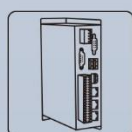
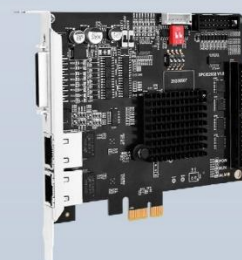
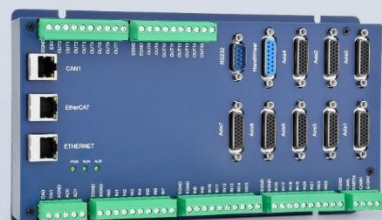
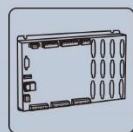


Zmotion PC Programming Manual

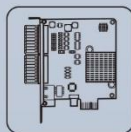
V2.1.4



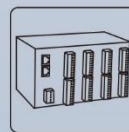
Vision Motion
Controller



Motion
Controller



Motion
Control Card



IO Expansion
Module



HMI

Copyright Statement



This manual describes Zmotion PC command definitions, syntax, usage, and examples.

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 error handling procedures in software. Zmotion has no obligation or responsibility for the loss.

Contents

Chapter I Product Descriptions	8
1.1. Zmotion Motion Controller Features	8
1.2. Product Framework	9
Chapter II How to Use & Encapsulate Function Library?	13
2.1. How to Call Function Library	13
2.1.1. Windows System	13
2.1.2. LINUX System.....	25
2.1.3. Wince System.....	31
2.2. How to Encapsulate Auxiliary Library?.....	34
2.2.1. Auxiliary Library Encapsulation Methods	34
2.2.2. Encapsulation Examples	36
Chapter III Instruction List.....	46
Chapter III How to Connect & Initialize?	57
4.1. How to Connect to Controller?	57
4.1.1. Instruction List	57
4.1.2. Principle & Description	57
4.1.3. Routines.....	79
4.2. How to Get Controller Information?.....	86
4.2.1. Instruction List	86
4.2.2. Principle & Description	86
4.2.3. Routines of Getting Controller Info	86
Chapter V How to Set Motion Parameters?	88
5.1. Motion Basic Parameters	89
5.1.1. Instruction List	89
5.1.2. Principle & Description	90
5.1.3. Routines.....	99
5.2. Other Axis Parameters.....	104
5.2.1. Instruction List	104
5.2.2. Principle & Description	104
5.2.3. Routines.....	110
5.3. Axis Parameters Instruction List.....	119
5.3.1. Instruction List	119
5.3.2. Detailed Description	119
5.3.3. Routines.....	121
Chapter VI How to Control Basic Motions?	124
6.1. Motion Buffer	124

6.1.1.	Instruction List	124
6.1.2.	Principle & Description	125
6.1.3.	Routines.....	127
6.2.	Single-axis Motion.....	130
6.2.1.	Instruction List	130
6.2.2.	Principle & Description	131
6.2.3.	Routines.....	131
6.3.	Homing Motion	137
6.3.1.	Instruction List	137
6.3.2.	Principle & Description	138
6.3.3.	Routines.....	142
6.4.	Electronic Gear.....	144
6.4.1.	Instruction List	144
6.4.2.	Principle & Description	144
6.4.3.	Routines.....	146
6.5.	Interpolation Motion	155
6.5.1.	Instruction List	155
6.5.2.	Principle & Description	157
6.5.3.	Routines.....	161
6.6.	Axis Pause / Resume / Stop.....	183
6.6.1.	Instruction List	183
6.6.2.	Principle & Description	184
6.6.3.	Routine.....	184
Chapter VII How to Control Other Motions?		186
7.1.	Coordinates Loop	186
7.1.1.	Instruction List	186
7.1.2.	Principle & Description	186
7.1.3.	Routine.....	187
7.2.	Pitch Compensation	189
7.2.1.	Instruction List	189
7.2.2.	Principle & Description	189
7.2.3.	Routine.....	190
7.3.	Backlash Compensation.....	193
7.3.1.	Instruction List	193
7.3.2.	Principle & Description	193
7.3.3.	Routine.....	193
Chapter VIII How to Set / Get Hard Port?		196
8.1.	Digital IN & OUT	196
8.1.1.	Instruction List	196
8.1.2.	Principle & Description	196
8.1.3.	Routines.....	197
8.2.	Analog IN & OUT	201
8.2.1.	Instruction List	201
8.2.2.	Principle & Description	202

8.2.3.	Routine.....	202
8.3.	PWM Control	203
8.3.1.	Instruction List	203
8.3.2.	Principle & Description	203
8.3.3.	Routine.....	205
Chapter IX	How to Ensure Motion Safe?	207
9.1.	Axis Soft Position Limit	207
9.1.1.	Instruction List	207
9.1.2.	Principle & Description	207
9.1.3.	Routine.....	208
9.2.	Axis Hard Position Limit	210
9.2.1.	Instruction List	210
9.2.2.	Principle & Description	210
9.2.3.	Routine.....	212
9.3.	Axis Alarm.....	213
9.3.1.	Instruction List	213
9.3.2.	Principle & Description	213
9.3.3.	Routine.....	214
Chapter X	How to Interact Data?	216
10.1.	VR Register	216
10.1.1.	Instruction List	216
10.1.2.	Principle & Description.....	216
10.1.3.	Routine	216
10.2.	Table Register	218
10.2.1.	Instruction List	218
10.2.2.	Principle & Description.....	218
10.2.3.	Routine	219
10.3.	Modbus Register.....	220
10.3.1.	Instruction List	220
10.3.2.	Principle & Description.....	220
10.3.3.	Routine	221
10.4.	File Reading & Writing (Flash, U Disk, File 3)	223
10.4.1.	Instruction List	223
10.4.2.	Principle & Description.....	224
10.4.3.	Routines	226
Chapter XI	Bus Related Information.....	231
11.1.	Bus Initialization	231
11.1.1.	Instruction List	231
11.1.2.	EtherCAT Bus.....	231
11.1.3.	Routines	237
11.2.	SDO/PDO Writing & Reading	241
11.2.1.	Instruction List	241
11.2.2.	Principle & Description.....	241
11.2.3.	Routines	242

11.3.	Bus Torque.....	244
11.3.1.	Instruction List	244
11.3.2.	Principle & Description.....	244
11.3.3.	Routine	245
Chapter XII Other Commands.....		247
12.1.	Online Command	247
12.1.1.	Instruction List	247
12.1.2.	Principle & Description.....	247
12.1.3.	Routine	248
12.2.	Controller RTBasic Related	250
12.2.1.	Instruction List	250
12.2.2.	What is RTBasic Program	251
12.2.3.	How to Use RTBasic in PC Side	251
12.2.4.	Routines	252
12.3.	Controller Auto-Reporting	254
12.3.1.	Instruction List	254
12.3.2.	Principle & Description.....	254
12.3.3.	Routine	255
12.4.	Debug Related.....	259
12.4.1.	Instruction List	259
12.4.2.	Debug & Output.....	259
12.4.3.	Oscilloscope.....	259
12.4.4.	Routines	261
12.5.	RTSys Debug Software	268
12.5.1.	What is RTSys	268
12.5.2.	How to Download RTSys.....	269
12.5.3.	How to Debug Controller.....	270
Chapter XIII Instruction Details		271
Chapter XIV Instruction Returned Values		436
14.1.	Instruction Returned Value.....	436
14.2.	Return Values Details (Error Codes).....	436
14.3.	Others: Quick Start / Common Problems.....	445
14.3.1.	Quick Start.....	445
14.3.2.	Common Problems	447
Chapter I Instruction List.....		452
Chapter II Axis Superposition (Electronic Cam).....		454
2.1.	Motion Superposition	454
2.1.1.	Instruction List	454
2.1.2.	Principle & Description	454
2.1.3.	Routine.....	455
2.2.	Electronic Cam & Synchronous Track	457
2.2.1.	Instruction List	457
2.2.2.	Principle & Description	457
2.2.3.	Routines.....	459

Chapter III Position Latch	472
3.1. High-Speed Latch	472
3.1.1. Instruction List	472
3.1.2. Principle & Description	472
3.1.3. Routines	475
Chapter IV Position Comparison Output	480
4.1. Position Comparison Output	480
4.1.1. Instruction List	480
4.1.2. Principle & Description	480
4.1.3. Routines	481
Chapter V PT Motion	498
5.1. PT Motion	498
5.1.1. Instruction List	498
5.1.2. Principle & Description	498
5.1.3. Routines	498
Chapter VI Robot Motion Control	504
6.1. Robot Model	504
6.1.1. Instruction List	504
6.1.2. Principle & Description	504
6.1.3. Routine	506
Chapter VII Instruction Details	511

Chapter I Product Descriptions

1.1. Zmotion Motion Controller Features

Zmotion motion controller supports "PC direct online control" and provides DLL function library and corresponding examples, such as, VC, VB, C#, PYTHON, LABVIEW, etc. Also, the function library supports both WINCE and LINUX. And the library is available for all types of controllers.



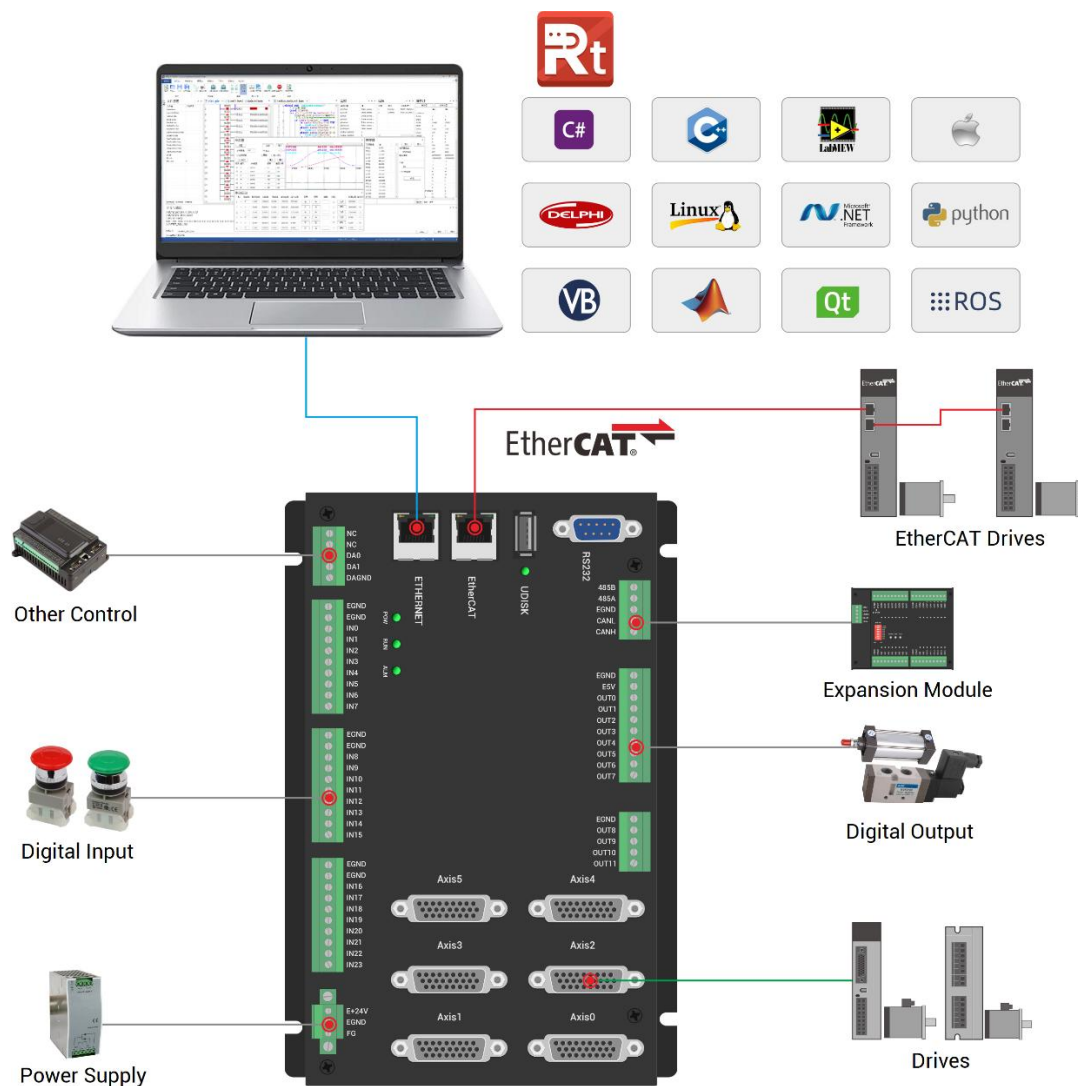
Compared with PCI, [ZMC](#) & [ECI](#) motion controllers are the following advantages:

1. No need to use the slot, which means the **stability is better**.
2. **Reduce the requirement** for PC, no PCI slot is needed.
3. You can use MINI PC or ARM IPC to **reduce the overall cost**.
4. The controller can be used as a wiring board directly to **save the space**.
5. The motion controller can run the program in parallel, it only needs to do simple interaction with PC, reducing PC software complexity.
6. The motion controller is a special controller that controls the operation mode of the motor. Generally, control structure mode: controller + driver + (step or servo) motor.
7. The controller supports **rich functions**: speed look-ahead, electronic cam, electronic gear, pitch compensation, fixed-step tracking, motion superposition, virtual axis, pulse closed loop, hardware position latch, position comparison output, continuous interpolation, and motion pause, etc.

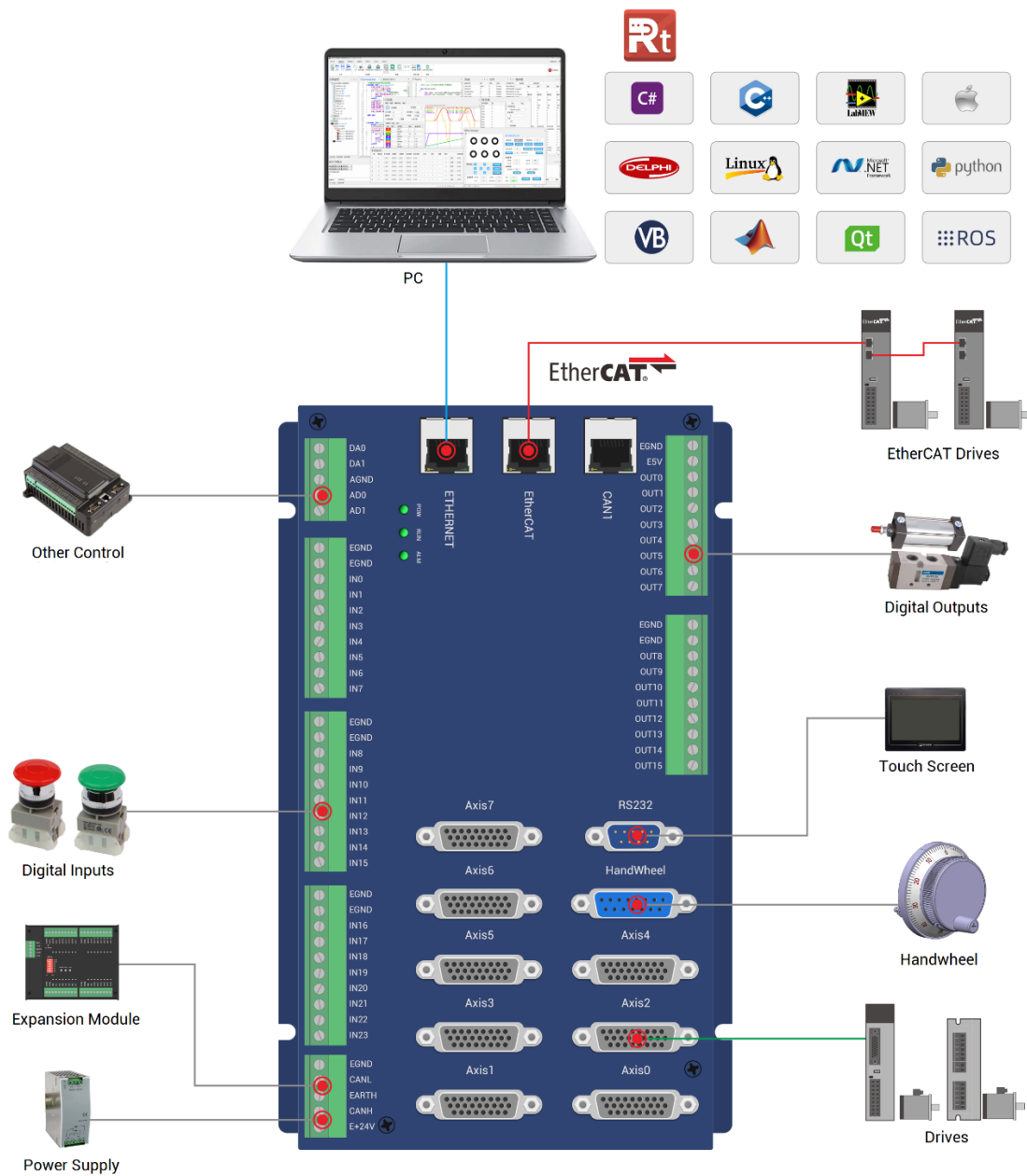
It can be seen that a motion controller with an Ethernet can be used to replace the PCI motion control card for saving space, reducing costs, optimizing programs, and making wiring more convenient, and that is why people prefer using Ethernet now.

1.2. Product Framework

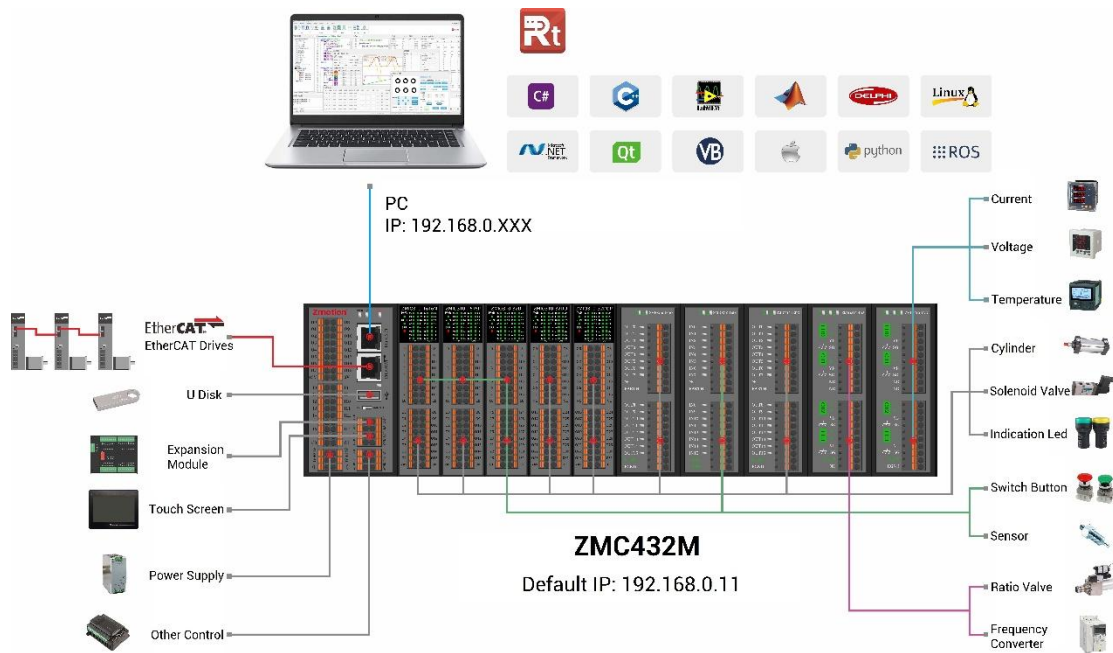
➤ ZMC432-V2 Connection Structure:



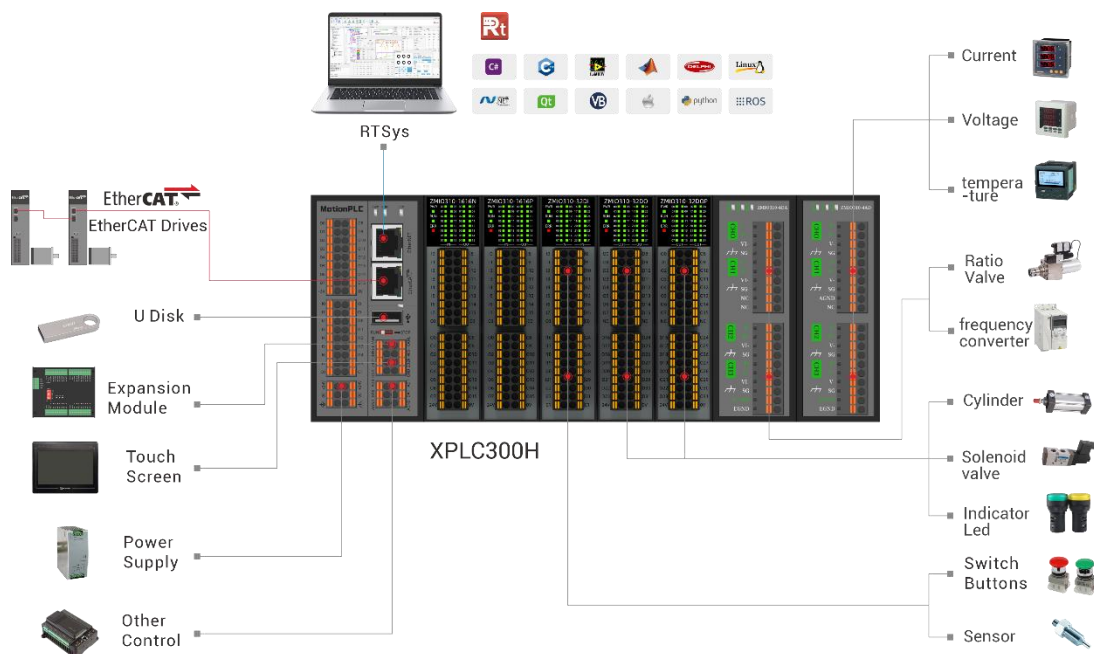
➤ **ECI2828-V2 Connection Structure:**



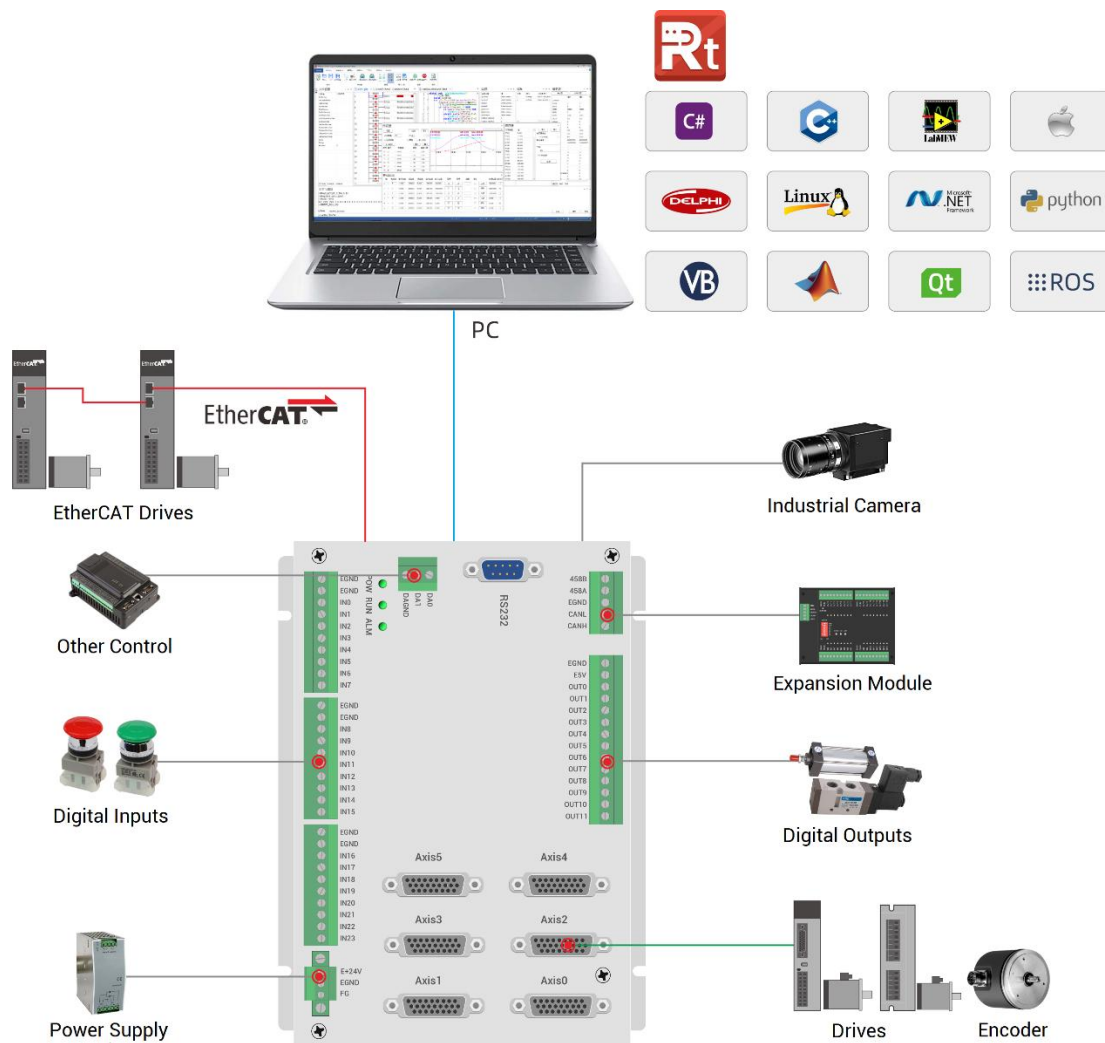
➤ **ZMC432M** Connection Structure:



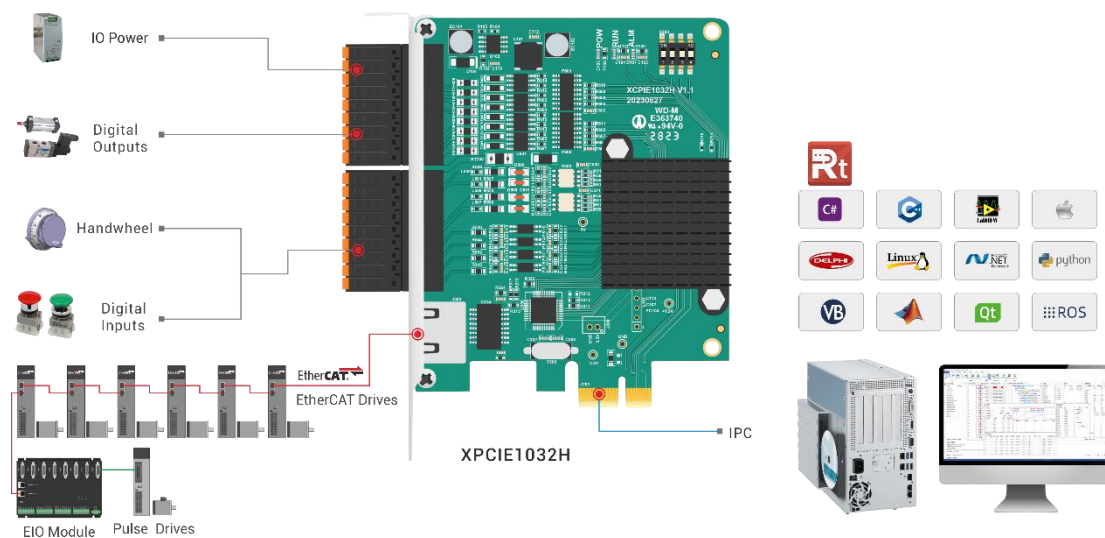
➤ **XPLC300H** Connection Structure:



➤ **VPLC532E** Connection Structure:



➤ **XPCIE1032H** Connection Structure:



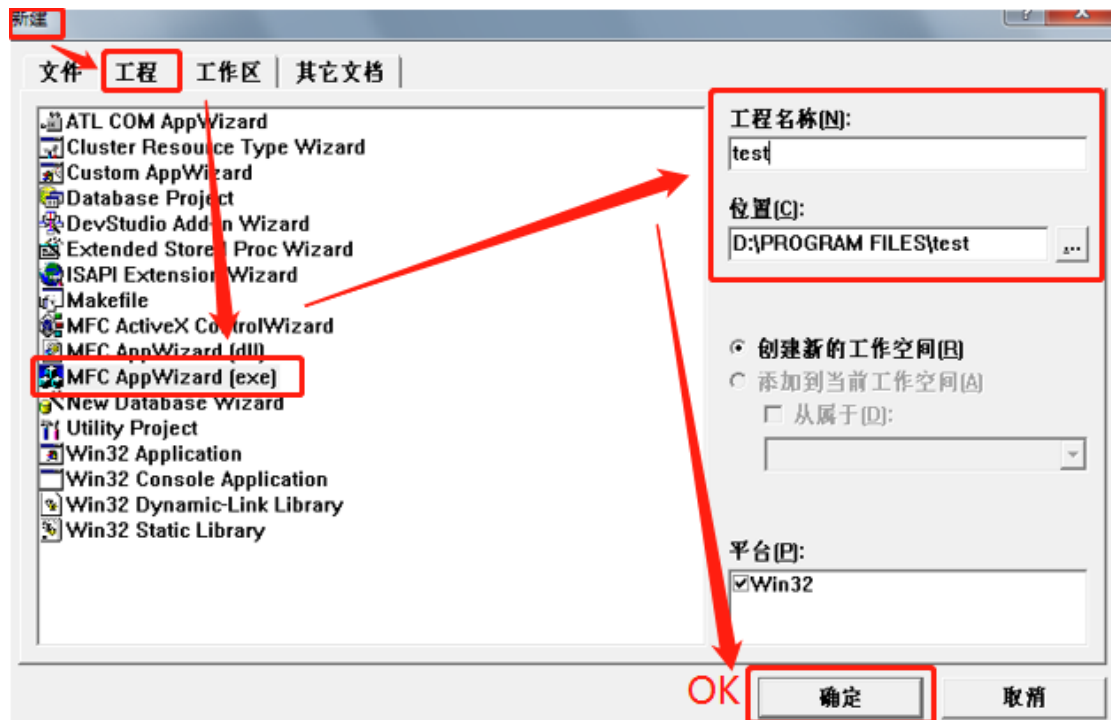
Chapter II How to Use & Encapsulate Function Library?

2.1. How to Call Function Library

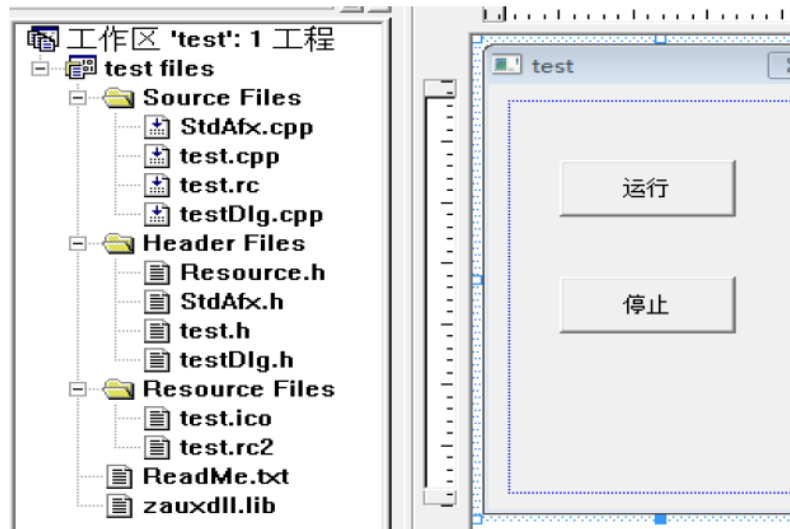
2.1.1. Windows System

VC++6.0

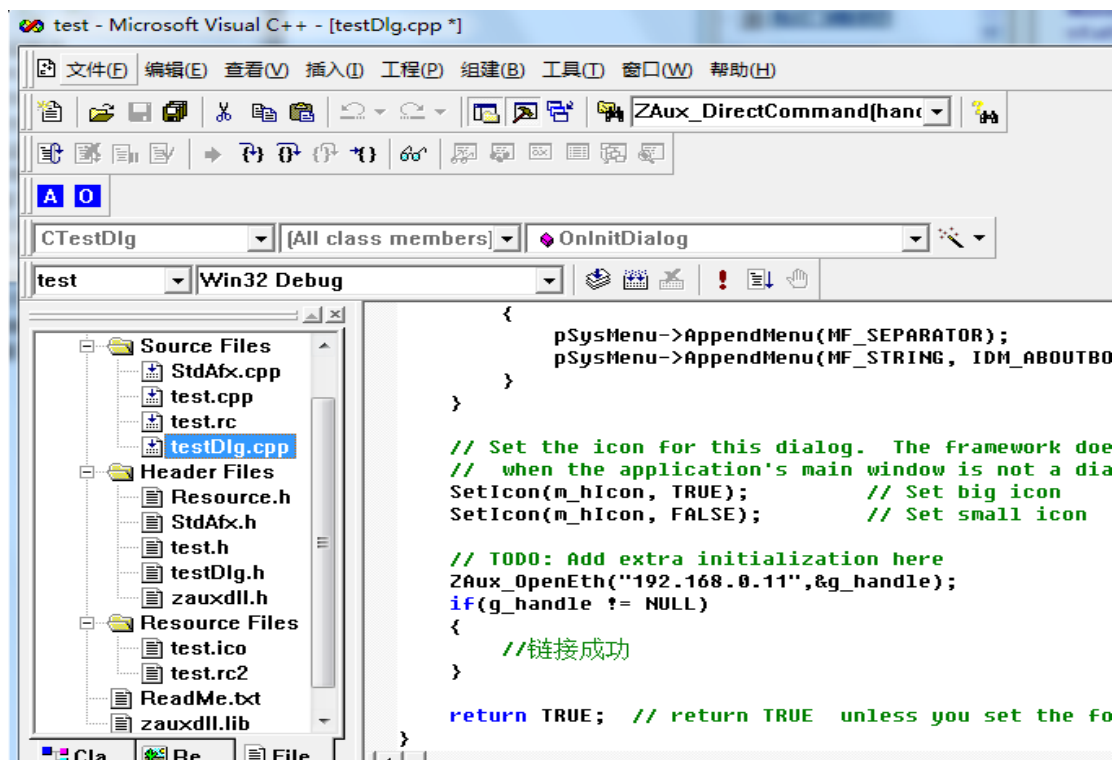
- 1) Open VC++6.0.
- 2) Build a new project.
- 3) Select [MFC AppWizard(exe)]
- 4) Select project storage route.
- 5) Set the name of project, select "OK".



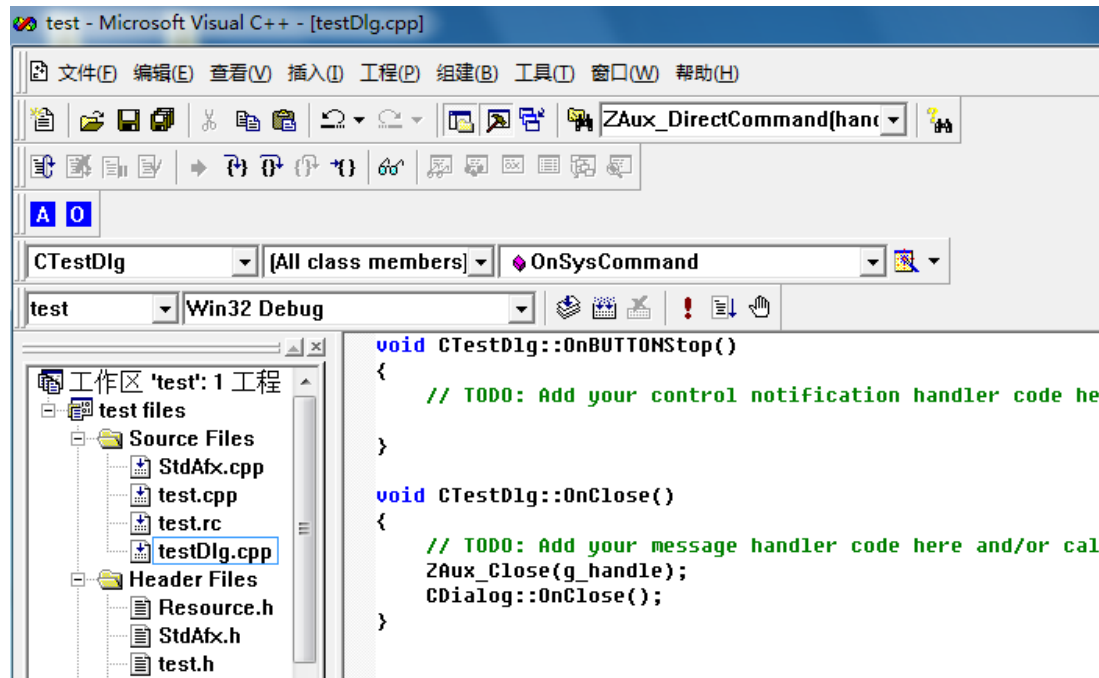
- 6) Choose "basic" procedure type, then new project is created.
- 7) Put libraries: zauxdll.h, zauxdll.lib, zauxdll.dll and zmotion.dll under the project path, or directly to import source code through zauxdll.cpp.



- 8) Add header file #include "zauxdll.h" under testDlg.cpp.
- 9) Find the CTestDlg::OnInitDialog() function, and add controller link code: ZAux_OpenEth("192.168.0.11",&g_handle).



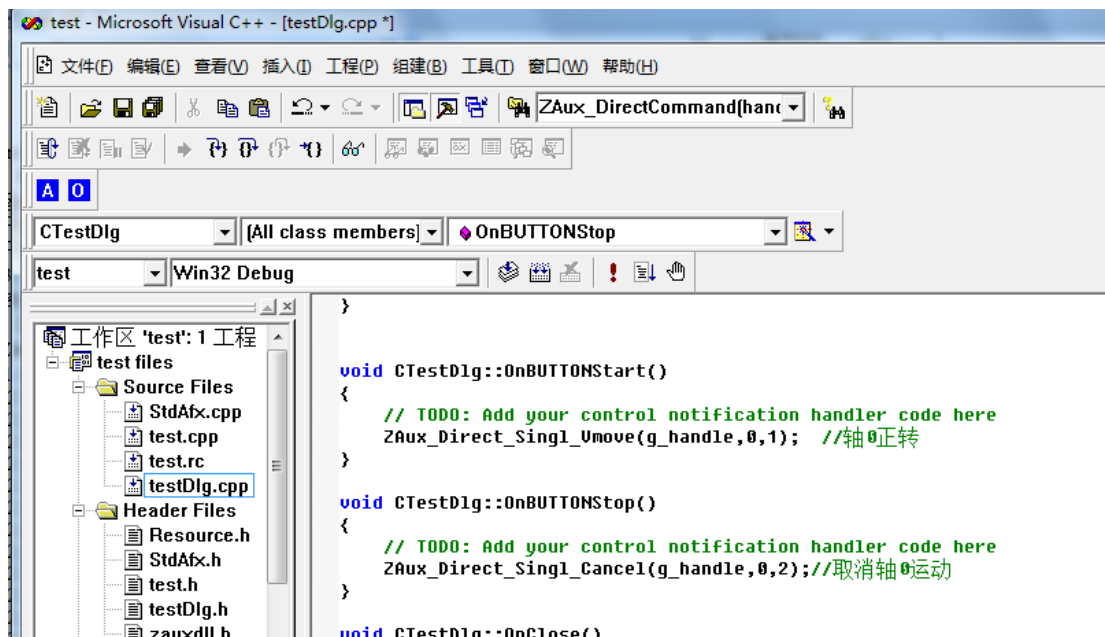
- 10) Add a member function: CTestDlg::OnClose() under CTestDlg to close window and disconnect controller link, and add function: ZAux_Close(g_handle) under CTestDlg::OnClose() to disconnect link.



11) Double click to start or stop to add relative codes under its corresponding event function:

ZAux_Direct_Single_Vmove(g_handle,0,1);

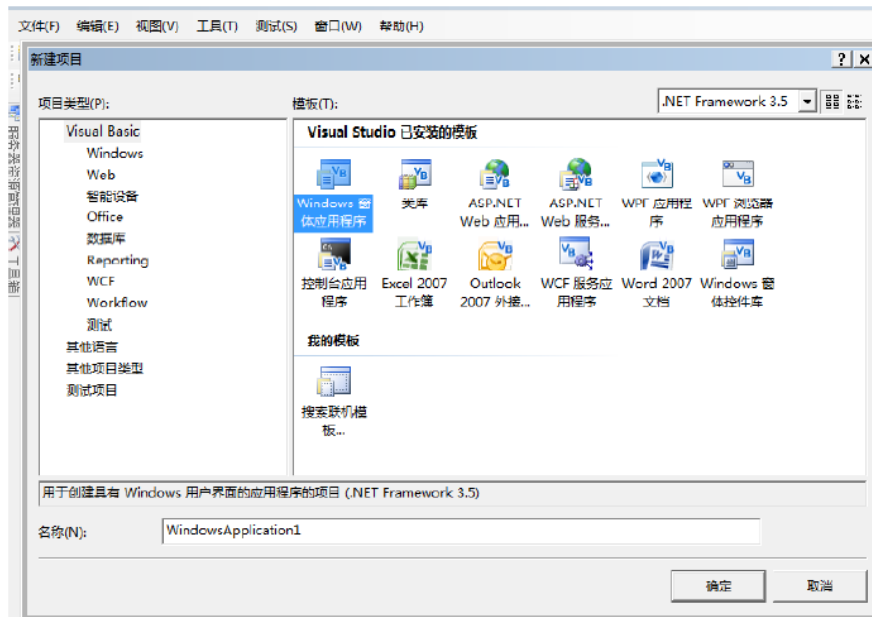
ZAux_Direct_Single_Cancel(g_handle,0,2).



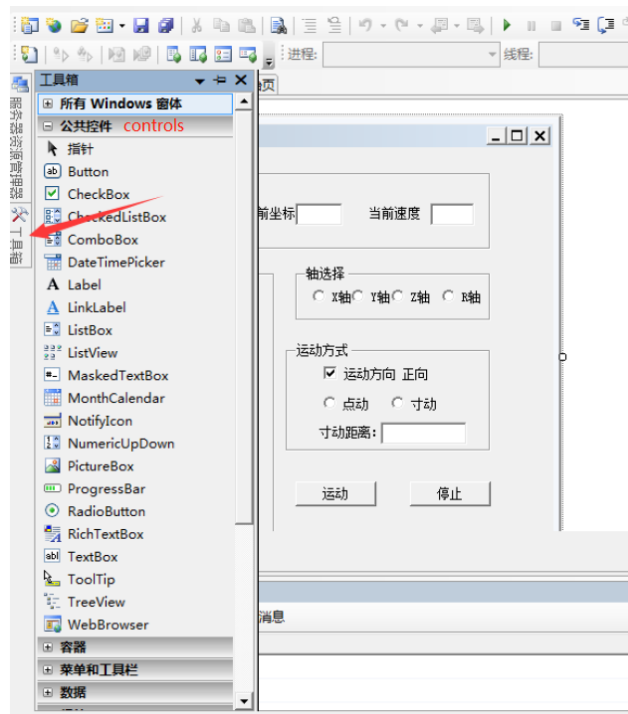
12) After the program was compiled, then run the program. Press "run" button on interface, axis 0 will move forward at its initial speed, press "stop" button, then axis 0 stop moving.

VB.NET

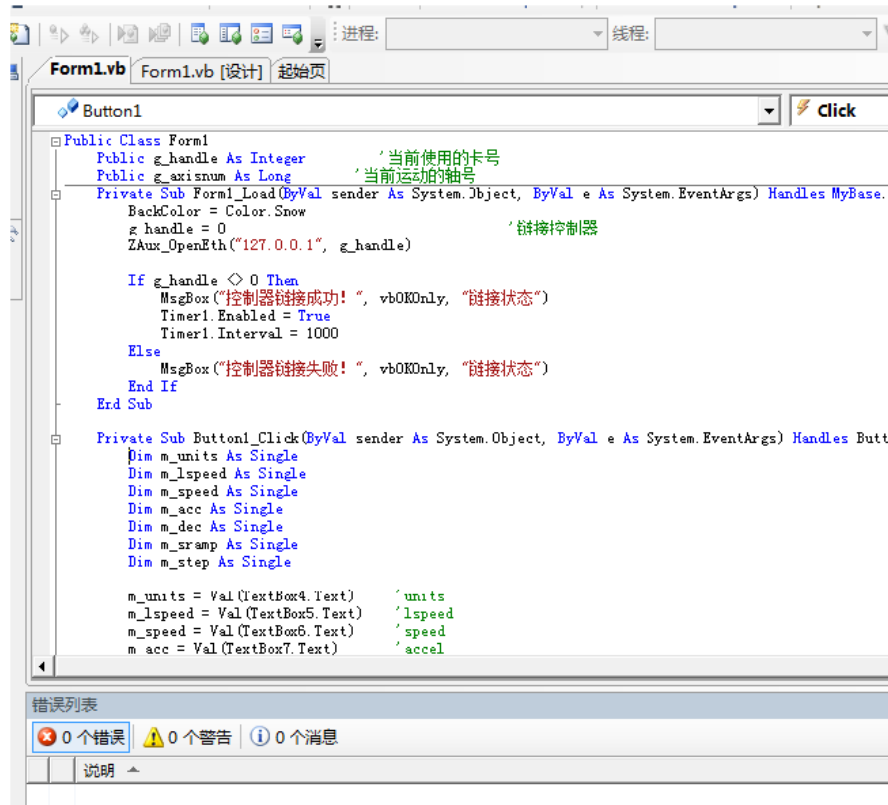
- 1) Build new project, and save.



- 2) Select corresponding widgets (controls) to complete operation interface according to required functions and interfaces.



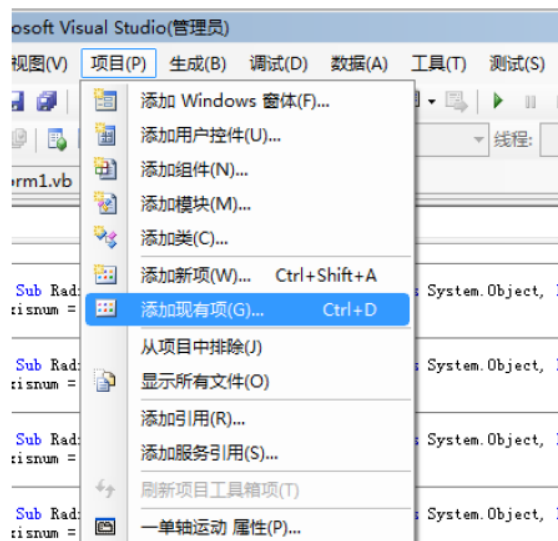
- 3) VB.Net is object-oriented programming language, choose reasonable trigger events to program, such as Button_Click, Checked, Load, etc.



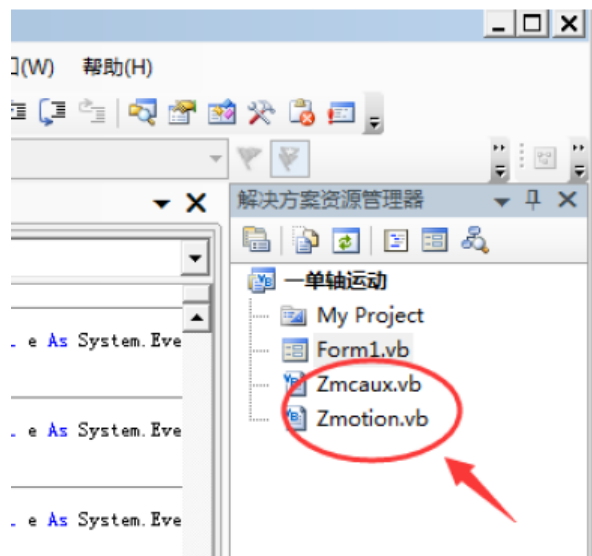
4) Add dll file and header file and put them in the same folder as the project file.

名称		修改日期	类型 ^	大小
zauxdll.dll	dll文件 dll file	2016/12/5 14:22	DLL 文件	1,445 KB
zmotion.dll		2016/12/5 14:22	DLL 文件	204 KB
Zmcaux.vb		2017/1/5 11:08	Visual Basic Sou...	143 KB
Zmotion.vb	zmotion库和辅助库头文件 zmotion library and auxiliary library head file	2015/6/10 15:28	Visual Basic Sou...	62 KB

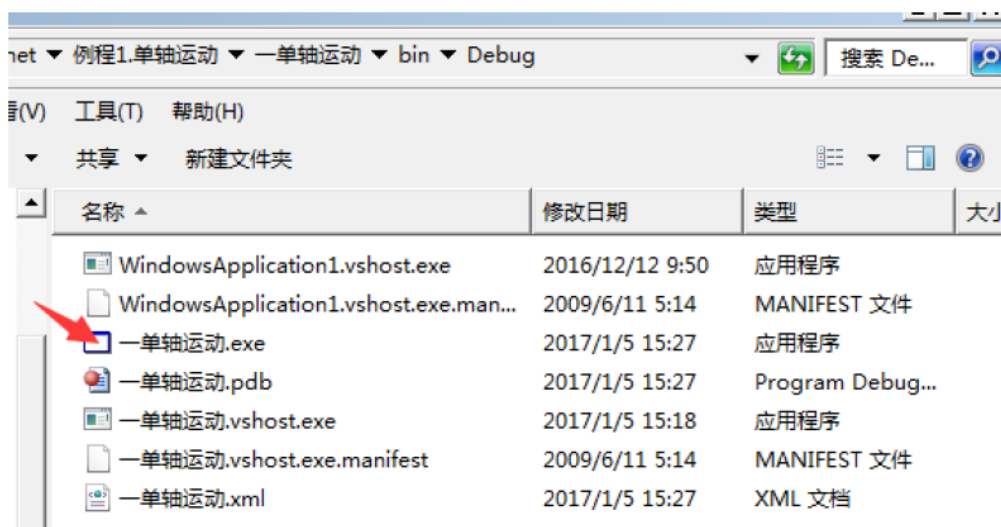
5) Add the header file in the project (2 .vb files)



After add it successfully, it will show:

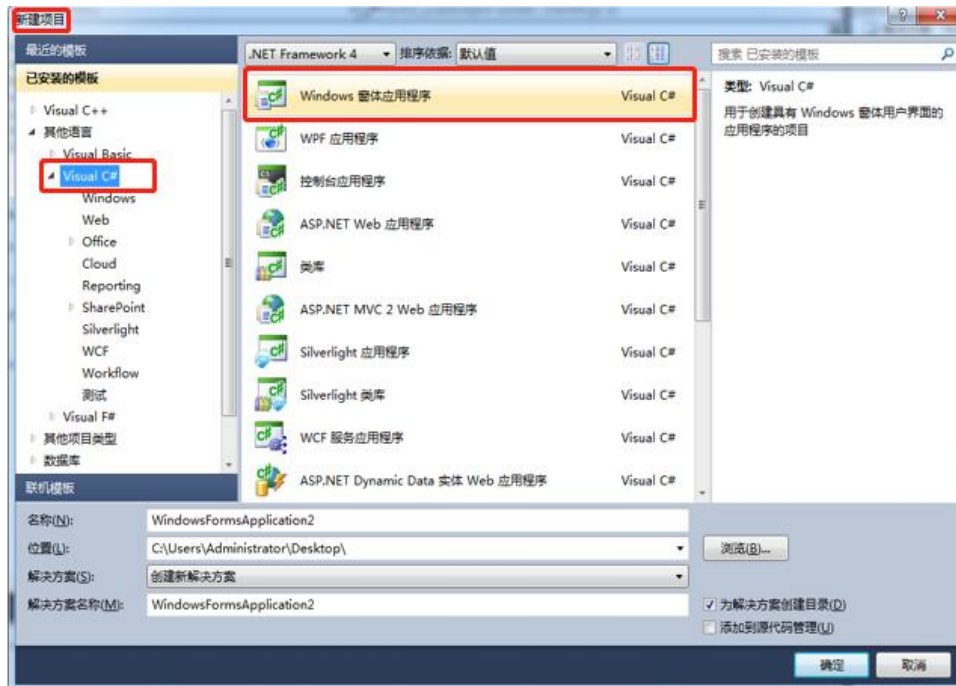


- 6) After PC links to controller successfully, then set IP well, start to debug.
- 7) After procedure was run and debugged, there is a .exe file in folder: "bin" \ "Debug", click it, then it can operate directly.



C#

- 1) Open VS2012, click "File" – "New" – "Project", select "windows application" type.



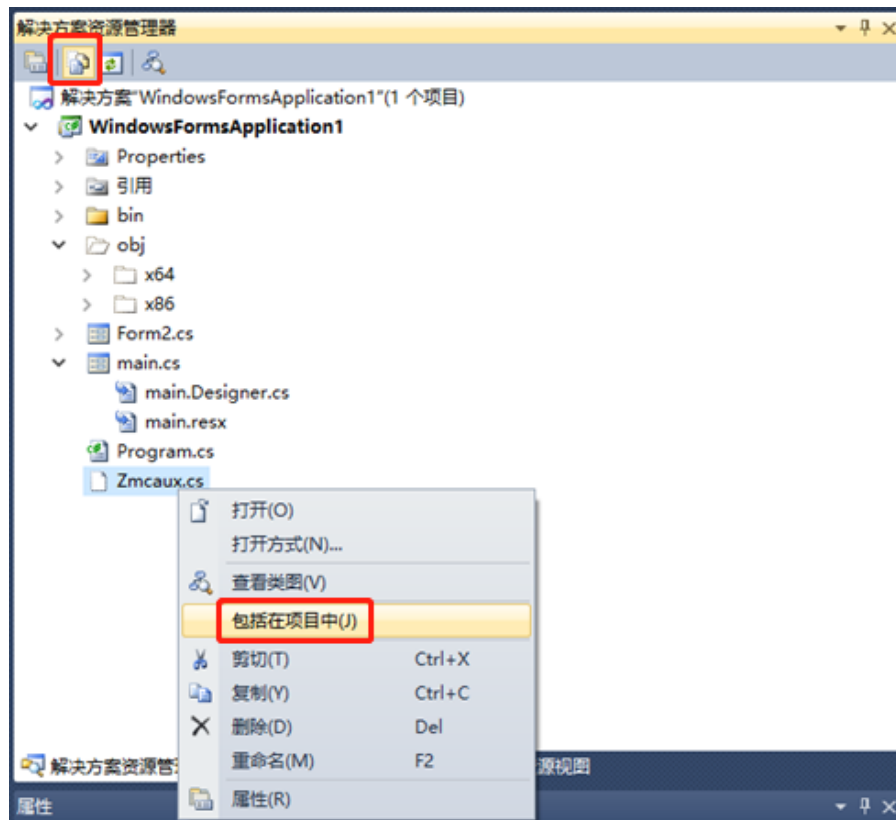
- 2) Find the folder built just now, then put file zmcaux.cs into folder.

名称	修改日期	类型	大小
bin	2022/5/14 9:36	文件夹	
obj	2022/5/14 9:36	文件夹	
Properties	2022/5/14 9:10	文件夹	
Form2.cs	2022/6/29 9:53	Visual C# Sourc...	1 KB
Form2.Designer.cs	2022/6/29 9:53	Visual C# Sourc...	2 KB
main.cs	2022/7/20 19:36	Visual C# Sourc...	63 KB
main.Designer.cs	2022/7/19 9:20	Visual C# Sourc...	186 KB
main.resx	2022/7/19 9:20	Microsoft .NET ...	7 KB
Program.cs	2022/7/18 16:04	Visual C# Sourc...	1 KB
WindowsFormsApplication1.csproj	2022/7/18 16:05	Visual C# Projec...	7 KB
WindowsFormsApplication1.csproj.user	2022/5/14 14:13	Per-User Project...	1 KB
Zmcaux.cs	2022/6/29 9:09	Visual C# Sourc...	216 KB

- 3) Put files zaux.dll and zmotion.dll into the folder "debug".

名称	修改日期	类型	大小
test_function1.exe	2017/1/5 9:15	应用程序	34 KB
test_function1.pdb	2017/1/5 9:15	Program Debug...	40 KB
test_function1.vshost.exe	2017/1/6 9:08	应用程序	12 KB
test_function1.vshost.exe.manifest	2010/3/17 22:39	MANIFEST 文件	1 KB
zauxdll.dll	2016/12/27 8:40	应用程序扩展	1,445 KB
zmotion.dll	2016/12/27 8:40	应用程序扩展	204 KB

- 4) Use "VS" to open new built project, then click "display all files" in the right "Resource Manager", then right click zmcaux.cs file – "included in Project".



- 5) Double click "form 1", code editing interface will appear, now, write "using cszmcaux" at the beginning of the file.

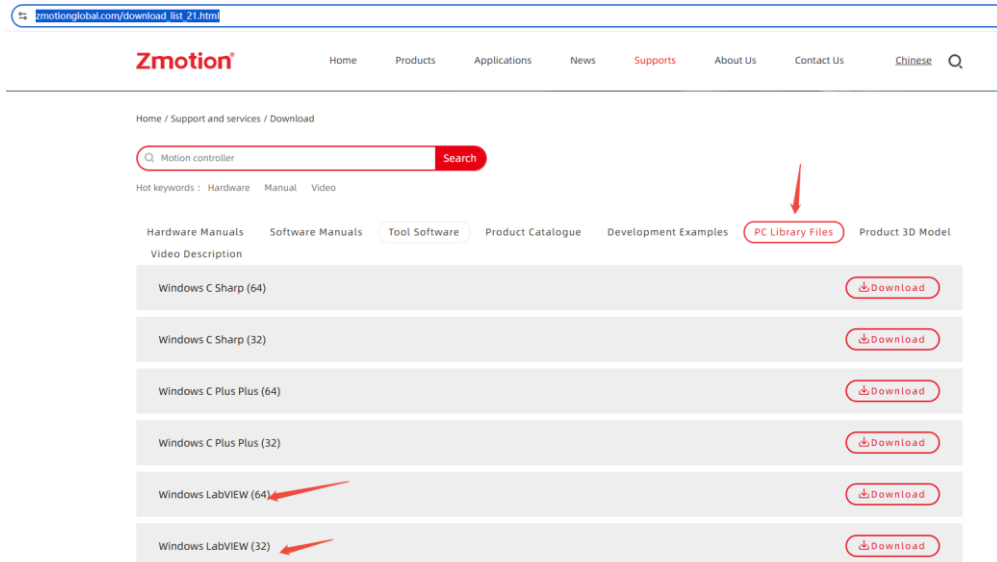
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Threading; //线程声明
using System.Text.RegularExpressions; //正则表达式声明

using cszmcaux;
namespace WindowsFormsApplication1
{
    public partial class main : Form
    {
        public IntPtr g_handle; //声明控制器连接句柄
    }
}
```

- 6) Now, it is time to program.

LABVIEW

- 1) Get / download (https://www.zmotionglobal.com/download_list_21.html) Zmotion LabVIEW VI library file.



[for the examples, please select "Development Examples"]

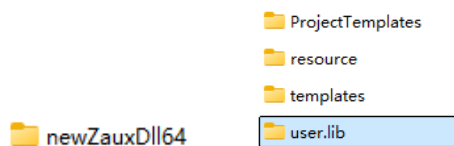
- 2) Copy the LabVIEW VI library file into LabVIEW installment file (LabVIEW / user.lib).
 - a. Find unzipped VI library folder (zaudll = 32-bit VI library, newZauxDll64 = 64 - bit VI library)



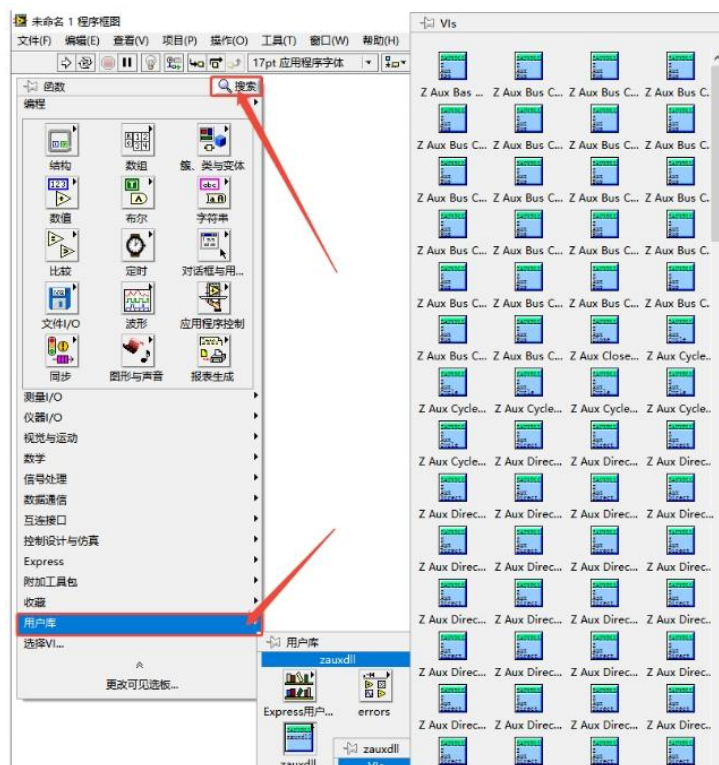
- b. Select LabVIEW, and right-click to choose "open file position" (here, take "LabVIEW2013-32" as example).



- c. Copy whole VI library folder into "user.lib", and turn off LabVIEW, then restart it.



- 3) After restart LabVIEW, choose "new build VI", and right click VI program empty, select user library for find added VI library, or directly search VI. At last, drag needed function to the board.



Python

1. Import dynamic library

- 1) In Python, please import platform (to gain the type of former operating system) and ctypes (to import related link library -- for dynamic library: C/C++) modules.
- 2) Choose related dynamic library(C/C++) according to different platforms, such as, libraries in widowsX86, windowsX64, Linux, etc.

```
1 import platform
2 import ctypes
3
4 systype = platform.system()
5 if systype == 'Windows':
6     if platform.architecture()[0] == '64bit':
7         zauxdll = ctypes.WinDLL('./zauxdll64.dll')
8         print('Windows x64')
9     else:
10        zauxdll = ctypes.WinDLL('./zauxdll.dll')
11        print('Windows x86')
12 elif systype == 'Darwin':
13     zmcddl = ctypes.CDLL('./zmotion.dylib')
14     print("macOS")
15 elif systype == 'Linux':
16     zmcddl = ctypes.CDLL('./zmotion.so')
17     print("Linux")
18 else:
19     print("OS Not Supported!!")
20
```

2. Python function encapsulation

- 1) To do function encapsulation together with dynamic libraries.
- 2) According to requirements, encapsulate corresponding functions.

```
22 class ZAUWrapper:
23     def __init__(self):
24         self.handle = ctypes.c_void_p()
25         self.sys_ip = ""
26         self.sys_info = ""
27         self.is_connected = False
28
29     def search(self, console=[]):
30         iplist = ctypes.create_string_buffer(b'', 1024) # create_string_buffer创建的是一个 ANSI 标准的 C类型字符串
31         zauxdll.ZAux_SearchEth(ctypes.byref(iplist), 1024, 200)
32         s = iplist.value.decode()
33         str_iplist = s.split()
34         num = len(str_iplist)
35         print(num, "Controller(s) Found:")
36         print(*str_iplist, sep='\n')
37         console.append("Searching...")
38         console.append(str(num) + " Controller(s) Found:")
39         return str_iplist, num
40
41     def connect(self, ip, console=[]):
42         if self.handle.value is not None:
43             self.disconnect()
44         ip_bytes = ip.encode('utf-8')
45         p_ip = ctypes.c_char_p(ip_bytes)
46         print("Connecting to", ip, "...")
47         ret = zauxdll.ZAux_OpenEth(p_ip, ctypes.pointer(self.handle))
48         msg = "Connected"
49         if ret == 0:
50             msg = ip + " Connected"
51             self.sys_ip = ip
```

Description: through this manual, you could know how to encapsulate needed functions.

3. Encapsulate Basic instructions in Python.

For example:

```

85     def vmove(self, iaxis, idir):
86         cmdbuffer = (ctypes.c_char * 2048)()
87         str1 = "VMOVE(%d) AXIS(%d)" % (idir, iaxis)
88         cmd = ctypes.c_char_p(bytes(str1, 'utf-8'))
89         zauxdll.ZAux_DirectCommand(self.handle, cmd, cmdbuffer, 2048)
90         return
91
92     def get_units(self, iaxis):
93         cmdbuffer = (ctypes.c_char * 2048)()
94         str(cmdbuffer, encoding='utf-8')
95         str1 = "?units"
96         cmd = ctypes.c_char_p(bytes(str1, 'utf-8'))
97         zauxdll.ZAux_DirectCommand(self.handle, cmd, cmdbuffer, 2048)
98         a = list(cmdbuffer)
99         b = b''.join(a)
100        c = b.decode().strip(b'\x00'.decode())
101        return c

```

Description: some functions can be combined with basic instructions to form string commands, and program through string commands & ZAux_DirectCommand command.

4. Use python function

For example:

The screenshot shows the PyCharm IDE with a project named 'untitled1'. The file explorer on the left shows the project structure, including 'venv', 'Scripts', and 'test.py'. The main editor displays the code for 'test.py', which includes the 'vmove' and 'get_units' functions from the previous example, followed by a sequence of operations: connecting to the device, getting axis positions, setting axis positions, getting axis units, and printing the results. The 'Run' window at the bottom shows the output of the script, which matches the expected results from the code.

```

95     str(cmdbuffer, encoding='utf-8')
96     str1 = "?units"
97     cmd = ctypes.c_char_p(bytes(str1, 'utf-8'))
98     zauxdll.ZAux_DirectCommand(self.handle, cmd, cmdbuffer, 2048)
99     a = list(cmdbuffer)
100    b = b''.join(a)
101    c = b.decode().strip(b'\x00'.decode())
102    return c
103
104    zaux = ZMCRWrapper()
105    zaux.connect("192.168.0.218")
106    pos = zaux.get_ax_pos(2)
107    print("Axis 0 MPOS:", pos[0])
108    print("Axis 1 MPOS:", pos[1])
109    pos1 = zaux.set_ax_pos(0, 5)
110    pos2 = zaux.get_ax_pos(0, 2)
111    print("modbus_reg(0):", pos2[0])
112    print("modbus_reg(1):", pos2[1])
113    zaux.vmove(0, 1)
114    number = zaux.get_units(0)
115    print("Axis 0 units:", number)
116
117
118

```

```

Run: test
C:\Users\Administrator\PycharmProjects\untitled1\venv\Scripts\python.exe C:\Users\Administrator\PycharmProjects\untitled1\test.py
Windows x86
Connecting to 192.168.0.218 ...
Axis 0 MPOS: 1024.0
Axis 1 MPOS: 500.0
modbus_reg(0): 300
modbus_reg(1): 400
Axis 0 units: 1000
Process finished with exit code 0

```

Description: build the object to be encapsulated, then call relative functions.

2.1.2. LINUX System

QT

- 1) Start Qt Creator 2.6.1, and build a new project.
- 2) Copy the dynamic link library (**zmotion.so**) and head file (**zmotion.h**) into folder “..\linux\function library\x86” of product matched disk into project file.
- 3) In the Qt Creator interface, open .pro file and add.

```
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets (version compatibility problem)
SOURCES += main.cpp\
    mainwindow.cpp\
    zaux.cpp
HEADERS += mainwindow.h\
    zmotion.h\
    zaux.h
```

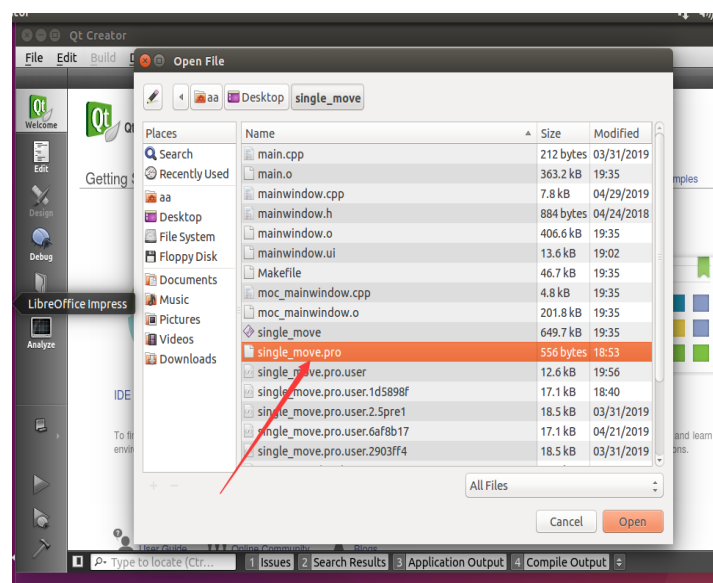
Modify the header file of main.c to `#include <QApplication>`

- 4) The way to add the dynamic library
 - 1> right click the project.
 - 2> click “add the library”.
 - 3> select the second external library, click OK.
 - 4> Click “browse library file”.

Note: dynamic library name should start with “lib” to make it identifiable. Then click “next”, information of dynamic library will be added into .pro automatically. Following, you can call the header file that includes function, and run program.

- 5) Open QT program, click file, find the .pro file in route, then click open.

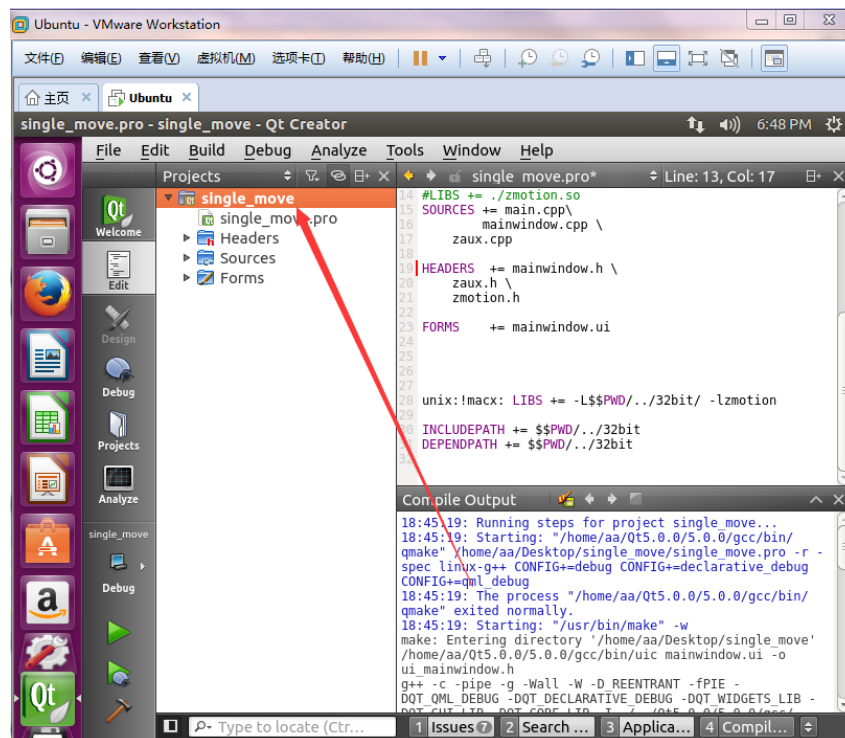
Note: path can't contain Chinese characters.



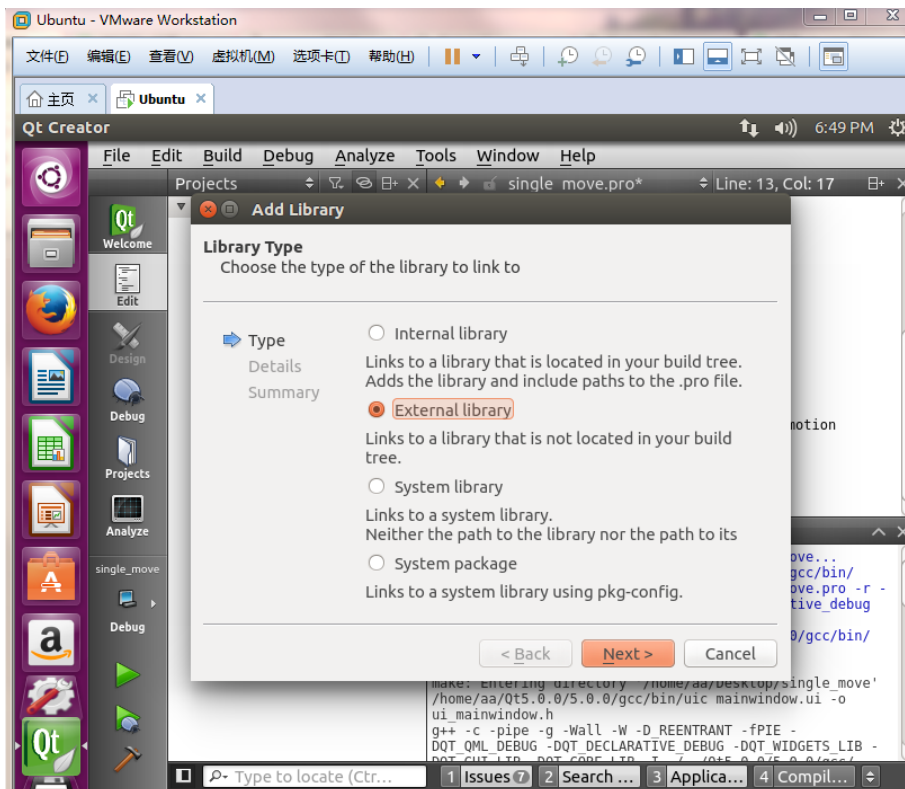
- 6) Add .pro of QT route in right path dynamic library, double click single_move.pro file, and check the dynamic library path set before, now, set own dynamic library path.



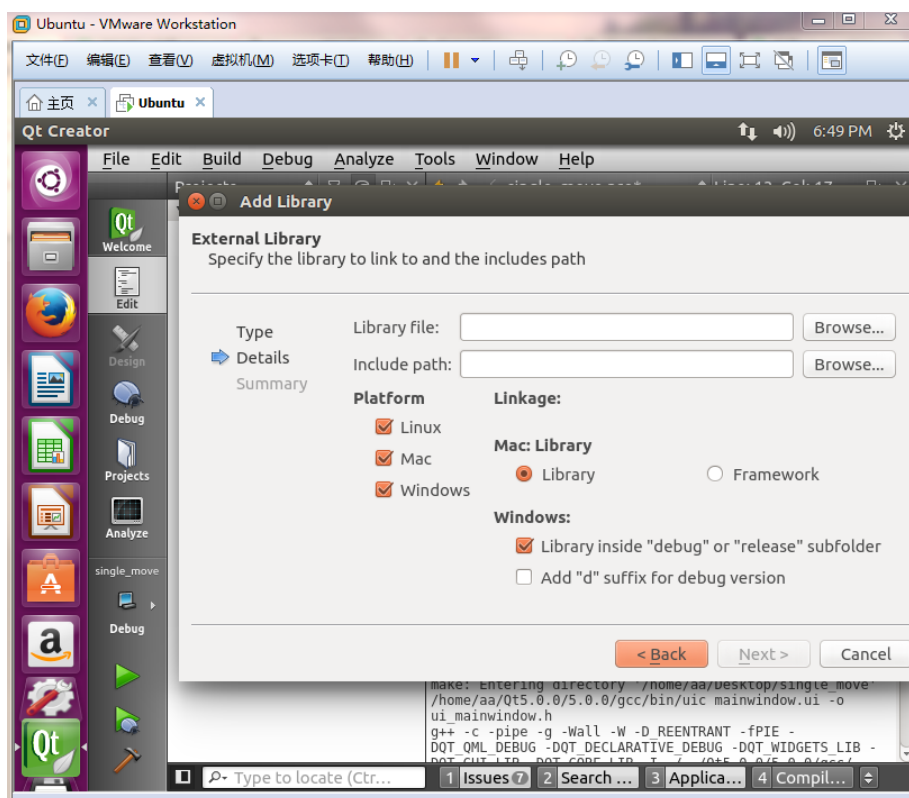
- 7) Delete dynamic library path in .pro file.
8) Put 32-bit dynamic library of folder into assigned path.
9) Add own dynamic library path, firstly select the project file.



10) Then right click add library.

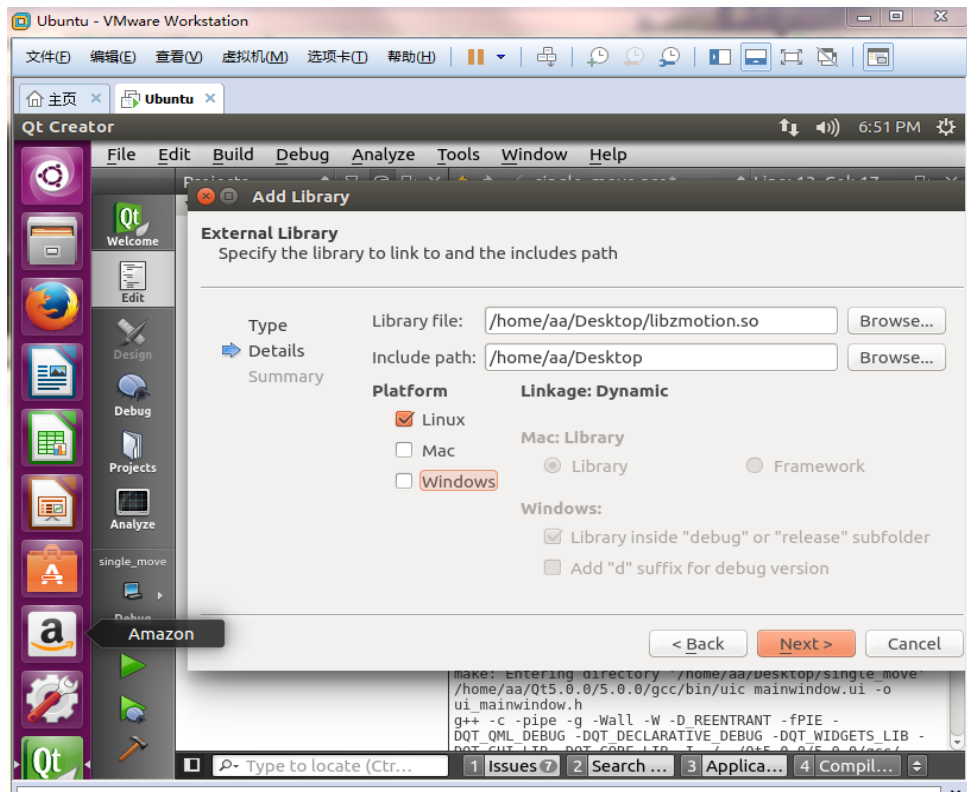


11) Select the second "Details" add external library.

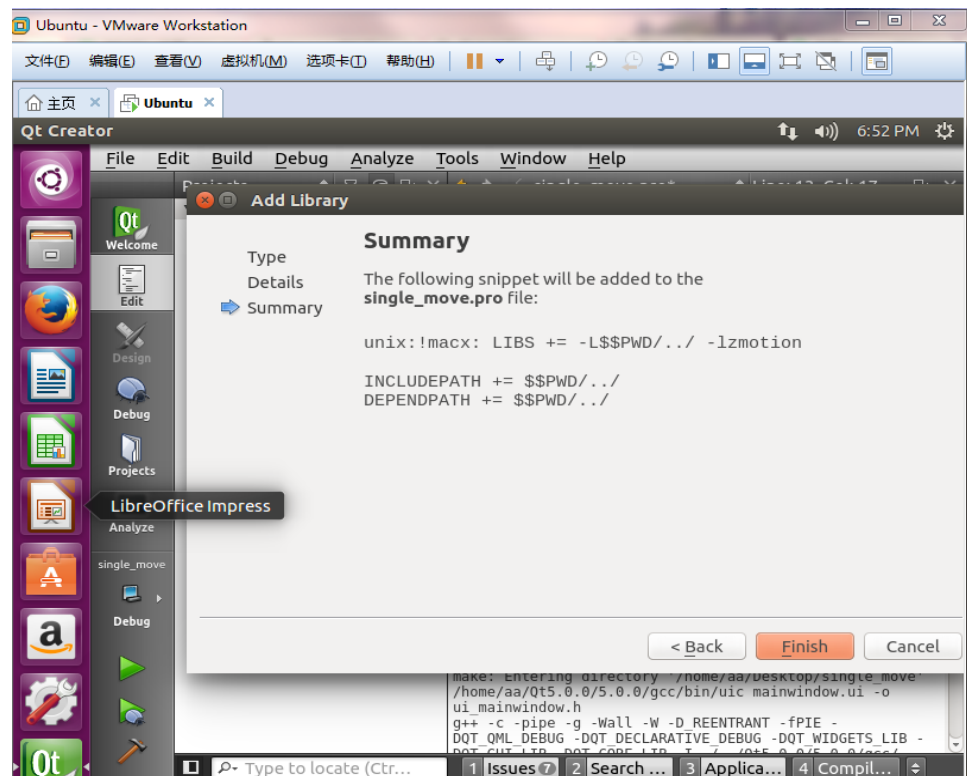


12) Now, find the path that saved the dynamic library just now.

Note: dynamic library name of Linux should start with "lib" to make it identifiable.
Don't change the file name casually.



13) Click "next".



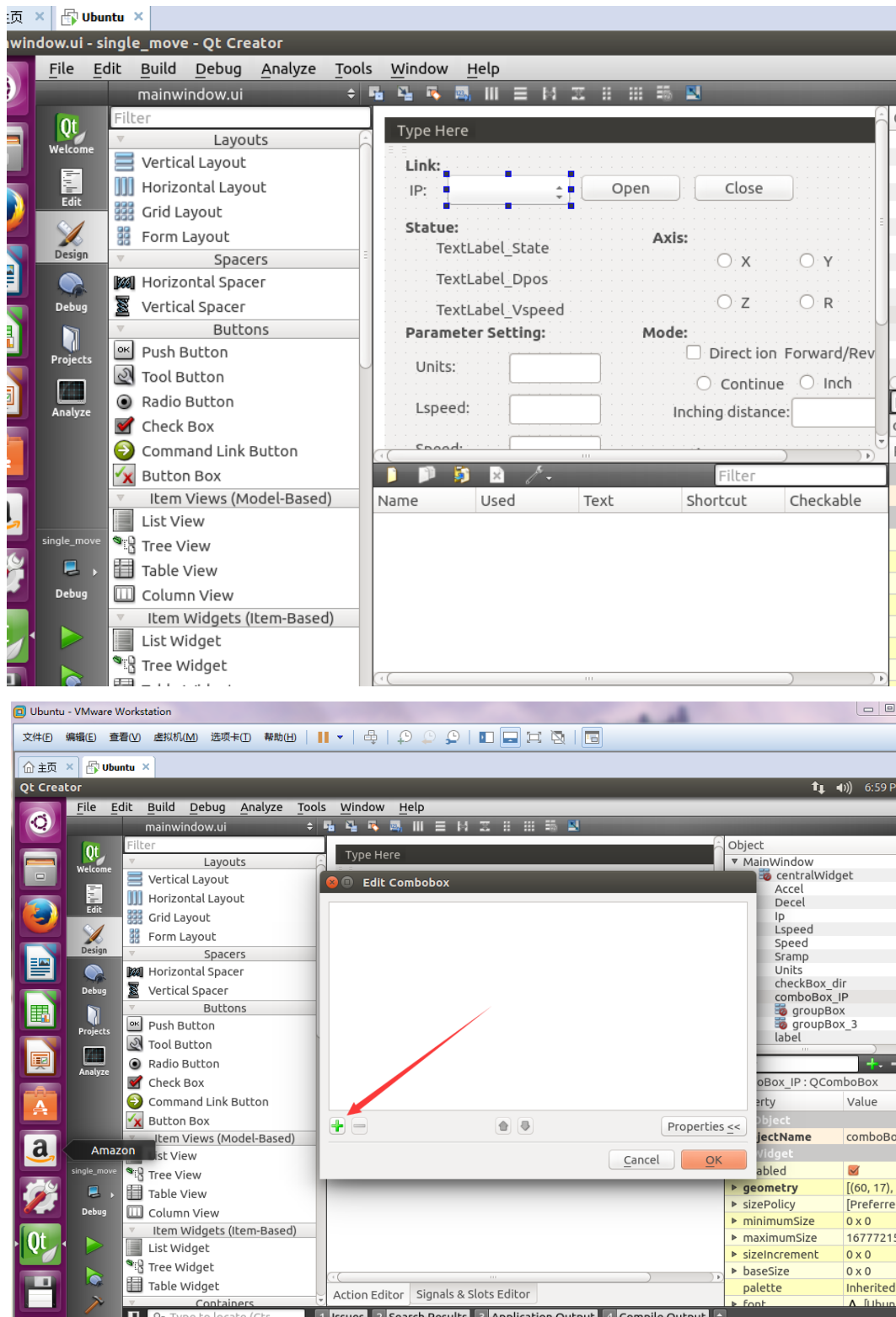
The screenshot shows the Qt Creator IDE with the project 'single_move.pro' open. The 'Headers' section lists 'mainwindow.h', 'zaux.h', and 'zmotion.h'. The 'Sources' section lists 'mainwindow.cpp' and 'zaux.cpp'. The 'Forms' section lists 'mainwindow.ui'. The 'Compile Output' window shows the compilation process, including the command 'make' and the output of the linker.

The screenshot shows the Qt Creator IDE with a C++ project named 'single_move'. A 'no line' dialog box is open, displaying parameters for a motion. The parameters are as follows:

- Link:** IP: 192.168.0.11 (Buttons: Open, Close)
- Statue:** Idle: stop
- Dpos:** 0
- Vspeed:** 0
- Parameter Setting:**
 - Units: 1000
 - Lspeed: 0
 - Speed: 10
 - Accel: 100
 - Decel: 100
 - Srapm: 20
- Axis:** X (selected), Y, Z, R
- Mode:**
 - ☐ Direct ion Forward/Reverse
 - ☒ Continue ☐ Inch
 - Inching distance: 10
- Opertion:** Run, Stop, Reset position

The background shows the Qt Creator interface with the 'single_move' project loaded. The 'no line' dialog box is a custom window for configuring motion parameters.

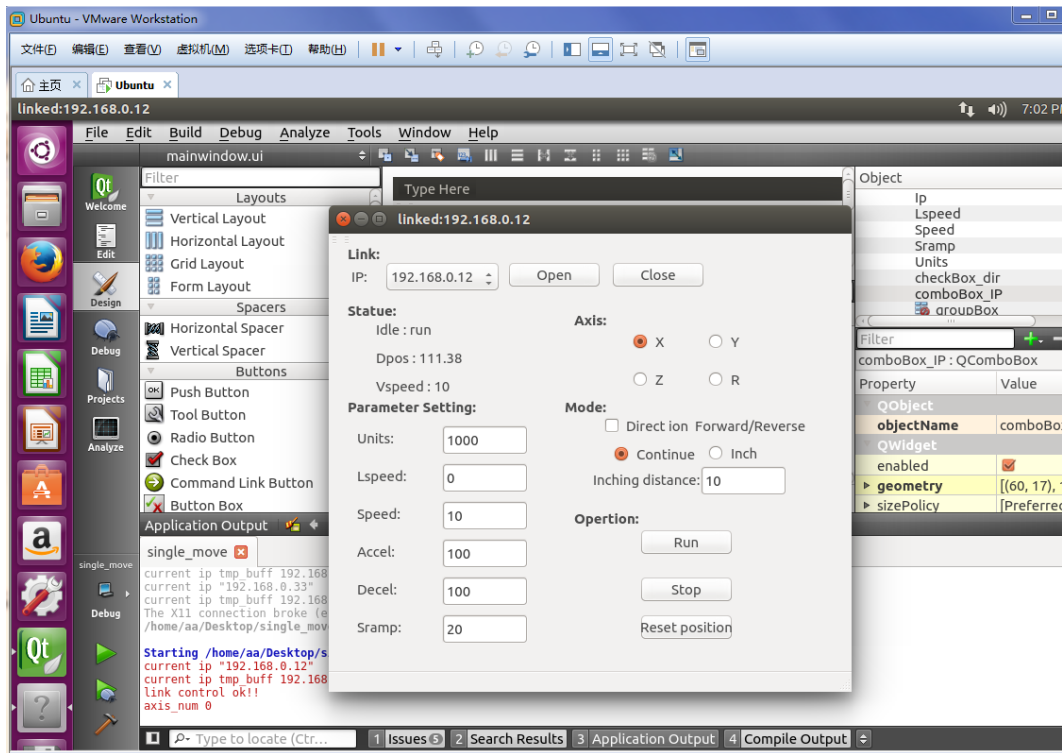
16) IP can be added manually, open ui interface, double click button.



17) Controller's IP network segment is 192.168.0.1

The default IP of controller is 192.168.0.11

Next, you can use the route, click "open" to connect to IP, then click "Run".



2.1.3. Wince System

C#

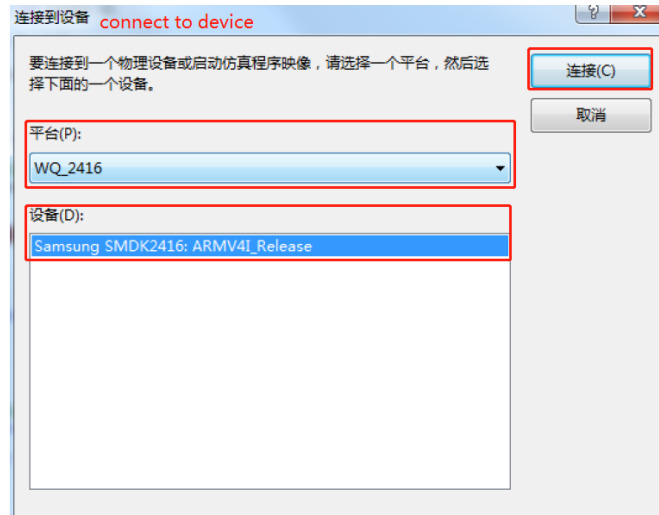
This route uses wince 6 mobile device as external connection device, and uses USB as connection main line.

- 1) Firstly, it needs to install the synchronization device driver and USB driver on the PC (not explained).
- 2) Then, use USB cable to build a connection with PC, device will connect to PC automatically. After connecting, Wince will show below information "Connected".

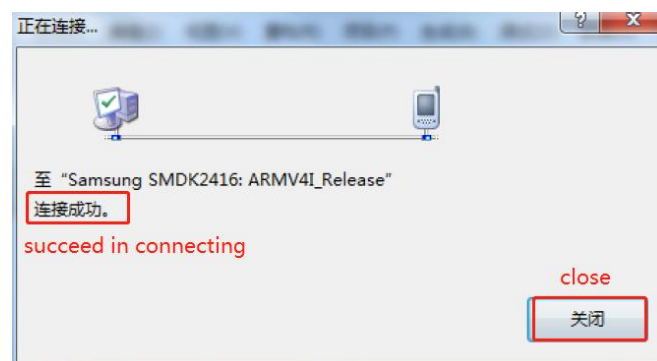


3) Now, connect to VS2008:

- 1> Click "tool" >> "connect to device".
- 2> Open below window, select "platform (P)" and "device (D)" at the same time, then click "connect" button.

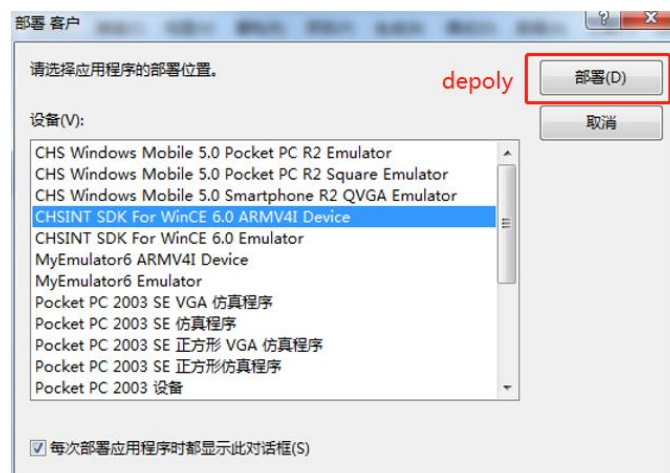


3> After connecting successfully, see below:



4> Click "close"/"OFF".

- 4) Start to debug / run program (here, debug is introduced).
- 5) Click "debug", then start to deploy platform.
- 6) Choose platform in image and click "deploy", then it will deploy automatically.

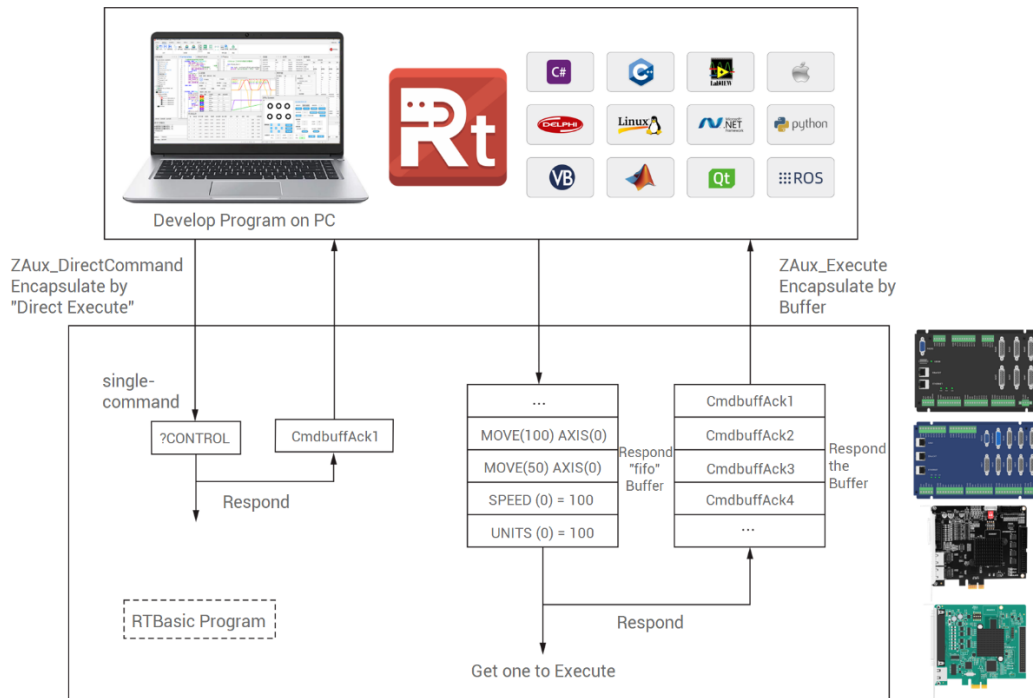


- 7) After deploying successfully, device will show the design window, it's time to debug and run.

- 8) After window code is finished, link wince6 and PC through USB, click "debug" or "run" to enter design window directly in device.
- 9) on wince6 interface, we can operate controller directly.
(**Note:** make sure IP of controller and Wince6 and PC are in the same segment. And when set Wince6 IP, don't set its gateway).
- 10) click "Search IP" to find available device IP, then select searched IP in the drop-down list, there needs 1~3 seconds, then choose required IP address to connect.
- 11) click "Connect", it will link with selected IP address. If succeeds, it will display "controller connected", or will display: "fail to connect, check IP".
- 12) Click "Axis X, Y, Z, R" separately to select motion axes, and they relate to "axis 0, 1, 2, 3", then we can choose motion mode "Jog or continuous", click "Move" to start move.
- 13) "Move / Stop / Position Clear" widgets are control buttons of the whole route, namely, they can control program run, stop, etc. (when controller is in disconnection, these key buttons will print "fail to connect with controller")
- 14) When operation is finished, click "disconnect" to break the link between device and controller.

2.2. How to Encapsulate Auxiliary Library?

2.2.1. Auxiliary Library Encapsulation Methods



In ZAux library, motion functions and basic parameters are encapsulated, and there are source codes for your reference.

ZAux library can send "RTBasic Commands" to controllers through ZAux_Execute or ZAux_Direct_Command. If you want to use unencapsulated commands, or you want to encapsulate customized functions, also, ZAux_Execute or ZAux_DirectCommand can be used, or modify / add corresponding functions according to existed codes.

There are 2 ways to send character string command: buffer mode and direct mode.

➤ **Direct Mode:**

Directly execute one single related command of variable / array / parameter, now, all transferred parameters must be exact values (not the expression). Please refer to ZAux_DirectCommand.

➤ **Buffer Mode:**

In this mode, all commands can be executed, and an expression can be used as the parameter, but the speed will be slower, please refer to ZAux_Execute.

- Example (set axis speed function): ZAux_Direct_SetSpeed (ZMC_HANDLE handle, int iaxis, float fValue), which is same as "SPEED (axis No.) = speed value" in RTBASIC. Then, when encapsulating, the command string is "speed(%d) = %f".

Encapsulation Steps:

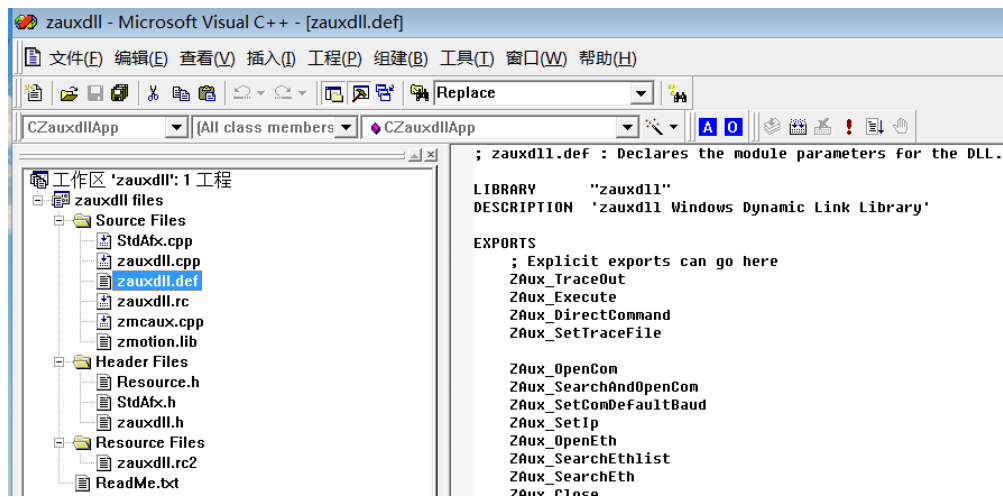
- (1) Edit source code: open auxiliary source code, add the function in "zmcaus.cpp".

