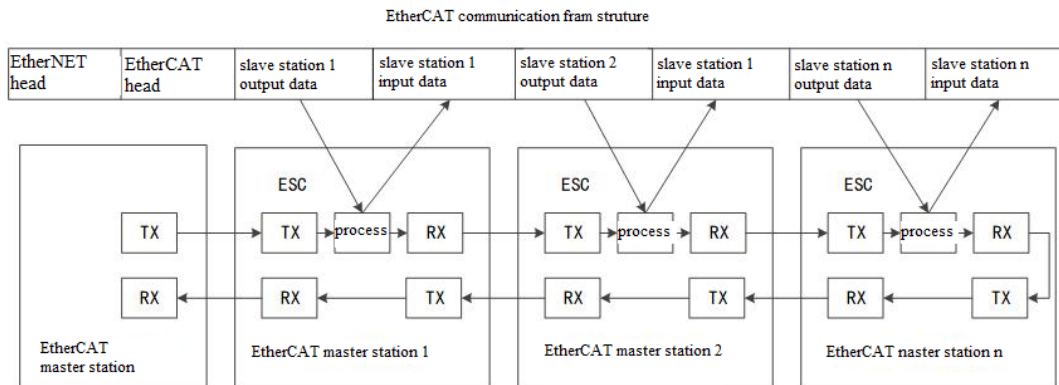
Touch screen simulation interface:



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.

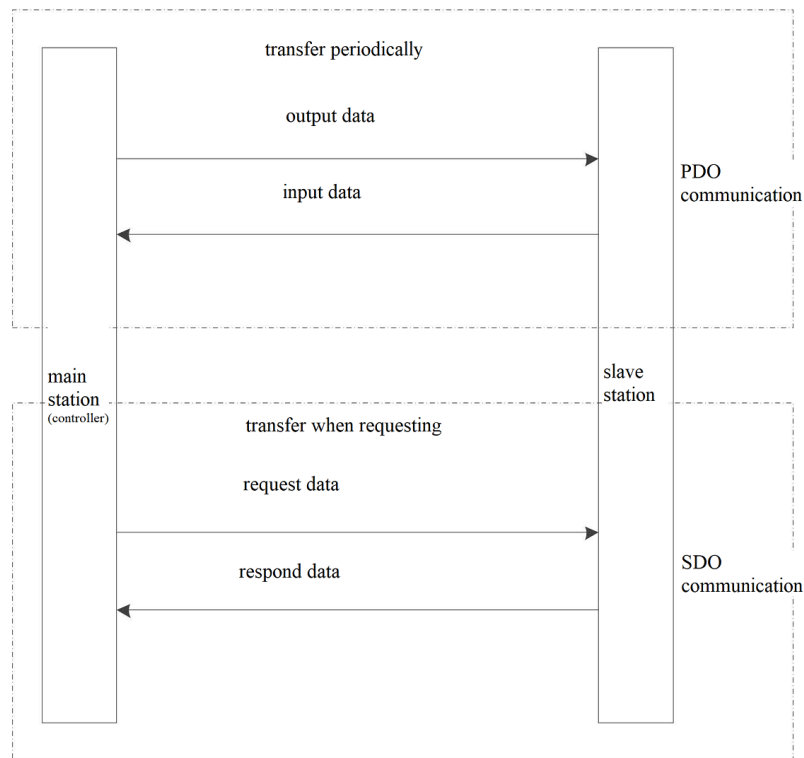
EtherCAT takes full advantage of the full-duplex nature of Ethernet, using master-slave mode media access control. The EtherCAT network is obviously different from the ordinary Ethernet. In the same EtherCAT network, there is only one EtherCAT master station, and the EtherCAT slave station has a chip 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.