```
int32 __stdcall ZAux_Direct_SetSpeed(ZMC_HANDLE handle, int iaxis, float fValue)
{
    char cmdbuff[2048];
    char cmdbuffAck[2048];
    if( iaxis > MAX_AXIS_AUX)
    {
        return ERR_AUX_PARAERR;
    }

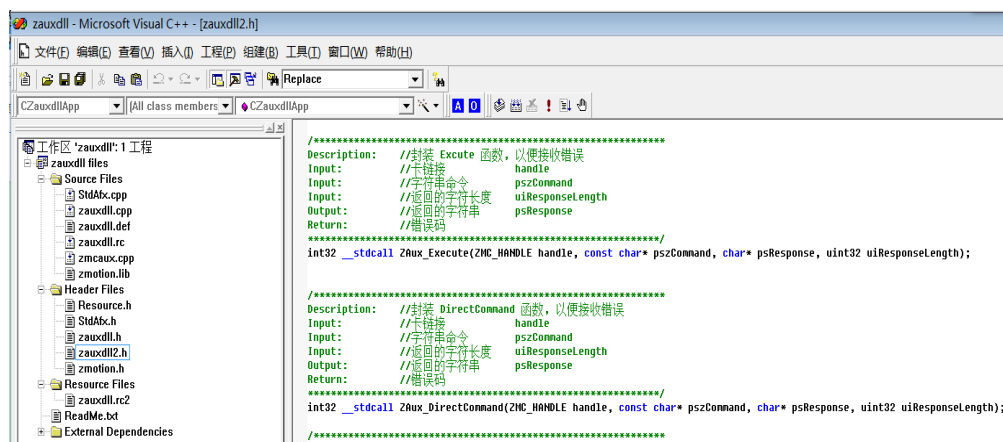
    //generate the command
    sprintf(cmdbuff, "SPEED(%d)=%f", iaxis, fValue);

    //call the command to execute the function
    return ZAux_DirectCommand(handle, cmdbuff, cmdbuffAck, 2048);
}
```

- (2) Define new added function definition in zauxdll.def.



- (3) Add corresponding declaration in the head file.



- (4) Compile new generated DLL.

2.2.2. Encapsulation Examples

Direct Get Multi-Type Data

If you want to obtain many kinds of data, such as DPOS (axis command position), MPOS (axis measurement position), IO signals, etc., generally they are obtained by a variety of separate and independent functions. Such accumulation (so many times of reading and writing) will cause the program to **freeze**. In order to improve the performance of a high-level program, **you can define one own function to quickly transfer data to the high-level program**, instead of obtaining different types of data through multiple cycles.

For example: suppose there is a simple 3-axis platform, it is necessary to read:

- DPOS of axis 0, axis 1, axis 2
- MPOS of axis 0, axis 1, axis 2
- AXISSTATUS of axis 0, axis 1, axis 2
- IN (0) ~ IN (32)
- OUT (0) ~ OUT (33)

The program for obtaining data is as follows:

```

拼接的字符串:
?DPOS(0),DPOS(1),DPOS(2),MPOS(0),MPOS(1),MPOS(2),AXISSTATUS(0),AXISSTATUS(1),AXISSTATUS(2),IN(0,31),IN(32,32),OUT(0,31),
OUT(32,33)
获取到的轴命令位置:
轴0 :1.000000 轴1 :2.000000 轴2 :3.000000
获取到的轴反馈位置:
轴0 :1.000000 轴1 :2.000000 轴2 :3.000000
获取到的轴状态(按位对应):
轴0 :512 轴1 :0 轴2 :1024
获取到的输入口状态:
IN(0):0 IN(1):0 IN(2):1 IN(3):0 IN(4):1 IN(5):0 IN(6):0 IN(7):0 IN(8):0
IN(9):0 IN(10):0 IN(11):0 IN(12):0 IN(13):0 IN(14):0 IN(15):0 IN(16):0
IN(17):0 IN(18):0 IN(19):0 IN(20):0 IN(21):0 IN(22):0 IN(23):1 IN(24):0
IN(25):0 IN(26):0 IN(27):0 IN(28):0 IN(29):0 IN(30):0 IN(31):0 IN(32):0
获取到的输出口状态:
OUT(0):0 OUT(1):0 OUT(2):0 OUT(3):0 OUT(4):0 OUT(5):0 OUT(6):0 OUT(7):0 OUT(8):0
OUT(9):0 OUT(10):0 OUT(11):0 OUT(12):0 OUT(13):0 OUT(14):0 OUT(15):1 OUT(16):0
OUT(17):0 OUT(18):0 OUT(19):0 OUT(20):0 OUT(21):0 OUT(22):0 OUT(23):0 OUT(24):0
OUT(25):0 OUT(26):0 OUT(27):0 OUT(28):0 OUT(29):0 OUT(30):0 OUT(31):0 OUT(32):1
OUT(33):0

```

```
// test1.cpp: define the entry point of control panel application program.
```

```
//
```

```
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"
```

```
void commandCheckHandler(const char *command, int ret)
{
```

```

    if (ret)//if it is not 0, it fails
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

/*****
Description: //directly get data function (by own customized)
Input:       //handle
              iaxisNum      total numbers of axes
              iaxislist     axis No. list
              fDposlist     output demand position value (DPOS)
              fMposlist     output measurement position value (MPOS)
              iAxisstatuslist output axis state position value, bit to bit
              startIn       starting IN No. to be got
              endIn         end IN No. to be got
              iIn           output IN state, bit to bit
              startOut      starting OUT No. to be got
              endOut        end OUT No. to be got
              iOut          output OUT state, bit to bit

Output:      //
Return:      //error codes
*****/
int Demo_Direct_MyGetData(ZMC_HANDLE handle,int iaxisNum, int* iaxislist, float* fDposlist,float*
fMposlist,int32* iAxisstatuslist,int startIn , int endIn,int *iIn,int startOut , int endOut,int *iOut)
{
    char cmdbuff[2048];
    char tempbuff[2048];
    char cmdbuffAck[20480];

    //If the transferred address is blank, exit.

    if(NULL == iaxislist || NULL == fDposlist || NULL == fMposlist || NULL == iAxisstatuslist || NULL
== iIn || NULL == iOut)
    {
        return ERR_AUX_PARAERR;
    }
    //If transferred end No. is smaller than starting No., exit.
    if ((endIn<startIn) || (endOut<startOut))
    {
        return ERR_AUX_PARAERR;
    }

    int ret=0;
    int i;
    //generate the command
    sprintf(cmdbuff, "?");
    //splice DPOS
    for (i=0;i<iaxisNum;i++)
    {
        sprintf(tempbuff,"DPOS(%d)",iaxislist[i]);//generate character string that corresponds to
the command

        strcat(cmdbuff, tempbuff);//splice character string

        if (strlen(cmdbuff)>1000)
        {
            return ERR_AUX_PARAERR; //wrong parameter, too long character string splicing
        }
    }
    //splice MPOS
    for (i=0;i<iaxisNum;i++)
    {

```

```
    sprintf(tempbuff,"MPOS(%d)",iaxislist[i]); //generate character string that corresponds to
the command

    strcat(cmdbuff, tempbuff); //splice character string

    if (strlen(cmdbuff)>1000)
    {
        return ERR_AUX_PARAERR; //wrong parameter, too long character string splicing
    }
}
//splice AXISSTATUS
for (i=0;i<iaxisNum;i++)
{
    sprintf(tempbuff,"AXISSTATUS(%d)",iaxislist[i]); //generate character string that
corresponds to the command

    strcat(cmdbuff, tempbuff); //splice character string
    if (strlen(cmdbuff)>1000)
    {
        return ERR_AUX_PARAERR; //wrong parameter, too long character string splicing
    }
}
int32 ostart,istart,iend,oend;          //32 at most in once time
bool addflag;

addflag=false;

int32 temp;          //32 at most in once time
int32 temp2;         //32 at most in once time

temp=endIn-startIn+1;
if (temp%32 == 0)
{
    temp=temp/32;
}
else
{
    temp=temp/32+1;
}

//splice IN
for (i=0;i<temp;i++)
{
    istart = startIn+32*i;
    iend =istart+31;
    if (iend>endIn)
    {
        iend=endIn;
    }
    //generate the command
    sprintf(tempbuff, "IN(%d,%d)", istart,iend);
    strcat(cmdbuff, tempbuff); //splice character string
    if (strlen(cmdbuff)>1000)
    {
        return ERR_AUX_PARAERR ; //wrong parameter, too long character string splicing
    }
}

temp2=endOut-startOut+1;
if (temp2%32 == 0)
{
    temp2=temp2/32;
```

```
}
else
{
    temp2=temp2/32+1;
}
//splice OUT
for (i=0;i<temp2;i++)
{
    ostart = startOut+32*i;
    oend =ostart+31;
    if (oend>endOut)
    {
        oend=endOut;
    }

    // generate the command
    sprintf(tempbuff, "OUT(%d,%d)", ostart,oend);

    strcat(cmdbuff, tempbuff); //splice character string
    if (i<temp-1)
    {
        strcat(cmdbuff, ","); //splice character string
    }

    if (strlen(cmdbuff)>1000)
    {
        return ERR_AUX_PARAERR; //wrong parameter, too long character string splicing
    }
}

printf("spliced string: \n",cmdbuff);
printf("%s\n",cmdbuff);

ret=ZAux_DirectCommand(handle,cmdbuff,cmdbuffAck,2048);
if(ERR_OK != ret)
{
    return ret;
}
//printf("%s\n",cmdbuffAck);
//printf("%d\n",strlen(cmdbuffAck));
//
if(0 == strlen(cmdbuffAck))
{
    return ERR_NOACK;
}
float ftempbuff[200];
int itempbuff[200];

ZAux_TransStringToFloat(cmdbuffAck,iaxisNum*2,ftempbuff); //character string is converted
to floating
//DPOS output
for(i=0;i<iaxisNum;i++)
{
    //printf("%f\n",ftempbuff[i]);
    fDposlist[i]=ftempbuff[i];
}
//MPOS output
for(i=0;i<iaxisNum;i++)
{
    //printf("%f\n",ftempbuff[i+iaxisNum]);
    fMposlist[i]=ftempbuff[i+iaxisNum];
}
```

```
}

ZAux_TransStringToInt(cmdbuffAck,iaxisNum*3+temp2+temp,itempbuff);//character string is
converted to integer
//AXISSTATUS output
for(i=0;i<iaxisNum;i++)
{
    //printf("%d\n",itempbuff[i+iaxisNum*2]);
    iAxisstatuslist[i]=itempbuff[i+iaxisNum*2];
}
//IN output
for(i=0;i<temp;i++)
{
    //printf("%d\n",itempbuff[i+iaxisNum*3]);
    iIn[i]=itempbuff[i+iaxisNum*3];
}
//OUT output
for(i=0;i<temp2;i++)
{
    //printf("%d\n",itempbuff[i+iaxisNum*3+temp]);
    iOut[i]=itempbuff[i+iaxisNum*3+temp];
}
return ERR_OK;
}
int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"127.0.0.1"; //controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("success to connect controller! \n");

    int axis[4]={0,1,2,4};
    float d_dpos[4];
    float d_mpos[4];
    int32 d_axis_status[4];
    int d_in[10];
    int d_out[10];

    ret =
    Demo_Direct_MyGetData(handle,3,axis,d_dpos,d_mpos,d_axis_status,0,32,d_in,0,33,d_out);
    int i;
    printf("obtained DPOS: \n");
    for (i=0;i<3;i++)
    {
        printf("\taxis%d :%f",i,d_dpos[i]);
    }
    printf("\n");
    printf("obtained MPOS: \n");
    for (i=0;i<3;i++)
    {
        printf("\taxis%d :%f",i,d_mpos[i]);
    }
    printf("\n");
    printf("obtained axis state(bit to bit): \n");
    for (i=0;i<3;i++)
    {
        printf("\taxis%d :%d",i,d_axis_status[i]);
```

```
}
printf("\n");

printf("obtained IN state: \n");
int j=0;
int tempval;
for (i=0;i<=32;i++)
{
    if (((i%32)==0)&&(i>0) )
    {
        j++;
    }
    //convert to bit
    tempval=d_in[j]>>(i-32*j);
    printf("  IN(%d):%d",i,tempval &(0x01));
    if (((i%8)==0)&&(i>0) )
    {
        printf("\n");
    }
}
printf("\n");

printf("obtained OUT state: \n");
j=0;

for (i=0;i<=33;i++)
{
    if (((i%32)==0)&&(i>0) )
    {
        j++;
    }
    // convert to bit
    tempval=d_out[j]>>(i-32*j);
    printf("  OUT(%d):%d",i,tempval &(0x01));
    if (((i%8)==0)&&(i>0) )
    {
        printf("\n");
    }
}
printf("\n");

Sleep(20000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;//whether the command is executed successfully?
printf("connection closed!\n");

handle = NULL;
return 0;
}
```

Buffer. Multi-Type Commands Execution by One Row Command

Generally, in dispensing industry and the woodworking industry, continuous trajectory is used, that is, there is an inserted buffer output between the continuous tracks. If the motion and the continuous track motion and the buffer output are sent

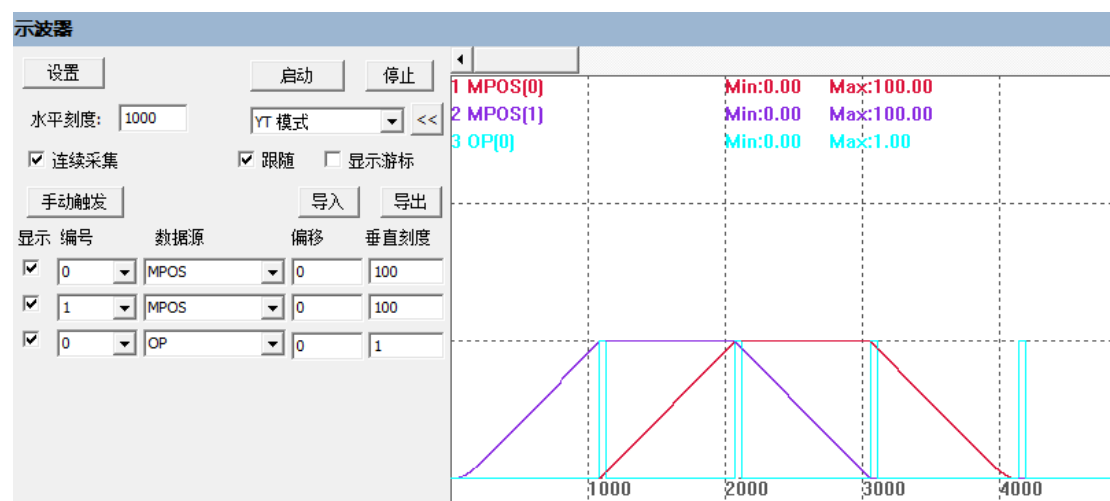
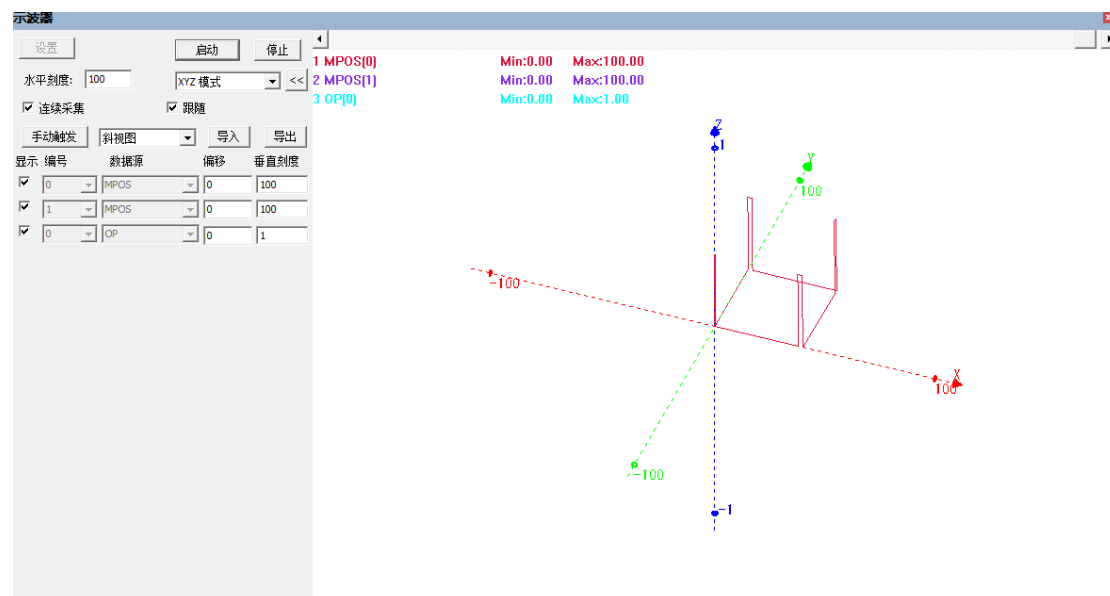
separately, it is inconvenient. Then, you can encapsulate one separate motion function that can execute multiple motions by one row command.

For Example:

Assuming to control an XY two-axis platform, it starts from the coordinate point (0,0) then moves a below trajectory:

- OUT0 outputs 50ms -- (100, 0)
- OUT0 outputs 100ms -- (100, 100)
- OUT0 outputs 50ms -- (0, 100)
- OUT0 outputs 50ms -- (0, 0)

Now, you can use one function to encapsulate and send it out quickly.



```
// test1.cpp: define entry point of control panel application program.
```

```
//
```

```
#include "stdafx.h"
```



```
#include <windows.h>
#include "zmotion.h"
#include "zaxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret)//if it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

/*****
Description: //customized motion function
Input:      //handle          card link/handle
            iMoveLen          filled motion length
            iaxisNum          the total axes in motion
            iaxislist         axis No. list
            fPoslist          distance list
            iout              buffer output port
            outlist           buffer output list (whether to output? 0 – no, 1 – output
                             after motion)
            outtime           buffer output time

Output:     //
Return:     //error code
*****/
int Demo_Direct_MyMoveABS(ZMC_HANDLE handle,int iMoveLen,int iaxisNum, int* iaxislist, float*
fPoslist,int iout,int *outlist,int outtime)
{
    char cmdbuff[2048];
    char tempbuff[2048];
    char cmdbuffAck[20480];

    //if transferred address is blank, exit

    int ret=0;
    int i;
    //first to read remain linear buffer
    int iBuffLen = 0;
    ret = ZAux_Direct_GetRemain_LineBuffer(handle,iaxislist[0],&iBuffLen);
    if(iBuffLen <= iMoveLen*2)
    {
        return 1002;          //motion buffer is not enough
    }

    //generate the command
    sprintf(cmdbuff, "BASE(");
    //splice motion axis list
    for (i=0;i<iaxisNum-1;i++)
    {
        sprintf(tempbuff,"%d,",iaxislist[i]);//generate the character string that corresponds to the
        command

        strcat(cmdbuff, tempbuff);//splice the character string

        if (strlen(cmdbuff)>1000)
        {
            return ERR_AUX_PARAERR; //wrong parameter, too long character string splicing
        }
    }
    sprintf(tempbuff,"%d)\n",iaxislist[i]);//generate the character string that corresponds to the
```

```
command
    strcat(cmdbuff,tempbuff);

    //splice the motion
    for (i=0;i<iMoveLen;i++)
    {
        //printf("%d,%d\n",i*iaxisNum,i*iaxisNum+1);
        if (outlist[i]==0) //this motion doesn't output
        {
            strcat(cmdbuff, "MoveAbs(");
            sprintf(tempbuff,"%f,%f)\n",fPoslist[i*iaxisNum],fPoslist[i*iaxisNum+1]); //generate
the character string that corresponds to the command
            strcat(cmdbuff, tempbuff); //splice the character string

        }
        else if (outlist[i]==1)
        {
            strcat(cmdbuff, "MoveAbs(");

            sprintf(tempbuff,"%f,%f)\n",fPoslist[i*iaxisNum],fPoslist[i*iaxisNum+1]); //generate the
character string that corresponds to the command
            strcat(cmdbuff, tempbuff); //splice the character string

            strcat(cmdbuff, "Move_op2(");
            sprintf(tempbuff,"%d,%d,%d)\n",iout,1,outtime); //generate the character string
that corresponds to the command
            strcat(cmdbuff, tempbuff); //splice the character string

        }
        else
        {
            return ERR_AUX_PARAERR; //parameter is wrong
        }
    }

    printf("spliced string: \n");
    printf("%s\n",cmdbuff);

    if (strlen(cmdbuff)>1000)
    {
        return ERR_AUX_PARAERR; //wrong parameter, too long character string splicing
    }
    ret=ZAux_DirectCommand(handle,cmdbuff,cmdbuffAck,2048);

    return ret;
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"127.0.0.1"; //Controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("success to connect controller!\n");
```

```
ret = ZAux_Direct_SetAtype(handle,0,1); //set axis type of axis 0 as 1
commandCheckHandler("ZAux_Direct_SetAtype", ret) ; //command is executed successfully?
ret = ZAux_Direct_SetAtype(handle,1,1); // set axis type of axis 1 as 1
commandCheckHandler("ZAux_Direct_SetAtype", ret) ; // command is executed successfully?

ret = ZAux_Direct_SetUnits(handle,0,100); //set pulse amount (UNITS) of axis 0 as 100
commandCheckHandler("ZAux_Direct_SetUnits", ret) ; // command is executed successfully?
ret = ZAux_Direct_SetUnits(handle,1,100); // set pulse amount (UNITS) of axis 1 as 100
commandCheckHandler("ZAux_Direct_SetUnits", ret) ; // command is executed successfully?

ret = ZAux_Direct_SetAccel(handle,0,500); //set axis 0 acceleration
commandCheckHandler("ZAux_Direct_SetAccel", ret) ; // command is executed successfully?
ret = ZAux_Direct_SetAccel(handle,1,500); //set axis 1 acceleration
commandCheckHandler("ZAux_Direct_SetAccel", ret) ; // command is executed successfully?

ret = ZAux_Direct_SetDecel(handle,0,500); //set axis 0 deceleration
commandCheckHandler("ZAux_Direct_SetDecel", ret) ; //command is executed successfully?
ret = ZAux_Direct_SetDecel(handle,1,500); //set axis 1 deceleration
commandCheckHandler("ZAux_Direct_SetDecel", ret) ; //command is executed successfully?

ret = ZAux_Direct_SetDpos(handle,0,0); //set DPOS clearing (0) of axis 0
commandCheckHandler("ZAux_Direct_SetDpos", ret) ; //command is executed successfully?
ret = ZAux_Direct_SetDpos(handle,1,0); //set DPOS clearing (0) of axis 1
commandCheckHandler("ZAux_Direct_SetDpos", ret) ; //command is executed successfully?

ret = ZAux_Direct_SetSpeed(handle,0,100); //set axis 0 speed
commandCheckHandler("ZAux_Direct_SetDecel", ret) ; //command is executed successfully?

ret = ZAux_Direct_SetSpeed(handle,1,100); // set axis 1 speed
commandCheckHandler("ZAux_Direct_SetDecel", ret) ; //command is executed successfully?

ret = ZAux_Direct_SetMerge(handle,0,1); //set opening continuous interpolation (only main axis
is opened, such as, axis 0 and axis 1 interpolation, axis 0 is the master axis, the main axis No. is the
first axis No. in continuous interpolation motion instructions)

int axis[2]={0,1};
float POS[12]={0,0,0,100,100,100,100,0,0,0};
int otlist[5]={0,1,1,1,1};

ZAux_Trigger(handle); //trigger the oscilloscope

ret = Demo_Direct_MyMoveABS(handle,5,2,axis,POS,0,otlist,50); //
commandCheckHandler("Demo_Direct_MyMoveABS", ret) ; // whether the command is
executed successfully?.

Sleep(20000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ; // whether the command is executed successfully?
printf("connection closed!\n");

handle = NULL;
return 0;
}
```

Chapter III Instruction List

Chapter IV Controller Connection	
4.1 Connect to Controller	
ZAux_OpenCom	Connect to controller by serial port
ZAux_SetComDefaultBaud	Set serial port communication parameters
ZAux_OpenEth	Connect to controller by Ethernet
ZAux_OpenPci	Connect to PCI card
ZAux_Close	Close controller connection
ZAux_SetIp	Set controller IP
ZAux_SearchEthlist	Search all controllers IP of current segment
ZAux_SearchAndOpenCom	Quick to build connection with controller
ZAux_SearchEth	Quick to search IP list
ZAux_GetMaxPciCards	Read PCI cards numbers
ZAux_FastOpen	Connect to MotionRT7
ZAux_GetLocalRTState	Get RT state
ZAux_LocalRTManage	Start and stop RT
4.2 Get Controller Information	
ZAux_GetControllerInfo	Get controller / card information
ZAux_GetSysSpecification	Get controller max specification
ZAux_GetRtcTime	Read controller RTC time
ZAux_SetRtcTime	Set controller RTC time

Chapter V Motion Control Parameters Configuration	
5.1 Motion Control Basic Parameters	
ZAux_Direct_SetAtype	Set axis type
ZAux_Direct_GetAtype	Read axis type
ZAux_Direct_SetUnits	Set pulse amount (units)
ZAux_Direct_GetUnits	Read pulse amount (units)
ZAux_Direct_SetAccel	Set acceleration, unit: units/s/s
ZAux_Direct_GetAccel	Read acceleration, unit: units/s/s
ZAux_Direct_SetDecel	Set deceleration, unit: units/s/s
ZAux_Direct_GetDecel	Read deceleration, unit: units/s/s
ZAux_Direct_SetSpeed	Set speed, unit: units/s/s
ZAux_Direct_GetSpeed	Read speed, unit: units/s/s
ZAux_Direct_SetDpos	Set axis demand position, unit: units
ZAux_Direct_GetDpos	Read axis demand position, unit: units
ZAux_Direct_SetMpos	Set axis measurement position, unit: units
ZAux_Direct_GetMpos	Read axis measurement position, unit: units

ZAux_Direct_GetEncoder	Get controller received pulses
ZAux_Direct_SetFastDec	Set fast deceleration, unit: units/s/s
ZAux_Direct_GetFastDec	Read fast deceleration, unit: units/s/s
ZAux_Direct_SetLspeed	Set axis initial speed, unit: units/s
ZAux_Direct_GetLspeed	Read axis initial speed, unit: units/s
ZAux_Direct_SetSrap	Set S curve, 0: T-shaped ACCEL & DECL
ZAux_Direct_GetSrap	Read S curve, 0: T-shaped ACCEL & DECL
ZAux_Direct_GetMspeed	Read measurement speed, unit: units/s
ZAux_Direct_GetVpSpeed	Read command speed: unit: units/s
ZAux_Direct_SetVpMode	Set the ACCEL & DECEL curve
ZAux_Direct_GetIdle	Read whether the axis ends
ZAux_Direct_GetAxisStatus	Read axis alarm state
ZAux_Direct_GetAxisStatus2	Read axis enable state
ZAux_Direct_GetAxisStopReason	Read axis history abnormal stop reason
5.2 Axis Other Parameters	
ZAux_GetModbusDpos	Quick to read multiple DPOS of current axis (MODBUS)
ZAux_GetModbusMpos	Quick to read multiple MPOS of current axis (MODBUS)
ZAux_GetModbusCurSpeed	Quick to read several current axis speed while moving (MODBUS)
ZAux_Direct_SetAxisAddress	Set axis address
ZAux_Direct_GetAxisAddress	Read axis address
ZAux_Direct_SetInvertStep	Set pulse output mode
ZAux_Direct_GetInvertStep	Read pulse output mode
ZAux_Direct_SetMaxSpeed	Set pulse output max frequency
ZAux_Direct_GetMaxSpeed	Read pulse output max frequency
ZAux_Direct_GetMtype	Read current motion's command
ZAux_Direct_GetNtype	Read the motion command of next motion
ZAux_Direct_SetOffpos	Set offset (relative position)
ZAux_Direct_GetOffpos	Read offset (relative position)
ZAux_Direct_Defpos	Define one new absolute offset position (not recommend)
ZAux_Direct_GetFe	Read following error
ZAux_Direct_SetFeLimit	Set max allowable following error (reserved)
ZAux_Direct_GetFeLimit	Read max allowed following error (reserved)
ZAux_Direct_SetFeRange	Set the following error when alarm (reserved)
ZAux_Direct_GetFeRange	Read following error when alarm (reserved)
ZAux_Direct_SetAxisMode	Set hold parameters for "connect" motion
5.3 Axis Parameters Commands	
ZAux_Direct_GetAllAxisPara	Read all axes parameters' state (in floating)
ZAux_Direct_GetAllAxisInfo	Read all axes parameters' state (in integer)
ZAux_Direct_SetParam	Set general axis parameters

ZAux_Direct_GetParam	Read general axis parameters

Chapter VI Basic Motion Control	
6.1 Motion Buffer	
ZAux_Direct_GetEndMove	Read axis motion's end position
ZAux_Direct_GetEndMoveBuffer	Read the end position of current motion and buffer motion
ZAux_Direct_GetLoaded	Read whether there is buffer except the current motion
ZAux_Direct_GetMovesBuffered	Read how many motions buffered
ZAux_Direct_GetMoveCurmark	Read current motion's mark No. "MOVE_MARK"
ZAux_Direct_SetMoveMark	Set motion's mark No. "MOVE_MARK"
ZAux_Direct_GetRemain	Read remaining distance of current motion
ZAux_Direct_GetRemain_LineBuffer	Read axis remaining linear buffer, calculate according to linear segments.
ZAux_Direct_GetRemain_Buffer	Read axis remain linear buffer, calculate according to the complex space arc.
ZAux_Direct_GetVectorBuffered	Read uncompleted distance of current axis and buffer motion
ZAux_Direct_MoveOp	Add "OP" command in motion buffer
ZAux_Direct_MoveOpAdist	Set distance (early to output) in buffer
ZAux_Direct_MoveOpMulti	Add "continuous output" command in motion buffer
ZAux_Direct_MoveOpDelay	Early / delay to open OP port
ZAux_Direct_MoveOp2	Add "OP" command in motion buffer, and make it invert the state after specified time
ZAux_Direct_MoveAout	Add "AOUT" command in motion buffer
ZAux_Direct_MoveDelay	Add "delay" command in motion buffer
ZAux_Direct_MoveWait	Add "one changeable delay" in BASE axis / axes motion buffer
ZAux_Direct_MoveTask	Add "one TASK" in BASE axis / axes motion buffer
ZAux_Direct_MovePara	Modify parameters while moving
ZAux_Direct_MovePwm	Modify PWM while moving
ZAux_Direct_MoveSynmove	Synchronize with other axes motions while moving
ZAux_Direct_MoveASynmove	Trigger other axes motions while moving
ZAux_Direct_MoveTable	Modify TABLE while moving
ZAux_Direct_MoveCancel	Cancel other axes motions while moving
ZAux_Direct_MoveModify	Modify the end position while moving
ZAux_Direct_MoveLimit	Add speed limit at the motion end, used to do forcible corner deceleration.

ZAux_Direct_MoveModify2	Modify the end position while moving 2
ZAux_Execute_GetMoveTime	Read corresponding motion remain time
ZAux_Execute_GetPreCalTime	Estimate motion time of the point, in ms
6.2 Point Motion	
ZAux_Direct_Single_Move	Single-axis relative motion
ZAux_Direct_Single_MoveAbs	Single-axis absolute motion
ZAux_Direct_Single_Vmove	Single-axis continuous motion
ZAux_Direct_SetJogSpeed	Set speed when JOG
ZAux_Direct_GetJogSpeed	Read speed when JOG
ZAux_Direct_GetFastJog	Read JOG speed IN: when there is IN, speed is given by SPEED; when there is no IN, speed is given by JOGSPEED
ZAux_Direct_SetFastJog	Set JOG speed IN: when there is IN, speed is given by SPEED; when there is no IN, speed is given by JOGSPEED
ZAux_Direct_SetFwdJog	Set forward JOG IN No., -1 = invalid
ZAux_Direct_GetFwdJog	Read forward JOG IN No., -1 = invalid
ZAux_Direct_SetRevJog	Set inverse JOG IN No., -1 = invalid
ZAux_Direct_GetRevJog	Read inverse JOG IN No., -1 = invalid
6.3 Homing	
ZAux_Direct_Single_Datum	Homing in controller way
ZAux_BusCmd_Datum	Homing in bus drive way
ZAux_Direct_UserDatum	Customized twice homing
ZAux_Direct_SetHomeWait	Set homing inverse-find waiting time
ZAux_Direct_GetHomeWait	Read homing inverse-find waiting time
ZAux_BusCmd_SetDatumOffpos	Set homing offset distance
ZAux_BusCmd_GetDatumOffpos	Read homing offset distance
ZAux_Direct_SetCreep	Set homing creep speed
ZAux_Direct_GetCreep	Read homing creep speed
ZAux_Direct_GetHomeStatus	Read homing completion state
ZAux_Direct_GetHomeStatus2	Read homing completion state 2
ZAux_BusCmd_GetHomeStatus	Read drive homing completion state
ZAux_BusCmd_GetHomeStatus2	Read drive homing completion state 2
ZAux_Direct_SetDatumIn	Set homing origin signal
ZAux_Direct_GetDatumIn	Read homing origin signal
6.4 Electronic Gear	
ZAux_Direct_SetClutchRate	Set connection velocity
ZAux_Direct_GetClutchRate	Read connection velocity
ZAux_Direct_EncoderRatio	Set encoder input gear ratio, default (1,1)
ZAux_Direct_StepRatio	Set pulse output gear ratio, default (1,1)
ZAux_Direct_Connect	"connection" & "synchronization" command for electronic gear
ZAux_Direct_Connpath	"connection" & "synchronization" command 2

	for electronic gear
ZAux_Direct_GetLinkax	Read current connected motion's reference axis No.
6.5 Continuous Interpolation	
ZAux_Direct_Move	Multi-axis relative linear interpolation
ZAux_Direct_MoveSp	Multi-axis relative linear interpolation, SP
ZAux_Direct_MoveAbs	Multi-axis absolute linear interpolation
ZAux_Direct_MoveAbsSp	Multi-axis absolute linear interpolation, SP
ZAux_Direct_MoveCirc	Relative circular interpolation, use circle center to make one circle
ZAux_Direct_MoveCircSp	Relative circular interpolation, use circle center to make one circle, SP
ZAux_Direct_MoveCircAbs	Absolute circular interpolation, use circle center
ZAux_Direct_MoveCircAbsSp	Absolute circular interpolation, use circle center, SP
ZAux_Direct_MoveCirc2	Relative circular interpolation by 3-point
ZAux_Direct_MoveCirc2Abs	Absolute circular interpolation by 3-point
ZAux_Direct_MoveCirc2Sp	Relative circular interpolation by 3-point, SP
ZAux_Direct_MoveCirc2AbsSp	Absolute circular interpolation by 3-point, SP
ZAux_Direct_MHelical	Relative 3-axis helical interpolation by circle center
ZAux_Direct_MHelicalAbs	Absolute 3-axis helical interpolation by circle center
ZAux_Direct_MHelicalSp	Relative 3-axis helical interpolation by circle center, SP
ZAux_Direct_MHelicalAbsSp	Absolute 3-axis helical interpolation by circle center, SP
ZAux_Direct_MHelical2	Relative 3-axis helical interpolation by 3-point
ZAux_Direct_MHelical2Abs	Absolute 3-axis helical interpolation by 3-point
ZAux_Direct_MHelical2Sp	Relative 3-axis helical interpolation by 3-point, SP
ZAux_Direct_MHelical2AbsSp	Absolute 3-axis helical interpolation by 3-point, SP
ZAux_Direct_MEclipse	Relative ellipse interpolation
ZAux_Direct_MEclipseAbs	Absolute ellipse interpolation
ZAux_Direct_MEclipseSp	Relative ellipse interpolation, SP
ZAux_Direct_MEclipseAbsSp	Absolute ellipse interpolation, SP
ZAux_Direct_MEclipseHelical	Relative ellipse + helical interpolation
ZAux_Direct_MEclipseHelicalAbs	Absolute ellipse + helical interpolation
ZAux_Direct_MEclipseHelicalSp	Relative ellipse + helical interpolation, SP
ZAux_Direct_MEclipseHelicalAbsSp	Absolute ellipse + helical interpolation, SP

ZAux_Direct_MSpherical	Space arc + helical interpolation
ZAux_Direct_MSphericalSp	Space arc + helical interpolation, SP
ZAux_Direct_MSphericalABSSp	Absolute space arc + helical interpolation, SP
ZAux_Direct_MoveSpiral	Relative involute circular interpolation
ZAux_Direct_MoveSpiralSp	Relative involute circular interpolation, SP
ZAux_Direct_MoveSmooth	Space linear interpolation
ZAux_Direct_MoveSmoothSp	Space linear interpolation, SP
ZAux_Direct_McircTurnabs	Turntable linear interpolation
ZAux_Direct_MoveTurnabs	Turntable circular + helical interpolation
ZAux_Direct_MultiMove	Several multi-axis relative interpolation
ZAux_Direct_MultiMoveAbs	Several multi-axis absolute interpolation
ZAux_Direct_MultiLineN	Multi-axis multi-segment linear continuous interpolation
ZAux_Direct_SetFhspeed	Set axis HOLD speed
ZAux_Direct_GetFhspeed	Read axis HOLD speed
ZAux_Direct_SetEndMoveSpeed	Set SP motion's end speed
ZAux_Direct_GetEndMoveSpeed	Read SP motion's end speed
ZAux_Direct_SetForceSpeed	Set SP motion's running speed
ZAux_Direct_GetForceSpeed	Read SP motion's running speed
ZAux_Direct_SetStartMoveSpeed	Set SP motion's starting speed
ZAux_Direct_GetStartMoveSpeed	Read SP motion's starting speed
ZAux_Direct_SetFholdIn	Set HOLD IN No., -1 = invalid
ZAux_Direct_GetFholdIn	Read HOLD IN No., -1 = invalid
ZAux_Direct_SetCornerMode	Set corner deceleration mode
ZAux_Direct_GetCornerMode	Read corner deceleration mode
ZAux_Direct_SetDecelAngle	Set corner start deceleration angle, radian
ZAux_Direct_GetDecelAngle	Read corner start deceleration angle, radian
ZAux_Direct_SetFullSpRadius	Set the minimal radius of small-circle speed limit
ZAux_Direct_GetFullSpRadius	Read the minimal radius of small-circle speed limit
ZAux_Direct_SetInterpFactor	Set whether the axis joins in speed calculation when interpolation
ZAux_Direct_GetInterpFactor	Read whether the axis joins in speed calculation when interpolation
ZAux_Direct_SetMerge	Set continuous interpolation state
ZAux_Direct_GetMerge	Read continuous interpolation state
ZAux_Direct_SetStopAngle	Set the minimum turning angle to reduce deceleration to the lowest level, in radian
ZAux_Direct_GetStopAngle	Read the minimum turning angle to reduce deceleration to the lowest level, in radian
ZAux_Direct_SetZsmooth	Set deceleration chamfering radius
ZAux_Direct_GetZsmooth	Read deceleration chamfering radius

6.6 Axis Pause / Resume / Stop	
ZAux_Direct_MovePause	Pause the motion, for interpolation, it pauses the spindle
ZAux_Direct_MoveResume	Cancel motion paused
ZAux_Direct_Rapidstop	Stop all axes immediately
ZAux_Direct_CancelAxisList	Stop multi-axis
ZAux_Direct_Single_Cancel	Stop single-axis

Chapter VII Advanced Motion Control	
7.1 Coordinated Loop	
ZAux_Direct_SetRepDist	Set loop range
ZAux_Direct_GetRepDist	Read loop range
ZAux_Direct_SetRepOption	Set coordinates loop mode
ZAux_Direct_GetRepOption	Read coordinates loop mode
7.2 Pitch Compensation	
ZAux_Direct_Pitchset	Set axis pitch compensation, it is invalid for extended axes
ZAux_Direct_Pitchset2	Set axis pitch dual-compensation, it is invalid for extended axes
ZAux_Direct_GetPitchStatus	Read axis pitch compensation state, it is invalid for extended axes
7.3 Backlash	
ZAux_Direct_Backlash	Set axis backlash compensation, it is invalid for extended axes

Chapter VIII Hardware Interface Configuration & Reading	
8.1 Digital IN & OP	
ZAux_Direct_GetIn	Read single IN signal
ZAux_Direct_SetOp	Set single IN signal
ZAux_Direct_GetOp	Read single OP state
ZAux_GetModbusIn	Quick to read multi-IN (Modbus)
ZAux_GetModbusOut	Quick to read multi-OP state (Modbus)
ZAux_Direct_GetInMulti	Read multiple IN signals
ZAux_Direct_SetOutMulti	Set multiple OPs to output
ZAux_Direct_GetOutMulti	Read multiple outputs states
ZAux_Direct_SetInvertIn	Set "invert" for valid input
ZAux_Direct_GetInvertIn	Read IN invert state
8.2 AD & DA	
ZAux_Direct_GetAD	Read single AIN
ZAux_Direct_SetDA	Open single AOUT
ZAux_Direct_GetDA	Read AOUT state
8.3 PWM Control	
ZAux_Direct_SetPwmFreq	Set PWM frequency

ZAux_Direct_GetPwmFreq	Read PWM frequency
ZAux_Direct_SetPwmDuty	Set PWM duty cycle
ZAux_Direct_GetPwmDuty	Read PWM duty cycle

Chapter IX Safe Configuration	
9.1 Axis Soft Position Limit	
ZAux_Direct_SetFsLimit	Set forward soft position limit
ZAux_Direct_GetFsLimit	Read forward soft position limit
ZAux_Direct_SetRsLimit	Set inverse soft position limit
ZAux_Direct_GetRsLimit	Read inverse soft position limit
9.2 Axis Hard Position Limit	
ZAux_Direct_SetFwdIn	Set forward hard position limit IN No.
ZAux_Direct_GetFwdIn	Read forward hard position limit IN No.
ZAux_Direct_SetRevIn	Set inverse hard position limit IN No.
ZAux_Direct_GetRevIn	Read inverse hard position limit IN No.
9.3 Axis Alarm	
ZAux_Direct_SetAlmIn	Set axis alarm IN No.
ZAux_Direct_GetAlmIn	Read axis alarm IN No.

Chapter X Data Interaction	
10.1 VR Register	
ZAux_Direct_SetVrf	Set VR register
ZAux_Direct_GetVrf	Read VR register
ZAux_Direct_GetVrInt	Read VR register in integer
10.2 Table Register	
ZAux_Direct_SetTable	Set system table register
ZAux_Direct_GetTable	Read system table register
ZAux_Direct_SetTableDouble	Write data into Table
ZAux_Direct_GetTableDouble	Read Table, it can read several at one time
10.3 Modbus Register	
ZAux_Modbus_Set0x	Set MODBUS bit register
ZAux_Modbus_Get0x	Read MODBUS bit register
ZAux_Modbus_Set4x	Set MODBUS register (REG)
ZAux_Modbus_Get4x	Read MODBUS register (REG)
ZAux_Modbus_Set4x_Long	Set MODBUS4x register (LONG)
ZAux_Modbus_Get4x_Long	Read MODBUS4x register (LONG)
ZAux_Modbus_Set4x_String	Set MODBUS word register (ASCII)
ZAux_Modbus_Get4x_String	Read MODBUS word register (ASCII)
ZAux_Modbus_Set4x_Float	Set MODBUS register (float)

ZAux_Modbus_Get4x_Float	Read MODBUS register (float)
10.4 Flash/File Reading & Writing	
ZAux_WriteUFile	Write file into U disk
ZAux_ReadUFile	Read file from U disk
ZAux_FlashWritef	Write into controller FLASH
ZAux_FlashReadf	Read from controller FLASH
ZAux_DownMem3FileRam	Send packed character string
ZAux_Run3FileRam	Run packed File 3
ZAux_Down3FileRam	Download file 3 to RAM
ZAux_Down3FileRom	Download file 3 to ROM
ZAux_3FileRamGetRemainSpace	Dynamically download file 3, read remain space
ZAux_3FileRamDownBegin	Dynamically download file 3, open downloading
ZAux_3FileRamDownPart	Dynamically download file 3
ZAux_3FileRamDownEnd	Dynamically download file 3, end downloading
ZAux_Check3File	Check whether file 3 exists
ZAux_Delete3File	Delete controller files
ZAux_FindFirst3File	Find controller files, if the filename is empty, which means there is no file
ZAux_FindNext3File	Find controller next files, if the filename is empty, which means there is no file
ZAux_GetCur3File	Obtain current file of controller
ZAux_RemoveAll3Files	Delete controller's all file 3
ZAux_Up3File	Upload file 3
ZAux_Up3FileToMem	Upload file 3

Chapter XI Bus Initialization	
11.1 Bus Initialization	
ZAux_BusCmd_GetNodeNum	Read node numbers on card slot
ZAux_BusCmd_GetNodeInfo	Read node information
ZAux_BusCmd_GetNodeStatus	Read node bus state
ZAux_BusCmd_DriveClear	Clear bus servo alarm
ZAux_Direct_SetErrormask	Set error mark
ZAux_Direct_GetErrormask	Read error mark
ZAux_Direct_SetAxisEnable	Set axis enable, it is only valid inn bus type controllers
ZAux_Direct_GetAxisEnable	Read axis enable, it is only valid inn bus type controllers
ZAux_Direct_GetBusAxisEnable	Read bus axis enable state

11.2 SDO/PDO Reading & Writing	
ZAux_BusCmd_SDOWrite	Write SDO by device No. and slot No.
ZAux_BusCmd_SDORead	Read SDO by device No. and slot No.
ZAux_BusCmd_SDOWrite2	Write node SDO parameters
ZAux_BusCmd_SDORead2	Read node SDO parameters
ZAux_BusCmd_RtexRead	Read RTEX parameters
ZAux_BusCmd_RtexWrite	Set RTEX parameters
ZAux_BusCmd_SDOReadAxis	Read axis SDO parameters
ZAux_BusCmd_SDOWriteAxis	Write axis SDO parameters
ZAux_BusCmd_NodePdoWrite	PDO writing
ZAux_BusCmd_NodePdoRead	PDO reading
ZAux_BusCmd_NodePdoWriteFloat	PDO writing
ZAux_BusCmd_NodePdoReadFloat	PDO reading
11.3 Bus Torque	
ZAux_BusCmd_GetDriveTorque	Read current bus drive's torque, it needs to set corresponding DRIVE_PROFFILE type.
ZAux_BusCmd_SetMaxDriveTorque	Set current bus drive's max torque, it needs to set corresponding DRIVE_PROFFILE type.
ZAux_BusCmd_GetMaxDriveTorque	Read current bus drive's max torque, it needs to set corresponding DRIVE_PROFFILE type.
ZAux_Direct_SetDAC	Set analog output, used under torque, speed mode, for bus drive, it needs to set corresponding DRIVE_PROFFILE type and axis type ATYPE.
ZAux_Direct_GetDAC	Read analog output, used under torque, speed mode, for bus drive, it needs to set corresponding DRIVE_PROFFILE type and axis type ATYPE.

Chapter XII Other Commands	
12.1 Online Commands	
ZAux_Execute	Send character string (RTBasic) in buffer way
ZAux_DirectCommand	Send character string (RTBasic) in direct way
12.2 Controller Basic Related Commands	
ZAux_Pause	Pause BASIC program that is running inside the controller
ZAux_Resume	Resume paused BASIC program that is running inside the controller
ZAux_BasDown	Generate single BAS file as ZAR, and

	download into controller for execution.
ZAux_ZarDown	Download ZAR file into controller
ZAux_ZpjDown	Download ZPJ file into controller
ZAux_Direct_GetVariablef	Read global variables, like, axis parameter, the returned value is floating type
ZAux_Direct_GetVariableInt	Read global variables, like, axis parameter, the returned value is integer type
ZAux_Direct_SetUserArray	Set BASIC customized global array
ZAux_Direct_GetUserArray	Read BASIC customized global array, it can read several at one time
ZAux_Direct_SetUserVar	Set customized global variables
ZAux_Direct_GetUserVar	Read customized global variables
ZAux_SetTimeOut	The command's delay waiting time
ZAux_TransStringtoFloat	Convert character string to float
ZAux_TransStringtoInt	Convert character string to int
12.3 Controller Auto-Report	
PZMCAutoUpCallBack	The callback function format of auto-report
ZAux_SetAutoUpCallBack	Controller auto-report
ZAux_CycleUpEnable	Enable period report
ZAux_CycleUpDisable	Cancel "period report enable"
ZAux_CycleUpGetRecvTimes	The number of packets received is periodically reported; if it exceeds, it is used for debugging.
ZAux_CycleUpForceOnce	Force a report once, in case the idle state after the motion command is inaccurate.
ZAux_CycleUpReadBuff	Read content from period report (floating)
ZAux_CycleUpReadBuffInt	Read content from period report (integer)
12.4 Debug	
ZAux_SetTraceFile	Set "command tracking"
ZAux_Trigger	Trigger oscilloscope
ZAux_Scope	Use SCOPE to capture data
ZAux_GetScopeLength	Get the SCOPE sampled length

Chapter III How to Connect & Initialize?

4.1. How to Connect to Controller?

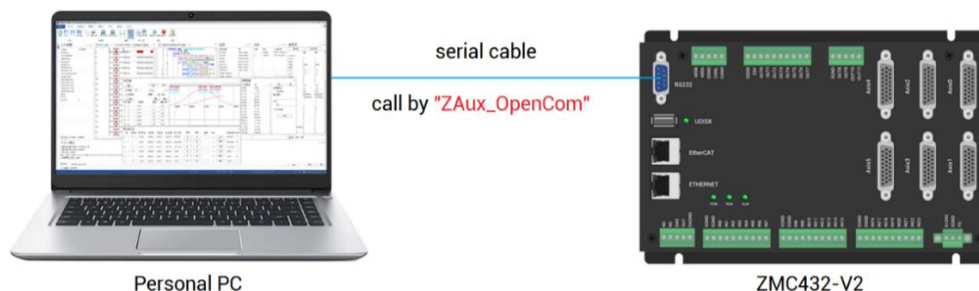
4.1.1. Instruction List

Commands	Descriptions
ZAux_OpenCom	Connect to controller by serial port
ZAux_SetComDefaultBaud	Set serial port communication parameters
ZAux_OpenEth	Connect to controller by Ethernet
ZAux_OpenPci	Connect to PCI card
ZAux_Close	Close controller connection
ZAux_SetIp	Set controller IP
ZAux_SearchEthlist	Search all controllers IP of current segment
ZAux_SearchAndOpenCom	Quick to build connection with controller
ZAux_SearchEth	Quick to search IP list
ZAux_GetMaxPciCards	Read PCI cards numbers
ZAux_FastOpen	Connect to MotionRT7
ZAux_GetLocalRTState	Get RT state
ZAux_LocalRTManage	Start and stop RT

4.1.2. Principle & Description

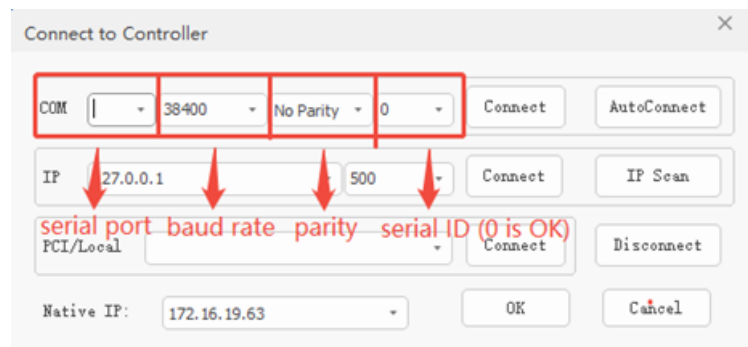
General Description

- Connect by **Serial Port**:



Controller Default Serial Port Parameters			
Boolean Rate	Data Bit	Stop Bit	Parity

38400	8	1	No parity
Note: PC serial port parameters should be same as controller parameters, then connection can be built.			



Station No. (serial ID): there may be several slave stations while using 485, you could select needed one by the No. usually fill in 0.

➤ **EtherNET:**

The network port adopts standard RJ45 cable interface, and the communication rate is 100Mbit/s. Zmotion motion controller comes with one Ethernet, and it supports 2 ethernet channels at least. If you need more, an interchanger can be used. Corresponding communication channel can be checked by "?*port".

More details of controller Ethernet:

- The factory IP of the controller is 192.168.0.11, which can be set by yourself. If you forget the IP, you should connect it by serial port at first, then check / set it.
- Ethernet is used for connecting to PC / HMI.
- The port No. is 502.
- While communicating / connecting with PC, they must be in the same network segment (the first three segments are the same, the last one is different).

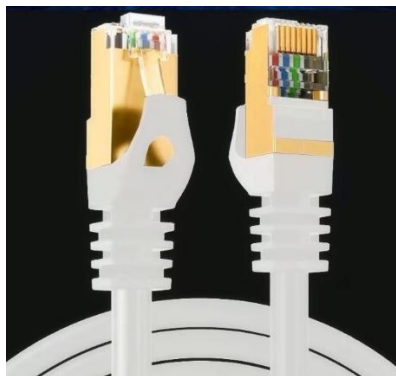
Ethernet connection ability and performance:

[illegible]

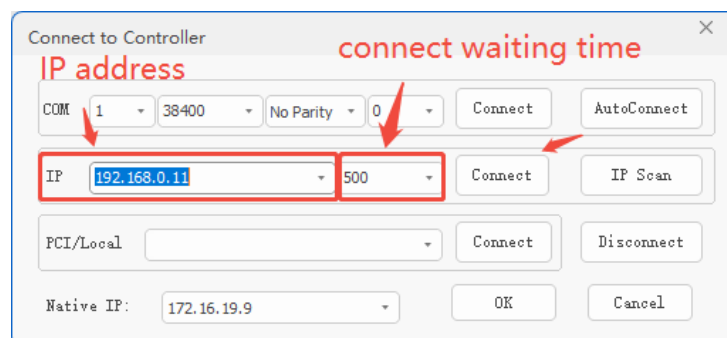
Cable recommendations:

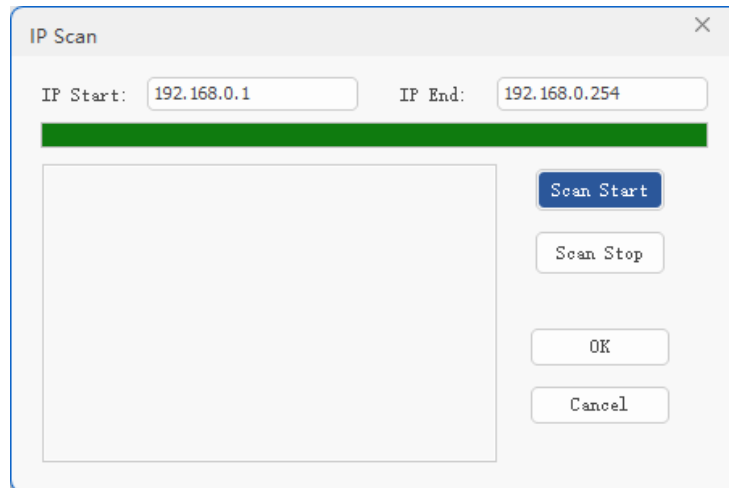
Ethernet Cable Type	Transfer Frequency	Transfer Velocity	Transfer Distance	Cable Type
CAT5	100Mhz	100Mbps	100m	Shielded/non-Shielded
CAT5E	100Mhz	1000Mbps	100m	Shielded/non-Shielded
CAT6	250Mhz	1Gbps	100m	Shielded/non-Shielded
CAT6E	500Mhz	10Gbps	100m	Shielded/non-Shielded
CAT7	600Mhz	10Gbps	100m	Pair Shielded
CAT8	2000Mhz	40Gbps	30m	Pair Shielded

In order to ensure the quality of communication, it is better to use a double-shielded network cable with a metal head above **Category 7** in occasions with **large interference**. In **normal occasions**, use a network cable with a shielded metal head of **Category 5e**. As shown in the figure below: the network cable with metal wire can play the role of grounding protection. If a network cable (without a metal headband shielded) is used in an actual interference site, uncertain things such as packet loss will occur.



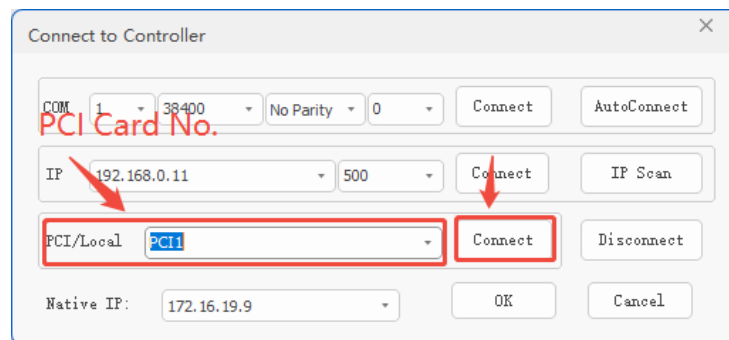
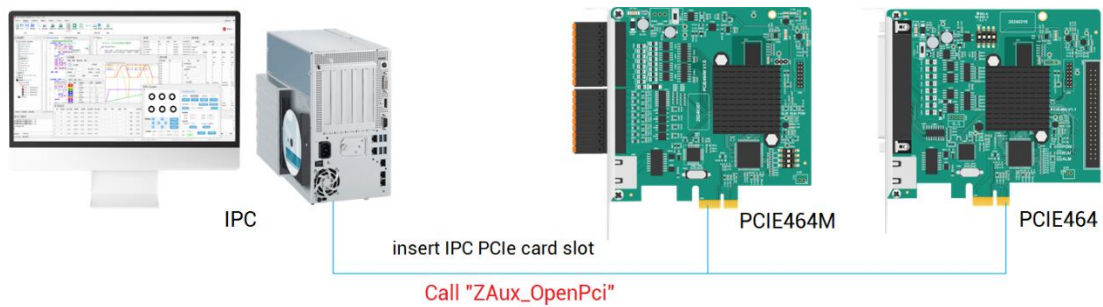
"Whether it is executed successfully" can be known by the return value, 0 means success, non-0 means failure. If fails, please check transferred IP, and make sure PC IP and controller IP are in same segment.



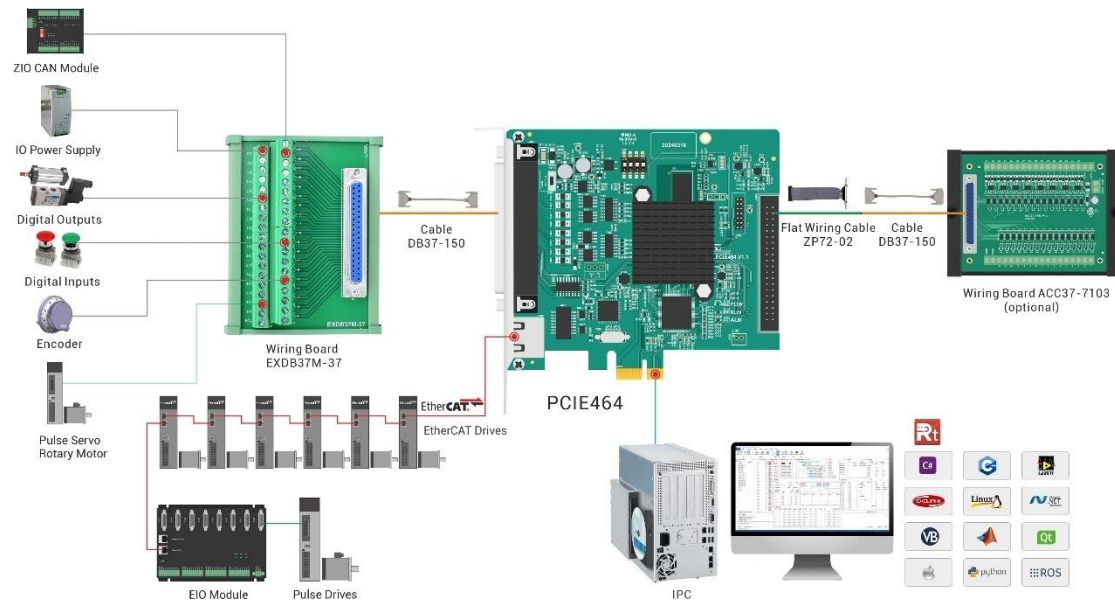


- Connection waiting time: it is valid for Ethernet connection. If the response time is short, you can select a larger value to extend the response time.
- IP Start, IP End: enter needed IP segments.

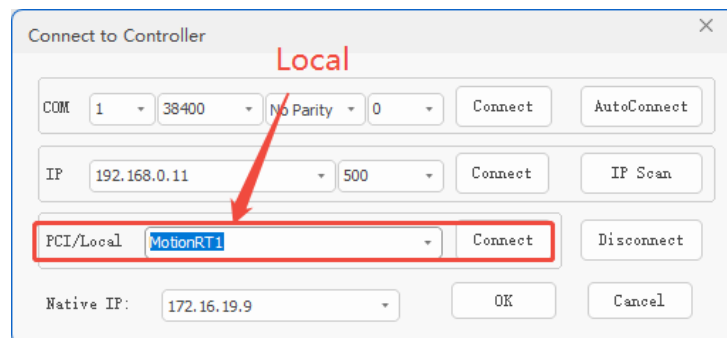
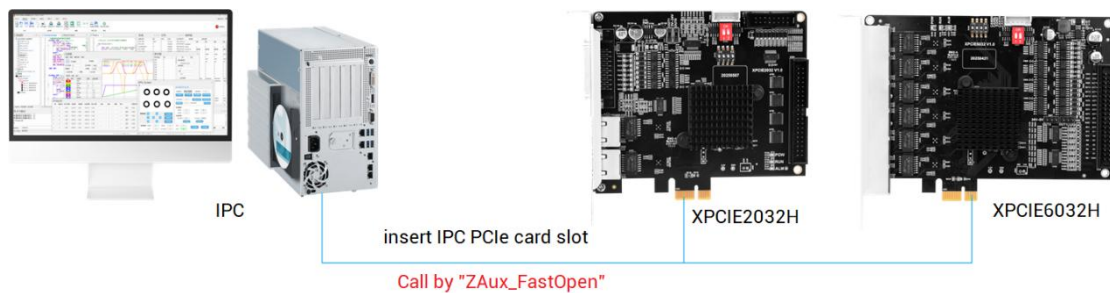
➤ **PCI:**



- PCI / Local: select the card slot No. of PCI card type (it can be scanned after installing drive), like, PCI, PCIe series cards, PCIe464, PCIe464M.



➤ **Local:**



Valid in VPLC7XX series, XPCle series, like, XPCle2032H. And it is used together with MotionRT7 real-time kernel.

MotonRT7

MotionRT7 is recommended to be used with XPCI/XPCIE control card, which can play a better performance, and the authorization information is stored in the control card. Each control card has a unique number on the card.

Prepare the latest version of the MotionRT7 installation package and decompress it before installation. The figure below shows the contents of the files after decompression of the MotionRT7 installation package.

此电脑 > 备份 (E:) > MotionRT7				
名称	修改日期	类型	大小	
xplcterm	2022/10/28 8:10	文件夹		
flash	2022/9/21 13:47	文件夹		
driver_signed	2022/10/31 16:22	文件夹		
zplib.dll	2022/9/28 10:57	应用程序扩展	11,257 KB	
zvapplib.dll	2022/8/22 20:15	应用程序扩展	432 KB	
zcam_zzioa.dll	2022/9/2 11:00	应用程序扩展	2,639 KB	
zcam_zmvcbase.dll	2022/9/2 11:00	应用程序扩展	2,876 KB	
zcam_zmotion.dll	2022/9/28 14:43	应用程序扩展	84 KB	
zcam_mvision.dll	2022/9/28 14:43	应用程序扩展	84 KB	
zcam_mindvision.dll	2022/9/28 14:43	应用程序扩展	57 KB	
zcam_mcamera.dll	2022/9/28 14:43	应用程序扩展	61 KB	
zcam_huaray.dll	2022/9/2 11:00	应用程序扩展	63 KB	
zcam_file.dll	2022/9/28 14:43	应用程序扩展	2,647 KB	
zcam_dvpcamera.dll	2022/9/28 14:43	应用程序扩展	59 KB	
zcam_daheng.dll	2022/9/28 14:43	应用程序扩展	50 KB	
zcam_basler.dll	2022/9/2 11:00	应用程序扩展	57 KB	
vcruntime140.dll	2018/11/20 9:38	应用程序扩展	84 KB	
vcomp140.dll	2018/11/20 9:38	应用程序扩展	151 KB	
ucrtbase.dll	2020/4/13 15:20	应用程序扩展	976 KB	
msvcr120.dll	2020/4/13 15:20	应用程序扩展	941 KB	
msvcp140.dll	2020/4/13 15:20	应用程序扩展	619 KB	
msvc120.dll	2020/4/13 15:20	应用程序扩展	645 KB	
mfc140.dll	2020/4/13 15:20	应用程序扩展	5,896 KB	
7z.dll	2019/2/22 0:00	应用程序扩展	1,640 KB	
MotionRT710.exe	2022/10/27 17:21	应用程序	514 KB	
7z.exe	2019/2/22 0:00	应用程序	458 KB	

- ✧ Driver_signed: the driver folder, which contains MotionRT7 driver installation information files, ECAT protocol installation information, installation wizard software, and there may include security directory files, sys system files, and signature files.

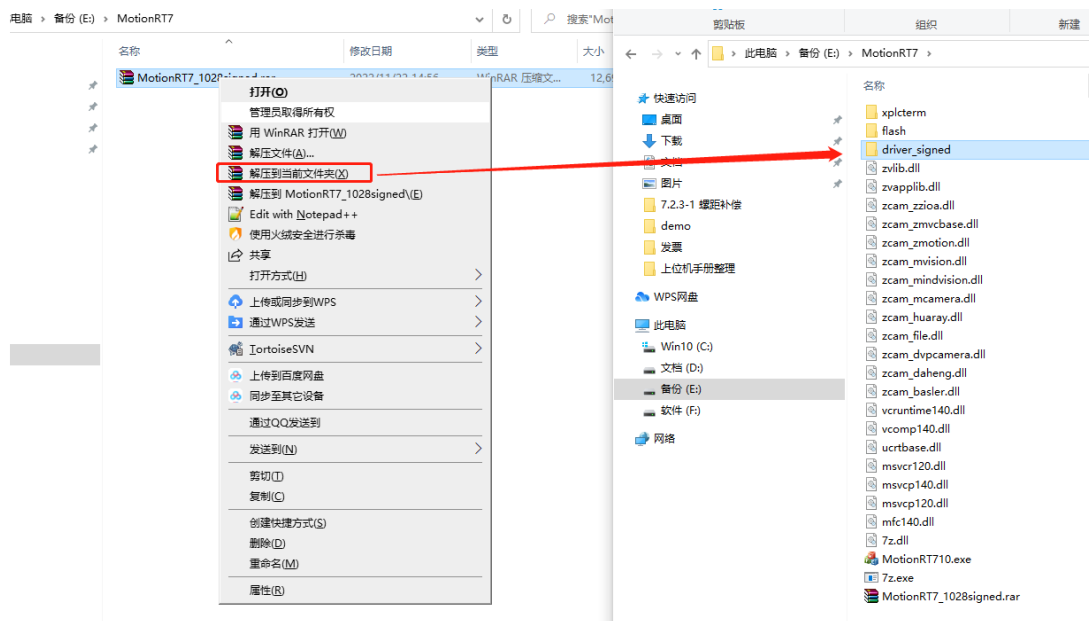
MotionRT7 > driver_signed				
名称	修改日期	类型	大小	
dpinst_amd64.exe	2022/9/6 11:21	应用程序	1,026 KB	
ZMotionRt64.cat	2022/10/30 23:47	安全目录	13 KB	
ZMotionRt64.inf	2022/10/18 19:06	安装信息	4 KB	
ZMotionRt64.sys	2022/10/30 23:47	系统文件	4,966 KB	
ZMotionRtPacket.inf	2022/10/18 19:06	安装信息	2 KB	

- Dpinst_amd64.exe: installation wizard software.
- ZMotionRt64.cat: digital signature file of the driver.
- ZMotionRt64.inf: MotionRT7 software driver installation information, browse the folder to select the file when installing the driver.
- ZMotionRt64.sys: system file.

- ZMotionRTPacket.inf: bus protocol installation information, install the EACT bus protocol, browse folder to select this file.
- ✧ flash: controller system folder.
- ✧ Udisk: U disk folder.
- ✧ Xplcterm: xplc screen folder, which contains xplcterm software, which is displayed as a screen when using HMI.

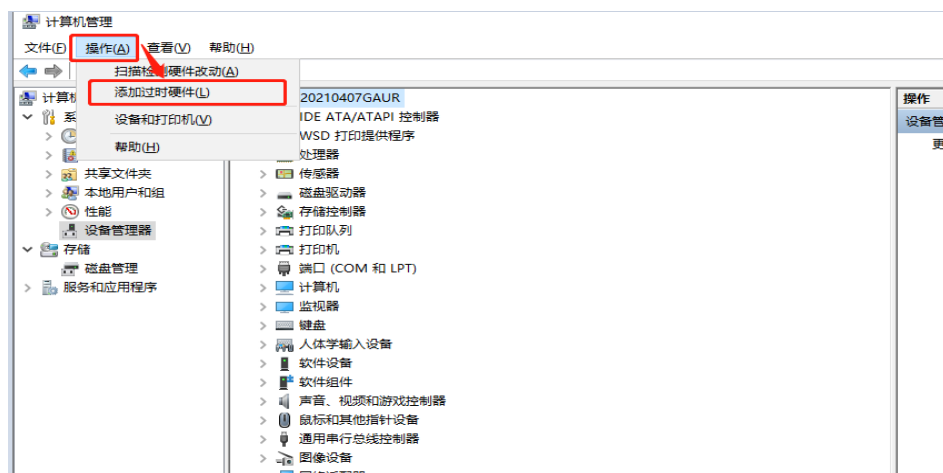
➤ MotionRT7 Drive Installation:

After obtaining the installation package of MotionRT7, unzip the installation package to the location of your choice, as shown below:



When there is no PCI card:

1. In the device manager, the menu: "Operation" - "Add Obsolete Hardware", if there is no "Operation", right click to add it.



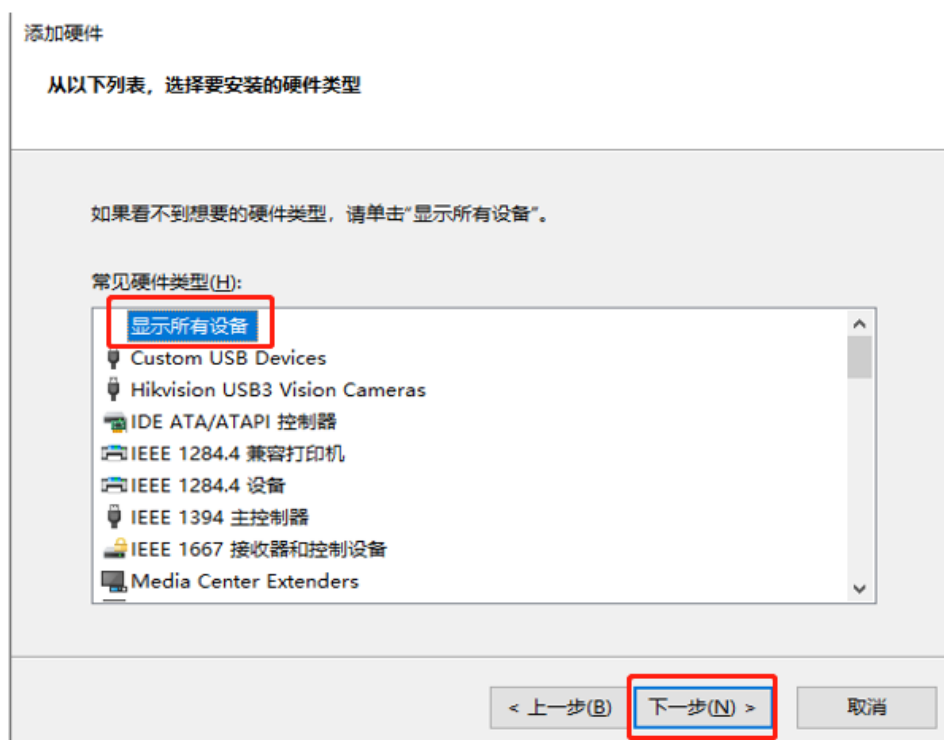
2. After that, below interface will pop up, click "next".



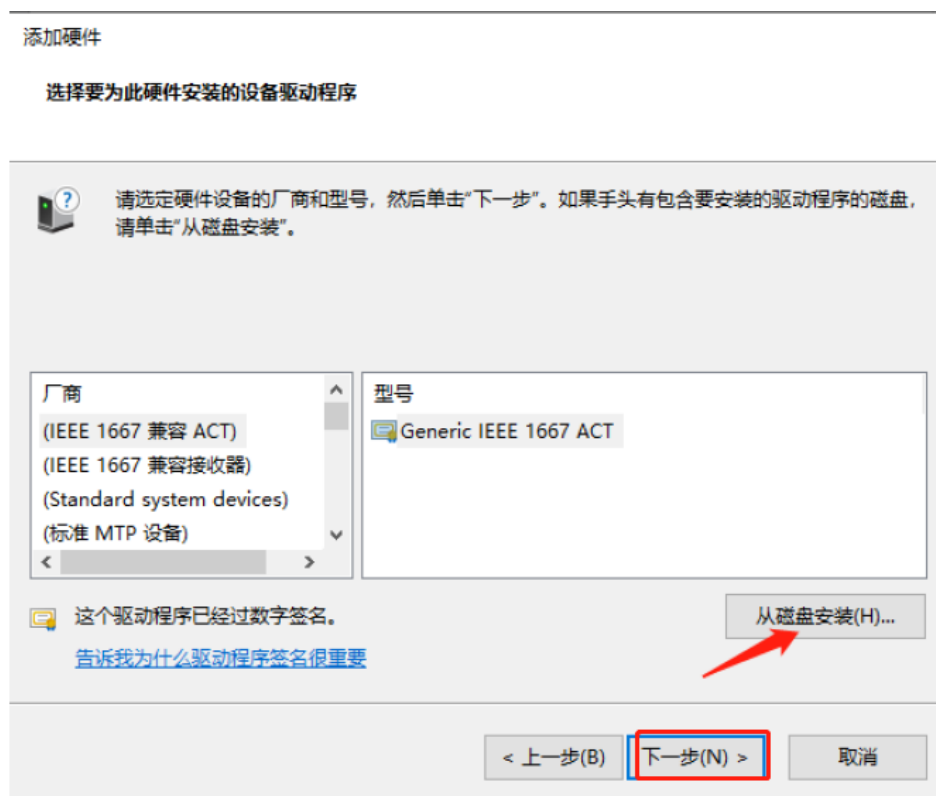
3. Then, in below window, select the second one (install the hardware that is selected from the list manually).



4. Select "display all devices", then click "next".

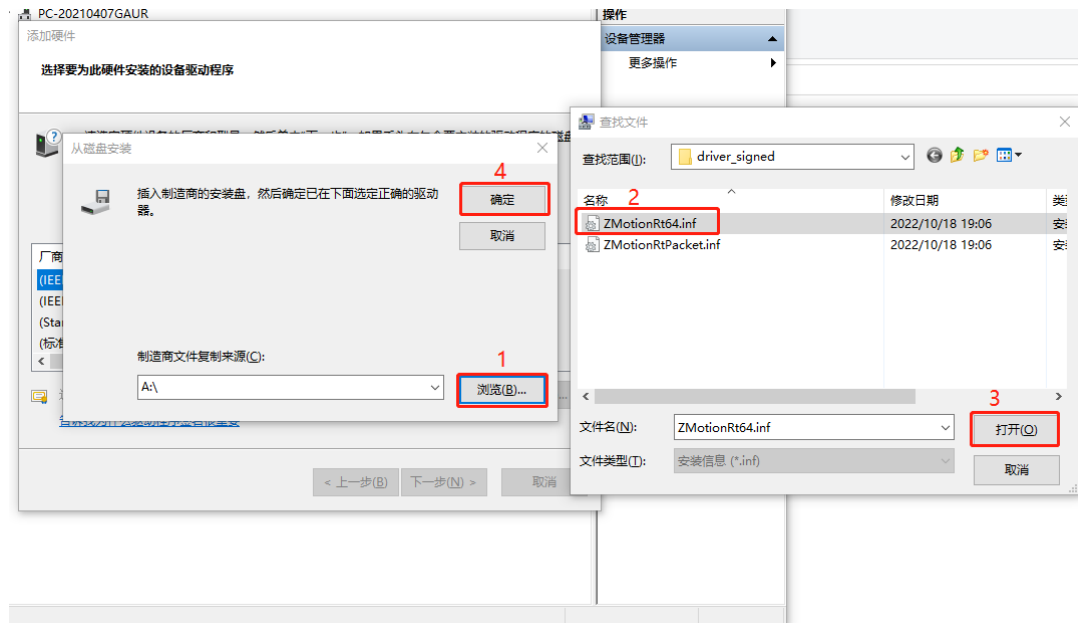


5. Choose to install from disk, (all options are default items, no need to choose manufacturer and model)

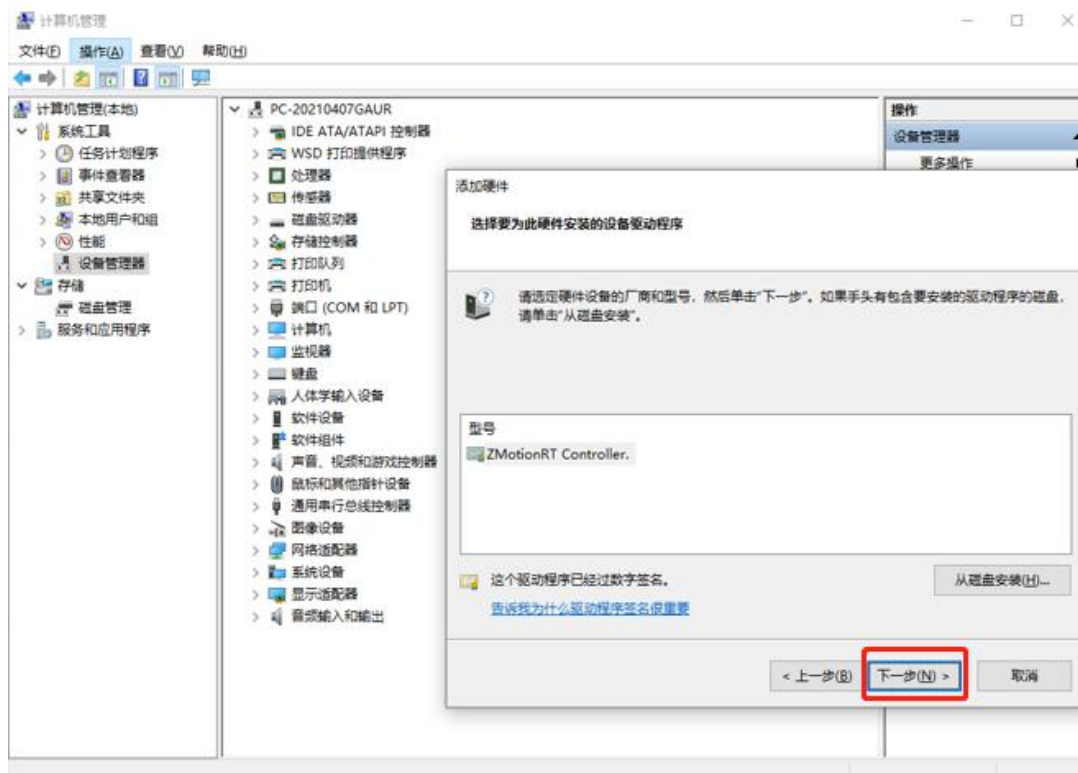


6. Then, below window appears:

- (1) Click "browse".
- (2) Open the file path MotionRT7\driver_signed\ZMotionRt64.inf under MotionRT7.
- (3) Open ZMotionRt64.inf file.
- (4) Click "OK".



7. After that, directly click "Next".



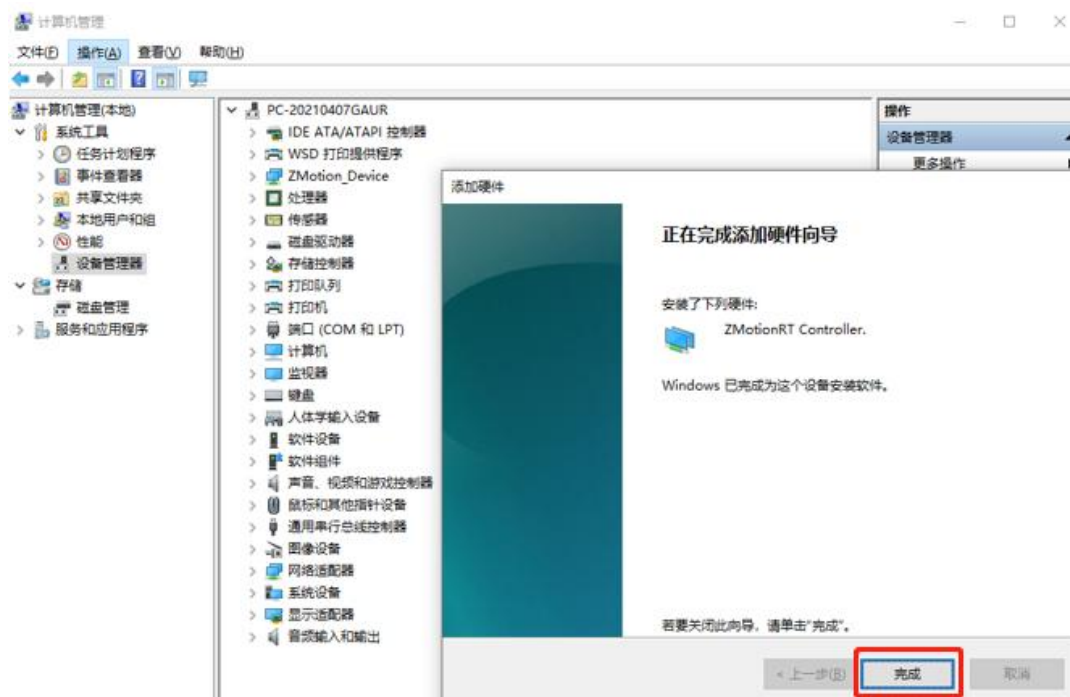
8. The same, click "Next".

添加硬件

向导准备安装你的硬件



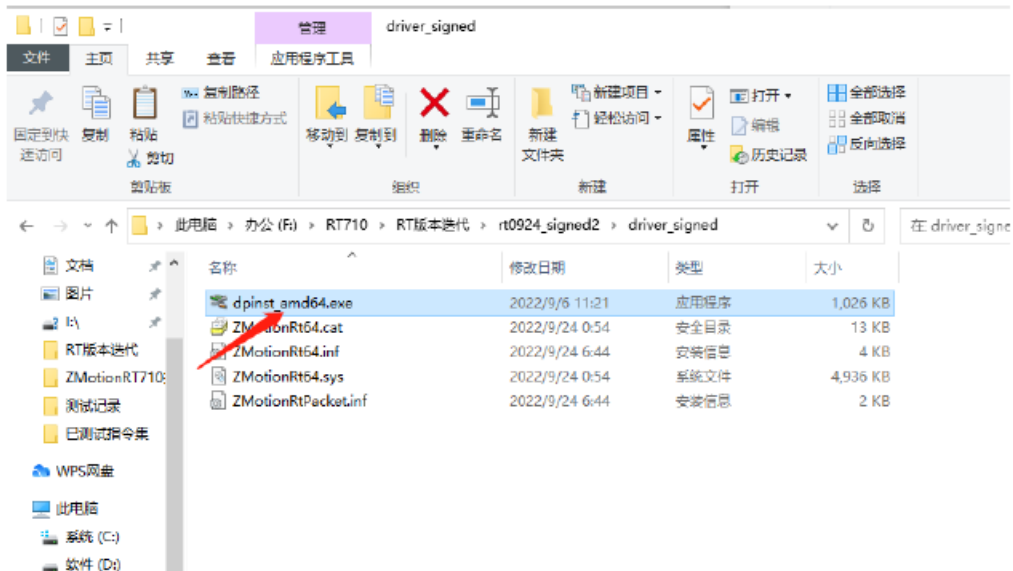
9. The last step, click "complete". If there is ZMotionRT Controller in the device manager, the installation is successful.



When there is PCI card:

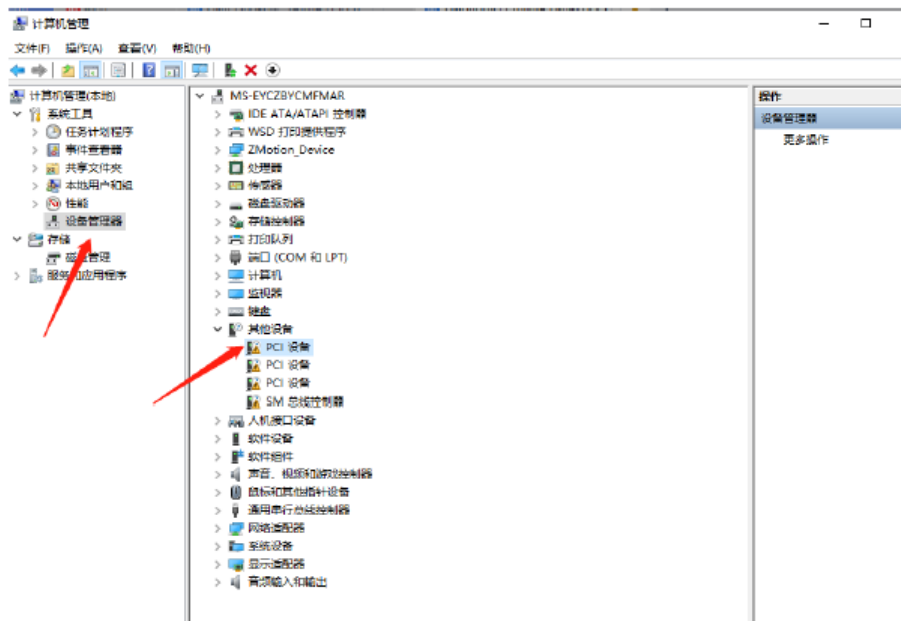
➤ Method 1: install automatically

Use the built-in installation wizard software **dpinst_amd64.exe** (MotionRT7 / driver_signed) in the driver directory to automatically install, and the specific operation is according to the software guide.



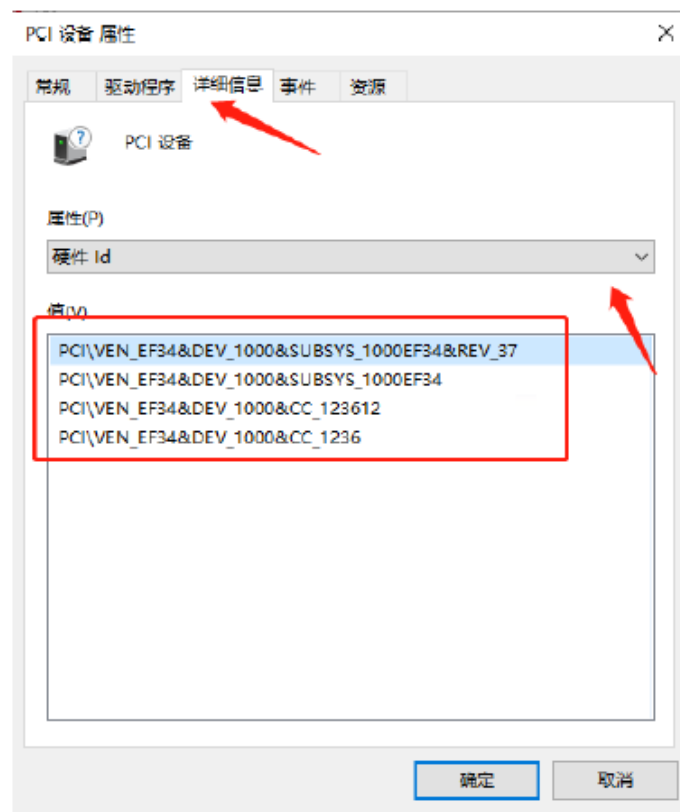
➤ Method 2: install manually

1. Open the Device Manager menu and select the PCI device in Other Devices.

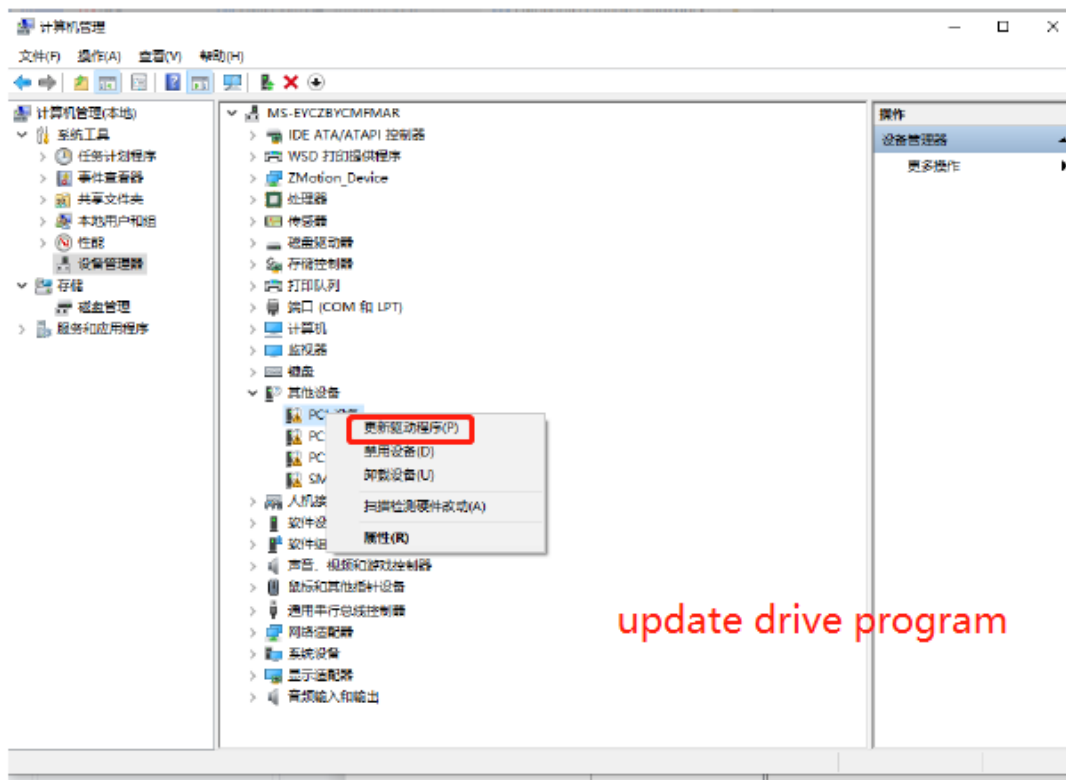


2. If there are multiple PCI devices, right-click "Properties" to view detailed information, select "Hardware ID" for properties, and confirm that it is a PCI device starting with

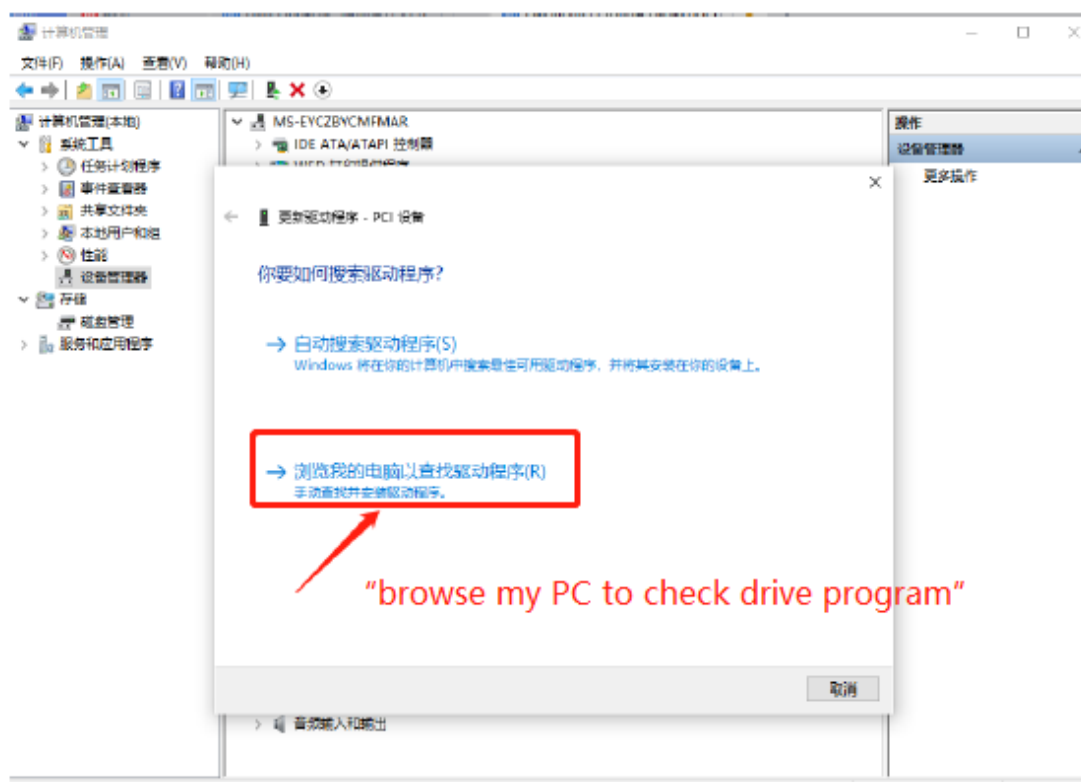
PCI\VEN_EF34&DEV_1000&.



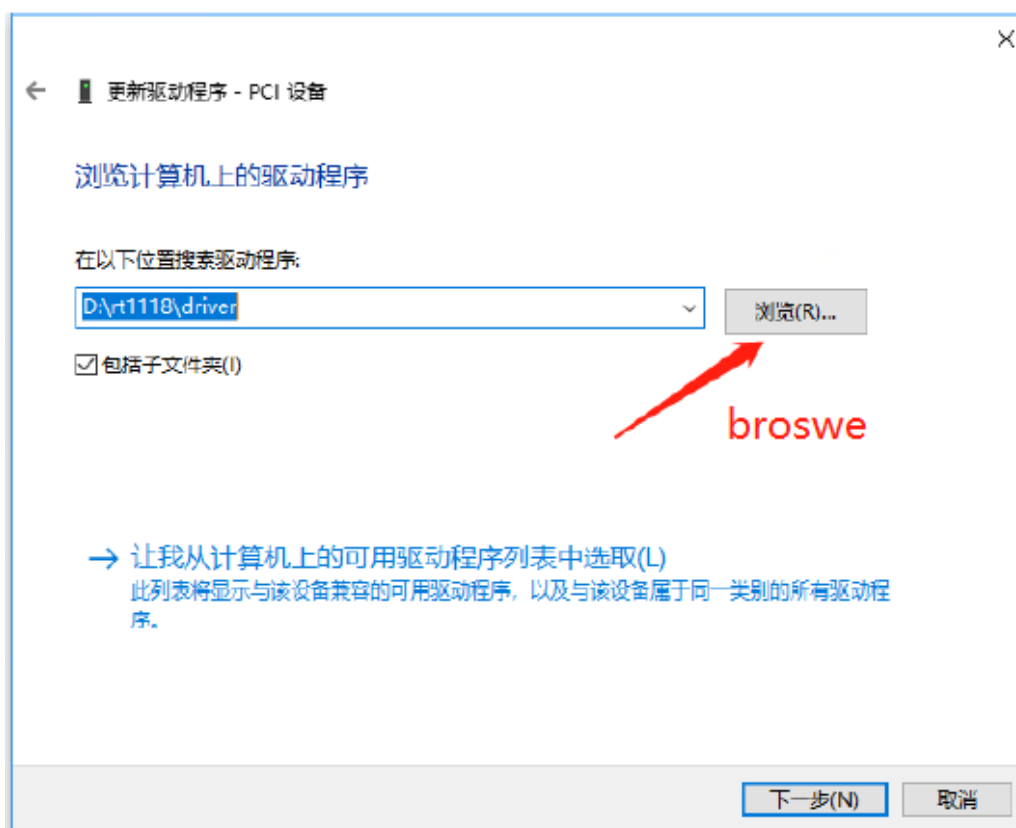
- Find PCI Device, right-click to select "update drive program".



4. Select "browse my PC to check drive program".



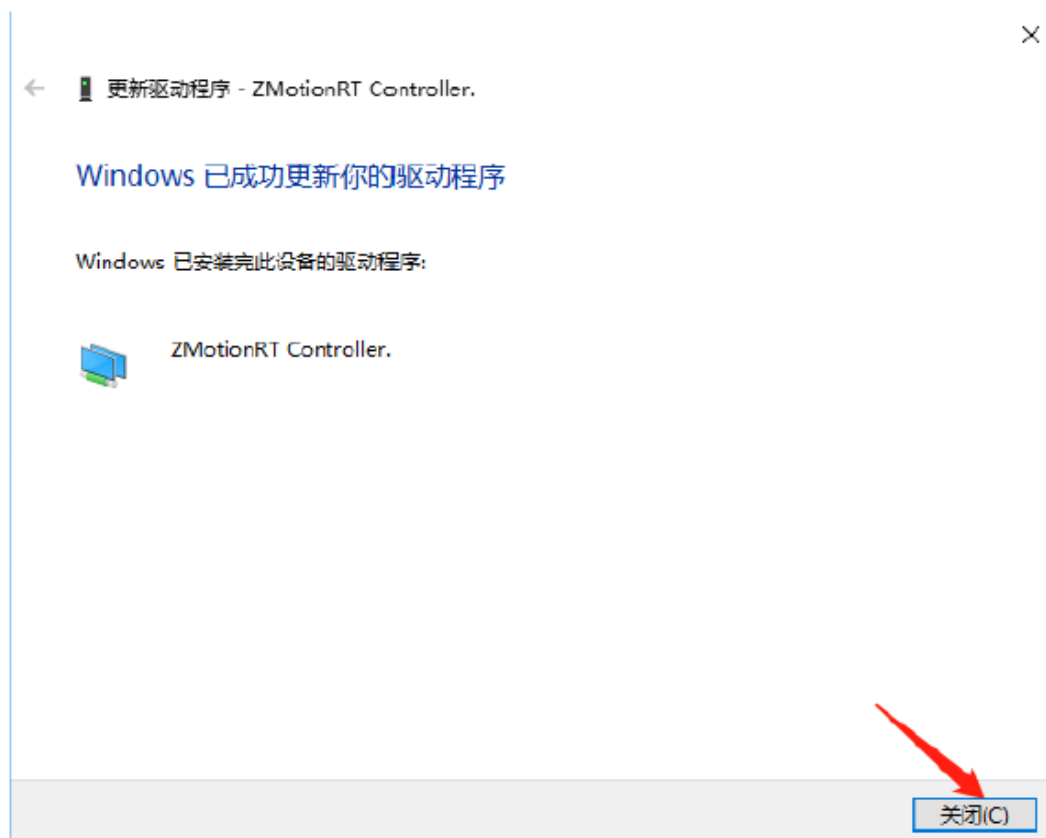
5. Click "browse", and select driver folder.



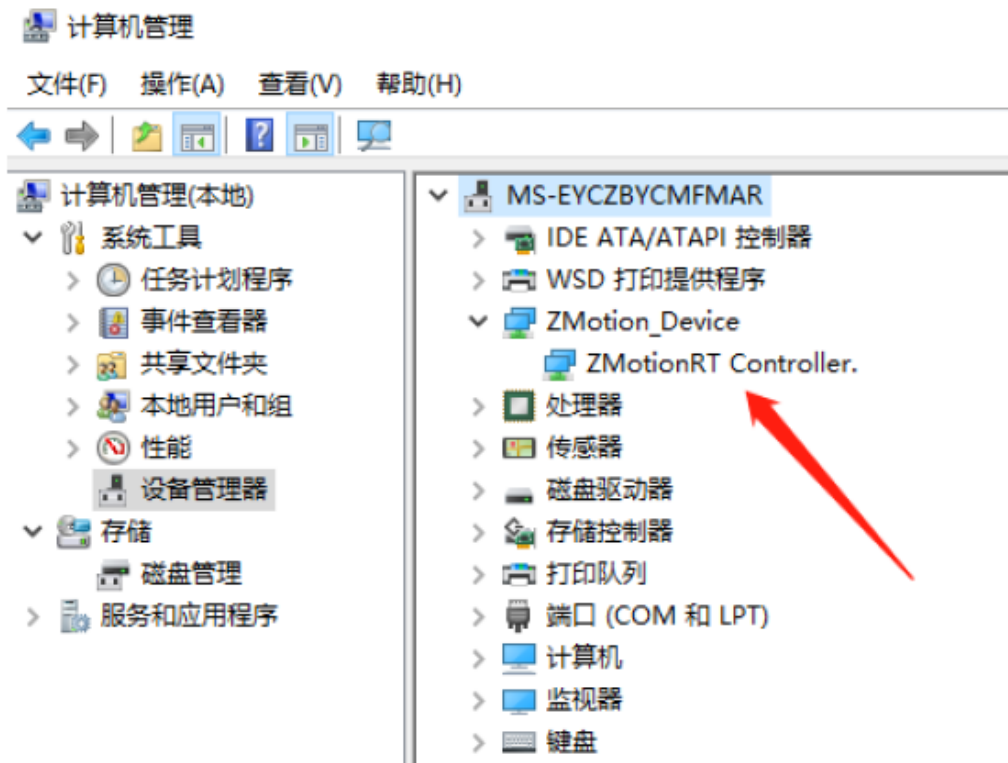
6. Click "next step".



7. Wait until installed, click close.



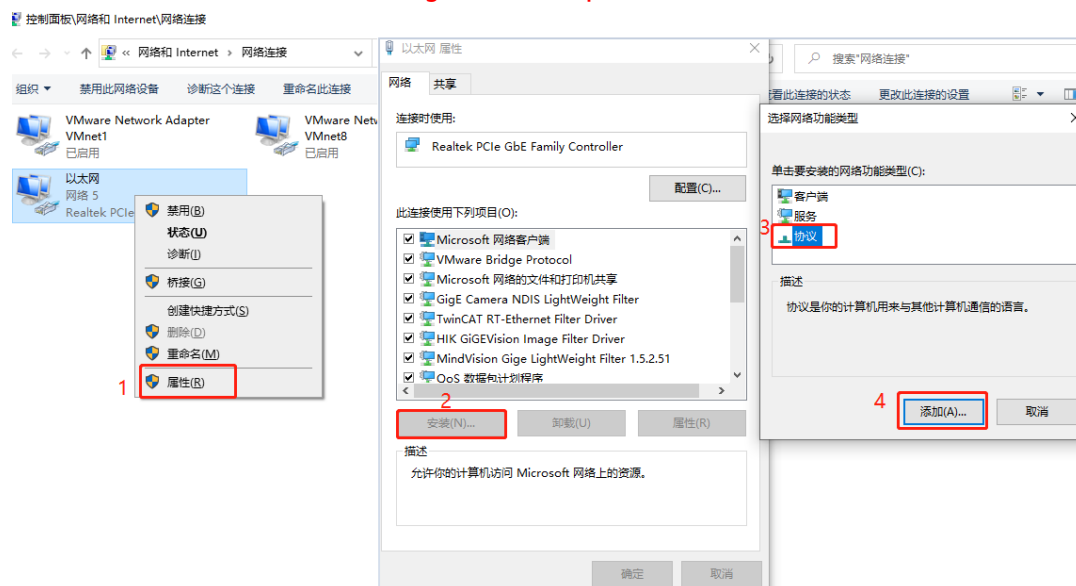
8. If there is ZMotionRTController in the device manager, the installation is successful.



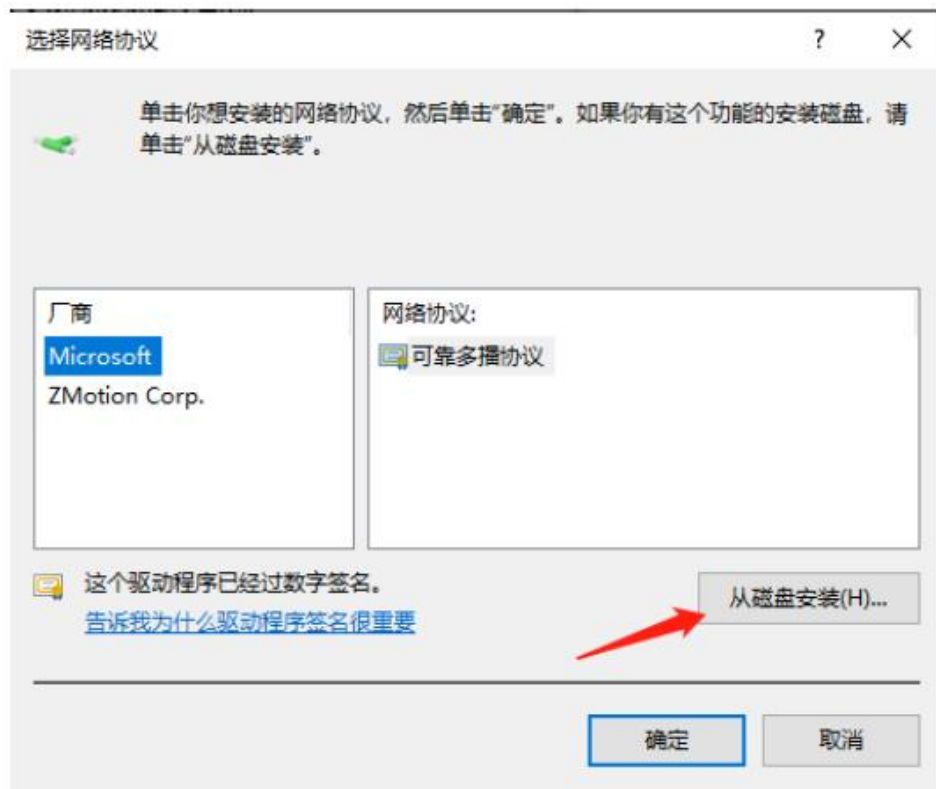
Install EtherCAT for normal Ethernet Card:

MotionRT710 supports the ECAT network port of XPCI/XPCIE, and also supports the common network port of the computer as ECAT.

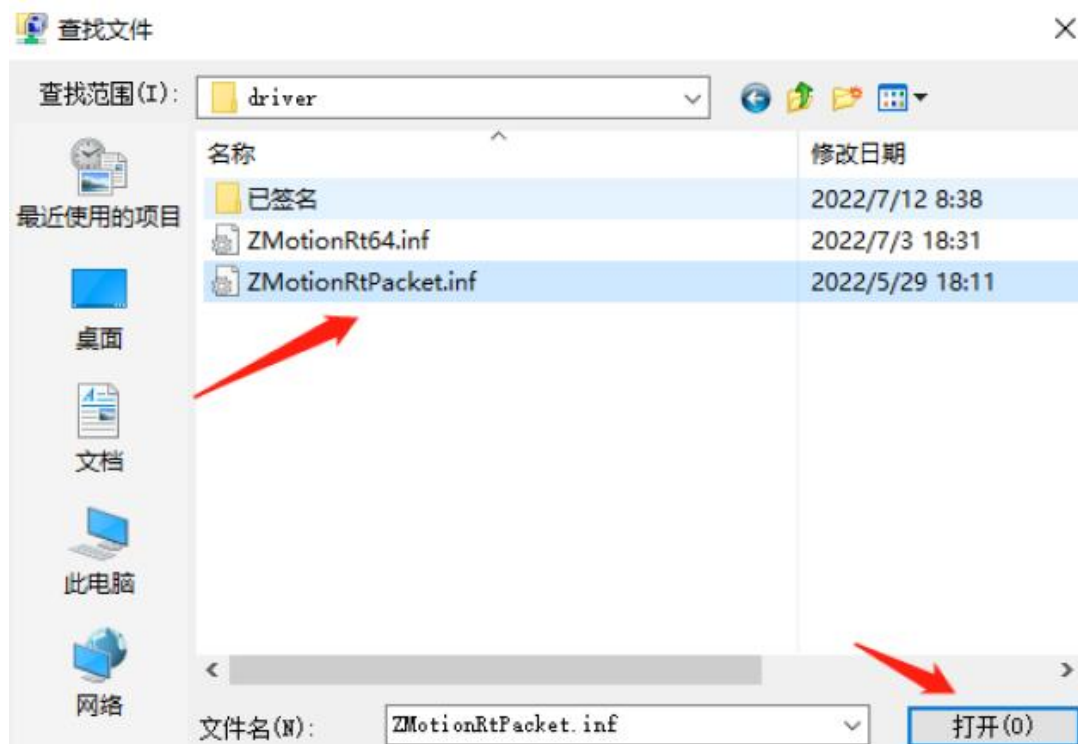
1. Open control panel, in Windows network connection page, select the Ethernet that needs to be used as the bus, **right-click Properties->Installation->Protocol->Add.**



2. Select "installation from disk".



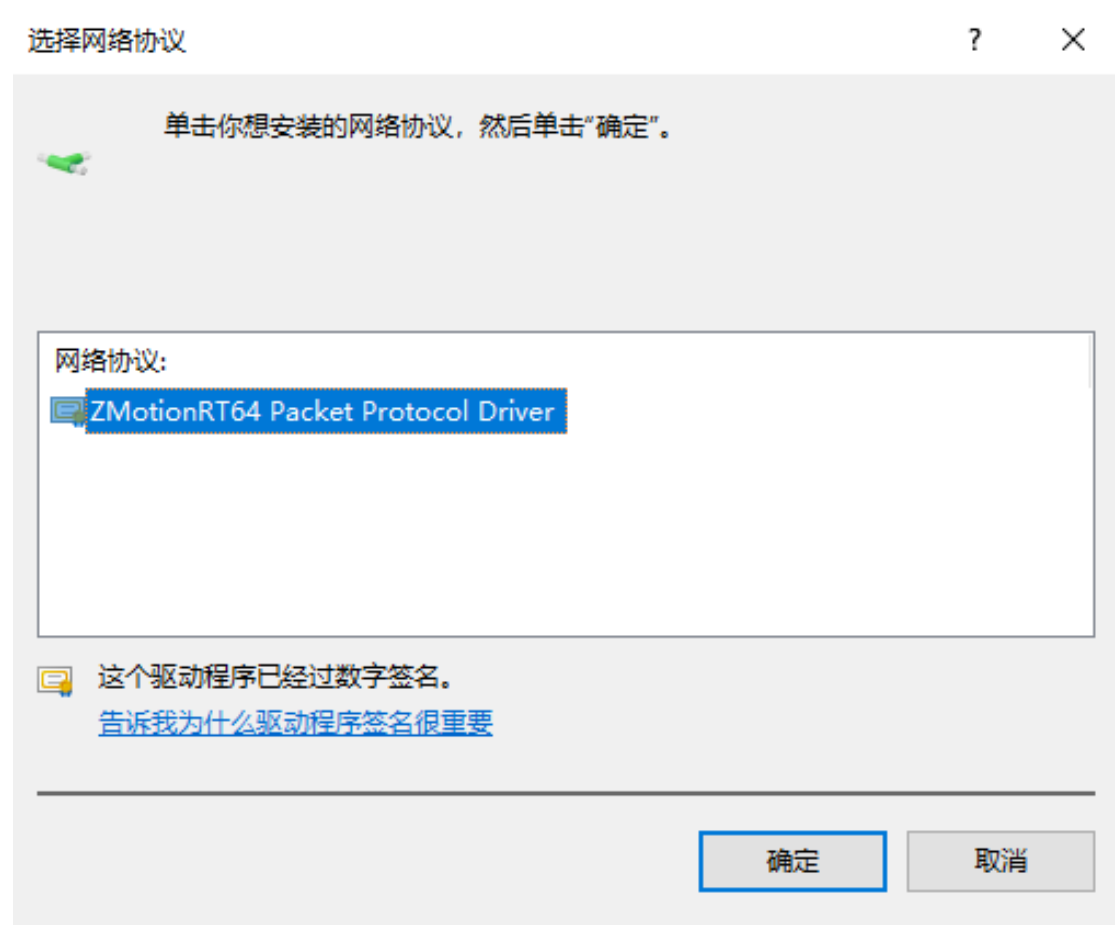
3. Browse drive position, select "ZMotionRtPacket.inf".



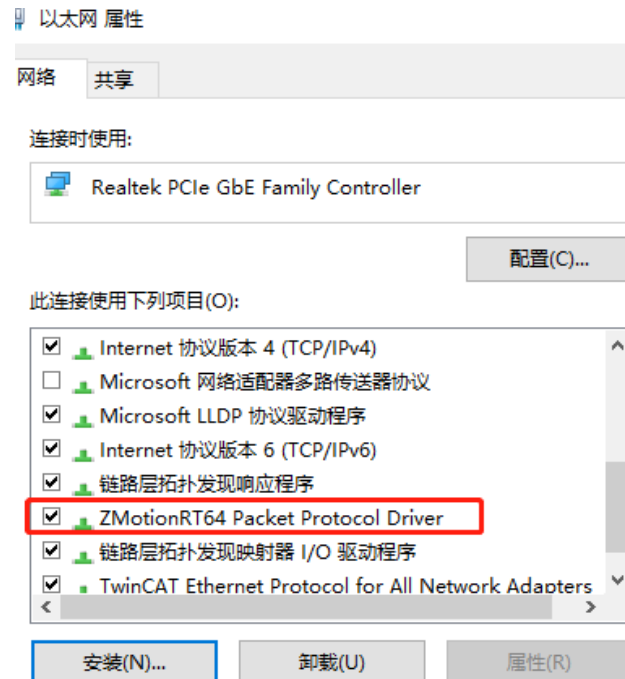
4. Click "ok".



5. Select "ZMotionRT64PacketProtocolDriver" in network protocol, then click "ok".

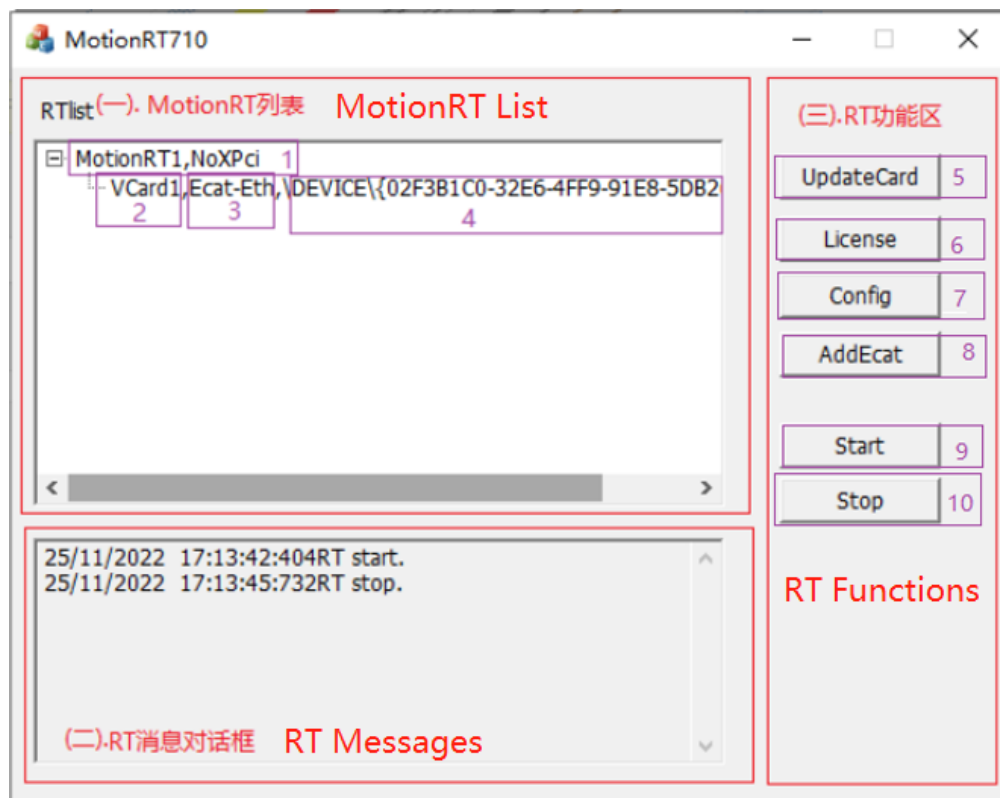


6. After completion, there will be an additional "ZMotionRT64 Packet Protocol Driver" in the Ethernet attribute protocol, which means the installation is successful.



How to Use MotionRT Control Panel

- Mian Interface



(1) MotionRT List:

That is, the list of PCI cards currently present on the computer

[1]: PCI card model, NoXPci means no card

[2]: Type of card.

[3]: ID of the card. Ecat-Eth refers to the network port used when there is no card.

[4]: The dialing address of the card. When there is no card, it means the unique No. of the network card

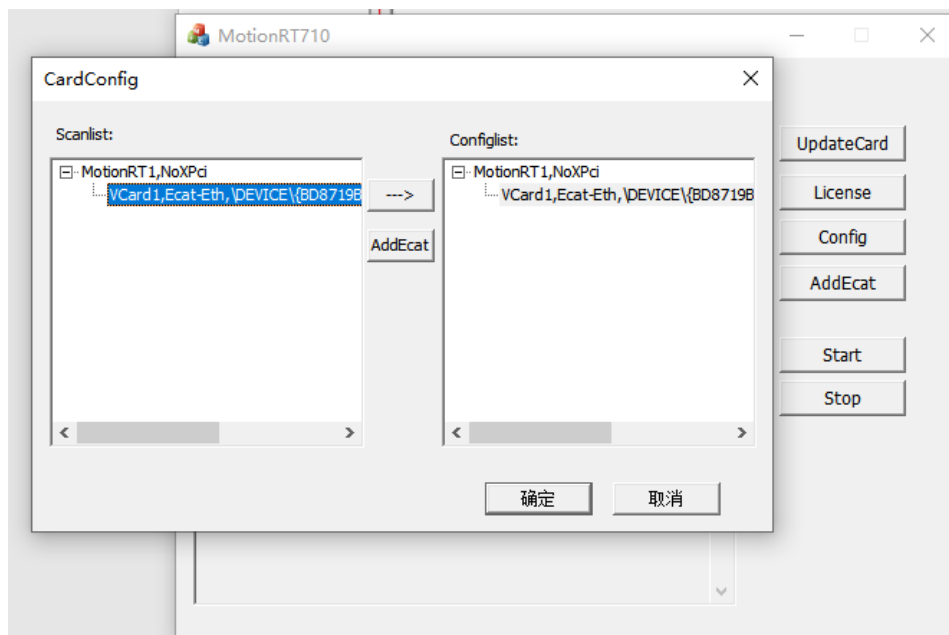
(2) RT Messages (dialogue area):

A message dialog box for interaction between the console and each card.

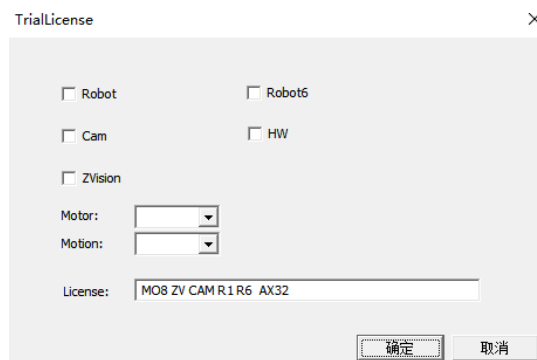
(3) RT Functions Area:

[5]: UpdateCard: refresh the configuration information of the card, and save it, and automatically load the configuration information when the console is opened next time.

When using it for the first time, you need to manually add the card number information (AddEcat), as shown in the figure below:



[6]: License function interface.



Function	Description
Robot	Normal robotic arm.
Robot6	6 axes and robotic arms are with below 6 axes.
Cam	Camera functions
HW	Hardware position comparison output function.
ZVision	Time instruction function.
Motor	The number of actual motors.
Motion	Valid motion control functions: MO0: point to point MO1: point to point, electronic cam. MO2: point to point, electronic cam, linear interpolation. MO6: point to point, electronic cam, linear interpolation, circular interpolation. MO8: point to point, electronic cam, linear interpolation, circular interpolation, continuous interpolation.
License	It is used to display the widow that saves License parameters configuration.

[7]: Config configuration interface.

Config

×

Total Memery	256	mbytes
Cpu Usage	90	%
Hmi num	2	1-4
Hmi size	5242880	128k-100m
Zvision Latin num	4	1-8
Zvision Latin size	5242880	128k-100m
Zar size	33554432	128k-512m
Task num	15	2-30
Array space	2560000	128k-10m
Zvobj num	2048	2048-16000
Sub num	4096	256-10000
Com num	0	0-8
Eth num	2	0-12
Ethcustom num	0	0-12
3Dsolid num	128	16-512

☐ AutoRunApp
☐ AutoRunRT
☐ RtLogSave

确定

取消

Function	Description
Total Memery	Total memory, including the memory occupied by all the spaces in the controller that can save data, such as, array space, Zar file size, channel size, hmi resolution, etc., it is best to set value above 200.
Cpu Usage	CPU usage limit.
Hmi num	The number of Hmi that can be used.
Hmi size	The resolution of Hmi.
Zvision Latin num	Vision channels.
Zvision Latin size	Single vision channel size.
Zar size	Zar file size.
Task num	The number of tasks that can be executed.
Array Space	Assigned array space.
Zvobj num	The number of Zvobject.
Sub num	The number of max sub-function.
Com num	The number of serial ports.
Eth num	The number of network ports PORT, the set value should be smaller than the max value.
Ethcustom num	Customized Ethernet PORT numbers.
3Dsolid num	The number of 3D solids.
AutoRunApp	Open the software automatically when power on.
AutoRunRT	Automatically run RT software when power on.
RtLogSave	Save logs into log file.

[8]: AddEcat

Add connectable card.

[9]: Start

Start the MotionRT, after starting, you can connect to the control card.

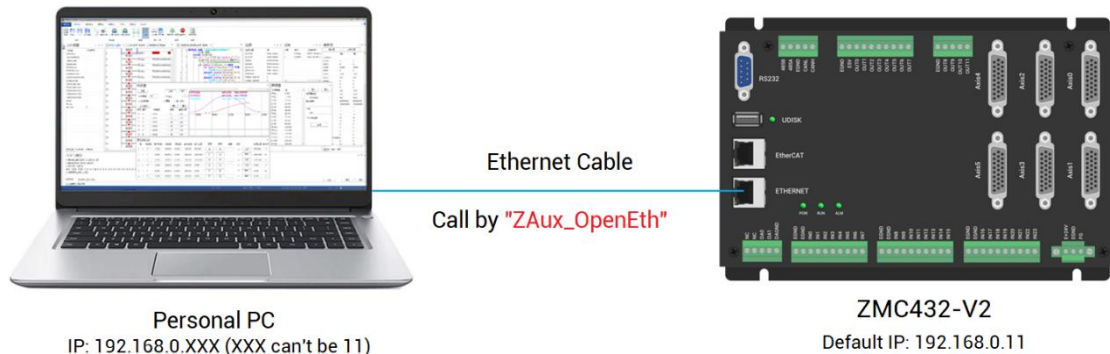
[10]: Stop

Stop the MotionRT, after stopping, the control card cannot be connected.

4.1.3. Routines

Connect to Controller by Ethernet

This example shows controller connection by Ethernet. If it fails, the error code will be printed.



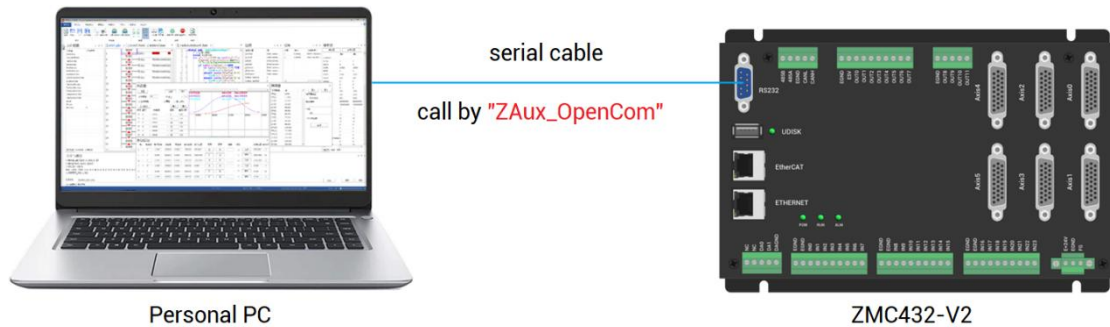
```
// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // if it is not 0, fail
    {
        printf("%s return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1";
    // Controller IP address is 192.168.0.11, simulator IP address is 127.0.0.1.
    ZMC_HANDLE handle = NULL; // link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); // connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        getchar();
        return -1;
    }
    printf("Controller Connect Succeed!\n");

    // get motion types
    Sleep(2000);
    ret = ZAux_Close(handle); // close the connection
    commandCheckHandler("ZAux_Close", ret); // if it executes successfully.
    printf("connection closed!\n");
    handle = NULL;
    return 0;
}
```

Connect to Controller by Serial Port



```
// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret)//if it is not 0, fail
    {
        printf("%s return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    ZMC_HANDLE handle = NULL; //link the returned handle

    uint32 comid = 1; //connected COM port

    int ret = ZAux_OpenCom(comid,&handle); //COM1 connects to controller
    commandCheckHandler("ZAux_OpenCom", ret);
    if (ERR_SUCCESS != ret)
    {
        printf("serial connect failed!\n");
        handle = NULL;
    }
    else
    {
        printf("serial connect succeeded!\n");
    }

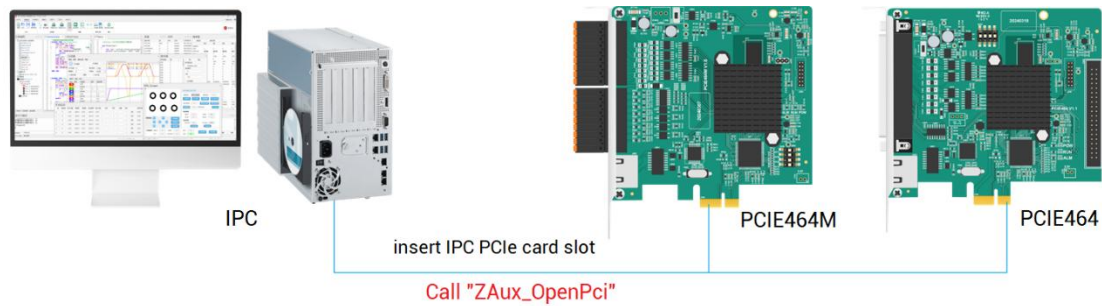
    uint32 dwBaudRate = 38400; //baud rate 38400
    uint32 dwByteSize = 8;    //8-bit data bit
    uint32 dwParity = 0;      //no parity +
    uint32 dwStopBits = 0;    //stop bit
    ret = ZAux_SetComDefaultBaud(dwBaudRate, dwByteSize, dwParity, dwStopBits); //set serial
    port communication parameters
    commandCheckHandler("ZAux_SetComDefaultBaud", ret);
    //do something...

    Sleep(2000);
    ret = ZAux_Close(handle); //close the connection
    commandCheckHandler("ZAux_Close", ret) ; //if it executes successfully.
    printf("connection closed!\n");
}
```

```
    handle = NULL;  
    return 0;  
}
```

Connect to Controller by PCI

If it fails, error codes will be printed, and return directly.



```
// test1.cpp: define the entry point of control panel application program  
//  
#include "stdafx.h"  
#include <windows.h>  
#include "zmotion.h"  
#include "zauxdll2.h"  
  
void commandCheckHandler(const char *command, int ret)  
{  
    if (ret) // if it is not 0, fail  
    {  
        printf("%s return code is %d\n", command, ret);  
    }  
}  
  
int _tmain(int argc, _TCHAR* argv[])  
{  
    ZMC_HANDLE handle = NULL; // link returned handle  
    uint32 cardnumcardnum = 0; // PCI0  
  
    int ret = ZAux_OpenPci(cardnumcardnum, &handle); // connect to controller by PCI  
    commandCheckHandler("ZAux_OpenPci", ret);  
    if (ERR_SUCCESS != ret)  
    {  
        printf("PCI connect failed!\n");  
        Sleep(3000);  
        handle = NULL;  
        return 0;  
    }  
    else  
    {  
        printf("PCI connect succeeded!\n");  
    }  
  
    // do something.....  
    Sleep(3000);  
  
    ret = ZAux_Close(handle);  
    commandCheckHandler("ZAux_OpenPci", ret);  
}
```

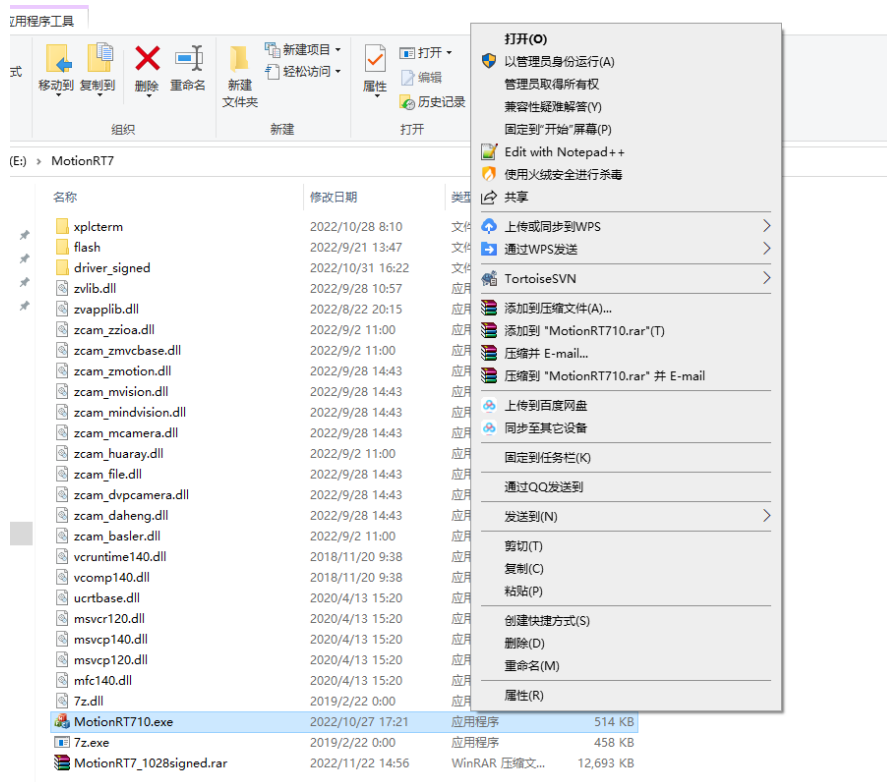
```
}    return 0;  
}
```

How to Modify IP Address

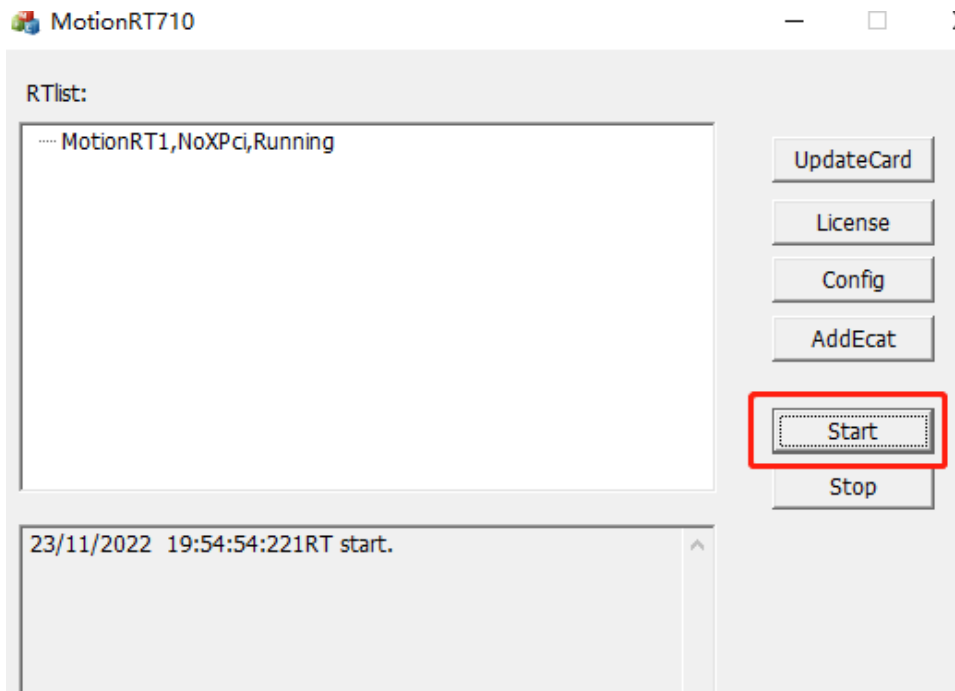
```
// test1.cpp: define the entry point of control panel application program  
//  
#include "stdafx.h"  
#include <windows.h>  
#include "zmotion.h"  
#include "zaxdll2.h"  
  
void commandCheckHandler(const char *command, int ret)  
{  
    if (ret) // if it is not 0, fail  
    {  
        printf("%s return code is %d\n", command, ret);  
    }  
}  
  
int _tmain(int argc, _TCHAR* argv[])  
{  
    char *ip_addr = (char *) "127.0.0.1";    //connect to controller / simulator.  
    ZMC_HANDLE handle = NULL;    //link handle  
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller  
    if (ERR_SUCCESS != ret)  
    {  
        printf("fail to connect controller!\n");  
        handle = NULL;  
        getchar();  
        return -1;  
    }  
    printf("success to connect controller\n");  
  
    ret = ZAux_SetIp( handle, (char *) "192.168.0.11"); //simulator IP can't be modified, only  
    //controller IP can be modified, this routine is for introducing, so connect to the simulator.  
    commandCheckHandler("ZAux_Close", ret); // if it executes successfully  
  
    //get the motion type}  
    Sleep(2000);  
    handle = NULL;  
    return 0;  
}
```

MotionRT Connection

Connect to MotionRT. At first, open MotionRT control panel.



Open the console, configure the function and parameters, then click Start



Note: zmotion.dll version must be above 3.8.8.50.

```
// test1.cpp: define the entry point of control panel application program.
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
```

```
{
    if (ret)//if it is not 0, fail.
    {
        printf("%s return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    ZMC_HANDLE handle = NULL;          //link handle

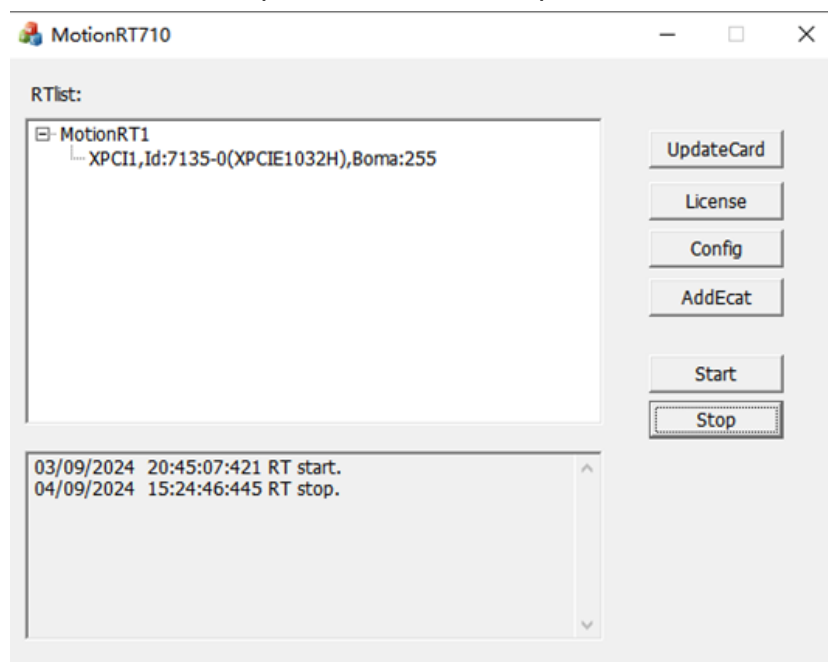
    char MotionID[32]= ""; //no requirements on character strings, empty string can be filled.
    int ret=ZAux_FastOpen(5, MotionID,1000 ,&handle); //link to MotionRT

    if (ERR_SUCCESS != ret)
    {
        printf("fail to connect controller!\n");
        handle = NULL;
        getchar();
        return -1;
    }
    printf("success to connect controller!\n");

    //get the motion type}
    Sleep(2000);
    handle = NULL;
    return 0;
}
```

MotionRT Start & Stop

Connect to MotionRT. At first, open MotionRT control panel.



```
// test1.cpp: define the entry point of control panel application program.
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret)    // if it is not 0, fail
    {
        printf("%s return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    ZMC_HANDLE handle = NULL;    //connection handle
    int isel=0;
    int ret=0;

    if (!ZAux_GetLocalRTState(isel))
    {
        printf("MotionRT Console OFF!!\n");
        ret=ZAux_LocalRTManage(isel,1);
        if (ERR_SUCCESS != ret)
        {
            printf("MotionRT Console Start Fail!!\n");
            ZAux_Close(handle);
            handle = NULL;
            getch();
            return -1;
        }
    }
    else
    {
        printf("MotionRT Console Started!!\n");
    }

    char MotionID[32]= ""; //no requirements on character strings, empty string can be filled.
    ret=ZAux_FastOpen(5, MotionID,1000,&handle);    //connect to MotionRT

    if (ERR_SUCCESS != ret)
    {
        printf("Controller Connect Failed!\n");
        handle = NULL;
        getch();
        return -1;
    }
    printf("Controller Connect Succeeded!\n");

    //get motion type}
    Sleep(2000);
    ZAux_Close(handle);
    handle = NULL;
    ret=ZAux_LocalRTManage(isel,2);
    if (ERR_SUCCESS != ret)
    {
```

```
        printf("Console Stop Failed!\n");

        getchar();
    }

    return 0;
}
```

4.2. How to Get Controller Information?

4.2.1. Instruction List

Commands	Descriptions
ZAux_GetControllerInfo	Get controller / card information
ZAux_GetSysSpecification	Get controller max specification
ZAux_GetRtcTime	Read controller RTC time
ZAux_SetRtcTime	Set controller RTC time

4.2.2. Principle & Description

In order to check the relevant information of the motion controller, there are 2 commands that can get controller information, ZAux_GetControllerInfo and ZAux_GetSysSpecification. It includes: controller model, controller software version (firmware version), unique ID of the controller, max virtual axes, max motors, max IN, OUT, AD, and DA, system date, system time, etc. If you want to get more parameters, you can send "ZAux_Execute" or use the string "?*max" or "?*set".

4.2.3. Routines of Getting Controller Info

This routine mainly shows how to obtain controller information by commands.

```
// test1.cpp: define the entry point of control panel application program.
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
```

```
if (ret)//if it is not 0, fail
{
    printf("%s return code is %d\n", command, ret);
}
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"127.0.0.1";    //connect to controller (default IP: 192.168.0.11) /
    simulator (127.0.0.1)
    ZMC_HANDLE handle = NULL;                //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("fail to connect controller! \n");
        handle = NULL;
        getchar();
        return -1;
    }
    printf("success to connect controller! \n");

    char SoftTypes[20];
    char SoftVersions[20];
    char ControllerIDs[20];

    ret = ZAux_GetControllerInfo(handle, SoftTypes, SoftVersions, ControllerIDs); //get control card
    information
    commandCheckHandler("ZAux_GetControllerInfo", ret) ;//if it executes successfully
    printf("SoftType = %s\n", SoftTypes);        //print controller model
    printf("SoftVersion = %s\n", SoftVersions);   //print version No.
    printf("ControllerId = %s\n", ControllerIDs); //print controller ID
    uint16 Max_VirtuAxes1;
    uint8 Max_motor1;
    uint8 Max_io1[4];
    ret = ZAux_GetSysSpecification(handle, &Max_VirtuAxes1, &Max_motor1, Max_io1); //get the
    max specification of control card.
    commandCheckHandler("ZAux_GetSysSpecification", ret) ;//if it executes successfully
    printf("Max_VirtuAxes1 = %u\n", Max_VirtuAxes1); //max virtual axes
    printf("Max_motor1 = %u\n", Max_motor1); //max motor axes
    printf("Max_io1[0] = %u\n", Max_io1[0]); //max IN
    printf("Max_io1[1] = %u\n", Max_io1[1]); //max out
    printf("Max_io1[2] = %u\n", Max_io1[2]); //max AD
    printf("Max_io1[3] = %u\n", Max_io1[3]); //max DA

    Sleep(2000);
    ret = ZAux_Close( handle); //close the connection
    commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
    handle = NULL;
    return 0;
}
```

Running result:

```
控制器连接成功!
控制器的型号 = VPLC5xx-Simu
软件版本号 = 20180511
控制器的ID = 1234
最大虚拟轴数 = 64
最大电机轴数 = 64
最大IN = 27
最大out = 15
最大AD = 0
最大DA = 2
-
```

Chapter V How to Set Motion Parameters?

In the motion control system, the object of motion control is called "axis", and a motor-controlled motion platform in the motion control system is called a motion axis. Each motion axis has only one degree of freedom, which can do linear motion or rotary motion.

Below shows Zmotion motion control axes types:

Axis Type	Description
Motor Axis	Use controller pulse axis, EtherCAT / RTEX to connect to the driver, and assign the axis No. to the device, then one motor is used as one axis.
Virtual Axis	The virtual axis is in the motion controller, a real drive is not used. Or it is used as a virtual master axis for synchronous control and as a Cartesian axis in the robot algorithm.
Encoder Axis	Use the controller's local encoder axis interface, assign it as the actual encoder input.

Motor axis: it actively operates, the motor moves according to the pulses sent by the controller. The number of pulses sent is determined by (motion parameter variation)*(Units set by ZAux_Direct_SetUnits), and the target demand position is reflected by the planning position parameter of ZAux_Direct_GetDpos.

Encoder axis: it passively operates, the encoder follows the rotation of the motor, generates pulses, and feeds back to the controller. The number of pulses received by the controller is determined by calling ZAux_Direct_GetEncoder, and the encoder feedback position is reflected by the obtained ZAux_Direct_GetMpos value.

After you connected a whole system (including motion controller, drive, motor), here will show you:

- ✓ how to check the status of the whole system?
- ✓ how to configure this system?
- ✓ how to check the alarm information?
- ✓ how to configure speed, acceleration and deceleration?
- ✓

5.1. Motion Basic Parameters

5.1.1. Instruction List

Commands	Descriptions
ZAux_Direct_SetAtype	Set axis type
ZAux_Direct_GetAtype	Read axis type
ZAux_Direct_SetUnits	Set pulse amount (units)
ZAux_Direct_GetUnits	Read pulse amount (units)
ZAux_Direct_SetAccel	Set acceleration, unit: units/s/s
ZAux_Direct_GetAccel	Read acceleration, unit: units/s/s
ZAux_Direct_SetDecel	Set deceleration, unit: units/s/s
ZAux_Direct_GetDecel	Read deceleration, unit: units/s/s
ZAux_Direct_SetSpeed	Set speed, unit: units/s/s
ZAux_Direct_GetSpeed	Read speed, unit: units/s/s
ZAux_Direct_SetDpos	Set axis demand position, unit: units
ZAux_Direct_GetDpos	Read axis demand position, unit: units
ZAux_Direct_SetMpos	Set axis measurement position, unit: units
ZAux_Direct_GetMpos	Read axis measurement position, unit: units
ZAux_Direct_GetEncoder	Get controller received pulses
ZAux_Direct_SetFastDec	Set fast deceleration, unit: units/s/s
ZAux_Direct_GetFastDec	Read fast deceleration, unit: units/s/s
ZAux_Direct_SetLspeed	Set axis initial speed, unit: units/s
ZAux_Direct_GetLspeed	Read axis initial speed, unit: units/s
ZAux_Direct_SetSramp	Set S curve, 0: T-shaped ACCEL & DECL
ZAux_Direct_GetSramp	Read S curve, 0: T-shaped ACCEL & DECL
ZAux_Direct_GetMspeed	Read measurement speed, unit: units/s
ZAux_Direct_GetVpSpeed	Read command speed: unit: units/s
ZAux_Direct_SetVpMode	Set the ACCEL & DECEL curve
ZAux_Direct_GetIfIdle	Read whether the axis ends
ZAux_Direct_GetAxisStatus	Read axis alarm state
ZAux_Direct_GetAxisStatus2	Read axis enable state
ZAux_Direct_GetAxisStopReason	Read axis history abnormal stop reason

5.1.2. Principle & Description

Axis State

You can get the axis' state by **ZAux_Direct_GetAxisStatus** command. The value is displayed in decimal system, but the status is judged by the corresponding bit in binary. Multiple errors can occur at the same time.

The word definition of the axis status is shown as below:

Bit	Description	Print Values	
1	Alarm: Follow-up errors exceed	2	2h
2	Error of communication with remote axis	4	4h
3	Remote driver alarms	8	8h
4	Forward hard position limit	16	10h
5	Reverse hard position limit	32	20h
6	While finding the origin	64	40h
7	HOLD speed keeps signal input	128	80h
8	Error: Follow-up error out of limit	256	100h
9	Exceed forward soft position limit	512	200h
10	Exceed reverse soft position limit	1024	400h
11	CANCEL is in execution	2048	800h
12	Pulse frequency exceeds MAX_SPEED, it needs to modify the speed (slow down) or MAX_SPEED	4096	1000h
14	Wrong robot instruction coordinates	16384	4000h
18	Abnormal power	262144	40000h
20	Axis speed protection	1048576	100000h
21	Fail to trigger special motion instruction in motion	2097152	200000h
22	Alarm signal inputs	4194304	400000h
23	Axis has entered in pause state	8388608	800000h

Call **ZAux_Direct_GetAxisStopReason** to view the axis history stop reasons, and write 0 to clear, latch bit by bit, and the information that is to be latched is axis status.

Call **ZAux_Direct_GetIdle** to judge whether the motion command added to the axis is completed, 0 = in motion, -1 = motion ends.

Axis Type

Use **ZAux_Direct_SetAtype/ZAux_Direct_GetAtype** (ATYPE) command to configure /

read the axis type. And please complete it in your program initialization. And it will report an error when the axis type is not matched.

Note: all unassigned axes default to virtual axes with an ATYPE of 0.

The axis types supported by the controller are as follows:

Atype	Description
0	Virtual axis
1	Servo or stepper of pulse direction
2	Servo controlled by analog signal (reserved)
3	Quadrature encoder
4	Pulse directional output + quadrature encoder input
5	Pulse directional output + pulse directional encoder input
6	Encoder of pulse directional method
7	Servo or stepper of pulse direction + EZ signal input
8	Expand servo or stepper of pulse direction through ZCAN
9	Expand quadrature encoder through ZCAN
10	Expand encoder of pulse direction through ZCAN
20	Galvanometer type, with galvanometer status feedback. AXISSTATUS Bit 2 will be reset if the galvanometer cannot be connected, and ENCODER will return to the original sending position, the unit is pulse. ✓ ZMC408SCAN, ZMC408SCAN-V22 supports.
21	Galvanometer axis type, it needs controller support. The default system period is 250us, and the galvanometer refresh period is 50us, which is related to the firmware. All motion control instructions of common axes can be used, and mixed interpolation of galvanometer axes and other axis types is supported.
22	Galvanometer type, with galvanometer status feedback. AXISSTATUS Bit 2 will be reset if the galvanometer cannot be connected, and bit3 of AXISSTATUS will be reset when the galvanometer alarms. MPOS will return to the feedback position (MPOS), and the anti-correction process is performed, ENCODER will return to original feedback position. ✓ ZMC408SCAN, ZMC408SCAN-V22 supports.
24	Remote encoder axis type. Valid in ZHD500X handwheel. It needs VPLC5XX series whose firmware version is above 20180404.
50	RTEX cycle position mode, valid in RTEX controller.
51	RTEX cycle speed mode, valid in RTEX controller.
52	RTEX cycle torque mode, valid in RTEX controller. Please close driver 2 degrees of freedom control mode at first, and then set speed limit.
65	EtherCAT cycle position mode, valid in EtherCAT controller.
66	EtherCAT cycle speed mode, valid in EtherCAT controller. DRIVE_PRODILE needs to be set as 20 or above.

67	EtherCAT cycle torque mode, valid in EtherCAT controller. DRIVE_PRODILE needs to be set as 30 or above.
70	ECAT self-defined operation, it only reads data of encoder. Valid in EtherCAT controller.

1. Axis type (ZAux_Direct_SetAtype)=0: virtual axis

When multi-axis synchronous motion can be used as the main axis of the machine, the slave axes all follow this virtual main axis.

As the superposition axis of other axes, a virtual axis is superimposed on the actual moving axis. These virtual axes can be set through the ADDAX command (axis superposition), and then the motion of each virtual axis is superimposed on the real axis.

2. Axis type (ZAux_Direct_SetAtype)=1 or 7: pulse axis

The movement of the axis is controlled by the controller sent pulses, the direction of the pulses determines the steering of the motor, and the speed of the axis movement is controlled according to the frequency of the pulses sent.

3. Axis type (ZAux_Direct_SetAtype)=3 or 6: encoder axis

When the encoder occupies one axis No. independently, the axis type can be 3 or 6 according to the encoder type.

4. Axis type (ZAux_Direct_SetAtype) = 4: pulse axis & encoder axis share axis No.

When the current pulse axis has encoder feedback, the axis type is set to 4, and the pulse output and encoder input signals are on the same axis.

5. Axis type (ZAux_Direct_SetAtype)=8: extended axis by CAN

When the CAN extension axis is used, the axis type of the extended pulse axis is set to 8, and the axis type connected to the encoder axis on the extended axis is set to 9.

6. Axis type (ZAux_Direct_SetAtype)=21: galvanometer axis No.

When connecting to the laser galvanometer device, it needs to set the galvanometer axis type to 21, which is supported by some models of the laser galvanometer axis.

7. Axis type (ZAux_Direct_SetAtype)=50,51,52: RTEX bus axis

When using the RTEX driver, the axis type can only be selected from 50, 51, 52.

- 50 means position mode, using motion commands to control the motor.
- 51 means speed mode, under the speed mode, use DAC to set the running

speed of the motor and make it keep running.

- 52 means torque mode, use the DAC to set the motor torque in torque mode and make it keep running.

To switch between speed and torque modes, firstly set DAC to 0 to prevent accidents, and then use the ZAux_Direct_SetAtype command to switch.

Note: before modifying the axis type (ZAux_Direct_SetAtype) and switching to the torque mode, please set 1st position of the drive parameter Pr6.47 to 0 and close the 2-DOF control mode. Then set the speed limit by parameter Pr3.17. When Pr3.17 (speed limit selection) is 0, the speed limit is set by Pr3.21, and when Pr3.17 is 1, determine the speed limit value (Pr3.21 / Pr3.22) when switching to torque control by SL_SW.

8. Axis type (ZAux_Direct_SetAtype)=65,66,67: EtherCAT axis No.

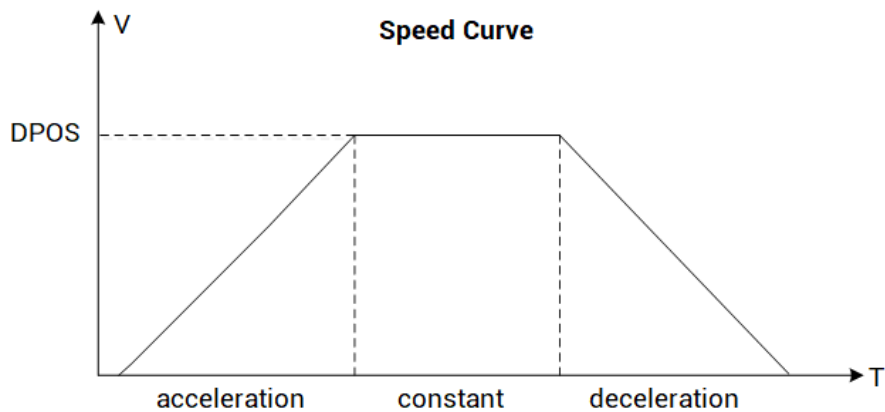
When using the EtherCAT driver, the axis type can only be selected from 65, 66, 67.

- 65 means position mode, using motion commands to control the motor.
- 66 means speed mode, under the speed mode, use DAC command to set the running speed of the motor and make it keep running. And here are 2 speed units, pulses/S & R/MIN, which is determined by the driver.
- 67 means torque mode, use the DAC command to set the motor torque in this torque mode, and make it keep running. The DAC value range is 0-1000 in torque control mode, corresponding to 0-100%.

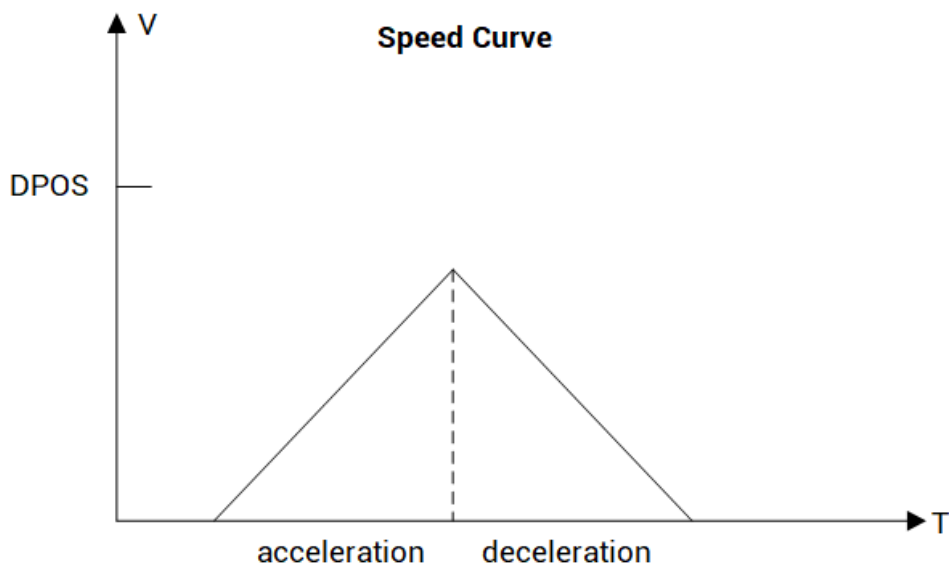
To switch between speed and torque modes, firstly set DAC to 0 to prevent accidents, and then use the ZAux_Direct_SetAtype command to switch.

Axis Speed

For one speed curve, which is generally divided into 3 parts: acceleration, constant speed, and deceleration, like below shown:



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



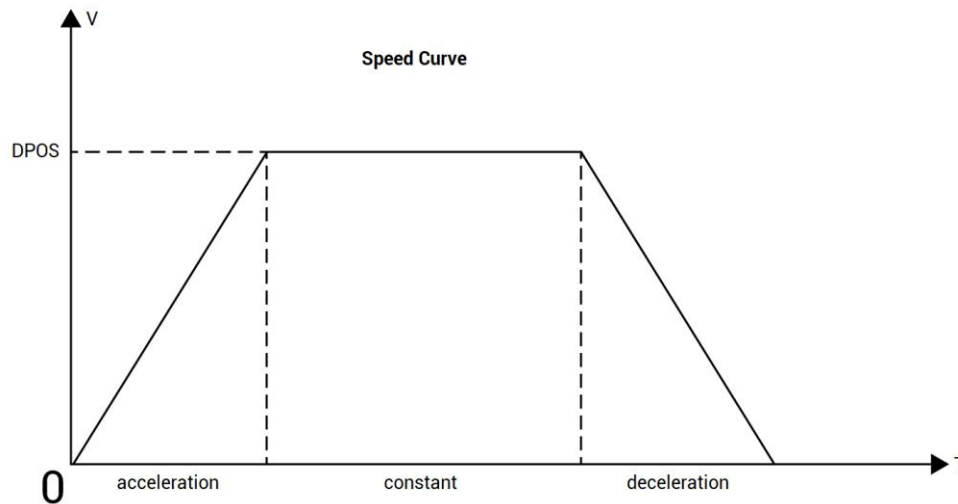
Commonly used speed commands include ***ZAux_Direct_SetSpeed*** (set motion speed), ***ZAux_Direct_SetAccel*** (set acceleration), ***ZAux_Direct_SetDecel*** (set deceleration) and ***ZAux_Direct_SetFastDec*** (set fast deceleration, it can be used to stop rapidly when there is emergency stop, alarm or position limit). And please complete it in axis parameter initialization, they are speed reference for motion commands.

1) Ladder Diagram Curve:

If SRAMP is not set (set SRAMP as 0 -- ***ZAux_Direct_SetSramp***), the speed curve is a trapezoidal curve. Under this speed planning mode, the speed curve changes according to a trapezoidal curve. And parameters don't change, such as speed, acceleration and deceleration.

After the speed reaches the set value, it moves at a constant speed. If only the

acceleration is set, and when the deceleration is 0, the deceleration will automatically be equal to the acceleration value. Generally, the corresponding acceleration and deceleration is set before the motion, and do not modify it during the motion. Otherwise, it will cause incorrect motion trajectory. As shown in the figure: S curve time (ZAux_Direct_SetSramp)=0 (the acceleration and deceleration process is faster, and the speed change has a greater impact on the machine tool.)



For example: suppose the target speed value (DPOS) is 100, the acceleration is 1000, and the deceleration is 1000, then:

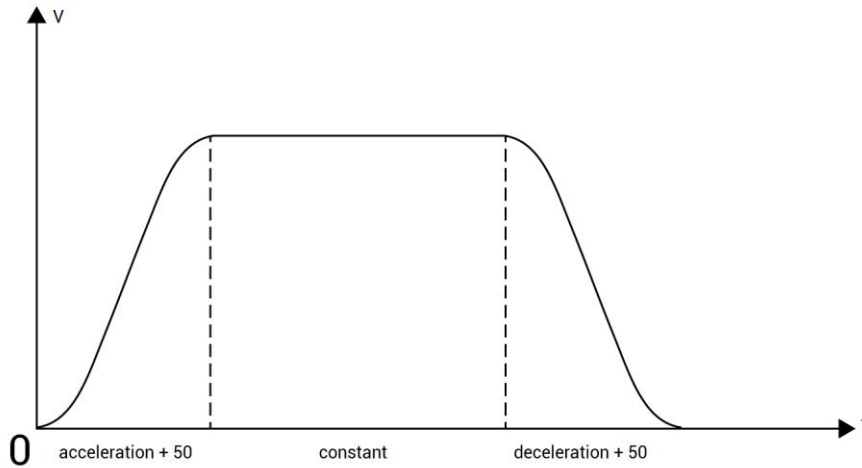
$$\text{Total acceleration time} = 100/1000 = 0.1\text{s}$$

$$\text{Total deceleration time} = 100/1000 = 0.1\text{s}$$

2) S-Curve:

By calling ZAux_Direct_SetSramp to set the appropriate acceleration and deceleration rate of change, so that the speed curve is smooth, and the vibration is reduced when the machine starts and stops or accelerates and decelerates. The value of S-curve time (ZAux_Direct_SetSramp) ranges from 0 to trapezoidal acceleration time (in ms). After setting, the acceleration and deceleration process will take longer for the corresponding time. The longer the time, the smoother the speed curve. If the setting time exceeds the trapezoidal acceleration time, proceed according to the trapezoidal acceleration time.

When the S-curve time (ZAux_Direct_SetSramp) = 50, the S-curve is obtained as follows: the acceleration and deceleration are softer.



For example: suppose the target speed value (DPOS) is 100, the acceleration is 1000, the deceleration is 1000, and the S-curve time is 50ms, then:

$$\text{Total acceleration time} = 100/1000 + 0.05 = 0.1\text{s} + 0.05\text{s} = 0.15\text{s}$$

$$\text{Total deceleration time} = 100/1000 + 0.05 = 0.1\text{s} + 0.05\text{s} = 0.15\text{s}$$

UNITS (Pulse Amount)

In the motion control device, when the mechanical structure is determined, the transmission relationship between the motor and the mechanical device is fixed, and the mechanical displacement generated by the motor per revolution is also fixed. The UNITS pulse equivalent (**ZAux_Direct_SetUnits**) is the corresponding number of pulses per unit, which can be unit distance, unit angle, etc., and it supports 5 decimal part precision.

The controller uses the pulse equivalent (**ZAux_Direct_SetUnits**) as the basic unit, and the target position, speed, acceleration and deceleration of the movement are all performed with the pulse equivalent (**ZAux_Direct_SetUnits**) as the basic unit. After the pulse equivalent (**ZAux_Direct_SetUnits**) is modified, they also will change proportionally.

Assuming a pulse-type motion controller, the controller sends 1000 pulses, and the equipment mechanism moves 1mm, then when the pulse equivalent units is set to 1000, the move-related instructions in motion move 1mm, and 1000 pulses are actually sent. The speed is set as 100 user unit, that is, 100mm/s, when the axis moves 50 user unit, actually it moves 50mm.

Syntax: ZAux_Direct_SetUnits (controller connection handle, axis No., variable)

1) Servo Pulse UNITS Configuration:

- **reduction ratio:** that is, the transmission ratio of the deceleration device, and refers to the ratio of the instantaneous input speed to the output speed in the deceleration mechanism
- **thread lead:** it is the axial distance that any point on the thread moves along the same helix for one revolution. The pitch refers to the axial distance between the corresponding points of two adjacent teeth on the thread, and the code is P. The lead of a single-thread thread is equal to the pitch, and the lead of a multi-thread thread is equal to the number of heads (n) multiplied by the pitch, that is, the lead $S=nP$

If the mechanism has a lead (in mm) and a reducer, the pulse equivalent should be converted into mm as:

Pulse equivalent = electronic gear ratio * lead / encoder resolution * mechanical reduction ratio

Notes:

- a) The motor rotates a circle, and the reducer output rotates b circles, then the reduction ratio is "a: b".
- b) If the motor is directly connected to screw, the mechanical reduction ratio is 1:1
- c) If the axis is a rotary shaft, the lead is 360°.
- d) The number of pulses per circle = encoder resolution / electronic gear ratio

2) Step Pulse UNITS Configuration:

- **step angle:** a pulse, the angle at which the stepper motor rotates.
- **subdivision:** subdivide the step angle into N equal parts, then one pulse, the angle at which the stepper motor rotates is "step angle/subdivision".

The number of pulses for one revolution of the stepping motor: $360 / (\text{step angle} / \text{subdivision}) = 360 * \text{subdivision} / \text{step angle}$

- **reduction ratio:** that is, the transmission ratio of the deceleration device, and refers to the ratio of the instantaneous input speed to the output speed in the deceleration mechanism

- **thread lead:** it is the axial distance that any point on the thread moves along the same helix for one revolution. The pitch refers to the axial distance between the corresponding points of two adjacent teeth on the thread, and the code is P. The lead of a single-thread thread is equal to the pitch, and the lead of a multi-thread thread is equal to the number of heads (n) multiplied by the pitch, that is, the lead $S=nP$

If the mechanism has a lead (in mm) and a reducer, the pulse equivalent should be converted into mm as:

$$\text{Pulse equivalent} = \text{lead} / ((360 / \text{step angle} / \text{subdivision}) * \text{reduction ratio}) = (\text{lead} * \text{step angle}) / (360 * \text{reduction ratio} * \text{subdivision})$$

Notes:

a) The motor rotates a circle, and the reducer output rotates b circles, then the reduction ratio is "a: b".

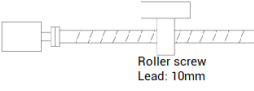
b) If the motor is directly connected to screw, the mechanical reduction ratio is 1:1

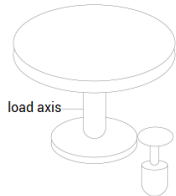
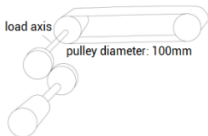
c) If the axis is a rotary shaft, the lead is 360°.

d) Lead: when the motor rotates one revolution, any point on its propulsion surface advances in the axial direction.

e) The stepper motor defaults to take one step and turn at an angle of 1.8°. Then 1.8° is the value of the step angle. Therefore, the stepper motor defaults to 360/1.8=200 pulses per revolution, but in some institutions, the precision requirements are high, and 1.8° is not suitable, so there is a subdivision setting on the driver. Like, there is a 5 subdivision on driver, that is, the angle is reduced by 5 times, 1.8/5=0.36, which means that the angle of motor every step is 0.36°, 360/0.36=1000 pulses per/circle.

The calculation methods of UNITS (pulse equivalent: ZAux_Direct_SetUnits) of several common mechanical transmissions are shown in the table below.

Mechanic Structure	Picture Example	Mechanic Specification	Encoder Resolution	UNITS
Roller screw		● Roller screw lead: 10mm ● Deceleration ratio: 1:1	● Drive electronic gear: 1:1 ● Encoder resolution: 10000 ● Pulses of one circle: 10000	Pulses in one circle / roller screw lead: UNITS = 10000 / 10 = 100, namely, one units mechanical structure moves 1mm.

Round table		● Rotate angle of one circle: 360 ● Deceleration ratio: 1:5	● Drive electronic gear: 1:1 ● Encoder resolution: 10000 ● Pulses of one circle: 10000/1	If round table and motor are connected directly, pulse of one degree of the motor, then $UNITS = 10000 / 360$. If round table is connected to servo motor through deceleration machine, the deceleration ratio is 1: 5, then motor rotates 5 degrees, and mechanical structure rotates 1 degree, so the $UNITS = 1000 / 360 / 5$.
Belt + pulley		● Pulley diameter: 100mm ● Pulley Circumference: 314mm ● Deceleration ratio: 1:1		Pulses / Circumference of one circle: $UNITS = 10000 / 314$

5.1.3. Routines

Set / Get Axis Basic Motion Parameters

```
// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s return code is %d\n", command, ret);
    }
}
```

```
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"127.0.0.1"; //controller IP is 192.168.0.11, simulator IP is 127.0.0.1
    ZMC_HANDLE handle = NULL;           //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("connect controller failed!\n");
        handle = NULL;
        getchar();
        return -1;
    }
    printf("connect controller succeeded!\n");

    ZAux_Direct_SetAtype(handle, 0, 1); //set axis type of axis 0 as 1
    ZAux_Direct_SetUnits(handle, 0, 100); //set pulse amount of axis 0 as 100
    ZAux_Direct_SetSpeed(handle, 0, 200); //set speed of axis 0 as 200units/s
    ZAux_Direct_SetAccel(handle, 0, 2000); //set acceleration of axis 0 as 2000units/s/s
    ZAux_Direct_SetDecel(handle, 0, 2000); //set deceleration of axis 0 as 2000units/s/s

    float dpos;
    int ifIdle;
    int status;
    float DposValue;

    int Atype;
    float Units;
    float Speed;
    float Accel;
    float Decel;
    ret = ZAux_Direct_GetAtype(handle, 0, &Atype); //get the axis type of axis 0
    commandCheckHandler("ZAux_Direct_GetAtype", ret); //if it executes successfully
    printf("axis type of axis 0 Atype = %d\n", Atype);

    ret = ZAux_Direct_GetUnits(handle, 0, &Units); //get the pulse amount of axis 0
    commandCheckHandler("ZAux_Direct_GetUnits", ret); //if it executes successfully
    printf("pulse amount of axis 0 Units = %f\n", Units);

    ret = ZAux_Direct_GetSpeed(handle, 0, &Speed); //get the speed of axis 0
    commandCheckHandler("ZAux_Direct_GetSpeed", ret); //if it executes successfully
    printf("speed of axis 0 Speed = %f\n", Speed);

    ret = ZAux_Direct_GetAccel(handle, 0, &Accel); //get the acceleration of axis 0
    commandCheckHandler("ZAux_Direct_GetAccel", ret); //if it executes successfully
    printf("acceleration of axis 0 Accel = %f\n", Accel);

    ret = ZAux_Direct_GetDecel(handle, 0, &Decel); //get the deceleration of axis 0
    commandCheckHandler("ZAux_Direct_GetDecel", ret); //if it executes successfully
    printf("deceleration of axis 0 Decel = %f\n", Decel);

    ret = ZAux_Direct_SetDpos(handle, 0, 0); //axis instruction position clearing
    commandCheckHandler("ZAux_Direct_SetDpos", ret); //if it executes successfully
    ret = ZAux_Direct_SetMpos(handle, 0, 0); //encoder feedback position clearing
    commandCheckHandler("ZAux_Direct_SetMpos", ret); //if it executes successfully

    ZAux_Trigger(handle); //oscilloscope trigger function
    ret = ZAux_Direct_Single_Move(handle, 0, 500);
    printf("drive axis 0 moves 500units\n");
    commandCheckHandler("ZAux_Direct_SetDpos", ret); //if it executes successfully
    Sleep(1000);

    ret = ZAux_Direct_GetDpos(handle, 0, &dpos); //get dpos
```

```

commandCheckHandler("ZAux_Direct_GetDpos", ret) ;//if it executes successfully
printf("after 1 second, axis 0 axis instruction position dpos = %f\n", dpos);

ret = ZAux_Direct_GetMpos(handle, 0, &DposValue);
commandCheckHandler("ZAux_Direct_GetMpos", ret) ;//if it executes successfully
printf("after 1 second, axis 0 encoder feedback position Mpos = %f\n", DposValue);

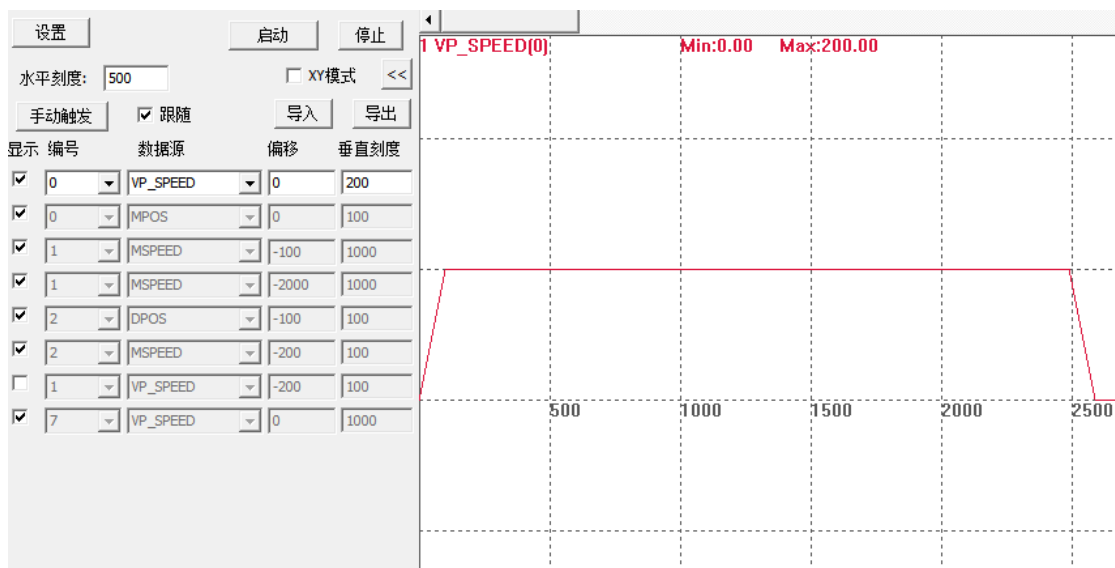
ret = ZAux_Direct_GetIfIdle(handle, 0, &ifIdle); //get whether the current axis is idle
commandCheckHandler("ZAux_Direct_GetIfIdle", ret) ;//if it executes successfully
printf("after 1 second, axis 0 motion state Idle: %s\n", ifIdle ? "Stop" : "Running");

ret = ZAux_Direct_GetAxisStatus(handle, 0, &status); //get the axis state
commandCheckHandler("ZAux_Direct_GetAxisStatus", ret) ;//if it executes successfully
printf("axis state AXISTATUS: %x h\n", status);

Sleep(2000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}

```

➤ Output Waveform Figure:



Set / Get Axis Basic Parameters (with initial speed & S curve)

```

// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {

```

```
        printf("%s return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"127.0.0.1";           //controller IP address
    ZMC_HANDLE handle = NULL;                     // link handle
    int ret = ZAux_OpenEth(ip_addr, &handle);     // connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        getchar();
        return -1;
    }
    printf("Success to connect controller!\n");

    ZAux_Direct_SetAtype(handle, 0, 1);           //set axis type of axis 0 as 1
    ZAux_Direct_SetUnits(handle, 0, 100);         //set pulse amount of axis 0 as 100
    ZAux_Direct_SetSpeed(handle, 0, 200);         //set speed of axis 0 as 200units/s
    ZAux_Direct_SetAccel(handle, 0, 2000);        //set acceleration of axis 0 as 2000units/s/s
    ZAux_Direct_SetDecel(handle, 0, 2000);        //set deceleration of axis 0 as 2000units/s/s
    ZAux_Direct_SetLspeed(handle, 0, 100);        //set starting speed of axis 0 as 20units/s
    ZAux_Direct_SetSramp(handle, 0, 200);         //set S curve time of axis 0 as 200ms
    float dpos;
    int ifIdle;
    int status;
    float DposValue;

    int Atype;
    float Units;
    float Speed;
    float Accel;
    float Decel;
    float Lspeed;
    float Sramp;

    ret = ZAux_Direct_GetAtype(handle, 0, &Atype); //get the axis type of axis 0
    commandCheckHandler("ZAux_Direct_GetAtype", ret); //if it executes successfully
    printf("axis type of axis 0 Atype = %d\n", Atype);

    ret = ZAux_Direct_GetUnits(handle, 0, &Units); //get pulse amount of axis 0
    commandCheckHandler("ZAux_Direct_GetUnits", ret); //if it executes successfully
    printf("pulse amount of axis 0 Units = %f\n", Units);

    ret = ZAux_Direct_GetSpeed(handle, 0, &Speed); //get the speed of axis 0
    commandCheckHandler("ZAux_Direct_GetSpeed", ret); //if it executes successfully
    printf("axis 0 speed Speed = %f\n", Speed);

    ret = ZAux_Direct_GetAccel(handle, 0, &Accel); //get acceleration of axis 0
    commandCheckHandler("ZAux_Direct_GetAccel", ret); //if it executes successfully
    printf("axis 0 acceleration Accel = %f\n", Accel);

    ret = ZAux_Direct_GetDecel(handle, 0, &Decel); //get deceleration of axis 0
    commandCheckHandler("ZAux_Direct_GetDecel", ret); //if it executes successfully
    printf("axis 0 deceleration Decel = %f\n", Decel);

    ret = ZAux_Direct_GetLspeed(handle, 0, &Lspeed); //get starting speed of axis 0
    commandCheckHandler("ZAux_Direct_GetLspeed", ret); //if it executes successfully
    printf("axis 0 starting speed Lspeed = %f\n", Lspeed);
}
```

```

ret = ZAux_Direct_GetSramp(handle, 0, &Sramp); //get S curve time of axis 0
commandCheckHandler("ZAux_Direct_GetDecel", ret); //if it executes successfully
printf("axis 0 S curve time Sramp = %f\n", Sramp);

ret = ZAux_Direct_SetDpos( handle, 0, 0); //axis instruction position clearing
commandCheckHandler("ZAux_Direct_SetDpos", ret); //if it executes successfully
ret = ZAux_Direct_SetMpos( handle, 0, 0); //encoder feedback position clearing
commandCheckHandler("ZAux_Direct_SetMpos", ret); //if it executes successfully

ZAux_Trigger(handle); //oscilloscope trigger function
ret = ZAux_Direct_Single_Move( handle, 0, 500);
printf("drive axis 0 moves 500units\n");
commandCheckHandler("ZAux_Direct_SetDpos", ret); //if it executes successfully
Sleep(1000);

ret = ZAux_Direct_GetDpos(handle, 0, &dpos); //get dpos
commandCheckHandler("ZAux_Direct_GetDpos", ret); //if it executes successfully
printf("after 1 second, axis 0 instruction position dpos = %f\n", dpos);

ret = ZAux_Direct_GetMpos(handle, 0, &DposValue);
commandCheckHandler("ZAux_Direct_GetMpos", ret); //if it executes successfully
printf("after 1 second, axis 0 encoder feedback position Mpos = %f\n", DposValue);

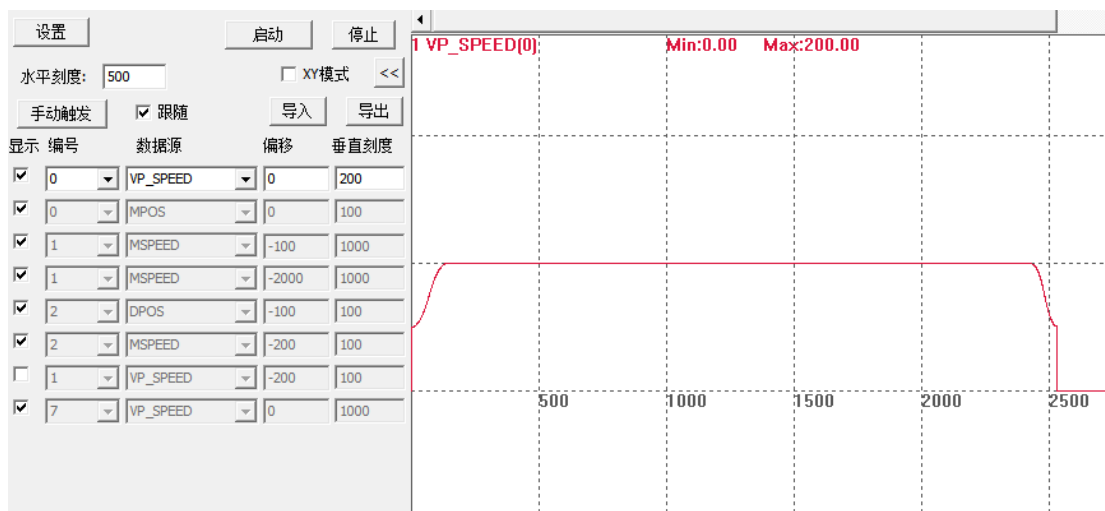
ret = ZAux_Direct_GetIfIdle(handle, 0, &ifIdle); //get whether the current axis is idle
commandCheckHandler("ZAux_Direct_GetIfIdle", ret); //if it executes successfully
printf("after 1 second, axis 0 motion state Idle: %s\n", ifIdle ? "Stop" : "Running");

ret = ZAux_Direct_GetAxisStatus(handle, 0, &status); //get axis state
commandCheckHandler("ZAux_Direct_GetAxisStatus", ret); //if it executes successfully
printf("axis state AXISSTATUS: %x h\n", status);

Sleep(2000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret); //if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}

```

➤ Output waveform figure:



5.2. Other Axis Parameters

5.2.1. Instruction List

Commands	Descriptions
ZAux_GetModbusDpos	Quick to read multiple DPOS of current axis (MODBUS)
ZAux_GetModbusMpos	Quick to read multiple MPOS of current axis (MODBUS)
ZAux_GetModbusCurSpeed	Quick to read several current axis speed while moving (MODBUS)
ZAux_Direct_SetAxisAddress	Set axis address
ZAux_Direct_GetAxisAddress	Read axis address
ZAux_Direct_SetInvertStep	Set pulse output mode
ZAux_Direct_GetInvertStep	Read pulse output mode
ZAux_Direct_SetMaxSpeed	Set pulse output max frequency
ZAux_Direct_GetMaxSpeed	Read pulse output max frequency
ZAux_Direct_GetMtype	Read current motion's command
ZAux_Direct_GetNtype	Read the motion command of next motion
ZAux_Direct_SetOffpos	Set offset (relative position)
ZAux_Direct_GetOffpos	Read offset (relative position)
ZAux_Direct_Defpos	Define one new absolute offset position (not recommend)
ZAux_Direct_GetFe	Read following error
ZAux_Direct_SetFeLimit	Set max allowable following error (reserved)
ZAux_Direct_GetFeLimit	Read max allowed following error (reserved)
ZAux_Direct_SetFeRange	Set the following error when alarm (reserved)
ZAux_Direct_GetFeRange	Read following error when alarm (reserved)
ZAux_Direct_SetAxisMode	Set hold parameters for "connect" motion

5.2.2. Principle & Description

For remain axis parameters, there are "parameter type" and "state type". Compared to basic parameters, it shows how to get related information under different situations. Like, [ZAux_GetModbusDpos](#), [ZAux_GetModbusMpos](#), [ZAux_GetModbusCurSpeed](#) are obtained in batches by obtaining the value of the register MODBUS_IEEE.

Axis Address

1. When ZCAN expands axes, there is an 8-bit DIP switch on the expansion board (hardware version should be V1.3 or above).

Please note ZCAN extended axes don't exceed 2 because it is limited by the bus bandwidth.

ZAux_Direct_SetAxisAddress must be used firstly to set the axis address, and then use **ZAux_Direct_SetAtype** to set the ZCAN extended axis type. After modification, you must use **ZAux_Direct_SetAtype** to set the axis type again. See example:

Bit 1-4	CAN address DIP, the combination value is 0-15.
Bit 5-6	CAN speed DIP, different combination values are with different speeds.
Bit 7	Reserved for special functions.
Bit 8	120ohm resistor DIP, dial ON to conduct the resistor.

Rule:

```
ZAux_Direct_SetAxisAddress (controller connection handle, axis No., (32*0)+ID)
//local axis interface 0 of the expansion board
```

```
ZAux_Direct_SetAxisAddress (controller connection handle, axis No., (32*1)+ID)
//local axis interface 1 of the expansion board
```

2. Map the axis No. of the bus driver, and map the connected drivers one by one according to the number.

The drive No. are arranged according to the wiring sequence, and the No. starts from 0 to the number of EtherCAT drives minus 1.

The drive No. is different from the device No. The device No. includes all the devices connected to the ECAT interface, but the drive No. only counts the connected drives.

The axis address must be set at first, and then the axis type (**ZAux_Direct_SetAxisAddress**) of the ECAT axis must be set. After modification, the axis type (**ZAux_Direct_SetAxisAddress**) must be reset. See example:

Bit 0-15	Drive No. + 1, 0-automatically specify
Bit 16-31	SLOT No. (when in multi-slot)

Rule:

```
ZAux_Direct_SetAxisAddress (controller connection handle, axis No., (slot No.<<16)
```

+ driver No. + 1)

3. Remap local pulse axis No., 4 series controllers support local pulse or encoder axis No. remapping, firmware version should be 160608 and above.

When remapping, firstly set the original pulse axis as a virtual axis. After modification, the axis type must be reset (ZAux_Direct_SetAtype). See example:

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

Rule:

Note: remap the axis No. of axis 0 of the physical interface of the controller to axis 3, then axis 3 is the target axis No., and axis 0 is the remapped axis No.

ZAux_Direct_SetParam (controller link handle, "ATYPE", remapped axis No.0, 0)

ZAux_Direct_SetParam (controller link handle, "ATYPE", target axis No.3, 0)

ZAux_Direct_SetAxisAddress (controller link handle, target axis No.3, (-1<<16) + remapped axis No.0)

ZAux_Direct_SetParam (controller link handle, "ATYPE", target axis No.3, 1 or 7)

Motion Type

For Zmotion motion control cards, each motion has different motion types. For example, a linear interpolation is one type of motion, a circular interpolation is another type of motion. At this time, motion type of the current motion can be obtained by calling the **ZAux_Direct_GetMtype**. And the motion type of the next motion can be obtained by calling the **ZAux_Direct_GetNtype**.

The movement type form is as follows:

MTYPE	Motion Instruction Type
0	IDLE (no motion)
1	MOVE (single-axis linear motion / linear interpolation motion)
2	MOVEABS (absolute single-axis linear motion / absolute linear interpolation motion)
3	MHELICAL (circle center spiral movement)
4	MOVECIRC (circular interpolation)
5	MOVEMODIFY (modify motion position)
6	MOVESP (single-axis linear motion of SP speed / linear interpolation)

	motion of SP speed)
7	MOVEABSSP (absolute single-axis linear motion of SP speed / absolute linear interpolation motion of SP speed)
8	MOVECIRCSP (circular interpolation of SP speed)
9	MHELICALSP (circle center spiral movement of SP speed)
10	FORWARD, VMOVE (1) (forward continuous motion)
11	REVERSE, VMOVE(-1) (reverse continuous motion)
12	DATUMING (in homing motion)
13	CAM (cam table motion)
14	FWD_JOG (mapping forward JOG motion)
15	REV_JOG (mapping reverse JOG motion)
20	CAMBOX (follow cam table motion)
21	CONNECT (synchronous motion)
22	MOVELINK (automatic cam motion)
23	CONNPATH (synchronous motion 2, vector type)
25	MOVESLINK (automatic cam motion 2)
26	MOVESPRIAL (involute arc)
27	MECLIPSE, MECLIPSEABS, MECLIPSESP, MECLIPSEABSSP (elliptical motion)
28	MOVE_OP/MOVE_OP2, MOVE_TABLE, MOVE_PWM, MOVE_TASKMOVE_PARA, MOVE_ASYNMOVE, MOVE_AOUT (buffer IO/ buffer register operation, etc.)
29	MOVE_DELAY, MOVE_WAIT, MOVE_SYNMOVE (buffer delay)
31	MSPHERICAL, MSPHERICALSP (space arc)
32	MOVE_PT (motion distance in unit time)
33	CONNFRAME (robotic arm inverse kinematic motion)
34	CONNREFRAME (robotic arm forward kinematic motion)

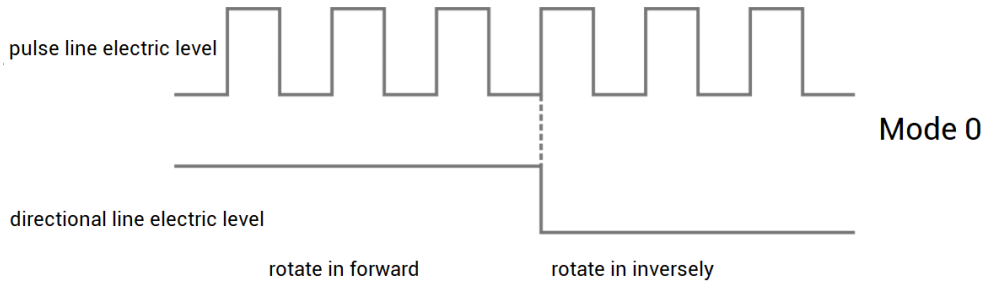
Pulse Mode

1) Pulse Directional Mode (PUL/DIR):

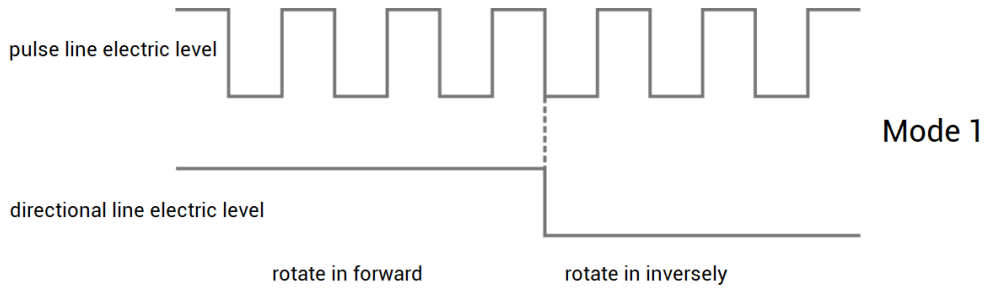
Output pulses through PUL+ and PUL- pulse lines, and the number of pulses corresponds to the running distance of the motor, and the pulse frequency corresponds to the running speed of the motor.

Output directional signals through DIR+ and DIR- direction lines, and the different levels of the signals correspond to different rotation directions of the motor. This mode is used most in the driver.

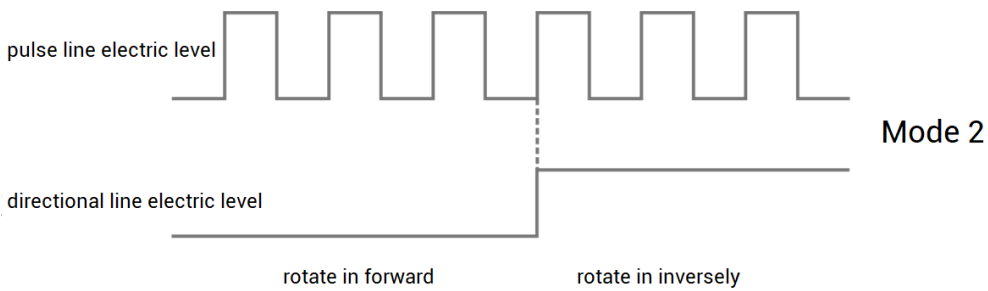
- ◆ Corresponding to **ZAux_Direct_SetInvertStep** (handle, axis No., 0) – mode 0



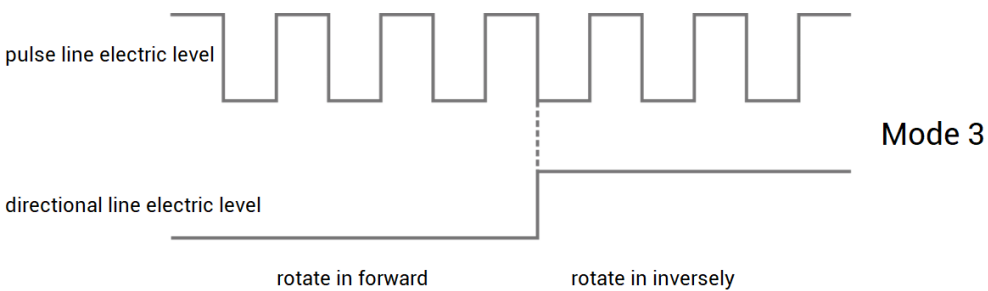
◆ Corresponding to **ZAux_Direct_SetInvertStep** (handle, axis No., 1) – mode 1



◆ Corresponding to **ZAux_Direct_SetInvertStep** (handle, axis No., 2) – mode 2



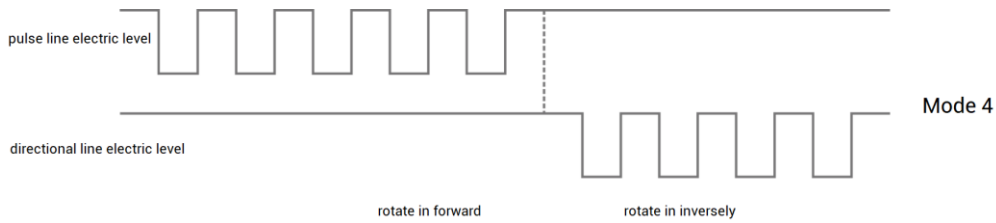
◆ Corresponding to **ZAux_Direct_SetInvertStep** (handle, axis No., 3) – mode 3



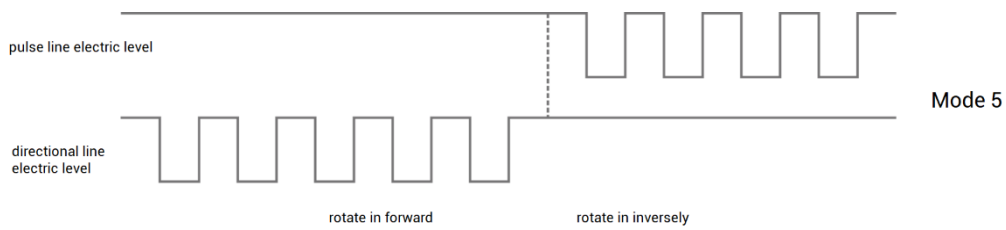
2) Dual Pulse Mode (CW/CCW):

Both 2 lines output pulse signals, CW is the pulse signal in the forward direction, and CCW is the pulse signal in the reverse direction. They are usually output in differential mode. The phase difference angle of the two signals is determined according to the phase lead or lag.

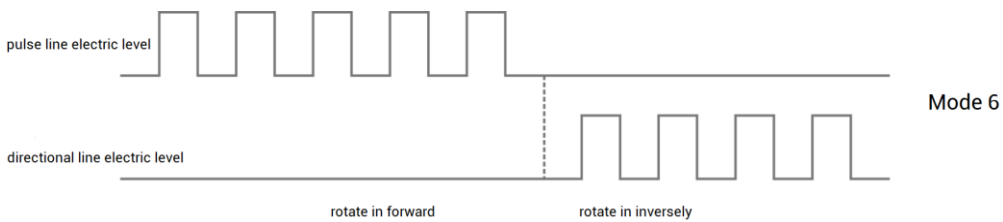
◆ Corresponding to **ZAux_Direct_SetInvertStep** (handle, axis No., 4) – mode 4



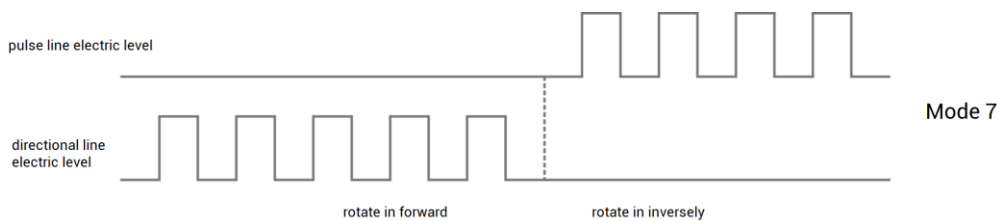
◆ Corresponding to **ZAux_Direct_SetInvertStep** (handle, axis No., 5) – mode 5



◆ Corresponding to **ZAux_Direct_SetInvertStep** (handle, axis No., 6) – mode 6



◆ Corresponding to **ZAux_Direct_SetInvertStep** (handle, axis No., 7) – mode 7

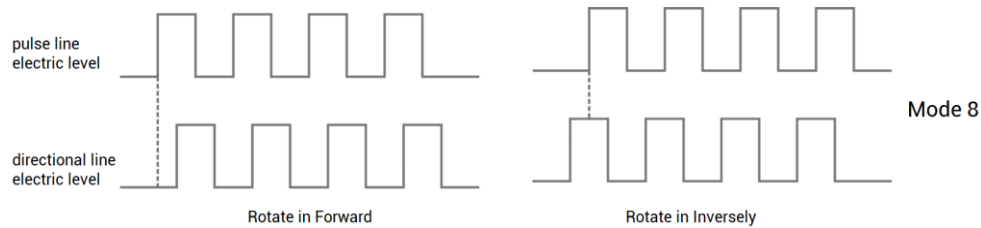


3) Quadrature Pulse Mode (A/B Phase):

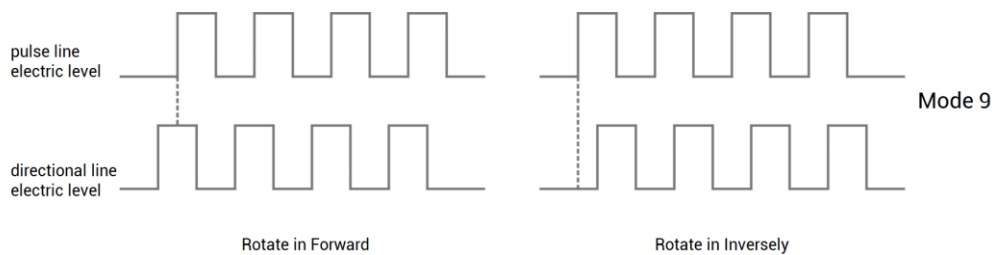
Orthogonal pulse refers to two identical pulse signals that are independent of each other. The positive direction pulse signal is generated before the negative direction pulse signal, and the phase difference between the two is 90 degrees. At this time, it is positive rotation, the negative direction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

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

- ◆ Corresponding to **ZAux_Direct_SetInvertStep** (handle, axis No., 8) – mode 8



- ◆ Corresponding to **ZAux_Direct_SetInvertStep** (handle, axis No., 9) – mode 9



5.2.3. Routines

The following routines will provide simple application examples of all the instructions mentioned in this chapter.

Remap Local Pulse Axis No.

In this routine, remap axis No. of axis 0 to axis 3.