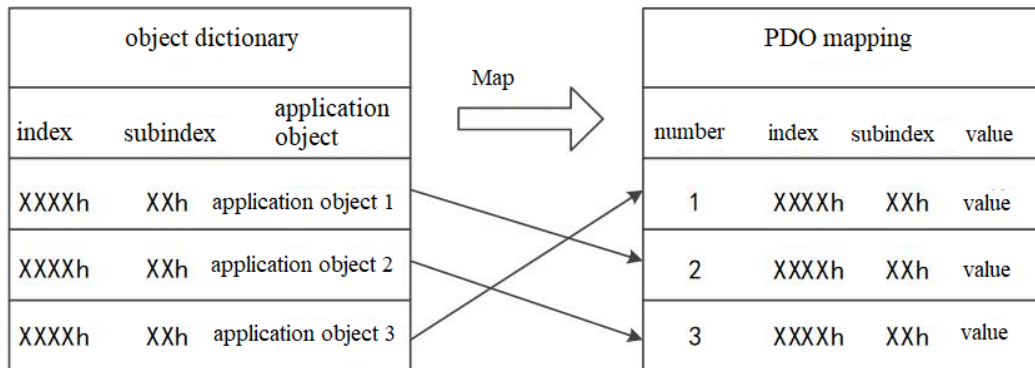
Drive PDO must be configured in EtherCAT initialization, and PDO list of drive is configured by DRIVE_PROFILE. Currently, there are more than 20 kinds of configuration to be selected, each configuration includes data dictionary description. If DRIVE_PROFILE can't meet, PDO can be self-defined, using 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				
6091h	01h	Electronic gear ratio numerator	1~4294967295	U32	RW	NO	All	YES
	02h	Electronic gear ratio denominator						
6098h	00h	Homing mode	-128~127	18			RxPDO	
60FDh	00h	Digital input	0~4294967295	U32	RO	TxPDO	All	NO
	01h	Digital output			RW			YES
60FFh	00h	Target speed	-2147483648 ~ 2147483647	I32	RW	RxPDO	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 Communication

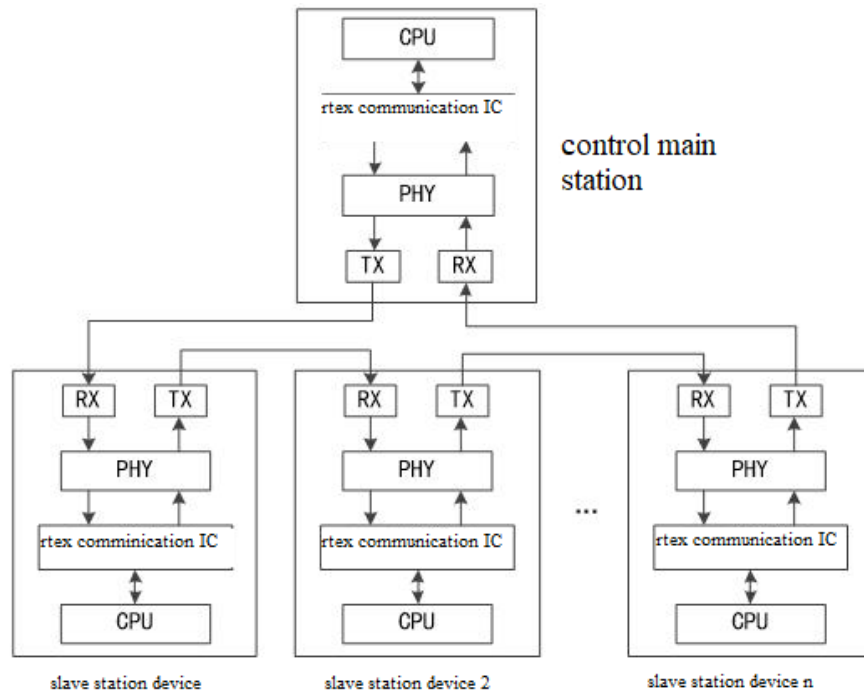
Some models of ZMOTION motion controller support RTEX bus, RTEX fieldbus axis, EtherCAT bus axis, pulse axis joint interpolation. RTEX bus is high-speed network fieldbus developed by Panasonic, which suits to real-time bus of small system, and the pipeline composed of small equipment is more flexible 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	0~20 ²³	pulse	Set the number of pulse when motor rotates one round
0	09	C	Electronic gear ratio numerator	0~20 ³⁰	-	Set the numerator of electronic gear ratio
0	10	C	Electronic gear ratio denominator	0~20 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	- 1073741 823 ~ 1073741 823	Instru ction 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	1~2	-	Set the ration of RTEX communication period and

			setting			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	0~20000 00	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=0.5ms*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	For RTEX communication command speed, they are set

	RTEX communication is small.	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.).