```
// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
```

```
{
    char *ip_addr = (char *)"192.168.0.181"; //controller IP address, hardware wiring is needed to
    run the waveform in manual, simulator is useless.
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller! \n");
    int i=0;
    //axis address is cleared to 0, reset
    for (i=0;i<10;i++)
    {
        ret = ZAux_Direct_SetAtype(handle,i,0); //clear ATYPE to 0 before set axis address
        commandCheckHandler("ZAux_Direct_SetParam", ret); //if it executes successfully
        ret = ZAux_Direct_SetAxisAddress(handle,i,0); //set axis address Axis_Address to clear to 0
        commandCheckHandler("ZAux_Direct_SetParam", ret); //if it executes successfully
    }
    int axis[2]={0,3};
    for (i=0;i<2;i++)
    {
        ret = ZAux_Direct_SetParam(handle,"ATYPE",axis[i],1); //use SetParam to set axis list, ATYPE=1
        commandCheckHandler("ZAux_Direct_SetParam", ret); //if it executes successfully

        ret = ZAux_Direct_SetParam(handle,"DPOS",axis[i],0); //use SetParam to set axis list, clear DPOS
        commandCheckHandler("ZAux_Direct_SetParam", ret); //if it executes successfully

        ret = ZAux_Direct_SetParam(handle,"ACCEL",axis[i],1000); //use SetParam to set axis list, set
        acceleration as 1000
        commandCheckHandler("ZAux_Direct_SetParam", ret); //if it executes successfully

        ret = ZAux_Direct_SetParam(handle,"DECEL",axis[i],1000); //use SetParam to set axis list, set
        deceleration as 1000
        commandCheckHandler("ZAux_Direct_SetParam", ret); //if it executes successfully

        ret = ZAux_Direct_SetParam(handle,"SPEED",axis[i],1000); //use SetParam to set axis list, set
        speed as 1000
        commandCheckHandler("ZAux_Direct_SetParam", ret); //if it executes successfully
    }
    ret = ZAux_Direct_SetParam(handle,"DPOS",5,0); //use SetParam to set axis 5, clear dpos
    commandCheckHandler("ZAux_Direct_SetParam", ret); //if it executes successfully

    ret = ZAux_Direct_SetParam(handle,"ATYPE",5,6); //use SetParam to set axis 5, axis type is set
    as 6, encoder axis, setAtype function also can be used.
    commandCheckHandler("ZAux_Direct_SetParam", ret); //if it executes successfully

    float len[2]={100,100};
    int AxisAddress0;
    int AxisAddress3;
    ret = ZAux_Direct_GetAxisAddress(handle,0,&AxisAddress0); //get axis 0 address
    commandCheckHandler("ZAux_Direct_GetAxisAddress", ret); //if it executes successfully
    printf("axis 0 address before modification: %d\n",AxisAddress0);

    ret = ZAux_Direct_GetAxisAddress(handle,3,&AxisAddress3); //get axis 3 address
    commandCheckHandler("ZAux_Direct_GetAxisAddress", ret); //if it executes successfully
    printf("axis 3 address before modification: Axis_Address: %d\n",AxisAddress3);
    printf("*****\n");
    ZAux_Trigger(handle); //trigger the oscilloscope (RTSys SCOPE)
}
```

```

float IDLE;
ZAux_Direct_Single_Move(handle,0,100);//axis 0 moves 100
while (1)//wait for axis to complete the motion
{
    Sleep(100);
    ZAux_Direct_GetParam(handle,"IDLE",0,&IDLE);
    if (IDLE<0)break;
}
Sleep(400);
ZAux_Direct_Single_Move(handle,3,100); //axis 3 moves 100
while (1) //wait for axis 3 to complete the motion
{
    Sleep(100);
    ZAux_Direct_GetParam(handle,"IDLE",3,&IDLE);
    if (IDLE<0)break;
}

//{remap axis No. of physical interface axis 0 to axis 3, then axis 3 is the axis No. that is to be
modified, axis 0 is the axis No. that is to be remapped.

ret = ZAux_Direct_SetParam(handle,"ATYPE",0,0);//use SetParam method to remap axis 0, set
axis type as virtual axis, setAtype function also can be used.
commandCheckHandler("ZAux_Direct_SetParam", ret) ;//if it executes successfully

ret =ZAux_Direct_SetParam(handle,"ATYPE",3,0);//use SetParam method to set axis type of
target axis No.3 as virtual axis, setAtype function also can be used.
commandCheckHandler("ZAux_Direct_SetParam", ret) ;// if it executes successfully

ret =ZAux_Direct_SetAxisAddress(handle,3,(-1<<16)+0);//remap axis No. of axis 0 into axis 3
commandCheckHandler("ZAux_Direct_SetAxisAddress", ret) ;// if it executes successfully

ret =ZAux_Direct_SetParam(handle,"ATYPE",3,1);//set axis type of target axis No. as 1
commandCheckHandler("ZAux_Direct_SetParam", ret) ;// if it executes successfully

ret =ZAux_Direct_GetAxisAddress(handle,0,&AxisAddress0);//get axis 0 address
commandCheckHandler("ZAux_Direct_GetAxisAddress", ret) ;// if it executes successfully
printf("axis 0 address before modification Axis_Address: %d\n",AxisAddress0);

ret =ZAux_Direct_GetAxisAddress(handle,3,&AxisAddress3);//get axis 3 address
commandCheckHandler("ZAux_Direct_GetAxisAddress", ret) ;// if it executes successfully
printf("axis 0 address before modification Axis_Address: %d\n",AxisAddress3);

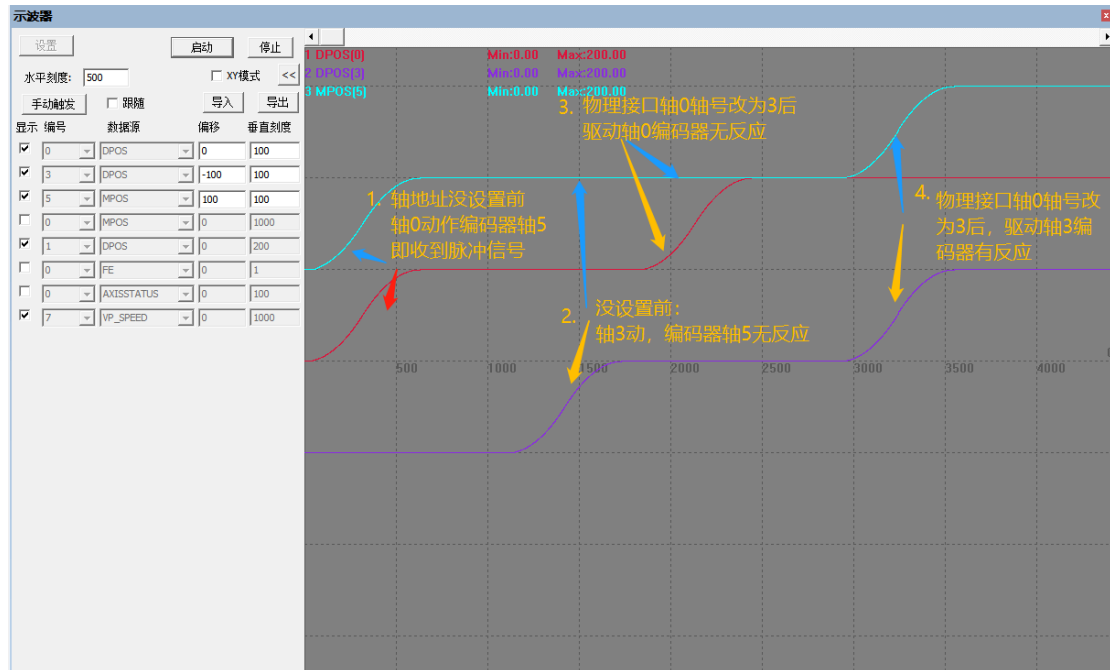
//for example: remap axis No. of physical interface axis 0 to axis 3, then axis 3 is the axis No.
that is to be modified, axis 0 is the axis No. that is to be remapped.

ZAux_Direct_Single_Move(handle,0,100);//axis 0 moves 100
while (1)
{
    Sleep(100);
    ZAux_Direct_GetParam(handle,"IDLE",0,&IDLE);
    if (IDLE<0)break;
}
Sleep(400);
ZAux_Direct_Single_Move(handle,3,100);//axis 3 moves 100

Sleep(2000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;// if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}

```

It can be seen the axis No. of axis 0 is changed into 3 from the below waveform.



Output:

```

控制器连接成功!
修改前轴0轴地址为Axis_Address: 0
修改前轴3轴地址为Axis_Address: 0
*****
修改后轴0轴地址为Axis_Address: 0
修改后轴3轴地址为Axis_Address: -65536
    
```

Set & Read Parameters

```

// test1.cpp : define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1"; //Controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAUX_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    
```

```
{
    printf("Fail to connect controller!\n");
    handle = NULL;
    Sleep(2000);
    return -1;
}
printf("Success to connect controller!\n");

ZAux_Direct_SetDpos(handle, 1, 100); //set axis 1 dpos=100
ZAux_Direct_SetDpos(handle, 2, 200); //set axis 2 dpos=200
ZAux_Direct_SetSpeed(handle, 0, 300); //set axis 0 speed as 300units/s
ZAux_Direct_SetSpeed(handle, 1, 400); //set axis 1 speed as 400units/s
ZAux_Direct_SetSpeed(handle, 2, 500); //set axis 2 speed as 500units/s

float Mpos[3]={0};
float Dpos[3]={0};
float Speed[3]={0};
int MaxSpeed;
int Invert_Step;
//{Invert_Step usage
//0-3 pulse directional mode, pulse line + direction line
//4-7 dual pulse method (or named as CW/CCW), forward pulse line + reverse pulse line
//8-9 AB output (some controllers need to be customized)
printf("*****\n");
ZAux_Direct_GetInvertStep(handle, 0, &Invert_Step); //get pulse mode of axis 0
commandCheckHandler("ZAux_GetModbusMpos", ret); //if it executes successfully
printf("axis 0 pulse mode Invert_Step(0): %d\n", Invert_Step);

ZAux_Direct_SetInvertStep(handle, 0, 1); //set pulse mode of axis 0 as 1
commandCheckHandler("ZAux_GetModbusMpos", ret); // if it executes successfully

ZAux_Direct_GetInvertStep(handle, 0, &Invert_Step); //get pulse mode of axis 0
commandCheckHandler("ZAux_GetModbusMpos", ret); // if it executes successfully
printf("axis 0 pulse mode Invert_Step(0)(after modification): %d\n", Invert_Step);

ZAux_Direct_SetInvertStep(handle, 0, 0); //set pulse mode of axis as 0 (default mode)
commandCheckHandler("ZAux_GetModbusMpos", ret); // if it executes successfully
printf("*****\n");
//Invert_Step usage}

//{simple application of max_speed (max speed can not exceed the value)
ret=ZAux_Direct_GetMaxSpeed(handle, 0, &MaxSpeed); //get max pulse output frequency of
axis 0
commandCheckHandler("ZAux_Direct_GetMaxSpeed", ret); // if it executes successfully
printf("max pulse output frequency of axis 0 Maxspeed: axis 0=%d\n", MaxSpeed);
ret=ZAux_Direct_SetMaxSpeed(handle, 0, MaxSpeed+100); //modify max pulse frequency
commandCheckHandler("ZAux_Direct_SetMaxSpeed", ret); // if it executes successfully

ret=ZAux_Direct_GetMaxSpeed(handle, 0, &MaxSpeed);
commandCheckHandler("ZAux_Direct_GetMaxSpeed", ret); //if it executes successfully
printf("max pulse output frequency of axis 0 Maxspeed (after modification): axis
0=%d\n", MaxSpeed);
ret=ZAux_Direct_SetMaxSpeed(handle, 0, MaxSpeed-100); //modify to max pulse frequency
commandCheckHandler("ZAux_Direct_SetMaxSpeed", ret); //if it executes successfully
//simple application of max_speed (max speed can not exceed this value)}

//use modbus method to get DPOS, MPOS, VP_SPEED
printf("*****\n");
ret=ZAux_GetModbusMpos( handle, 3, Mpos); //read MPOS of axis 0-axis 2
commandCheckHandler("ZAux_GetModbusMpos", ret); // if it executes successfully
printf("MPOS: axis 0=%f axis 1=%f axis 2=%f\n", Mpos[0], Mpos[1], Mpos[2]);

ret=ZAux_GetModbusDpos(handle, 3, Dpos);
commandCheckHandler("ZAux_GetModbusDpos", ret); // if it executes successfully
```



```
printf("DPOS: axis 0=%f axis 1=%f axis 2=%f\n",Dpos[0],Dpos[1],Dpos[2]);

ret=ZAux_GetModbusCurSpeed(handle, 3, Speed);//read VP_SPEED of axis 0-axis 2 (namely, in
motion, pulse speed sent by controller is not ideal)
commandCheckHandler("ZAux_GetModbusCurSpeed", ret) ;// if it executes successfully
printf("current planning speed VP_SPEED (before motion): axis 0=%f axis 1=%f axis
2=%f\n\n",Speed[0],Speed[1],Speed[2]);
//use modbus method to DPOS, MPOS, VP_SPEED}

//{{motions
ret=ZAux_Direct_Single_Move(handle,0,200);//drive axis 0 moves 200
commandCheckHandler("ZAux_Direct_Single_Move", ret) ;// if it executes successfully
ret=ZAux_Direct_Single_Move(handle,1,200);// drive axis 1 moves 200
commandCheckHandler("ZAux_Direct_Single_Move", ret) ;// if it executes successfully
ret=ZAux_Direct_Single_Move(handle,2,200);// drive axis 2 moves 200
commandCheckHandler("ZAux_Direct_Single_Move", ret) ;// if it executes successfully
Sleep(200);
//motion}

//{{read information again
ret=ZAux_GetModbusMpos( handle, 3, Mpos);//read MPOS of axis 0 – axis 2
commandCheckHandler("ZAux_GetModbusMpos", ret) ;// if it executes successfully
printf("MPOS (after motion: axis 0=%f axis 1=%f axis 2 = %f\n", Mpos[0], Mpos[1], Mpos[2]);

ret=ZAux_GetModbusDpos(handle, 3, Dpos);
commandCheckHandler("ZAux_GetModbusDpos", ret) ;// if it executes successfully
printf("DPOS (after motion): axis 0=%f axis 1=%f axis 2 = %f\n", Dpos[0], Dpos[1], Dpos[2]);

ret=ZAux_GetModbusCurSpeed(handle, 3, Speed);//read VP_SPEED of axis 0 – axis 3 (namely,
when in motion, the pulse speed sent by controller)
commandCheckHandler("ZAux_GetModbusCurSpeed", ret) ;// if it executes successfully
printf("current planning speed VP_SPEED (after motion): axis 0=%f axis 1=%f axis
2=%f\n",Speed[0],Speed[1],Speed[2]);
printf("*****\n");
//read information again}

//{{get the motion type
//value = 0, no motion. If it is >0, there is the motion, please refer to 5.2.2.2 for details
int Mtype=0;
int Ntype=0;
ret=ZAux_Direct_GetMtype(handle, 0,&Mtype);
commandCheckHandler("ZAux_Direct_GetMtype", ret) ;// if it executes successfully
printf("current motion type of axis 0 Mtype: axis 0 =%d \n",Mtype);

ret=ZAux_Direct_GetNtype(handle, 0,&Ntype);
commandCheckHandler("ZAux_Direct_GetNtype", ret) ;// if it executes successfully
printf("next motion type of axis 0 that is running Ntype: =%d \n",Ntype);

printf("*****\n");
//get the motion type}
Sleep(2000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;// if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}
```

Program result outputs:

```
控制器连接成功!
*****
轴0脉冲模式Invert_Step(0): 0
轴0脉冲模式Invert_Step(0) (修改后): 1
*****
轴0最大脉冲输出频率Maxspeed: 轴0=1000000
轴0最大脉冲输出频率Maxspeed (修改后): 轴0=1000100
*****
反馈位置MPOS: 轴0=-900.000000 轴1=100.000000 轴2=200.000000
规划位置DPOS: 轴0=-900.000000 轴1=100.000000 轴2=200.000000
当前运动的规划速度VP_SPEED (运动前): 轴0=0.000000 轴1=0.000000 轴2=0.000000

反馈位置MPOS (运动后): 轴0=-879.900024 轴1=120.099998 轴2=220.100006
规划位置DPOS (运动后): 轴0=-879.900024 轴1=120.099998 轴2=220.100006
当前运动的规划速度VP_SPEED (运动后): 轴0=202.000000 轴1=202.000000 轴2=202.000000
*****
轴0当前运动类型Mtype: 轴0=1
轴0当前正在运动的下一条运动指令类型Ntype: =0
*****
```

Set Axis Coordinates

```
// test1.cpp: define the entry point of control panel application program
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1"; //controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");

    ret = ZAux_Direct_SetAtype(handle, 0, 1); //set axis type
    commandCheckHandler("ZAux_Direct_SetAtype", ret); //if it executes successfully
    ret = ZAux_Direct_SetAccel(handle, 0, 500); //set axis acceleration
    commandCheckHandler("ZAux_Direct_SetAccel", ret); //if it executes successfully
    ret = ZAux_Direct_SetDecel(handle, 0, 500); //set axis deceleration
    commandCheckHandler("ZAux_Direct_SetDecel", ret); //if it executes successfully
    ret = ZAux_Direct_SetSpeed(handle, 0, 200); //set axis speed
    commandCheckHandler("ZAux_Direct_SetSpeed", ret); //if it executes successfully
    ret = ZAux_Direct_SetDpos(handle, 0, 0); //set axis DPOS clearing
    commandCheckHandler("ZAux_Direct_SetDpos", ret); //if it executes successfully
    float DPOS;
```

```
float OFFPOS;
ret =ZAux_Direct_GetDpos(handle,0,&DPOS);//get axis DPOS value
commandCheckHandler("ZAux_Direct_SetDpos", ret) ;// if it executes successfully
printf("before motion, axis 0 DPOS=%f\n",DPOS);

float IDLE;
ZAux_Direct_Single_Move(handle,0,200);//axis 0 moves 200
while (1)//wait for axis 0 to complete the motion
{
    Sleep(100);
    ZAux_Direct_GetParam(handle,"IDLE",0,&IDLE);
    if (IDLE<0)break;
}

ret =ZAux_Direct_GetDpos(handle,0,&DPOS);//get axis DPOS value
commandCheckHandler("ZAux_Direct_SetDpos", ret) ;// if it executes successfully
printf("after motion completed, axis 0 DPOS=%f\n",DPOS);

ret =ZAux_Direct_SetOffpos(handle,0,800);//offset relatively axis 0 coordinates
commandCheckHandler("ZAux_Direct_SetDpos", ret) ;// if it executes successfully

//ret =ZAux_Direct_GetOffpos(handle,0,&OFFPOS);
//commandCheckHandler("ZAux_Direct_SetDpos", ret) ;// if it executes successfully
//printf("axis 0 offset OFFPOS=%f\n",OFFPOS);

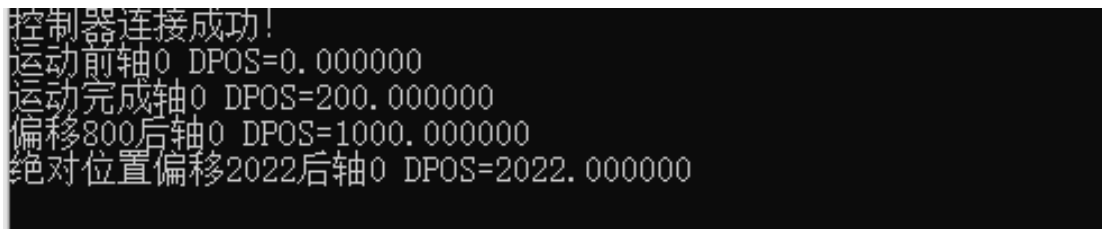
ret =ZAux_Direct_GetDpos(handle,0,&DPOS);// get axis DPOS value
commandCheckHandler("ZAux_Direct_SetDpos", ret) ;// if it executes successfully
printf("after offsets 800, axis 0 DPOS=%f\n",DPOS);

ret =ZAux_Direct_Defpos(handle,0,2022);//offset axis 0 coordinates as absolute position 2022.
commandCheckHandler("ZAux_Direct_SetDpos", ret) ;// if it executes successfully

ret =ZAux_Direct_GetDpos(handle,0,&DPOS);//get axis DPOS value
commandCheckHandler("ZAux_Direct_SetDpos", ret) ;//if it executes successfully
printf("after absolute position offsets 2022, the axis 0 DPOS=%f\n",DPOS);

Sleep(2000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;// if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}
```

Program result output: (only the coordinates are changed, the axis does not move)



```
控制器连接成功!
运动前轴0 DPOS=0.000000
运动完成轴0 DPOS=200.000000
偏移800后轴0 DPOS=1000.000000
绝对位置偏移2022后轴0 DPOS=2022.000000
```

Get Follow-up Errors

For the servo control system, under some abnormal conditions, the actual position

of the motor may differ greatly from the planned position. At this time, there are usually some dangerous situations, such as motor failure, encoder A and B phase signals are reversed or disconnected, too much mechanical friction or mechanical failure causes the motor to stall, etc. In order to detect these situations in time, enhance the safety of the system and prolong the service life of the equipment, the motion controller provides a safety protection mechanism that can obtain the following error exceeding the limit.

The following routine is to obtain the following error of axis 0.

```
// test1.cpp: define the entry point of control panel application program

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1"; //controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");

    ret = ZAux_Direct_SetAtype(handle, 0, 5); //set axis type as 5 (with encoder feedback), obvious
    following errors will be caused because of no encoder feedback wiring under simulator
    commandCheckHandler("ZAux_Direct_SetAtype", ret); // if it executes successfully
    ret = ZAux_Direct_SetAccel(handle, 0, 500); //set axis acceleration
    commandCheckHandler("ZAux_Direct_SetAccel", ret); // if it executes successfully
    ret = ZAux_Direct_SetDecel(handle, 0, 500); //set axis deceleration
    commandCheckHandler("ZAux_Direct_SetDecel", ret); // if it executes successfully
    ret = ZAux_Direct_SetSpeed(handle, 0, 200); //set axis speed
    commandCheckHandler("ZAux_Direct_SetSpeed", ret); // if it executes successfully
    ret = ZAux_Direct_SetDpos(handle, 0, 0); // set axis DPOS clearing
    commandCheckHandler("ZAux_Direct_SetDpos", ret); // if it executes successfully

    float IDLE;
    ZAux_Direct_Single_Move(handle, 0, 200); //axis 0 moves 200
    while (1) //wait for axis 0 to complete the motion
    {
        Sleep(100);
        ZAux_Direct_GetParam(handle, "IDLE", 0, &IDLE);
        if (IDLE < 0) break;
    }
}
```

```
float FE;
float DPOS;
float MPOS;
ret = ZAux_Direct_GetDpos(handle,0,&DPOS);//clear DPOS
commandCheckHandler("ZAux_Direct_GetDpos", ret) ;// if it executes successfully
printf("current DPOS=%f\n",DPOS);
ret = ZAux_Direct_GetMpos(handle,0,&MPOS);//clear MPOS
commandCheckHandler("ZAux_Direct_GetMpos", ret) ;// if it executes successfully
printf("current MPOS=%f\n",MPOS
);

ret = ZAux_Direct_GetFe(handle,0,&FE);//clear FE
commandCheckHandler("ZAux_Direct_GetFe", ret) ;// if it executes successfully
printf("current FE=%f\n",FE);

Sleep(2000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;// if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}
```

5.3. Axis Parameters Instruction List

5.3.1. Instruction List

Commands	Descriptions
ZAux_Direct_GetAllAxisPara	Read all axes parameters' state (in floating)
ZAux_Direct_GetAllAxisInfo	Read all axes parameters' state (in integer)
ZAux_Direct_SetParam	Set general axis parameters
ZAux_Direct_GetParam	Read general axis parameters

5.3.2. Detailed Description

Motion Control Parameters List:

Instruction	Description	Read & Write	No.
Atype	Axis type	R & W	0
Units	Pulse amount	R & W	1
Accel	Acceleration	R & W	2
Decel	Deceleration	R & W	3
Speed	Running speed	R & W	4
Creep	Crawling speed	R & W	5
Lspeed	Initial speed	R & W	6
Merge	Continuous interpolation switch	R & W	7

Sramp	The curve time of accel and decel	R & W	8
Dpos	Axis planning (demand) position	R & W	9
Mpos	Encoder feedback (measurement) position	R & W	10
Endmove	Current motion target position	RO	11
Fs_Limit	Forward soft position limit setting	R & W	12
Rs_Limit	Reverse soft position limit setting	R & W	13
Datum_In	Origin IN map	R & W	14
Fwd_In	Map forward hard position limit input	R & W	15
Rev_In	Map reverse hard position limit input	R & W	16
Idle	Motion state	RO	17
Loaded	Whether the buffer is in empty state	RO	18
Mspeed	Actual encoder feedback speed	RO	19
Mtype	Current motion type	RO	20
Ntype	Next motion type	RO	21
Remain	Current motion remain distance	RO	22
Vp_Speed	Current motion speed	RO	23
Axisstatus	Axis state	RO	24
Move_Mark	Motion mark	R & W	25
Move_Curmark	Current motion mark	RO	26
Vector_Buffered	Current motion remain distance	RO	27
Axis_StopReason	Axis stop reasons	RO	28
Moves_Buffered	Current motion buffers	RO	29
Axis_Address	Axis address	R & W	30
Axis_Enable	Single-axis enable	R & W	31
Force_Speed	SP motion speed	R & W	32
Startmove_Speed	SP motion starting speed	R & W	33
Endmove_Speed	SP motion ending speed	R & W	34
Fastdec	Fast deceleration	R & W	35
Addax_Axis	Superposition axis No.	RO	36
Link_Axis	Connecting axis No.	RO	37
Corner_Mode	Corner mode	R & W	38
Decel_Angle	Corner deceleration start angle	R & W	39
Stop_Angle	Corner deceleration end angle	R & W	40
Full_Sp_Radius	Speed limit radius	R & W	41
Splimit_Radius	Speed limit value	R & W	42
Zsmooth	Chamfer radius	R & W	43
Vector_Moved	Currently moved distance	R & W	44
Endmove_Buffer	The end position of buffer	RO	45
Homewait	Homing reverse delay	R & W	46
Fast_Jog	Map jog input	R & W	47
Fwd_Jog	Map forward Jog input	R & W	48
Rev_Jog	Map reverse Jog input	R & W	49
Jogspeed	Map Jog speed	R & W	50
Fhold_in	Map holding input	R & W	51

Fhspeed	Holding speed	R & W	52
Encoder	Encoder original value	RO	53
Encoder_Status	Encoder state	RO	54
PP_Step	Encoder inner ratio	R & W	55
Reg_Inputs	Latch input latch	R & W	56
Mark	Latch trigger	RO	57
MarkB	Latch 2 trigger	RO	58
MarkC	Latch 3 trigger	RO	59
MarkD	Latch 4 trigger	RO	60
Reg_Pos	Latch position	RO	61
Reg_PosB	Latch 2 position	RO	62
Reg_PosC	Latch 3 position	RO	63
Reg_PosD	Latch 4 position	RO	64
Alm_In	Map alarm input	R & W	65
Rep_Option	Coordinates loop mode	R & W	66
Rep_Dist	Coordinates loop position	R & W	67
Invert_Step	Pulse mode configuration	R & W	68
Max_speed	Pulse frequency limit	R & W	69
Axis_Zset	Precision output configuration	R & W	70
Dac	Bus axis analog control	R & W	71
ErrorMask	Operation when in error	R & W	72

The state of the above motion parameters can be set or read through the ZAux_Direct_SetParam and ZAux_Direct_GetParam functions, and the string commands are not case-sensitive.

5.3.3. Routines

Set & Get Basic Parameters by Character String

```
// test1.cpp: define the entry point of control panel application program
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
```

```
{
    char *ip_addr = (char *)"127.0.0.1";           //controller IP address
    ZMC_HANDLE handle = NULL;                     //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle);     //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        getchar();
        return -1;
    }
    printf("Success to connect controller!\n");

    ZAux_Direct_SetParam(handle, "ATYPE", 0, 1);   //set axis type of axis 0 as 1
    ZAux_Direct_SetParam(handle, "Units", 0, 100); //set pulse amount of axis 0 as 100
    ZAux_Direct_SetParam(handle, "SPEED", 0, 200); //set speed of axis 0 as 200units/s
    ZAux_Direct_SetParam(handle, "ACCEL", 0, 2000); //set acceleration of axis 0 as 2000units/s/s,
    ZAux_Direct_SetParam(handle, "DECEL", 0, 2000); //set deceleration of axis 0 as 2000units/s/s,

    float dpos;
    float ifIdle;
    float status;
    float DposValue;

    float Atype;
    float Units;
    float Speed;
    float Accel;
    float Decel;

    ret = ZAux_Direct_GetParam(handle, "ATYPE", 0, &Atype); //get axis 0 type, the type is 1
    commandCheckHandler("ZAux_Direct_GetAtype", ret); //if it exceeds successfully
    printf("axis type of axis 0 Atype = %d\n", Atype);

    ret = ZAux_Direct_GetParam(handle, "Units", 0, &Units); //get axis 0 pulse amount, it is 100
    commandCheckHandler("ZAux_Direct_GetUnits", ret); //if it exceeds successfully
    printf("pulse amount of axis 0 Units = %f\n", Units);

    ret = ZAux_Direct_GetParam(handle, "Speed", 0, &Speed); //get axis 0 speed, 200units/s
    commandCheckHandler("ZAux_Direct_GetSpeed", ret); //if it exceeds successfully
    printf("speed of axis 0 Speed = %f\n", Speed);

    ret = ZAux_Direct_GetParam(handle, "Accel", 0, &Accel); //get axis 0 acceleration, 2000units/s/s
    commandCheckHandler("ZAux_Direct_GetAccel", ret); //if it exceeds successfully
    printf("acceleration of axis 0 Accel = %f\n", Accel);

    ret = ZAux_Direct_GetParam(handle, "Decel", 0, &Decel); //get axis 0 deceleration, 2000units/s/s
    commandCheckHandler("ZAux_Direct_GetDecel", ret); //if it exceeds successfully
    printf("deceleration of axis 0 Decel = %f\n", Decel);

    ret = ZAux_Direct_SetParam(handle, "DPOS", 0, 0); //axis DPOS is cleared
    commandCheckHandler("ZAux_Direct_SetDpos", ret); //if it exceeds successfully

    ret = ZAux_Direct_SetParam(handle, "MPOS", 0, 0); //MPOS is cleared
    commandCheckHandler("ZAux_Direct_SetMpos", ret); //if it exceeds successfully

    ZAux_Trigger(handle); //oscilloscope trigger function
    ret = ZAux_Direct_Single_Move(handle, 0, 500);
    printf("drive axis 0 to move 500units\n");
    commandCheckHandler("ZAux_Direct_Single_Move", ret); //if it exceeds successfully
    Sleep(1000);

    ret = ZAux_Direct_GetParam(handle, "DPOS", 0, &dpos); //get dpos
    commandCheckHandler("ZAux_Direct_GetDpos", ret); //judge whether the instruction is
```



```
executed successfully
printf("after 1s, axis 0 instruction demand position dpos = %f\n", dpos);

ret = ZAux_Direct_GetParam(handle,"MPOS", 0, &DposValue);
commandCheckHandler("ZAux_Direct_GetMpos", ret) ;//judge whether the instruction is
executed successfully
printf("after 1s, axis 0 encoder measurement position Mpos = %f\n", DposValue);

ret = ZAux_Direct_GetParam(handle,"Idle", 0, &ifIdle); //get whether the current axis is idle
commandCheckHandler("ZAux_Direct_GetIfIdle", ret) ;//judge whether the instruction is
executed successfully
printf("after 1s, axis 0 motion state Idle: %s\n", ifIdle ? "Stop" : "Running");

ret = ZAux_Direct_GetParam(handle,"AxisStatus", 0, &status); //get axis state
commandCheckHandler("ZAux_Direct_GetAxisStatus", ret) ;//judge whether the instruction is
executed successfully
printf("axis state AXISSTATUS: %x h\n", status);

Sleep(2000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;//judge whether the instruction is executed
successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}
```

Chapter VI How to Control Basic Motions?

6.1. Motion Buffer

6.1.1. Instruction List

Commands	Descriptions
ZAux_Direct_GetEndMove	Read axis motion's end position
ZAux_Direct_GetEndMoveBuffer	Read the end position of current motion and buffer motion
ZAux_Direct_GetLoaded	Read whether there is buffer except the current motion
ZAux_Direct_GetMovesBuffered	Read how many motions buffered
ZAux_Direct_GetMoveCurmark	Read current motion's mark No. "MOVE_MARK"
ZAux_Direct_SetMoveMark	Set motion's mark No. "MOVE_MARK"
ZAux_Direct_GetRemain	Read remaining distance of current motion
ZAux_Direct_GetRemain_LineBuffer	Read axis remaining linear buffer, calculate according to linear segments.
ZAux_Direct_GetRemain_Buffer	Read axis remain linear buffer, calculate according to the complex space arc.
ZAux_Direct_GetVectorBuffered	Read uncompleted distance of current axis and buffer motion
ZAux_Direct_MoveOp	Add "OP" command in motion buffer
ZAux_Direct_MoveOpAdist	Set distance (early to output) in buffer
ZAux_Direct_MoveOpMulti	Add "continuous output" command in motion buffer
ZAux_Direct_MoveOpDelay	Early / delay to open OP port
ZAux_Direct_MoveOp2	Add "OP" command in motion buffer, and make it invert the state after specified time
ZAux_Direct_MoveAout	Add "AOUT" command in motion buffer
ZAux_Direct_MoveDelay	Add "delay" command in motion buffer
ZAux_Direct_MoveWait	Add "one changeable delay" in BASE axis / axes motion buffer
ZAux_Direct_MoveTask	Add "one TASK" in BASE axis / axes motion buffer
ZAux_Direct_MovePara	Modify parameters while moving
ZAux_Direct_MovePwm	Modify PWM while moving
ZAux_Direct_MoveSynmove	Synchronize with other axes motions while moving
ZAux_Direct_MoveASynmove	Trigger other axes motions while moving

ZAux_Direct_MoveTable	Modify TABLE while moving
ZAux_Direct_MoveCancel	Cancel other axes motions while moving
ZAux_Direct_MoveModify	Modify the end position while moving
ZAux_Direct_MoveLimit	Add speed limit at the motion end, used to do forcible corner deceleration.
ZAux_Direct_MoveModify2	Modify the end position while moving 2
ZAux_Execute_GetMoveTime	Read corresponding motion remain time
ZAux_Execute_GetPreCalTime	Estimate motion time of the point, in ms

6.1.2. Principle & Description

- **Functions of Motion Control Commands:**

Achieve point to point, linear interpolation, circular interpolation, gear motion, etc.

And set axis parameters before motion.

- **Types of Motion Control Commands:**

- a) single axis motion instruction: only can operate one axis at one time, and it operates master axis by default
- b) multi-axis motion instruction: can operate multi axes at one time.

- **Instruction Period of Motion Control Commands:**

get – analyze – execute the instruction, then repeat the process.

Zmotion motion controller contain multilevel **motion buffer**, when there is instruction in process, then new called motion instructions will enter buffer in order to avoid program blocking. When the multilevel motion buffer is full, and if new motion instructions are still called in program, same, program will be blocked until there is space in buffer again.

- **Basic Motion Commands**

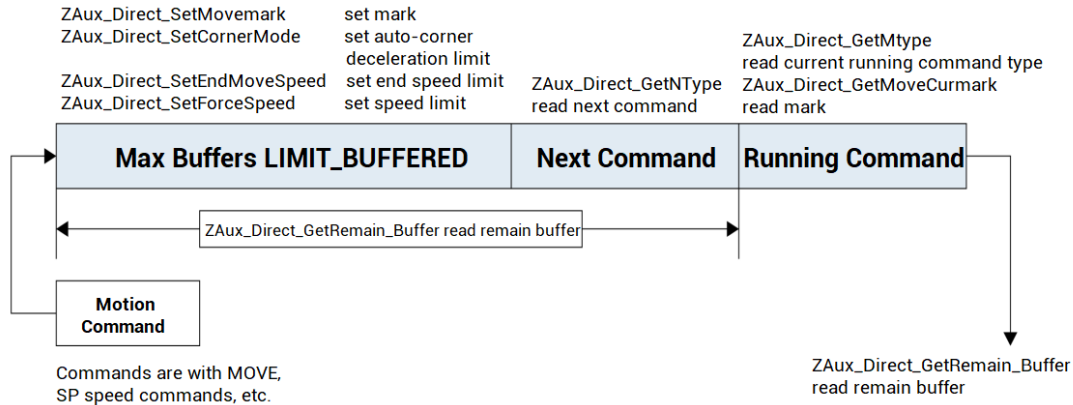
linear, circular, and all kinds of interpolation motions and some instructions with prefix "MOVE".

- **SP Motion Commands**

When using instructions with SP, ZAux_Direct_SetForceSpeed, ZAux_Direct_SetEndMoveSpeed and ZAux_Direct_SetStartMoveSpeedFORCE_SPEED will follow SP type motion instructions to enter motion buffer.

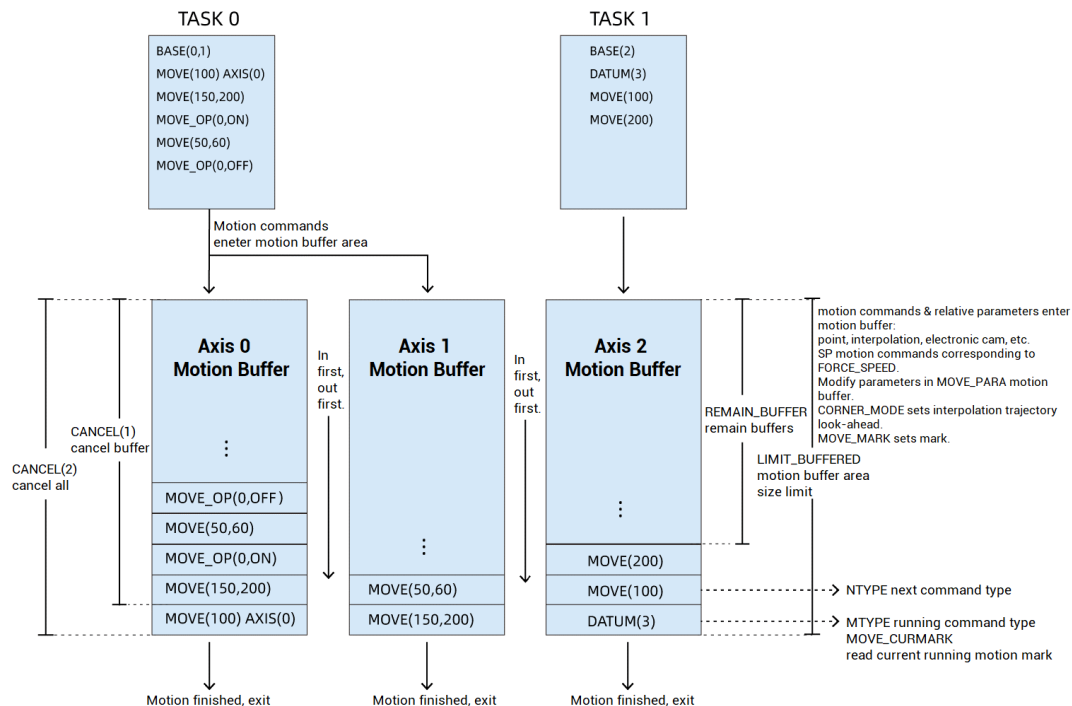
Also, ZAux_Direct_MoveOp instruction can be used to enter IO operation command into buffer, in this way, IO output is added automatically between motion instructions. And use ZAux_Direct_MoveDelay to enter delay command into buffer, then, it will

automatically delay between motion instructions.



Each axis has independent motion buffer, no conflict between each other, such as [ZMC4 series](#), its motion buffer of each axis is 4096. See related hardware manual for details of motion buffer.

Take Zmotion [ZMC406-V2](#) controller as the example, motion buffer execution system is shown below.



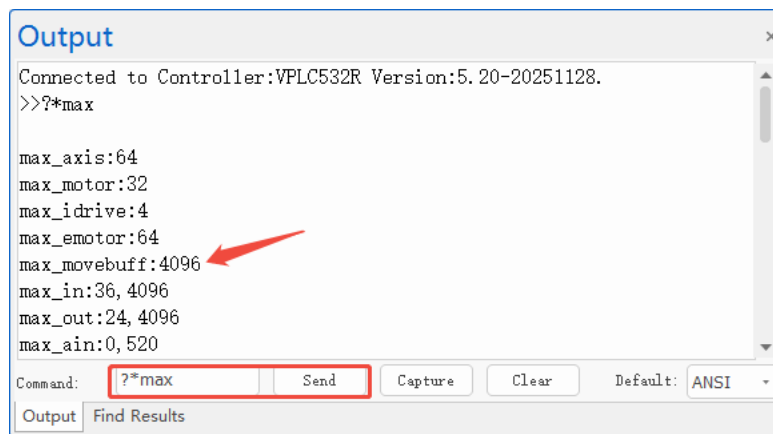
Motion Buffer Commands: `ZAux_Direct_MoveSp`, `ZAux_Direct_MoveAbsSp`, `ZAux_Direct_MoveCircSpSP`, etc., namely, commands are with SP function suffix.

Invalid Buffer Motion Commands: `ZAux_Direct_MoveModify`, etc.

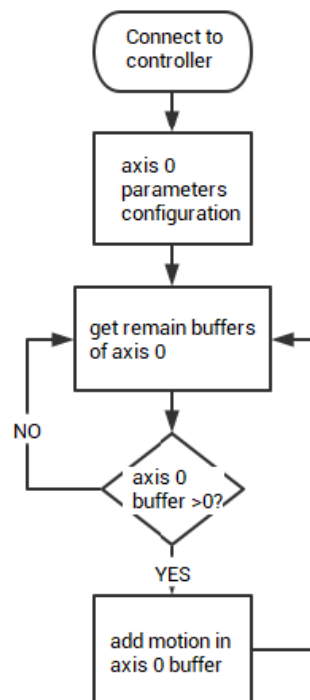
6.1.3. Routines

Get Remain Motion Buffers

In the RTSys development environment, you can check the maximum motion buffer size by entering “?*max” in the output window on the command line. For example, the VPLC532R motion buffer size is max_movebuff:4096.



This routine obtains the remaining buffers of axis 0, and adds the motion into the buffer according to whether the remaining buffer is greater than 0.



```
// test1.cpp: define the entry point of control panel application program
//
```

```
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1"; //controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        getchar();
        return -1;
    }
    printf("Success to connect controller!\n");

    ZAux_Direct_SetAtype(handle, 0, 1); //set axis type of axis 0 as 1
    ZAux_Direct_SetUnits(handle, 0, 100); //set pulse amount of axis 0 as 100
    ZAux_Direct_SetSpeed(handle, 0, 200); //set speed of axis 0 as 200units/s
    ZAux_Direct_SetAccel(handle, 0, 2000); //set acceleration of axis 0 as 2000units/s/s
    ZAux_Direct_SetDecel(handle, 0, 2000); //set deceleration of axis 0 as 2000units/s/s

    int MoveTime = 0;
    int GetValue;

    ret = ZAux_Direct_SetDpos( handle, 0 , 0); //clear DPOS
    commandCheckHandler("ZAux_Direct_SetDpos", ret);

    while (MoveTime < 100)
    {
        ret = ZAux_Direct_GetRemain_Buffer( handle, 0, &GetValue); //get remain buffers of axis 0
        commandCheckHandler("ZAux_Direct_GetRemain_Buffer", ret);
        printf("Remain_Buffer = %d \n", GetValue);

        if (GetValue > 0) //if the number of axis 0 buffers is > 0
        {
            ret = ZAux_Direct_Single_Move( handle, 0, 10); //add the motion into axis 0 buffer
            commandCheckHandler("ZAux_Direct_Single_Move", ret);
            MoveTime += 1;
        }
    }

    Sleep(2000);
    ret = ZAux_Close(handle); //close the connection
    commandCheckHandler("ZAux_Close", ret) ; //if it executes successfully
    printf("connection closed!\n");
    handle = NULL;
    return 0;
}
```

Trigger Output in Motion (move_op2)

For example, suppose that in a certain actuator, 100 pulses are required to realize the movement of axis 0 by 1mm, the total movement of axis 0 is 2 meters, and the OUT0 is triggered every 100mm for dispensing.

```
// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1"; //controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        getchar();
        return -1;
    }
    printf("Success to connect controller!\n");

    ZAux_Direct_SetAtype(handle, 0, 1); //set axis type of axis 0 as 4
    ZAux_Direct_SetUnits(handle, 0, 100); //set pulse amount of axis 0 as 100, relate to 1mm

    ZAux_Direct_SetSpeed(handle, 0, 200); //set speed of axis 0 as 200units/s
    ZAux_Direct_SetAccel(handle, 0, 2000); //set acceleration of axis 0 as 2000units/s/s
    ZAux_Direct_SetDecel(handle, 0, 2000); //set deceleration of axis 0 as 2000units/s/s

    int MoveTime = 0;
    int GetValue;

    ret = ZAux_Direct_SetDpos( handle, 0 , 0); //clear DPOS position to 0
    commandCheckHandler("ZAux_Direct_SetDpos",ret);
    ret = ZAux_Direct_SetMpos( handle, 0 , 0); // clear MPOS position to 0
    commandCheckHandler("ZAux_Direct_SetMpos",ret);
    ret = ZAux_Direct_SetMerge( handle, 0 , 1); //open continuous interpolation
    commandCheckHandler("ZAux_Direct_SetMerge",ret);

    ret = ZAux_Trigger(handle);
    commandCheckHandler("ZAux_Trigger",ret);

    for (int i=0;i<20;i++)
    {
```

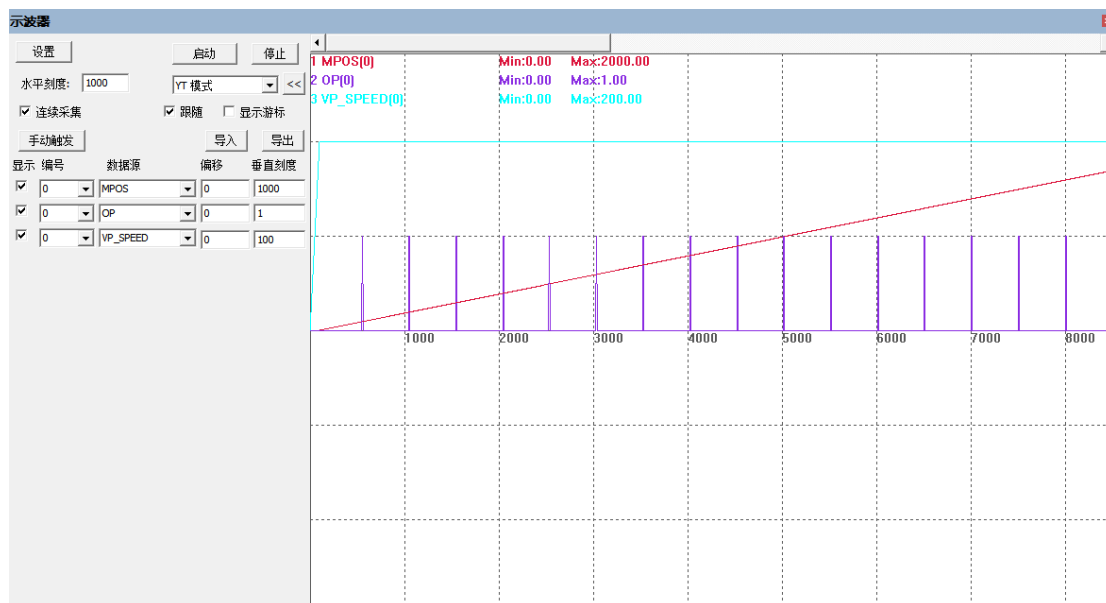
```

    ret = ZAux_Direct_Single_Move(handle, 0, 100);
    commandCheckHandler("ZAux_Direct_Single_Move", ret);
    ret = ZAux_Direct_MoveOp2(handle, 0, 0, 1, 10);
    commandCheckHandler("ZAux_Direct_MoveOp2", ret);
}

Sleep(2000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}

```

It can be observed from the oscilloscope shown in the figure below that, the OUT port 0 outputs once every 100 pulses position when the speed is the same.



6.2. Single-axis Motion

6.2.1. Instruction List

Commands	Descriptions
ZAux_Direct_Single_Move	Single-axis relative motion
ZAux_Direct_Single_MoveAbs	Single-axis absolute motion
ZAux_Direct_Single_Vmove	Single-axis continuous motion
ZAux_Direct_SetJogSpeed	Set speed when JOG
ZAux_Direct_GetJogSpeed	Read speed when JOG
ZAux_Direct_GetFastJog	Read JOG speed IN: when there is IN, speed is given by SPEED; when there is no IN, speed is

	given by JOGSPEED
ZAux_Direct_SetFastJog	Set JOG speed IN: when there is IN, speed is given by SPEED; when there is no IN, speed is given by JOGSPEED
ZAux_Direct_SetFwdJog	Set forward JOG IN No., -1 = invalid
ZAux_Direct_GetFwdJog	Read forward JOG IN No., -1 = invalid
ZAux_Direct_SetRevJog	Set inverse JOG IN No., -1 = invalid
ZAux_Direct_GetRevJog	Read inverse JOG IN No., -1 = invalid

6.2.2. Principle & Description

Each axis has its independent parameters, and parameters are global, which means it will take effect immediately after modification, like, speed, acceleration, then dynamic speed can be achieved.

When `ZAux_Direct_Single_Cancel` is called to stop the motion, it needs to decelerate by `ZAux_Direct_SetFastDec` (default: use deceleration value). To achieve an emergency stop, the fast deceleration can be set to the maximum allowable value of the drive.

Single-axis continuous movement (`ZAux_Direct_Single_Vmove`) moves in one direction continuously. If the previous single-axis continuous motion is not stopped, the new single-axis continuous motion command will automatically replace the previous single-axis continuous motion and modify the direction, so there is no need to call `ZAux_Direct_Single_Cancel` to stop the previous single-axis continuous motion.

6.2.3. Routines

Single-Axis Point Motion

In initialization, set basic parameters at first, like, `DPOS = 0`, pulse amount (UNITS) = 100, speed (SPEED) = 200 units/s, acceleration (ACCEL) = 2000units/s/s, deceleration (DECEL) = 2000units/s/s, S curve acceleration and deceleration time is 200ms, and make it do relative motion, the distance is 800units, then do absolute motion, to absolute position 100units.

PS: the acceleration and deceleration time of the S-shaped curve is the change time of the motion acceleration and deceleration, and the trajectory is S-shaped, not

trapezoidal (T-shaped).



```

// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1"; //controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        getchar();
        return -1;
    }
    printf("Success to connect controller!\n");

    ZAux_Direct_SetAtype(handle, 0, 1); //set axis type of axis 0 as 1
    ZAux_Direct_SetUnits(handle, 0, 100); //set pulse amount of axis 0 as 100
    ZAux_Direct_SetSpeed(handle, 0, 200); //set speed of axis 0 as 200units/s
    ZAux_Direct_SetAccel(handle, 0, 2000); //set acceleration of axis 0 as 2000units/s/s
    ZAux_Direct_SetDecel(handle, 0, 2000); //set deceleration of axis 0 as 2000units/s/s
    ZAux_Direct_SetSramp(handle, 0, 200); //set S curve of axis 0 as 200ms

    ret = ZAux_Direct_SetDpos(handle, 0, 0); //clear DPOS
    commandCheckHandler("ZAux_Direct_SetDpos", ret); //if it executes successfully
    ret = ZAux_Direct_SetMpos(handle, 0, 0); //clear MPOS
    commandCheckHandler("ZAux_Direct_SetMpos", ret); //if it executes successfully

    ZAux_Trigger(handle); //trigger oscilloscope to capture data
    ret = ZAux_Direct_Single_Move(handle, 0, 800);
    commandCheckHandler("ZAux_Direct_SetDpos", ret); //if it executes successfully

    float IDLE;
    while (1) //wait for axis 0 to complete the motion
    {
        Sleep(100);
        ZAux_Direct_GetParam(handle, "IDLE", 0, &IDLE);
        if (IDLE < 0) break;
    }
    ret = ZAux_Direct_Single_MoveAbs(handle, 0, 100);
  
```

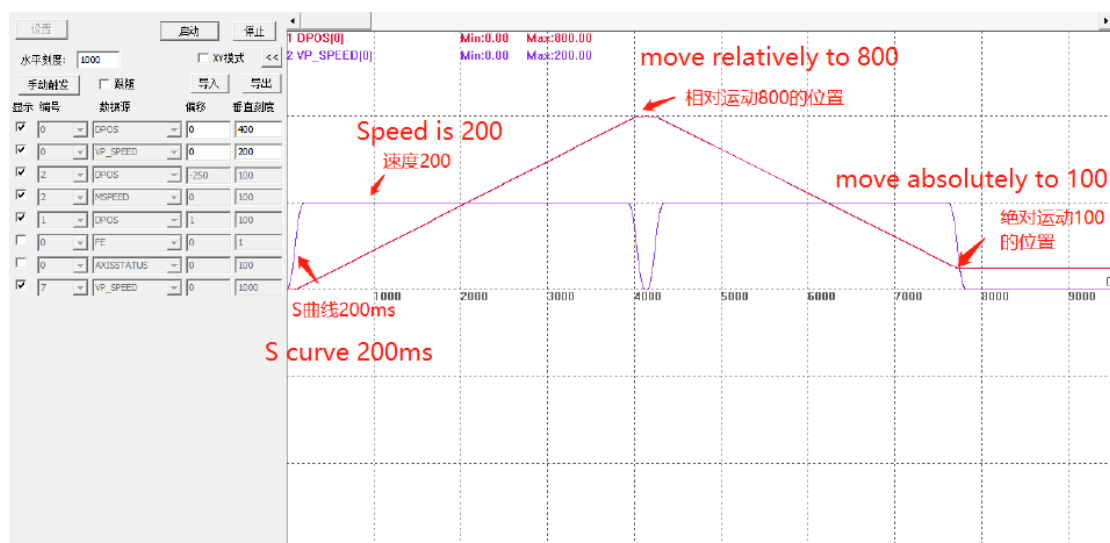
```

commandCheckHandler("ZAux_Direct_SetDpos", ret); //if it executes successfully
while (1) // wait for axis 0 to complete the motion
{
    Sleep(100);
    ZAux_Direct_GetParam(handle, "IDLE", 0, &IDLE);
    if (IDLE < 0) break;
}

Sleep(2000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret); //if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}

```

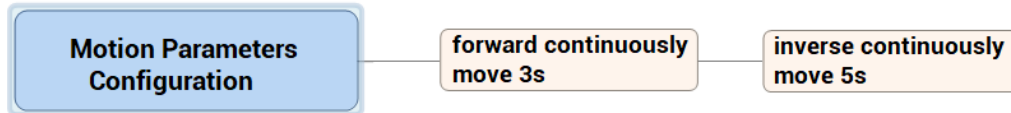
➤ Waveform:



Single-Axis Continuous Motion

In initialization, set basic parameters at first, like, DPOS = 0, pulse amount (UNITS) = 100, speed (SPEED) = 200 units/s, acceleration (ACCEL) = 2000units/s/s, deceleration (DECEL) = 2000units/s/s, S curve acceleration and deceleration time is 200ms. Then, set the motion, make it do continuous motion 3 seconds in forward, next make it do continuous motion 5 seconds in inversely. At last, stop the motion.

PS: S-shaped curve acceleration and deceleration time is the change time of the motion acceleration and deceleration process, and the trajectory is S-shaped, not trapezoidal (T-shaped).



```
// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zaxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"127.0.0.1"; //controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        getchar();
        return -1;
    }
    printf("Success to connect controller!\n");

    ZAux_Direct_SetAtype(handle, 0, 1); //set axis type of axis 0 as 1
    ZAux_Direct_SetUnits(handle, 0, 100); //set pulse amount of axis 0 as 100
    ZAux_Direct_SetSpeed(handle, 0, 200); //set speed of axis 0 as 200units/s
    ZAux_Direct_SetAccel(handle, 0, 2000); //set acceleration of axis 0 as 2000units/s/s
    ZAux_Direct_SetDecel(handle, 0, 2000); //set deceleration of axis as 2000units/s/s
    ZAux_Direct_SetSramp(handle, 0, 200); //set S curve of axis 0 as 200ms

    ret = ZAux_Direct_SetDpos( handle, 0, 0); //clear DPOS (axis command position)
    commandCheckHandler("ZAux_Direct_SetDpos", ret); // if it executes successfully
    ret = ZAux_Direct_SetMpos( handle, 0, 0); //clear MPOS (encoder feedback position)
    commandCheckHandler("ZAux_Direct_SetMpos", ret); // if it executes successfully

    ZAux_Trigger(handle); //trigger oscilloscope to capture data
    ret = ZAux_Direct_Single_Vmove( handle, 0, 1);
    commandCheckHandler("ZAux_Direct_Single_Vmove", ret); //if it executes successfully

    Sleep(3000); //delay 3s

    ret = ZAux_Direct_Single_Vmove( handle, 0, -1);
    commandCheckHandler("ZAux_Direct_SetDpos", ret); //if it executes successfully

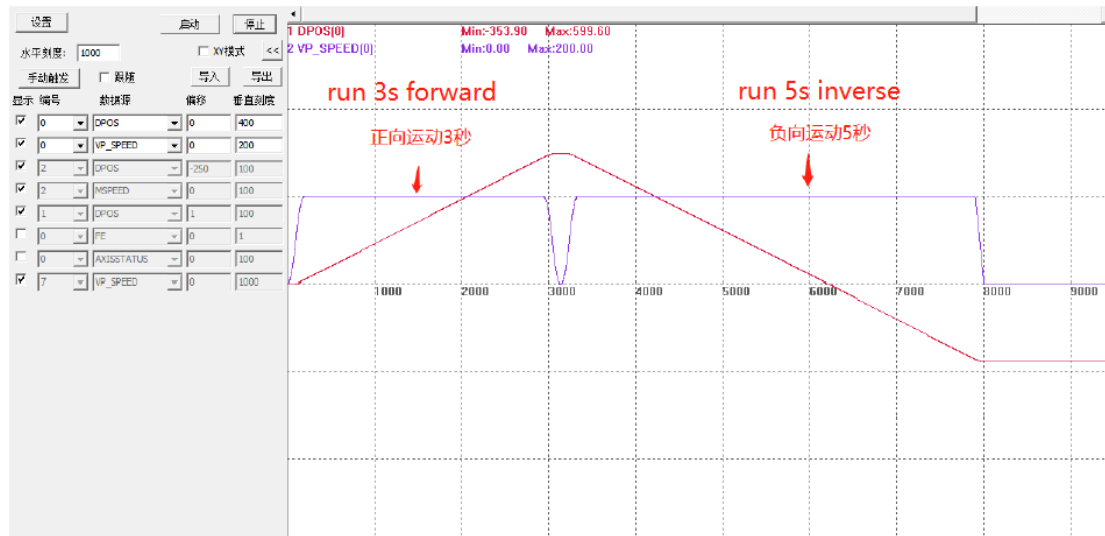
    Sleep(5000); //delay 5s
    ret = ZAux_Direct_Single_Cancel(handle, 0, 2); //stop the motion
    commandCheckHandler("ZAux_Direct_Single_Cancel", ret); //if it executes successfully
    ret = ZAux_Close(handle); //close the connection
    commandCheckHandler("ZAux_Close", ret); //if it executes successfully
    printf("connection closed!\n");
}
```

```

    handle = NULL;
    return 0;
}

```

➤ Waveform:



Quick Jog (IN Control)

Step 1: set basic parameters at first in initialization. Like, DPOS = 0, pulse amount (UNITS) = 100, speed (SPEED) = 200 units/s, acceleration (ACCEL) = 2000units/s/s, deceleration (DECEL) = 2000units/s/s, S curve acceleration and deceleration time is 200ms. And the JOG speed is set as 100.

Step 2: map IN, make IN0 for JOG speed and SPEED speed, make IN1 for forward motion, make IN2 for inverse motion.

PS: S-shaped curve acceleration and deceleration time is the change time of the motion ACCEL and DECEL, and the trajectory is S-shaped, not trapezoidal (T-shaped).



```

// test1.cpp: define the entry point of control panel application program
//

```

```

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

```

```

void commandCheckHandler(const char *command, int ret)

```

```

{
    if (ret)//it is not 0, fail
    {
        printf("%s return code is %d\n", command, ret);
    }
}

int32 __stdcall ZAux_Direct_ZSIMU_IN(ZMC_HANDLE handle,int link_axis,int move_axis)
{

    char cmdbuff[2048];
    char cmdbuffAck[2048];
    //generate the command
    sprintf(cmdbuff, "ZSIMU_IN(%d,%d) ", link_axis,move_axis);

    //call command to execute the function
    return ZAux_DirectCommand(handle, cmdbuff, cmdbuffAck, 2048);
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"127.0.0.1"; //controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        getchar();
        return -1;
    }
    printf("Success to connect controller!\n");

    ZAux_Direct_SetAtype(handle, 0, 1); //set axis type of axis 0 as 1
    ZAux_Direct_SetUnits(handle, 0, 100); //set pulse amount of axis 0 as 100
    ZAux_Direct_SetSpeed(handle, 0, 200); //set speed of axis 0 as 200units/s
    ZAux_Direct_SetAccel(handle, 0, 2000); //set acceleration of axis 0 as 2000units/s/s
    ZAux_Direct_SetDecel(handle, 0, 2000); //set deceleration of axis 0 as 2000units/s/s
    ZAux_Direct_SetSramp(handle, 0, 200); //set S curve of axis 0 as 200ms

    ret = ZAux_Direct_SetDpos( handle, 0, 0);//clear DPOS (axis command position)
    commandCheckHandler("ZAux_Direct_SetDpos", ret);//if it executes successfully
    ret = ZAux_Direct_SetMpos( handle, 0, 0);//clear MPOS (encoder feedback position)
    commandCheckHandler("ZAux_Direct_SetMpos", ret);//if it executes successfully

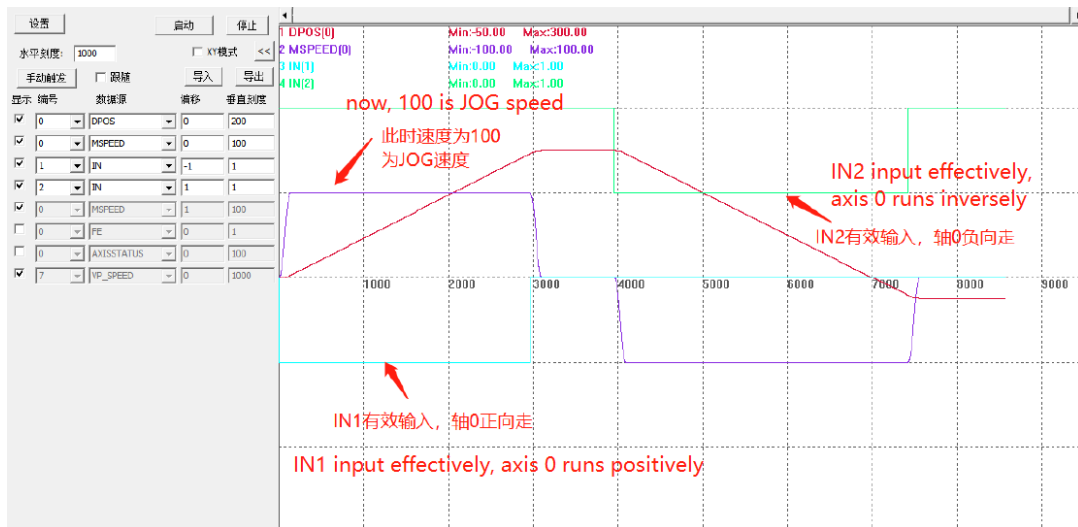
    ZAux_Direct_SetJogSpeed(handle, 0, 100);//set speed of JOG as 100
    ZAux_Direct_SetFastJog(handle, 0, 0);//set whether JOG speed opens control IN
    ZAux_Direct_SetFwdJog(handle, 0, 1);//set JOG forward control IN 0
    ZAux_Direct_SetRevJog(handle, 0, 2);// set JOG inverse control IN 1

    ZAux_Direct_SetInvertIn(handle, 0, 1);// "Invert IN" is valid, it is not needed for ECI cards.
    ZAux_Direct_SetInvertIn(handle, 1, 1);// "Invert IN" is valid, it is not needed for ECI cards.
    ZAux_Direct_SetInvertIn(handle, 2, 1);// "Invert IN" is valid, it is not needed for ECI cards.
    ZAux_Trigger(handle);//trigger oscilloscope to capture data

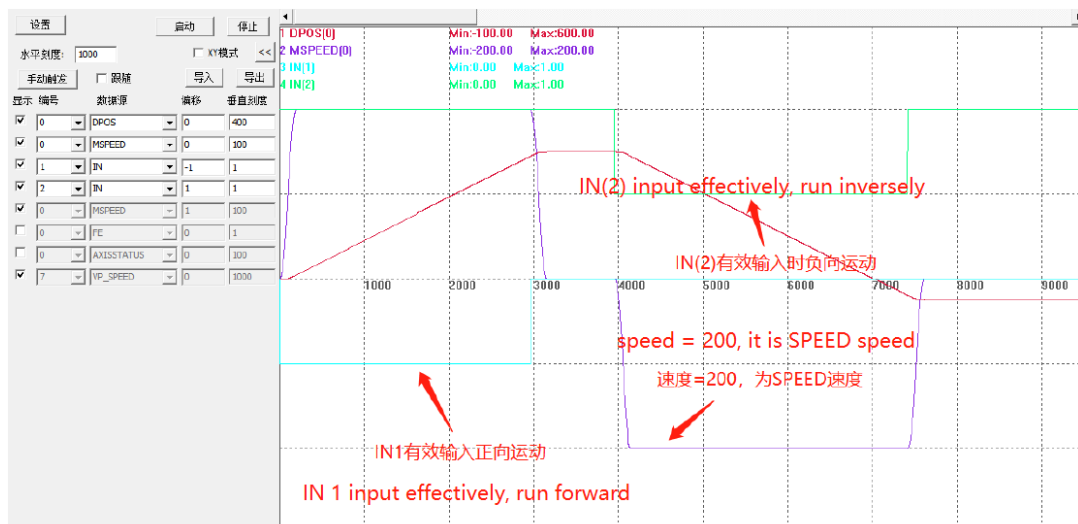
    Sleep(500);
    ret = ZAux_Close(handle); //close the connection
    commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
    printf("connection closed!\n");
    handle = NULL;
    return 0;
}

```

- When there is no valid input from IN(0):



- When there is valid input from IN(0):



6.3. Homing Motion

6.3.1. Instruction List

Commands	Descriptions
ZAux_Direct_Single_Datum	Homing in controller way
ZAux_BusCmd_Datum	Homing in bus drive way
ZAux_Direct_UserDatum	Customized twice homing
ZAux_Direct_SetHomeWait	Set homing inverse-find waiting time

ZAux_Direct_GetHomeWait	Read homing inverse-find waiting time
ZAux_BusCmd_SetDatumOffpos	Set homing offset distance
ZAux_BusCmd_GetDatumOffpos	Read homing offset distance
ZAux_Direct_SetCreep	Set homing creep speed
ZAux_Direct_GetCreep	Read homing creep speed
ZAux_Direct_GetHomeStatus	Read homing completion state
ZAux_Direct_GetHomeStatus2	Read homing completion state 2
ZAux_BusCmd_GetHomeStatus	Read drive homing completion state
ZAux_BusCmd_GetHomeStatus2	Read drive homing completion state 2
ZAux_Direct_SetDatumIn	Set homing origin signal
ZAux_Direct_GetDatumIn	Read homing origin signal

6.3.2. Principle & Description

Original input is configured by **ZAux_Direct_SetDatumIn** command, positive and negative position limit is configured through **ZAux_Direct_SetFwdIn** and **ZAux_Direct_SetRevIn** commands.

For **ZMC motion controllers**, the signal is valid when it is **in 0 state**, that is, when it is in OFF state, it reaches origin / position limit. For commonly-ON signal type, please use **ZAux_Direct_SetInvertIn** to invert the electricity level.

For **ECI motion controllers**, the signal is valid when it is **in 1 state**, that is, when it is in ON state, it reaches origin / position limit. For commonly-OFF signal type, please use **ZAux_Direct_SetInvertIn** to invert the electricity level.

Notes:

- For Z signal homing, the axis type must be configured as the type with signal Z.
- When multiple axes are homing, each axis needs to use homing function.
- When homing is finished in BUS (EtherCAT, RTEX, etc.) motion controller, MPOS is needed to be cleared by manual.

Homing Modes:

1) Mode 1:

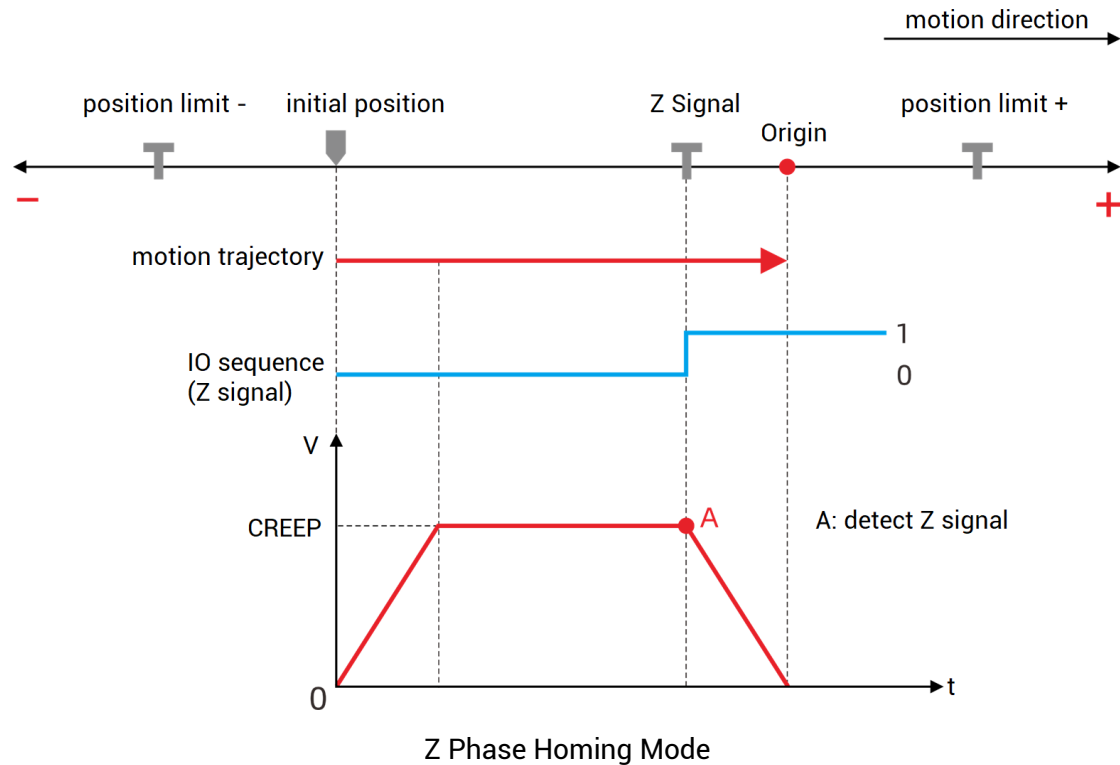
As shown in the figure below, set **ZAux_Direct_Single_Datum** as mode 1:

The axis runs forward at the speed of **ZAux_Direct_SetCreep** until the Z signal appears and starts to decelerate. After the deceleration reaches 0, the command

position (DPOS) value is reset to 0 at this time, and the position after stopped is the origin. In addition, it will stop directly if it touches the position limit switch while homing.

2) Mode 2:

Mode 2 is same as mode 1, except the motion direction, opposite to mode 1.



Note: after met Z phase signal in homing, it uses the DECEL value to decelerate. If there is a small error, which can be reduced by reducing the speed of CREEP and increasing the deceleration of DECEL. If there is a high-precision homing, it needs to use the Z-phase to latch and manually do homing for achieving the high-precision.

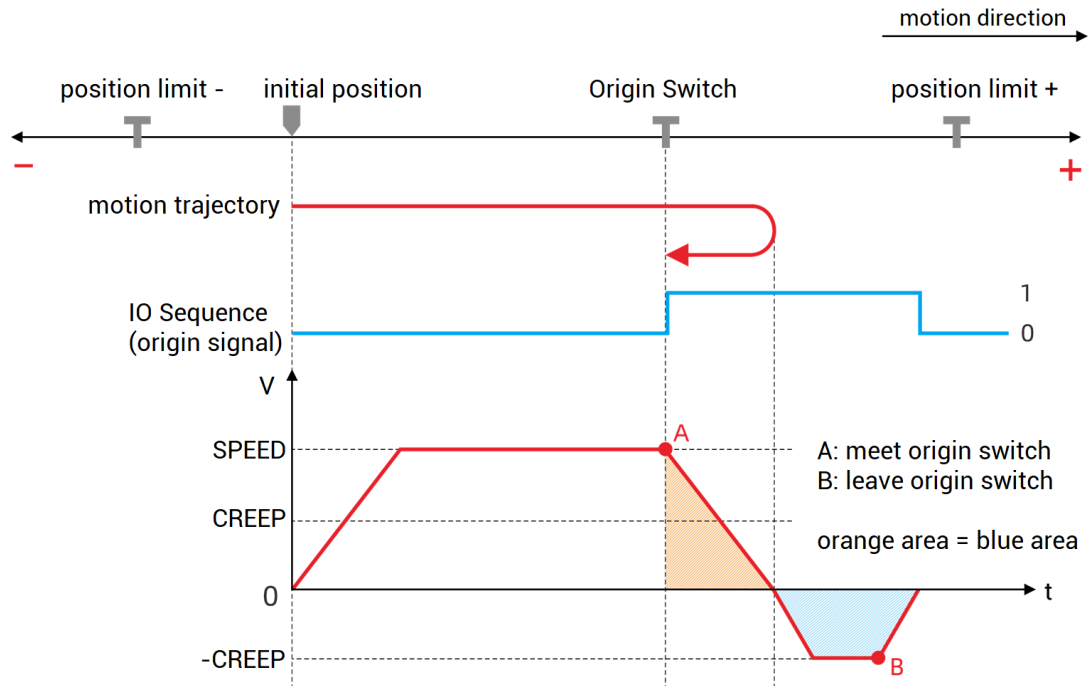
3) Mode 3:

As shown in the figure below, **ZAux_Direct_Single_Datum** mode 3:

The axis runs forward quickly at the speed of **ZAux_Direct_SetSpeed** until it meets the origin switch and starts to decelerate. After decelerating to 0, it inversely runs away the origin at the speed of **ZAux_Direct_SetCreep**. When meets the origin again, it will decelerate to stop, and the position value is reset to 0, the current position is the origin. In addition, it will stop directly if it touches the position limit switch while homing.

4) Mode 4:

Mode 4 is same as mode 3, except the motion direction, opposite to mode 3.



Note: after met Z phase signal in homing, it decelerates to the CREEP speed, and moves in reverse until it meets the origin IO induction, and then decelerates to 0 to stop. If there is a small error from IO (sense) to deceleration (to 0), which can be reduced by reducing the speed of CREEP and increasing the deceleration of DECEL. If there is a high-precision homing, it needs to use the Z-phase to latch and manually do homing for achieving the high-precision.

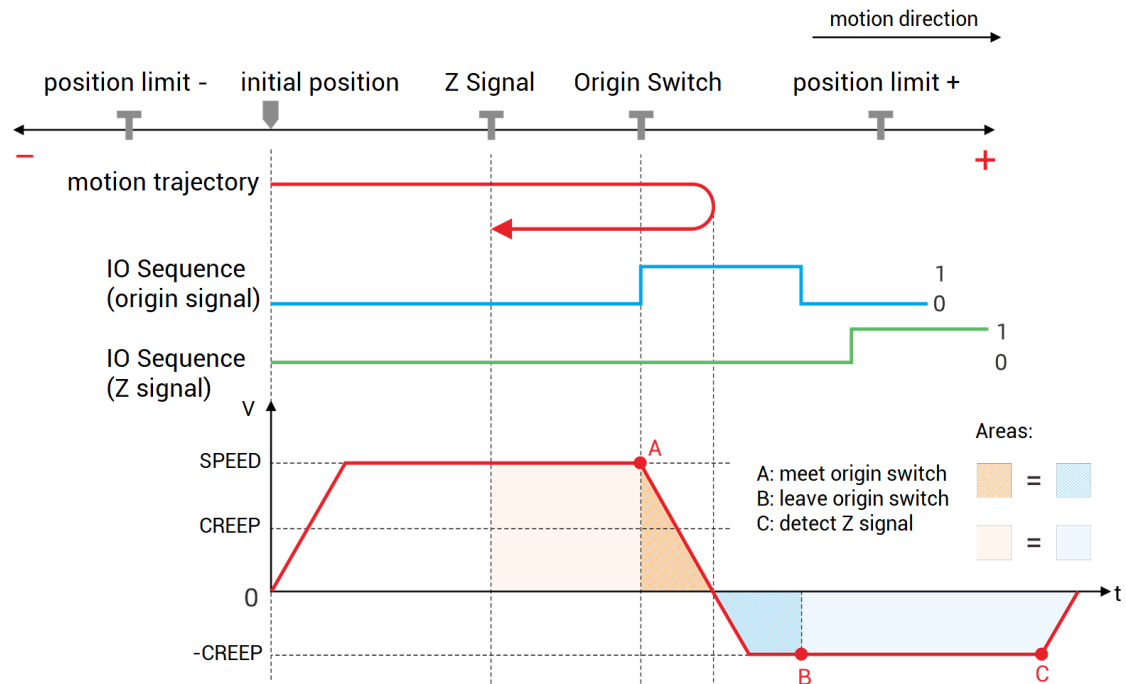
5) Mode 5:

As shown in the figure below, **ZAux_Direct_Single_Datum** mode 5:

The axis runs forward quickly at the speed of **ZAux_Direct_SetSpeed** until it meets the origin switch and starts to decelerate. After decelerating to 0, it reversely runs at the speed of **ZAux_Direct_SetCreep**, it only decelerates to stop until Z signal appears, and DPOS will be reset as 0 when meeting the Z signal. And the position where Z signal appears is the origin. In addition, it will stop directly if it touches the position limit switch while homing.

6) Mode 6:

Mode 6 is same as mode 5, except the motion direction, opposite to mode 5.



Note: after met origin signal in homing, it uses DECEL value to decelerate to CREEP speed, and moves reversely until it leaves the origin IO sense, and after met the Z signal, using DECEL to decelerate to 0 (stop). If there is a small error from "run away from Z signal sense" to "deceleration (to 0)", which can be reduced by reducing the speed of CREEP and increasing the deceleration of DECEL. If there is a high-precision homing, it needs to use the Z-phase to latch and manually homing for achieving the high-precision.

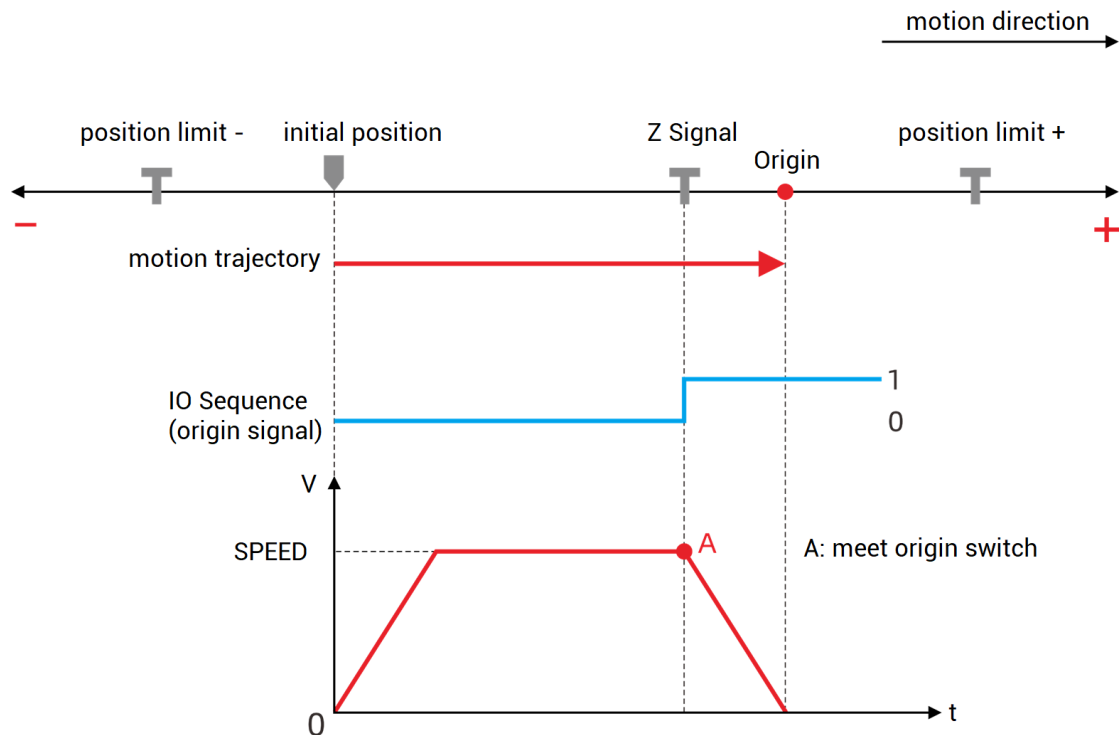
7) Mode 8:

As shown in the figure below, **ZAux_Direct_Single_Datum** mode 8:

The axis runs forward quickly at the speed of **ZAux_Direct_SetSpeed**, until it meets the origin switch and starts to decelerate. After decelerating to 0, DPOS will be reset as 0. And the stopped position is the origin. In addition, it will stop directly if it touches the position limit switch while homing.

8) Mode 9:

Mode 9 is same as mode 8, except the motion direction, opposite to mode 8.



Note: after met origin signal in homing, it uses DECEL value to decelerate to 0 (stop). If there is a small, which can be reduced by reducing the speed of CREEP and increasing the deceleration of DECEL. If there is a high-precision homing, it needs to use the Z-phase to latch and manually homing for achieving the high-precision.

6.3.3. Routines

Single-Axis Homing Motion

Here adopts homing mode 3.

The axis runs forward at the speed of ZAux_Direct_SetSpeed until it touches the origin switch. Then it moves reversely at the ZAux_Direct_SetCreep speed until it leaves the origin switch. What's more, the position will be automatically cleared to 0.

In the example, IN0 is used as the origin switch, and the level inversion is turned on.



```
// test1.cpp: define the entry point of control panel application program
#include "stdafx.h"
#include <windows.h>
```

```
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1"; //controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        getchar();
        return -1;
    }
    printf("Success to connect controller!\n");

    ZAux_Direct_SetAtype(handle, 0, 1); //set axis type of axis 0 as 1
    ZAux_Direct_SetUnits(handle, 0, 100); //set pulse amount of axis 0 as 100
    ZAux_Direct_SetSpeed(handle, 0, 200); //set speed of axis 0 as 200units/s
    ZAux_Direct_SetAccel(handle, 0, 2000); //set acceleration of axis 0 as 2000units/s/s
    ZAux_Direct_SetDecel(handle, 0, 2000); //set deceleration of axis 0 as 2000units/s/s
    ZAux_Direct_SetSramp(handle, 0, 200); //set S curve time of axis 0 as 200ms
    ret = ZAux_Direct_SetCreep(handle, 0, 50); //set inverse creep speed when homing
    commandCheckHandler("ZAux_Direct_SetCreep", 0);

    ret = ZAux_Direct_SetDatumIn(handle, 0, 0); //set origin switch
    commandCheckHandler("ZAux_Direct_SetDatumIn", 0);

    ret = ZAux_Direct_SetHomeWait(handle, 0, 1000); //set homing waiting time
    commandCheckHandler("ZAux_Direct_SetHomeWait", 0);

    ret = ZAux_Direct_SetInvertIn(handle, 0, 1); //set input 0 electric level invert
    commandCheckHandler("ZAux_Direct_SetInvertIn", 0);

    ret = ZAux_Direct_SetDpos(handle, 0, 0); //clear DPOS axis command position
    commandCheckHandler("ZAux_Direct_SetDpos", ret); //if it executes successfully
    ret = ZAux_Direct_SetMpos(handle, 0, 0); //clear MPOS encoder feedback position
    commandCheckHandler("ZAux_Direct_SetMpos", ret); //if it executes successfully

    ZAux_Trigger(handle); //trigger oscilloscope to capture data
    ret = ZAux_Direct_Single_Datum(handle, 0, 3); //homing, mode 3
    commandCheckHandler("ZAux_Direct_Single_Datum", 0);

    uint32 homestatus;
    while (1) //wait for axis 0 to complete homing
    {
        Sleep(100);
        ZAux_Direct_GetHomeStatus(handle, 0, &homestatus); //get homing motion finish state
        if (homestatus == 1) break;
    }

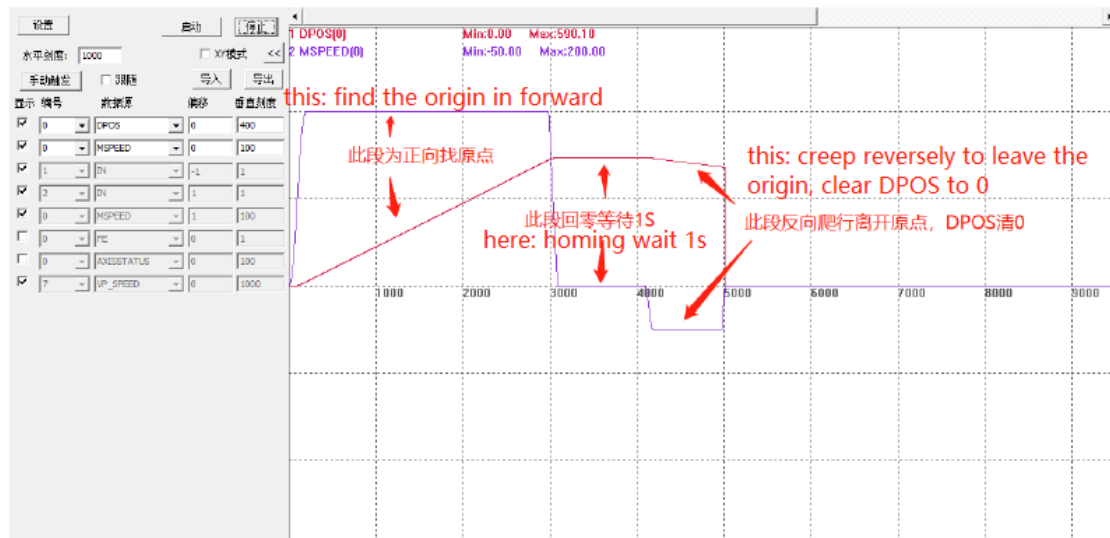
    Sleep(2000);
    ret = ZAux_Close(handle); //close the connection
    commandCheckHandler("ZAux_Close", ret); // if it executes successfully
    printf("connection closed!\n");
}
```

```

    handle = NULL;
    return 0;
}

```

➤ Waveform:



6.4. Electronic Gear

6.4.1. Instruction List

Commands	Descriptions
ZAux_Direct_SetClutchRate	Set connection velocity
ZAux_Direct_GetClutchRate	Read connection velocity
ZAux_Direct_EncoderRatio	Set encoder input gear ratio, default (1,1)
ZAux_Direct_StepRatio	Set pulse output gear ratio, default (1,1)
ZAux_Direct_Connect	"connection" & "synchronization" command for electronic gear
ZAux_Direct_Connpath	"connection" & "synchronization" command 2 for electronic gear
ZAux_Direct_GetLinkax	Read current connected motion's reference axis No.

6.4.2. Principle & Description

The electronic gear is used for the connection of two axes. The main axis and the slave axis are connected according to a **constant gear ratio**. There is no need for physical gears, and the ratio of the electronic gear can be directly set by using

instructions. And the electronic gear ratio can be modified at any time because it is realized by the software.

The related commands are **ZAux_Direct_Connect** and **ZAux_Direct_Connpath**. Connect one axis to another axis for follow-up motion according to a certain ratio, then one motion command can drive the operation of two motors. By detecting the movement of the two motor axes, feedback the displacement deviation to the controller and obtain the synchronization compensation, so that the displacement deviation between the two axes can be controlled within the allowable range of precision.

The electronic gear is connected to the **number of pulses**. For example, the connection ratio of the master and slave axes is 1:5, and one pulse is sent to the master axis, and 5 pulses are correspondingly sent to the slave axis.

After 2 axes are connected by the synchronous command, it needs to use the "cancel" command to cancel the synchronous motion of the current axis, and then call other motions. When switching the synchronous override, you can directly call ZAux_Direct_Connect to replace the previous ZAux_Direct_Connect command. What is connected is the number of pulses, and the ratio of different axis UNITS should be considered.

It is unlike "cam", the connection of electronic gears is linear.

Electronic Gear Functions:

- Pulse compensation, which can reduce the burden on the host computer (because the current components used to send pulses have limitations on the frequency of sending pulses).
- It can match the number of pulses sent by the motor and the minimum movement of the machine, and the movement of the workpiece (or the motor) corresponding to one pulse of the command input can be set to any value. In addition, it can achieve the infinitely variable speed of the motor, that is, when the motor starts and stops, it can prevent out-of-step and overshoot, so that the potential of the motor can be fully utilized.
- Transmit synchronous motion information, then realize coordinate linkage, transformation between motion forms (rotation-rotation, rotation-line, line-line), simplified control, etc.

The **same point** between **ZAux_Direct_Connpath** and **ZAux_Direct_Connect**: the usage syntax of the two is the same, both connect "pulses". In a word, effect is same

when only one axis is connected through ZAux_Direct_Connpath.

The difference between **ZAux_Direct_Connpath** and **ZAux_Direct_Connect**: ZAux_Direct_Connect connects the DPOS (target position) of a single axis. ZAux_Direct_Connpath connects the vector length of interpolation axis. At this time, it needs to be connected to the main axis of the interpolation movement, and it cannot follow the interpolation movement when connected to the interpolation slave axis. ZAux_Direct_Connpath will track the vector length change for XY interpolation instead of tracking individual X or Y axes.

6.4.3. Routines

Synchronous Motion

In this routine, drive axis 0 to move one distance that corresponds to 100 pulses, and set same parameters for axis 0 and axis 1, next see the effect before and after synchronization.

```
// test1.cpp: define the entry point of control panel application program.
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail! return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1"; // controller IP address, there may be errors when using
    // simulator to check the curve, it is best to connect to controller for viewing.
    ZMC_HANDLE handle = NULL; // link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); // connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        getchar();
        return -1;
    }
}
```



```

printf("Success to connect controller! \n");

ret = ZAux_Direct_SetAtype(handle,0,1); //set axis type of axis 0 as 1
commandCheckHandler("ZAux_Direct_SetAtype", ret) ;// if it executes successfully
ret = ZAux_Direct_SetAtype(handle,1,1); // set axis type of axis 1 as 1
commandCheckHandler("ZAux_Direct_SetAtype", ret) ;// if it executes successfully

ret = ZAux_Direct_SetUnits(handle,0,100); //set pulse amount (UNITS) of axis 0 as 100
commandCheckHandler("ZAux_Direct_SetUnits", ret) ;// if it executes successfully
ret = ZAux_Direct_SetUnits(handle,1,100); // set pulse amount (UNITS) of axis 1 as 100
commandCheckHandler("ZAux_Direct_SetUnits", ret) ;// if it executes successfully

ret = ZAux_Direct_SetAccel(handle,0,500); //set the acceleration of axis 0
commandCheckHandler("ZAux_Direct_SetAccel", ret) ;// if it executes successfully
ret = ZAux_Direct_SetAccel(handle,1,500); // set the acceleration of axis 1
commandCheckHandler("ZAux_Direct_SetAccel", ret) ;// if it executes successfully

ret = ZAux_Direct_SetDecel(handle,0,500); //set the deceleration of axis 0
commandCheckHandler("ZAux_Direct_SetDecel", ret) ;// if it executes successfully
ret = ZAux_Direct_SetDecel(handle,1,500); //set the deceleration of axis 1
commandCheckHandler("ZAux_Direct_SetDecel", ret) ;// if it executes successfully

ret = ZAux_Direct_SetDpos(handle,0,0); //set clearing DPOS of axis 0 (to 0)
commandCheckHandler("ZAux_Direct_SetDpos", ret) ;// if it executes successfully
ret = ZAux_Direct_SetDpos(handle,1,0); //set clearing DPOS of axis 1 (to 0)
commandCheckHandler("ZAux_Direct_SetDpos", ret) ;// if it executes successfully

ret = ZAux_Direct_SetSpeed(handle,0,100); //set the speed of axis 0
commandCheckHandler("ZAux_Direct_SetDecel", ret) ;// if it executes successfully

ret = ZAux_Direct_SetSpeed(handle,1,100); // set the speed of axis 1
commandCheckHandler("ZAux_Direct_SetDecel", ret) ;// if it executes successfully

ret = ZAux_Direct_Connect(handle,1,0,1); // open synchronous connection, axis 1 follows axis 0
commandCheckHandler("ZAux_Direct_Connect", ret) ;// if it executes successfully

float IDLE;
ZAux_Trigger( handle); //trigger the oscilloscope

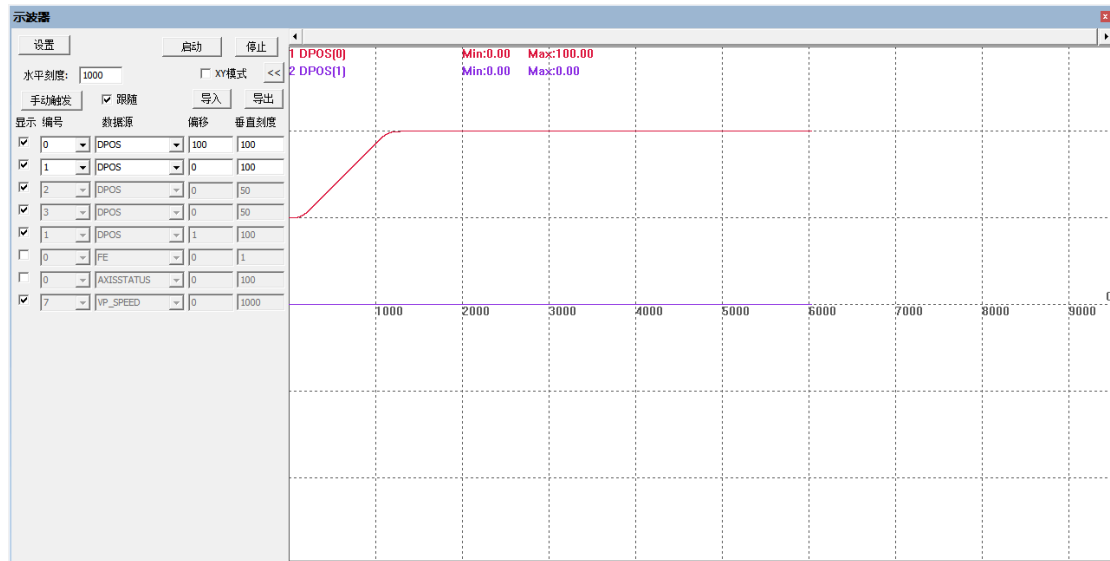
ret = ZAux_Direct_Single_Move(handle, 0,100.0); //axis 1 moves 100
commandCheckHandler("ZAux_Direct_Single_Move", ret) ;// if it executes successfully

while (1) //wait for axis 0 motion to complete
{
    Sleep(100);
    ZAux_Direct_GetParam(handle,"IDLE",0,&IDLE);
    if (IDLE<0) break;
}
ret = ZAux_Direct_Single_Cancel(handle,0,2); //cancel the synchronization motion
commandCheckHandler("ZAux_Direct_Connect", ret) ;// if it executes successfully

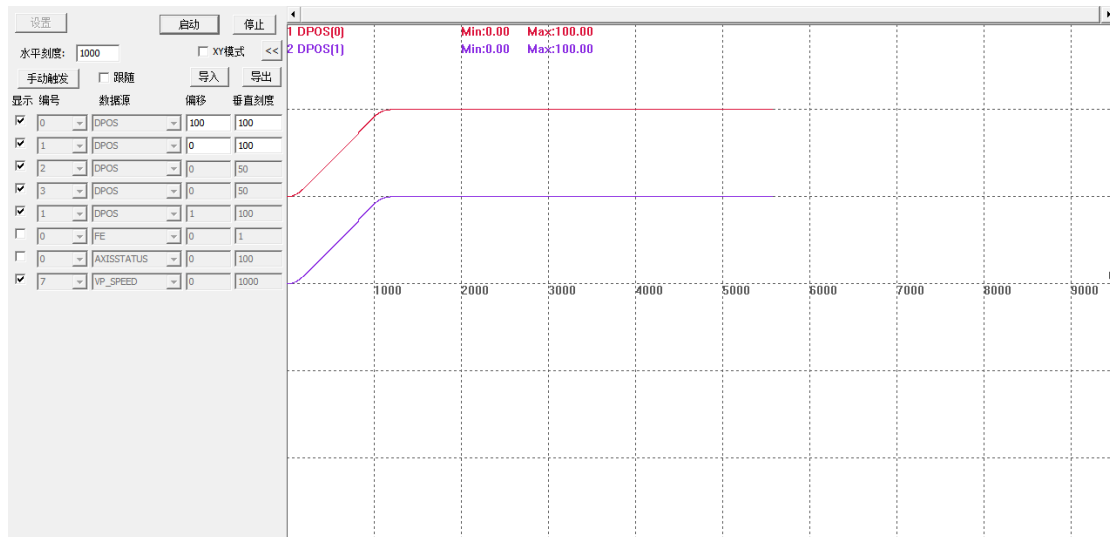
getchar();
ret = ZAux_Close(handle); //close the link
commandCheckHandler("ZAux_Close", ret) ;// if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}

```

➤ Before:



➤ After: ZAux_Direct_Connect (controller link handle, 1, 0, 1)



Synchronous Motion 2

Drive axis 2 to follow with the interpolation (vector) motion of axis 0 and axis 1, that is, axis 2 synchronizes with the interpolation motion.

```
// test1.cpp: define the entry point of control panel application program.
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {

```

```

        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"127.0.0.1"; // controller IP address, there may be errors when using
    simulator to check the curve, it is best to connect to controller for viewing.
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        getchar();
        return -1;
    }
    printf("Success to connect controller!\n");

    ret = ZAux_Direct_SetAtype(handle, 0, 1); // set axis type of axis 0 as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret); // if it executes successfully
    ret = ZAux_Direct_SetAtype(handle, 1, 1); // set axis type of axis 1 as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret); // if it executes successfully
    ret = ZAux_Direct_SetAtype(handle, 2, 1); // set axis type of axis 2 as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret); // if it executes successfully

    ret = ZAux_Direct_SetUnits(handle, 0, 100); // set pulse amount of axis 0 as 100
    commandCheckHandler("ZAux_Direct_SetUnits", ret); // if it executes successfully
    ret = ZAux_Direct_SetUnits(handle, 1, 100); // set pulse amount of axis 1 as 100
    commandCheckHandler("ZAux_Direct_SetUnits", ret); // if it executes successfully
    ret = ZAux_Direct_SetUnits(handle, 2, 100); // set pulse amount of axis 2 as 100
    commandCheckHandler("ZAux_Direct_SetUnits", ret); // if it executes successfully

    ret = ZAux_Direct_SetAccel(handle, 0, 500); // set the acceleration of axis 0
    commandCheckHandler("ZAux_Direct_SetAccel", ret); // if it executes successfully
    ret = ZAux_Direct_SetAccel(handle, 1, 500); // set the acceleration of axis 1
    commandCheckHandler("ZAux_Direct_SetAccel", ret); // if it executes successfully
    ret = ZAux_Direct_SetAccel(handle, 2, 500); // set the acceleration of axis 2
    commandCheckHandler("ZAux_Direct_SetAccel", ret); // if it executes successfully

    ret = ZAux_Direct_SetDecel(handle, 0, 500); // set the deceleration of axis 0
    commandCheckHandler("ZAux_Direct_SetDecel", ret); // if it executes successfully
    ret = ZAux_Direct_SetDecel(handle, 1, 500); // set the deceleration of axis 1
    commandCheckHandler("ZAux_Direct_SetDecel", ret); // if it executes successfully
    ret = ZAux_Direct_SetDecel(handle, 2, 500); // set the deceleration of axis 2
    commandCheckHandler("ZAux_Direct_SetDecel", ret); // if it executes successfully

    ret = ZAux_Direct_SetDpos(handle, 0, 0); // set clearing DPOS of axis 0 (to 0)
    commandCheckHandler("ZAux_Direct_SetDpos", ret); // if it executes successfully
    ret = ZAux_Direct_SetDpos(handle, 1, 0); // set clearing DPOS of axis 1 (to 0)
    commandCheckHandler("ZAux_Direct_SetDpos", ret); // if it executes successfully
    ret = ZAux_Direct_SetDpos(handle, 2, 0); // set clearing DPOS of axis 2 (to 0)
    commandCheckHandler("ZAux_Direct_SetDpos", ret); // if it executes successfully

    ret = ZAux_Direct_SetSpeed(handle, 0, 150); // set the speed of axis 0
    commandCheckHandler("ZAux_Direct_SetDecel", ret); // if it executes successfully
    ret = ZAux_Direct_SetSpeed(handle, 1, 100); // set the speed of axis 1
    commandCheckHandler("ZAux_Direct_SetDecel", ret); // if it executes successfully
    ret = ZAux_Direct_SetSpeed(handle, 2, 100); // set the speed of axis 2
    commandCheckHandler("ZAux_Direct_SetDecel", ret); // if it executes successfully

    ret = ZAux_Direct_Connpath(handle, 1, 0, 2); //open synchronous connection 2, axis 2 connects
    the axis 0 (interpolation master axis)
    commandCheckHandler("ZAux_Direct_Connect", ret); // if it executes successfully

```

```

float IDLE;
ZAux_Trigger( handle);// trigger the oscilloscope
int axislist[2]={0,1};
float dposlist[2]={300,400};
ret = ZAux_Direct_Move(handle, 2,axislist,dposlist); //interpolation motion, 300,400,
commandCheckHandler("ZAux_Direct_Single_Move", ret) ;// if it executes successfully

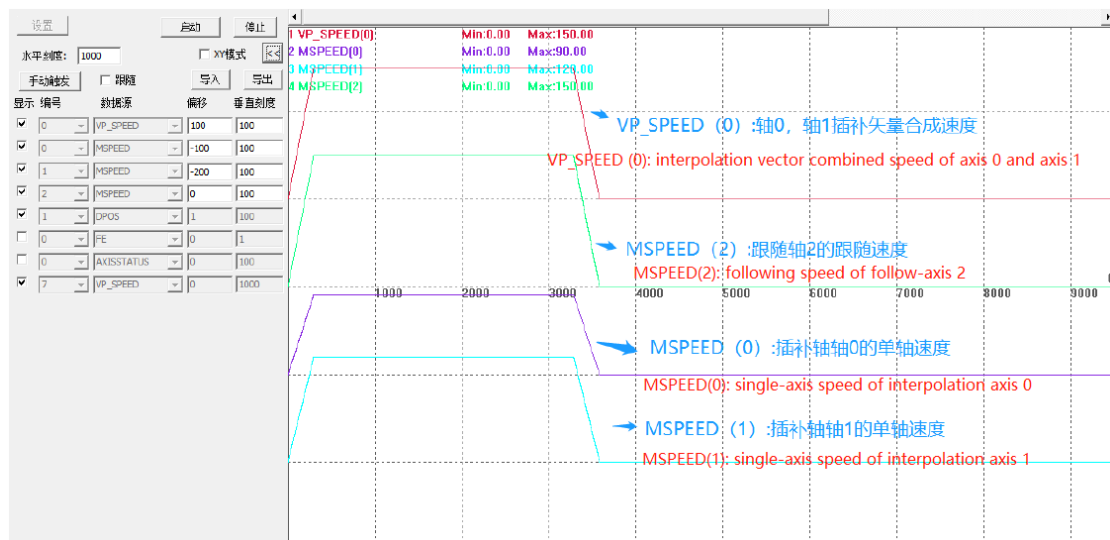
while (1)// wait for axis 0 motion to complete
{
    Sleep(100);
    ZAux_Direct_GetParam(handle,"IDLE",0,&IDLE);
    if (IDLE<0)break;
}
ret =ZAux_Direct_Single_Cancel(handle,0,2);// cancel the synchronization motion
commandCheckHandler("ZAux_Direct_Connect", ret) ;// if it executes successfully

getchar();
ret = ZAux_Close(handle); // close the link
commandCheckHandler("ZAux_Close", ret) ;// if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}

```

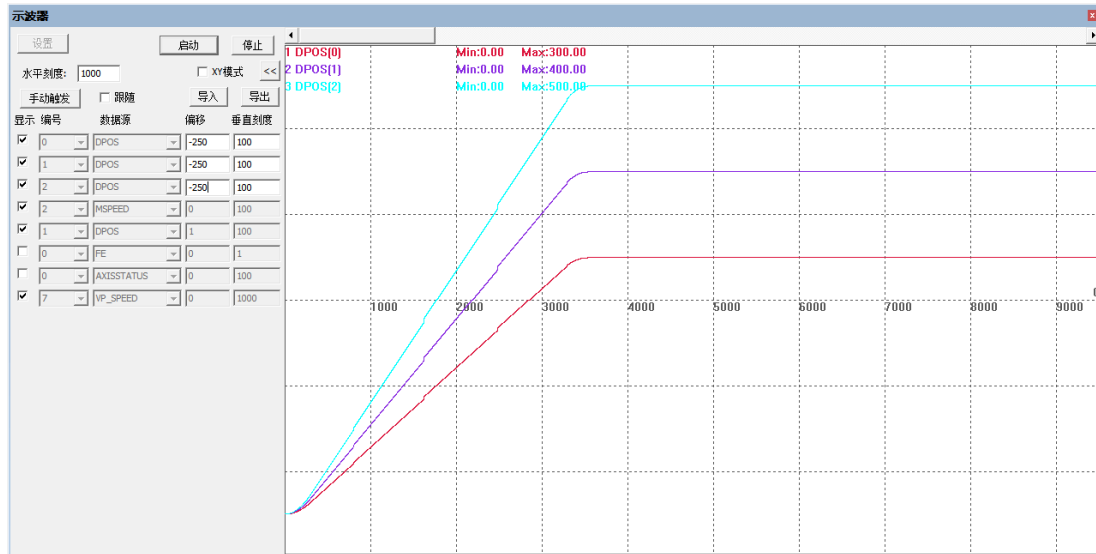
Below shows the curve of interpolation speed & following axis speed:

- it can be seen that the following axis changes with the speed change of the vector synthesized by the interpolation motion)



- it can be seen that the following axis moves with the trajectory of the vector synthesized by the interpolation motion.)

$$\text{Synthetic vector distance} = \text{DPOS} (2) = \sqrt{300^2 + 400^2} = 500$$



Pulse Electronic Gear Ratio Configuration

In this routine, when the encoder gear ratio defaults to 1, modify the pulse output gear ratio, and connect the pulse output of axis 0 to the encoder input of axis 5, so as to observe the actual situation of the pulse output of axis 0.

Hardware Wiring: wire the axis 0 pulse output to the axis 5 encoder.

```
// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "192.168.0.181"; //controller IP address, this routine needs to use
encoder for comparing, so it is necessary to connect to actual controller.
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        getchar();
        return -1;
    }
}
```

```
printf("Success to connect controller!\n");

ret =ZAux_Direct_SetAtype(handle,0,1);//set axis type of axis 0 as 1
commandCheckHandler("ZAux_Direct_SetAtype", ret) ;//if it executes successfully
ret =ZAux_Direct_SetAtype(handle,5,6);//set axis type of axis 5 as pulse encoder
commandCheckHandler("ZAux_Direct_SetAtype", ret) ;//if it executes successfully

ret =ZAux_Direct_SetUnits(handle,0,100);//set pulse amount of axis 0 as 100
commandCheckHandler("ZAux_Direct_SetUnits", ret) ;//if it executes successfully
ret =ZAux_Direct_SetUnits(handle,5,100);// set pulse amount of axis 5 as 100
commandCheckHandler("ZAux_Direct_SetUnits", ret) ;//if it executes successfully

ret =ZAux_Direct_SetAccel(handle,0,500);//set acceleration of axis 0
commandCheckHandler("ZAux_Direct_SetAccel", ret) ;//if it executes successfully

ret =ZAux_Direct_SetDecel(handle,0,500);//set deceleration of axis 0
commandCheckHandler("ZAux_Direct_SetDecel", ret) ;//if it executes successfully

ret =ZAux_Direct_SetDpos(handle,0,0);//set clear axis 0 DPOS (to 0)
commandCheckHandler("ZAux_Direct_SetDpos", ret) ;//if it executes successfully
ret =ZAux_Direct_SetMpos(handle,5,0);// set clear axis 5 MPOS (to 0)
commandCheckHandler("ZAux_Direct_SetDpos", ret) ;//if it executes successfully

ret =ZAux_Direct_SetSpeed(handle,0,100);//set speed of axis 0
commandCheckHandler("ZAux_Direct_SetDecel", ret) ;//if it executes successfully

ret =ZAux_Direct_StepRatio(handle,0,2,1);//set pulse gear ratio of axis 0 as 2, namely, 2 times
pulse output
commandCheckHandler("ZAux_Direct_StepRatio", ret) ;//if it executes successfully
float IDLE;
ZAux_Trigger( handle);// trigger the oscilloscope

ret = ZAux_Direct_Single_Move(handle, 0,100); // axis 0 moves 100
commandCheckHandler("ZAux_Direct_Single_Move", ret) ;//if it executes successfully

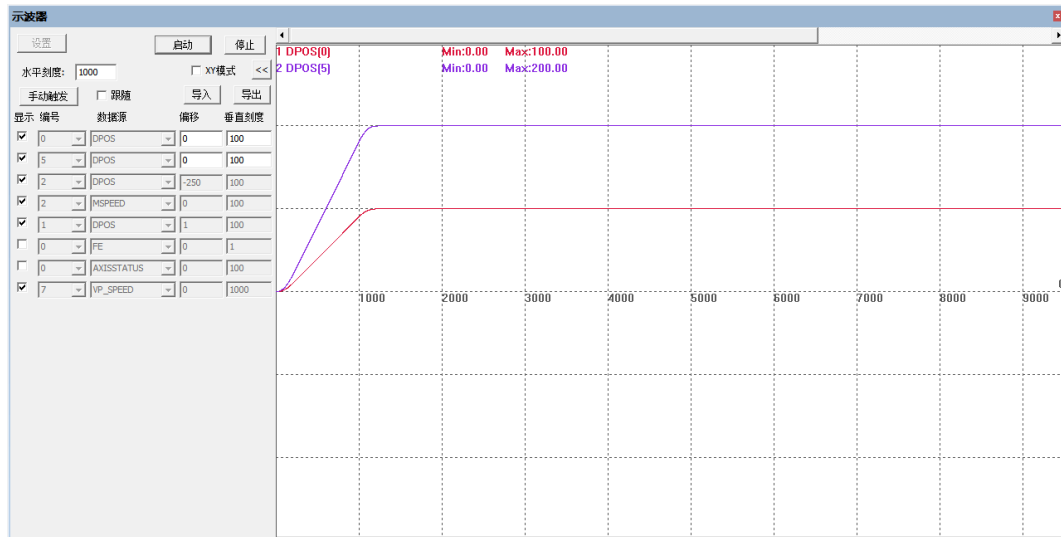
while (1)// wait for axis 0 motion to complete
{
    Sleep(100);
    ZAux_Direct_GetParam(handle,"IDLE",0,&IDLE);
    if (IDLE<0)break;
}

ret =ZAux_Direct_StepRatio(handle,0,1,1);//resume the pulse gear ratio of axis 0 as 1
commandCheckHandler("ZAux_Direct_StepRatio", ret) ;// if it executes successfully

getchar();
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}
```

➤ **Waveform:**

It can be seen that the pulse received by encoder axis 5 is twice that of axis 0, which is just in line with the configured pulse output gear ratio.



Encoder Gear Ratio Configuration

In this routine, when the pulse output gear ratio defaults to 1, modify the encoder input gear ratio, and connect the pulse output of axis 0 to the encoder input of axis 5, so as to observe the actual situation of pulse reception of axis 5.

Routine Hardware Wiring: wire the axis 0 pulse output to the axis 5 encoder.

```
// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "192.168.0.181"; //controller IP address, this routine needs to use
encoder for comparing, so it is necessary to connect to actual controller.
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        getchar();
        return -1;
    }
}
```

```

printf("Success to connect controller!\n");

ret =ZAux_Direct_SetAtype(handle,0,1);// set axis type of axis 0 as 1
commandCheckHandler("ZAux_Direct_SetAtype", ret) ;//if it executes successfully
ret =ZAux_Direct_SetAtype(handle,5,6);// set axis type of axis 5 as pulse encoder
commandCheckHandler("ZAux_Direct_SetAtype", ret) ;//if it executes successfully

ret =ZAux_Direct_SetUnits(handle,0,100);// set pulse amount of axis 0 as 100
commandCheckHandler("ZAux_Direct_SetUnits", ret) ;// if it executes successfully
ret =ZAux_Direct_SetUnits(handle,5,100);// set pulse amount of axis 5 as 100
commandCheckHandler("ZAux_Direct_SetUnits", ret) ;// if it executes successfully

ret =ZAux_Direct_SetAccel(handle,0,500);// set acceleration of axis 0
commandCheckHandler("ZAux_Direct_SetAccel", ret) ;// if it executes successfully

ret =ZAux_Direct_SetDecel(handle,0,500);// set deceleration of axis 0
commandCheckHandler("ZAux_Direct_SetDecel", ret) ;// if it executes successfully

ret =ZAux_Direct_SetDpos(handle,0,0);// set clear axis 0 DPOS (to 0)
commandCheckHandler("ZAux_Direct_SetDpos", ret) ;// if it executes successfully
ret =ZAux_Direct_SetMpos(handle,5,0);// set clear axis 5 MPOS (to 0)
commandCheckHandler("ZAux_Direct_SetDpos", ret) ;// if it executes successfully

ret =ZAux_Direct_SetSpeed(handle,0,100);// set speed of axis 0
commandCheckHandler("ZAux_Direct_SetDecel", ret) ;// if it executes successfully

ret =ZAux_Direct_EncoderRatio(handle,5,2,1);// set pulse gear ratio of axis 5 as 2, namely, 2
times pulse output
commandCheckHandler("ZAux_Direct_EncoderRatio", ret) ;// if it executes successfully
float IDLE;
ZAux_Trigger( handle);//trigger the oscilloscope

ret = ZAux_Direct_Single_Move(handle, 0,100); //axis 0 moves 100
commandCheckHandler("ZAux_Direct_Single_Move", ret) ;// if it executes successfully

while (1)// wait for axis 0 motion to complete
{
    Sleep(100);
    ZAux_Direct_GetParam(handle,"IDLE",0,&IDLE);
    if (IDLE<0)break;
}

ret =ZAux_Direct_EncoderRatio(handle,5,1,1);// resume the pulse gear ratio of axis 5 as 1
commandCheckHandler("ZAux_Direct_EncoderRatio", ret) ;// if it executes successfully

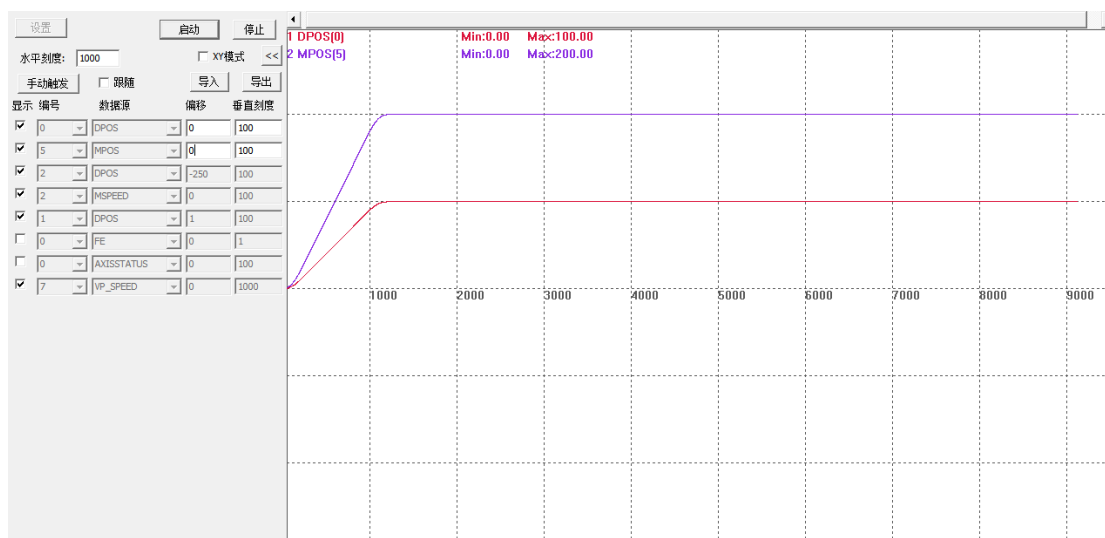
getchar();
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;// if it executes successfully

printf("connection closed!\n");
handle = NULL;
return 0;
}

```

➤ Output waveform:

It can be seen that the pulse received by encoder axis 5 is twice that of axis 0, which is just in line with the input gear ratio of the encoder.



6.5. Interpolation Motion

6.5.1. Instruction List

Commands	Descriptions
ZAux_Direct_Move	Multi-axis relative linear interpolation
ZAux_Direct_MoveSp	Multi-axis relative linear interpolation, SP
ZAux_Direct_MoveAbs	Multi-axis absolute linear interpolation
ZAux_Direct_MoveAbsSp	Multi-axis absolute linear interpolation, SP
ZAux_Direct_MoveCirc	Relative circular interpolation, use circle center to make one circle
ZAux_Direct_MoveCircSp	Relative circular interpolation, use circle center to make one circle, SP
ZAux_Direct_MoveCircAbs	Absolute circular interpolation, use circle center
ZAux_Direct_MoveCircAbsSp	Absolute circular interpolation, use circle center, SP
ZAux_Direct_MoveCirc2	Relative circular interpolation by 3-point
ZAux_Direct_MoveCirc2Abs	Absolute circular interpolation by 3-point
ZAux_Direct_MoveCirc2Sp	Relative circular interpolation by 3-point, SP
ZAux_Direct_MoveCirc2AbsSp	Absolute circular interpolation by 3-point, SP
ZAux_Direct_MHelical	Relative 3-axis helical interpolation by circle center
ZAux_Direct_MHelicalAbs	Absolute 3-axis helical interpolation by circle center
ZAux_Direct_MHelicalSp	Relative 3-axis helical interpolation by circle center, SP

<u>ZAux_Direct_MHelicalAbsSp</u>	Absolute 3-axis helical interpolation by circle center, SP
<u>ZAux_Direct_MHelical2</u>	Relative 3-axis helical interpolation by 3-point
<u>ZAux_Direct_MHelical2Abs</u>	Absolute 3-axis helical interpolation by 3-point
<u>ZAux_Direct_MHelical2Sp</u>	Relative 3-axis helical interpolation by 3-point, SP
<u>ZAux_Direct_MHelical2AbsSp</u>	Absolute 3-axis helical interpolation by 3-point, SP
<u>ZAux_Direct_MEclipse</u>	Relative ellipse interpolation
<u>ZAux_Direct_MEclipseAbs</u>	Absolute ellipse interpolation
<u>ZAux_Direct_MEclipseSp</u>	Relative ellipse interpolation, SP
<u>ZAux_Direct_MEclipseAbsSp</u>	Absolute ellipse interpolation, SP
<u>ZAux_Direct_MEclipseHelical</u>	Relative ellipse + helical interpolation
<u>ZAux_Direct_MEclipseHelicalAbs</u>	Absolute ellipse + helical interpolation
<u>ZAux_Direct_MEclipseHelicalSp</u>	Relative ellipse + helical interpolation, SP
<u>ZAux_Direct_MEclipseHelicalAbsSp</u>	Absolute ellipse + helical interpolation, SP
<u>ZAux_Direct_MSpherical</u>	Space arc + helical interpolation
<u>ZAux_Direct_MSphericalSp</u>	Space arc + helical interpolation, SP
<u>ZAux_Direct_MSphericalABSSp</u>	Absolute space arc + helical interpolation, SP
<u>ZAux_Direct_MoveSpiral</u>	Relative involute circular interpolation
<u>ZAux_Direct_MoveSpiralSp</u>	Relative involute circular interpolation, SP
<u>ZAux_Direct_MoveSmooth</u>	Space linear interpolation
<u>ZAux_Direct_MoveSmoothSp</u>	Space linear interpolation, SP
<u>ZAux_Direct_McircTurnabs</u>	Turntable linear interpolation
<u>ZAux_Direct_MoveTurnabs</u>	Turntable circular + helical interpolation
<u>ZAux_Direct_MultiMove</u>	Several multi-axis relative interpolation
<u>ZAux_Direct_MultiMoveAbs</u>	Several multi-axis absolute interpolation
<u>ZAux_Direct_MultiLineN</u>	Multi-axis multi-segment linear continuous interpolation
<u>ZAux_Direct_SetFhspeed</u>	Set axis HOLD speed
<u>ZAux_Direct_GetFhspeed</u>	Read axis HOLD speed
<u>ZAux_Direct_SetEndMoveSpeed</u>	Set SP motion's end speed
<u>ZAux_Direct_GetEndMoveSpeed</u>	Read SP motion's end speed
<u>ZAux_Direct_SetForceSpeed</u>	Set SP motion's running speed
<u>ZAux_Direct_GetForceSpeed</u>	Read SP motion's running speed
<u>ZAux_Direct_SetStartMoveSpeed</u>	Set SP motion's starting speed
<u>ZAux_Direct_GetStartMoveSpeed</u>	Read SP motion's starting speed
<u>ZAux_Direct_SetFholdIn</u>	Set HOLD IN No., -1 = invalid
<u>ZAux_Direct_GetFholdIn</u>	Read HOLD IN No., -1 = invalid
<u>ZAux_Direct_SetCornerMode</u>	Set corner deceleration mode
<u>ZAux_Direct_GetCornerMode</u>	Read corner deceleration mode
<u>ZAux_Direct_SetDecelAngle</u>	Set corner start deceleration angle, radian

ZAux_Direct_GetDecelAngle	Read corner start deceleration angle, radian
ZAux_Direct_SetFullSpRadius	Set the minimal radius of small-circle speed limit
ZAux_Direct_GetFullSpRadius	Read the minimal radius of small-circle speed limit
ZAux_Direct_SetInterpFactor	Set whether the axis joins in speed calculation when interpolation
ZAux_Direct_GetInterpFactor	Read whether the axis joins in speed calculation when interpolation
ZAux_Direct_SetMerge	Set continuous interpolation state
ZAux_Direct_GetMerge	Read continuous interpolation state
ZAux_Direct_SetStopAngle	Set the minimum turning angle to reduce deceleration to the lowest level, in radian
ZAux_Direct_GetStopAngle	Read the minimum turning angle to reduce deceleration to the lowest level, in radian
ZAux_Direct_SetZsmooth	Set deceleration chamfering radius
ZAux_Direct_GetZsmooth	Read deceleration chamfering radius

6.5.2. Principle & Description

SP Speed

1) Difference between SP and non-SP motion

Set the axis speed through **ZAux_Direct_SetSpeed**, the speed changes immediately, then dynamic speed can be realized, and all interpolation motions have corresponding SP motions that are with custom speeds, such as **ZAux_Direct_MoveSp**, **ZAux_Direct_MoveAbsSp**, and so on.

Note: for SP motion, it doesn't use global speed ZAux_Direct_SetSpeed, but sets starting, running, and end speeds by corresponding commands, and they also enter the motion buffer along with the motion.

2) SP Speed

SP speed is set and mainly for SP motions (the motion is with SP suffix), which can achieve customized speed control.

For SP motion, there are 3 speed parameters:

- ZAux_Direct_SetForceSpeed = forced speed of SP

- ZAux_Direct_SetStartMoveSpeed = starting speed of SP
- ZAux_Direct_SetEndMoveSpeed = end speed of SP

Note: above 3 parameters are only valid in SP motion, and parameters are all brought into the motion buffer. If no need to use it, please set a bigger value for ZAux_Direct_SetStartMoveSpeed, ZAux_Direct_SetEndMoveSpeed, otherwise this parameter will be used in the next movement command.

Continuous Interpolation & Trajectory Look-ahead

1) Continuous Interpolation

In the actual processing process, continuous interpolation is enabled in order to pursue machining efficiency. If the corners of the motion trajectory are not decelerated, when the corners are large, **it will cause a large impact on the machine and affect the machining accuracy.** If the continuous interpolation is turned off, the deceleration at the corner is reduced to 0. Although the machine is protected, the processing efficiency is greatly affected. **Therefore, a look-ahead command is provided** to automatically judge whether to reduce the corner speed to a reasonable value at the corner. In this way, it will not affect the machining accuracy and can improve the machining speed, which is the role of the trajectory look-ahead function.

The main command is **ZAux_Direct_SetMerge**.

2) Trajectory Look-Ahead

The trajectory look-ahead of the motion controller can automatically calculate a smooth speed according to the motion path, reducing the impact of the machine, thereby improving the machining accuracy. Automatically analyze the inflection point that will appear in the command trajectory of the motion buffer, and automatically calculate the motion speed at the corner according to the configured corner conditions, also calculate the speed based on the set maximum acceleration value, so that speed in any acceleration & deceleration will not exceed the value of ZAux_Direct_SetAccel and ZAux_Direct_SetDecel (prevent damage to the mechanical part).

The main command is **ZAux_Direct_SetCornerMode**, there are 3 normal modes:

Bit	Value	Description
0	1	Reserved

1	2	Automatic corner deceleration. Make acceleration and deceleration take effect according to ZAux_Direct_SetAccel and ZAux_Direct_SetDecel . The deceleration angle is set according to the ZAux_Direct_SetDecelAngle and ZAux_Direct_SetStopAngle . The reference speed of the deceleration corner is based on the FORCE_SPEED speed, please set a reasonable value.
2	4	Reserved
3	8	Automatic small circle speed limit, it will limit the speed when the radius is < the set value. The limit speed is related to ZAux_Direct_SetForceSpeed . Limit speed = ZAux_Direct_SetForceSpeed * actual radius / ZAux_Direct_SetFullSpRadius . The speed limit radius is set by ZAux_Direct_SetFullSpRadius .
4	16	Reserved
5	32	Automatic chamfering. The setting value of the radius of the chamfer is not the actual value, only when it is 90 degrees. This linear motion is automatically chamfered with the previous linear motion, and the chamfer radius refers to ZAux_Direct_SetZsmooth .

ZAux_Direct_SetCornerMode = 2 corner deceleration application: it will not change the motion trajectory, only automatically judge whether to decelerate at the corner, generally it is used to improve the problem of machine vibration. Occasions: requirement on trajectory accuracy, no need high-speed.

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

ZAux_Direct_SetCornerMode = 32 automatic chamfering application: it will change the motion trajectory but it will not reduce the speed. When there is big trajectory corner, the chamfering will be automatically smoothed. Occasions: requirement on trajectory speed, no need high-accuracy.

Notes:

- this parameter takes effect before the interpolation motion command is called, and the corner mode is generally set in parameter initialization. Because the look-ahead motion parameters will be stored in the motion buffer together with the motion command. If you want to cancel the parameters, please set ZAux_Direct_SetCornerMode as 0.

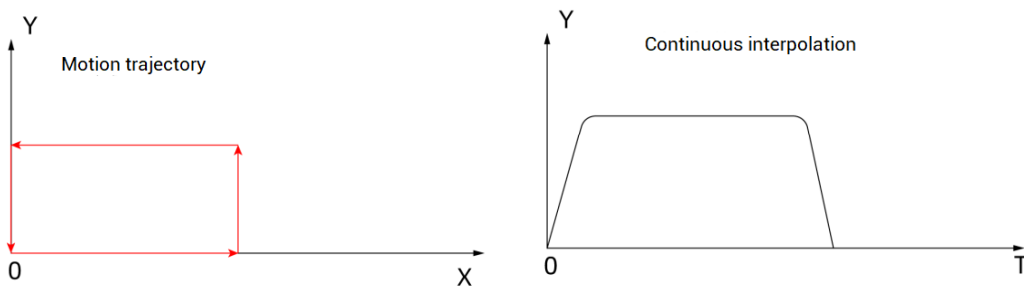
- parameters can be called multiple times, and different modes can also be used in combination. For example, ZAux_Direct_SetCornerMode = 2+8, automatic corner deceleration and small circle speed limit are used at the same time.

3) Let's see the speed planning when there is trajectory look-ahead and when without:

If it moves one rectangle (4 linear interpolation motions), as shown in the left figure.

a) Mode 1: MERGE=ON (open continuous interpolation)

After the continuous interpolation is turned on, the curve of the main axis speed changing with time is as shown in the right figure. The main axis speed is continuous, and the corner of the track still does not decelerate, and the impact at the corner is large when running at high speed.

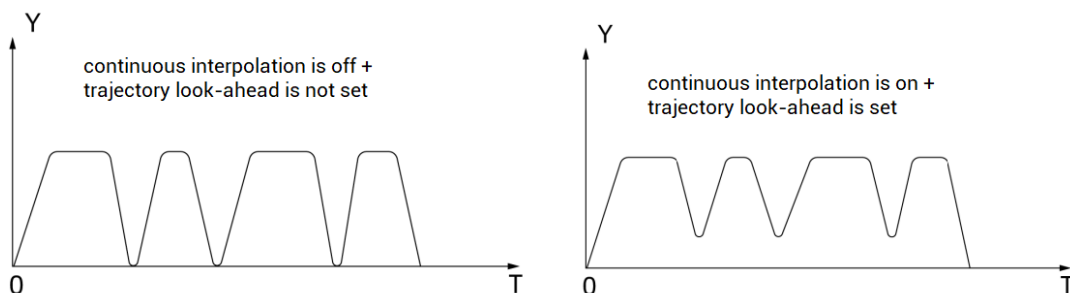


b) Mode 2: MERGE=OFF (close continuous interpolation)

Same parameters and motion as mode 1, but here turns off the continuous interpolation motion. The curve is shown in left. It can be seen it decelerates to 0 at first, then starts next motion, which is not efficient.

c) Mode 3: MERGE=ON + Set Look-Ahead Parameters

Same as mode 1, continuous interpolation is turned on, but here trajectory look-ahead parameters are set, see the right curve. It can be seen the corners are decelerated according to a certain ratio, the motion trajectory is smoother, and the processing efficiency is higher than that of mode 2.



If you want the speed curve to be softer in the above modes, you only need to set the speed to S curve through the **ZAux_Direct_SetSramp**.

6.5.3. Routines

Corner Deceleration

After the corner deceleration is turned on, if SP start speed or SP end speed (ZAux_Direct_SetStartMoveSpeed / ZAux_Direct_SetEndMoveSpeed) is set, the corner deceleration will be invalid, because the corner will move at set these two speeds.

In this example, move 3 linear motions. Set corner deceleration mode as 2, open continuous interpolation, and the deceleration angle is set between 15-45 degrees.

```
// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"
#define PI 3.1415926

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "192.168.0.181"; //controller IP address, there may be error if checking
    the curve through the simulator, so it is better to connect to actual controller to check.
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        getchar();
        return -1;
    }
    printf("Success to connect controller!\n");

    ret = ZAux_Direct_SetAtype(handle, 0, 1); // set axis type of axis 0 as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret); // if it executes successfully
    ret = ZAux_Direct_SetAtype(handle, 1, 1); // set axis type of axis 1 as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret); // if it executes successfully

    ret = ZAux_Direct_SetUnits(handle, 0, 100); // set pulse amount of axis 0 as 100
    commandCheckHandler("ZAux_Direct_SetUnits", ret); // if it executes successfully
```

```

ret =ZAux_Direct_SetUnits(handle,1,100);// set pulse amount of axis 1 as 100
commandCheckHandler("ZAux_Direct_SetUnits", ret) ;// if it executes successfully

ret =ZAux_Direct_SetAccel(handle,0,500);// set acceleration of axis 0
commandCheckHandler("ZAux_Direct_SetAccel", ret) ;// if it executes successfully
ret =ZAux_Direct_SetAccel(handle,1,500);// set acceleration of axis 1
commandCheckHandler("ZAux_Direct_SetAccel", ret) ;// if it executes successfully

ret =ZAux_Direct_SetDecel(handle,0,500);// set deceleration of axis 0
commandCheckHandler("ZAux_Direct_SetDecel", ret) ;// if it executes successfully
ret =ZAux_Direct_SetDecel(handle,1,500);// set deceleration of axis 1
commandCheckHandler("ZAux_Direct_SetDecel", ret) ;// if it executes successfully

ret =ZAux_Direct_SetDpos(handle,0,0);// set clear axis 0 DPOS (to 0)
commandCheckHandler("ZAux_Direct_SetDpos", ret) ;// if it executes successfully
ret =ZAux_Direct_SetDpos(handle,1,0);// set clear axis 1 DPOS (to 0)
commandCheckHandler("ZAux_Direct_SetDpos", ret) ;// if it executes successfully

ret =ZAux_Direct_SetSpeed(handle,0,100);// set speed of axis 0
commandCheckHandler("ZAux_Direct_SetDecel", ret) ;// if it executes successfully

ret =ZAux_Direct_SetSpeed(handle,1,100);// set speed of axis 1
commandCheckHandler("ZAux_Direct_SetDecel", ret) ;// if it executes successfully

ret =ZAux_Direct_SetMerge(handle,0,1);//open continuous interpolation (only main axis is
opened, for example, axis 0 and axis 1 interpolate, axis 0 is the main axis. The main axis is
determined by the first axis No. in continuous interpolation motion command list)
commandCheckHandler("ZAux_Direct_SetMerge", ret) ;// if it executes successfully

ret =ZAux_Direct_SetCornerMode(handle,0,2);//open corner mode (only main axis is opened,
for example, axis 0 and axis 1 interpolate, axis 0 is the main axis. The main axis is determined by
the first axis No. in continuous interpolation motion command list)
commandCheckHandler("ZAux_Direct_SetCornerMode", ret) ;// if it executes successfully

ret =ZAux_Direct_SetDecelAngle(handle,0,15 * (PI/180));//set starting deceleration angle (only
main axis is opened, for example, axis 0 and axis 1 interpolate, axis 0 is the main axis. The main
axis is determined by the first axis No. in continuous interpolation motion command list)
commandCheckHandler("ZAux_Direct_SetDecelAngle", ret) ;// if it executes successfully

ret =ZAux_Direct_SetStopAngle(handle,0,45 * (PI/180));//set end deceleration angle (only main
axis is opened, for example, axis 0 and axis 1 interpolate, axis 0 is the main axis. The main axis is
determined by the first axis No. in continuous interpolation motion command list)
commandCheckHandler("ZAux_Direct_SetStopAngle", ret) ;// if it executes successfully

ret =ZAux_Direct_SetForceSpeed(handle,0,100);//set the speed when in equal deceleration
(only main axis is opened, for example, axis 0 and axis 1 interpolate, axis 0 is the main axis. The
main axis is determined by the first axis No. in continuous interpolation motion command list)
commandCheckHandler("ZAux_Direct_SetForceSpeed", ret) ;// if it executes successfully

float IDLE;
float DposList[2]={0};
int Axislist[2]={0,1};

ZAux_Trigger( handle);// trigger the oscilloscope
DposList[0]=100;
DposList[1]=0;
ret =ZAux_Direct_Move(handle,2,Axislist,DposList);//interpolation (100,0)
commandCheckHandler("ZAux_Direct_SetForceSpeed", ret) ;// if it executes successfully
DposList[0]=0;
DposList[1]=100;
ret =ZAux_Direct_Move(handle,2,Axislist,DposList);//interpolation (0,100), motion angle is
more than 45°, full deceleration
commandCheckHandler("ZAux_Direct_SetForceSpeed", ret) ;// if it executes successfully
DposList[0]=60;

```



```

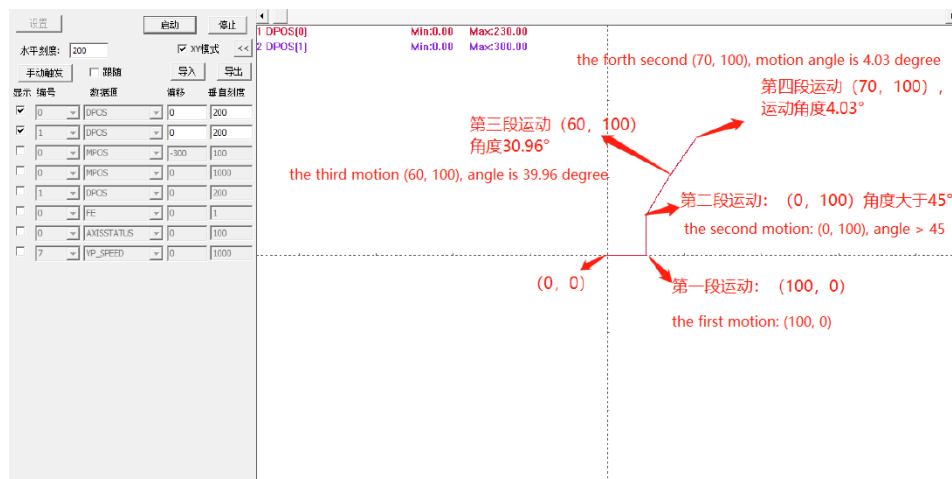
DposList[1]=100;
ret =ZAux_Direct_Move(handle,2,Axislist,DposList);//interpolation (60, 100), motion angle
30.96° is 15°~45°, full deceleration
commandCheckHandler("ZAux_Direct_SetForceSpeed", ret) ;// if it executes successfully
DposList[0]=70;
DposList[1]=100;
ret =ZAux_Direct_Move(handle,2,Axislist,DposList);//interpolation (70, 100), motion angle 4.03°
is 15°, not to decelerate
commandCheckHandler("ZAux_Direct_SetForceSpeed", ret) ;// if it executes successfully

while (1)// wait for axis 0 motion to complete
{
    Sleep(100);
    ZAux_Direct_GetParam(handle,"IDLE",0,&IDLE);
    if (IDLE<0)break;
}

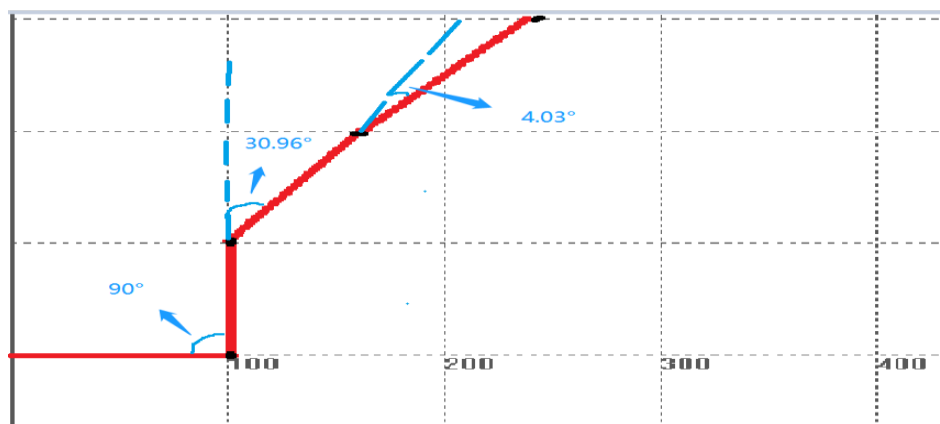
getchar();
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret); //if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}

```

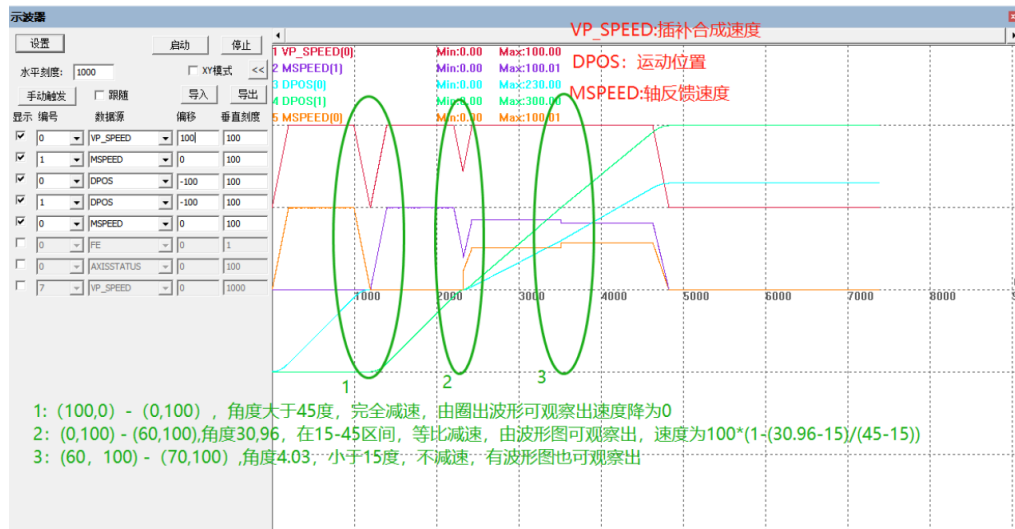
➤ Trajectory Curve:



➤ Angle analysis:



➤ Speed Curve:



- VP_SPEED: interpolation composite speed
- DPOS: motion position
- MSPEED: axis feedback speed
- 1. (100,0) – (0,100), angle is more than 45 degrees, full decelerate, it can be seen the speed is decelerated to 0 from the marked waveform (green, 1)
- 2. (0,100) – (70,100), angle is 90.96 degrees, which is between 15 to 45, proportionally decelerate, it can be seen the speed is $100 \times (1 - (30.96 - 15) / (45 - 15))$ from marked green 2 waveform.
- 3. (60,100) – (70,100), angle is 4.03 degrees, which is less than 15 degrees, not to decelerate, please check marked green 3 waveform

Draw A Circle with Plane Center & Small-Circle Speed Limit

Set the speed limit as 60 for small circle. When the radius is $>$ or $=$ the limit value, the speed is not limited, and run at the speed of SPEED, at this time, the speed is 100.

```
// test1.cpp: define the entry point of control panel application program
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"
#define PI 3.1415926

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}
```

```
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"192.168.0.181"; //controller IP address, there may be error if checking
    the curve through the simulator, so it is better to connect to actual controller to check.
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        getchar();
        return -1;
    }
    printf("Success to connect controller!\n");

    ret = ZAux_Direct_SetAtype(handle, 0, 1); // set axis type of axis 0 as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret) ;//if it executes successfully
    ret = ZAux_Direct_SetAtype(handle, 1, 1); // set axis type of axis 1 as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret) ;//if it executes successfully

    ret = ZAux_Direct_SetUnits(handle, 0, 100); // set pulse amount of axis 0 as 100
    commandCheckHandler("ZAux_Direct_SetUnits", ret) ;//if it executes successfully
    ret = ZAux_Direct_SetUnits(handle, 1, 100); // set pulse amount of axis 1 as 100
    commandCheckHandler("ZAux_Direct_SetUnits", ret) ;//if it executes successfully

    ret = ZAux_Direct_SetAccel(handle, 0, 500); // set acceleration of axis 0
    commandCheckHandler("ZAux_Direct_SetAccel", ret) ;//if it executes successfully
    ret = ZAux_Direct_SetAccel(handle, 1, 500); // set acceleration of axis 1
    commandCheckHandler("ZAux_Direct_SetAccel", ret) ;//if it executes successfully

    ret = ZAux_Direct_SetDecel(handle, 0, 500); // set deceleration of axis 0
    commandCheckHandler("ZAux_Direct_SetDecel", ret) ;//if it executes successfully
    ret = ZAux_Direct_SetDecel(handle, 1, 500); // set deceleration of axis 1
    commandCheckHandler("ZAux_Direct_SetDecel", ret) ;//if it executes successfully

    ret = ZAux_Direct_SetDpos(handle, 0, 0); // set clear axis 0 DPOS (to 0)
    commandCheckHandler("ZAux_Direct_SetDpos", ret) ;//if it executes successfully
    ret = ZAux_Direct_SetDpos(handle, 1, 0); // set clear axis 1 DPOS (to 0)
    commandCheckHandler("ZAux_Direct_SetDpos", ret) ;//if it executes successfully

    ret = ZAux_Direct_SetSpeed(handle, 0, 100); // set speed of axis 0
    commandCheckHandler("ZAux_Direct_SetDecel", ret) ;//if it executes successfully

    ret = ZAux_Direct_SetSpeed(handle, 1, 100); // set speed of axis 1
    commandCheckHandler("ZAux_Direct_SetDecel", ret) ;//if it executes successfully

    ret = ZAux_Direct_SetMerge(handle, 0, 1); //open continuous interpolation (only main axis is
    opened, for example, axis 0 and axis 1 interpolate, axis 0 is the main axis. The main axis is
    determined by the first axis No. in continuous interpolation motion command list)
    commandCheckHandler("ZAux_Direct_SetMerge", ret) ;//if it executes successfully

    ret = ZAux_Direct_SetCornerMode(handle, 0, 8); // open small circle speed limit (only main axis is
    opened, for example, axis 0 and axis 1 interpolate, axis 0 is the main axis. The main axis is
    determined by the first axis No. in continuous interpolation motion command list)
    commandCheckHandler("ZAux_Direct_SetCornerMode", ret) ;//if it executes successfully

    ret = ZAux_Direct_SetFullSpRadius(handle, 0, 60); // set starting deceleration angle (only main
    axis is opened, for example, axis 0 and axis 1 interpolate, axis 0 is the main axis. The main axis is
    determined by the first axis No. in continuous interpolation motion command list)
    commandCheckHandler("ZAux_Direct_SetFullSpRadius", ret) ;//if it executes successfully

    ret = ZAux_Direct_SetForceSpeed(handle, 0, 100); //set the speed when in equal deceleration
```

(only main axis is opened, for example, axis 0 and axis 1 interpolate, axis 0 is the main axis. The main axis is determined by the first axis No. in continuous interpolation motion command list)
 commandCheckHandler("ZAux_Direct_SetForceSpeed", ret) ;//if it executes successfully

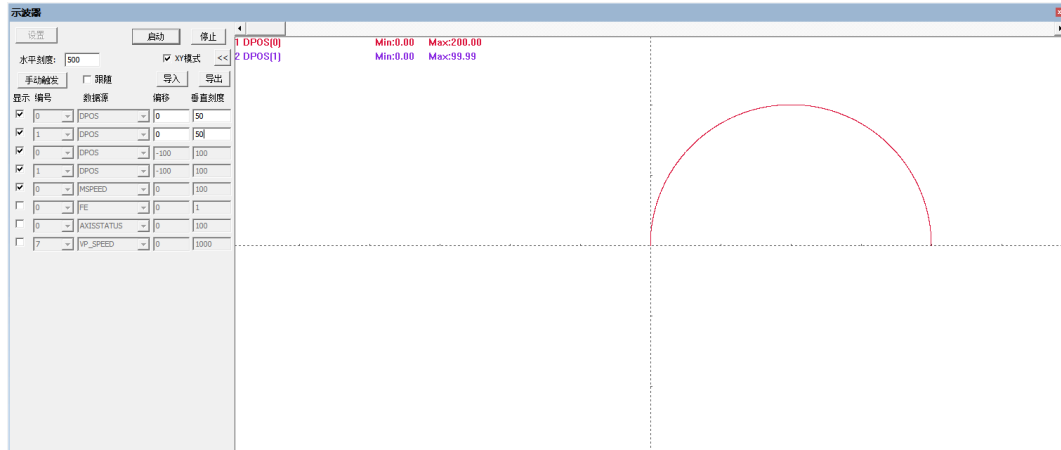
```
float IDLE;
int Axislist[2]={0,1};
```

```
ZAux_Trigger( handle);//trigger the oscilloscope
//the coordinate is 0,0 before the motion
//interpolation trajectory end point coordinate is 100,0
//circle center coordinate is 50,0,
//the radius is 50 from above
ret =ZAux_Direct_MoveCirc(handle,2,Axislist,100,0,50,0,1);//circular interpolation motion
commandCheckHandler("ZAux_Direct_MoveCirc", ret) ;//if it executes successfully
```

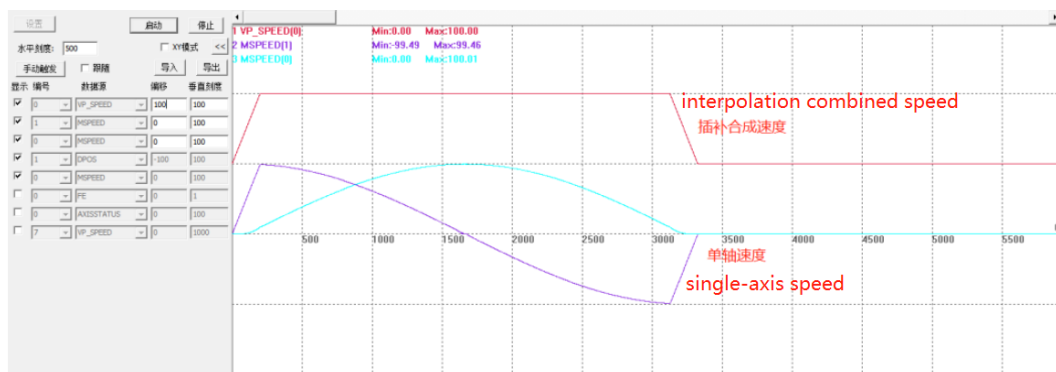
```
while (1)//wait for axis 0 motion to complete
{
    Sleep(100);
    ZAux_Direct_GetParam(handle,"IDLE",0,&IDLE);
    if (IDLE<0)break;
}
```

```
getchar();
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}
```

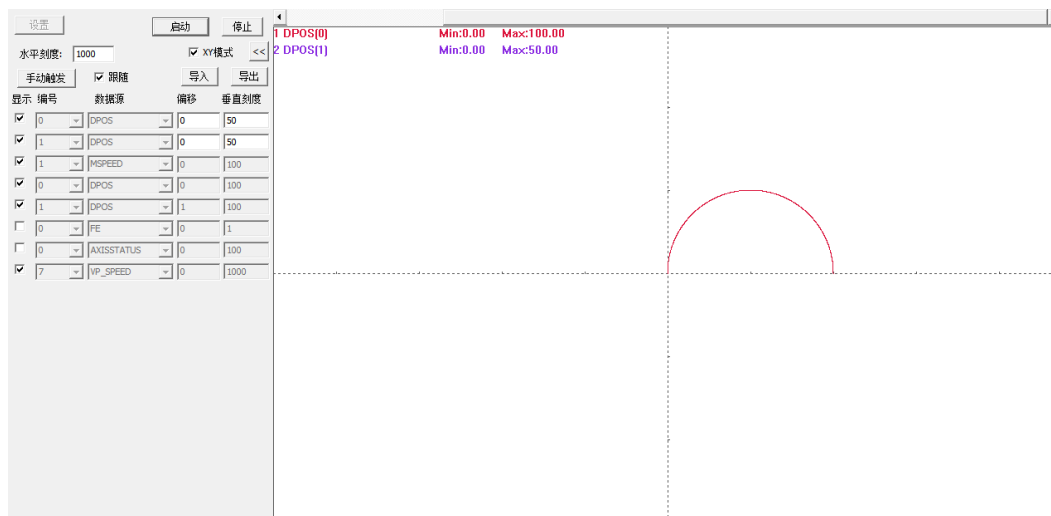
- The radius is 100, $100 > 60$, trajectory waveform:



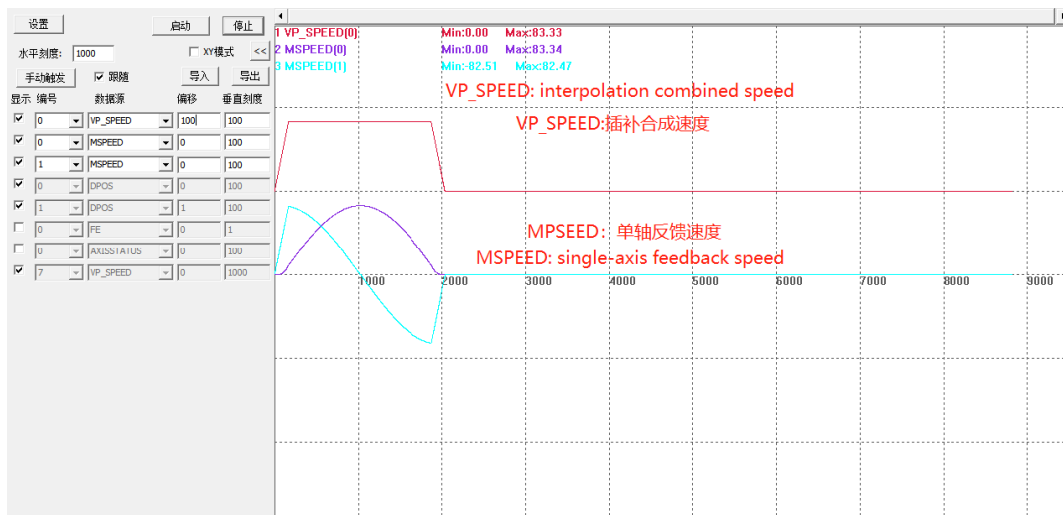
- Set the radius as 100, $100 > 60$, speed curve waveform (not to decelerate totally):



- The radius is 50, $50 < 60$, trajectory waveform:



- The radius is 50, $50 < 60$, speed curve waveform (small circle deceleration)
 //when radius is equal to limit value, $\text{limit} = \text{FORCE\_SPEED} * \text{actual radius} / \text{FULL\_SP\_RADIUS}$
 //Speed = $100 * 50 * 60 = 83.33$



Chamfering

Chamfering can be used for motions such as straight lines, arcs, and spirals. For intuitive display, only straight-line corners are made in the routine.

Automatically calculate the actual corner radius size based on the corner angle. Limit to 50% when path is exceeded (example: move two placements (100,0), (0,100), the maximum chamfer radius is 50).

// test1.cpp: define the entry point of control panel application program

```
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"
#define PI 3.1415926

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "192.168.0.181"; //controller IP address, there may be error if checking
    the curve through the simulator, so it is better to connect to actual controller to check.
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        getchar();
        return -1;
    }
    printf("Success to connect controller!\n");

    ret = ZAux_Direct_SetAtype(handle, 0, 1); // set axis type of axis 0 as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret); // if it executes successfully
    ret = ZAux_Direct_SetAtype(handle, 1, 1); // set axis type of axis 1 as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret); // if it executes successfully

    ret = ZAux_Direct_SetUnits(handle, 0, 100); // set pulse amount of axis 0 as 100
    commandCheckHandler("ZAux_Direct_SetUnits", ret); // if it executes successfully
    ret = ZAux_Direct_SetUnits(handle, 1, 100); // set pulse amount of axis 1 as 100
    commandCheckHandler("ZAux_Direct_SetUnits", ret); // if it executes successfully

    ret = ZAux_Direct_SetAccel(handle, 0, 500); // set acceleration of axis 0
    commandCheckHandler("ZAux_Direct_SetAccel", ret); // if it executes successfully
    ret = ZAux_Direct_SetAccel(handle, 1, 500); // set acceleration of axis 1
    commandCheckHandler("ZAux_Direct_SetAccel", ret); // if it executes successfully

    ret = ZAux_Direct_SetDecel(handle, 0, 500); // set deceleration of axis 0
    commandCheckHandler("ZAux_Direct_SetDecel", ret); // if it executes successfully
    ret = ZAux_Direct_SetDecel(handle, 1, 500); // set deceleration of axis 1
    commandCheckHandler("ZAux_Direct_SetDecel", ret); // if it executes successfully

    ret = ZAux_Direct_SetDpos(handle, 0, 0); // set clear axis 0 DPOS (to 0)
    commandCheckHandler("ZAux_Direct_SetDpos", ret); // if it executes successfully
    ret = ZAux_Direct_SetDpos(handle, 1, 0); // set clear axis 1 DPOS (to 0)
    commandCheckHandler("ZAux_Direct_SetDpos", ret); // if it executes successfully

    ret = ZAux_Direct_SetSpeed(handle, 0, 100); // set speed of axis 0
    commandCheckHandler("ZAux_Direct_SetDecel", ret); // if it executes successfully

    ret = ZAux_Direct_SetSpeed(handle, 1, 100); // set speed of axis 1
    commandCheckHandler("ZAux_Direct_SetDecel", ret); // if it executes successfully

    ret = ZAux_Direct_SetMerge(handle, 0, 1); //open continuous interpolation (only main axis is
```

opened, for example, axis 0 and axis 1 interpolate, axis 0 is the main axis. The main axis is determined by the first axis No. in continuous interpolation motion command list)

```
commandCheckHandler("ZAux_Direct_SetMerge", ret) ;// if it executes successfully
```

ret =ZAux_Direct_SetCornerMode(handle,0,32);//set automatic chamfer (only main axis is opened, for example, axis 0 and axis 1 interpolate, axis 0 is the main axis. The main axis is determined by the first axis No. in continuous interpolation motion command list)

```
commandCheckHandler("ZAux_Direct_SetCornerMode", ret) ;// if it executes successfully
```

ret =ZAux_Direct_SetZsmooth(handle,0,50);//set chamfer radius (only main axis is opened, for example, axis 0 and axis 1 interpolate, axis 0 is the main axis. The main axis is determined by the first axis No. in continuous interpolation motion command list)

```
commandCheckHandler("ZAux_Direct_SetZsmooth", ret) ;// if it executes successfully
```

```
float IDLE;
```

```
int Axislist[2]={0,1};
```

```
float Dposlist[2]={0};
```

```
ZAux_Trigger( handle);// trigger the oscilloscope
```

```
Dposlist[0]=100;
```

```
Dposlist[1]=0;
```

```
ret =ZAux_Direct_Move(handle,2,Axislist,Dposlist);//linear interpolation
```

```
commandCheckHandler("ZAux_Direct_MoveCirc", ret) ;// if it executes successfully
```

```
Dposlist[0]=0;
```

```
Dposlist[1]=100;
```

```
ret =ZAux_Direct_Move(handle,2,Axislist,Dposlist);//linear interpolation
```

```
commandCheckHandler("ZAux_Direct_MoveCirc", ret) ;// if it executes successfully
```

```
while (1)// wait for axis 0 motion to complete
```

```
{
```

```
    Sleep(100);
```

```
    ZAux_Direct_GetParam(handle,"IDLE",0,&IDLE);
```

```
    if (IDLE<0)break;
```

```
}
```

```
getchar();
```

```
ret = ZAux_Close(handle); //close the connection
```

```
commandCheckHandler("ZAux_Close", ret) ;// if it executes successfully
```

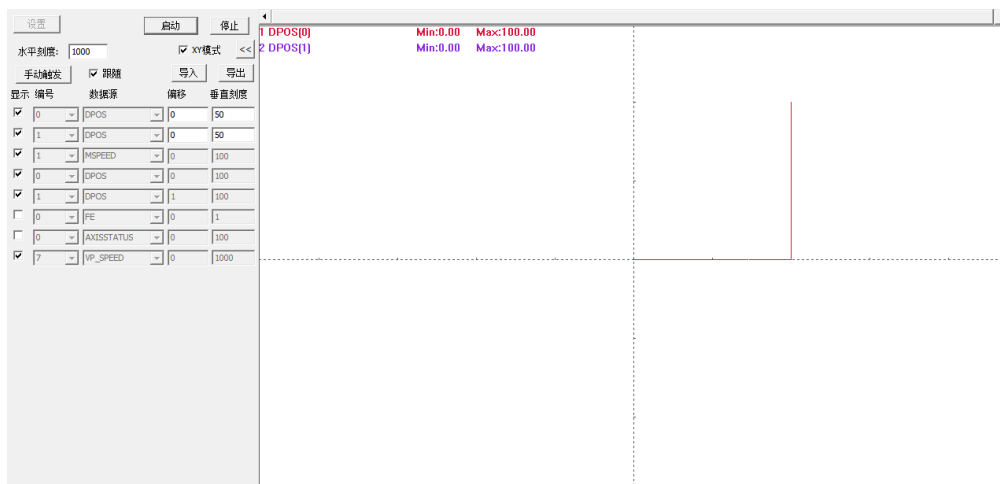
```
printf("connection closed!\n");
```

```
handle = NULL;
```

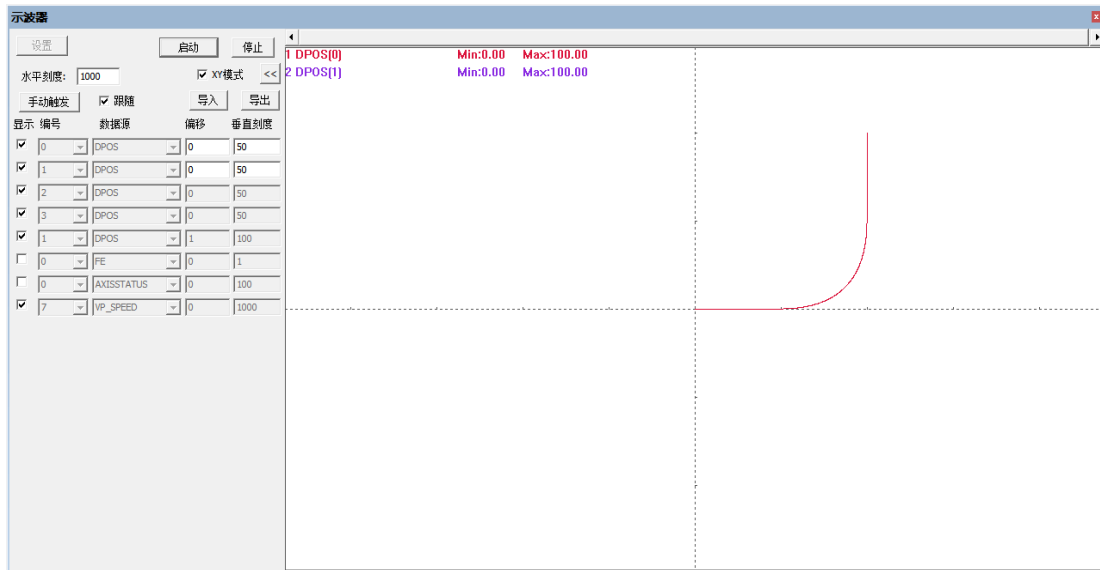
```
return 0;
```

```
}
```

➤ Interpolation trajectory, no chamfering processing:



- Interpolation trajectory, after using chamfer, chamfer radius is set as the max value 50 of this trajectory, trajectory waveform:



Axis Hold IN

When the axis 0 moves a SP point of 1500, if it meets the signal of IN (0) during the movement, the speed is automatically switched to the holding speed.

```
// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail! return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1"; // controller IP address
    ZMC_HANDLE handle = NULL; // link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); // connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        getchar();
        return -1;
    }
}
```



```
printf("Success to connect controller!\n");

ret = ZAux_Direct_SetAtype(handle,0,1); //set axis type as 1
commandCheckHandler("ZAux_Direct_SetAtype", ret) ;// if it executes successfully
ret = ZAux_Direct_SetAccel(handle,0,500); //set axis acceleration
commandCheckHandler("ZAux_Direct_SetAccel", ret) ;// if it executes successfully
ret = ZAux_Direct_SetDecel(handle,0,500); //set axis deceleration
commandCheckHandler("ZAux_Direct_SetDecel", ret) ;// if it executes successfully
ret = ZAux_Direct_SetDpos(handle,0,0); //set clear DPOS (to 0)
commandCheckHandler("ZAux_Direct_SetDpos", ret) ;// if it executes successfully

ret = ZAux_Direct_SetForceSpeed(handle,0,200); //set axis SP running speed
commandCheckHandler("ZAux_Direct_SetSpeed", ret) ;// if it executes successfully

ret = ZAux_Direct_SetFhspeed(handle,0,100); //set holding speed as 100
commandCheckHandler("ZAux_Direct_SetDpos", ret) ;// if it executes successfully

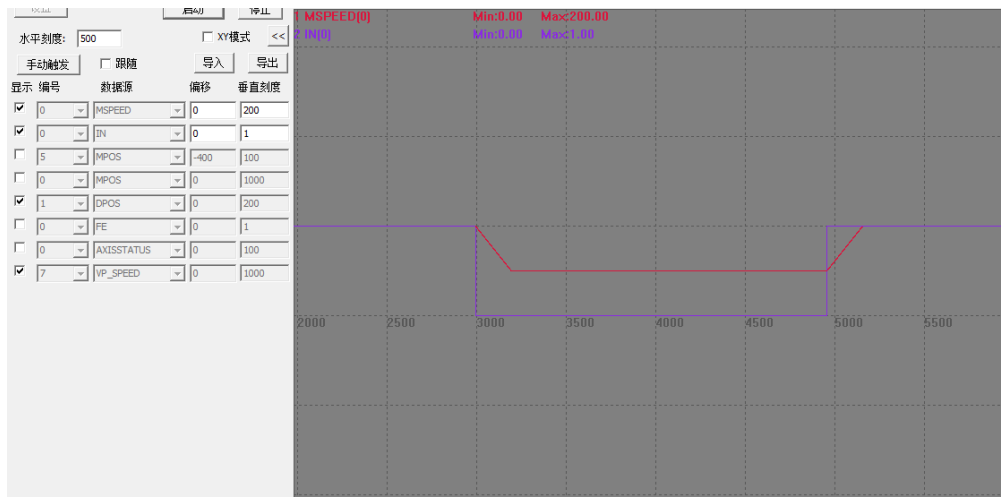
ret = ZAux_Direct_SetFholdIn(handle,0,0); //set holding speed, map IN as 0
commandCheckHandler("ZAux_Direct_SetDpos", ret) ;// if it executes successfully

ret = ZAux_Direct_SetInvertIn(handle,0,1); //invert the hold speed IN electric level.
commandCheckHandler("ZAux_Direct_SetDpos", ret) ;// if it executes successfully

float IDLE;
int axislist=0;
float dposlist=1500;
ZAux_Trigger( handle); //trigger the oscilloscope
ZAux_Direct_MoveSp(handle,1,&axislist,&dposlist); //axis 0 moves 1500
while (1) //wait for axis 0 to complete the motion
{
    Sleep(100);
    ZAux_Direct_GetParam(handle,"IDLE",0,&IDLE);
    if (IDLE<0) break;
}

getchar();
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;// if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}
```

➤ Speed Curve: MSPEED (encoder feedback real-time speed)



Continuous Interpolation with SP Speed Control

```
// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1"; //controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        getchar();
        return -1;
    }
    printf("Success to connect controller!\n");

    ret = ZAux_Direct_SetAtype(handle, 0, 1); //set axis type as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret); // if it executes successfully
    ret = ZAux_Direct_SetAccel(handle, 0, 500); //set axis acceleration
    commandCheckHandler("ZAux_Direct_SetAccel", ret); // if it executes successfully
    ret = ZAux_Direct_SetDecel(handle, 0, 500); //set axis deceleration
    commandCheckHandler("ZAux_Direct_SetDecel", ret); // if it executes successfully
    ret = ZAux_Direct_SetDpos(handle, 0, 0); //clear DPOS to 0
    commandCheckHandler("ZAux_Direct_SetDpos", ret); // if it executes successfully

    ret = ZAux_Direct_SetMerge(handle, 0, 1); //open continuous interpolation
    commandCheckHandler("ZAux_Direct_SetMerge", ret); // if it executes successfully
    float IDLE;

    int axislist = 0;
    float dposlist = 40;
    ZAux_Trigger(handle); //trigger the oscilloscope
    //the first segment
    ret = ZAux_Direct_SetForceSpeed(handle, 0, 30); //set axis SP running speed
    commandCheckHandler("ZAux_Direct_SetSpeed", ret); // if it executes successfully

    ret = ZAux_Direct_SetStartMoveSpeed(handle, 0, 1000); //cancel setting axis starting running
    speed, the set value should be more than ForceSpeed.
    commandCheckHandler("ZAux_Direct_SetStartMoveSpeed", ret); // if it executes successfully

    ret = ZAux_Direct_SetEndMoveSpeed(handle, 0, 1000); // cancel setting axis end running speed,
    the set value should be more than ForceSpeed.
    commandCheckHandler("ZAux_Direct_SetEndMoveSpeed", ret); // if it executes successfully

    ZAux_Direct_MoveSp(handle, 1, &axislist, &dposlist); //axis 0 moves 40
    //the second segment
```

```

ret =ZAux_Direct_SetForceSpeed(handle,0,50);//set axis SP running speed
commandCheckHandler("ZAux_Direct_SetSpeed", ret) ;// if it executes successfully

ret =ZAux_Direct_SetStartMoveSpeed(handle,0,20);//set starting running speed as 20
commandCheckHandler("ZAux_Direct_SetStartMoveSpeed", ret) ;// if it executes successfully

ret =ZAux_Direct_SetEndMoveSpeed(handle,0,40);//set axis end running speed as 40
commandCheckHandler("ZAux_Direct_SetEndMoveSpeed", ret) ;// if it executes successfully

dposlist=50;
ZAux_Direct_MoveSp(handle,1,&axislist,&dposlist);//axis 0 moves 50
//the third segment
ret =ZAux_Direct_SetForceSpeed(handle,0,60);//set axis SP running speed
commandCheckHandler("ZAux_Direct_SetSpeed", ret) ;// if it executes successfully

ret =ZAux_Direct_SetStartMoveSpeed(handle,0,1000);// cancel setting axis starting running
speed, the set value should be more than ForceSpeed.
commandCheckHandler("ZAux_Direct_SetStartMoveSpeed", ret) ;// if it executes successfully

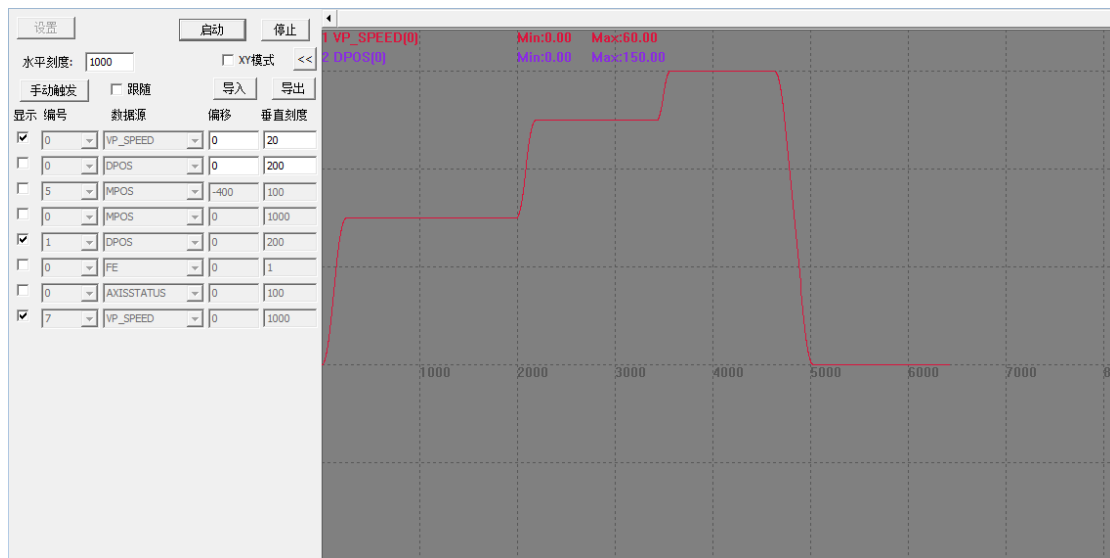
ret =ZAux_Direct_SetEndMoveSpeed(handle,0,1000);// cancel setting axis end running speed,
the set value should be more than ForceSpeed.
commandCheckHandler("ZAux_Direct_SetEndMoveSpeed", ret) ;// if it executes successfully

dposlist=60;
ZAux_Direct_MoveSp(handle,1,&axislist,&dposlist);//axis 0 moves 60

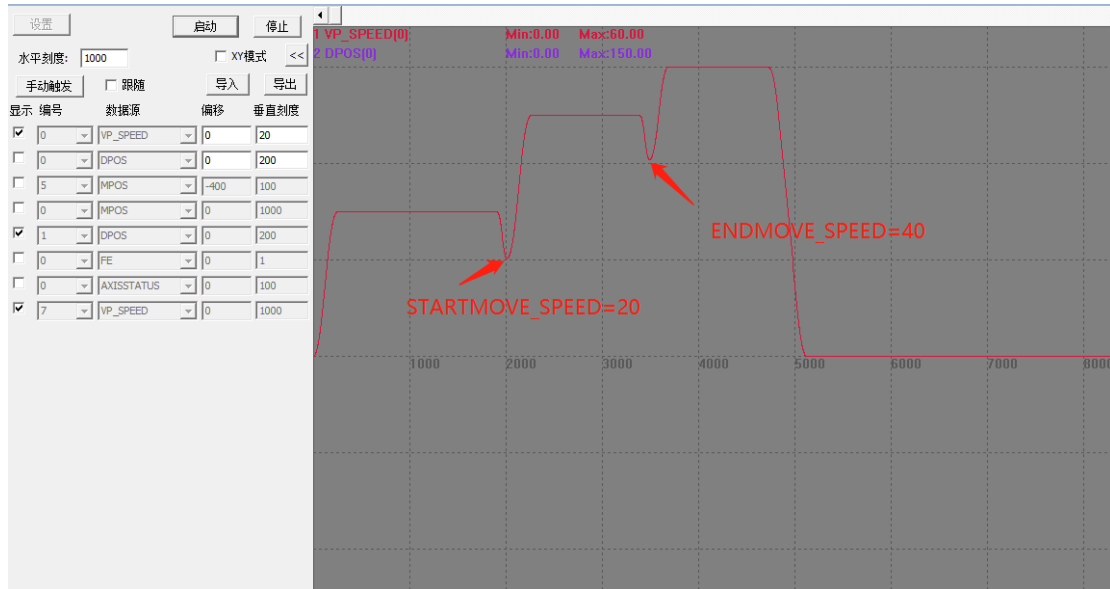
getchar();
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;// if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}

```

➤ When SP speed is not used:



➤ When SP speed is used:



Multi-Axis Linear Interpolation

```
// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1"; //simulator IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");

    ret = ZAux_Direct_SetAtype(handle, 0, 1); //set axis type of axis 0 as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret); // if it executes successfully

    ret = ZAux_Direct_SetAtype(handle, 1, 1); // set axis type of axis 1 as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret); // if it executes successfully
}
```

```

ret = ZAux_Direct_SetAtype(handle,2,1); // set axis type of axis 2 as 1
commandCheckHandler("ZAux_Direct_SetAtype", ret) ;// if it executes successfully

ret = ZAux_Direct_Single_Cancel(handle, 0, 0); //axis 0 motion stops
commandCheckHandler("ZAux_Direct_Single_Cancel", ret);
ret = ZAux_Direct_Single_Cancel(handle, 1, 0); // axis 1 motion stops
commandCheckHandler("ZAux_Direct_Single_Cancel", ret);
ret = ZAux_Direct_Single_Cancel(handle, 2, 0); // axis 2 motion stops
commandCheckHandler("ZAux_Direct_Single_Cancel", ret);

ret = ZAux_Direct_SetDpos(handle, 0, 0); //set axis 0 dpos
commandCheckHandler("ZAux_Direct_SetDpos", ret);
ret = ZAux_Direct_SetDpos(handle, 1, 0); // set axis 1 dpos
commandCheckHandler("ZAux_Direct_SetDpos", ret);
ret = ZAux_Direct_SetDpos(handle, 2, 0); // set axis 2 dpos
commandCheckHandler("ZAux_Direct_SetDpos", ret);

ret = ZAux_Direct_SetMerge(handle, 0,1); //set continuous interpolation
commandCheckHandler("ZAux_Direct_SetMerge", ret) ;// if it executes successfully

int axislist[3] = { 1,0,2 }; //motion axis list, axis 1 is the master axis
float poslist[3] = { 100,200,300 }; //motion list, axis 1-100, axis 0-200, axis 2-300
ZAux_Trigger( handle); //trigger the oscilloscope

ret = ZAux_Direct_SetSpeed(handle, axislist[0], 100); //set interpolation speed 100, the
setting is on the master axis
commandCheckHandler("ZAux_Direct_SetSpeed", ret);

ret = ZAux_Direct_SetAccel(handle, axislist[0], 500); // set interpolation acceleration 500, the
setting is on the master axis commandCheckHandler("ZAux_Direct_SetAccel", ret);

ret = ZAux_Direct_SetDecel(handle, axislist[0], 500); // set interpolation deceleration 500, the
setting is on the master axis commandCheckHandler("ZAux_Direct_SetDecel", ret);

ret = ZAux_Direct_MoveAbs(handle, 3, axislist, poslist); //call the motion, axis0, 1, 2 move to
corresponding absolute position
commandCheckHandler("ZAux_Direct_MoveAbs", ret);

int state[3];

do
{
    ZAux_Direct_GetIfIdle(handle, axislist[0], state);
    ZAux_Direct_GetIfIdle(handle, axislist[1], state + 1);
    ZAux_Direct_GetIfIdle(handle, axislist[2], state + 2);

    if (!(state[0] || state[1] || state[2]))
    {
        Sleep(100);
    }
    else
    {
        break;
    }
} while (true); //wait for the motion to stop

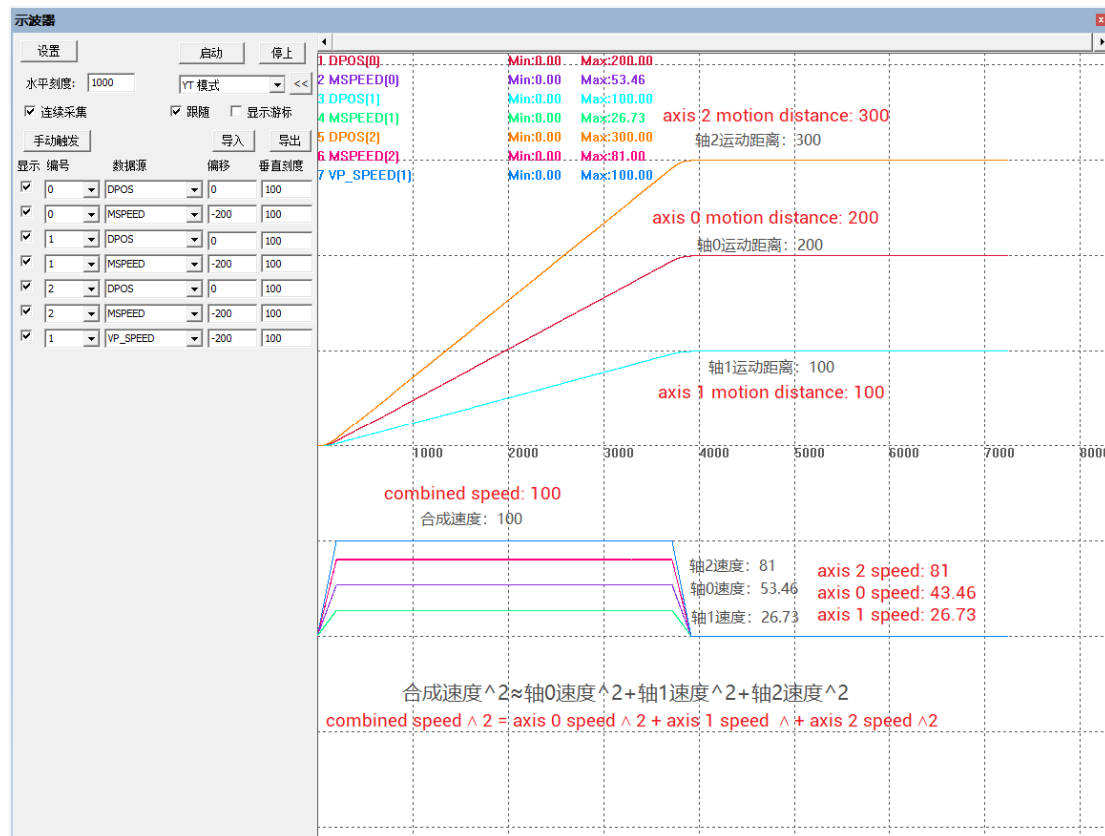
printf("Idle\n");

Sleep(2000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;// if it executes successfully

```

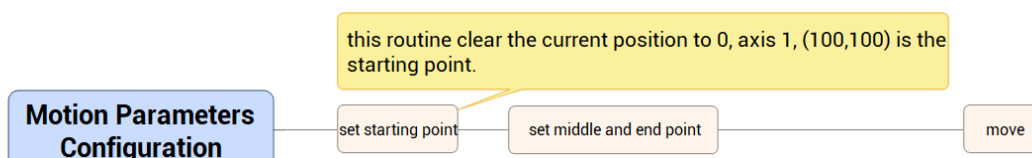
```
printf("connection closed!\n");
handle = NULL;
return 0;
}
```

- The figure shows the movement speed changes of axes 0, 1, and 2, and the meaning of the line segment is shown in the figure.



Draw the Plane Arc by 3-Point

This routine uses 2 axes to do planar circular arc movement, and completes it by 3 points: starting point (100, 100), end point (300, 100), middle point (200, 200).



```
// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"
```

```

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1"; //simulator IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");

    float UnitsValue = 100;
    float DposValue = 0;
    float SpeedValue = 100;
    float AccelValue = 10000;
    float DecelValue = 10000;
    int AxisList[2] = { 0,1 };
    float DiatanceValue = 100;

    ret = ZAux_Direct_SetAtype(handle,0,1); //set axis type of axis 0 as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret); // if it executes successfully

    ret = ZAux_Direct_SetAtype(handle,1,1); // set axis type of axis 1 as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret); // if it executes successfully

    ret = ZAux_Direct_SetUnits(handle, AxisList[0], UnitsValue); //set pulse amount of axis 0
    commandCheckHandler("ZAux_Direct_SetUnits", ret);

    ret = ZAux_Direct_SetUnits(handle, AxisList[1], UnitsValue); // set pulse amount of axis 1
    commandCheckHandler("ZAux_Direct_SetUnits", ret);

    ret = ZAux_Direct_SetSpeed(handle, AxisList[0], SpeedValue); //set speed of axis 0
    commandCheckHandler("ZAux_Direct_SetSpeed", ret);

    ret = ZAux_Direct_SetSpeed(handle, AxisList[1], SpeedValue); //set speed of axis 1
    commandCheckHandler("ZAux_Direct_SetSpeed", ret);

    ret = ZAux_Direct_SetAccel(handle, AxisList[0], AccelValue); //set acceleration of axis 0
    commandCheckHandler("ZAux_Direct_SetAccel", ret);

    ret = ZAux_Direct_SetAccel(handle, AxisList[1], AccelValue); // set acceleration of axis 1
    commandCheckHandler("ZAux_Direct_SetAccel", ret);

    ret = ZAux_Direct_SetDecel(handle, AxisList[0], DecelValue); //set deceleration of axis 0
    commandCheckHandler("ZAux_Direct_SetDecel", ret);

    ret = ZAux_Direct_SetDecel(handle, AxisList[1], DecelValue); // set deceleration of axis 1
    commandCheckHandler("ZAux_Direct_SetDecel", ret);

    ret = ZAux_Direct_SetDpos(handle, AxisList[0], 0); //set axis 0 dpos as 0
    commandCheckHandler("ZAux_Direct_SetDpos", ret);

```

```

ret = ZAux_Direct_SetDpos(handle, AxisList[1], 0); // set axis 1 dpos as 0
commandCheckHandler("ZAux_Direct_SetDpos", ret);

ret = ZAux_Direct_Single_Cancel(handle, 0, 0); //axis 0 motion stops
commandCheckHandler("ZAux_Direct_Single_Cancel", ret);
ret = ZAux_Direct_Single_Cancel(handle, 1, 0); //axis 1 motion stops
commandCheckHandler("ZAux_Direct_Single_Cancel", ret);
ret = ZAux_Direct_Single_Cancel(handle, 2, 0); //axis 2 motion stops
commandCheckHandler("ZAux_Direct_Single_Cancel", ret);

int run_state = 0; //axis motion state

ZAux_Trigger( handle); //trigger the oscilloscope

//move to starting position (100,100)
ret = ZAux_Direct_Single_Move(handle, AxisList[0], DistanceValue);
commandCheckHandler("ZAux_Direct_Single_Move", ret);

ret = ZAux_Direct_Single_Move(handle, AxisList[1], DistanceValue);
commandCheckHandler("ZAux_Direct_Single_Move", ret);
//move to starting position (100,100)
int state[2];
do
{
    ret = ZAux_Direct_GetIfIdle(handle, AxisList[0], state);
    commandCheckHandler("ZAux_Direct_GetIfIdle", ret);

    ret = ZAux_Direct_GetIfIdle(handle, AxisList[1], state + 1);
    commandCheckHandler("ZAux_Direct_GetIfIdle", ret);

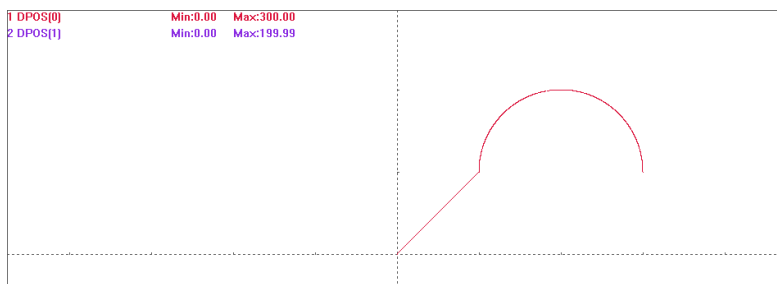
    if (!(state[0] || state[1] ))
    {
        Sleep(100);
    }
    else
    {
        break;
    }
} while (true); //wait for motion to stop

//3 point draw an arc, starting point (100, 100), the middle point (100,100), end point (300,100)
ret = ZAux_Direct_MoveCirc2(handle, 2, AxisList, 200-100, 200-100, 300-100, 100-100);
commandCheckHandler("ZAux_Direct_MoveCirc2", ret);

Sleep(2000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret); // if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}

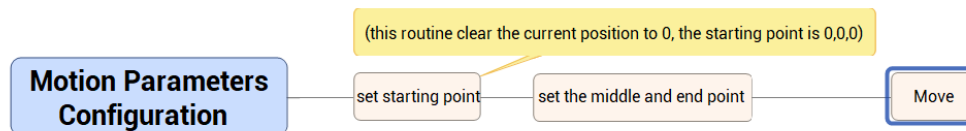
```

➤ XY plane result:



Draw Space Arc by 3-Point

This routine uses 3 axes to do space arc movement, also, it is completed by 3 points: the current position point (as starting point (0, 0, 0)), the end point (120, 160, 400), the middle point (240, 320, 300).



```

// test1.cpp: define the entry point of control panel application program
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1"; //simulator IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");

    ret = ZAux_Direct_SetAtype(handle, 0, 1); //set axis type of axis 0 as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret); // if it executes successfully

    ret = ZAux_Direct_SetAtype(handle, 1, 1); //set axis type of axis 1 as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret); // if it executes successfully

    ret = ZAux_Direct_SetAtype(handle, 2, 1); //set axis type of axis 2 as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret); // if it executes successfully

    ret = ZAux_Direct_SetSpeed(handle, 0, 200); //set speed of main axis 0 as 200units/s
    commandCheckHandler("ZAux_Direct_SetSpeed", ret);
    ret = ZAux_Direct_SetAccel(handle, 0, 2000); // set acceleration main axis 0 as 2000units/s/s
    commandCheckHandler("ZAux_Direct_SetAccel", ret);
    ret = ZAux_Direct_SetDecel(handle, 0, 2000); // set deceleration of axis 0 as 2000units/s/s
    commandCheckHandler("ZAux_Direct_SetDecel", ret);

    ret = ZAux_Direct_Single_Cancel(handle, 0, 0); //axis 0 motion stops
  
```

```
commandCheckHandler("ZAux_Direct_Single_Cancel", ret);
ret = ZAux_Direct_Single_Cancel(handle, 1, 0); //axis 1 motion stops
commandCheckHandler("ZAux_Direct_Single_Cancel", ret);
ret = ZAux_Direct_Single_Cancel(handle, 2, 0); //axis 2 motion stops
commandCheckHandler("ZAux_Direct_Single_Cancel", ret);

int run_state = 0; //axis motion state

ret = ZAux_Direct_SetDpos(handle, 0, 0); //set axis 0 dpos as 0
commandCheckHandler("ZAux_Direct_SetDpos", ret);
ret = ZAux_Direct_SetDpos(handle, 1, 0); //set axis 1 dpos as 0
commandCheckHandler("ZAux_Direct_SetDpos", ret);
ret = ZAux_Direct_SetDpos(handle, 2, 0); //set axis 2 dpos as 0
commandCheckHandler("ZAux_Direct_SetDpos", ret);

ZAux_Trigger( handle); //trigger the oscilloscope

//the list of axes that participate in the motion
int axisList[3] = { 0,1,2};

//starting point (0,0,0), end point (120, 160, 400), the middle point (240, 320, 300), mode 0,
these three points draw the arc in space.
ret = ZAux_Direct_MSpherical(handle, 3, axisList, 120, 160, 400, 240, 320, 300, 0, 0, 0);
commandCheckHandler("ZAux_Direct_MSpherical", ret);

int state[3];

do
{
    ret = ZAux_Direct_GetIfIdle(handle, axisList[0], state);
    commandCheckHandler("ZAux_Direct_GetIfIdle", ret);

    ret = ZAux_Direct_GetIfIdle(handle, axisList[1], state + 1);
    commandCheckHandler("ZAux_Direct_GetIfIdle", ret);

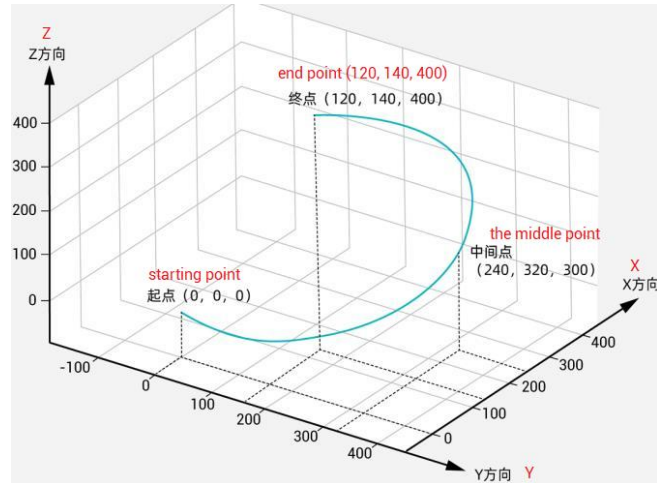
    ret = ZAux_Direct_GetIfIdle(handle, axisList[2], state + 2);
    commandCheckHandler("ZAux_Direct_GetIfIdle", ret);

    if (!(state[0] || state[1] || state[2]))
    {
        Sleep(100);
    }
    else
    {
        break;
    }
} while (true); //wait for motion to stop

printf("Idle\n");

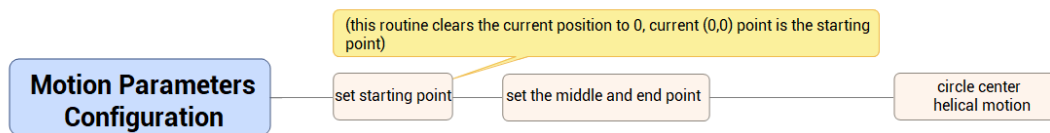
Sleep(2000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;// if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}
```

➤ Trajectory Effect:



Helical Curve

This routine uses 3 axes to do space arc movement, and completes it by 3 points: the current position point / the origin (as the starting point (0, 0)), the end point (200, -200), and the middle point (200, 0). And the Z axis participates in the movement, it moves 100, makes a clockwise spiral movement.



```

// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1"; //simulator IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAUX_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
    }
}
  
```

```

        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");

    ret = ZAux_Direct_SetAtype(handle,0,1);//set axis type of axis 0 as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret) ;// if it executes successfully

    ret = ZAux_Direct_SetAtype(handle,1,1);//set axis type of axis 1 as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret) ;// if it executes successfully

    ret = ZAux_Direct_SetAtype(handle,2,1);//set axis type of axis 2 as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret) ;// if it executes successfully

    ret = ZAux_Direct_Single_Cancel(handle, 0, 0); //axis 0 motion stops
    commandCheckHandler("ZAux_Direct_Single_Cancel", ret);
    ret = ZAux_Direct_Single_Cancel(handle, 1, 0); //axis 1 motion stops
    commandCheckHandler("ZAux_Direct_Single_Cancel", ret);
    ret = ZAux_Direct_Single_Cancel(handle, 2, 0); //axis 2 motion stops
    commandCheckHandler("ZAux_Direct_Single_Cancel", ret);

    int run_state = 0; //axis motion state

    ret = ZAux_Direct_SetDpos(handle, 0, 0); //set axis 0 dpos as 0
    commandCheckHandler("ZAux_Direct_SetDpos", ret);
    ret = ZAux_Direct_SetDpos(handle, 1, 0); //set axis 1 dpos as 0
    commandCheckHandler("ZAux_Direct_SetDpos", ret);
    ret = ZAux_Direct_SetDpos(handle, 2, 0); //set axis 2 dpos as 0
    commandCheckHandler("ZAux_Direct_SetDpos", ret);

    ret = ZAux_Direct_SetSpeed(handle, 0, 100); //set master axis 0 speed as 100units/s
    commandCheckHandler("ZAux_Direct_SetSpeed", ret);
    ret = ZAux_Direct_SetAccel(handle, 0, 10000); // set axis 0 acceleration as 10000units/s/s
    commandCheckHandler("ZAux_Direct_SetAccel", ret);
    ret = ZAux_Direct_SetDecel(handle, 0, 10000); // set axis 0 deceleration as 10000units/s/s
    commandCheckHandler("ZAux_Direct_SetDecel", ret);

    //the list of axes that participate in motion
    int axisList[3] = { 0,1,2 };
    ZAux_Trigger( handle);//trigger the oscilloscope

    //the origin is used as the starting point, the center is (200,0), the end point is (200,-200), in
    clockwise, the Z axis participates in the speed calculation, the movement is 100, and then do the
    circle center spiral movement
    ret = ZAux_Direct_MHelical(handle, 3, axisList, 200,-200, 200, 0, 1, 100,0);
    commandCheckHandler("ZAux_Direct_MoveCirc", ret);

    int state[3];
    do
    {
        ret = ZAux_Direct_GetIfIdle(handle, axisList[0], state);
        commandCheckHandler("ZAux_Direct_GetIfIdle", ret);

        ret = ZAux_Direct_GetIfIdle(handle, axisList[1], state + 1);
        commandCheckHandler("ZAux_Direct_GetIfIdle", ret);

        ret = ZAux_Direct_GetIfIdle(handle, axisList[2], state + 2);
        commandCheckHandler("ZAux_Direct_GetIfIdle", ret);

        if (!(state[0] || state[1] || state[2]))
        {
            Sleep(100);
        }
    }
    else

```

```

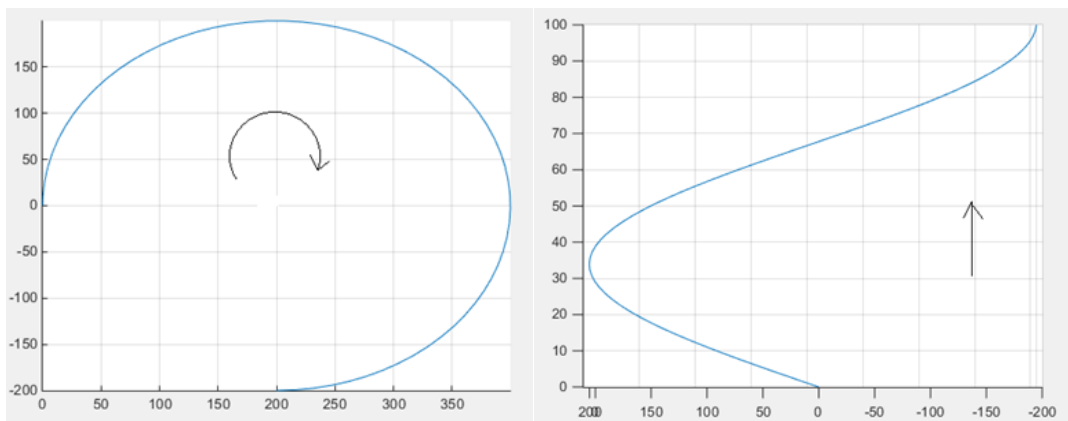
    {
        break;
    }
} while (true); //wait for the motion to stop

printf("Idle\n");

Sleep(2000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;// if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}

```

➤ Interpolation Trajectory:



6.6. Axis Pause / Resume / Stop

6.6.1. Instruction List

Commands	Descriptions
ZAux_Direct_MovePause	Pause the motion, for interpolation, it pauses the spindle
ZAux_Direct_MoveResume	Cancel motion paused
ZAux_Direct_Rapidstop	Stop all axes immediately
ZAux_Direct_CancelAxisList	Stop multi-axis
ZAux_Direct_Single_Cancel	Stop single-axis

6.6.2. Principle & Description

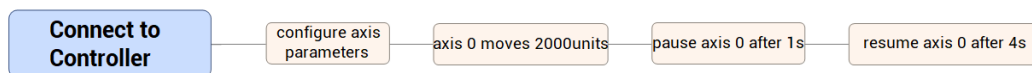
During single-axis or multi-axis interpolation motion, the motion control card supports functions such as motion pause and axis stop.

Note: when using the axis stop function, the movement will stop at the fast deceleration (**ZAux_Direct_SetFastDec**). If the fast deceleration is not set, the fast deceleration will be equal to the deceleration (**ZAux_Direct_SetDecel**), therefore, deceleration or fast deceleration must be set when using axis stop function. What's more, ZAux_Direct_Single_Cancel mode 3 cannot be used to stop the superimposed axis, otherwise it may cause the position of the superimposed axis to be inconsistent.

6.6.3. Routine

Axis Pause & Resume

This routine makes axis 0 do single-axis motion, it moves 2000 units in positive direction. Make it pause after moving 1s, and resume it after 4s.



```
// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1"; // controller IP address
    ZMC_HANDLE handle = NULL; // link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); // connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
    }
}
```

```

        handle = NULL;
        getchar();
        return -1;
    }
    printf("Success to connect controller!\n");

    ZAux_Direct_SetAtype(handle, 0, 1); //set axis type of axis 0 as 1
    ZAux_Direct_SetUnits(handle, 0, 100); //set pulse amount of axis 0 as 100
    ZAux_Direct_SetSpeed(handle, 0, 200); //set speed of axis 0 as 200units/s
    ZAux_Direct_SetAccel(handle, 0, 2000); //set acceleration of axis 0 as 2000units/s/s
    ZAux_Direct_SetDecel(handle, 0, 2000); //set deceleration of axis 0 as 2000units/s/s
    ZAux_Direct_SetSramp(handle, 0, 200); //set S curve of axis 0 as 200ms

    ret = ZAux_Direct_SetDpos( handle, 0, 0); //clear axis command position to 0
    commandCheckHandler("ZAux_Direct_SetDpos", ret); //if it executes successfully
    ret = ZAux_Direct_SetMpos( handle, 0, 0); //clear encoder feedback position to 0
    commandCheckHandler("ZAux_Direct_SetMpos", ret); //if it executes successfully
    int GetVal;
    ret = ZAux_Direct_Single_Move( handle, 0, 2000);
    commandCheckHandler("ZAux_Direct_SetDpos", ret); //if it executes successfully

    Sleep(1000); //wait 1s
    ZAux_Direct_MovePause(handle, 0, 0); //pause the current motion of axis 0

    Sleep(2000); //wait 2s

    ZAux_Direct_GetAxisStatus(handle, 0, &GetVal); //get the axis stop reason
    printf("current axis state: %x h\n", GetVal);

    Sleep(2000); //wait 2s
    ZAux_Direct_MoveResume(handle, 0); //resume axis 0 motion

    float IDLE;
    while (1) //wait for axis 0 to complete the motion
    {
        Sleep(100);
        ZAux_Direct_GetParam(handle, "IDLE", 0, &IDLE);
        if (IDLE < 0) break;
    }

    Sleep(2000);
    ret = ZAux_Close(handle); //close the connection
    commandCheckHandler("ZAux_Close", ret); //if it executes successfully
    printf("connection closed!\n");
    handle = NULL;
    return 0;
}

```

Chapter VII How to Control Other Motions?

7.1. Coordinates Loop

7.1.1. Instruction List

Commands	Descriptions
ZAux_Direct_SetRepDist	Set loop range
ZAux_Direct_GetRepDist	Read loop range
ZAux_Direct_SetRepOption	Set coordinates loop mode
ZAux_Direct_GetRepOption	Read coordinates loop mode

7.1.2. Principle & Description

The instructions in this chapter can be used to limit the coordinate cycle range of the cam main axis to realize the continuity of multiple cam curves.

When using the absolute motion mode, if the target position is within the range of the coordinate cycle, it can move correctly, if it is outside the range of the coordinate cycle, the motion is incorrect.

Relative motion will not be influenced.

The main command: **ZAux_Direct_SetRepOption** (coordinate cycle mode):

Bit	Value	Descriptions
0	1	0 – loop range: -coordinate loop position (ZAux_Direct_SetRepDist) ~ + coordinate loop position (ZAux_Direct_SetRepDist) 1 – loop range: 0 ~ coordinate loop position (ZAux_Direct_SetRepDist)
1	2	The repeat motion from this position 1 cam table following motion (ZAux_Direct_Cambox) to automatic cam (ZAux_Direct_MoveLink), and it restores to 0 after disabled.
2	4	Reserved
4	16	1 – not to use ZAux_Direct_SetRepDist . 0 – use ZAux_Direct_SetRepDist .

7.1.3. Routine

Coordinates Loop

This routine shows coordinates loop motion, axis 0 starts to do loop motion from the origin, the loop is from -100 to 100, and the direct loop from 0 to 100.

Configure Axis
Parameters

Set loop position 0-100/-100-100

continuous
motion

```
// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail! return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1"; // controller IP address
    ZMC_HANDLE handle = NULL; // link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); // connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        getchar();
        return -1;
    }
    printf("Success to connect controller!\n");

    ret = ZAux_Direct_SetAtype(handle, 0, 1); // set axis type as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret); // if it executes successfully
    ret = ZAux_Direct_SetAccel(handle, 0, 500); // set axis acceleration
    commandCheckHandler("ZAux_Direct_SetAccel", ret); // if it executes successfully
    ret = ZAux_Direct_SetDecel(handle, 0, 500); // set axis deceleration
    commandCheckHandler("ZAux_Direct_SetDecel", ret); // if it executes successfully
    ret = ZAux_Direct_SetDpos(handle, 0, 0); // set clear axis DPOS to 0
    commandCheckHandler("ZAux_Direct_SetDpos", ret); // if it executes successfully

    ret = ZAux_Direct_SetSpeed(handle, 0, 250); // set axis speed
    commandCheckHandler("ZAux_Direct_SetDecel", ret); // if it executes successfully

    ret = ZAux_Direct_SetRepDist(handle, 0, 100); // set loop coordinates as 100
```

```
commandCheckHandler("ZAux_Direct_SetRepDist", ret) ;//if it executes successfully
ret = ZAux_Direct_SetRepOption(handle,0,1); //set loop mode as 1
commandCheckHandler("ZAux_Direct_SetDpos", ret) ;//if it executes successfully

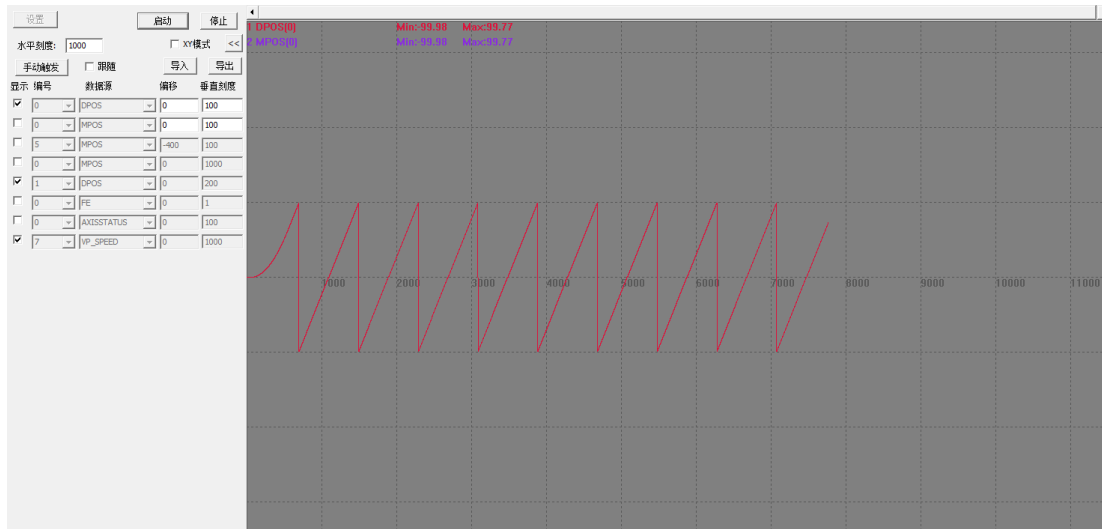
float IDLE;
int mode;
int AxisStatus;
ZAux_Trigger( handle); //trigger the oscilloscope

ZAux_Direct_Single_Vmove(handle,0,1); //axis 0 keeps moving

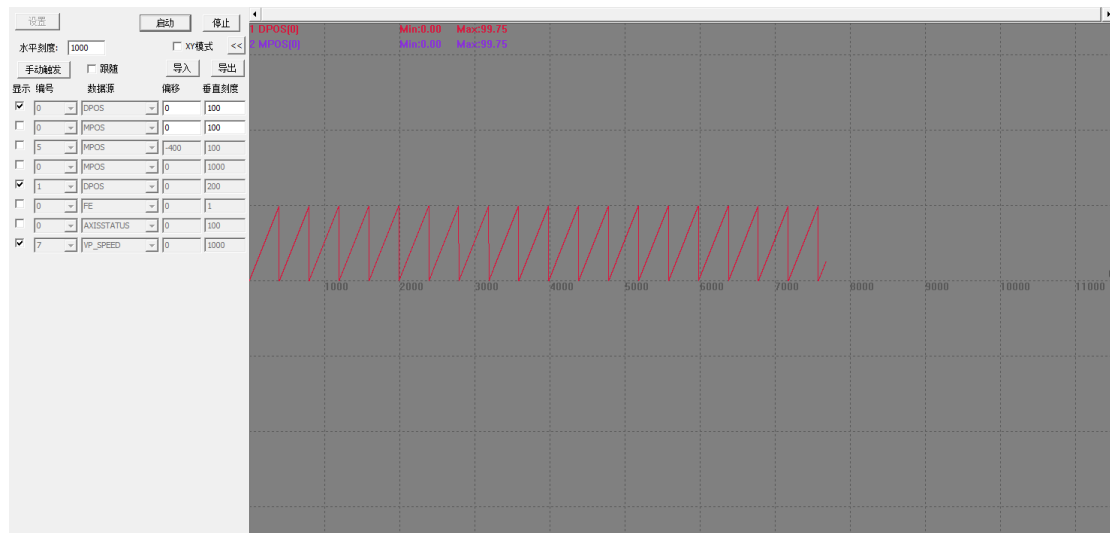
while (1) //wait for axis 0 to complete the motion
{
    Sleep(100);
    ZAux_Direct_GetParam(handle,"IDLE",0,&IDLE);
    if (IDLE<0) break;
}

getchar();
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}
```

- (coordinates loop mode) = 0, it loops from -100 to 100.



- **ZAux_Direct_SetRepOption** (coordinates loop mode) = 1, it loops from 0 to 100.



7.2. Pitch Compensation

7.2.1. Instruction List

Commands	Descriptions
ZAux_Direct_Pitchset	Set axis pitch compensation, it is invalid for extended axes
ZAux_Direct_Pitchset2	Set axis pitch dual-compensation, it is invalid for extended axes
ZAux_Direct_GetPitchStatus	Read axis pitch compensation state, it is invalid for extended axes

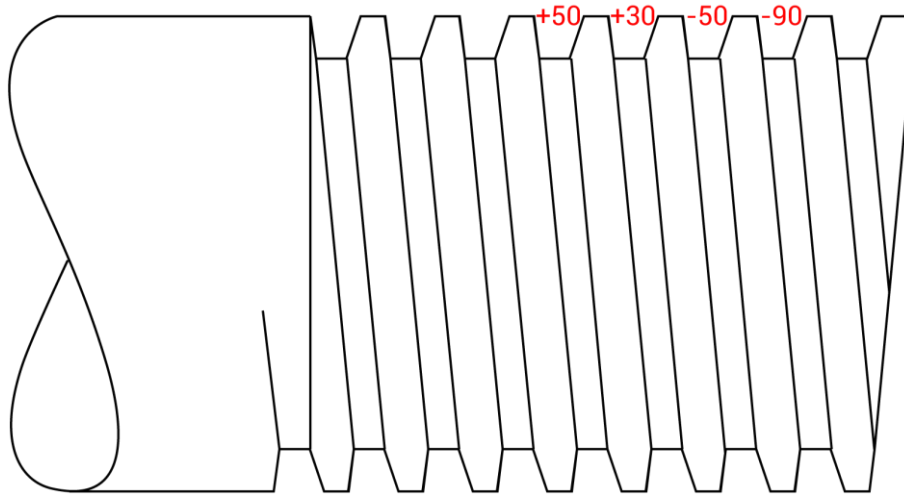
7.2.2. Principle & Description

For open-loop and semi-closed-loop systems, the positioning accuracy is largely affected by the accuracy of the ball screw. Although the accuracy of the ball screw is high, there are always manufacturing errors. In order to obtain a motion accuracy that exceeds the accuracy of the ball screw, the pitch error compensation function must be used in the system, that is, the system is used to compensate and correct the error to ensure the machining accuracy.

The instructions in this chapter are mainly functions of the pitch compensation, which are mainly used when the equipment mechanism has manufacturing errors, then the pitch compensation can be used to ensure the accuracy.

7.2.3. Routine

Pitch Compensation



There is a slight error in the pitch of each circle, and it is not strictly uniform, so pitch compensation is required for adjustment in high-precision applications.

Here drives axis 0 to move back and forth with a distance of 700, and the corresponding pulse needs to be compensated in the following list.

Starting compensation point position (MPOS)	Compensation value (the number of pulses when the distance is 100)
100	0
200	50 pulses
300	30 pulses
400	-50 pulses
500	-90 pulses
600	0

```
// test1.cpp: define the entry point of control panel application program
//
```

```
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"
```

```
void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail! return code is %d\n", command, ret);
    }
}
```

```

}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"192.168.0.11";    //controller IP address
    ZMC_HANDLE handle = NULL;                //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        getchar();
        return -1;
    }
    printf("Success to connect controller!\n");

    ret = ZAux_Direct_SetAtype(handle,0,1); //set axis type as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret) ;//if it executes successfully
    ret = ZAux_Direct_SetAtype(handle,5,6); //set axis type as 6
    commandCheckHandler("ZAux_Direct_SetAtype", ret) ;//if it executes successfully

    ret = ZAux_Direct_SetAccel(handle,0,500); //set axis acceleration
    commandCheckHandler("ZAux_Direct_SetAccel", ret) ;//if it executes successfully
    ret = ZAux_Direct_SetDecel(handle,0,500); //set axis deceleration
    commandCheckHandler("ZAux_Direct_SetDecel", ret) ;//if it executes successfully
    ret = ZAux_Direct_SetDpos(handle,0,0); //set clear axis DPOS to 0
    commandCheckHandler("ZAux_Direct_SetDpos", ret) ;//if it executes successfully
    ret = ZAux_Direct_SetMpos(handle,0,0); //set clear axis MPOS to 0
    commandCheckHandler("ZAux_Direct_SetMpos", ret) ;//if it executes successfully

    ret = ZAux_Direct_SetDpos(handle,5,0); // set clear axis DPOS to 0
    commandCheckHandler("ZAux_Direct_SetDpos", ret) ;//if it executes successfully
    ret = ZAux_Direct_SetMpos(handle,5,0); // set clear axis MPOS to 0
    commandCheckHandler("ZAux_Direct_SetMpos", ret) ;//if it executes successfully

    ret = ZAux_Direct_SetSpeed(handle,0,250); //set axis speed
    commandCheckHandler("ZAux_Direct_SetDecel", ret) ;//if it executes successfully

    float numlist[4]={50,100,50,25};

    //the position of starting compensation (MPOS)
    //compensation value (the number of pulses when the distance is 100)
    // 100 0
    // 200 50 pulses
    // 300 30 pulses
    // 400 -50 pulses
    // 500 -90 pulses
    // 600 0
    ret = ZAux_Direct_Pitchset(handle,0,1,100,4,100,0,numlist); //set pitch compensation
    commandCheckHandler("ZAux_Direct_Pitchset", ret) ;//if it executes successfully

    float IDLE;
    int mode;
    int AxisStatus;
    ZAux_Trigger( handle); //trigger the oscilloscope

    ZAux_Direct_Single_Move(handle,0,700); //axis 0 moves 700

    while (1) //wait for axis 0 to complete the motion
    {
        Sleep(100);
        ZAux_Direct_GetParam(handle,"IDLE",0,&IDLE);
        if (IDLE<0)break;
    }
}

```

```

}
ZAux_Direct_Single_Move(handle,0,-700); //axis 0 moves -700

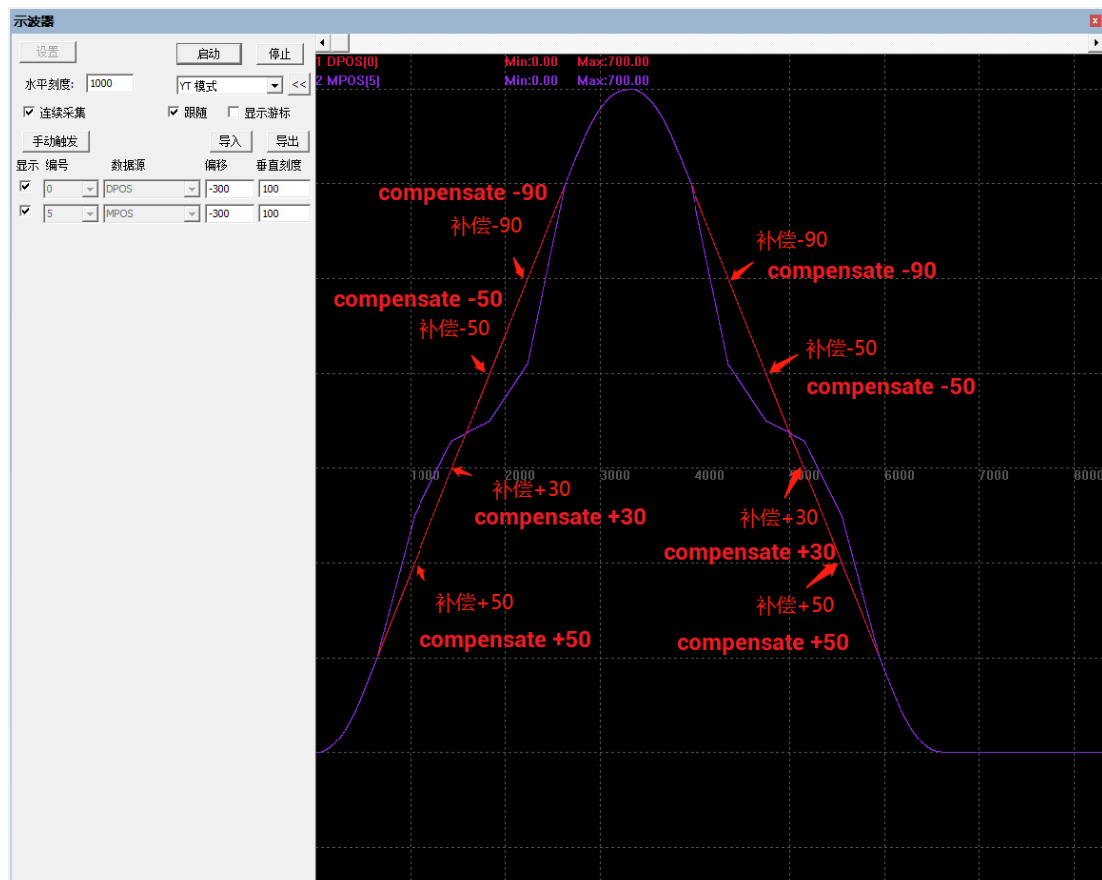
while (1) //wait for axis 0 to complete the motion
{
    Sleep(100);
    ZAux_Direct_GetParam(handle,"IDLE",0,&IDLE);
    if (IDLE<0) break;
}

ret =ZAux_Direct_Pitchset(handle,0,0,100,4,100,0,numlist); //close pitch compensation
commandCheckHandler("ZAux_Direct_Pitchset", ret) ; //if it executes successfully

getchar();
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ; //if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}

```

- Waveform: (in this waveform diagram, in order to better display the compensation effect, the pulse output of axis 0 is connected to the encoder input of axis 5, so that it is convenient to observe the difference between the planned position and the actual position to see the compensation effect)



7.3. Backlash Compensation

7.3.1. Instruction List

Commands	Descriptions
ZAux_Direct_Backlash	Set axis backlash compensation, it is invalid for extended axes

7.3.2. Principle & Description

Because there must be a certain gap between the screw and the nut, so when it is converted into reverse from forward, within a certain angle, although the screw rotates, the nut can only move the work table after the gap is eliminated. This gap is the backlash, but it should be reflected in the rotation angle of the lead screw.

On the equipment, there is a backlash in each link of the transmission chain, such as gear transmission, ball screw nut pair, etc. The backlash is one of the factors that affect the machining accuracy. When the CNC machine tool table changes direction in its direction of motion, the backlash will cause the servo motor to idle without actually moving the table, which is called loss of motion. If the value of the backlash is small, there is no big effect on the machining accuracy, but if the value is large, the stability of the system will be significantly reduced, and the machining accuracy will be obviously reduced, especially for curve processing, which will affect the dimensional tolerance and the consistency of the curve, at this time, **the elimination or compensation of the backlash must be carried out to improve the machining accuracy.**

7.3.3. Routine

Backlash Compensation

Assume that a mechanism has a backlash of 10 pulse amount. The following example demonstrates that the mechanism moves 200 in the forward direction and then moves 100 in the reverse direction. Using backlash compensation, the backlash is automatically compensated.

```
// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "192.168.0.11"; //controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");

    ret = ZAux_Direct_SetAtype(handle, 0, 4); //set axis type as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret); // if it executes successfully
    ret = ZAux_Direct_SetUnits(handle, 0, 1); //set pulse amount as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret); // if it executes successfully

    ret = ZAux_Direct_SetAccel(handle, 0, 500); //set axis acceleration
    commandCheckHandler("ZAux_Direct_SetAccel", ret); // if it executes successfully
    ret = ZAux_Direct_SetDecel(handle, 0, 500); //set axis deceleration
    commandCheckHandler("ZAux_Direct_SetDecel", ret); // if it executes successfully
    ret = ZAux_Direct_SetDpos(handle, 0, 0); //set clear axis DPOS to 0
    commandCheckHandler("ZAux_Direct_SetDpos", ret); // if it executes successfully
    ret = ZAux_Direct_SetMpos(handle, 0, 0); //set clear axis MPOS to 0
    commandCheckHandler("ZAux_Direct_SetMpos", ret); // if it executes successfully

    ret = ZAux_Direct_SetSpeed(handle, 0, 1000); //set axis speed
    commandCheckHandler("ZAux_Direct_SetDecel", ret); // if it executes successfully

    ret = ZAux_Direct_SetMerge(handle, 0, 1); //set axis speed
    commandCheckHandler("ZAux_Direct_SetDecel", ret); // if it executes successfully

    ret = ZAux_Direct_Backlash(handle, 0, 1, 10, 100, 100); //open axis 0 backlash compensation,
    compensate 10
    commandCheckHandler("ZAux_Direct_Backlash", ret); // if it executes successfully

    float IDLE;
    int mode;
    int AxisStatus;
    ZAux_Trigger(handle); //trigger the oscilloscope

    ZAux_Direct_Single_Move(handle, 0, 200); //axis 0 moves 200
    ZAux_Direct_Single_Move(handle, 0, -100); //axis 0 moves -100

    while (1) //wait for axis 0 to complete the motion
    {
```



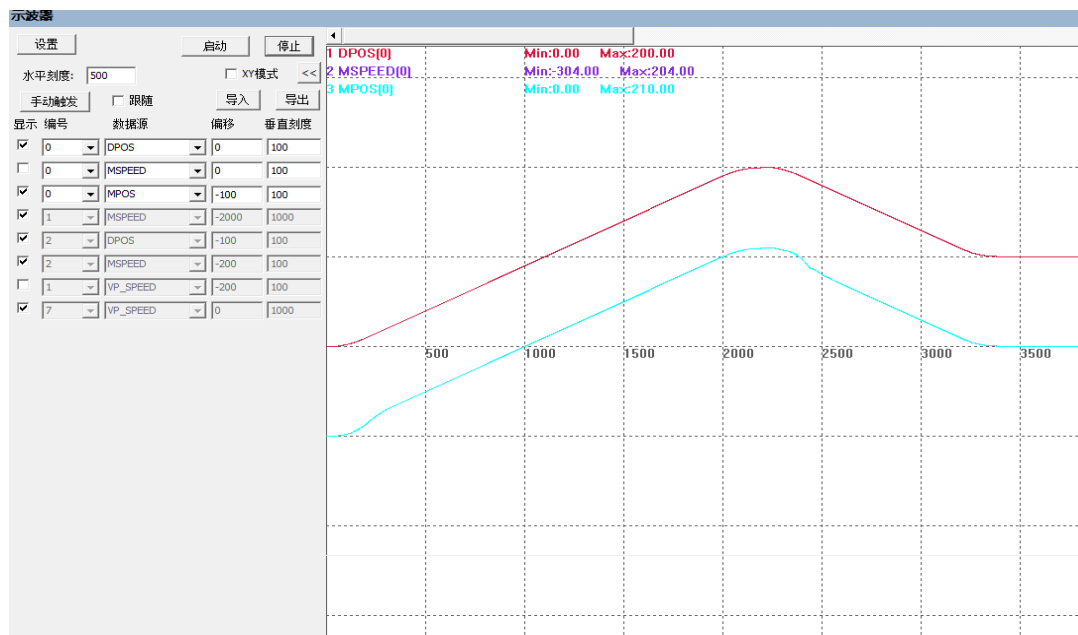
```

        Sleep(100);
        ZAux_Direct_GetParam(handle,"IDLE",0,&IDLE);
        if (IDLE<0)break;
    }
    ret =ZAux_Direct_Backlash(handle,0,0,0,100,100);//close axis 0 backlash compensation
    commandCheckHandler("ZAux_Direct_Backlash", ret) ;// if it executes successfully

    Sleep(2000);
    ret = ZAux_Close(handle); //close the connection
    commandCheckHandler("ZAux_Close", ret) ;// if it executes successfully
    printf("connection closed!\n");
    handle = NULL;
    return 0;
}

```

- Waveform: it can be seen when the direction is switched, the blue one (actual position) moves more than red one (planning position):



Chapter VIII How to Set / Get Hard Port?

8.1. Digital IN & OUT

8.1.1. Instruction List

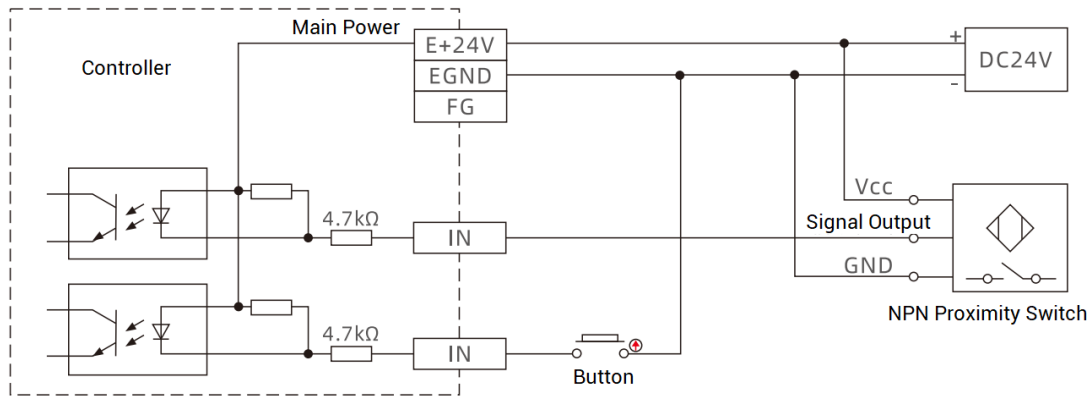
Commands	Descriptions
ZAux_Direct_GetIn	Read single IN signal
ZAux_Direct_SetOp	Set single IN signal
ZAux_Direct_GetOp	Read single OP state
ZAux_GetModbusIn	Quick to read multi-IN (Modbus)
ZAux_GetModbusOut	Quick to read multi-OP state (Modbus)
ZAux_Direct_GetInMulti	Read multiple IN signals
ZAux_Direct_SetOutMulti	Set multiple OPs to output
ZAux_Direct_GetOutMulti	Read multiple outputs states
ZAux_Direct_SetInvertIn	Set "invert" for valid input
ZAux_Direct_GetInvertIn	Read IN invert state

8.1.2. Principle & Description

Take the ZMC406-V2 motion controller as an example, the input and output ports of the controller do not need to be powered separately by an external IO power supply, and the IO points of the expansion module can be expanded up to 512 points.

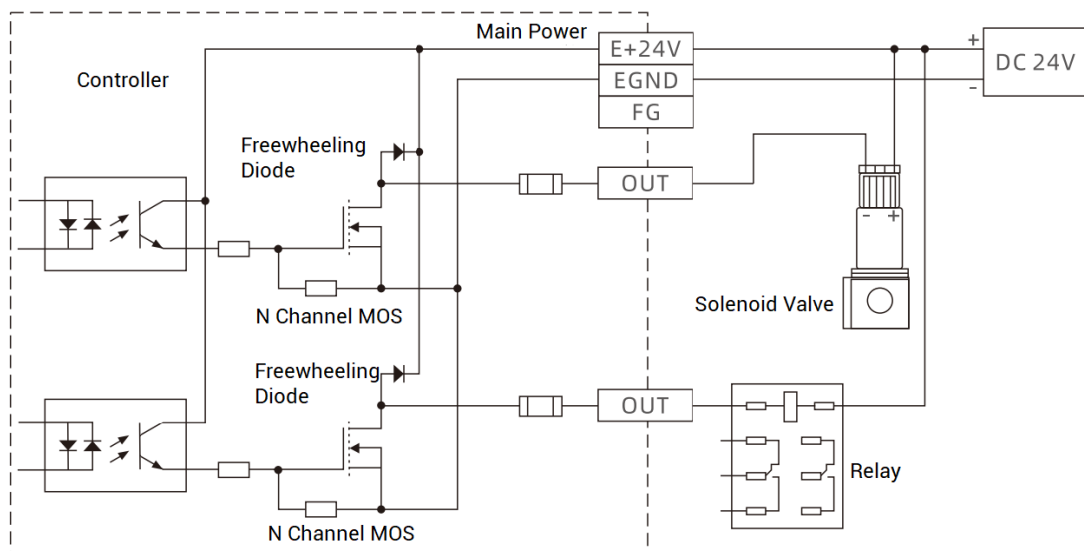
1) Digital IN

The input signal type allowed to be connected to the digital input port IN of the controller is NPN, 24V signal, and the internal circuit diagram is as follows. IN voltage <7V is low level. And there are **high-speed inputs** (allowable IN frequency: 100kHz) and **normal speed inputs** (allowable IN frequency: 5kHz).



2) Digital OUT

The output signal type of the digital output port of the controller is NPN, 0V signal, and the output current is 300mA. The internal circuit diagram is as follows. And there are **high-speed outputs** (allowable OUT frequency: 400kHz) and **normal speed outputs** (allowable OUT frequency: 8kHz).



➤ Please refer to corresponding hardware manuals for detailed parameters.

8.1.3. Routines

IO Getting & Settings

This routine gets input state of simulator IN0, and gets the state of OP0, then resets OP0 according to OP0 input state, at last, reads OP0 state.

**Connect to
Controller**

Read IN state

Read OP state

Set OP state

Read OP state

```
// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"127.0.0.1";    //controller IP address
    ZMC_HANDLE handle = NULL;              //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");

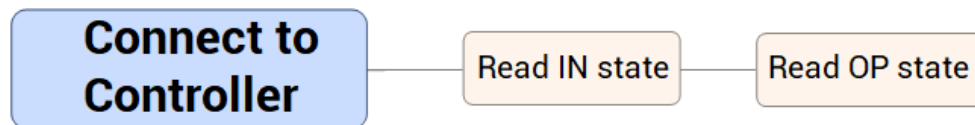
    unsigned int in_value;
    unsigned int out_value;
    ret = ZAux_Direct_GetIn(handle, 0, &in_value); //read IN0 state
    commandCheckHandler("ZAux_Direct_GetIn", ret);
    printf("IN0 status: %d\n", in_value);
    ret = ZAux_Direct_GetOp(handle, 0, &out_value); //read OP0 state
    commandCheckHandler("ZAux_Direct_GetOp", 0);
    printf("OP0 status: %d\n", out_value);

    if(out_value==0)
    {
        ret = ZAux_Direct_SetOp(handle, 0, 1); //set OP0 state
        commandCheckHandler("ZAux_Direct_SetOp", 0);
    }else
    {
        ret = ZAux_Direct_SetOp(handle, 0, 0); // set OP0 state
        commandCheckHandler("ZAux_Direct_SetOp", 0);
    }
    ret = ZAux_Direct_GetOp(handle, 0, &out_value); //read OP0 state
    commandCheckHandler("ZAux_Direct_GetOp", 0);
    printf("OP0 status: %d\n", out_value);

    Sleep(2000);
    ret = ZAux_Close(handle); //close the connection
    commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
    printf("connection closed!\n");
    handle = NULL;
    return 0;
}
```

Multi-IO Getting (GetModbusOut/GetModbusIn)

This routine gets input states of simulator IN0-24, then gets states of OP0-OP7.



```

// test1.cp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zaxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1"; //controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");

    uint8 GetValue[8]; //read set value

    ret = ZAux_GetModbusIn( handle, 0, 24, GetValue);

    commandCheckHandler("ZAux_GetModbusIn", ret);

    for (int i = 0; i <= 24; i++)
    {
        if (i%8==0)
        {
            printf("\n");
        }
        //take 8 bits at a time, each bit corresponds to an IO
        //(GetValue[(int)(i/8)] & (0x1 <<(i%8))): only the corresponding bit is reserved, and the rest
        //are all set to 0
        //(i%8): take out the corresponding bit state
        printf("IN(%d) = %d ", i, (GetValue[(int)(i/8)] & (0x1 <<(i%8))) >> (i%8));
    }
}
  
```

```
    }

    printf("\n*****\n");
    ret = ZAux_GetModbusOut(handle, 0, 7, GetValue);
    commandCheckHandler("ZAux_GetModbusOut", ret);

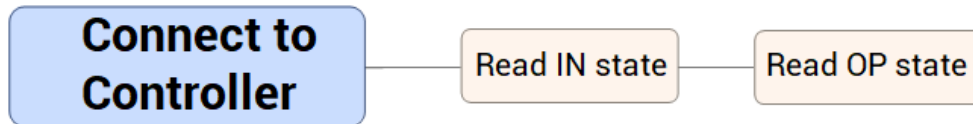
    for (int i = 0; i <= 7; i++)
    {
        printf("OP(%d) = %d ", i, (GetValue[(int)(i/8)] & (0x1 << (i%8))) >> (i%8));
    }

    printf("\n");

    Sleep(20000);
    ret = ZAux_Close(handle); //close the connection
    commandCheckHandler("ZAux_Close", ret); //if it executes successfully
    printf("connection closed!\n");
    handle = NULL;
    return 0;
}
}
```

Multi-IO Getting & Settings

This routine gets input states of simulator IN0-47, then gets states of OP0-OP7.



```
// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1"; //controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
}
```

```
}
printf("Success to connect controller\n");

int32 GetValue;                               //read set value
uint32 uValue;

ret = ZAux_Direct_GetInMulti( handle, 0 , 48, &GetValue);

commandCheckHandler("ZAux_GetModbusIn", ret);

for (int i = 0; i <= 48; i++)
{
    if (i%10==0)
    {
        printf("\n");
    }
    printf("\n*****\n");
};

uValue=1;
ret = ZAux_Direct_SetOutMulti(handle, 0 , 7 , &uValue);
commandCheckHandler("ZAux_GetModbusOut", ret);

ret = ZAux_Direct_GetOutMulti(handle, 0 , 7 , &uValue);
commandCheckHandler("ZAux_GetModbusOut", ret);

for (int i = 0; i <= 7; i++)
{
    printf("OP(%d) = %d ", i, (uValue & (0x1 << i)) >> i);
}
printf("\n");

Sleep(2000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}
```

8.2. Analog IN & OUT

8.2.1. Instruction List

Commands	Descriptions
ZAux_Direct_GetAD	Read single AIN
ZAux_Direct_SetDA	Open single AOUT
ZAux_Direct_GetDA	Read AOUT state

8.2.2. Principle & Description

The controller AD/DA uses an internal power supply. For example, the resolution of ZMC416BE analog is 12 bits, 0-10V voltage type, and the corresponding scale is 0-4095. The controller has 2 channels of AD and 2 channels of DA, which can be expanded by analog expansion module (up to 256 channels of AD and 128 channels of DA). There are 12-bit and 16-bit expansion modules. The controller defaults to 12-bit resolution.

PIN	Name	Description
1	DA0	0-10V analog output 0
2	DA1	0-10V analog output 1
3	AGND	Analog GND
4	AD0	0-10V analog input 0
5	AD1	0-10V analog input 1

8.2.3. Routine

AD/DA Getting & Settings

This routine obtains the input state of the simulator AD0, and obtains the state of DA0, then re-reads the state of DA0 after setting DA0



```
// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"127.0.0.1"; //controller IP address
```



```
ZMC_HANDLE handle = NULL;           //link handle
int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
if (ERR_SUCCESS != ret)
{
    printf("Fail to connect controller!\n");
    handle = NULL;
    Sleep(2000);
    return -1;
}
printf("Success to connect controller!\n");

float AD_value;
float DA_value;
ret = ZAux_Direct_GetAD(handle,0,&AD_value); //read AD0 state
commandCheckHandler("ZAux_Direct_GetAD", ret);
printf("read AD_value: %f\n",AD_value);

ret = ZAux_Direct_GetDA(handle,0,&DA_value); //read DA0 state
commandCheckHandler("ZAux_Direct_GetDA", ret);
printf("read DA_value: %f\n",DA_value);
Sleep(1000);
ret = ZAux_Direct_SetDA(handle,0,3000); //set DA0 state
commandCheckHandler("ZAux_Direct_SetDA", ret);
Sleep(1000);
ret = ZAux_Direct_GetDA(handle,0,&DA_value); // read DA0 state
commandCheckHandler("ZAux_Direct_GetDA", ret);
printf("read DA_value: %f\n",DA_value);

Sleep(2000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}
```

8.3. PWM Control

8.3.1. Instruction List

Commands	Descriptions
ZAux_Direct_SetPwmFreq	Set PWM frequency
ZAux_Direct_GetPwmFreq	Read PWM frequency
ZAux_Direct_SetPwmDuty	Set PWM duty cycle
ZAux_Direct_GetPwmDuty	Read PWM duty cycle

8.3.2. Principle & Description

PWM means "Pulse Width Modulation", which is a very effective technology to

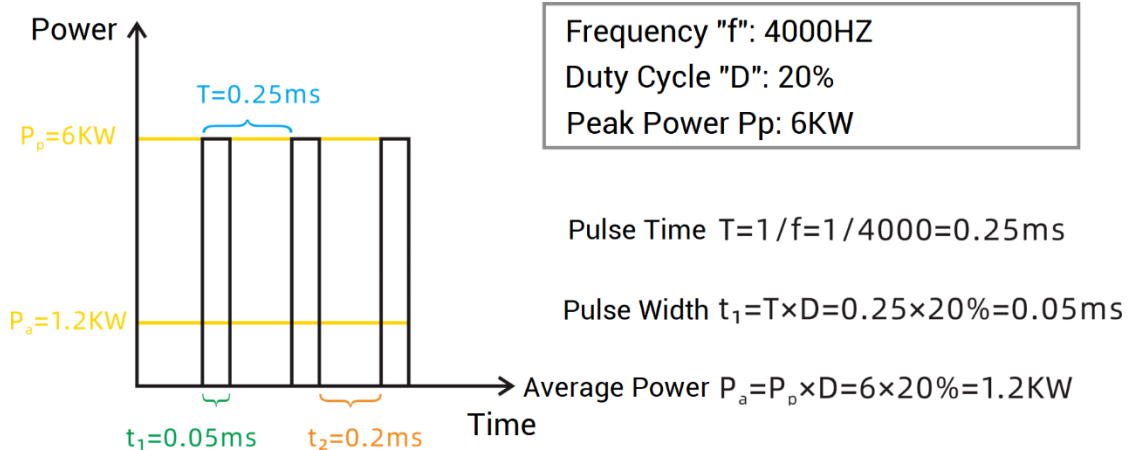
control analog circuits by using the digital output of a microprocessor. Specifically, the pulse width is adjusted by setting the duty cycle.

Note: it is only valid in controllers that support PWM.

When adjusting the process parameters of laser cutting, **the pulse frequency and duty cycle** are often used.

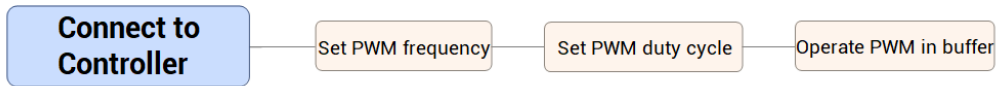
- cycle signal: one cycle is from high level to low level and back to high level
- Pulse frequency: how many cycles of PWM in 1 second.
- Pulse duty cycle: the ratio of the high-level time to the entire cycle time in one period.
- Pulse time: the time used for one cycle.
- Pulse width: the time used for high level in one cycle.
- Laser peak power: the instantaneous energy output within a pulse width.
- Average power: the energy output by the laser in one repetition period.

Next, see an example, how to calculate the various parameters of PWM.



8.3.3. Routine

PWM Usage



```
// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zaxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"192.168.0.11"; //controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");

    float freq_value;
    float duty_value;
    ret = ZAux_Direct_SetPwmFreq(handle, 0, 20); //set the PWM frequency, it must be set
    before setting the duty cycle, here sets it as 20, to use RTSys oscilloscope to capture the waveform
    commandCheckHandler("ZAux_Direct_SetPwmFreq", ret);
    ZAux_Trigger(handle); //trigger the oscilloscope

    ret = ZAux_Direct_SetPwmDuty(handle, 0, 0.2); //set PWM duty cycle, if it is > 0, output PWM
    commandCheckHandler("ZAux_Direct_SetPwmDuty", ret);

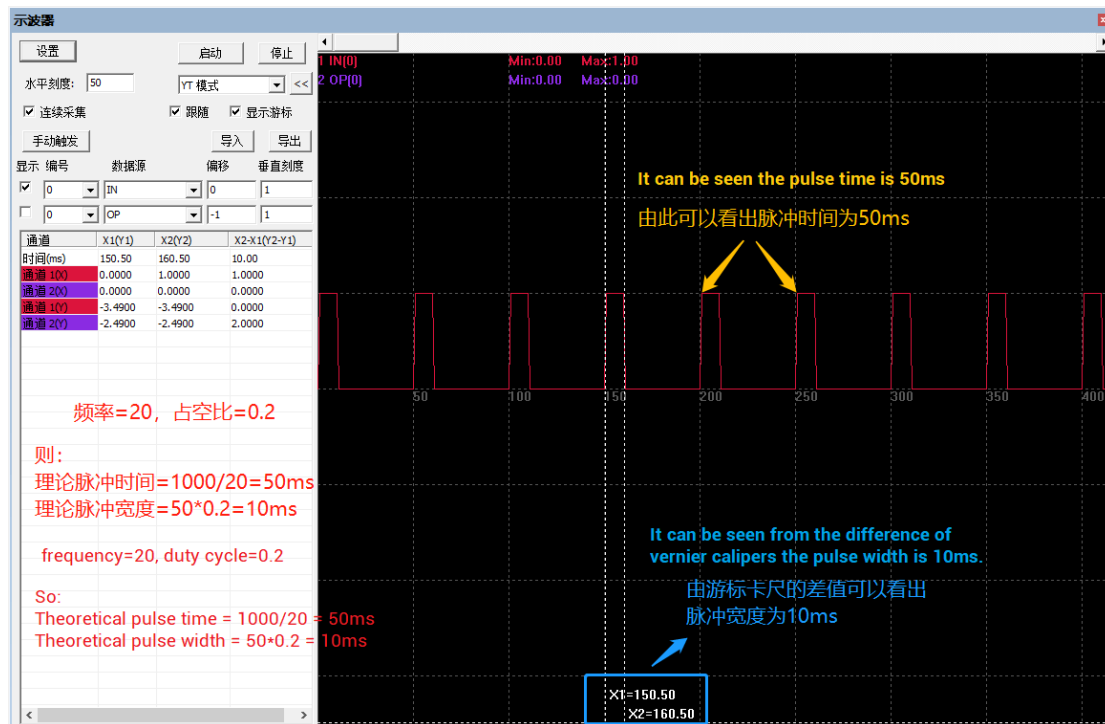
    Sleep(1000);

    ret = ZAux_Direct_SetPwmDuty(handle, 0, 0); //set PWM duty cycle as 0, close PWM output
    commandCheckHandler("ZAux_Direct_SetPwmDuty", ret);

    Sleep(2000);
    ret = ZAux_Close(handle); //close the connection
    commandCheckHandler("ZAux_Close", ret); //if it executes successfully
    printf("connection closed!\n");
    handle = NULL;
    return 0;
}
```

}

➤ Waveform captured by the oscilloscope:



Chapter IX How to Ensure Motion Safe?

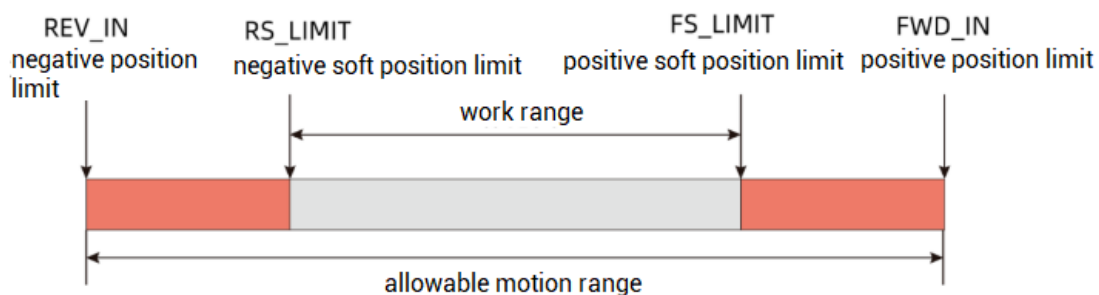
9.1. Axis Soft Position Limit

9.1.1. Instruction List

Commands	Descriptions
ZAux_Direct_SetFsLimit	Set forward soft position limit
ZAux_Direct_GetFsLimit	Read forward soft position limit
ZAux_Direct_SetRsLimit	Set inverse soft position limit
ZAux_Direct_GetRsLimit	Read inverse soft position limit

9.1.2. Principle & Description

Each axis motion range can be limited by installing position limit switch or setting the soft position limit. That is, both are used to set allowable motion and working range for object axis.



Here, soft position limit is talked mainly.

For soft level position limit, you only need to set it through command according to real trajectory, no need external switch. When the axis moved at the position, it will be stopped by fast deceleration (ZAux_Direct_SetFastDec), and corresponding axis status bit will be set as 1. Please note that soft position limits are within the hard limits (like above graphic shown).

In a short, when the workbench meets the position limit / when the DPOS exceeds the position limit, the motion controller will make workbench emergency stop. At this time, please adjust axis position, then restart the axis motion. During the process, the axis will only generate a stop signal when it hits the limit. However, it takes a certain

amount of time to decelerate, and the actual axis position will exceed the limit by a certain distance. Assume that the SPEED (ZAux_Direct_SetSpeed) speed is v_0 when stopping, and the fast deceleration STDEC (ZAux_Direct_SetFastDec) is a , the calculation formula is $vt^2 - v_0^2 = 2as$, then bring in the following data: $0 - 100^2 = 2 * (-1000) * s$, get the overshoot distance $s = 5$. Therefore, overshoot could be reduced through increasing fast deceleration and reducing speed.

The main commands are **ZAux_Direct_SetFsLimit** (positive software position limit), **ZAux_Direct_SetRsLimit** (negative software position limit), **ZAux_Direct_SetRepDist** (coordinate cycle position, refer to [Chapter 7.1](#) for details).

- When soft position limits are larger than coordinate cycle position, the parameters are invalid, position limits are disabled.
- If you need to cancel the position limit, set a bigger value for them, don't recommend to modify coordinate cycle value.
- Forward position limit default value: 200000000.
- Inverse position limit default value: -200000000.

9.1.3. Routine

Soft Positive & Negative Position Limit Settings

```
// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail! return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1"; // controller IP address
    ZMC_HANDLE handle = NULL; // link handle
```

```
int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
if (ERR_SUCCESS != ret)
{
    printf("Fail to connect controller!\n");
    handle = NULL;
    getchar();
    return -1;
}
printf("Success to connect controller!\n");

ret = ZAux_Direct_SetAtype(handle, 0, 1); //set axis type as 1
commandCheckHandler("ZAux_Direct_SetAtype", ret); //if it executes successfully
ret = ZAux_Direct_SetAccel(handle, 0, 500); //set axis acceleration
commandCheckHandler("ZAux_Direct_SetAccel", ret); //if it executes successfully
ret = ZAux_Direct_SetDecel(handle, 0, 500); //set axis deceleration
commandCheckHandler("ZAux_Direct_SetDecel", ret); //if it executes successfully
ret = ZAux_Direct_SetDpos(handle, 0, 0); //set clear axis DPOS to 0
commandCheckHandler("ZAux_Direct_SetDpos", ret); //if it executes successfully

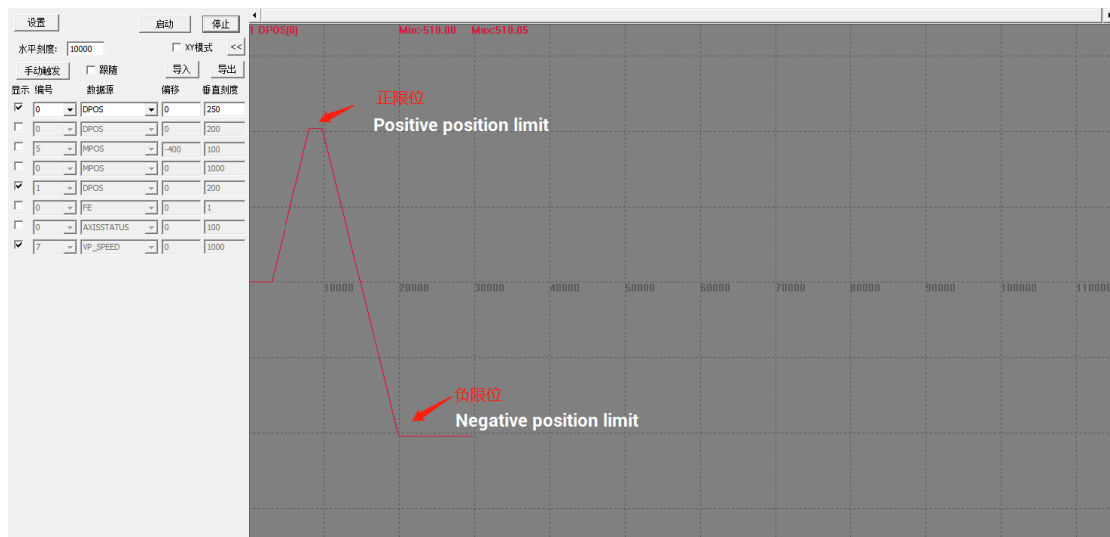
ret = ZAux_Direct_SetFsLimit(handle, 0, 500); //set forward soft position limit as 500
commandCheckHandler("ZAux_Direct_SetDecel", ret); //if it executes successfully
ret = ZAux_Direct_SetRsLimit(handle, 0, -500); //set inverse soft position limit as -500
commandCheckHandler("ZAux_Direct_SetDpos", ret); //if it executes successfully

float IDLE;
int mode;
int AxisStatus;
ZAux_Trigger(handle); //trigger the oscilloscope
while(1)
{
    printf("please enter motion mode, 1-forward, -1-inverse: \n");
    scanf("%d", &mode);

    if (mode != 1 && mode != -1 && mode != 0)
    {
        printf("please input motion mode error, %d\n", mode);
        Sleep(2000);
        return 0;
    }
    if (mode == 0)
    {
        printf("exit\n");
        break;
    }
    ZAux_Direct_Single_Vmove(handle, 0, mode); //axis 0 continuous motion

    while (1) //wait for axis 0 to complete the motion
    {
        Sleep(100);
        ZAux_Direct_GetParam(handle, "IDLE", 0, &IDLE);
        if (IDLE < 0) break;
    }
    ZAux_Direct_GetAxisStatus(handle, 0, &AxisStatus);
    printf("AxisStatus axis state: %s %d : \n", AxisStatus > 0 ? "axis alarm": "axis
normal", AxisStatus);
}
getchar();
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret); //if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}
```

➤ Output Waveform:



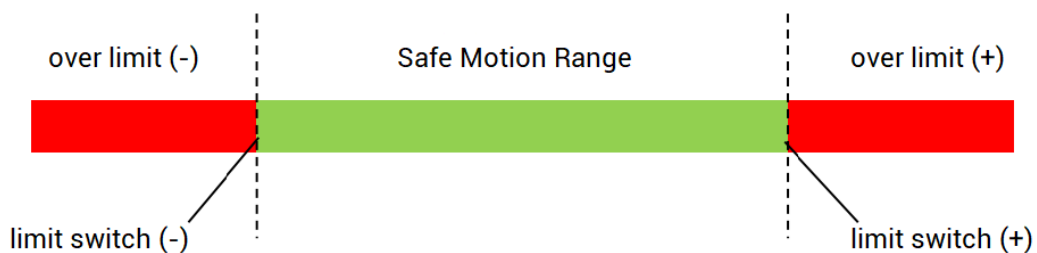
9.2. Axis Hard Position Limit

9.2.1. Instruction List

Commands	Descriptions
ZAux_Direct_SetFwdLn	Set forward hard position limit IN No.
ZAux_Direct_GetFwdLn	Read forward hard position limit IN No.
ZAux_Direct_SetRevLn	Set inverse hard position limit IN No.
ZAux_Direct_GetRevLn	Read inverse hard position limit IN No.

9.2.2. Principle & Description

The controller provides two inputs for each axis as travel control switches, and hardware sensors need to be installed. If the axis touches the limit switch during movement, it will immediately alarm and stop.



The main commands are **ZAux_Direct_SetFwdLn** (positive software position limit), **ZAux_Direct_SetRevLn** (negative software position limit).

Notes:

1) Running after Alarm

- If the limit sends an alarm and reports an error AXISSTATUS: 10h (positive limit alarm), then the axis can only run in the negative direction.
- If the limit sends an alarm and reports an error AXISSTATUS: 20h (negative limit alarm), then the axis can only run in the positive direction.
- If the limit sends an alarm and reports an error AXISSTATUS: 30h (positive limit alarm + negative limit alarm), please check whether the level (valid state) of external equipment is correct.

2) Fast Deceleration Configuration

Remember to set the fast deceleration STDEC (**ZAux_Direct_SetFastDec**)

After the axis is configured as valid positive and negative limits, if the axis exceeds the position, the limit switch will be triggered, then the motion controller will prohibit the movement in the direction of the position limit, and the axis will stop.

However, it takes a certain amount of time to decelerate, and the actual axis position will exceed the limit by a certain distance. Assume that the SPEED (**ZAux_Direct_SetSpeed**) speed is v_0 when stopping, and the fast deceleration STDEC (**ZAux_Direct_SetFastDec**) is a , the calculation formula is $vt^2 - v_0^2 = 2as$, then bring in the following data: $0 - 100^2 = 2 * (-1000) * s$, get the overshoot distance $s = 5$. Therefore, overshoot could be reduced through increasing fast deceleration and reducing speed.

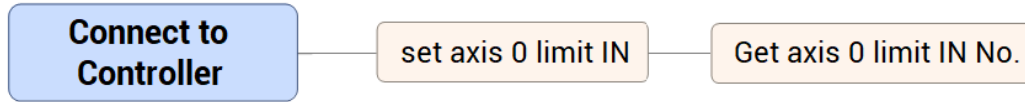
If the deceleration or the fast deceleration is not set, the axis will not stop when it hits the limit.

3) Position Limit Config

The positive and negative limits must be set correctly. That is: when the positive limit is met, the axis can run in inversely normally, when the negative limit is met, the axis can run in forward normally. If they are opposite, the axis cannot move normally when it encounters the positive and negative limits.

9.2.3. Routine

Hard Positive & Negative Position Limit Settings



```
// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"127.0.0.1";           //controller IP address
    ZMC_HANDLE handle = NULL;                     //link handle
    int ret = ZAUX_OpenEth(ip_addr, &handle);     //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");

    int axislist[1] = { 0 };

    /*set the parameter*/
    ret = ZAUX_Direct_SetAtype(handle, axislist[0], 1);           //set axis 0 type
    commandCheckHandler("ZAUX_Direct_SetAtype", ret);
    ZAUX_Direct_SetAtype(handle, axislist[0], 1);               //set axis type of axis 0 as 1
    ZAUX_Direct_SetUnits(handle, axislist[0], 100);             //set pulse amount of axis 0 as 100
    ZAUX_Direct_SetSpeed(handle, axislist[0], 200);             //set axis 0 speed as 200units/s
    ZAUX_Direct_SetAccel(handle, axislist[0], 2000);            //set axis 0 acceleration as 2000units/s/s
    ZAUX_Direct_SetDecel(handle, axislist[0], 2000);            //set axis 0 deceleration as 2000units/s/s
    ZAUX_Direct_SetFastDec(handle, axislist[0], 3000);           //set fast deceleration of axis 0
    float nvalue;
    ZAUX_Direct_GetFastDec(handle, axislist[0], &nvalue);        //obtain axis 0 fast deceleration

    ret = ZAUX_Direct_SetFwdIn(handle, axislist[0], 1);         //map positive limit input to input 1
    commandCheckHandler("ZAUX_Direct_SetFwdIn", ret);

    ZAUX_Direct_SetRevIn(handle, axislist[0], 2);               //map negative limit input to input 2
    commandCheckHandler("ZAUX_Direct_SetRevIn", ret);
}
```

```
ret = ZAux_Direct_SetInvertIn(handle,1, 0); //set origin signal, which means low level is valid
commandCheckHandler("ZAux_Direct_SetInvertIn", ret);

ret = ZAux_Direct_SetInvertIn(handle,2, 0); // set origin signal, which means low level is valid
commandCheckHandler("ZAux_Direct_SetInvertIn", ret);

/*obtain parameters*/
int GetValue;

ret = ZAux_Direct_GetAtype(handle, axislist[0], &GetValue); //set axis type of axis 0
printf("ZAux_Direct_GetAtype = %d \n", GetValue);

ZAux_Direct_GetFwdIn(handle,axislist[0],&GetValue); //get IN No. that sets positive limit input
printf("ZAux_Direct_GetFwdIn = %d \n", GetValue);

ZAux_Direct_GetRevIn(handle,axislist[0],&GetValue); //get IN No. that sets negative limit input
printf("ZAux_Direct_GetRevIn = %d \n", GetValue);

Sleep(2000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}
```

9.3. Axis Alarm

9.3.1. Instruction List

Commands	Descriptions
ZAux_Direct_SetAlmIn	Set axis alarm IN No.
ZAux_Direct_GetAlmIn	Read axis alarm IN No.

9.3.2. Principle & Description

The controller provides a dedicated drive alarm signal input interface. This interface should be connected to the alarm output signal interface of the driver. After detecting the driver alarm signal, the motion controller will turn off the servo enable of the axis.

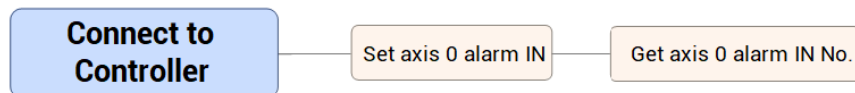
Notes:

- after setting the alarm input port, for ZMC motion controller, it is valid when OFF. Please use **ZAux_Direct_SetInvertIn** to invert the level for normal-ON type.
- for ECI motion controller, it is valid when ON. Please use

ZAux_Direct_SetInvertIn to invert the level for normal-OFF type.

9.3.3. Routine

Alarm IN Settings



```
// test1.cpp: define the entry point of control panel application program

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail! return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1"; // controller IP address
    ZMC_HANDLE handle = NULL; // link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); // connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");

    /* set axis related parameters */
    ZAux_Direct_SetAlmIn(handle, 0, 0); // define axis 0 alarm signal into IN0
    commandCheckHandler("ZAux_Direct_SetAlmIn ", ret);

    ZAux_Direct_SetInvertIn(handle, 0, 0); // set input 0 low level as valid
    commandCheckHandler("ZAux_Direct_SetInvertIn ", ret);

    /* obtain axis related parameters */
    int GetValue;

    ZAux_Direct_GetAlmIn(handle, 0, &GetValue); // get axis 0's alarm signal input port
    printf("ZAux_Direct_GetAlmIn = %d\n", GetValue);

    Sleep(2000);
    ret = ZAux_Close(handle); // close the connection
```

```
commandCheckHandler("ZAux_Close", ret) ;// if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}
```

Chapter X How to Interact Data?

10.1. VR Register

10.1.1. Instruction List

Commands	Descriptions
ZAux_Direct_SetVrf	Set VR register
ZAux_Direct_GetVrf	Read VR register
ZAux_Direct_GetVrInt	Read VR register in integer

10.1.2. Principle & Description

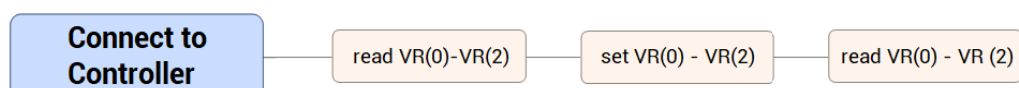
The controller has battery protection registers or ferroelectric memory VR, which can be used to save data when power is lost. The ONPOWER power-down interrupt can be used to record the position at power-down to VR.

Notes:

- VR power-down storage function is only valid in ZMC and XPLC controllers. Although ECI has VR register, which can't achieve real function.
- VR registers are stored ferroelectrically and can be stored for up to 10 years or so. The key parameters of the machine are recommended to be stored in Flash.
- For some ZMC, XPLC 1 / XPLC 2 series products, they save the data by battery, which can be used normally for several years, and they are rechargeable. However, if they are not turned on for a long time, they will be affected.

10.1.3. Routine

VR Register Usage



```
// test1.cpp:
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"127.0.0.1";           //controller IP address
    ZMC_HANDLE handle = NULL;                     //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle);     //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");
    float getVrfData[3] = {0.0};

    ret = ZAux_Direct_GetVrf(handle, 0, 3, getVrfData);//read VR(0)-VR(2)
    commandCheckHandler("ZAux_Direct_GetVrf", ret);

    printf("getVrfData[0] = %f\n", getVrfData[0]);
    printf("getVrfData[1] = %f\n", getVrfData[1]);
    printf("getVrfData[2] = %f\n", getVrfData[2]);

    float setVrfData[3] = {1.1, 2.2, 3.3};
    //write values in array into vr
    ret = ZAux_Direct_SetVrf(handle, 0, 3, setVrfData);//set VR(0)-VR(2)
    commandCheckHandler("ZAux_Direct_SetVrf", ret);

    //get values that was written just now
    ret = ZAux_Direct_GetVrf(handle, 0, 3, getVrfData);//read VR(0)-VR(2)
    commandCheckHandler("ZAux_Direct_GetVrf", ret);

    printf("getVrfData[0] = %f\n", getVrfData[0]);
    printf("getVrfData[1] = %f\n", getVrfData[1]);
    printf("getVrfData[2] = %f\n", getVrfData[2]);

    Sleep(2000);
    ret = ZAux_Close(handle); //close the connection
    commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
    printf("connection closed!\n");
    handle = NULL;
    return 0;
}
```

10.2. Table Register

10.2.1. Instruction List

Commands	Descriptions
ZAux_Direct_SetTable	Set system table register
ZAux_Direct_GetTable	Read system table register
ZAux_Direct_SetTableDouble	Write data into Table
ZAux_Direct_GetTableDouble	Read Table, it can read several at one time

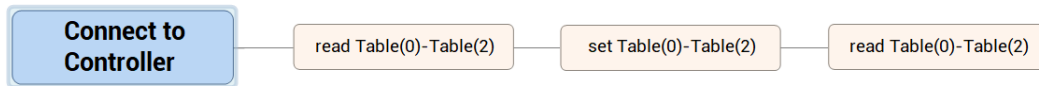
10.2.2. Principle & Description

TABLE is a super large array that comes with the controller. The data type is 32-bit floating point (64-bit floating point for ZMC4XX series and above), but data will not be saved when power off.

- No need to define it when programming, it can be used directly, starting from 0.
- It is global array by default, that is, all program can access TABLE data, like, data recording buffer, cam data table, pitch compensation table, manipulator parameters, etc. are all stored in table.
- For some RTBasic commands, they can directly read TABLE data as parameters, such as cam motion, cam table motion, synchronous motion, forward and reverse kinematic, position comparison output, and so on.
- Parameters captured by the oscilloscope are also stored in TABLE. Therefore, please assign reasonable TABLE areas, do not overlap with the data storage area sampled by the oscilloscope.

10.2.3. Routine

Table Register Usage



```
// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zaxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"127.0.0.1";           //controller IP address
    ZMC_HANDLE handle = NULL;                     //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle);     //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");
    float getTableValue[3] = {0.0};
    //obtain table value
    ret = ZAux_Direct_GetTable(handle, 0, 3, getTableValue);
    commandCheckHandler("ZAux_Direct_GetTable", ret);
    //print the values obtained from table
    printf("getTableValue[0] = %f\n", getTableValue[0]);
    printf("getTableValue[1] = %f\n", getTableValue[1]);
    printf("getTableValue[2] = %f\n", getTableValue[2]);

    float setTableValue[3] = {3.14, 1.44, 1.73};
    //set table
    ret = ZAux_Direct_SetTable(handle, 0, 3, setTableValue);
    commandCheckHandler("ZAux_Direct_SetTable", ret);

    //get table values
    ret = ZAux_Direct_GetTable(handle, 0, 3, getTableValue);
    commandCheckHandler("ZAux_Direct_GetTable", ret);

    //print the values obtained from table
    printf("getTableValue[0] = %f\n", getTableValue[0]);
    printf("getTableValue[1] = %f\n", getTableValue[1]);
    printf("getTableValue[2] = %f\n", getTableValue[2]);
}
```

```
Sleep(2000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}
```

10.3. Modbus Register

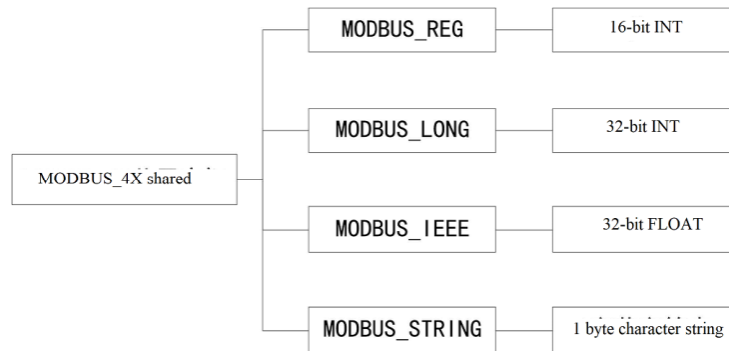
10.3.1. Instruction List

Commands	Descriptions
ZAux_Modbus_Set0x	Set MODBUS bit register
ZAux_Modbus_Get0x	Read MODBUS bit register
ZAux_Modbus_Set4x	Set MODBUS register (REG)
ZAux_Modbus_Get4x	Read MODBUS register (REG)
ZAux_Modbus_Set4x_Long	Set MODBUS4x register (LONG)
ZAux_Modbus_Get4x_Long	Read MODBUS4x register (LONG)
ZAux_Modbus_Set4x_String	Set MODBUS word register (ASCII)
ZAux_Modbus_Get4x_String	Read MODBUS word register (ASCII)
ZAux_Modbus_Set4x_Float	Set MODBUS register (float)
ZAux_Modbus_Get4x_Float	Read MODBUS register (float)

10.3.2. Principle & Description

The MODBUS register uses MODBUS standard communication protocol, and there are 2 types: bit register and word register.

- ✓ Bit Register: MODBUS_BIT, for touch screen, it is MODBUS_0X, Boolean type.
- ✓ Word Register: MODBUS_REG, MODBUS_LONG, MODBUS_IEEE, MODBUS_STRING, for the touch screen, it is MODBUS_4X, the type is as shown below.

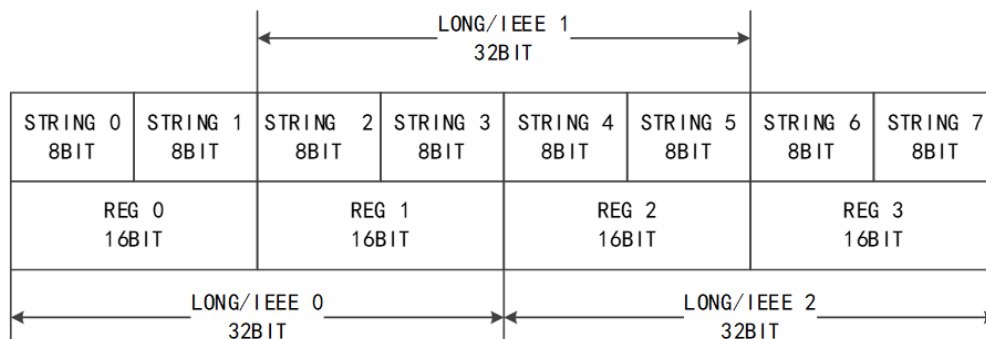


It can be seen the word registers occupy the same storage space, among them, one LONG occupies two REGs, so pay attention to make separate register No.

- $\text{MODBUS\_LONG}(0) = 2 \text{ REG } [\text{MODBUS\_REG}(0) + \text{MODBUS\_REG}(1)]$
- $\text{MODBUS\_LONG}(1) = 2 \text{ REG } [\text{MODBUS\_REG}(1) + \text{MODBUS\_REG}(2)]$
- $\text{MODBUS\_IEEE}(0) = 2 \text{ REG } [\text{MODBUS\_REG}(0) + \text{MODBUS\_REG}(1)]$
- $\text{MODBUS\_IEEE}(1) = 2 \text{ REG } [\text{MODBUS\_REG}(1) + \text{MODBUS\_REG}(2)]$

Therefore, MODBUS_REG, MODBUS_LON, MODBUS_IEEE addresses can't be same.

➤ 4X Space Diagram:



10.3.3. Routine

Modbus Register Usage

```

// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"
  
```

```
void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1";           //controller IP address
    ZMC_HANDLE handle = NULL;                       //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle);       //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");
    uint16 getValue[3];
    ZAux_Modbus_Get4x(handle, 0, 3, getValue);
    commandCheckHandler("ZAux_Modbus_Get4x", ret);
    printf("*****Modbus_4X setting*****\n");
    //print obtained value
    printf("getValue[0] = %d\n", getValue[0]);
    printf("getValue[1] = %d\n", getValue[1]);
    printf("getValue[2] = %d\n", getValue[2]);

    uint16 setValue[3] = {4,5,6};
    //set modbus_4x data, starting from 0, there are 3 values
    //modbus_4x(0) is set to 4
    //modbus_4x(1) is set to 5
    //modbus_4x(2) is set to 6
    ret = ZAux_Modbus_Set4x(handle, 0, 3, setValue);
    commandCheckHandler("ZAux_Modbus_Set4x", ret);

    //obtain modbus_4x's value, here get above set values
    //getValue[0]'s value is the value in modbus_4x(0)
    //getValue[1]'s value is the value in modbus_4x(1)
    //getValue[2]'s value is the value in modbus_4x(2)
    ZAux_Modbus_Get4x(handle, 0, 3, getValue);
    commandCheckHandler("ZAux_Modbus_Get4x", ret);

    //print obtained values
    printf("getValue[0] = %d\n", getValue[0]);
    printf("getValue[1] = %d\n", getValue[1]);
    printf("getValue[2] = %d\n", getValue[2]);
    printf("*****Modbus_String setting *****\n");
    char getStringData[13] = {0};
    //get "100-113" content from the modbus
    ZAux_Modbus_Get4x_String(handle, 100, 13, getStringData);
    commandCheckHandler("ZAux_Modbus_Get4x_String", 0);

    //print obtained character string
    printf("getStringData = %s\n", getStringData);

    char setStringData[13] = "hello world!";
    //write "hello world!" in modbus, the second parameter is the address that starts to write.
    //the character string length is 13, so the third parameter value is 13
    ret = ZAux_Modbus_Set4x_String(handle, 100, 13, setStringData);
    commandCheckHandler("ZAux_Modbus_Set4x_String", ret);
}
```

```
//get "hello world!" from modbus
ZAux_Modbus_Get4x_String(handle,100,13,getStringData);
commandCheckHandler("ZAux_Modbus_Get4x_String", 0);

//print obtained character string
printf("getStringData = %s\n",getStringData);
uint8 GetValue[3];
ret = ZAux_Modbus_Get0x( handle, 0, 24, GetValue);
commandCheckHandler("ZAux_Modbus_Get0x",ret);

printf("*****Modbus_0x setting *****\n");

for (int j=0;j<=2;j++)
{
    printf("%d-%d:",8*j,8*j+7);
    for (int i=0;i<=7;i++)
    {
        putchar((GetValue[j]>>i & 1) ? '1' : '0');
    }
    printf("\n");
}

uint8 SetValue[3] = {1, 2, 3};

ret = ZAux_Modbus_Set0x( handle, 0, 24, SetValue);
commandCheckHandler("ZAux_Modbus_Set0x",ret);

ret = ZAux_Modbus_Get0x( handle, 0, 24, GetValue);
commandCheckHandler("ZAux_Modbus_Get0x",ret);
for (int j=0;j<=2;j++)
{
    printf("%d-%d:",8*j,8*j+7);
    for (int i=0;i<=7;i++)
    {
        putchar((GetValue[j]>>i & 1) ? '1' : '0');
    }
    printf("\n");
}

Sleep(10000);
ret = ZAux_Close(handle); // close the connection
commandCheckHandler("ZAux_Close", ret) ;// if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}
```

10.4. File Reading & Writing (Flash, U Disk, File 3)

10.4.1. Instruction List

Commands	Descriptions
ZAux_WriteUFile	Write file into U disk

ZAux_ReadUFile	Read file from U disk
ZAux_FlashWritef	Write into controller FLASH
ZAux_FlashReadf	Read from controller FLASH
ZAux_DownMem3FileRam	Send packed character string
ZAux_Run3FileRam	Run packed File 3
ZAux_Down3FileRam	Download file 3 to RAM
ZAux_Down3FileRom	Download file 3 to ROM
ZAux_3FileRamGetRemainSpace	Dynamically download file 3, read remain space
ZAux_3FileRamDownBegin	Dynamically download file 3, open downloading
ZAux_3FileRamDownPart	Dynamically download file 3
ZAux_3FileRamDownEnd	Dynamically download file 3, end downloading
ZAux_Check3File	Check whether file 3 exists
ZAux_Delete3File	Delete controller files
ZAux_FindFirst3File	Find controller files, if the filename is empty, which means there is no file
ZAux_FindNext3File	Find controller next files, if the filename is empty, which means there is no file
ZAux_GetCur3File	Obtain current file of controller
ZAux_RemoveAll3Files	Delete controller's all file 3
ZAux_Up3File	Upload file 3
ZAux_Up3FileToMem	Upload file 3

10.4.2. Principle & Description

For some controllers, there are U disk interface and flash.

1) Flash File

Power-down data can be saved into Flash, which can be checked from "?FLASH_SECTES" online command in RTSys output window.

- Flash has a write life limit and cannot be erased and written infinitely.
Therefore, it is recommended to write data that is frequently rewritten into VR.

2) U Disk

U disk can be used to communicate with the controller (U disk file format must be consistent with file reading and writing format / sequence). Reading and writing file types are SD (filenum).BIN or SD (filenum).CSV. different commands are with different file types.

- Usage

It is standard U-disk interface (UDISK), so other standard U disk devices all can be connected. When the controller is powered on and a U disk is inserted, the U disk indicator light will be ON.

PIN	Name	Description
1	V	Internal +5V power
2	D-	Differential data D-
3	D+	Differential data D+
4	GND	Internal power ground

- Notes

Before operating, please use U_STATE command to judge U disk state to ensure U disk state. Recommend to use USB2.0.

Invalid U disk formats: ntfs, exFAT.

Valid U disk formats: fat32, fat16.

- Functions

- upgrade program:** download ZAR encryption program for updating the system program by U disk.
- load Z3P file:** load and execute the Z3P file (file 3) that is saved into U disk.
- interact register data:** U disk data can be interacted with VA register and TABLE register data. And when there are multiple controllers, data in FLASH can be copies each other by U disk.

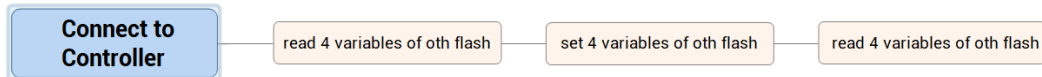
3) File 3 (Z3P File)

If there are too many small segments in multi-segment motion, it can't send all data at one time by Ethernet communication. But if you send the data several times, the motion speed and execution efficiency can't be met. Therefore, "File 3 (third time file)" function is added.

10.4.3. Routines

Flash Writing & Reading

Here shows an example of flash reading and writing, firstly write data into the 0th flash of the controller, and then read the Flash data.



```
// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zAuxDll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"127.0.0.1";           //controller IP address
    ZMC_HANDLE handle = NULL;                     //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle);     //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");
    float readValues[4] = {0};
    uint32 readValuesNum = 0;
    //read flash data from block 0
    ret = ZAux_FlashReadf(handle, 0, 4, readValues, &readValuesNum);
    commandCheckHandler("ZAux_FlashReadf", ret);

    //print obtained data
    printf("readValues[0] = %f\n", readValues[0]);
    printf("readValues[1] = %f\n", readValues[1]);
    printf("readValues[2] = %f\n", readValues[2]);
    printf("readValues[3] = %f\n", readValues[3]);

    float writeValues[4] = { 1.0, 2.0, 3.0, 4.0 };

    //write data into the block 0 of flash
    ret = ZAux_FlashWritef(handle, 0, 4, writeValues);
    commandCheckHandler("ZAux_FlashWritef", ret);
}
```



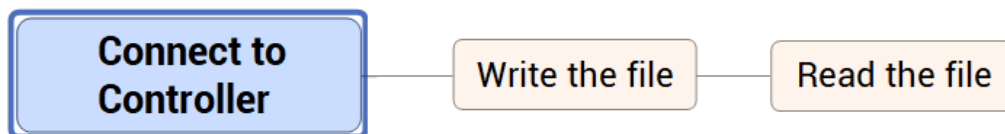
```
//read flash data from block 0
ret = ZAux_FlashReadf(handle, 0,4,readValues,&readValuesNum);
commandCheckHandler("ZAux_FlashReadf", ret);

//print obtained data
printf("readValues[0] = %f\n",readValues[0]);
printf("readValues[1] = %f\n", readValues[1]);
printf("readValues[2] = %f\n", readValues[2]);
printf("readValues[3] = %f\n", readValues[3]);

Sleep(2000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}
```

U Disk File Writing & Reading

Write the data to the file, the format of the file must be consistent with the file format of the U disk, and then read the written data.



```
// test1.cpp: define the entry point of control panel application program //

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"127.0.0.1"; //controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
}
```

```
printf("Success to connect controller!\n");

//save data into test.txt file, it will generate in same program path
float writepvarlist[10] = {1.1,2.2,3.3,4.4,5.5,6.6,7.7,8.8,9.9,10.2};
ret = ZAux_WriteUFile("test123.txt",writepvarlist,10);
commandCheckHandler("ZAux_WriteUFile", ret);

float readpvarlist[10] = {0};
int readnum = 0;
//read data that has been saved into file just now to floating array
ret = ZAux_ReadUFile("test123.txt", readpvarlist, &readnum);
commandCheckHandler("ZAux_ReadUFile", ret);

//print read results:
printf("readpvarlist[0] = %f\n",readpvarlist[0]);
printf("readpvarlist[1] = %f\n", readpvarlist[1]);
printf("readpvarlist[2] = %f\n", readpvarlist[2]);
printf("readpvarlist[3] = %f\n", readpvarlist[3]);
printf("readpvarlist[4] = %f\n", readpvarlist[4]);
printf("readpvarlist[5] = %f\n", readpvarlist[5]);
printf("readpvarlist[6] = %f\n", readpvarlist[6]);
printf("readpvarlist[7] = %f\n", readpvarlist[7]);
printf("readpvarlist[8] = %f\n", readpvarlist[8]);
printf("readpvarlist[9] = %f\n", readpvarlist[9]);

Sleep(2000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}
```

File 3 Usage

Here uses simulator to dynamically load file 3, dynamically appending character string to control axis 0.

```
// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
}
```

```
//step 1: initialize, connect to controller / simulator
char *ip_addr = (char *)"127.0.0.1"; //simulator IP address
ZMC_HANDLE handle = NULL; //connection handle
int ret = ZAux_OpenEth(ip_addr, &handle); //connect to simulator
if (ERR_SUCCESS != ret)
{
    printf("Controller Connect Failed!\n");
    handle = NULL;
    Sleep(2000);
    return -1;
}
printf("Controller Connect Successed!\n");

//Step 2: open file 3, dynamically load file 3
uint32 iRemainSpace;
uint32 nRemain;
ret = ZAux_3FileRamDownBegin(handle, 0, &iRemainSpace, "Zmc.z3p");
commandCheckHandler("ZAux_3FileRamDownBegin", ret);

//Step 3: edit file contents in character string
char cmdbuff[262144];

strcpy(cmdbuff, "SPEED(0)=100\n");
strcat(cmdbuff, "ACCEL(0)=1000\n");
strcat(cmdbuff, "DECEL(0)=1000\n");
strcat(cmdbuff, "DPOS(0)=0\n");
strcat(cmdbuff, "MERGE(0)=1\n");
strcat(cmdbuff, "MOVE(100)axis(0)\n");
strcat(cmdbuff, "MOVE(200)axis(0)\n");
strcat(cmdbuff, "MOVE_RESUME\n"); //it must append for the first ending.

//if the buffer is not enough, please wait enough file 3 buffer.
ret = ZAux_3FileRamGetRemainSpace(handle, 0, &nRemain);
commandCheckHandler("ZAux_3FileRamGetRemainSpace", ret);
if (nRemain <= strlen(cmdbuff))
{
    printf("too large splicing file!\n");
    Sleep(5000);
    return 0;
}

//load file 3 character string
ret = ZAux_3FileRamDownPart(handle, 0, cmdbuff, (uint)strlen(cmdbuff), &nRemain);
commandCheckHandler("ZAux_3FileRamDownPart", ret);

//run file 3
ret = ZAux_Run3FileRam(handle, 0, 0);
commandCheckHandler("ZAux_Run3FileRam", ret);

//step 3.5: dynamically append and load
//if file / trajectory data is not super large, you could run the data in step 1 directly, no need to
dynamically append, skip step 3.5, execute step 4.

strcpy(cmdbuff, "MOVE(1)axis(0)\n");
for (int i=0; i<999; i++)
{
    strcat(cmdbuff, "MOVE(1)axis(0)\n");
}

ret = ZAux_3FileRamGetRemainSpace(handle, 0, &nRemain);
while (nRemain <= strlen(cmdbuff))
{
    ret = ZAux_3FileRamGetRemainSpace(handle, 0, &nRemain);
    commandCheckHandler("ZAux_3FileRamGetRemainSpace", ret);
}
```

```
        ret = ZAux_3FileRamDownPart(handle, 0, cmdbuff, (uint)strlen(cmdbuff), &nRemain);  
//append  
        commandCheckHandler("ZAux_3FileRamDownPart",ret);  
  
//step 4: all files are downloaded  
        ret = ZAux_3FileRamDownEnd(handle, 0);  
  
        Sleep(20000);  
//step 5: end  
        ret = ZAux_Close(handle); //close the connection  
        commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully  
        printf("connection closed!\n");  
        handle = NULL;  
        return 0;
```

Chapter XI Bus Related Information

11.1. Bus Initialization

11.1.1. Instruction List

Commands	Descriptions
ZAux_BusCmd_GetNodeNum	Read node numbers on card slot
ZAux_BusCmd_GetNodeInfo	Read node information
ZAux_BusCmd_GetNodeStatus	Read node bus state
ZAux_BusCmd_DriveClear	Clear bus servo alarm
ZAux_Direct_SetErrormask	Set error mark
ZAux_Direct_GetErrormask	Read error mark
ZAux_Direct_SetAxisEnable	Set axis enable, it is only valid inn bus type controllers
ZAux_Direct_GetAxisEnable	Read axis enable, it is only valid inn bus type controllers
ZAux_Direct_GetBusAxisEnable	Read bus axis enable state

11.1.2. EtherCAT Bus

EtherCAT Protocol

EtherCAT is an industrial Ethernet technology, which is high performance, low cost, flexible topology. It can be used as an ultra-high-speed I/O network at the industrial field level, using standard Ethernet physical layer, transmission media twisted pair or optical fiber (100Base-TX or 100Base-FX).

EtherCAT system consists of master station and slave station.

EtherCAT Communication Specifications:

Item		Specification
Communication protocol		IEC61800-7 CiA 402 Drive Profile
Application Layer	SDO	SDO request, SDO response
	PDO	changeable PDO mapping
	CiA 402	Synchronous Cycle Position Mode (CSP)

		Synchronous Cycle Velocity Mode (CSV) Synchronous Cyclic Torque Mode (CST) HM (homing) TouchProbe (probe)
Physical Layer	Transfer protocol	100BASE-TX(IEEE802.3)
	Max distance	100M
	Interface	RJ45(EtherCat □)

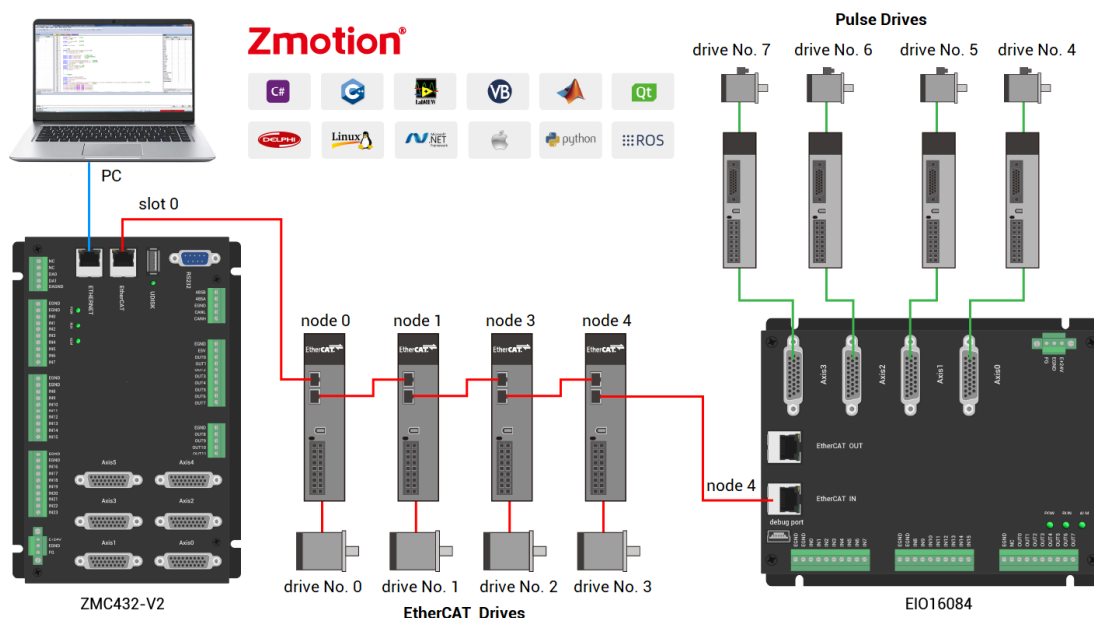
EtherCAT Interface

The EtherCAT interface uses the RJ45 standard network cable interface, same as the EtherNet, which is used to connect the EtherCAT drivers and EtherCAT expansion modules [Not all motion controllers support EtherCAT].

After EtherCAT driver is connected, please perform EtherCAT initialization before using, and use the commands to map drive axis No., enable the driver, etc. For details, refer to the bus initialization reference program.

What's more, EtherCAT axis and pulse axis can be used together.

➤ EtherCAT Wiring Reference:



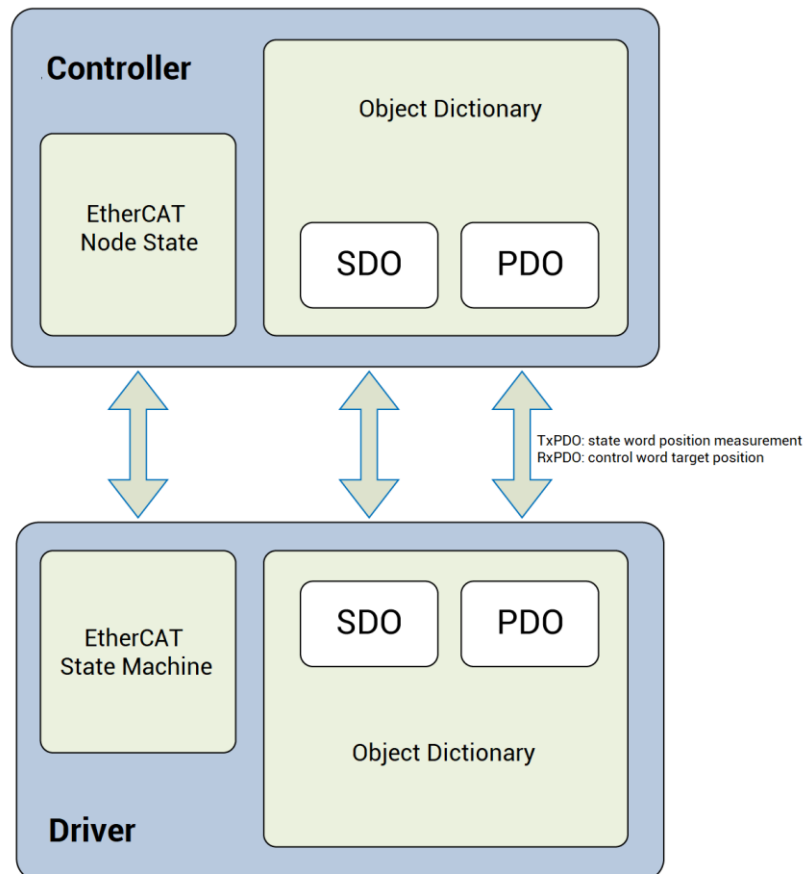
Basic Concepts:

- Slot No.: EtherCAT slot No. on the controller. There is only one EtherCat slot by

default, and the default slot No. is 0.

- **Node:** the devices on the EtherCAT. They are arranged according to the wiring sequence of EtherCAT devices, starting from 0 (to the number of devices -1). For axis No. and IO No., they are configured through IO numbers.
- **Axis No.:** axis No. for the controller, starting from 0 (to the total number of axes connected to -1), and the axis No. can be mapped to any connected drive device through the AXIS_ADDRESS command.
- **EtherCAT Period Time:** 1ms by default. You could modify it as 500us – 4ms for ZMC4XX controllers.

EtherCAT Communication Structure



In above structure, the PDO process data object contains the real-time data during the operation of the servo drive, and is accessed periodically for read and write. For SDO mailbox communication, access and modify some communication parameter objects,

PDO process data objects in the way of non-periodical.

Bus Initialization

(1) The difference between pulse controller and bus controller

Bus Control: send command planning position values through the network cable. The initialization program is needed to enable the motor, then to control the motor.

Pulse Control: directly open the corresponding output port to enable the motor, then send pulses to control the motor.

(2) Bus initialization

The process of bus initialization (this process is the basic file initialization process):

- scan the nodes, scan all nodes on the EtherCAT.
- for EtherCAT nodes, perform PDO preset mapping, such as mapping the target position 607ah to the planned position of the axis, and mapping the feedback position 6064h to the feedback position of the axis, etc.
- start the bus, switch the drive state machine, "initialize → pre-run → safety run → run normally"

(3) Bus Homing

Some EtherCAT drives support the drive itself to return to zero, and provide a variety of homing methods. When the driver returns to zero, the origin and the limit sensor switch need to be directly connected to the corresponding IO port of the driver (when homing by controller mode, it needs to be connected to the IO port of the controller). Then, send homing commands by master station to complete homing.

Bus Mode

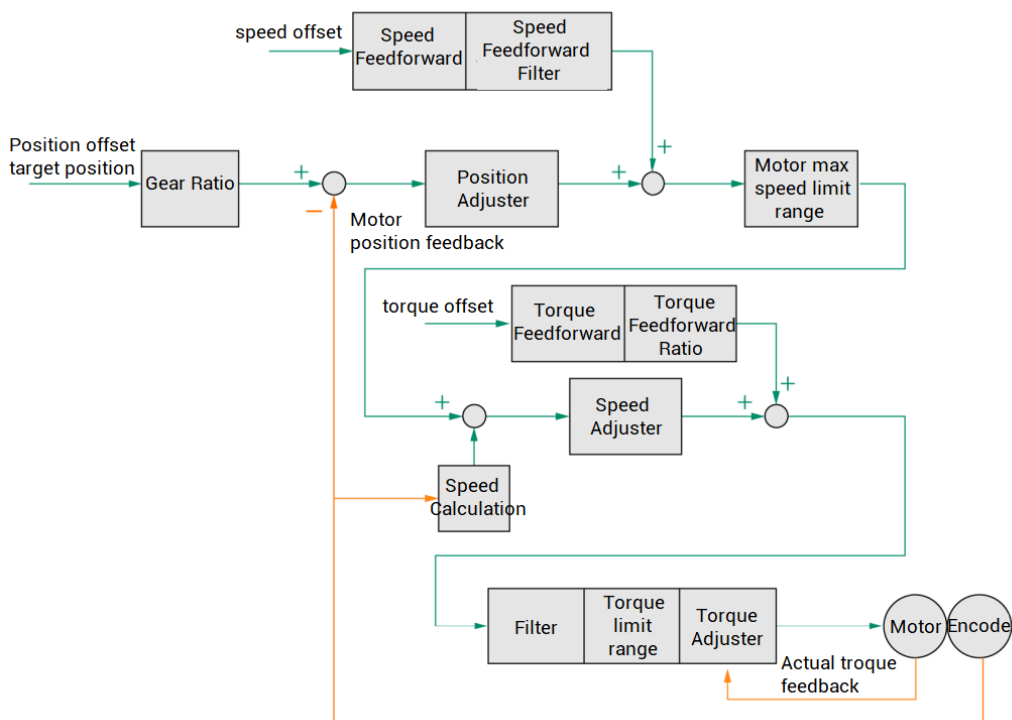
There are 3 commonly used control modes for the EtherCAT, namely **CSP cycle position mode, CSV cycle velocity mode, and CST cycle torque mode.**

Before that, preset drive PDO list, because it can switch modes only when the PDO includes corresponding data dictionary. For PDO list, please refer to drive manual.

- when the driver PDO list contains data dictionary **607Ah**, the axis type can be set to **65, CSP periodical position control mode**, at this time, use the position motion command to control the motor, and set the movement speed.
- when the PDO list of the drive contains the data dictionary **60FFh**, the axis type can be set to **66, CSV periodical velocity control mode**. At this time, use the dedicated speed command to control the motor to run at the set speed, and the position motion command cannot be used.
- when the drive PDO list contains the data dictionary **6071h**, the axis type can be set to **67, CST cycle torque control mode**, at this time, use the dedicated torque command to control the motor to run at the set torque, and the position motion command cannot be used.

1) CSP Mode:

Under periodic synchronous position mode, the upper controller completes the position command planning, and then periodically sends the planned target position to the servo driver, and position, speed, torque control are completed by the servo driver.

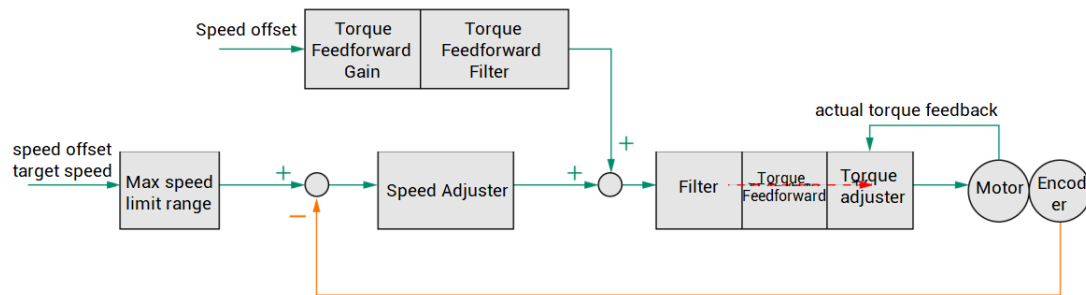


CSP Mode

2) CSV Mode:

Under periodic synchronous speed mode, the upper controller periodically and

synchronously sends the calculated target speed to the servo drive, and the speed and torque adjustments are performed internally by the servo.

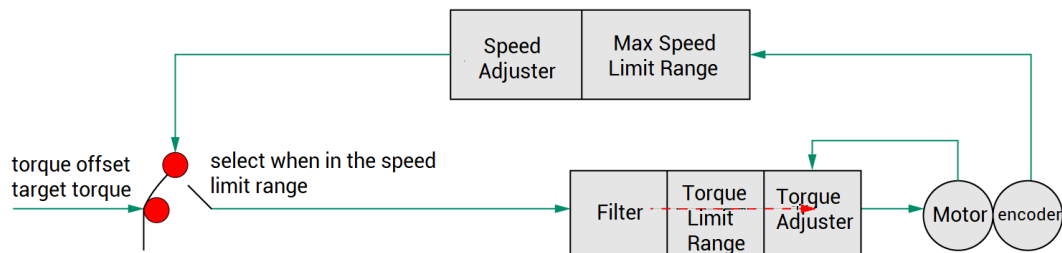


For speed control, please check drive specific unit.

Modify as correct axis type (ATYPE), and modify correct profile (speed control is set to 20), then modify DAC value through online command or in program.

3) CST Mode:

Under periodic synchronous torque mode, the host controller periodically and synchronously sends the calculated target torque to the servo driver, and the torque adjustment is performed internally by the servo.



For EtherCAT torque control, please set the axis type as 67, also modify the profile to be >30. Then modify DAC value (don't make it be different a lot) through online command or in program to control motor.

4) RTEX Torque Mode:

Set correct axis type, and then modify corresponding profile. For details, please refer to RTBasic programming manual.

Send the value of DAC to control the rotation of the motor through the online command, or control the rotation of the motor through the program. The unit of DAC is one thousandth of the torque in torque control, and DAC = 1000 means 100%

Command: dac	Description
Size	With symbol 32bit

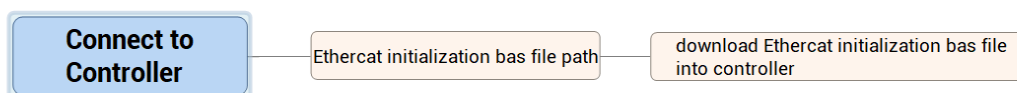
Unit	0.1%
Set range	-motor max torque ~ max motor torque * max torque limit = 100 x Pr9.07 / (Pr9.06 x $\sqrt{2}$)

If the DAC (torque) is too small, the motor cannot overcome the friction and cannot rotate.

11.1.3. Routines

EtherCAT Initialization (with Basic)

If you are not familiar with Zmotion RTBasic programming, you could get Zmotion C++ / C# ECAT initialization routine.



```
// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"192.168.0.11";           //controller IP address
    ZMC_HANDLE handle = NULL;                         //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle);         //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");

    const char * filename= ".\\ECAT init.bas";

    //download the program into ROM, 0-RAM 1-ROM
    ret = ZAux_BasDown(handle,filename,0);//download initialization bas file into controller
```

```

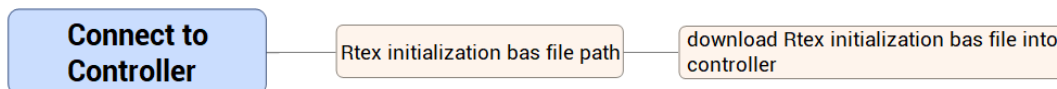
commandCheckHandler("ZAux_BasDown",ret);

Sleep(2000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}

```

Rtex Initialization (with Basic)

If you are not familiar with Zmotion RTBasic programming, you could get Zmotion C++ / C# ECAT initialization routine.



```

// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"192.168.0.11"; //controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");
    //under program path
    const char * filename= ".\\RTEX init.bas";

    //download the program into ROM, 0-RAM 1-ROM
    ret = ZAux_BasDown(handle,filename,0);//download initialization bas file into controller
    commandCheckHandler("ZAux_BasDown",ret);
}

```

```
Sleep(2000);  
ret = ZAux_Close(handle); //close the connection  
commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully  
printf("connection closed!\n");  
handle = NULL;  
return 0;  
}
```

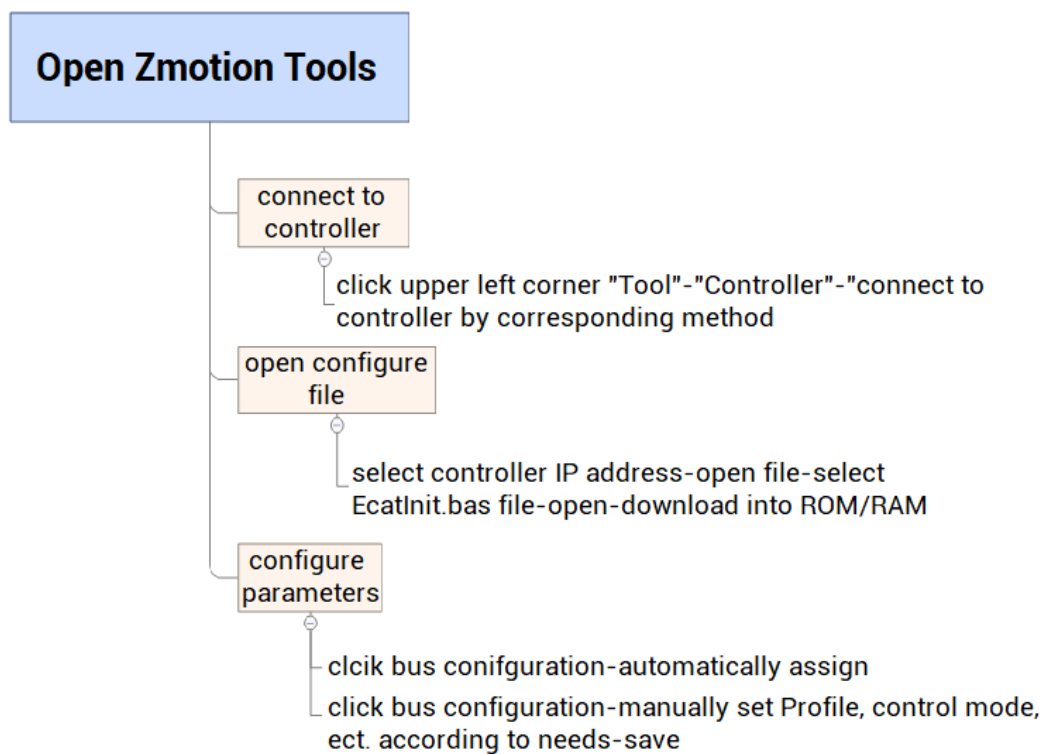
EtherCAT Axis Enable

Before enabling EtherCAT axis, the EtherCAT axis initialization is required. There are 2 methods to achieve that:

- Method 1: use Zmotion RTbasic programming language to write a "ECAT Init" basic file, and download it to the controller through ZAux_BasDown. Or use C++ / C# ECAT init file.
- Method 2: use Zmotion Tools to initialize EtherCAT.

This routine uses method 2 (one EtherCAT motor is connected to the controller, initialize EtherCAT).

Zmotion Tools tool configurations:



```
// test1.cpp: define the entry point of control panel application program  
//
```

```
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"192.168.0.11";    //controller IP address
    ZMC_HANDLE handle = NULL;                //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");

    //ZAux_BusCmd_InitBus(handle);//bus initialization (it is valid in controller that has configured
    bus parameters through "Zmotion tools")

    int GetValue;
    ret = ZAux_BusCmd_GetInitStatus(handle, &GetValue); //obtain bus initialization state, 0
    means failure, 1 means success (it is valid in controller that has configured bus parameters through
    "Zmotion tools")

    commandCheckHandler("ZAux_BusCmd_GetInitStatus", ret);
    printf("Bus initial state= %d \n", GetValue);
    int AxisEnable;
    ret = ZAux_Direct_GetAxisEnable(handle, 0, &AxisEnable);//enable axis, 0 = off, 1 = on
    commandCheckHandler("ZAux_BusCmd_GetInitStatus", ret);
    printf("axis 0 bus init enable state: %d \n", AxisEnable);

    if (AxisEnable<1)
    {
        ret = ZAux_Direct_SetAxisEnable(handle, 0, 1);// enable axis, 0 = off, 1 = on
        commandCheckHandler("ZAux_Direct_SetAxisEnable", ret);
        printf("open enable\n");
    }
    else
    {
        ret = ZAux_Direct_SetAxisEnable(handle, 0, 0);// enable axis, 0 = off, 1 = on
        commandCheckHandler("ZAux_Direct_SetAxisEnable", ret);
        printf("close enable\n");
    }

    Sleep(2000);
    ret = ZAux_Close(handle); //close the connection
    commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
    printf("connection closed!\n");
    handle = NULL;
    return 0;
}
```

11.2. SDO/PDO Writing & Reading

11.2.1. Instruction List

Commands	Descriptions
ZAux_BusCmd_SDOWrite	Write SDO by device No. and slot No.
ZAux_BusCmd_SDORead	Read SDO by device No. and slot No.
ZAux_BusCmd_SDOWrite2	Write node SDO parameters
ZAux_BusCmd_SDORead2	Read node SDO parameters
ZAux_BusCmd_RtexRead	Read RTEX parameters
ZAux_BusCmd_RtexWrite	Set RTEX parameters
ZAux_BusCmd_SDOReadAxis	Read axis SDO parameters
ZAux_BusCmd_SDOWriteAxis	Write axis SDO parameters
ZAux_BusCmd_NodePdoWrite	PDO writing
ZAux_BusCmd_NodePdoRead	PDO reading
ZAux_BusCmd_NodePdoWriteFloat	PDO writing
ZAux_BusCmd_NodePdoReadFloat	PDO reading

11.2.2. Principle & Description

Exchange SDO data through Mailbox communication. So please note that the data refresh time of SDO becomes unstable.

The master side reads and writes data in the records in the object dictionary, and can set objects and monitor various statuses of the slaves.

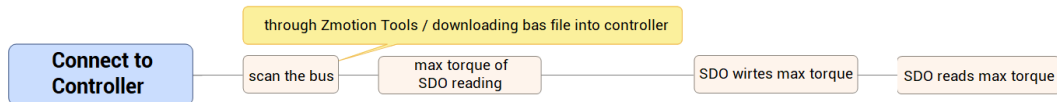
The main commands: **ZAux_BusCmd_SDOWrite** and **ZAux_BusCmd_SDORead**, they only can be used for EtherCAT, writing SDO to the slot No. through the device No.

Note: it takes time to respond SDO writing and reading.

11.2.3. Routines

EtherCAT SDO Writing & Reading

This routine is operated after the bus is initialized and a new project is created.



```
// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail! return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "192.168.0.11"; // controller IP address
    ZMC_HANDLE handle = NULL; // link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); // connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");
    int slot = 0;
    int node = 0;
    int index = 0x6041;
    int subindex = 0;
    int type = 6;
    int value;
    int setValue = 8;

    /* read the state word on slot 0 */
    ret = ZAux_BusCmd_SDORead(handle, slot, node, index, subindex, type, &value);
    commandCheckHandler("ZAux_BusCmd_SDORead", ret);
    printf("Value=%d\n", value);

    /* set the control mode on slot 0 */
    index = 0x6060;
    ret = ZAux_BusCmd_SDOWrite(handle, slot, node, index, subindex, 2, setValue);
    commandCheckHandler("ZAux_BusCmd_SDOWrite", ret);
}
```



```
/*read the control mode on slot 0 */
ret = ZAux_BusCmd_SDORed(handle,slot,node, index, subindex,type,&value);
commandCheckHandler("ZAux_BusCmd_SDORed",ret);
printf("Value=%d\n",value);

Sleep(2000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}
```

RTEX Parameters Related Information

This routine is operated after the bus is initialized and a new project is created.

```
// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"192.168.0.11"; //controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");
    float GetValue;
    int ipara = 7 * 256 + 11;
    /*set the value of Pr7.11*/
    ret = ZAux_BusCmd_RtexWrite(handle,0,ipara,500);

    /*read the value of Pr7.11*/
    ret = ZAux_BusCmd_RtexRead(handle,0, ipara,&GetValue);
    commandCheckHandler("ZAux_BusCmd_RtexRead", ret);
    printf("GetValue=%f\n", GetValue);

    Sleep(2000);
    ret = ZAux_Close(handle); //close the connection
    commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
    printf("connection closed!\n");
}
```

```
    handle = NULL;  
    return 0;  
}
```

11.3. Bus Torque

11.3.1. Instruction List

Commands	Descriptions
ZAux_BusCmd_GetDriveTorque	Read current bus drive's torque, it needs to set corresponding DRIVE_PROFFILE type.
ZAux_BusCmd_SetMaxDriveTorque	Set current bus drive's max torque, it needs to set corresponding DRIVE_PROFFILE type.
ZAux_BusCmd_GetMaxDriveTorque	Read current bus drive's max torque, it needs to set corresponding DRIVE_PROFFILE type.
ZAux_Direct_SetDAC	Set analog output, used under torque, speed mode, for bus drive, it needs to set corresponding DRIVE_PROFFILE type and axis type ATYPE.
ZAux_Direct_GetDAC	Read analog output, used under torque, speed mode, for bus drive, it needs to set corresponding DRIVE_PROFFILE type and axis type ATYPE.

11.3.2. Principle & Description

Motor torque refers to the value of the rotating force. But the torque of the motor is proportional to the strength of the rotating magnetic field and the current in the rotor cage, and is proportional to the square of the power supply voltage, so the torque is determined by the factors of current and voltage.

Motor torque and calculation: $T=9550 * P / n$.

P is the rated (output) power of the motor in kilowatts (KW).

The denominator is the rated speed n and the unit is revolutions per minute (r/min)

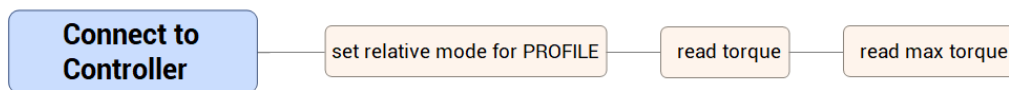
P and n can be found directly from the motor nameplate. Since P and n are the rated values of the motor, T is the rated torque of the motor. Generally speaking, the force that the motor drives the machine to rotate is the torque of the motor. The torque of the

motor = the force of the pulley to drag the belt x the radius of the pulley.

Note: please set correct axis type (ATYPE) and drive PDO, then it can read successfully.

11.3.3. Routine

Get Bus Drive's Current Torque



```
// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail! return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "192.168.0.11"; // controller IP address
    ZMC_HANDLE handle = NULL; // link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); // connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");

    int gValue;
    ret = ZAux_BusCmd_GetDriveTorque(handle, 0, &gValue); // get the current torque, PROFILE needs
    // to be configured corresponding mode.
    commandCheckHandler("ZAux_BusCmd_GetDriveTorque", ret); // if it executes successfully
    printf("axis 0 current torque: %d\n", gValue);

    ret = ZAux_BusCmd_GetMaxDriveTorque(handle, 0, &gValue); // get the current torque, PROFILE
    // needs to be configured corresponding mode.
    commandCheckHandler("ZAux_BusCmd_GetDriveTorque", ret); // if it executes successfully
    printf("axis 0 current torque: %d\n", gValue);
}
```

```
Sleep(2000);  
ret = ZAux_Close(handle); //close the connection  
commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully  
printf("connection closed!\n");  
handle = NULL;  
return 0;  
}
```

Chapter XII Other Commands

12.1. Online Command

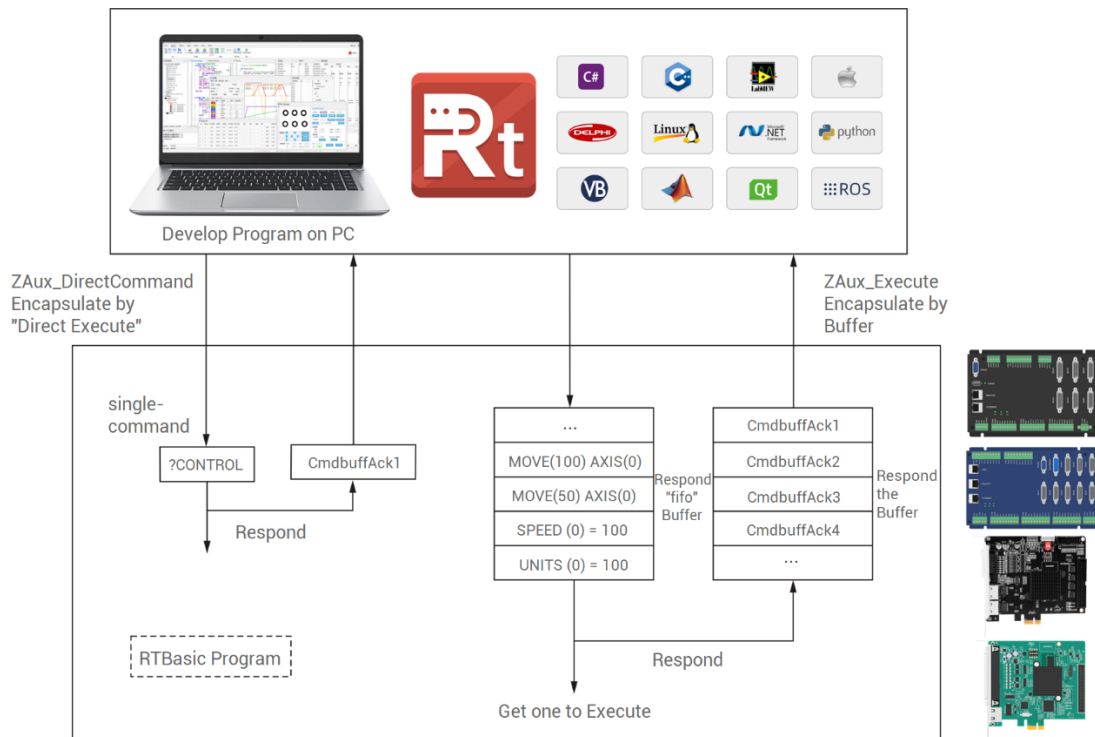
12.1.1. Instruction List

Commands	Descriptions
ZAux_Execute	Send character string (RTBasic) in buffer way
ZAux_DirectCommand	Send character string (RTBasic) in direct way

12.1.2. Principle & Description

ZAux_Execute or **ZAux_DirectCommand** can be used to encapsulate basic commands. If you use unencapsulated commands or want to encapsulate your own functions, you can send them through ZAux_Execute or ZAux_DirectCommand, or refer to existing codes to modify and add corresponding functions.

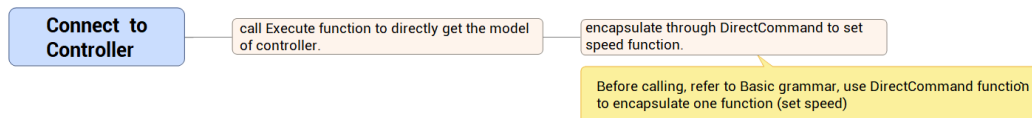
There are two ways to send string commands, buffered and direct.



12.1.3. Routine

“Online Command” Usage

This routine shows how to use the Execute to directly obtain the model of the controller, and how to use the DirectCommand to encapsulate a function that sets speed.



```

// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int Test_Direct_SetSpeed(ZMC_HANDLE handle, int iaxis, float fValue)
{
    char cmdbuff[2048];
    char cmdbuffAck[2048];
    if (iaxis > MAX_AXIS_AUX)
    {
        return ERR_AUX_PARAERR;
    }
    sprintf(cmdbuff, "SPEED(%d)=%f", iaxis, fValue);//generate command's character string

    ZAux_DirectCommand(handle, cmdbuff, cmdbuffAck, 2048);
}

/*parameters:
mode    25, 26, 2D comparison mode
Opnum    corresponding output
Opstate    output state of the comparison point.
maxerr    the pulse deviation of comparison position's each axis left and right, when
entered the deviation range, start to compare.
num    saved comparison points in TABLE
tablepos    TABLE No. of the first comparison point's coordinate

when it is used with hwtimer, hwtimer parameter can be adjusted dynamically.
ModePara1: pulse time
ModePara2: the number of pulses
ModePara3: pulse period
*/
int32 __stdcall ZAux_Direct_HwPswitch2_2D(ZMC_HANDLE handle, int Axisnum, int Mode, int
Opnum, int Opstate, int maxerr, int num, int tablepos, float ModePara1=0, float ModePara2=0, float
ModePara3=0)
  
```

```

{
    if(0 > Axisnum || Axisnum > MAX_AXIS_AUX)
    {
        return ERR_AUX_PARAERR;
    }
    char cmdbuff[2048];
    char cmdbuffAck[2048];
    //generate the command
    switch(Mode)
    {
        case 25:
            sprintf(cmdbuff, "HW_PSWITCH2(%d,%d,%d,%d,%d,%d) AXIS(%d)", Mode, Opnum, Opstate,
maxerr,num,tablepos,Axisnum);
            break;
        case 26:
            sprintf(cmdbuff, "HW_PSWITCH2(%d,%d,%d,%d,%d,%d,%f,%f,%f) AXIS(%d)", Mode, Opnum,
Opstate, maxerr,num,tablepos,ModePara1,ModePara2,ModePara3,Axisnum);
            break;
    }
    //call the command to execute the function

    return ZAux_Execute(handle, cmdbuff, cmdbuffAck,2048);
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"127.0.0.1"; //controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");

    const char * command="?control"; //send the command that obtains controller model
    char psRespons[20]; //the array is used to receive the controller model
    int length=20; //controller model length, which must be larger than received character string
    float GetValue1;
    ret = ZAux_Execute(handle, command, psRespons, length); //obtain controller model
through execute command directly
    commandCheckHandler("ZAux_Execute",ret);

    printf("psRespons = %s\n", psRespons); //print returned character string
    printf("length = %d\n",length);

    ret=ZAux_Direct_GetSpeed(handle,0,&GetValue1);//obtain speed before setting the speed
    commandCheckHandler("ZAux_Direct_GetSpeed",ret);
    printf("the speed value before setting the speed = %f\n",GetValue1);

    ret=Test_Direct_SetSpeed(handle,0,1231);//use own encapsulated speed command to set axis
0 speed
    commandCheckHandler("ZAux_Direct_GetSpeed",ret);

    ret=ZAux_Direct_GetSpeed(handle,0,&GetValue1);//obtain the speed after setting the speed
    commandCheckHandler("ZAux_Direct_GetSpeed",ret);
    printf("speed value after setting the speed = %f\n",GetValue1);
    Sleep(2000);
    ret = ZAux_Close(handle); //close the connection

```

```
commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
printf("connection closed!\n");

handle = NULL;
return 0;
}
```

“Online Command” Usage 2

Please refer to [Chapter 2.2.2-1](#).

“Online Command” Usage 3

Please refer to [Chapter 2.2.2-2](#).

12.2. Controller RTBasic Related

12.2.1. Instruction List

Commands	Descriptions
ZAux_Pause	Pause BASIC program that is running inside the controller
ZAux_Resume	Resume paused BASIC program that is running inside the controller
ZAux_BasDown	Generate single BAS file as ZAR, and download into controller for execution.
ZAux_ZarDown	Download ZAR file into controller
ZAux_ZpjDown	Download ZPJ file into controller
ZAux_Direct_GetVariablef	Read global variables, like, axis parameter, the returned value is floating type
ZAux_Direct_GetVariableInt	Read global variables, like, axis parameter, the returned value is integer type
ZAux_Direct_SetUserArray	Set BASIC customized global array
ZAux_Direct_GetUserArray	Read BASIC customized global array, it can read several at one time
ZAux_Direct_SetUserVar	Set customized global variables

ZAux_Direct_GetUserVar	Read customized global variables
ZAux_SetTimeOut	The command's delay waiting time
ZAux_TransStringtoFloat	Convert character string to float
ZAux_TransStringtoInt	Convert character string to int

12.2.2. What is RTBasic Program

RTBasic program: the program is downloaded into the controller and it runs independent in program. Before control by this program, you need to know about corresponding commands, syntax, and downloading steps. Note: different controllers' series and models are with different spaces. Please check corresponding user manual.

Advantages of Independent Control Program:

- **Independence:** the control program can be executed independently from the PC without occupying PC task resources. The PC only needs simple logic control to complete the entire motion control.
- **Real-Time:** WINDOWS is not a real-time system, the task scheduling of the program is completely controlled by the PC, and data interaction between the program and the controller needs to be carried out continuously, therefore, in some high-speed response control, the program by PC can't respond in time.
- **Confidentiality:** the development can download the non-exportable independent control program in advance or generate a download package to put on the application program, **the file cannot be uploaded**, but can be encrypted as a ZAR file for download. The program can be encrypted, or some control functions can be encrypted to protect your interests.

12.2.3. How to Use RTBasic in PC Side

1) How to Write the Control Program

Please edit it in RTSys "XXX.Bas" file, please check RTBasic Programming Manual.

2) How to Download the Control Program

Use **ZAux_BasDown** command to download the program. (Note: correct xxx.bas file

path must be filled in, and don't make the file path too long. (within 1000 bytes))

12.2.4. Routines

Control Program Downloading & Editing

This routine mainly shows how to use PC library function to download the written xxx.bas program to the controller. Firstly, write the control program, and save the written program to the specified path, then use ZAux_BasDown to download it.

Note: the file path must be represented by "\\", not "/" or "\". (for example: the path of the .bas file is "F:\111\Basic1.bas", in C language, the "\" in the path should be replaced by "\\ ", namely: "F:\\111\\Basic1 .bas").

The routine:

- Hardware: ZMC432-V2 controller.
- Program: ECAT Init.bas, one EtherCAT initialization program
- Purpose: use "ECAT Init.bas" file to do initialization for ZMC432-V2 EtherCAT axis.
- Functions: download "ECAT Init.bas" into ZMC432-V2 controller, and configure 3 variables (PUL_AxisStart – local pulse axis starting axis No., PUL_AxisNum – local pulse axes numbers, Bus_AxisStart – bus axis starting axis No.).

```
*****ECAT Bus Initialization
global CONST BUS_TYPE = 0      'bus type. (can be used to distinguish the current bus type for host computer)
global CONST MAX_AXISNUM = 16  'max axes
global CONST Bus_Slot = 0      'slot No.: 0 (for single bus controller, default is 0)
global CONST PUL_AxisStart = 0 'local pulse-axis starting axis No.
global CONST PUL_AxisNum = 0   'the number of local pulse axes
global CONST Bus_AxisStart = 0 'bus axis starting axis No.
global CONST Bus_NodeNum = 0   'configure the number of nodes to see whether it is consistent with real detected number of slave station

global Bus_InitStatus          'bus initialization completion state
Bus_InitStatus = -1
global Bus_TotalAxisnum        'check how many axes in total scanned
```



```
// test1.cpp: define the entry point of control panel application program
//
```

```
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"
```

```

void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"192.168.0.11";    //controller IP address
    ZMC_HANDLE handle = NULL;                //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");

    const char * filename= ".\\ECAT Init.bas";

    //download into ROM, 0-RAM, 1-ROM
    ret = ZAux_BasDown(handle,filename,1);//download the ECAT init bas file to the controller for
    initialization. After the program is downloaded, after a 7S delay (waiting for the driver to power on
    the reserved time) to initialize the Ethercat bus
    commandCheckHandler("ZAux_BasDown",ret);
    Sleep(500);//delay 500ms, wait for the program is downloaded to controller
    //PUL_AxisStart, PUL_AxisNum, Bus_AxisStart are 3 global variables defined by ECAT init.bas.

    float GetUserVar;
    ret = ZAux_Direct_GetUserVar(handle,"PUL_AxisStart",&GetUserVar);//obtain controller local
    pulse axis starting axis No. before modifying
    commandCheckHandler("ZAux_BasDown",ret);
    printf("before set PUL_AxisStart=%f\n",GetUserVar);
    ret = ZAux_Direct_GetUserVar(handle,"PUL_AxisNum",&GetUserVar);//obtain controller local
    pulse total axes before modifying
    commandCheckHandler("ZAux_BasDown",ret);
    printf("before set PUL_AxisNum=%f\n",GetUserVar);
    ret = ZAux_Direct_GetUserVar(handle,"Bus_AxisStart",&GetUserVar);//obtain controller bus
    axis starting axis No. before modifying
    commandCheckHandler("ZAux_BasDown",ret);
    printf("before set Bus_AxisStart=%f\n",GetUserVar);

    //Because this routine uses ZMC432 this time, the controller body has 6 local pulse axes, so
    set the bus axis number to 0,
    //modify some for variables:
    //controller local pulse axis starting axis No.: PUL_AxisStart is modified as 0, 0 is by default
    //controller local pulse total axes.: PUL_AxisNum is modified as 6, 0 is by default
    //controller bus axis starting axis No.: Bus_AxisSt is modified as 6, 0 is by default
    ret = ZAux_Direct_SetUserVar(handle,"PUL_AxisStart",0);//set controller local pulse axis
    starting axis No.
    commandCheckHandler("ZAux_BasDown",ret);
    ret = ZAux_Direct_SetUserVar(handle,"PUL_AxisNum",6);//set controller local pulse total axes
    commandCheckHandler("ZAux_BasDown",ret);
    ret = ZAux_Direct_SetUserVar(handle,"Bus_AxisStart",6);//set controller bus axis starting axis
    No.
    commandCheckHandler("ZAux_BasDown",ret);

    ret = ZAux_Direct_GetUserVar(handle,"PUL_AxisStart",&GetUserVar);//obtain controller local
    pulse axis starting axis No. after modifying
    commandCheckHandler("ZAux_BasDown",ret);

```

```
printf("after set PUL_AxisStart=%f\n",GetUserVar);
ret = ZAux_Direct_GetUserVar(handle,"PUL_AxisNum",&GetUserVar);//obtain controller local
pulse total axes after modifying
commandCheckHandler("ZAux_BasDown",ret);
printf("after set PUL_AxisNum=%f\n",GetUserVar);
ret = ZAux_Direct_GetUserVar(handle,"Bus_AxisStart",&GetUserVar);//obtain controller bus
axis starting axis No. after modifying
commandCheckHandler("ZAux_BasDown",ret);
printf("after set Bus_AxisStart=%f\n",GetUserVar);

Sleep(2000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) //if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}
```

12.3. Controller Auto-Reporting

12.3.1. Instruction List

Commands	Descriptions
PZMCAutoUpCallBack	The callback function format of auto-report
ZAux_SetAutoUpCallBack	Controller auto-report
ZAux_CycleUpEnable	Enable period report
ZAux_CycleUpDisable	Cancel "period report enable"
ZAux_CycleUpGetRecvTimes	The number of packets received is periodically reported; if it exceeds, it is used for debugging.
ZAux_CycleUpForceOnce	Force a report once, in case the idle state after the motion command is inaccurate.
ZAux_CycleUpReadBuff	Read content from period report (floating)
ZAux_CycleUpReadBuffInt	Read content from period report (integer)

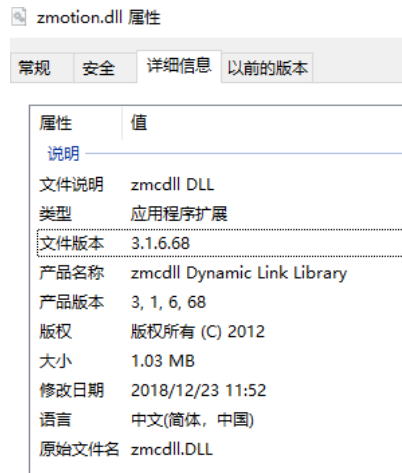
12.3.2. Principle & Description

✓ **Actively Report:**

- 1) the controller actively reports.
- 2) actively report message: controller sends a message to the PC by

SEND_AUTOUP.

- 3) valid in DLL after 3.1.6.68, as shown in the below.



- 4) if you want to know which PORT support "active report", you could send online command "?*IFAUTOUP_PORT" in RTSys output window. -1 = automatically select PORT to actively report message.

✓ **Periodically Report:**

- 1) Both zmotion.dll and controller firmware should support "periodically report".
- 2) The time for PC active polling can be reduced by setting the active periodic reporting for frequently read parameters in advance.

12.3.3. Routine

Report Actively

 BASIC Program:

```
SEND_AUTOUP(-1,2022,"Hello PC")
```

Run the program on the PC at first (it will automatically open the port to be reported), and then execute SEND_AUTOUP(-1,2022,"Hello PC") by RTSys, it will report once every time execution.



```

// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"
//this command is to check the returned value, if the returned value is not 0, print the returned value
to screen.
void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
    }
}
//define the call-back function of reporting periodically
void AutoUpCallBack(ZMC_HANDLE handle, int32 itypecode, int32 idatalength, uint8 *pdata)
{
    printf("upload data code: %d, upload data length: %d, upload content: %s\n",
itypecode,idatalength,(char*)pdata);
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"127.0.0.1"; //controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");
    ret = ZAux_SetAutoUpCallBack( handle, AutoUpCallBack);
    commandCheckHandler("ZAux_SetAutoUpCallBack", ret);

    printf("input 1 character + \"Enter\" button to end program!\n");
    char a[128];
    scanf("%s",&a);

    ret = ZAux_Close(handle); //close the connection
    commandCheckHandler("ZAux_Close", ret) ;// if it executes successfully
    printf("connection closed!\n");
    handle = NULL;
    return 0;
}
  
```

➤ PC Side:

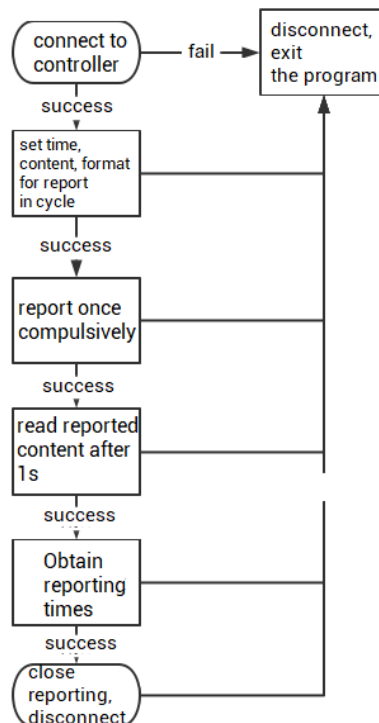
```
控制器连接成功!  
输入1个字符+回车结束程序!  
上传数据码为: 2022、上传数据长度为: 8、上报内容为: Hello PC  
上传数据码为: 2022、上传数据长度为: 8、上报内容为: Hello PC  
上传数据码为: 2022、上传数据长度为: 8、上报内容为: Hello PC
```

➤ Controller side

```
>>SEND_AUTOUP(-1,2022,"Hello PC")  
>>SEND_AUTOUP(-1,2022,"Hello PC")  
>>SEND_AUTOUP(-1,2022,"Hello PC")
```

Report in Cycle

Periodic reporting: the time for PC active polling can be reduced by setting the active periodic reporting for frequently read parameters in advance.



```
// test1.cpp: define the entry point of control panel application program  
//  
#include "stdafx.h"  
#include <windows.h>  
#include "zmotion.h"  
#include "zauxdll2.h"
```

```
#include <time.h>
//this command is to check the returned value, if the returned value is not 0, print the returned value
to screen.
void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
        Sleep(2000);
        exit(0);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"127.0.0.1";           //controller IP address
    ZMC_HANDLE handle = NULL;                     //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle);     //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to connect controller!\n");

    ret = ZAux_CycleUpEnable(handle, 0, 700, "idle, mpos(0,3), dpos(1)");//report motion state of
axis 0, DPOS of axis 1 and MPOS of axis 0-2 in cycle
    commandCheckHandler("ZAux_CycleUpEnable", ret) ;//if it executes successfully

    ret = ZAux_CycleUpForceOnce(handle, 0);//report once compulsively, 0 is the channel No.
    commandCheckHandler("ZAux_CycleUpForceOnce", ret) ;//if it executes successfully

    double dtemp;//read periodical reported content in floating type
    int32 itemp,addrnum;//read periodical reported content in integer type
    addnum=0;

    Sleep(1000);
    ret = ZAux_CycleUpReadBuff(handle, 0, "mpos", 0, &dtemp);//read MPOS (0) in floating type
    commandCheckHandler("ZAux_CycleUpReadBuff", ret) ;//if it executes successfully

    ret = ZAux_CycleUpReadBuffInt(handle, 0, "dpos", 1, &itemp);//read DPOS (1) in integer
    commandCheckHandler("ZAux_CycleUpReadBuff", ret) ;//if it executes successfully
    printf("itemp=%d,mpos=%f\n",itemp,dtemp);

    addnum = ZAux_CycleUpGetRecvTimes(handle, 0);//obtain report times
    printf("report times=%d\n",addnum);

    Sleep(2000);

    ret = ZAux_CycleUpDisable(handle, 0);//close periodical reporting
    commandCheckHandler("ZAux_CycleUpEnable", ret) ;//if it executes successfully

    ret = ZAux_Close(handle); //close the connection
    commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
    printf("connection closed!\n");
    handle = NULL;
    return 0;
}
```


12.4. Debug Related

12.4.1. Instruction List

Commands	Descriptions
ZAux_SetTraceFile	Set "command tracking"
ZAux_Trigger	Trigger oscilloscope
ZAux_Scope	Use SCOPE to capture data
ZAux_GetScopeLength	Get the SCOPE sampled length

12.4.2. Debug & Output

The **ZAux_SetTraceFile** should be used together with the **ZAux_Execute**. After using the debug information, the information sent by the ZAux_Execute command and the returned content will be saved by default, and they are saved into \ZmotionLog\ under the current program path in the form of a log.

Print & Output Modes:

0: OFF, not output information

1: output error command

2: only the motion and parameter setting commands are output

3: output all commands

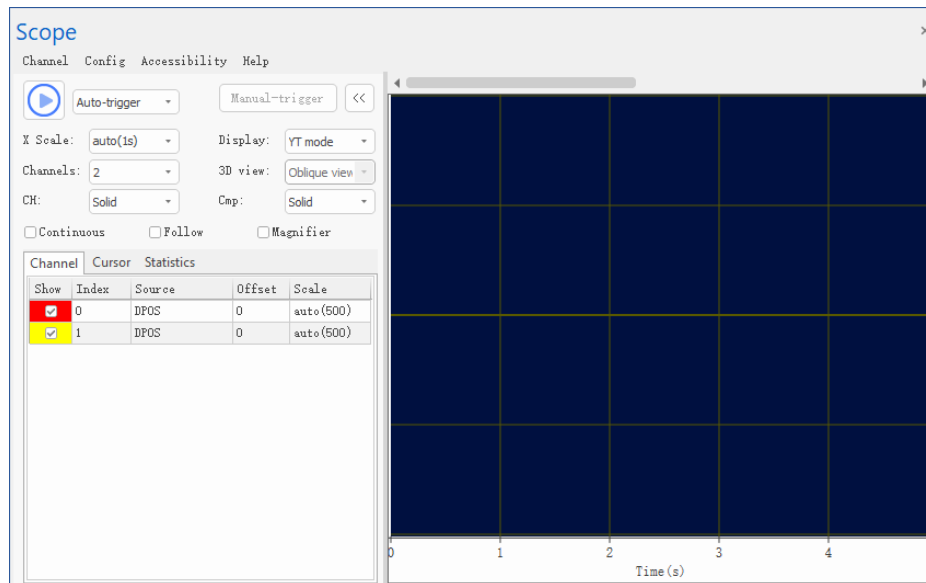
Note: this command is only used during debugging. When not in use, remember to turn off the print output mode, otherwise it will affect the communication efficiency.

12.4.3. Oscilloscope

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.

--How to Operate--

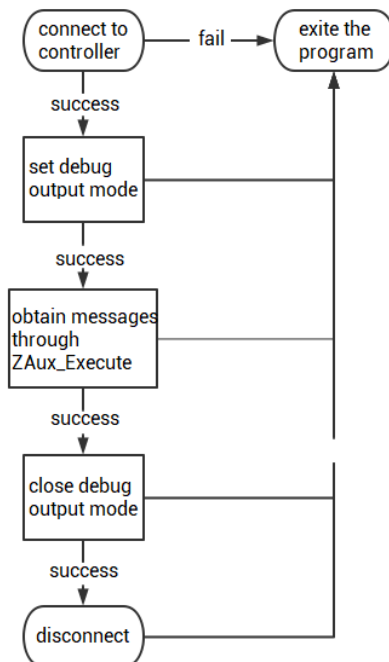
Please check [“RTSys User Manual – 5.1”](#).



For PC side, you also could edit ***“ZAux_Trigger”*** command in your program to trigger oscilloscope to capture data.

12.4.4. Routines

Debug & Print Messages



```

// test1.cpp: define the entry point of control panel application program

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"
#include <time.h>
//this command checks the returned value, if it is not 0, print the returned value to screen.
void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
        Sleep(2000);
        exit(0);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    //char *ip_addr = (char *)"192.168.1.11"; //controller IP address
    char *ip_addr = (char *)"127.0.0.1"; //simulator IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to connect controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
}
  
```

```
printf("Success to connect controller!\n");

//0 close 1-only error commands are output 2-only motion and setting commands are output
3 all commands are output
int bifTofile = 3; //set debug output mode as 3

ret = ZAux_SetTraceFile(bifTofile, "reserved"); //set print output mode
commandCheckHandler("ZAux_SetTraceFile", ret);

const char * command = "?control"; //send the command that gets controller model

const char * command1 = "?*max"; //send the command that sends controller messages

char psRespons[1024]; //the array is used to receive controller model
int length = 1024; //controller model length, which must be > received character string length

ret = ZAux_Execute(handle, command, psRespons, length); //obtain controller model
commandCheckHandler("ZAux_Execute", ret);

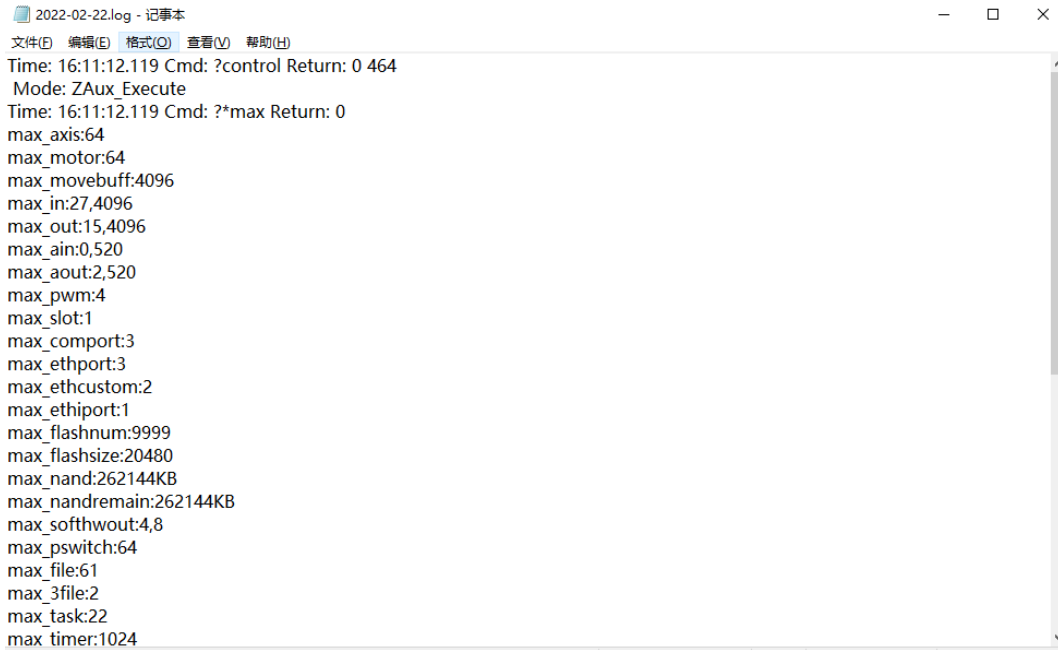
ret = ZAux_Execute(handle, command1, psRespons, length); //obtain controller model
commandCheckHandler("ZAux_Execute", ret);

Sleep(2000);
ret = ZAux_SetTraceFile(0, "reserved"); //close print output mode, after debugged, print output
mode must be closed
commandCheckHandler("ZAux_Execute", ret);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret); // if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}
```

➤ Output Result:

上位机手册整理 > 例程测试 > 第12章 > 12.4.3-1 调试打印信息 > Debug > ZmotionLog

名称	修改日期	类型	大小
 2022-02-21.log	2022/2/21 20:05	文本文档	2 KB
 2022-02-22.log	2022/2/22 16:11	文本文档	1 KB



```
2022-02-22.log - 记事本
文件(F)  编辑(E)  格式(O)  查看(V)  帮助(H)
Time: 16:11:12.119 Cmd: ?control Return: 0 464
Mode: ZAux_Execute
Time: 16:11:12.119 Cmd: ?*max Return: 0
max_axis:64
max_motor:64
max_movebuff:4096
max_in:27,4096
max_out:15,4096
max_ain:0,520
max_aout:2,520
max_pwm:4
max_slot:1
max_comport:3
max_ethport:3
max_ethcustom:2
max_ethipport:1
max_flashnum:9999
max_flashsize:20480
max_nand:262144KB
max_nandremain:262144KB
max_softwout:4,8
max_pswitch:64
max_file:61
max_3file:2
max_task:22
max_timer:1024
```

Oscilloscope Usage

Make axis 0 and axis 1 move from (0,0) to (100,100), then use oscilloscope to capture the motion trajectory and speed curve.

➤ PC Program:

```
// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"
#include <time.h>
//this command is checks returned value, if it is not 0, print the returned value to screen.
void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
        Sleep(2000);
        exit(0);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    //char *ip_addr = (char *)"192.168.1.11"; //controller IP address
    char *ip_addr = (char *)"127.0.0.1"; //simulator IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {

```

```

    printf("Fail to connect controller!\n");
    handle = NULL;
    Sleep(2000);
    return -1;
}
printf("Success to connect controller!\n");

ret = ZAux_Direct_SetAtype(handle,0,1);//set axis type of axis 0 as 1
commandCheckHandler("ZAux_Direct_SetAtype", ret) ;// if it executes successfully

ret = ZAux_Direct_SetAtype(handle,1,1);//set axis type of axis 1 as 1
commandCheckHandler("ZAux_Direct_SetAtype", ret) ;// if it executes successfully
commandCheckHandler("ZAux_Execute", ret);

ret = ZAux_Direct_SetDpos(handle, 0, 0);    //set dpos of axis 0
commandCheckHandler("ZAux_Direct_SetDpos", ret);
ret = ZAux_Direct_SetDpos(handle, 1, 0);    //set dpos of axis 1
commandCheckHandler("ZAux_Direct_SetDpos", ret);

int axislist[2] = { 0,1 };                //motion axes list, axis 0 is the main axis
float poslist[2] = { 100,100};            //motion axes list, axis 0-100, axis1-100
ZAux_Trigger( handle);//trigger the oscilloscope

ret = ZAux_Direct_SetSpeed(handle, axislist[0], 100); //set interpolation speed as 100, set it
on the main axis
commandCheckHandler("ZAux_Direct_SetSpeed", ret);

ret = ZAux_Direct_SetAccel(handle, axislist[0], 500); //set interpolation acceleration as 500,
set it on the main axis
commandCheckHandler("ZAux_Direct_SetAccel", ret);

ret = ZAux_Direct_SetDecel(handle, axislist[0], 500); //set interpolation deceleration as 100,
set it on the main axis
commandCheckHandler("ZAux_Direct_SetDecel", ret);

ret = ZAux_Direct_MoveAbs(handle, 2, axislist, poslist); //call the motion, axis 0 and axis 0
move to corresponding 100, 100 absolute position
commandCheckHandler("ZAux_Direct_MoveAbs", ret);

int state[2];

do
{
    ZAux_Direct_GetIfIdle(handle, axislist[0], state);
    ZAux_Direct_GetIfIdle(handle, axislist[1], state + 1);

    if (!(state[0] || state[1]))
    {
        Sleep(100);
    }
    else
    {
        break;
    }
} while (true);//wait for the motion to complete

Sleep(2000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;// if it executes successfully
printf("connection closed!\n");
handle = NULL;
return 0;
}

```

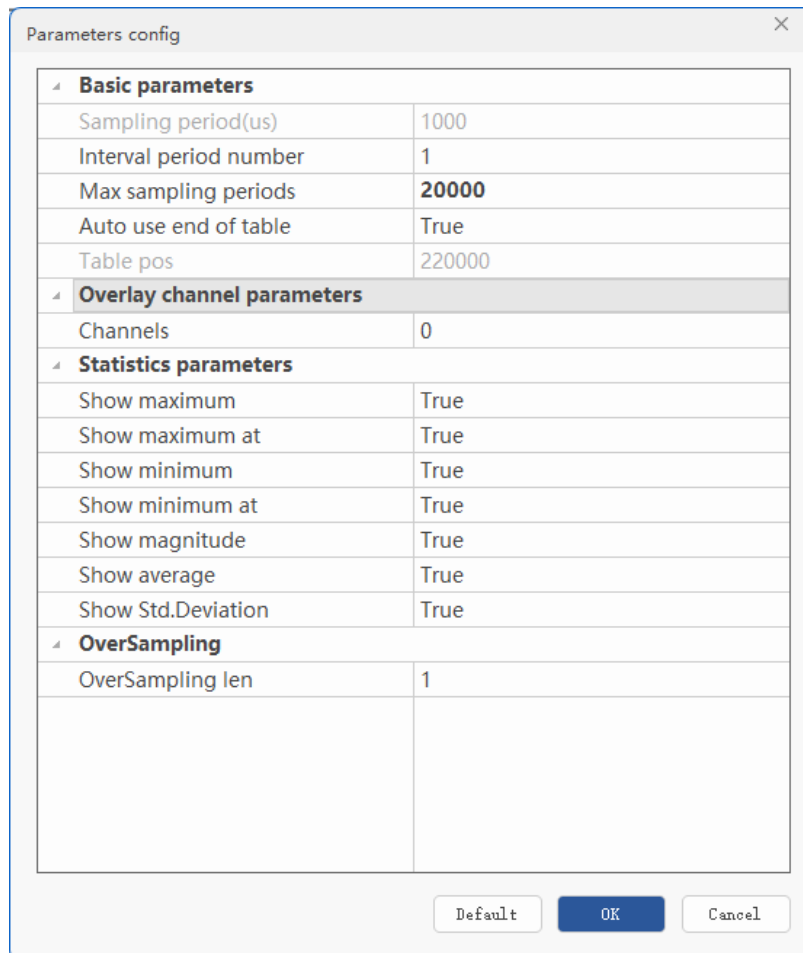
➤ In RTSys "SCOPE" Window:

Step 1: open RTSys software.

Step 2: connect to your controller.

Step 3: open "Tool -- SCOPE".

Step 4: configure the scope parameters, for this routine, parameters are set as below:



The image shows a 'Parameters config' dialog box with a close button (X) in the top right corner. The dialog contains several sections of parameters:

Basic parameters	
Sampling period(us)	1000
Interval period number	1
Max sampling periods	20000
Auto use end of table	True
Table pos	220000

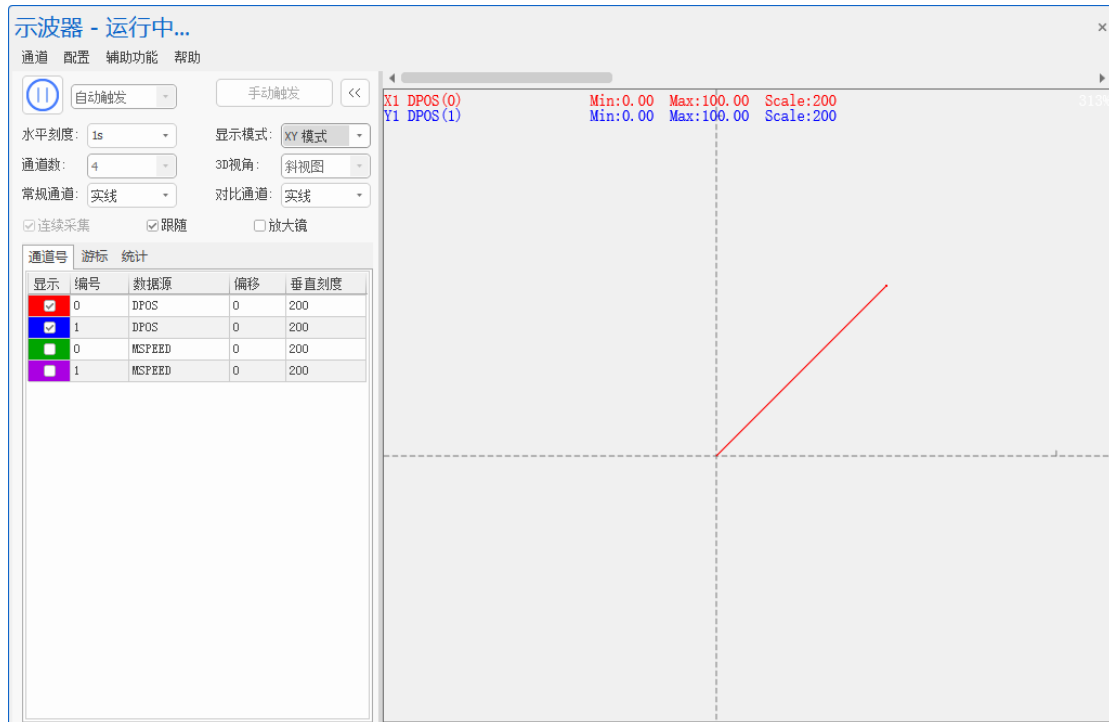
Overlay channel parameters	
Channels	0

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

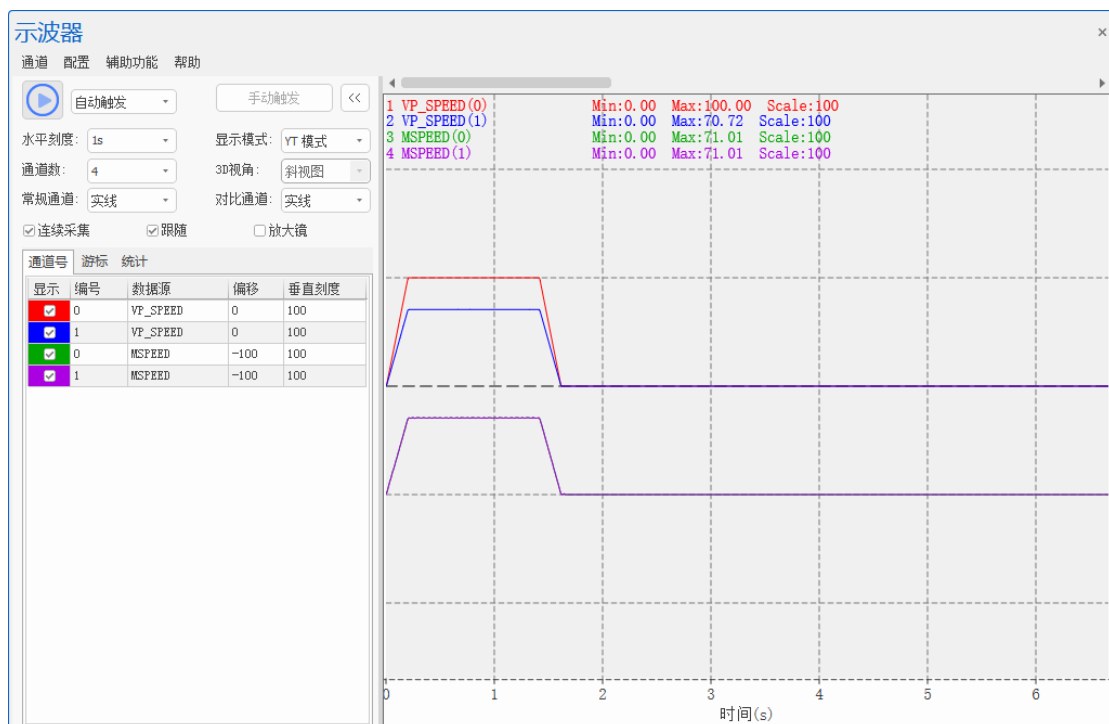
OverSampling	
OverSampling len	1

At the bottom of the dialog, there are three buttons: 'Default', 'OK', and 'Cancel'.

Step 5: capture trajectory data (set No., data source, offset, XY scales, select **XY** mode, select auto-trigger, then, click "start" button to run PC program).



Step 6: capture speed data (set No., data source, offset, XY scales, select **XT** mode, select auto-trigger, then, click "start" button to run PC program).



Data Sampling

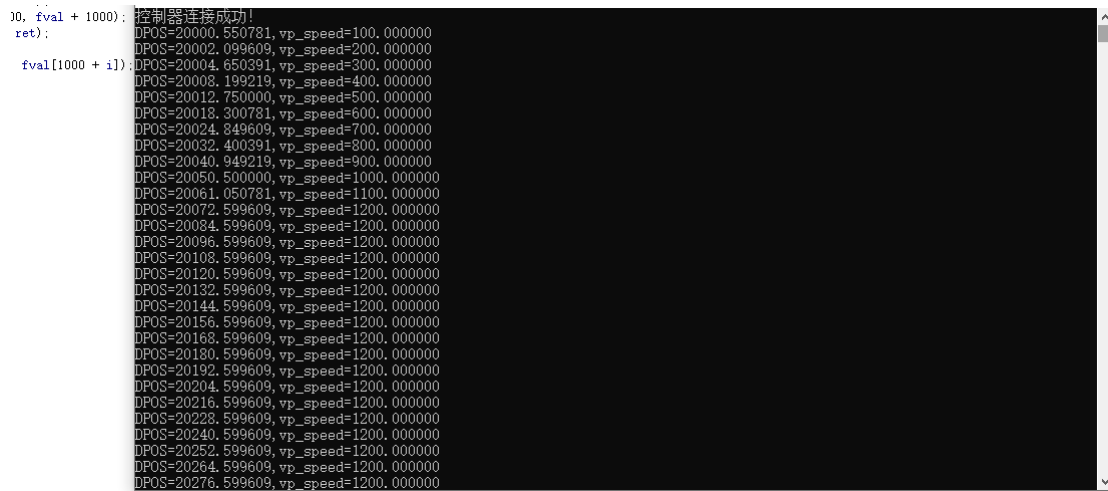
Make axis 0 moves 10000 units, and sample once each 10ms, it captures axis 0 DPOS (demand position) and VP_SPEED (command speed), 1000 times in total.

```
// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zaxdll2.h"
#include <time.h>
// the global variable, which saves the controller connection handle
ZMC_HANDLE handle = NULL;
// check the returned value, if it is not 0, it will print out error info and exit program.
// command: command name
// ret: returned code
void commandCheckHandler(const char *command, int ret) {
    if (ret) {
        printf("%s fail!return code is %d\n", command, ret);
        Sleep(5000);
        exit(0);
    }
}

int _tmain(int argc, _TCHAR* argv[]) {
    // controller IP address, set correct IP
    char *ip_addr = (char *)"127.0.0.1";
    // connect to controller
    int ret = ZAux_OpenEth(ip_addr, &handle);
    if (ERR_SUCCESS != ret) {
        printf("Controller Connect Failed!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Controller Connect Succeeded!\n");
    // configure scope sampling parameters, capture axis 0 DPOS and VP_SPEED.
    ret = ZAux_Scope(handle, 1, 10, 1000, 3000, "DPOS(0),VP_SPEED(0)");
    commandCheckHandler("ZAux_Scope", ret);
    // trigger scope to capture data
    char ackbuff[2048];
    ret = ZAux_Execute(handle, "TRIGGER", ackbuff, 2048);
    commandCheckHandler("ZAux_Execute", ret);
    // make axis 0 do single-axis motion
    ret = ZAux_Direct_Single_Move(handle, 0, 10000);
    commandCheckHandler("ZAux_Direct_Single_Move", ret);
    // wait until sampling completed
    int len = 0;
    float fval[2000] = {};
    Sleep(10000);
    // get sampling data length
    ret = ZAux_GetScopeLength(handle, &len);
    commandCheckHandler("ZAux_GetScopeLength", ret);
    // read and print out sampling data
    if (len > 0) {
        ret = ZAux_Direct_GetTable(handle, 1000, 1000, fval);
        commandCheckHandler("ZAux_Direct_GetTable", ret);
        ret = ZAux_Direct_GetTable(handle, 2000, 1000, fval + 1000);
        commandCheckHandler("ZAux_Direct_GetTable", ret);
        for (int i = 0; i < len; i++) {
            printf("DPOS=%f,vp_speed=%f\n", fval[i], fval[1000 + i]);
        }
    }
}
```

```
    }  
}  
// wait again  
Sleep(10000);  
// close connection  
ret = ZAux_Close(handle);  
commandCheckHandler("ZAux_Close", ret);  
printf("connection closed!\n");  
handle = NULL;  
return 0;  
}
```

➤ Output Result:



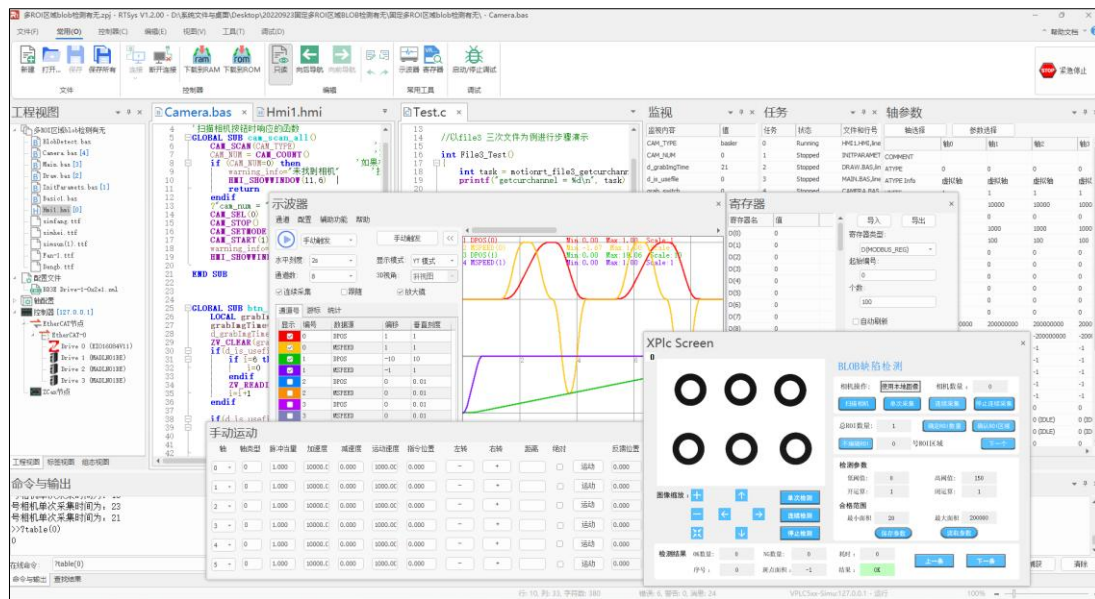
```
0), fval + 1000); 控制器连接成功!  
ret);  
fval[1000 + i]);  
DPOS=20000.550781, vp_speed=100.000000  
DPOS=20002.099609, vp_speed=200.000000  
DPOS=20004.650391, vp_speed=300.000000  
DPOS=20008.199219, vp_speed=400.000000  
DPOS=20012.750000, vp_speed=500.000000  
DPOS=20018.300781, vp_speed=600.000000  
DPOS=20024.849609, vp_speed=700.000000  
DPOS=20032.400391, vp_speed=800.000000  
DPOS=20040.949219, vp_speed=900.000000  
DPOS=20050.500000, vp_speed=1000.000000  
DPOS=20061.050781, vp_speed=1100.000000  
DPOS=20072.599609, vp_speed=1200.000000  
DPOS=20084.599609, vp_speed=1200.000000  
DPOS=20096.599609, vp_speed=1200.000000  
DPOS=20108.599609, vp_speed=1200.000000  
DPOS=20120.599609, vp_speed=1200.000000  
DPOS=20132.599609, vp_speed=1200.000000  
DPOS=20144.599609, vp_speed=1200.000000  
DPOS=20156.599609, vp_speed=1200.000000  
DPOS=20168.599609, vp_speed=1200.000000  
DPOS=20180.599609, vp_speed=1200.000000  
DPOS=20192.599609, vp_speed=1200.000000  
DPOS=20204.599609, vp_speed=1200.000000  
DPOS=20216.599609, vp_speed=1200.000000  
DPOS=20228.599609, vp_speed=1200.000000  
DPOS=20240.599609, vp_speed=1200.000000  
DPOS=20252.599609, vp_speed=1200.000000  
DPOS=20264.599609, vp_speed=1200.000000  
DPOS=20276.599609, vp_speed=1200.000000
```

12.5. RTSys Debug Software

12.5.1. What is RTSys

[RTSys](#) is one development software that integrates motion control and machine vision functions developed by Zmotion. It supports [ZBaisc \(RTBasic\)](#), ZPLC (RTPlc), [ZHMI \(RTHmi\)](#) and [ZVision \(RTVision\)](#) machine vision secondary development. Also, there are many functions, it supports hybrid programming, real-time simulation, online tracking, diagnosis and debugging, etc. What's more, for vision function, vision positioning, vision measurement, vision identification and vision detection of intellectual equipment can be realized rapidly, and complex motion control systems can be developed. RTSys software development page refers to below.

You can connect PC to controller through serial port or Ethernet, and program written by RTSys software can be downloaded into Zmotion motion controller directly to run offline or simulate on PC.



RTSys software supports 4 programming methods, Basic, PLC (ladder of diagram), HMI configuration and C language. In RTSys software, please note, for Basic language, it supports multiple Basic tasks operation, for PLC, it only can use one PLC task, for HMI, it only supports two HMI tasks. In addition, these languages' tasks can run at the same time.

RTSys supports Chinese and English interfaces by default, please check refer to the video showing in ["RTSys -- System Architecture -- E"](#).

RTSys software supports online simulation debugging, and it has simulator "ZMC Simulator" and configuration program simulation tool "xplc screen".

RTSys Doesn't Support Down to Compatible, that is, Please Don't Open RTSys File in ZDevelop. (ZDevelop is the original version)

For HMI Function, Controller's Firmware Must Support HMI Function.

12.5.2. How to Download RTSys

Please check ["RTSys User Manual – 1.2"](#).

12.5.3. How to Debug Controller

Please check ["RTSys User Manual – Chpater II & Chapter III"](#).

It includes RTSys interface introduction, controller connection, controller info checking, IO usage, axis parameters checking & configuration, manual motion, IP checking & configuration, etc.

Chapter XIII Instruction Details

Instruction 1	ZAux_OpenCom											
Original Format	int32 __stdcall ZAux_OpenCom(uint32 comid,ZMC_HANDLE *phandle)											
Description	Connect to controller through serial port.											
Input parameters	<table><tr><td>Parameter name</td><td colspan="3">Description</td></tr><tr><td>Comid</td><td colspan="3">Connected COM No.</td></tr></table>				Parameter name	Description			Comid	Connected COM No.		
Parameter name	Description											
Comid	Connected COM No.											
Output parameters	<table><tr><td>Parameter name</td><td colspan="3">Description</td></tr><tr><td>Phandle</td><td colspan="3">Controller link handle</td></tr></table>				Parameter name	Description			Phandle	Controller link handle		
Parameter name	Description											
Phandle	Controller link handle											
Return value	0 = success, not-0 = failure (please check error codes for details).											
Example	Connect to Controller by Serial Port											
Details	There are many ways to connect to controller, this is mainly for serial (but communication speed is lower than ethernet). ➤ ZMC_HANDLE type: in Zmotion library, it is specified for type definition for control card connection. ➤ Controller default serial port parameters: <table><tr><td>Baud rate</td><td>Data bit</td><td>Stop bit</td><td>Parity</td></tr><tr><td>38400</td><td>8</td><td>1</td><td>No</td></tr></table>				Baud rate	Data bit	Stop bit	Parity	38400	8	1	No
Baud rate	Data bit	Stop bit	Parity									
38400	8	1	No									

Instruction 2	ZAux_SetComDefaultBaud										
Original Format	int32 __stdcall ZAux_SetComDefaultBaud(uint32 dwBaudRate, uint32 dwByteSize, uint32 dwParity, uint32 dwStopBits)										
Description	Set serial port communication parameters.										
IN parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>dwBaudRate</td><td>Baud rate (default): 38400</td></tr> <tr> <td>dwByteSize</td><td>Data bit size (default): 8-bit</td></tr> <tr> <td>dwParity</td><td>Parity (default): 0-no parity</td></tr> <tr> <td>dwStopBits</td><td>Stop bit (default): 1-bit</td></tr> </tbody> </table>	Parameter name	Description	dwBaudRate	Baud rate (default): 38400	dwByteSize	Data bit size (default): 8-bit	dwParity	Parity (default): 0-no parity	dwStopBits	Stop bit (default): 1-bit
Parameter name	Description										
dwBaudRate	Baud rate (default): 38400										
dwByteSize	Data bit size (default): 8-bit										
dwParity	Parity (default): 0-no parity										
dwStopBits	Stop bit (default): 1-bit										
OP Parameters	/										
Return value	0 = success, not-0 = failure (please check error codes for details).										
Example	Connect to Controller by Serial Port										
Details	Not same as RTBasic SetCom, this is only to set serial parameters.										

Instruction 3	ZAux_OpenEth
Original Format	int32 __stdcall ZAux_OpenEth(char *ipaddr, ZMC_HANDLE *phandle)
Description	Connect to controller through ethernet port.

Input parameters	Parameter name	Description
	Ipaddr	Controller IP
Output parameters	Parameter name	Description
	Phandle	Returned link handle
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Connect to Controller by Ethernet Port	
Details	- It uses standard RJ45 interface of 100Mbit/s. - The factory IP address of the controller is 192.168.0.11, and the port No. is 502. And they should in same network segment (front 3 segments are same, that last one is different). - It is used mostly - ZMC_HANDLE type: in the Zmotion library, it is specially used to define the control card connection type.	

Instruction 4	ZAux_OpenPci	
Original Format	int32 __stdcall ZAux_OpenPci(uint32 cardnum, ZMC_HANDLE *phandle)	
Description	Connect to controller by PCI.	
Input parameters	Parameter name	Description
	Cardnum	PCI card No., default starts from 0.
Output parameters	Parameter name	Description
	Phandle	link handle
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Connect to controller by PCI.	
Details	- It is only valid in PCI type card, it is high communication velocity, but please insert it into IPC that has PCI slot. - ZMC_HANDLE type: in the Zmotion library, it is specially used to define the control card connection type.	

Instruction 5	ZAux_Close	
Original Format	int32 __stdcall ZAux_Close(ZMC_HANDLE handle)	
Description	Close controller connection	
Input parameters	Parameter name	Description
	Handle	Link handle
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	

Example	Connect to Controller by Serial Port
Details	/

Instruction 6	ZAux_SetIp						
Original Format	int32 __stdcall ZAux_SetIp(ZMC_HANDLE handle, char * ipaddress)						
Description	Set controller IP address						
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Handle</td><td>Link handle</td></tr> <tr> <td>Ipaddress</td><td>Set IP address value</td></tr> </tbody> </table>	Parameter name	Description	Handle	Link handle	Ipaddress	Set IP address value
Parameter name	Description						
Handle	Link handle						
Ipaddress	Set IP address value						
OP Parameters	/						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	Modify IP address						
Details	If the controller is connected by Ethernet, after the IP address is successfully modified, the controller will automatically disconnect from the PC, at this time, please set PC IP as the corresponding network segment before reconnecting.						

Instruction 7	ZAux_SearchEthlist						
Original Format	int32 __stdcall ZAux_SearchEthlist(char *ipaddrlist, uint32 addrbufflength, uint32 uims)						
Description	Search IP addresses under the current network segment.						
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Addrbufflength</td><td>Search returned IP address total length</td></tr> <tr> <td>Uims</td><td>Search timeout time.</td></tr> </tbody> </table>	Parameter name	Description	Addrbufflength	Search returned IP address total length	Uims	Search timeout time.
Parameter name	Description						
Addrbufflength	Search returned IP address total length						
Uims	Search timeout time.						
Output parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Ipaddrlist</td><td>Searched all IP addresses</td></tr> </tbody> </table>	Parameter name	Description	Ipaddrlist	Searched all IP addresses		
Parameter name	Description						
Ipaddrlist	Searched all IP addresses						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	<pre>//search IP address automatically char buffer[4096+256+1]; int32 iresult; iresult = ZAux_SearchEthlist(buffer, 4096+256+1, 100); //search IP address to Buffer if(0!=iresult){return 0;}//fail and exit //converted from character string int ipos =0; const char * pstring; pstring = buffer; char buffer2[256]; buffer2[0] = '\0'; for(int j= 0; j< 20;j++)</pre>						

	<pre> { //skip blank space while(' ' == pstring[0]) { pstring++; } memset(buffer2, 0, sizeof(buffer2)); ipos = sscanf(pstring, "%s", &buffer2); //obtain address to buffer2 if(-1 == ipos){break;} printf("%s\n",buffer2); //point to IP address ending while((' ' != pstring[0]) && ('\t' != pstring[0]) && ('\0' != pstring[0])) { pstring++; } } </pre>
Details	Controller default IP address is 192.168.0.11. If the IP addresses of the controller and the PC are not in the same network segment, the IP address will not be scanned

Instruction 8	ZAux_SearchAndOpenCom								
Original Format	int32 __stdcall ZAux_SearchAndOpenCom(uint32 uimincomidfind, uint32 uimaxcomidfind, uint *pcomid, uint32 uims, ZMC_HANDLE *phandle)								
Description	Search serial port automatically and connect to controller.								
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>uimincomidfind</td><td>Searched minimal COM No.</td></tr> <tr> <td>uimaxcomidfind</td><td>Searched maximum COM No.</td></tr> <tr> <td>uims</td><td>Connection respond time MS</td></tr> </tbody> </table>	Parameter name	Description	uimincomidfind	Searched minimal COM No.	uimaxcomidfind	Searched maximum COM No.	uims	Connection respond time MS
Parameter name	Description								
uimincomidfind	Searched minimal COM No.								
uimaxcomidfind	Searched maximum COM No.								
uims	Connection respond time MS								
Output parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Pcomid</td><td>Searched COM No.</td></tr> <tr> <td>phandle</td><td>Returned handle</td></tr> </tbody> </table>	Parameter name	Description	Pcomid	Searched COM No.	phandle	Returned handle		
Parameter name	Description								
Pcomid	Searched COM No.								
phandle	Returned handle								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	/								
Details	/								

Instruction 9	ZAux_SearchEth
Original Format	int32 __stdcall ZAux_SearchEth(const char *ipaddress, uint32 uims)
Description	Search whether the current IP address is in this network segment.

Input parameters	Parameter name	Description
	Pcomid	Search judged IP address
	phandle	Search timeout time
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	/	

Instruction 10	ZAux_GetMaxPciCards
Original Format	int32 __stdcall ZAux_GetMaxPciCards()
Description	Search current max PCI card numbers.
IN Parameters	/
OP Parameters	/
Return value	Max card numbers.
Example	Int Card; Card=ZAux_GetMaxPciCards(); //obtain max card numbers
Details	/

Instruction 11	ZAux_FastOpen	
Original Format	int32 __stdcall ZAux_FastOpen(int type, char *pconnectstring, uint32 uims ,ZMC_HANDLE * phandle)	
Description	Build the connection with controller, it can specify the connection waiting time.	
Input parameters	Parameter name	Description
	type	Connection type type: 1-COM, 2-ETH, 4-PCI, 5-MotionRT
	pconnectstring	Connection character string: type=1: COM + serial port No. type=2: IP address type=4: Pci + card No. type=5: LOCAL + card No.
	uims	Connection timeout time uims, unit: ms.
Output parameters	Parameter name	Description
	phandle	Link handle
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Connect by serial port <pre>ZMC_HANDLE phandle; //controller connection handle Char comID[32]= "0"; //serial port ID, serial port No. cannot be written directly, Com should be added before the serial port No. ZAux_FastOpen(1, comID,1000s,&phandle);</pre>	

	- Connect by ethernet ZMC_HANDLE phandle; //controller connection handle Char EthID[32]= "192.168.0.11";//ethernet port ID ZAux_FastOpen(1, EthID,1000s,&phandle); - Connect by PCI ZMC_HANDLE phandle; //controller connection handle ZAux_FastOpen(4, "PCI",1000,&phandle); - Connect by MotionRT ZMC_HANDLE phandle; //controller connection handle ZAux_FastOpen(5, "LOCAL1",1000,&phandle);
Details	"type" = 5" ✓ Zmotion.dll version should be above 3.8.8.50 ✓ if the character string is not LOCAL + card No., it is LOCAL1 automatically.

Instruction 12	ZAux_GetControllerInfo	
Original Format	int32 __stdcall ZAux_GetControllerInfo(ZMC_HANDLE handle ,char *SoftType ,char *SoftVersion ,char *ControllerId)	
Description	Read controller manufacturer information	
Input parameters	Parameter name	Description
	handle	Controller link handle
Output parameters	Parameter name	Description
	SoftType	Controller software model
	SoftVersion	Controller firmware version No.
	ControlllerId	Controller unique ID
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Controller Information	
Details	/	

Instruction 13	ZAux_GetSysSpecification	
Original Format	int32 __stdcall ZAux_GetSysSpecification(ZMC_HANDLE handle, uint16 *Max_VirtuAxes,uint8 *Max_motor,uint8 *Max_io)	
Description	Read the controller max specification.	
Input parameters	Parameter name	Description
	handle	Controller link handle
Output parameters	Parameter name	Description
	Max_VirtuAxes	Controller max virtual axes numbers.
	Max_motor	The number of controller actual axes.
	Max_io	IN, OUT, AD, DA, each occupies one unit8.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Controller Information	

Details	/

Instruction 14	ZAux_GetRtcTime	
Original Format	int32 __stdcall ZAux_GetRtcTim(ZMC_HANDLE handle, char *RtcDate, char * RtcTime)	
Description	Read controller system time & date. Default starts from 2000.1.1.	
Input parameters	Parameter name	Description
	handle	Controller link handle
Output parameters	Parameter name	Description
	RtcData	Controller system data. Format: YYMMDD / YYYYMMDD
	RtcTime	Controller system time. Format: HHMMSS
Return value	Please refer to error codes .	
Example	<pre>char RtcDate[6]; char RtcTime[6]; ZAux_GetRtcTime(handle,RtcDate,RtcTime);</pre>	
Details	- YYMMDD format: e.g.: 220101 – 1st Jan, 2022 - YYYYMMDD format: e.g.: 20220101 – 1st Jan, 2022 - HHMMSS format: e.g.: 131400 – 13:14:00	

Instruction 15	ZAux_SetRtcTime	
Original Format	int32 __stdcall ZAux_SetRtcTim(ZMC_HANDLE handle, char *RtcDate, char * RtcTime)	
Description	Set controller system time & date, they are saved when power-off . Default starts from 2000.1.1. But it can't modify simulator time.	
Input parameters	Parameter name	Description
	handle	Controller link handle
	RtcData	Controller system data. Formats: YYMMDD / YYYYMMDD
	RtcTime	Controller system time. Format: HHMMSS
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	<pre>char RtcDate[20]="20220101"; char RtcTime[20]="000000"; ZAux_SetRtcTime(handle,RtcDate,RtcTime);</pre>	
Details	- YYMMDD format: e.g.: 220101 – 1st Jan, 2022 - YYYYMMDD format: e.g.: 20220101 – 1st Jan, 2022 - HHMMSS format: e.g.: 131400 – 13:14:00	

Instruction 16	ZAux_Direct_SetAtype	
Original Format	int32 __stdcall ZAux_Direct_SetAtype(ZMC_HANDLE handle, int iaxis, int iValue)	
Description	Set the axis type of specified axis	
Input parameters	Parameter name	Description
	handle	Controller link handle
	iaxis	Axis No.
	iValue	Axis type
OP Parameters	/	
Return value	Please refer to error codes.	
Example	Get axis basic motion parameters configuration	
Details 1	● Please set ATYPE in "initialization" part. ● For axis that is extended by ZCAN must set AXIS_ADDRESS at first, after that, delay 2 task cycles, then call motion command. Due to the limitation of the bus bandwidth, do not set more than 2 ZCAN expanded axes. ● For some product models that have the independent encoder, the corresponding virtual axis can be used as the encoder axis. For example, the motor axis of ZMC206 is 0-5 axis, and the encoder can be controlled by axis 6-11. For details, please refer to controller state in RTSys.	
Details 2	Atype setting	

Instruction 17	ZAux_Direct_GetAtype	
Original Format	int32 __stdcall ZAux_Direct_GetAtype(ZMC_HANDLE handle, int iaxis, int *piValue)	
Description	Read the axis type of specified axis	
Input parameters	Parameter name	Description
	handle	Controller link handle
	iaxis	Axis No.
Output parameters	Parameter name	Description
	piValue	Axis type
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Get axis basic motion parameters configuration	
Detail 1	/	
Details 2	Atype setting	

Instruction 18	ZAux_Direct_SetUnits	
Original Format	int32 __stdcall ZAux_Direct_SetUnits(ZMC_HANDLE handle, int iaxis, float fValue)	
Description	This is pulse amount value, which is controller basic unit ,	

	specifying how many pulses per unit, the precision can achieve 5-digit after decimal. For example, 1 means the unit is 1 pulse. Note: the coordinates displayed will change with pulse amount.								
Input parameters	<table><tr><th>Parameter name</th><th>Description</th></tr><tr><td>handle</td><td>Controller link handle</td></tr><tr><td>iaxis</td><td>Axis No.</td></tr><tr><td>fValue</td><td>pulse amount that is set.</td></tr></table>	Parameter name	Description	handle	Controller link handle	iaxis	Axis No.	fValue	pulse amount that is set.
Parameter name	Description								
handle	Controller link handle								
iaxis	Axis No.								
fValue	pulse amount that is set.								
OP Parameters	/								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	Get axis basic motion parameters configuration								
Details	How to set? If the motor's pulse is (U) 3600, to make one circle, then the screw pitch P=2mm per revolution: • The pulse amount UNITS when the motor rotates 1°: $UNITS = U / 360 = 3600 / 360 = 10$. • The pulse amount UNTIS when the workbench moves 1mm: $UNITS = U / P = 3600 / 2 = 1800$. If the machine has a reduction ratio , please include it: • When the reduction ratio i=2:1: $UNITS = U * i / P = 3600 * 2 / 2 = 3600$								

Instruction 19	ZAux_Direct_GetUnits						
Original Format	int32 __stdcall ZAux_Direct_GetUnits(ZMC_HANDLE handle,int iaxis, float *pfValue)						
Description	It reads the pulse amount value. Pulse amount value is controller basic unit , specifying how many pulses per unit, the precision can achieve 5-digit after decimal. Note: the coordinates displayed will change with pulse amount.						
Input parameters	<table><tr><th>Parameter name</th><th>Description</th></tr><tr><td>handle</td><td>Controller link handle</td></tr><tr><td>iaxis</td><td>Axis No.</td></tr></table>	Parameter name	Description	handle	Controller link handle	iaxis	Axis No.
Parameter name	Description						
handle	Controller link handle						
iaxis	Axis No.						
Output parameters	<table><tr><th>Parameter name</th><th>Description</th></tr><tr><td>pfValue</td><td>Returned pulse amount</td></tr></table>	Parameter name	Description	pfValue	Returned pulse amount		
Parameter name	Description						
pfValue	Returned pulse amount						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	Get axis basic motion parameters configuration						
Details	/						

Instruction 20	ZAux_Direct_SetAccel
Original Format	int32 __stdcall ZAux_Direct_SetAccel(ZMC_HANDLE handle,int iaxis, float fValue)
Description	Set acceleration, the unit: units/s/s.

Input parameters	Parameter name	Description
	handle	Controller link handle
	iaxis	Axis No.
	pfValue	Acceleration that is set.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Get axis basic motion parameters configuration	
Details	- For multi-axis interpolation motion, the acceleration refers to spindle acceleration (in axis list, the first axis No.). - It is recommended to set the acceleration & deceleration before motion, and don't modify it during motion, otherwise it will cause incorrect speed curve.	

Instruction 21	ZAux_Direct_GetAccel	
Original Format	int32 __stdcall ZAux_Direct_GetAccel(ZMC_HANDLE handle,int iaxis,float *pfValue)	
Description	Read acceleration, unit: units/s/s.	
Input parameters	Parameter name	Description
	handle	Controller link handle
	iaxis	Axis No.
Output parameters	Parameter name	Description
	pfValue	Returned acceleration
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Get axis basic motion parameters configuration	
Details	- For multi-axis interpolation motion, the acceleration refers to spindle acceleration (in axis list, the first axis No.). - It is recommended to set the acceleration & deceleration before motion, and don't modify it during motion, otherwise it will cause incorrect speed curve.	

Instruction 22	ZAux_Direct_SetDecel	
Original Format	int32 __stdcall ZAux_Direct_SetDecel(ZMC_HANDLE handle,int iaxis,float fValue)	
Description	Set deceleration, unit: units/s/s	
Input parameters	Parameter name	Description
	handle	Controller link handle
	iaxis	Axis No.
	pfValue	Deceleration that is set.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Get axis basic motion parameters configuration	

Details	- For multi-axis interpolation motion, the deceleration refers to spindle deceleration (in axis list, the first axis No.). - It is recommended to set the acceleration & deceleration before motion, and don't modify it during motion, otherwise it will cause incorrect speed curve. - When it is 0, DECEL uses ACCEL value.

Instruction 23	ZAux_Direct_GetDecel						
Original Format	int32 __stdcall ZAux_Direct_GetDecel(ZMC_HANDLE handle,int iaxis, float *pfValue);						
Description	Read deceleration, unit: units/s/s						
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Controller link handle</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </table>	Parameter name	Description	handle	Controller link handle	iaxis	Axis No.
Parameter name	Description						
handle	Controller link handle						
iaxis	Axis No.						
Output parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>pfValue</td><td>Get deceleration</td></tr> </table>	Parameter name	Description	pfValue	Get deceleration		
Parameter name	Description						
pfValue	Get deceleration						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	Get axis basic motion parameters configuration						
Details	- For multi-axis interpolation motion, the deceleration refers to spindle deceleration (in axis list, the first axis No.). - It is recommended to set the acceleration & deceleration before motion, and don't modify it during motion, otherwise it will cause incorrect speed curve. - When it is 0, DECEL uses ACCEL value.						

Instruction 24	ZAux_Direct_SetSpeed								
Original Format	int32 __stdcall ZAux_Direct_SetSpeed(ZMC_HANDLE handle,int iaxis, float fValue)								
Description	Set axis speed, unit: units/s.								
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Controller link handle</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>pfValue</td><td>Speed that is set.</td></tr> </table>	Parameter name	Description	handle	Controller link handle	iaxis	Axis No.	pfValue	Speed that is set.
Parameter name	Description								
handle	Controller link handle								
iaxis	Axis No.								
pfValue	Speed that is set.								
OP Parameters	/								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	Get axis basic motion parameters configuration								
Details	- For multi-axis interpolation motion, the speed refers to spindle speed (in axis list, the first axis No.) - After the speed is modified, it will take effect immediately, therefore, dynamic speed can be realized (it will shake on the instant of changing). If you want to change speed smoothly, please use SPEED_RATIO (this is RTbasic command,								

	corresponding PC command: ZAux_Direct_SetParam (handle, "SPEED_RATIO", axis No., current speed ratio).

Instruction 25	ZAux_Direct_GetSpeed						
Original Format	int32 __stdcall ZAux_Direct_GetSpeed(ZMC_HANDLE handle,int iaxis, float *pfValue)						
Description	Read axis speed.						
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Controller link handle</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </table>	Parameter name	Description	handle	Controller link handle	iaxis	Axis No.
Parameter name	Description						
handle	Controller link handle						
iaxis	Axis No.						
Output parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>pfValue</td><td>Get speed</td></tr> </table>	Parameter name	Description	pfValue	Get speed		
Parameter name	Description						
pfValue	Get speed						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	Get axis basic motion parameters configuration						
Details	For multi-axis interpolation motion, the speed refers to spindle speed (in axis list, the first axis No.)						

Instruction 26	ZAux_Direct_SetDpos								
Original Format	int32 __stdcall ZAux_Direct_SetDpos(ZMC_HANDLE handle,int iaxis, float fValue)								
Description	Set axis current position or the demand position sent by controller, the unit: units.								
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Controller link handle</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>pfValue</td><td>Current axis position</td></tr> </table>	Parameter name	Description	handle	Controller link handle	iaxis	Axis No.	pfValue	Current axis position
Parameter name	Description								
handle	Controller link handle								
iaxis	Axis No.								
pfValue	Current axis position								
OP Parameters	/								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	Get axis basic motion parameters configuration								
Details	It will automatically convert into OFFPOS offset when DPOS is written, motor will not be moved.								

Instruction 27	ZAux_Direct_GetDpos						
Original Format	int32 __stdcall ZAux_Direct_GetDpos(ZMC_HANDLE handle, int iaxis, float *pfValue)						
Description	Read axis current position or the demand position sent by controller, the unit: units.						
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Controller link handle</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </table>	Parameter name	Description	handle	Controller link handle	iaxis	Axis No.
Parameter name	Description						
handle	Controller link handle						
iaxis	Axis No.						

Output parameters	Parameter name	Description
	pfValue	Get current axis position.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Get axis basic motion parameters configuration	
Details	/	

Instruction 28	ZAux_Direct_SetMpos									
Original Format	int32 __stdcall ZAux_Direct_SetMpos(ZMC_HANDLE handle, int iaxis, float fValue)									
Description	Set axis measurement position, the unit is units.									
Input parameters	<table><tr><td>Parameter name</td><td>Description</td></tr><tr><td>handle</td><td>Controller link handle</td></tr><tr><td>iaxis</td><td>Axis No.</td></tr><tr><td>pfValue</td><td>Current axis measurement position value</td></tr></table>		Parameter name	Description	handle	Controller link handle	iaxis	Axis No.	pfValue	Current axis measurement position value
	Parameter name	Description								
	handle	Controller link handle								
	iaxis	Axis No.								
pfValue	Current axis measurement position value									
OP Parameters	/									
Return value	0 = success, not-0 = failure (please check error codes for details).									
Example	Get axis basic motion parameters configuration									
Details	It will automatically convert into OFFPOS offset when MPOS is written, motor will not be moved.									

Instruction 29	ZAux_Direct_GetMpos	
Original Format	int32 __stdcall ZAux_Direct_GetMpos(ZMC_HANDLE handle, int iaxis, float *pfValue)	
Description	Read axis measurement position, the unit is units.	
Input parameters	Parameter name	Description
	handle	Controller link handle
	iaxis	Axis No.
Output parameters	Parameter name	Description
	pfValue	Get current axis measurement position.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Get axis basic motion parameters configuration	
Details	/	

Instruction 30	ZAux_Direct_GetEncoder
Original Format	int32 __stdcall ZAux_Direct_GetEncoder(ZMC_HANDLE handle, int iaxis, float *pfValue);
Description	Read internal encoder value, the unit is pulse

Input parameters	Parameter name	Description
	handle	Controller link handle
	iaxis	Axis No.
Output parameters	Parameter name	Description
	pfValue	Return to current axis measurement position whose unit is pulse.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	<pre>float fValue; ret = ZAux_Direct_GetEncoder(handle, 0, &fValue); //read axis 0 inner encoder value</pre>	
Details	- The internal parameters can be read only when the ATYPE is configured as encoder type. - When the drive has multi-turn absolute encoder, the multi-turn value is read.	

Instruction 31	ZAux_Direct_GetFastDec	
Original Format	int32 __stdcall ZAux_Direct_GetFastDec(ZMC_HANDLE handle, int iaxis, float * fValue)	
Description	Read the fast deceleration. The unit is units/s/s. It will be used automatically when it stops abnormally.	
Input parameters	Parameter name	Description
	handle	Controller link handle
	iaxis	Axis No.
Output parameters	Parameter name	Description
	pfValue	Get fast deceleration value
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Position limit setting	
Details	It is used automatically when CANCEL or abnormally stops. If it is 0, the value equals to DECEL.	

Instruction 32	ZAux_Direct_SetFastDec	
Original Format	int32 __stdcall ZAux_Direct_SetFastDec(ZMC_HANDLE handle, int iaxis, float fValue)	
Description	Set the fast deceleration. The unit is units/s/s. It will be used automatically when it stops abnormally.	
Input parameters	Parameter name	Description
	handle	Controller link handle
	iaxis	Axis No.
	pfValue	Fast deceleration value
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	

Example	Position limit setting
Details	It is used automatically when CANCEL or abnormally stops. When the value is set to 0, it uses DECEL automatically.

Instruction 33	ZAux_Direct_SetLspeed								
Original Format	int32 __stdcall ZAux_Direct_SetLspeed(ZMC_HANDLE handle, int iaxis, float fValue)								
Description	Set starting speed, the unit is units/s.								
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Controller link handle</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>pfValue</td><td>Starting speed that is set</td></tr> </tbody> </table>	Parameter name	Description	handle	Controller link handle	iaxis	Axis No.	pfValue	Starting speed that is set
Parameter name	Description								
handle	Controller link handle								
iaxis	Axis No.								
pfValue	Starting speed that is set								
OP Parameters	/								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	Get axis basic motion parameters configuration (with initial speed & S curve)								
Details	- When in multi-axis motion, set the main axis as the initial speed of interpolation motion. - When you need the efficiency, you could set one start speed.								

Instruction 34	ZAux_Direct_GetLspeed						
Original Format	int32 __stdcall ZAux_Direct_GetLspeed(ZMC_HANDLE handle, int iaxis, float *pfValue)						
Description	Read starting speed, the unit is units/s.						
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Controller link handle</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </tbody> </table>	Parameter name	Description	handle	Controller link handle	iaxis	Axis No.
Parameter name	Description						
handle	Controller link handle						
iaxis	Axis No.						
Output parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>pfValue</td><td>Returned starting speed</td></tr> </tbody> </table>	Parameter name	Description	pfValue	Returned starting speed		
Parameter name	Description						
pfValue	Returned starting speed						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	Get axis basic motion parameters configuration (with initial speed & S curve)						
Details	- When in multi-axis motion, set the main axis as the initial speed of interpolation motion. - When you need the efficiency, you could set one start speed.						

Instruction 35	ZAux_Direct_SetSramp
Original Format	int32 __stdcall ZAux_Direct_SetSramp(ZMC_HANDLE handle, int iaxis, float fValue)
Description	Set S curve time, the unit is ms0. 0 = T-shaped acceleration & deceleration

Input parameters	Parameter name	Description
	handle	Controller link handle
	iaxis	Axis No.
	pfValue	S curve that is set
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Get axis basic motion parameters configuration (with initial speed & S curve)	
Details	- After that, the acceleration & deceleration will be longer. - If the max value is set, the acceleration time will be original T-shape acceleration.	

Instruction 36	ZAux_Direct_GetSramp	
Original Format	int32 __stdcall ZAux_Direct_GetSramp(ZMC_HANDLE handle, int iaxis, float *pfValue)	
Description	Read S curve time.	
Input parameters	Parameter name	Description
	handle	Controller link handle
	iaxis	Axis No.
Output parameters	Parameter name	Description
	pfValue	Returned S curve time value
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Get axis basic motion parameters configuration (with initial speed & S curve)	
Details	- After that, the acceleration & deceleration will be longer. - If the max value is set, the acceleration time will be original T-shape acceleration.	

Instruction 37	ZAux_Direct_GetMspeed	
Original Format	int32 __stdcall ZAux_Direct_GetMspeed(ZMC_HANDLE handle, int iaxis, float *pfValue)	
Description	Read current axis' real-time measurement speed, the unit is units.	
Input parameters	Parameter name	Description
	handle	Controller link handle
	iaxis	Axis No.
OP Parameters	Parameter name	Description
	pfValue	real-time axis measurement speed.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	float fValue; ret = ZAux_Direct_GetMspeed(handle, 0, &fValue);	

	//read axis 0 current measurement speed. handle: controller link handle
Details	/

Instruction 38	ZAux_Direct_GetVpSpeed						
Original Format	int32 __stdcall ZAux_Direct_GetVpSpeed (ZMC_HANDLE handle, int iaxis, float *pfValue)						
Description	Read current real-time axis running speed sent by the controller, the unit is units.						
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Controller link handle</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </tbody> </table>	Parameter name	Description	handle	Controller link handle	iaxis	Axis No.
Parameter name	Description						
handle	Controller link handle						
iaxis	Axis No.						
Output parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>pfValue</td><td>Returned real-time axis speed.</td></tr> </tbody> </table>	Parameter name	Description	pfValue	Returned real-time axis speed.		
Parameter name	Description						
pfValue	Returned real-time axis speed.						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	<pre>float fValue; ret = ZAux_Direct_GetVpSpeed(handle, 0, &fValue); //read axis 0 current command motion speed. handle: controller link handle</pre>						
Details	- For multi-axis interpolation motion, it returns main axis speed for interpolation speed, not the sub-speed of the main axis. - The non-spindle axis returns the corresponding sub-speed, which is consistent with the feedback speed effect. - By default, the planning speed is designed to display the multi-axis composed speed, there is no negative value, unless the SYSTEM_ZSET (this is RTBasic command, corresponding PC command: ZAux_Direct_SetParam (handle, "SYSTEM_ZSET", axis No., setting value) bit0 value is set to 0, then it can be used to display single axis' command speed, which can be positive or negative.						

Instruction 39	ZAux_Direct_GetIfIdle						
Original Format	int32 __stdcall ZAux_Direct_GetIfIdle(ZMC_HANDLE handle, int iaxis, int *piValue)						
Description	Read whether the current axis is running.						
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Controller link handle</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </tbody> </table>	Parameter name	Description	handle	Controller link handle	iaxis	Axis No.
Parameter name	Description						
handle	Controller link handle						
iaxis	Axis No.						
Output parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>pfValue</td><td>Returned state: 0-in motion, -1-no motion</td></tr> </tbody> </table>	Parameter name	Description	pfValue	Returned state: 0-in motion, -1-no motion		
Parameter name	Description						
pfValue	Returned state: 0-in motion, -1-no motion						
Return value	0 = success, not-0 = failure (please check error codes for details).						

Example	Multi-axis linear interpolation
Details	When the axis is associated with a manipulator, under CONNFRAME inverse mode, the joint axis always returns 0, under CONNREFRAME forward mode, the virtual axis always returns 0.

Instruction 40	ZAux_Direct_GetAxisStatus	
Original Format	int32 __stdcall ZAux_Direct_GetAxisStatus(ZMC_HANDLE handle, int iaxis, int *piValue)	
Description	Read current axis' state. The value is displayed in decimal system, but the state is judged in binary system.	
Input parameters	Parameter name	Description
	handle	Controller link handle
	iaxis	Axis No.
Output parameters	Parameter name	Description
	pfValue	Returned state value (different bit mean different states).
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Get axis basic motion parameters configuration (with initial speed & S curve)	
Details	Returned state: corresponding bit means different states.	

Instruction 41	ZAux_Direct_GetAxisStopReason	
Original Format	int32 __stdcall ZAux_Direct_GetAxisStopReason(ZMC_HANDLE handle, int iaxis, int *piValue)	
Description	Read axis history state.	
Input parameters	Parameter name	Description
	handle	Controller link handle
	iaxis	Axis No.
Output parameters	Parameter name	Description
	pfValue	Returned state value (different bit mean different states).
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	float fValue; ZAux_Direct_GetAxisStopReason(handle, 0, &fValue); //read why axis 0 stops. handle: controller link handle	
Details	Returned state: corresponding bit means different states.	

Instruction 42	ZAux_GetModbusDpos	
Original	int32 __stdcall ZAux_GetModbusDpos (ZMC_HANDLE handle , int	

Format	imaxaxes, float * pValueList);	
Description	Rapidly read current DPOS of multi-axis. Use MODBUS register.	
Input parameters	Parameter name	Description
	handle	Controller link handle
	iaxis	Axis numbers
Output parameters	Parameter name	Description
	pfValue	DPOS coordinates value that is read, starting from 0.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Get and set parameters	
Details	It reads data starting from 0, this is the fastest way to read speed.	

Instruction 43	ZAux_GetModbusMpos	
Original Format	int32 __stdcall ZAux_GetModbusMpos (ZMC_HANDLE handle , int imaxaxes, float * pValueList);	
Description	Rapidly read current MPOS of multi-axis. Use MODBUS register.	
Input parameters	Parameter name	Description
	handle	Controller link handle
	iaxis	Axis numbers
Output parameters	Parameter name	Description
	pfValue	Measurement coordinates value that is read, starting from 0.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Get and set parameters	
Details	It reads data starting from 0, this is the fastest way to read speed.	

Instruction 44	ZAux_GetModbusCurSpeed	
Original Format	int32 __stdcall ZAux_GetModbusCurSpeed (ZMC_HANDLE handle , int imaxaxes, float * pValueList);	
Description	Rapidly read current speed of multi-axis. Use MODBUS register.	
Input parameters	Parameter name	Description
	handle	Controller link handle
	iaxis	Axis numbers
Output parameters	Parameter name	Description
	pfValue	Current speed that is read, starting from 0.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Get and set parameters	
Details	It reads data starting from 0, this is the fastest way to read speed.	

Instruction 45	ZAux_Direct_SetAxisAddress												
Original Format	int32 __stdcall ZAux_Direct_SetAxisAddress (ZMC_HANDLE handle, int iaxis, int iValue)												
Description	Set axis address.												
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Controller link handle</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>iValue</td><td>Configured address</td></tr> </table>	Parameter name	Description	handle	Controller link handle	iaxis	Axis No.	iValue	Configured address				
Parameter name	Description												
handle	Controller link handle												
iaxis	Axis No.												
iValue	Configured address												
OP Parameters	/												
Return value	0 = success, not-0 = failure (please check error codes for details).												
Example	Local pulse axis No. remapping												
Details	1) When ZCAN expands the axis, there is an 8-bit DIP switch on the expansion board (hardware version above V1.3). - Limited by the bus bandwidth, up to 2 CAN extended axes. - AXIS_ADDRESS must be set firstly, then set the axis type ATYPE for ZCAN extended axis (ATYPE must be reset after modification). <table> <tr> <td>Code 1-4</td><td>For CAN address, combination value 0-15</td></tr> <tr> <td>Code 5-6</td><td>For CAN speed, different combination values are with different speed values.</td></tr> <tr> <td>Code 7</td><td>Reserved for special functions</td></tr> <tr> <td>Code 8</td><td>For 120-ohm resistor, the resistance is connected when dialing ON.</td></tr> </table> Rule: <i>ZAux_Direct_SetAxisAddress (handle, axis No., (32*0) + ID)</i> //The local axis 0 of the CAN expansion board <i>ZAux_Direct_SetAxisAddress (handle, axis No., (32*1) + ID)</i> // The local axis 1 of the CAN expansion board 2) Bus drive axis No. mapping: map connected drives one by one according to the No. - The drive No. is arranged by the wiring sequence, starting from 0 to the number of EtherCAT drives - 1. - The drive No. is different from the device No.: the device No. includes all the devices that are connected to ECAT, the drive No. only contains the connected drives. - AXIS_ADDRESS must be set firstly, then set ECAT axis type (ATYPE must be reset after modification). <table> <tr> <td>Bit 0-15</td><td>drive No. + 1. 0 = auto-specify</td></tr> <tr> <td>Bit 16-31</td><td>SLOT No. (under multi-slot)</td></tr> </table> Rule: <i>ZAux_Direct_SetAxisAddress (handle, axis No., (slot No. << 16) + driver No. + 1)</i>	Code 1-4	For CAN address, combination value 0-15	Code 5-6	For CAN speed, different combination values are with different speed values.	Code 7	Reserved for special functions	Code 8	For 120-ohm resistor, the resistance is connected when dialing ON.	Bit 0-15	drive No. + 1. 0 = auto-specify	Bit 16-31	SLOT No. (under multi-slot)
Code 1-4	For CAN address, combination value 0-15												
Code 5-6	For CAN speed, different combination values are with different speed values.												
Code 7	Reserved for special functions												
Code 8	For 120-ohm resistor, the resistance is connected when dialing ON.												
Bit 0-15	drive No. + 1. 0 = auto-specify												
Bit 16-31	SLOT No. (under multi-slot)												

	3) Local pulse axis No. remapping (for ZMC4XX series and above, they support local pulse or encoder axis No. remapping, and the firmware version should be 160608 or above).			
	When remapping, please set the original pulse axis as a virtual axis at first (ATYPE must be reset after modification). See command example: Local pulse axis No. remapping			
	<table border="1"> <tr> <td>Bit 0-15</td><td>Local pulse axis No. mapping</td></tr> <tr> <td>Bit 16-31</td><td>the upper 16 bits are all set to 1 (which means "high 16 bits in decimal = -1")</td></tr> </table>	Bit 0-15	Local pulse axis No. mapping	Bit 16-31
Bit 0-15	Local pulse axis No. mapping			
Bit 16-31	the upper 16 bits are all set to 1 (which means "high 16 bits in decimal = -1")			

	Rule: <i>ZAux_Direct_SetAtype (handle, remapped axis No., 0)</i> //set the axis type to 0, it will alarm for low version. <i>ZAux_Direct_SetAtype(handle, modified local pulse axis No., 0)</i> //set the axis type to 0, it will alarm for low version. <i>ZAux_Direct_SetAxisAddress (handle, remapped axis No., (-1 << 16) + local pulse axis No. to be modified)</i> <i>ZAux_Direct_SetAtype (handle, remapped axis No., 1 or 7)</i>

Instruction 46	ZAux_Direct_GetAxisAddress						
Original Format	int32 __stdcall ZAux_Direct_GetAxisAddress (ZMC_HANDLE handle, int iaxis, int *piValue)						
Description	Read axis address.						
Input parameters	<table border="1"> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Controller link handle</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </table>	Parameter name	Description	handle	Controller link handle	iaxis	Axis No.
Parameter name	Description						
handle	Controller link handle						
iaxis	Axis No.						
Output parameters	<table border="1"> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>pfValue</td><td>Returned axis address</td></tr> </table>	Parameter name	Description	pfValue	Returned axis address		
Parameter name	Description						
pfValue	Returned axis address						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	Local pulse axis No. remapping						
Details	/						

Instruction 47	ZAux_Direct_SetAxisEnable								
Original Format	int32 __stdcall ZAux_Direct_SetAxisEnable(ZMC_HANDLE handle, int iaxis, int iValue);								
Description	Set axis enable.								
Input parameters	<table border="1"> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link handle</td></tr> <tr> <td>axis</td><td>Slot No.</td></tr> <tr> <td>iValue</td><td>State: 0-off, 1-on</td></tr> </table>	Parameter name	Description	handle	Link handle	axis	Slot No.	iValue	State: 0-off, 1-on
Parameter name	Description								
handle	Link handle								
axis	Slot No.								
iValue	State: 0-off, 1-on								
OP Parameters	One parameter								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	EtherCAT Bus axis enable								

Instruction 48	ZAux_Direct_GetAxisEnable	
Original Format	int32 __stdcall ZAux_Direct_GetAxisEnable (ZMC_HANDLE handle , int iaxis, int * piValue);	
Description	Read EtherCAT axis enable state	
Input parameters	Parameter name	Description
	handle	Controller link handle
	iaxis	Axis No.
Output parameters	Parameter name	Description
	piValue	Returned enable state
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	EtherCAT Bus axis enable	
Details	/	

Instruction 49	ZAux_Direct_SetInvertStep				
Original Format	int32 __stdcall ZAux_Direct_SetInvertStep(ZMC_HANDLE handle, int iaxis, int iValue)				
Description	Set pulse output mode.				
Input parameters	Parameter name	Description			
	handle	Link handle			
	axis	Slot No.			
	iValue	Mode selection, set bit by bit, default is 0			
OP Parameters	/				
Return value	0 = success, not-0 = failure (please check error codes for details).				
Example	Get and Set parameters				
Details	Mode Selection (default = 0)				
	➤ Mode values of lower 8-bit (bit 0 – bit 7):				
	✓ 0-3: pulse direction mode, pulse line + direction line.				
	✓ 4-7: double pulse mode (or CW/CCW), forward pulse line + inverse pulse line				
	✓ 8-9: AB output (customized modes for some controllers)				
	➤ Electric level of each mode:				
	Mode value	Description	Panasonic Setting		Mitsubishi Setting
			Pr0.06	Pr0.07	PA13
	0	Pulse / directional (pulse positive logic) (+)	0	3	××01h
	1	Pulse / directional (pulse negative logic) (+)	/	/	××11h
	2	Pulse / directional (pulse positive logic) (-)	1	3	××01h
	3	Pulse / directional (pulse negative logic) (-)	/	/	××11h
4	Double pulse (pulse negative logic) (+)	/	/	××10h	

	5	Double pulse (pulse negative logic) (-)	/	/	××10h
	6	Double pulse (pulse positive logic) (+)	1	1	××00h (default)
	7	Double pulse (pulse positive logic) (-)	0 default	1 default	××00h (default)
➤ Upper 8-bit (bit 8 – bit 15): the protection time for direction modification, in microseconds: 0-255 ✓ Commonly used modes are 0, 2, 6, 7. ✓ If the mode is not set correctly, the stepper motor may lose a step position when commutating. If you are not sure the stepper motor configuration, you could set the protection time of about 100 microseconds.					

Instruction 50	ZAux_Direct_GetInvertStep				
Original Format	int32 __stdcall ZAux_Direct_GetInvertStep(ZMC_HANDLE handle, int iaxis, int *piValue)				
Description	Read pulse output mode.				
Input parameters	Parameter name	Description			
	handle	Controller link handle			
	iaxis	Axis No.			
Output parameters	Parameter name	Description			
	piValue	Returned current mode.			
Return value	0 = success, not-0 = failure (please check error codes for details).				
Example	Get and Set parameters				
Details	Mode value	Description	Panasonic Setting		Mitsubishi Setting
			Pr0.06	Pr0.07	PA13
	0	Pulse / directional (pulse positive logic) (+)	0	3	××01h
	1	Pulse / directional (pulse negative logic) (+)	/	/	××11h
	2	Pulse / directional (pulse positive logic) (-)	1	3	××01h
	3	Pulse / directional (pulse negative logic) (-)	/	/	××11h
	4	Double pulse (pulse negative logic) (+)	/	/	××10h
	5	Double pulse (pulse negative logic) (-)	/	/	××10h
	6	Double pulse (pulse positive logic) (+)	1	1	××00h (default)
	7	Double pulse (pulse positive logic) (-)	0 default	1 default	××00h (default)
➤ Upper 8-bit (bit 8 – bit 15): the protection time for direction					

	modification, in microseconds: 0-255 ✓ Commonly used modes are 0, 2, 6, 7. ✓ If the mode is not set correctly, the stepper motor may lose a step position when commutating. If you are not sure the stepper motor configuration, you could set the protection time of about 100 microseconds.

Instruction 51	ZAux_Direct_GetMaxSpeed						
Original Format	int32 __stdcall ZAux_Direct_GetMaxSpeed (ZMC_HANDLE handle , int iaxis, int * piValue);						
Description	Read max frequency of pulse output						
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Controller link handle</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </table>	Parameter name	Description	handle	Controller link handle	iaxis	Axis No.
Parameter name	Description						
handle	Controller link handle						
iaxis	Axis No.						
Output parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>piValue</td><td>Returned pulse frequency</td></tr> </table>	Parameter name	Description	piValue	Returned pulse frequency		
Parameter name	Description						
piValue	Returned pulse frequency						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	Get and Set parameters						
Details	● Once the pulse frequency > this value, it will make one limit alarm, and limit the frequency can't exceed MAX_SPEED. ● For the encoder axis, it will start encoder filtering when the setting value is < 500K; and it will cancel encoder filtering when the setting value is > 1M. The default value is 1000000 (the default pulse frequency of old firmware is 500000). ● When the linear motor is used at a high speed, the pulse frequency overlimit will be generated easily, so please set a larger value appropriately.						

Instruction 52	ZAux_Direct_SetMaxSpeed								
Original Format	int32 __stdcall ZAux_Direct_SetMaxSpeed (ZMC_HANDLE handle , int iaxis, int iValue);								
Description	Set max frequency of pulse output								
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link handle</td></tr> <tr> <td>axis</td><td>Slot No.</td></tr> <tr> <td>iValue</td><td>Max pulse frequency that is set.</td></tr> </table>	Parameter name	Description	handle	Link handle	axis	Slot No.	iValue	Max pulse frequency that is set.
Parameter name	Description								
handle	Link handle								
axis	Slot No.								
iValue	Max pulse frequency that is set.								
OP Parameters	/								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	Get and Set parameters								
Details	● Once the pulse frequency > this value, it will make one limit alarm, and limit the frequency can't exceed MAX_SPEED. ● For the encoder axis, it will start encoder filtering when the								

	setting value is < 500K; and it will cancel encoder filtering when the setting value is > 1M. The default value is 1000000 (the default pulse frequency of old firmware is 500000). When the linear motor is used at a high speed, the pulse frequency overlimit will be generated easily, so please set a larger value appropriately.

Instruction 53	ZAux_Direct_GetMtype	
Original Format	int32 __stdcall ZAux_Direct_GetMtype(ZMC_HANDLE handle, int iaxis, int *piValue)	
Description	Read current motion's command type.	
Input parameters	Parameter name	Description
	handle	Controller link handle
	iaxis	Axis No.
Output parameters	Parameter name	Description
	piValue	Returned motion instruction type
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Get and Set parameters	
Details	Please refer to 5.5.2.2 for details.	

Instruction 54	ZAux_Direct_GetNtype	
Original Format	int32 __stdcall ZAux_Direct_GetNtype(ZMC_HANDLE handle, int iaxis, int *piValue)	
Description	Read current motion's next motion command type.	
Input parameters	Parameter name	Description
	handle	Controller link handle
	iaxis	Axis No.
Output parameters	Parameter name	Description
	piValue	Returned motion instruction type
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Get and Set parameters	
Details	Please refer to 5.5.2.2 for details.	

Instruction 55	ZAux_Direct_SetOffpos	
Original Format	int32 __stdcall ZAux_Direct_SetOffpos (ZMC_HANDLE handle , int iaxis, int fValue);	
Description	Set offset.	
Input parameters	Parameter name	Description
	handle	Link handle
	axis	Axis No.
	fValue	Offset value that will be set
OP Parameters	/	

Return value	0 = success, not-0 = failure (please check error codes for details).
Example	Axis Coordinates Configuration
Details	- Make relative offset, to modify all coordinates. - This will not influence on motions that have already run / have entered the buffer. - After modified, OFFPOS will be reset to 0.

Instruction 56	ZAux_Direct_GetOffpos						
Original Format	int32 __stdcall ZAux_Direct_GetOffpos (ZMC_HANDLE handle , int iaxis, int *piValue);						
Description	Read modified offset position.						
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Controller link handle</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </table>	Parameter name	Description	handle	Controller link handle	iaxis	Axis No.
Parameter name	Description						
handle	Controller link handle						
iaxis	Axis No.						
Output parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>fValue</td><td>Obtained offset value</td></tr> </table>	Parameter name	Description	fValue	Obtained offset value		
Parameter name	Description						
fValue	Obtained offset value						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	Axis Coordinates Configuration						
Details	After modified, OFFPOS will be reset to 0.						

Instruction 57	ZAux_Direct_Defpos								
Original Format	int32 __stdcall ZAux_Direct_Defpos(ZMC_HANDLE handle, int iaxis, float pfDpos)								
Description	Define current position as one new absolute position.								
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link handle</td></tr> <tr> <td>axis</td><td>Axis No.</td></tr> <tr> <td>fValue</td><td>Distance coordinates that will be set</td></tr> </table>	Parameter name	Description	handle	Link handle	axis	Axis No.	fValue	Distance coordinates that will be set
Parameter name	Description								
handle	Link handle								
axis	Axis No.								
fValue	Distance coordinates that will be set								
OP Parameters	/								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	Axis Coordinates Configuration								
Details	Not influence on motions that have already run / entered the buffer.								

Instruction 58	ZAux_Direct_GetFe						
Original Format	int32 __stdcall ZAux_Direct_GetFe (ZMC_HANDLE handle , int iaxis, float*pfValue);						
Description	Read following errors.						
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Controller link handle</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </table>	Parameter name	Description	handle	Controller link handle	iaxis	Axis No.
Parameter name	Description						
handle	Controller link handle						
iaxis	Axis No.						
Output parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>fValue</td><td>Obtained offset value</td></tr> </table>	Parameter name	Description	fValue	Obtained offset value		
Parameter name	Description						
fValue	Obtained offset value						

Return value	0 = success, not-0 = failure (please check error codes for details).
Example	Axis Coordinates Configuration
Details	Follow-up error = controller DPOS – encoder MPOS

Instruction 59	ZAux_Direct_GetFeLimit	
Original Format	int32 __stdcall ZAux_Direct_GetFeLimit (ZMC_HANDLE handle, int iaxis, float*pfValue);	
Description	Read allowable max following error (requires closed-loop firmware, reserved function).	
Input parameters	Parameter name	Description
	handle	Controller link handle
	iaxis	Axis No.
Output parameters	Parameter name	Description
	fValue	Max allowable follow-up error values
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	float fValue; ZAux_Direct_GetFeLimit(handle, 0, &fValue); //read the max allowable following error of axis 0. handle: controller link handle	
Details	/	

Instruction 60	ZAux_Direct_SetFeLimit	
Original Format	int32 __stdcall ZAux_Direct_SetFeLimit (ZMC_HANDLE handle , int iaxis, float fValue);	
Description	Set allowable max following error (requires closed-loop firmware, reserved function).	
Input parameters	Parameter name	Description
	handle	Link mark
	axis	Axis No.
	fValue	Max allowable follow-up error values
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	float fValue=100; ZAux_Direct_SetFeLimit(handle, 0, fValue); //set the maximum allowable following error of axis 0 as 100 handle: controller continuous handle	
Details	/	

Instruction 61	ZAux_Direct_GetFeRange	
Original Format	int32 __stdcall ZAux_Direct_GetFeRange (ZMC_HANDLE handle , int iaxis, int*pfValue);	
Description	Read the following error when in alarm (requires closed-loop	

	firmware, reserved function).	
Input parameters	Parameter name	Description
	handle	Controller link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	fValue	Follow-up error values when in alarm.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Int iValue; ZAux_Direct_GetFeRanget(handle, 0, & iValue); //read follow-up error when axis 0 alarms. handle: controller continuous handle	
Details	/	

Instruction 62	ZAux_Direct_SetFeRange	
Original Format	int32 __stdcall ZAux_Direct_SetFeRange (ZMC_HANDLE handle , int iaxis, float fValue)	
Description	Set the following error when in alarm (requires closed-loop firmware, reserved function).	
Input parameters	Parameter name	Description
	handle	Link mark
	axis	Axis No.
	fValue	Follow-up error values when in alarm
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Int iValue=100; ZAux_Direct_SetFeRanget(handle, 0, iValue); //set the maximum allowable following error of axis 0. handle: controller link handle	
Details	/	

Instruction 63	ZAux_Direct_GetAllAxisPara	
Original Format	int32 __stdcall ZAux_Direct_GetAllAxisPara(ZMC_HANDLE handle, const char *sParam,int imaxaxis,float *pfValue)	
Description	Read multi-axis' axis parameters at once time.	
Input parameters	Parameter name	Description
	handle	Link mark
	sParam	Character string name by RTBasic syntax
	imaxaxis	Axis numbers
Output parameters	Parameter name	Description
	fValue	Returned parameter values
Return value	0 = success, not-0 = failure (please check error codes for details).	

Example	float fAtype[8]; ZAux_Direct_GetAllAxisPara(handle,"ATYPE",8,fAtype); //get the axis type of axis 0-7
Details	/

Instruction 64	ZAux_Direct_GetAllAxisInfo										
Original Format	int32 __stdcall ZAux_Direct_GetAllAxisInfo(ZMC_HANDLE handle,int imaxaxis,int *IdleStatus,float *DposStatus,float *MposStatus, int *AxisStatus)										
Description	Read basic axis status of multiple axes. The value is displayed in decimal, but the state is judged in binary.										
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>imaxaxis</td><td>Axis numbers</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	imaxaxis	Axis numbers				
Parameter name	Description										
handle	Link mark										
imaxaxis	Axis numbers										
Output parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>IdleStatus</td><td>Axis running state</td></tr> <tr> <td>DposStatus</td><td>Axis command coordinates</td></tr> <tr> <td>MposStatus</td><td>Axis measurement coordinates</td></tr> <tr> <td>AxisStatus</td><td>Axis state</td></tr> </tbody> </table>	Parameter name	Description	IdleStatus	Axis running state	DposStatus	Axis command coordinates	MposStatus	Axis measurement coordinates	AxisStatus	Axis state
Parameter name	Description										
IdleStatus	Axis running state										
DposStatus	Axis command coordinates										
MposStatus	Axis measurement coordinates										
AxisStatus	Axis state										
Return value	0 = success, not-0 = failure (please check error codes for details).										
Example	int ildle[8]; Float iDpos[8]; Float iMpos[8]; int iStatus[8]; ZAux_Direct_GetAllAxisPara(handle,,8,ildle,iDpos,iMpos,iStatus); //get the axis type of axis 0-7										
Details	Axis state, bit by bit, please refer to 5.1.2.1 for details.										

Instruction 65	ZAux_Direct_SetParam										
Original Format	int32 __stdcall ZAux_Direct_SetParam(ZMC_HANDLE handle,const char *sParam,int iaxis, float fset)										
Description	This is parameter modification function, "sParam", filling in parameter name.										
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>sParam</td><td>Character string name by RTBasic syntax</td></tr> <tr> <td>axis</td><td>Axis No.</td></tr> <tr> <td>fset</td><td>Parameter values</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	sParam	Character string name by RTBasic syntax	axis	Axis No.	fset	Parameter values
Parameter name	Description										
handle	Link mark										
sParam	Character string name by RTBasic syntax										
axis	Axis No.										
fset	Parameter values										
OP Parameters	/										
Return value	0 = success, not-0 = failure (please check error codes for details).										
Example	Get basic parameters by character string.										
Details	The upper computer uses the RTBasic instruction to obtain the										

	controller value, which is often used for the unencapsulated Basic instruction of the upper computer. Specifically, this function can be used to replace some commands to rapidly obtain values.

Instruction 66	ZAux_Direct_GetParam								
Original Format	int32 __stdcall ZAux_Direct_GetParam(ZMC_HANDLE handle, const char *sParam, int iaxis, float *pfValue)								
Description	This is parameter reading function, "sParam", filling in parameter name.								
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>sParam</td><td>Character string parameter name in Basic grammar</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	sParam	Character string parameter name in Basic grammar	iaxis	Axis No.
Parameter name	Description								
handle	Link mark								
sParam	Character string parameter name in Basic grammar								
iaxis	Axis No.								
Output parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>pfValue</td><td>Returned parameter value</td></tr> </tbody> </table>	Parameter name	Description	pfValue	Returned parameter value				
Parameter name	Description								
pfValue	Returned parameter value								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	Get basic parameters by character string.								
Details	The upper computer uses the RTBasic instruction to obtain the controller value, which is often used for the unencapsulated Basic instruction of the upper computer. Specifically, this function can be used to replace some commands to rapidly obtain values.								

Instruction 67	ZAux_Direct_GetEndMove						
Original Format	int32 __stdcall ZAux_Direct_GetEndMove(ZMC_HANDLE handle, int iaxis, float *pfValue)						
Description	Read the end position of the current motion, the unit is units.						
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.
Parameter name	Description						
handle	Link mark						
iaxis	Axis No.						
Output parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>pfValue</td><td>Returned parameter value</td></tr> </tbody> </table>	Parameter name	Description	pfValue	Returned parameter value		
Parameter name	Description						
pfValue	Returned parameter value						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	Float fValue; ZAux_Direct_GetEndMove(handle, 0, &fValue); //get the end position of axis 0 current motion						
Details	For non-fixed distance movements such as VMOVE and DATUM, ENDMOVE is not accurate, or it changes all the time.						

Instruction 68	ZAux_Direct_GetEndMoveBuffer
Original Format	int32 __stdcall ZAux_Direct_GetEndMoveBuffer(ZMC_HANDLE handle, int iaxis, float *pfValue)

Description	Read the end position of current motion and motion in buffer which can be used to convert (absolute & relative), the unit is units.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	pfValue	Return to end target positions of current motion and motions in buffer.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Float fValue; ZAux_Direct_GetEndMoveBuffer (handle,0,&fValue); //get the end position of axis 0 motions in buffer.	
Details	- For non-fixed distance motions, such as, VMOVE, DATUM, ENDMOVE_BUFFER is not accurate, or it changes all the time. - After the loop coordinate instruction REP_OPTION is used, ENDMOVE_BUFFER decreases in turn according to the REP_DIST value in REP_OPTION mode, the minimal precision is REP_DIST (mode 1) / 2 times of REP_DIST (mode 0)	

Instruction 69	ZAux_Direct_GetLoaded	
Original Format	int32 __stdcall ZAux_Direct_GetLoaded (ZMC_HANDLE handle , int iaxis,int *pfValue);	
Description	Read whether there is motion buffer except the current motion.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	pfValue	Returned value: -1: there is no remain motion buffer 0: there is remain motion buffer
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Int iValue; ZAux_Direct_GetLoaded (handle,0,&iValue); //get whether axis 0 has motion buffer or not.	
Details	This instruction cannot judge whether the axis is stopped, please use IDLE to judge the stopped state.	

Instruction 70	ZAux_Direct_GetMovesBuffered	
Original Format	int32 __stdcall ZAux_Direct_GetMoveBuffered (ZMC_HANDLE handle , int iaxis, int * piValue);	
Description	Read the number of motions buffered currently	

Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	pfValue	Buffer numbers.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	<pre>Int iValue; ZAux_Direct_GetMoveBuffered(handle,0,&iValue); //get the number of motions that has been buffered of axis 0</pre>	
Details	Do not judge whether there is remain buffer through "total buffer - MOVES_BUFFERED". Because special motion commands may occupy multiple buffer spaces. Instead, the result of REMAIN_BUFFER is more accurate.	

Instruction 71	ZAux_Direct_GetMoveCurmark	
Original Format	int32 __stdcall ZAux_Direct_GetMoveCurmark (ZMC_HANDLE handle , int iaxis, int * piValue);	
Description	Read MOVE_MARK mark No. of now running motion.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	piValue	Current Mark mark No.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	<pre>Int iValue; ZAux_Direct_GetMoveCurmark(handle,0,&iValue); //get the Mark No. of the motion that is running for axis 0</pre>	
Details	/	

Instruction 72	ZAux_Direct_SetMoveMark	
Original Format	int32 __stdcall ZAux_Direct_SetMoveMark (ZMC_HANDLE handle , int iaxis, int iValue);	
Description	Set the MARK No. of the next motion command. And it will be written into the motion buffer together. Each motion command is called, MOVE_MARK will be + 1 automatically.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
	piValue	MARK value that is set.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	

Example	ZAux_Direct_GetMoveCurmark(handle,0,1); //set next command's Mark as 1 of axis 0
Details	/

Instruction 73	ZAux_Direct_GetRemain						
Original Format	int32 __stdcall ZAux_Direct_GetRemain(ZMC_HANDLE handle,int iaxis, float *pfValue)						
Description	Read remain distance of current motion, the unit is units.						
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.
Parameter name	Description						
handle	Link mark						
iaxis	Axis No.						
Output parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>piValue</td><td>Returned remain distance</td></tr> </tbody> </table>	Parameter name	Description	piValue	Returned remain distance		
Parameter name	Description						
piValue	Returned remain distance						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	Float fValue; ZAux_Direct_GetRemain(handle,0,&fValue); //get axis 0 current motion's remain distance						
Details	/						

Instruction 74	ZAux_Direct_GetRemain_LineBuffer						
Original Format	int32 __stdcall ZAux_Direct_GetRemain_LineBuffer (ZMC_HANDLE handle , int iaxis, int *piValue);						
Description	Axis remaining buffers, it is calculated according to linear segment. REMAIN_BUFFER is the unique one that can add AXIS and can be obtained through Zaux_DirectCommand.						
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.
Parameter name	Description						
handle	Link mark						
iaxis	Axis No.						
Output parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>piValue</td><td>Remaining linear buffers</td></tr> </tbody> </table>	Parameter name	Description	piValue	Remaining linear buffers		
Parameter name	Description						
piValue	Remaining linear buffers						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	Int iValue; ZAux_Direct_GetRemain_LineBuffer(handle,0,&iValue); //get axis 0 current remaining linear motion buffers.						
Details	/						

Instruction 75	ZAux_Direct_GetRemain_Buffer
Original Format	int32 __stdcall ZAux_Direct_GetRemain_Buffer(ZMC_HANDLE handle, int iaxis, int *piValue)
Description	Read axis's remaining motion buffers, it is obtained according to the most complex remaining space arc buffers.

Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	piValue	Remaining space arc buffers
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Get remain buffers.	
Details	/	

Instruction 76	ZAux_Direct_GetVectorBuffered	
Original Format	int32 __stdcall ZAux_Direct_GetVectorBuffered(ZMC_HANDLE handle , int iaxis, float *pfValue);	
Description	Get the completed distance of the current motion & buffer motion. For multi-axis interpolation, it gets the vector distance.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	piValue	The distance has already completed
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Float fValue; ZAux_Direct_GetMoveCurmark(handle,0,&fValue); //get the distance that axis 0's current movement and buffer movement have completed.	
Details	For multi-axis interpolation, it is vector distance.	

Instruction 77	ZAux_Direct_MoveOp	
Original Format	int32 __stdcall ZAux_Direct_MoveOp(ZMC_HANDLE handle, int iaxis, int ioutnum, int ivalue);	
Description	Write output into motion buffer.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
	ioutnum	IO No.
	ivalue	IO state
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Example 1: ZAux_Direct_Single_Move(handle, 0, 100); //axis 0 moves 100 in positive diection. ZAux_Direct_MoveOp(handle, 0,1, 1);	

	<pre>//set the state of out1 to 1 after axis 0 moves forward 100</pre> ● Example 2: Continuous Position Latching
Details	When this instruction is in LOAD execution, it does not perform any motion, it only operates the output port (same as OP). ➤ Normal Mode The normal mode is valid in all motion controllers. The error is within one scan period. ➤ High Precision Mode The precision mode is only valid in ZMC4XX series whose version is above 20170421. The error is within 1 microsecond. ● Not valid in all OP ports, only for the OP that supports HW hardware comparison output. ● 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. That is, if there is no span time between continuous MOVE_OP commands, only the first one is valid. ● For some controllers, they can trigger several precision outputs at the same time, see each manual for details. For example, first 8 outputs of ZMC420SCAN all support precision output, and every output can use precision mode synchronously, but 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. ● 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 they are not isolate, and when multi-axis MOVE_OP is enabled, precision outputs maybe not triggered or triggered wrongly. ● MOVE_OP precision function is used for the main axis of BASE (the first axis in BASE), that is, 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.

Instruction 78	ZAux_Direct_MoveOpMulti												
Original Format	int32 __stdcall ZAux_Direct_MoveOpMulti(ZMC_HANDLE handle, int iaxis, int ioutnumfirst, int ioutnumend, int ivalue)												
Description	Write multiple output instructions into motion buffer.												
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>ioutnumfirst</td><td>The first output channel to be operated</td></tr> <tr> <td>ioutnumend</td><td>The last output channel to be operated</td></tr> <tr> <td>ivalue</td><td>IO state</td></tr> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.	ioutnumfirst	The first output channel to be operated	ioutnumend	The last output channel to be operated	ivalue	IO state
Parameter name	Description												
handle	Link mark												
iaxis	Axis No.												
ioutnumfirst	The first output channel to be operated												
ioutnumend	The last output channel to be operated												
ivalue	IO state												
OP Parameters	/												
Return value	0 = success, not-0 = failure (please check error codes for details).												
Example	ZAux_Direct_Single_Move(handle, 0, 100);												

	//axis 0 moves 100 in forward ZAux_Direct_MoveOpMulti((handle, 0,0,8, 1); //after axis 0 moves 100 in forward, set out0-out4 state as 1
Details	/

Instruction 79	ZAux_Direct_MoveOp2												
Original Format	int32 __stdcall ZAux_Direct_MoveOp2(ZMC_HANDLE handle, int iaxis, int ioutnum, int ivalue, int iofftimems)												
Description	Write output instructions into motion buffer. And invert the level after iofftimems time to produce pulse output.												
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>ioutnum</td><td>IO No.</td></tr> <tr> <td>ivalue</td><td>IO state</td></tr> <tr> <td>iofftimems</td><td>invert after iofftimems time to produce pulse output.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.	ioutnum	IO No.	ivalue	IO state	iofftimems	invert after iofftimems time to produce pulse output.
Parameter name	Description												
handle	Link mark												
iaxis	Axis No.												
ioutnum	IO No.												
ivalue	IO state												
iofftimems	invert after iofftimems time to produce pulse output.												
OP Parameters	/												
Return value	0 = success, not-0 = failure (please check error codes for details).												
Example	Fly-shearing												
Details	- When this instruction is in LOAD execution, it does not perform any movement, but only operates the output port. - A single axis only supports one pulse output at a time, and the second MOVE_OP2 instruction will automatically turn off the pulse of the previous instruction.												

Instruction 80	ZAux_Direct_MoveAout												
Original Format	int32 __stdcall ZAux_Direct_MoveAout(ZMC_HANDLE handle, int iaxis, int ioutnum, float fvalue)												
Description	Add analog output instructions in motion buffer.												
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>ioutnum</td><td>AOUT No.</td></tr> <tr> <td>ivalue</td><td>IO state</td></tr> <tr> <td>fvalue</td><td>Output analog values. 12-bit scale value range 0~4095 corresponds to 0~10V voltage. 16-bit scale value range 0~65536 corresponds to 0~10V voltage.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.	ioutnum	AOUT No.	ivalue	IO state	fvalue	Output analog values. 12-bit scale value range 0~4095 corresponds to 0~10V voltage. 16-bit scale value range 0~65536 corresponds to 0~10V voltage.
Parameter name	Description												
handle	Link mark												
iaxis	Axis No.												
ioutnum	AOUT No.												
ivalue	IO state												
fvalue	Output analog values. 12-bit scale value range 0~4095 corresponds to 0~10V voltage. 16-bit scale value range 0~65536 corresponds to 0~10V voltage.												
OP Parameters	/												
Return value	0 = success, not-0 = failure (please check error codes for details).												

Example	<pre> ZAux_Direct_Single_Move(handle, 0, 100); //axis 0 moves 100 in forward ZAux_Direct_MoveAout(handle, 0,1, 2048); //after axis 0 moved 100 forward, set DA1 state to 2048. </pre>
Details	- When this instruction is in LOAD execution, it does not perform any movement, but only modifies the AOUT value. - The type of this instruction is consistent with MOVE_OP.

Instruction 81	ZAux_Direct_MoveDelay								
Original Format	int32 __stdcall ZAux_Direct_MoveDelay(ZMC_HANDLE handle, int iaxis, int itimems)								
Description	Add delay in motion buffer.								
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>itimems</td><td>Delay time MS</td></tr> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.	itimems	Delay time MS
Parameter name	Description								
handle	Link mark								
iaxis	Axis No.								
itimems	Delay time MS								
OP Parameters	/								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	➤ <u>Example 1:</u> <pre> ZAux_Direct_Single_Move(handle, 0, 100); //axis 0 moves 100 in forward ZAux_Direct_MoveDelay(handle, 0,1000); //after the motion, delay 1s, then execute the next buffer. </pre> ➤ <u>Example 2:</u> Continuous Position Latch								
Details	When former motion command ends, the speed will automatically become 0.								

Instruction 82	ZAux_Direct_MoveWait										
Original Format	int32 __stdcall ZAux_Direct_MoveWait(ZMC_HANDLE handle, uint32 base_axis, char *paraname, int inum, int Cmp_mode, float fvalue)										
Description	Add one condition judge in motion buffer, it is valid in firmware above 150802 or XPLC160405 above versions.										
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>base_axis</td><td>Axis No. of motion axis</td></tr> <tr> <td>paraname</td><td>Character string names: DPOS MPOS IN AIN VPSPEED MSPEED MSPEED MODBUS_REG MODBUS_IEEE MODBUS_BIT NVRAM VECT_BUFFERED REMAIN</td></tr> <tr> <td>inum</td><td>Axis parameter No.</td></tr> </table>	Parameter name	Description	handle	Link mark	base_axis	Axis No. of motion axis	paraname	Character string names: DPOS MPOS IN AIN VPSPEED MSPEED MSPEED MODBUS_REG MODBUS_IEEE MODBUS_BIT NVRAM VECT_BUFFERED REMAIN	inum	Axis parameter No.
Parameter name	Description										
handle	Link mark										
base_axis	Axis No. of motion axis										
paraname	Character string names: DPOS MPOS IN AIN VPSPEED MSPEED MSPEED MODBUS_REG MODBUS_IEEE MODBUS_BIT NVRAM VECT_BUFFERED REMAIN										
inum	Axis parameter No.										

	Cmp_mode	Comparison mode: 1: $\geq$. 0: $=$. -1: $\leq$
	Fvalue	Comparison values
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	➤ <u>Example 1:</u> ZAux_Direct_Single_Move(handle, 0, 100); //axis 0 moves 100 in forward ZAux_Direct_MoveWait(handle, 0,"IN",1,0,1); //after axis 0 moves 100 in forward, waiting for IN1 signal, then execute the next buffer. ➤ <u>Example 2:</u> position latch	
Details	● When this instruction is in LOAD execution, it does not perform any movement, it only waits for the condition to be met. ● When former motion command ends, the speed will automatically become 0.	

Instruction 83	ZAux_Direct_MoveTask	
Original Format	int32 __stdcall ZAux_Direct_MoveTask(ZMC_HANDLE handle,uint32 base_axis,uint32 tasknum,char *labelname)	
Description	The motion buffer is added to a TASK task. When the task has been started, an error will be reported, but it will not affect the program execution. This is only valid in some control cards.	
Input parameters	Parameter name	Description
	handle	Link mark
	base_axis	Axis No. of motion axis
	tasknum	Task No.
	lablename	Triggered global SUB function/lable No.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	- When this instruction is in LOAD execution, it does not perform any movement, it only starts the task. - This command type is consistent with MOVE_OP. - When the task has been started, it will report an error in RTSys, but this will not affect program execution.	

Instruction 84	ZAux_Direct_MovePara		
Original Format	int32 __stdcall ZAux_Direct_MovePara(ZMC_HANDLE handle,uint32 base_axis,char *paraname,uint32 iaxis,float fvalue)		
Description	To write parameter settings into the motion buffer, firmware version 20170503 or above is required.		
Input parameters	Parameter name	Description	
	handle	Link mark	

	base_axis	Axis No. of motion axis
	paraname	Parameter name in character string
	iaxis	Axis No. that modifies the parameter
	fvalue	Parameter set value.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	The axis 0 will move 300 in total, when it reaches 100, make the speed as 100 for following remain motion. <pre> ZAux_Direct_SetMerge(handle, 0,1); //open continuous interpolation ZAux_Direct_Single_Move(handle, 0, 100); //axis 0 moves 100 in forward ZAux_Direct_MovePara(handle, 0,"SPEED",0,100); //after the axis 0 moves forward 100, the speed is changed to 100 ZAux_Direct_Single_Move(handle, 0, 200); //axis 0 moves remain 200 in forward </pre>	
Details	- When this instruction is in LOAD execution, it does not perform any movement, it only modifies the parameter. - This command type is consistent with MOVE_OP.	

Instruction 85	ZAux_Direct_MovePwm	
Original Format	int32 __stdcall ZAux_Direct_MovePwm(ZMC_HANDLE handle,uint32 base_axis,uint32 pwm_num,float pwm_duty,float pwm_freq)	
Description	Write PWM output into the motion buffer, firmware version 20170503 or above is required.	
Input parameters	Parameter name	Description
	handle	Link mark
	base_axis	Axis No. of motion axis
	Pwm_num	PWM No.
	Pwm_duty	Duty cycle that is set
	Pwm_freq	The frequency that is set
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	How to use PWM	
Details	- When this instruction is in LOAD execution, it does not perform any movement, it only modifies the parameter. - This command type is consistent with MOVE_OP.	

Instruction 86	ZAux_Direct_MoveSynmove	
Original Format	int32 __stdcall ZAux_Direct_MoveSynmove(ZMC_HANDLE handle,uint32 base_axis, uint32 iaxis, float fdist, uint32 ifsp)	
Description	Write "trigger other axes motions when running" into buffer, current axis waits.	

Input parameters	Parameter name	Description
	handle	Link mark
	base_axis	Axis No. of motion axis
	iaxis	Synchronous axis No.
	fdist	Synchronous motion distance
	ifsp	Whether the synchronous motion is SP motion -- 0: NO, 1: YES.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	The axis 0 will move 300 in total, when it reaches 100, triggering axis 1 to move 100. After axis 1 motion is completed, axis 0 continues to finish its motion. <pre> ZAux_Direct_SetMerge(handle, 0,1); /open continuous interpolation ZAux_Direct_Single_Move(handle, 0, 100); //axis 0 moves 100 in forward ZAux_Direct_MoveSynmove(handle, 0,1,100,0); //after that, start axis 1 motion ZAux_Direct_Single_Move(handle, 0, 200); //axis 0 moves remain 200 in forward </pre>	
Details	- When this instruction is in LOAD execution, it does not perform any movement, it only modifies the parameter. - This command type is consistent with MOVE_OP.	

Instruction 87	ZAux_Direct_MoveASynmove	
Original Format	int32 __stdcall ZAux_Direct_MoveASynmove(ZMC_HANDLE handle, uint32 base_axis, uint32 iaxis, float fdist, uint32 ifsp)	
Description	Write "trigger other axes motions when running" into buffer, the current axis will not stop. It requires 20170503 firmware / above.	
Input parameters	Parameter name	Description
	handle	Link mark
	base_axis	Axis No. of motion axis
	iaxis	Triggered axis No.
	fdist	Triggered motion distance
	ifsp	Whether the trigger motion is SP motion.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	The axis 0 will move 300 in total, when it reaches 100, triggering axis 1 to move 100. Axis 0 also keeps running. <pre> ZAux_Direct_SetMerge(handle, 0,1); /open continuous interpolation ZAux_Direct_Single_Move(handle, 0, 100); //axis 0 moves 100 in forward ZAux_Direct_MoveASynmove(handle, 0,1,100,0); </pre>	

	<pre>//after that, start axis 1 motion ZAux_Direct_Single_Move(handle, 0, 200); //axis 0 moves remain 200 in forward</pre>
Details	- When this instruction is in LOAD execution, it does not perform any movement, it only modifies the parameter. - This command type is consistent with MOVE_OP.

Instruction 88	ZAux_Direct_MoveTable											
Original Format	int32 __stdcall ZAux_Direct_MoveTable(ZMC_HANDLE handle,uint32 base_axis,uint32 table_num,float fvalue)											
Description	Modify Table value in motion buffer, in this way, Table value can be modified while moving.											
Input parameters	<table><tr><td>Parameter name</td><td>Description</td></tr><tr><td>handle</td><td>Link mark</td></tr><tr><td>base_axis</td><td>Axis No. of motion axis</td></tr><tr><td>Table_num</td><td>TABLE No.</td></tr><tr><td>fvalue</td><td>Value that is set</td></tr></table>		Parameter name	Description	handle	Link mark	base_axis	Axis No. of motion axis	Table_num	TABLE No.	fvalue	Value that is set
	Parameter name	Description										
	handle	Link mark										
	base_axis	Axis No. of motion axis										
	Table_num	TABLE No.										
fvalue	Value that is set											
OP Parameters	/											
Return value	0 = success, not-0 = failure (please check error codes for details).											
Example	The axis 0 will move 300 in total, when it reaches 100, triggering “set TABLE (1) as 10” ZAux_Direct_Single_Move(handle, 0, 100); //axis 0 moves 100 in forward ZAux_Direct_MoveTable(handle, 0,1,10); //after that, wait until IN1 has a signal, then execute the next buffer ZAux_Direct_Single_Move(handle, 0,200); //axis 0 moves remain 200 in forward											
Details	- When this instruction is in LOAD execution, it does not perform any movement, it only modifies the parameter. - This command type is consistent with MOVE_OP.											

Instruction 89	ZAux_Direct_MoveCancel	
Original Format	int32 __stdcall ZAux_Direct_MoveCancel(ZMC_HANDLE handle, int32 base_axis, int32 Cancel_Axis,int iMode)	
Description	Cancel other axes motions in motion buffer, in this way, other axes motions can be canceled while moving.	
Input parameters	Parameter name	Description
	handle	Link mark
	base_axis	Axis No. of motion axis
	Cancel_axis	Axis No. of stop axis
	iMode	Stop mode: 0: cancel current motion 1: cancel buffer motion 2: cancel current motion and buffered

	<table border="1"> <tr> <td></td><td>motion 3: interrupt pulse sending immediately</td></tr> </table>		motion 3: interrupt pulse sending immediately
	motion 3: interrupt pulse sending immediately		
OP Parameters	/		
Return value	0 = success, not-0 = failure (please check error codes for details).		
Example	The axis 0 will move 300 in total, when it reaches 100, stop axis 1. <pre>ZAux_Direct_Single_Move(handle, 0, 100); //axis 0 moves 100 in forward ZAux_Direct_MoveCancel(handle, 0,1,10); //stop axis 1 ZAux_Direct_Single_Move(handle, 0,200); //axis 0 moves 200 in forward</pre>		
Details	/		

Instruction 90	ZAux_Direct_MoveModify								
Original Format	int32 __stdcall ZAux_Direct_MoveModify(ZMC_HANDLE handle, int iaxis, float pfDisance)								
Description	Modify the target position of the previous movement, the effect is the same as MOVEABS when there is no movement in front. However, during continuous interpolation, MOVEMODIFY will break velocity continuity.								
Input parameters	<table border="1"> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>pfDistance</td><td>Motion distance</td></tr> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.	pfDistance	Motion distance
Parameter name	Description								
handle	Link mark								
iaxis	Axis No.								
pfDistance	Motion distance								
OP Parameters	/								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	The axis 0 will move 300 at a low speed, after 1s, set the axis 0 motion as 200 from 300. <pre>ZAux_Direct_Single_Move(handle, 0, 300); //axis 0 moves 300 in forward Sleep (1000); ZAux_Direct_MoveModify(handle, 0, 200); //set the axis motion as 200</pre>								
Details	- Same effect as MOVEABS when there is no movement in front, but will not enter the motion buffer - During continuous interpolation, MOVEMODIFY will break velocity continuity. - When MOVEMODIFY is used for multi-axis motion, it maybe not the linear interpolation.								

Instruction 91	ZAux_Direct_MoveLimit
Original Format	int32 __stdcall ZAux_Direct_MoveLimit(ZMC_HANDLE handle, int iaxis,float limitspeed)
Description	Add speed limit at the end position of current motion to decelerate corner compulsively.

Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
	limitspeed	Limited speed
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	ZAux_Direct_SetMerge(handle, 0,1); //open continuous interpolation ZAux_Direct_Single_Move(handle, 0, 200); //axis 0 moves 100 forward ZAux_Direct_MoveLimit(handle, 0,100); //decrease to 100 between 2 motions ZAux_Direct_Single_Move(handle, 0, 200); //axis 0 moves 200 forward	
Details	/	

Instruction 92	ZAux_Direct_Single_Move	
Original Format	int32 __stdcall ZAux_Direct_Single_Move(ZMC_HANDLE handle, int iaxis, float fdistance)	
Description	Single-axis relative motion.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
	fdistance	Distance
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Single-axis Point to Point	
Details	/	

Instruction 93	ZAux_Direct_Single_MoveAbs	
Original Format	int32 __stdcall ZAux_Direct_Single_MoveAbs(ZMC_HANDLE handle, int iaxis, float fdistance)	
Description	Single-axis absolute motion.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
	fdistance	Absolute distance
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Single-axis Point to Point	
Details	/	

Instruction 94	ZAux_Direct_Single_Vmove
Original	int32 __stdcall ZAux_Direct_Single_Vmove(ZMC_HANDLE handle,

Format	int iaxis, int idir)	
Description	Single-axis continuous motion: moving in one direction.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
	idir	Direction: -1: negative, 1-positive
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Single-axis Continuous Motion	
Details	When the previous VMOVE movement does not stop, this VMOVE command will automatically replace it and modify the direction, so no need to cancel the previous VMOVE command.	

Instruction 95	ZAux_Direct_GetJogSpeed	
Original Format	int32 __stdcall ZAux_Direct_GetJogSpeed (ZMC_HANDLE handle , int iaxis, float *pfValue);	
Description	Read Jog speed when in FastJog motion.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	pfValue	Returned JOG speed value
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Fast Jog (IN Control)	
Details	If REV_JOG/FWD_JOG is set, when the corresponding input point is pressed and the current input state is maintained, the motor will move at the speed set by this function. And the motion will stop when the input point is released.	

Instruction 96	ZAux_Direct_SetJogSpeed	
Original Format	int32 __stdcall ZAux_Direct_SetJogSpeed (ZMC_HANDLE handle , int iaxis, float fValue);	
Description	Set Jog speed when in FastJog motion.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
	fValue	Speed value that is set
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Fast Jog (IN Control)	

Details	If REV_JOG/FWD_JOG is set, when the corresponding input point is pressed and the current input state is maintained, the motor will move at the speed set by this function. And the motion will stop when the input point is released.

Instruction 97	ZAux_Direct_GetFastJog						
Original Format	int32 __stdcall ZAux_Direct_GetFastJog (ZMC_HANDLE handle , int iaxis, int*piValue);						
Description	Read axis fast jog speed mapping IN No. When the set value is -1, it doesn't take effect.						
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.
Parameter name	Description						
handle	Link mark						
iaxis	Axis No.						
Output parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>piValue</td><td>Returned JOG input No.</td></tr> </tbody> </table>	Parameter name	Description	piValue	Returned JOG input No.		
Parameter name	Description						
piValue	Returned JOG input No.						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	Fast Jog (IN Control)						
Details	If fast jog input is set, speed is given by "SPEED" parameter. If no, speed is given by "JOGSPEED" parameter.						

Instruction 98	ZAux_Direct_SetFastJog								
Original Format	int32 __stdcall ZAux_Direct_SetFastJog (ZMC_HANDLE handle , int iaxis, int iValue);								
Description	Set axis fast jog speed mapping No. When the set value is -1, it doesn't take effect.								
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>iValue</td><td>Set fast JOG input No.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.	iValue	Set fast JOG input No.
Parameter name	Description								
handle	Link mark								
iaxis	Axis No.								
iValue	Set fast JOG input No.								
OP Parameters	/								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	Fast Jog (IN Control)								
Details	If fast jog input is set, speed is given by "SPEED" parameter. If no, speed is given by "JOGSPEED" parameter.								

Instruction 99	ZAux_Direct_SetFwdJog				
Original Format	int32 __stdcall ZAux_Direct_SetFwdJog (ZMC_HANDLE handle , int iaxis, int iValue);				
Description	Set forward JOG IN No. When the set value is -1, it doesn't take effect.				
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark
Parameter name	Description				
handle	Link mark				

	iaxis	Axis No.
	iValue	Set forward JOG input No.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Fast Jog (IN Control)	
Details	When there is signal, corresponding axis moves forward at the speed of JOGSPEED.	

Instruction 100	ZAux_Direct_GetFwdJog	
Original Format	int32 __stdcall ZAux_Direct_GetFwdJog(ZMC_HANDLE handle, int iaxis, int *piValue);	
Description	Read forward JOG input No. When the set value is -1, it doesn't take effect.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	piValue	Returned forward JOG input No.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Fast Jog (IN Control)	
Details	When there is signal, corresponding axis moves forward at the speed of JOGSPEED.	

Instruction 101	ZAux_Direct_GetRevJog	
Original Format	int32 __stdcall ZAux_Direct_GetRevJog (ZMC_HANDLE handle, int iaxis, int * piValue);	
Description	Read inverse JOG input No. When the set value is -1, it doesn't take effect.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	piValue	Returned inverse JOG input No.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Fast Jog (IN Control)	
Details	- When there is signal, corresponding axis moves inverse at the speed of JOGSPEED. - When REV_JOG and FWD_JOG both have signal inputs, axis moves according to FWD_JOG.	

Instruction 102	ZAux_Direct_SetRevJog								
Original Format	int32 __stdcall ZAux_Direct_SetRevJog (ZMC_HANDLE handle, int iaxis, int iValue);								
Description	Set inverse JOG input No. When the set value is -1, it doesn't take effect.								
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>iValue</td><td>Set inverse JOG input No.</td></tr> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.	iValue	Set inverse JOG input No.
Parameter name	Description								
handle	Link mark								
iaxis	Axis No.								
iValue	Set inverse JOG input No.								
OP Parameters	/								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	Fast Jog (IN Control)								
Details	- When there is signal, corresponding axis moves inverse at the speed of JOGSPEED. - When REV_JOG and FWD_JOG both have signal inputs, axis moves according to FWD_JOG.								

Instruction 103	ZAux_Direct_Single_Datum										
Original Format	int32 __stdcall ZAux_Direct_Single_Datum(ZMC_HANDLE handle, int iaxis, int imode)										
Description	Single-axis homing motion										
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>imode</td><td>Homing mode (please refer to details)</td></tr> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.	imode	Homing mode (please refer to details)		
Parameter name	Description										
handle	Link mark										
iaxis	Axis No.										
imode	Homing mode (please refer to details)										
OP Parameters	/										
Return value	0 = success, not-0 = failure (please check error codes for details).										
Example	Single-axis Homing Motion										
Details	mode: homing mode, +10 means "find inversely after meeting the position limit", and will not stop, for example, 13 = mode 3 + inverse-find, it is used when the origin is in the middle. <table> <tr> <th>Value</th><th>Description</th></tr> <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 inversely 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 inversely at CREEP until away from origin switch. - During finding origin, it will directly stop when meeting limit +. - During creeping, it will directly stop when meeting limit -. </td></tr> </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 inversely 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 inversely at CREEP until away from origin switch. - During finding origin, it will directly stop when meeting limit +. - During creeping, it will directly stop when meeting 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 inversely 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 inversely at CREEP until away from origin switch. - During finding origin, it will directly stop when meeting limit +. - During creeping, it will directly stop when meeting limit -.										

		DPOS value will be reset to 0, at the same time, correct MPOS.
	4	Axis runs inversely at the speed of SPEED, until meeting origin switch, then axis runs forward at CREEP until away from origin switch. - During finding origin, it will directly stop when meeting limit -. - During creeping, it will directly stop when meeting 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 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 inversely at the speed of SPEED, until meeting origin switch, then axis runs forward at 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.

Instruction 104	ZAux_BusCmd_Datum	
Original Format	int32 __stdcall ZAux_BusCmd_Datum (ZMC_HANDLE handle, uint32 iaxis , uint32 homemode);	
Description	Bus driver homing	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
	Homemode	Homing mode, check driver manual
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	- Set the drive's homing mode (6098h), default is 0, which means "use the drive's current homing mode". - It will use SPEED, CREEP, ACCEL, DECEL of the axis, after multiplying by UNITS, it will automatically set the drive's 6099h, 609Ah. Action sequence: 6098 homing mode → 6099 Speed → 609A Acceleration → 6060 Switch current mode	

Instruction 105	ZAux_Direct_UserDatum	
Original Format	int32 __stdcall ZAux_Direct_UserDatum(ZMC_HANDLE handle, int iaxis, int imode,float HighSpeed,float LowSpeed,float DatumOffset);	

Description	Customized secondary homing.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
	imode	Homing mode
	HihSpeed	Homing high-speed
	LowSpeed	Homing low-speed
	Datumoffset	Offset distance of secondary homing
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	Homing mode (it meets position limit to reversely find by default)	
	Value	Description
	31	Process Steps: 1) Axis runs forward at the speed of CREEP until signal Z appeared. 2) It will offset at homing high-speed. 3) Axis runs forward at CREEP again until Z signal appeared. It will inversely find when met the position limit. DPOS value will be reset to 0, at the same time, correct MPOS. Execute Twice.
	32	Process Steps: 1) Axis runs inversely at the speed of CREEP until signal Z appeared. 2) It will offset at homing high-speed. 3) Axis runs inversely at CREEP again until Z signal appeared. It will inversely find when met the position limit. DPOS value will be reset to 0, at the same time, correct MPOS. Execute Twice.
	33	Process Steps: 1) Axis runs forward at SPEED, until meeting origin switch. 2) Axis runs inversely at CREEP until away from origin switch. 3) It will offset at homing high-speed. 4) Axis runs forward at SPEED again, until meeting the origin switch. 5) Axis inversely moves at CREEP until away from the origin switch. ● During finding origin, it will inversely find when meeting limit +. ● During creeping, it will directly stop when meeting limit -. DPOS value will be reset to 0, at the same time, correct MPOS. Execute Twice.
	34	Process Steps: 1) Axis runs inversely at SPEED, until meeting origin switch 2) Axis runs forward at CREEP until away from origin switch. 3) It will offset at homing high-speed. 4) Axis runs inversely at SPEED again, until meeting origin switch, 5) Axis moves forward at CREEP until away from the origin switch. ● During finding origin, it will inversely find when meeting limit -. ● During creeping, it will directly stop when meeting limit +. DPOS value will be reset to 0, at the same time, correct MPOS. Execute Twice.

	35	Process Steps: 1) Axis runs forward at SPEED, until meeting origin switch. 2) Axis runs inversely at CREEP until away from origin switch. 3) Axis keeps moving at CREEP reversely until meeting signal Z. 4) It will offset at homing high-speed. 5) Axis runs forward at SPEED again, until meeting the origin switch. 6) Axis inversely moves at CREEP until away from the origin switch. 7) Axis keeps moving inversely at CREEP until meeting Z signal. It will inversely find when met limit switch. DPOS value will be reset to 0, at the same time, correct MPOS. Execute Twice.
	36	Process Steps: 1) Axis runs inversely at SPEED, until meeting origin switch. 2) Axis runs forward at CREEP until away from origin switch. 3) Axis keeps moving forward at CREEP until meeting signal Z. 4) It will offset at homing high-speed. 5) Axis runs inversely at SPEED again, until meeting origin switch. 6) Axis moves forward at CREEP until away from the origin switch. 7) Axis keeps moving forward at CREEP until meeting Z signal. It will inversely find when met limit switch. DPOS value will be reset to 0, at the same time, correct MPOS. Execute Twice.
	38	Process Steps: 1) Axis runs forward at SPEED until meeting origin switch. 2) It will offset at homing high-speed. 3) Axis runs forward at SPEED again until meeting origin switch. It will inversely find when met the position limit. Execute Twice.
	39	Process Steps: 1) Axis runs inversely at SPEED until meeting origin switch. 2) It will offset at homing high-speed. 3) Axis runs inversely at SPEED again until meeting origin switch. It will inversely find when met the position limit. Execute Twice.

Instruction 106	ZAux_Direct_SetHomeWait	
Original Format	int32 __stdcall ZAux_Direct_SetHomeWait (ZMC_HANDLE handle, int iaxis, int fValue);	
Description	Set homing reverse finding waiting time.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
	fValue	Homing reverse waiting time MS
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Single-axis Homing Motion	
Details	For pulse type servo driver, it needs to wait one certain of time when finding inversely. The controller default value is 2ms.	

Instruction 107	ZAux_Direct_GetHomeWait	
Original Format	int32 __stdcall ZAux_Direct_GetHomeWait (ZMC_HANDLE handle, int iaxis, int *pfValue);	
Description	Read reverse homing waiting time.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	piValue	Obtained homing reverse waiting time
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	<pre>Int iValue; ZAux_Direct_GetHomeWait (handle, 0, &iValue); //read the setting value of the homing and homing waiting time of axis 0</pre>	
Details	For pulse type servo driver, it needs to wait one certain of time when finding inversely. The controller default value is 2ms.	

Instruction 108	ZAux_BusCmd_SetDatumOffpos	
Original Format	int32 __stdcall ZAux_BusCmd_SetDatumOffpos (ZMC_HANDLE handle, int iaxis , float fValue);	
Description	Set homing offset distance	
Input parameters	Parameter name	Description
	handle	Link mark
	axis	Axis No.
	fvalue	Set offset distance
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	<pre>ZAux_BusCmd_SetDatumOffpos(handle, 0, 100); //offset 100 after axis 0 homing completed.</pre>	
Details	It is mainly for bus motors. If it is a pulse motor, please check whether the firmware supports this function.	

Instruction 109	ZAux_BusCmd_GetDatumOffpos	
Original Format	int32 __stdcall ZAux_BusCmd_GetDatumOffpos (ZMC_HANDLE handle, int iaxis , float *fValue);	
Description	Get homing offset distance.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	piValue	Obtained homing offset distance
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	float fValue;	

	ZAux_BusCmd_GetDatumOffpos(handle, 0, &fValue); //read homing offset value of axis 0
Details	It is mainly for bus motors. If it is a pulse motor, please check whether the firmware supports this function.

Instruction 110	ZAux_Direct_SetCreep								
Original Format	int32 __stdcall ZAux_Direct_SetCreep (ZMC_HANDLE handle, int iaxis, int fValue);								
Description	Set homing creep speed, which is used to search for the origin. The unit is units/s.								
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>fValue</td><td>Creep speed that is set</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.	fValue	Creep speed that is set
Parameter name	Description								
handle	Link mark								
iaxis	Axis No.								
fValue	Creep speed that is set								
OP Parameters	/								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	Single-axis Homing Motion								
Details	/								

Instruction 111	ZAux_Direct_GetHomeStatus						
Original Format	int32 __stdcall ZAux_Direct_GetCreep (ZMC_HANDLE handle , uint32 iaxis, uint32 *homestatus);						
Description	Homing completion state.						
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>c</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	iaxis	c
Parameter name	Description						
handle	Link mark						
iaxis	c						
Output parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>homestatue</td><td>Read homing state value: 0-abnormal homing. 1-normal homing.</td></tr> </tbody> </table>	Parameter name	Description	homestatue	Read homing state value: 0-abnormal homing. 1-normal homing.		
Parameter name	Description						
homestatue	Read homing state value: 0-abnormal homing. 1-normal homing.						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	Single-axis Homing Motion						
Details	/						

Instruction 112	ZAux_BusCmd_GetHomeStatus						
Original Format	int32 __stdcall ZAux_BusCmd_GetHomeStatus (ZMC_HANDLE handle, uint32 iaxis , uint32 * homestate);						
Description	Bus driver homing completion state.						
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.
Parameter name	Description						
handle	Link mark						
iaxis	Axis No.						

Output parameters	Parameter name	Description
	homestatue	Read homing state value: 0-abnormal homing. 1-normal homing.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	/	

Instruction 113	ZAux_Direct_SetDatumIn	
Original Format	int32 __stdcall ZAux_Direct_SetDatumIn(ZMC_HANDLE handle, int iaxis, int iValue)	
Description	Set axis mapping origin IN, -1 means cancel.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
	iValue	IO No., when it is -1, setting is cancelled.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Single-axis Homing Motion	
Details	After setting the origin switch:	
	- For ZMC controller, when IN is OFF, it will give one signal. - For ECI controllers, when IN is OFF, it will give one signal. - If you need inverse effect, you can use INVERT_IN.	

Instruction 114	ZAux_Direct_GetDatumIn	
Original Format	int32 __stdcall ZAux_Direct_GetDatumIn(ZMC_HANDLE handle, int iaxis, int *piValue)	
Description	Read axis mapping origin IN.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	piValue	IN No., -1 means no setting.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Single-axis Homing Motion	
Details	After setting the origin switch:	
	- For ZMC controller, when IN is OFF, it will give one signal. - For ECI controllers, when IN is OFF, it will give one signal. - If you need inverse effect, you can use INVERT_IN.	

Instruction 115	ZAux_Direct_SetClutchRate								
Original Format	int32 __stdcall ZAux_Direct_SetClutchRate(ZMC_HANDLE handle, int iaxis, float fValue)								
Description	Speed of "connect", ratio/s, default value is 1000000.								
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>pfValue</td><td>Synchronous connection ratio</td></tr> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.	pfValue	Synchronous connection ratio
Parameter name	Description								
handle	Link mark								
iaxis	Axis No.								
pfValue	Synchronous connection ratio								
OP Parameters	/								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	Robotic arm								
Details	- When it is set to 0, the connection is tracked according to the speed/acceleration parameters of the axis, which is more suitable for handwheel movement (when the speed is not high enough, it may move for a period of time to end). - If the setting value cannot be much larger than the CONNECT connection ratio, the actual connection ratio will be smaller. - It is generally used to define the change time of the connection rate from 0 to the setting rate.								

Instruction 116	ZAux_Direct_GetClutchRate						
Original Format	int32 __stdcall ZAux_Direct_GetClutchRate(ZMC_HANDLE handle, int iaxis, float *pfValue)						
Description	Read connection speed ratio.						
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.
Parameter name	Description						
handle	Link mark						
iaxis	Axis No.						
Output parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>pfValue</td><td>Current axis' synchronous connection ratio</td></tr> </table>	Parameter name	Description	pfValue	Current axis' synchronous connection ratio		
Parameter name	Description						
pfValue	Current axis' synchronous connection ratio						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	/						
Details	- When it is set to 0, the connection is tracked according to the speed/acceleration parameters of the axis, which is more suitable for handwheel movement (when the speed is not high enough, it may move for a period of time to end). - If the setting value cannot be much larger than the CONNECT connection ratio, the actual connection ratio will be smaller. - It is generally used to define the change time of the connection rate from 0 to the setting rate.						

Instruction 117	ZAux_Direct_EncoderRatio
Original	int32 __stdcall ZAux_Direct_EncoderRatio(ZMC_HANDLE handle,

Format	int iaxis,int output_count,int input_count)										
Description	Set encoder input gear ratio, default is (1,1). Set a negative value to switch direction.										
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>Output_count</td><td>Molecular</td></tr> <tr> <td>Input_count</td><td>Denominator</td></tr> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.	Output_count	Molecular	Input_count	Denominator
Parameter name	Description										
handle	Link mark										
iaxis	Axis No.										
Output_count	Molecular										
Input_count	Denominator										
OP Parameters	/										
Return value	0 = success, not-0 = failure (please check error codes for details).										
Example	Encoder gear ratio configuration										
Details	/										

Instruction 118	ZAux_Direct_StepRatio										
Original Format	int32 __stdcall ZAux_Direct_StepRatio(ZMC_HANDLE handle, int iaxis,int output_count,int input_count)										
Description	Set stepper output gear ratio, default (1,1), range is 1-65535.										
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>Output_count</td><td>molecular</td></tr> <tr> <td>Input_count</td><td>denominator</td></tr> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.	Output_count	molecular	Input_count	denominator
Parameter name	Description										
handle	Link mark										
iaxis	Axis No.										
Output_count	molecular										
Input_count	denominator										
OP Parameters	/										
Return value	0 = success, not-0 = failure (please check error codes for details).										
Example	Pulse output electronic gear ratio configuration										
Details	• A negative value can modify the motor direction, but is generally not recommended. • For the pulse motor, you can use INVERT_STEP, but for the bus motor, it is best to modify on the driver. • It is recommended not to modify this parameter casually, the same effect can be achieved by setting the pulse equivalent.										

Instruction 119	ZAux_Direct_Connect								
Original Format	int32 __stdcall ZAux_Direct_Connect(ZMC_HANDLE handle, float ratio,int link_axis, int move_axis)								
Description	Synchronous motion. The electronic gear connects the target position of the current axis to the interpolation vector length of the idring_axis axis through the electronic gear.								
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>ratio</td><td>Ratio, can be + / -, note the ratio is the number of pulses</td></tr> <tr> <td>Link_axis</td><td>Axis No. of connecting axis (main axis)</td></tr> </table>	Parameter name	Description	handle	Link mark	ratio	Ratio, can be + / -, note the ratio is the number of pulses	Link_axis	Axis No. of connecting axis (main axis)
Parameter name	Description								
handle	Link mark								
ratio	Ratio, can be + / -, note the ratio is the number of pulses								
Link_axis	Axis No. of connecting axis (main axis)								

	Move_axis	Axis No. of motion axis
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Synchronous motion	
Details	- The number of pulses is connected, please consider different axes' UNITS. Use CANCEL to cancel the connection. - Suppose the UNIST of the connecting axis 0 is 100, and the UNITS of the connected axis 1 is 10. - Use ZAux_Direct_Connect to connect, the ratio is 1, when the axis 0 moves $s1 = 100$, the axis 1 moves $= s1 * UNITS(0) * ratio / UNITS(1)$, now it moves 1000. - The ratio can be dynamically changed by calling the instruction repeatedly - Often used in handwheels.	

Instruction 120	ZAux_Direct_Connpath	
Original Format	int32 __stdcall ZAux_Direct_Connpath(ZMC_HANDLE handle, float ratio, int link_axis, int move_axis)	
Description	Synchronous motion 2. The electronic gear connects the target position of the current axis to the interpolation vector length of the link_axis axis through the electronic gear.	
Input parameters	Parameter name	Description
	handle	Link mark
	ratio	Ratio, can be + / -, note the ratio is the number of pulses
	Link_axis	Axis No. of connecting axis (main axis)
	Move_axis	Axis No. of motion axis
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Synchronous motion 2	
Details	- It needs to connect interpolation spindle with the interpolation vector length. If the slave axis is connected, the effect is same as CONNECT. - The number of pulses is connected, please consider different axes' UNITS. Use CANCEL to cancel the connection. - The ratio can be dynamically changed by calling the instruction repeatedly	

Instruction 121	ZAux_Direct_GetLinkax	
Original Format	int32 __stdcall ZAux_Direct_GetLinkax (ZMC_HANDLE handle , int iaxis, int *piValue);	
Description	Read current connected reference axis No.	

Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	pfValue	Returned connected reference axis No.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Float fValue; ZAux_Direct_GetLinkax (handle ,0, &fValue); //check axis 0 connection axis axis No.	
Details	Return to the current connect motion's reference axis No. If there is no connection, it will return -1.	

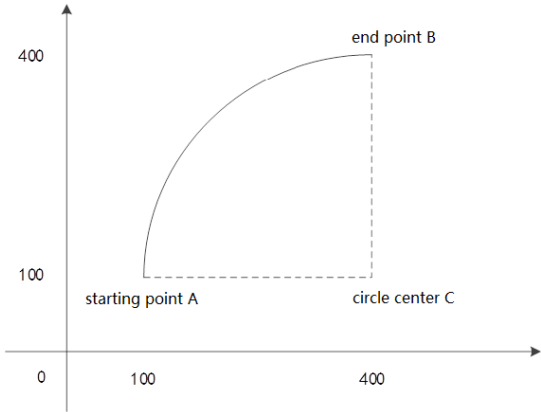
Instruction 122	ZAux_Direct_Move	
Original Format	int32 __stdcall ZAux_Direct_Move(ZMC_HANDLE handle, int imaxaxes, int *piAxislist,float *pfDisancelist)	
Description	Multi-axis relative linear interpolation motion.	
Input parameters	Parameter name	Description
	handle	Link mark
	imaxaxes	Motion axes
	piAxislist	Axis list array
	pfDistancelist	Motion distance list
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Multi-axis linear interpolation	
Details	● Interpolation motion distance $X = \sqrt{X_0^2 + X_1^2 + X_2^2 + \dots + X_n^2}$ ● Motion time $T = X / \text{main axis SPEED}$ ● Only the main axis speed parameter is valid during interpolation (the main axis is the first axis in the axis list), and the motion refers to the parameters of this main axis.	

Instruction 123	ZAux_Direct_MoveSp	
Original Format	int32 __stdcall ZAux_Direct_MoveSp(ZMC_HANDLE handle, int imaxaxes , int *piAxislist, float *pfDisancelist)	
Description	SP motion of relative linear interpolation.	
Input parameters	Parameter name	Description
	handle	Link mark
	imaxaxes	Motion axes
	piAxislist	Axis list array
	pfDistancelist	Motion distance list
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	continuous interpolation SP speed motion control	

Details	- Interpolation motion distance $X = \sqrt{X_0^2 + X_1^2 + X_2^2 + \dots + X_n^2}$ - Motion time $T = X / \text{main axis SPEED}$ - Only the main axis speed parameter is valid during interpolation (the main axis is the first axis in the axis list), and the motion refers to the parameters of this main axis.

Instruction 124	ZAux_Direct_MoveAbs										
Original Format	int32 __stdcall ZAux_Direct_MoveAbs(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float *pfDisancelist)										
Description	Absolute linear interpolation.										
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>imaxaxes</td><td>Motion axes</td></tr> <tr> <td>piAxislist</td><td>Axis list array</td></tr> <tr> <td>pfDistancelist</td><td>Motion distance list</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	imaxaxes	Motion axes	piAxislist	Axis list array	pfDistancelist	Motion distance list
Parameter name	Description										
handle	Link mark										
imaxaxes	Motion axes										
piAxislist	Axis list array										
pfDistancelist	Motion distance list										
OP Parameters	/										
Return value	0 = success, not-0 = failure (please check error codes for details).										
Example	multi-axis linear interpolation										
Details	- Interpolation motion distance $X = \sqrt{X_0^2 + X_1^2 + X_2^2 + \dots + X_n^2}$ - Motion time $T = X / \text{main axis SPEED}$ - Only the main axis speed parameter is valid during interpolation (the main axis is the first axis in the axis list), and the motion refers to the parameters of this main axis.										

Instruction 125	ZAux_Direct_MoveAbsSp										
Original Format	int32 __stdcall ZAux_Direct_MoveAbsSp(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float *pfDisancelist)										
Description	SP motion of absolute linear interpolation.										
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>imaxaxes</td><td>Motion axes</td></tr> <tr> <td>piAxislist</td><td>Axis list array</td></tr> <tr> <td>pfDistancelist</td><td>Motion distance list</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	imaxaxes	Motion axes	piAxislist	Axis list array	pfDistancelist	Motion distance list
Parameter name	Description										
handle	Link mark										
imaxaxes	Motion axes										
piAxislist	Axis list array										
pfDistancelist	Motion distance list										
OP Parameters	/										
Return value	0 = success, not-0 = failure (please check error codes for details).										
Example	/										
Details	- Interpolation motion distance $X = \sqrt{X_0^2 + X_1^2 + X_2^2 + \dots + X_n^2}$ - Motion time $T = X / \text{main axis SPEED}$ - Only the main axis speed parameter is valid during interpolation (the main axis is the first axis in the axis list), and the motion refers to the parameters of this main axis.										

Instruction 126	ZAux_Direct_MoveCirc																		
Original Format	int32 __stdcall ZAux_Direct_MoveCirc(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fend1, float fend2, float fcenter1, float fcenter2, int idirection)																		
Description	Use the circle center to draw the arc. Do circular interpolation of the first axis and the second axis in axis list, it is relative motion method.																		
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>imaxaxes</td><td>Total axes</td></tr> <tr> <td>piAxislist</td><td>Axis list array</td></tr> <tr> <td>fend1</td><td>Motion end coordinate of the first axis</td></tr> <tr> <td>fend2</td><td>Motion end coordinate of the second axis</td></tr> <tr> <td>fcenter1</td><td>The first axis motion circle center, corresponding to starting point.</td></tr> <tr> <td>fcenter2</td><td>The second axis motion circle center, corresponding to starting point.</td></tr> <tr> <td>idirection</td><td>0-anticlockwise, 1-clockwise</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	imaxaxes	Total axes	piAxislist	Axis list array	fend1	Motion end coordinate of the first axis	fend2	Motion end coordinate of the second axis	fcenter1	The first axis motion circle center, corresponding to starting point.	fcenter2	The second axis motion circle center, corresponding to starting point.	idirection	0-anticlockwise, 1-clockwise
Parameter name	Description																		
handle	Link mark																		
imaxaxes	Total axes																		
piAxislist	Axis list array																		
fend1	Motion end coordinate of the first axis																		
fend2	Motion end coordinate of the second axis																		
fcenter1	The first axis motion circle center, corresponding to starting point.																		
fcenter2	The second axis motion circle center, corresponding to starting point.																		
idirection	0-anticlockwise, 1-clockwise																		
OP Parameters	/																		
Return value	0 = success, not-0 = failure (please check error codes for details).																		
Example	Draw the circle by plane center + small-round speed limit																		
Details	1st axis and 2nd axis do circular interpolation in relative motion. When the end point distance is 0, it is one full circle. - When using, circle center and circular end point coordinates that correspond to starting point should be obtained. - Please make sure coordinates are correct, otherwise, actual motion trajectory will be wrong.  If the starting point A coordinate is (100,100), circle center C coordinate is (300,300), end point coordinate is (400,400). Then: the coordinate of circle center C that corresponds to starting point A is (300,0), and the coordinate of end point B that corresponds to starting point A is (300,300).																		

Instruction 127	ZAux_Direct_MoveCircSp																		
Original Format	int32 __stdcall ZAux_Direct_MoveCircSp(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fend1, float fend2, float fcenter1, float fcenter2, int idirection)																		
Description	Draw arc by the circle center (relative) – SP motion command																		
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>imaxaxes</td><td>Total axes</td></tr> <tr> <td>piAxislist</td><td>Axis list array</td></tr> <tr> <td>fend1</td><td>Motion end coordinate of the first axis</td></tr> <tr> <td>fend2</td><td>Motion end coordinate of the second axis</td></tr> <tr> <td>fcenter1</td><td>The first axis motion circle center, corresponding to starting point.</td></tr> <tr> <td>fcenter2</td><td>The second axis motion circle center, corresponding to starting point.</td></tr> <tr> <td>idirection</td><td>0-anticlockwise, 1-clockwise</td></tr> </table>	Parameter name	Description	handle	Link mark	imaxaxes	Total axes	piAxislist	Axis list array	fend1	Motion end coordinate of the first axis	fend2	Motion end coordinate of the second axis	fcenter1	The first axis motion circle center, corresponding to starting point.	fcenter2	The second axis motion circle center, corresponding to starting point.	idirection	0-anticlockwise, 1-clockwise
Parameter name	Description																		
handle	Link mark																		
imaxaxes	Total axes																		
piAxislist	Axis list array																		
fend1	Motion end coordinate of the first axis																		
fend2	Motion end coordinate of the second axis																		
fcenter1	The first axis motion circle center, corresponding to starting point.																		
fcenter2	The second axis motion circle center, corresponding to starting point.																		
idirection	0-anticlockwise, 1-clockwise																		
Output parameters	/																		
Return value	0 = success, not-0 = failure (please check error codes for details).																		
Example	Reference: example of Instruction 126 ZAux_Direct_MoveCirc																		
Details	Reference: example of Instruction 126 ZAux_Direct_MoveCirc																		

Instruction 128	ZAux_Direct_MoveCircAbs																		
Original Format	int32 __stdcall ZAux_Direct_MoveCircAbs(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fend1, float fend2, float fcenter1, float fcenter2, int idirection)																		
Description	Use the circle center to draw the arc. Do circular interpolation of the first axis and the second axis in axis list, it is absolute motion method.																		
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>imaxaxes</td><td>Total axes</td></tr> <tr> <td>piAxislist</td><td>Axis list array</td></tr> <tr> <td>fend1</td><td>Motion end coordinate of the first axis</td></tr> <tr> <td>fend2</td><td>Motion end coordinate of the second axis</td></tr> <tr> <td>fcenter1</td><td>The first axis motion circle center, which is absolute to starting point.</td></tr> <tr> <td>fcenter2</td><td>The second axis motion circle center, which is absolute to starting point.</td></tr> <tr> <td>idirection</td><td>0-anticlockwise, 1-clockwise</td></tr> </table>	Parameter name	Description	handle	Link mark	imaxaxes	Total axes	piAxislist	Axis list array	fend1	Motion end coordinate of the first axis	fend2	Motion end coordinate of the second axis	fcenter1	The first axis motion circle center, which is absolute to starting point.	fcenter2	The second axis motion circle center, which is absolute to starting point.	idirection	0-anticlockwise, 1-clockwise
Parameter name	Description																		
handle	Link mark																		
imaxaxes	Total axes																		
piAxislist	Axis list array																		
fend1	Motion end coordinate of the first axis																		
fend2	Motion end coordinate of the second axis																		
fcenter1	The first axis motion circle center, which is absolute to starting point.																		
fcenter2	The second axis motion circle center, which is absolute to starting point.																		
idirection	0-anticlockwise, 1-clockwise																		
Return value	0 = success, not-0 = failure (please check error codes for details).																		
Example	Reference: example of Instruction 126 ZAux_Direct_MoveCirc																		
Details	Reference: example of Instruction 126 ZAux_Direct_MoveCirc																		

Instruction 129	ZAux_Direct_MoveCircAbsSp																		
Original Format	int32 __stdcall ZAux_Direct_MoveCircAbsSp(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fend1, float fend2, float fcenter1, float fcenter2, int idirection)																		
Description	Draw arc by the circle center (absolute) – SP motion command																		
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>imaxaxes</td><td>Total axes</td></tr> <tr> <td>piAxislist</td><td>Axis list array</td></tr> <tr> <td>fend1</td><td>Motion end coordinate of the first axis</td></tr> <tr> <td>fend2</td><td>Motion end coordinate of the second axis</td></tr> <tr> <td>fcenter1</td><td>The first axis motion circle center, which is absolute to starting point.</td></tr> <tr> <td>fcenter2</td><td>The second axis motion circle center, which is absolute to starting point.</td></tr> <tr> <td>idirection</td><td>0-anticlockwise, 1-clockwise</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	imaxaxes	Total axes	piAxislist	Axis list array	fend1	Motion end coordinate of the first axis	fend2	Motion end coordinate of the second axis	fcenter1	The first axis motion circle center, which is absolute to starting point.	fcenter2	The second axis motion circle center, which is absolute to starting point.	idirection	0-anticlockwise, 1-clockwise
Parameter name	Description																		
handle	Link mark																		
imaxaxes	Total axes																		
piAxislist	Axis list array																		
fend1	Motion end coordinate of the first axis																		
fend2	Motion end coordinate of the second axis																		
fcenter1	The first axis motion circle center, which is absolute to starting point.																		
fcenter2	The second axis motion circle center, which is absolute to starting point.																		
idirection	0-anticlockwise, 1-clockwise																		
OP Parameters	/																		
Return value	0 = success, not-0 = failure (please check error codes for details).																		
Example	Reference: example of Instruction 126 ZAux_Direct_MoveCirc																		
Details	Reference: example of Instruction 126 ZAux_Direct_MoveCirc																		

Instruction 130	ZAux_Direct_MoveCirc2																
Original Format	int32 __stdcall ZAux_Direct_MoveCirc2(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fmid1, float fmid2, float fend1, float fend2)																
Description	Use 3 points to draw the arc. Do circular interpolation of the first axis and the second axis in axis list, it is relative motion method.																
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>imaxaxes</td><td>Total axes</td></tr> <tr> <td>piAxislist</td><td>Axis list array</td></tr> <tr> <td>fmid1</td><td>1st middle point, related to start point</td></tr> <tr> <td>fmid2</td><td>2nd middle point, related to start point</td></tr> <tr> <td>fend1</td><td>1st end point, related to start point</td></tr> <tr> <td>fend2</td><td>2nd end point, related to start point</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	imaxaxes	Total axes	piAxislist	Axis list array	fmid1	1 st middle point, related to start point	fmid2	2 nd middle point, related to start point	fend1	1 st end point, related to start point	fend2	2 nd end point, related to start point
Parameter name	Description																
handle	Link mark																
imaxaxes	Total axes																
piAxislist	Axis list array																
fmid1	1 st middle point, related to start point																
fmid2	2 nd middle point, related to start point																
fend1	1 st end point, related to start point																
fend2	2 nd end point, related to start point																
OP Parameters	/																
Return value	0 = success, not-0 = failure (please check error codes for details).																
Example	Plane 3-point Arc Drawing																
Details	This can't do full-circle interpolation, you can use MOVECIRC (relative) / this command twice continuously to draw one full circle.																

Instruction 131	ZAux_Direct_MoveCirc2Sp																
Original Format	int32 __stdcall ZAux_Direct_MoveCirc2Sp(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fmid1, float fmid2, float fend1, float fend2)																
Description	Use 3 points to draw the arc. Do circular interpolation of the first axis and the second axis in axis list, it is relative motion method. This uses SP motion commands.																
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>imaxaxes</td><td>Total axes</td></tr> <tr> <td>piAxislist</td><td>Axis list array</td></tr> <tr> <td>fmid1</td><td>1st middle point, related to start point</td></tr> <tr> <td>fmid2</td><td>2nd middle point, related to start point</td></tr> <tr> <td>fend1</td><td>1st end point, related to start point</td></tr> <tr> <td>fend2</td><td>2nd end point, related to start point</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	imaxaxes	Total axes	piAxislist	Axis list array	fmid1	1 st middle point, related to start point	fmid2	2 nd middle point, related to start point	fend1	1 st end point, related to start point	fend2	2 nd end point, related to start point
Parameter name	Description																
handle	Link mark																
imaxaxes	Total axes																
piAxislist	Axis list array																
fmid1	1 st middle point, related to start point																
fmid2	2 nd middle point, related to start point																
fend1	1 st end point, related to start point																
fend2	2 nd end point, related to start point																
OP Parameters	/																
Return value	0 = success, not-0 = failure (please check error codes for details).																
Example	Refer to instruction 130 ZAux_Direct_MoveCirc2.																
Details	This can't do full-circle interpolation, you can use MOVECIRC (relative) / this command twice continuously to draw one full circle.																

Instruction 132	ZAux_Direct_MoveCirc2Abs																
Original Format	int32 __stdcall ZAux_Direct_MoveCirc2Abs(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fmid1, float fmid2, float fend1, float fend2)																
Description	Use 3 points to draw the arc. Do circular interpolation of the first axis and the second axis in axis list, it is absolute motion method.																
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>imaxaxes</td><td>Total axes</td></tr> <tr> <td>piAxislist</td><td>Axis list array</td></tr> <tr> <td>fmid1</td><td>1st middle point, related to start point</td></tr> <tr> <td>fmid2</td><td>2nd middle point, related to start point</td></tr> <tr> <td>fend1</td><td>1st end point, related to start point</td></tr> <tr> <td>fend2</td><td>2nd end point, related to start point</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	imaxaxes	Total axes	piAxislist	Axis list array	fmid1	1 st middle point, related to start point	fmid2	2 nd middle point, related to start point	fend1	1 st end point, related to start point	fend2	2 nd end point, related to start point
Parameter name	Description																
handle	Link mark																
imaxaxes	Total axes																
piAxislist	Axis list array																
fmid1	1 st middle point, related to start point																
fmid2	2 nd middle point, related to start point																
fend1	1 st end point, related to start point																
fend2	2 nd end point, related to start point																
OP Parameters	/																
Return value	0 = success, not-0 = failure (please check error codes for details).																
Example	Refer to instruction 130 ZAux_Direct_MoveCirc2.																
Details	This can't do full-circle interpolation, you can use MOVECIRC (relative) / this command twice continuously to draw one full circle.																

Instruction 133	ZAux_Direct_MoveCirc2AbsSp																
Original Format	int32 __stdcall ZAux_Direct_MoveCirc2AbsSp(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fmid1, float fmid2, float fend1, float fend2)																
Description	Use 3 points to draw the arc. Do circular interpolation of the first axis and the second axis in axis list, it is absolute motion method. This uses SP motion commands.																
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>imaxaxes</td><td>Total axes</td></tr> <tr> <td>piAxislist</td><td>Axis list array</td></tr> <tr> <td>fmid1</td><td>1st middle point, related to start point</td></tr> <tr> <td>fmid2</td><td>2nd middle point, related to start point</td></tr> <tr> <td>fend1</td><td>1st end point, related to start point</td></tr> <tr> <td>fend2</td><td>2nd end point, related to start point</td></tr> </table>	Parameter name	Description	handle	Link mark	imaxaxes	Total axes	piAxislist	Axis list array	fmid1	1 st middle point, related to start point	fmid2	2 nd middle point, related to start point	fend1	1 st end point, related to start point	fend2	2 nd end point, related to start point
Parameter name	Description																
handle	Link mark																
imaxaxes	Total axes																
piAxislist	Axis list array																
fmid1	1 st middle point, related to start point																
fmid2	2 nd middle point, related to start point																
fend1	1 st end point, related to start point																
fend2	2 nd end point, related to start point																
OP Parameters	/																
Return value	0 = success, not-0 = failure (please check error codes for details).																
Example	Refer to instruction 130 ZAux_Direct_MoveCirc2.																
Details	This can't do full-circle interpolation, you can use MOVECIRC (relative) / this command twice continuously to draw one full circle.																

Instruction 134	ZAux_Direct_MHelical																						
Original Format	int32 __stdcall ZAux_Direct_MHelical(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fend1, float fend2, float fcenter1, float fcenter2, int idirection, float fDistance3, int imode)																						
Description	Use the circle center to do helical motion. Do circular interpolation of the 1 st and 2 nd in axis list, the third axis does helical motion, which corresponds to starting point.																						
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>imaxaxes</td><td>Total axes</td></tr> <tr> <td>piAxislist</td><td>Axis list array</td></tr> <tr> <td>fend1</td><td>1st axis motion coordinates</td></tr> <tr> <td>fend2</td><td>2nd axis motion coordinates</td></tr> <tr> <td>fcentre1</td><td>1st axis motion circle center, related to starting point.</td></tr> <tr> <td>fcentre2</td><td>2nd axis motion circle center, related to starting point.</td></tr> <tr> <td>idirection</td><td>0-anticlockwise, 1-clockwise</td></tr> <tr> <td>fdistance3</td><td>Motion distance of the 3rd axis</td></tr> <tr> <td>imode</td><td>If the 3rd axis joins in speed calculation? 0(default): YES 1: NO</td></tr> </table>	Parameter name	Description	handle	Link mark	imaxaxes	Total axes	piAxislist	Axis list array	fend1	1 st axis motion coordinates	fend2	2 nd axis motion coordinates	fcentre1	1 st axis motion circle center, related to starting point.	fcentre2	2 nd axis motion circle center, related to starting point.	idirection	0-anticlockwise, 1-clockwise	fdistance3	Motion distance of the 3 rd axis	imode	If the 3 rd axis joins in speed calculation? 0(default): YES 1: NO
Parameter name	Description																						
handle	Link mark																						
imaxaxes	Total axes																						
piAxislist	Axis list array																						
fend1	1 st axis motion coordinates																						
fend2	2 nd axis motion coordinates																						
fcentre1	1 st axis motion circle center, related to starting point.																						
fcentre2	2 nd axis motion circle center, related to starting point.																						
idirection	0-anticlockwise, 1-clockwise																						
fdistance3	Motion distance of the 3 rd axis																						
imode	If the 3 rd axis joins in speed calculation? 0(default): YES 1: NO																						

OP Parameters	/
Return value	0 = success, not-0 = failure (please check error codes for details).
Example	Helical curve
Details	- It can complete a full spiral circle: it is one full circle from the Z direction. - Please make sure the coordinates are correct, otherwise the actual motion trajectory will be wrong.

Instruction 135	ZAux_Direct_MHelicalSp	
Original Format	int32 __stdcall ZAux_Direct_MHelicalSp(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fend1, float fend2, float fcenter1, float fcenter2, int idirection, float fDistance3, int imode)	
Description	Use circle center to do helical motion, use SP motion commands.	
Input parameters	Parameter name	Description
	handle	Link mark
	imaxaxes	Total axes
	piAxislist	Axis list array
	fend1	1 st axis motion coordinates
	fend2	2 nd axis motion coordinates
	fcentre1	1 st axis motion circle center, related to starting point.
	fcentre2	2 nd axis motion circle center, related to starting point.
	idirection	0-anticlockwise, 1-clockwise
	fdistance3	Motion distance of the third axis
	imode	If the 3 rd axis joins in speed calculation? 0(default): YES 1: NO
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	refer to instruction 134 ZAux_Direct_MHelical.	
Details	- It can complete a full spiral circle: it is one full circle from the Z direction. - Please make sure the coordinates are correct, otherwise the actual motion trajectory will be wrong.	

Instruction 136	ZAux_Direct_MHelicalAbs	
Original Format	int32 __stdcall ZAux_Direct_MHelicalAbs(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fend1, float fend2, float fcenter1, float fcenter2, int idirection, float fDistance3, int imode)	
Description	Use the circle center to do helical motion. Do circular interpolation of the 1 st and 2 nd in axis list, the third axis does helical motion. In absolute way.	

Input parameters	Parameter name	Description
	handle	Link mark
	imaxaxes	Total axes
	piAxislist	Axis list array
	fend1	1 st axis motion coordinates
	fend2	2 nd axis motion coordinates
	fcentre1	1 st axis motion circle center, absolute.
	fcentre2	2 nd axis motion circle center, absolute.
	idirection	0-anticlockwise, 1-clockwise
	fdistance3	Motion distance of the 3 rd axis
	imode	If the 3 rd axis joins in speed calculation? 0(default): YES 1: NO
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	refer to instruction 134 ZAux_Direct_MHelical.	
Details	- It can complete a full spiral circle: it is one full circle from the Z direction. - Please make sure the coordinates are correct, otherwise the actual motion trajectory will be wrong.	

Instruction 137	ZAux_Direct_MHelicalAbsSp	
Original Format	int32 __stdcall ZAux_Direct_MHelicalAbsSp(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fend1, float fend2, float fcenter1, float fcenter2, int idirection, float fDistance3, int imode)	
Description	Use the circle center to do helical motion. Do circular interpolation of the 1st and 2nd in axis list, the third axis does helical motion. In absolute way. Use SP motion commands.	
Input parameters	Parameter name	Description
	handle	Link mark
	imaxaxes	Total axes
	piAxislist	Axis list array
	fend1	1 st axis motion coordinates
	fend2	2 nd axis motion coordinates
	fcentre1	1 st axis motion circle center, absolute.
	fcentre2	2 nd axis motion circle center, absolute.
	idirection	0-anticlockwise, 1-clockwise
	fdistance3	Motion distance of the 3 rd axis
	imode	If the 3 rd axis joins in speed calculation? 0(default): YES 1: NO
OP Parameters	/	

Return value	0 = success, not-0 = failure (please check error codes for details).
Example	refer to instruction 134 ZAux_Direct_MHelical.
Details	- It can complete a full spiral circle: it is one full circle from the Z direction. - Please make sure the coordinates are correct, otherwise the actual motion trajectory will be wrong.

Instruction 138	ZAux_Direct_MHelical2Abs																				
Original Format	int32 __stdcall ZAux_Direct_MHelical2Abs(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fmid1, float fmid2, float fend1, float fend2, float fDistance3, int imode)																				
Description	Use 3 points to do helical motion. Do circular interpolation of the 1 st and 2 nd in axis list, the third axis does helical motion. In absolute way.																				
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>imaxaxes</td><td>Total axes</td></tr> <tr> <td>piAxislist</td><td>Axis list array</td></tr> <tr> <td>fmid1</td><td>1st axis middle point position, absolute.</td></tr> <tr> <td>fmid2</td><td>2nd axis middle point position, absolute.</td></tr> <tr> <td>fend1</td><td>1st axis end point position, absolute.</td></tr> <tr> <td>fend2</td><td>2nd axis end point position, absolute.</td></tr> <tr> <td>fdistance3</td><td>Motion distance of the 3rd axis</td></tr> <tr> <td>imode</td><td>If the 3rd axis joins in speed calculation? 0(default): YES 1: NO</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	imaxaxes	Total axes	piAxislist	Axis list array	fmid1	1 st axis middle point position, absolute.	fmid2	2 nd axis middle point position, absolute.	fend1	1 st axis end point position, absolute.	fend2	2 nd axis end point position, absolute.	fdistance3	Motion distance of the 3 rd axis	imode	If the 3 rd axis joins in speed calculation? 0(default): YES 1: NO
Parameter name	Description																				
handle	Link mark																				
imaxaxes	Total axes																				
piAxislist	Axis list array																				
fmid1	1 st axis middle point position, absolute.																				
fmid2	2 nd axis middle point position, absolute.																				
fend1	1 st axis end point position, absolute.																				
fend2	2 nd axis end point position, absolute.																				
fdistance3	Motion distance of the 3 rd axis																				
imode	If the 3 rd axis joins in speed calculation? 0(default): YES 1: NO																				
OP Parameters	/																				
Return value	0 = success, not-0 = failure (please check error codes for details).																				
Example	/																				
Details	It can't move one helical, please use MHELICAL or MHELICALABS to achieve that.																				

Instruction 139	ZAux_Direct_MHelical2AbsSp										
Original Format	int32 __stdcall ZAux_Direct_MHelical2AbsSp(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fmid1, float fmid2, float fend1, float fend2, float fDistance3, int imode)										
Description	Do circular interpolation of the 1 st and 2 nd in axis list, the 3 rd axis does helical motion. In absolute way. Use SP motion commands.										
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>imaxaxes</td><td>Total axes</td></tr> <tr> <td>piAxislist</td><td>Axis list array</td></tr> <tr> <td>fmid1</td><td>1st axis middle point position, absolute.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	imaxaxes	Total axes	piAxislist	Axis list array	fmid1	1 st axis middle point position, absolute.
Parameter name	Description										
handle	Link mark										
imaxaxes	Total axes										
piAxislist	Axis list array										
fmid1	1 st axis middle point position, absolute.										

	fmid2	2 nd axis middle point position, absolute.
	fend1	1 st axis end point position, absolute.
	fend2	2 nd axis end point position, absolute.
	fdistance3	Motion distance of the 3 rd axis
	imode	If the 3 rd axis joins in speed calculation? 0(default): YES 1: NO
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	It can't move one helical, please use MHELICAL or MHELICALABS to achieve that.	

Instruction 140	ZAux_Direct_MHelical2	
Original Format	int32 __stdcall ZAux_Direct_MHelical2(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fmid1, float fmid2, float fend1, float fend2, float fDistance3, int imode)	
Description	Do circular interpolation of the 1 st and 2 nd in axis list, the 3 rd axis does helical motion. In relative way.	
Input parameters	Parameter name	Description
	handle	Link mark
	imaxaxes	Total axes
	piAxislist	Axis list array
	fmid1	1 st axis middle point position, related to starting point.
	fmid2	2 nd axis middle point position, related to starting point.
	fend1	1 st axis end point position, related to starting point.
	fend2	2 nd axis end point position, related to starting point.
	fdistance3	Motion distance of the 3 rd axis
	imode	If the 3 rd axis joins in speed calculation? 0(default): YES 1: NO
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	It can't move one helical, please use MHELICAL or MHELICALABS to achieve that.	

Instruction 141	ZAux_Direct_MHelical2Sp	
Original Format	int32 __stdcall ZAux_Direct_MHelical2Sp(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fmid1, float fmid2, float	

	fend1, float fend2, float fDistance3, int imode)																				
Description	Do circular interpolation of the 1 st and 2 nd in axis list, the 3 rd axis does helical motion. In relative way. Use SP motion commands.																				
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>imaxaxes</td><td>Total axes</td></tr> <tr> <td>piAxislist</td><td>Axis list array</td></tr> <tr> <td>fmid1</td><td>1st axis middle point position, related to starting point.</td></tr> <tr> <td>fmid2</td><td>2nd axis middle point position, related to starting point.</td></tr> <tr> <td>fend1</td><td>1st axis end point position, related to starting point.</td></tr> <tr> <td>fend2</td><td>2nd axis end point position, related to starting point.</td></tr> <tr> <td>fdistance3</td><td>Motion distance of the 3rd axis</td></tr> <tr> <td>imode</td><td>If the 3rd axis joins in speed calculation? 0(default): YES 1: NO</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	imaxaxes	Total axes	piAxislist	Axis list array	fmid1	1 st axis middle point position, related to starting point.	fmid2	2 nd axis middle point position, related to starting point.	fend1	1 st axis end point position, related to starting point.	fend2	2 nd axis end point position, related to starting point.	fdistance3	Motion distance of the 3 rd axis	imode	If the 3 rd axis joins in speed calculation? 0(default): YES 1: NO
Parameter name	Description																				
handle	Link mark																				
imaxaxes	Total axes																				
piAxislist	Axis list array																				
fmid1	1 st axis middle point position, related to starting point.																				
fmid2	2 nd axis middle point position, related to starting point.																				
fend1	1 st axis end point position, related to starting point.																				
fend2	2 nd axis end point position, related to starting point.																				
fdistance3	Motion distance of the 3 rd axis																				
imode	If the 3 rd axis joins in speed calculation? 0(default): YES 1: NO																				
OP Parameters	/																				
Return value	0 = success, not-0 = failure (please check error codes for details).																				
Example	/																				
Details	It can't move one helical, please use MHELICAL or MHELICALABS to achieve that.																				

Instruction 142	ZAux_Direct_MEclipse																
Original Format	int32 __stdcall ZAux_Direct_MEclipse(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fend1, float fend2, float fcenter1, float fcenter2, int idirection, float fADis, float fBDis)																
Description	Do eclipse interpolation, draw the arc by the center. To do ellipse interpolation of 1 st axis and 2 nd axis in axis list. It is relative motion.																
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>imaxaxes</td><td>Total axes</td></tr> <tr> <td>piAxislist</td><td>Axis list array</td></tr> <tr> <td>fend1</td><td>1st axis motion coordinates</td></tr> <tr> <td>fend2</td><td>2nd axis motion coordinates</td></tr> <tr> <td>fcentre1</td><td>1st axis motion circle center, which relates to starting point.</td></tr> <tr> <td>fcentre2</td><td>2nd axis motion circle center, which relates to starting point.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	imaxaxes	Total axes	piAxislist	Axis list array	fend1	1 st axis motion coordinates	fend2	2 nd axis motion coordinates	fcentre1	1 st axis motion circle center, which relates to starting point.	fcentre2	2 nd axis motion circle center, which relates to starting point.
Parameter name	Description																
handle	Link mark																
imaxaxes	Total axes																
piAxislist	Axis list array																
fend1	1 st axis motion coordinates																
fend2	2 nd axis motion coordinates																
fcentre1	1 st axis motion circle center, which relates to starting point.																
fcentre2	2 nd axis motion circle center, which relates to starting point.																

	idirection	0-anticlockwise, 1-clockwise
	fADis	1 st axis' ellipse radius, semi-major or semi-minor are OK.
	fBDis	1 st axis' ellipse radius, semi-major or semi-minor are OK. When A=B, it will automatically do arc / spiral.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	- You can set 3rd to do synchronous helical. - Full ellipse can be drawn. - It can only draw an ellipse whose major axis (short axis) is parallel or perpendicular to X	

Instruction 143	ZAux_Direct_MEclipseSp	
Original Format	int32 __stdcall ZAux_Direct_MEclipseSp(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fend1, float fend2, float fcenter1, float fcenter2, int idirection, float fADis, float fBDis)	
Description	Ellipse interpolation, SP motion command is used.	
Input parameters	Parameter name	Description
	handle	Link mark
	imaxaxes	Total axes
	piAxislist	Axis list array
	fend1	1 st axis motion coordinates
	fend2	2 nd axis motion coordinates
	fcentre1	1 st axis motion circle center, which relates to starting point.
	fcertre2	2 nd axis motion circle center, which relates to starting point.
	idirection	0-anticlockwise, 1-clockwise
	fADis	1 st axis' ellipse radius, semi-major or semi-minor are OK.
	fBDis	1 st axis' ellipse radius, semi-major or semi-minor are OK. When A=B, it will automatically do arc / spiral.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	● You can set 3rd to do synchronous helical. ● Full ellipse can be drawn. ● It can only draw an ellipse whose major axis (short axis) is parallel or perpendicular to X	

Instruction 144	ZAux_Direct_MEclipseAbs																						
Original Format	int32 __stdcall ZAux_Direct_MEclipseAbs(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fend1, float fend2, float fcenter1, float fcenter2, int idirection, float fADis, float fBDis)																						
Description	To do ellipse interpolation of the 1 st axis and 2 nd second axis in axis list. It is absolute motion.																						
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>imaxaxes</td><td>Total axes</td></tr> <tr> <td>piAxislist</td><td>Axis list array</td></tr> <tr> <td>fend1</td><td>1st axis motion coordinates</td></tr> <tr> <td>fend2</td><td>2nd axis motion coordinates</td></tr> <tr> <td>fcentre1</td><td>1st axis motion circle center, absolute</td></tr> <tr> <td>fcentre2</td><td>2nd axis motion circle center, absolute</td></tr> <tr> <td>idirection</td><td>0-anticlockwise, 1-clockwise</td></tr> <tr> <td>fADis</td><td>1st axis' ellipse radius, semi-major or semi-minor are OK.</td></tr> <tr> <td>fBDis</td><td>1st axis' ellipse radius, semi-major or semi-minor are OK. When A=B, it will automatically do arc / spiral.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	imaxaxes	Total axes	piAxislist	Axis list array	fend1	1 st axis motion coordinates	fend2	2 nd axis motion coordinates	fcentre1	1 st axis motion circle center, absolute	fcentre2	2 nd axis motion circle center, absolute	idirection	0-anticlockwise, 1-clockwise	fADis	1 st axis' ellipse radius, semi-major or semi-minor are OK.	fBDis	1 st axis' ellipse radius, semi-major or semi-minor are OK. When A=B, it will automatically do arc / spiral.
Parameter name	Description																						
handle	Link mark																						
imaxaxes	Total axes																						
piAxislist	Axis list array																						
fend1	1 st axis motion coordinates																						
fend2	2 nd axis motion coordinates																						
fcentre1	1 st axis motion circle center, absolute																						
fcentre2	2 nd axis motion circle center, absolute																						
idirection	0-anticlockwise, 1-clockwise																						
fADis	1 st axis' ellipse radius, semi-major or semi-minor are OK.																						
fBDis	1 st axis' ellipse radius, semi-major or semi-minor are OK. When A=B, it will automatically do arc / spiral.																						
OP Parameters	/																						
Return value	0 = success, not-0 = failure (please check error codes for details).																						
Example	/																						
Details	- You can set 3rd to do synchronous helical. - Full ellipse can be drawn. - It can only draw an ellipse whose major axis (short axis) is parallel or perpendicular to X																						

Instruction 145	ZAux_Direct_MEclipseAbsSp																
Original Format	int32 __stdcall ZAux_Direct_MEclipseAbsSp(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fend1, float fend2, float fcenter1, float fcenter2, int idirection, float fADis, float fBDis)																
Description	To do ellipse interpolation of the 1 st axis and 2 nd second axis in axis list. It is absolute motion. SP motion command is used.																
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>imaxaxes</td><td>Total axes</td></tr> <tr> <td>piAxislist</td><td>Axis list array</td></tr> <tr> <td>fend1</td><td>1st axis motion coordinates</td></tr> <tr> <td>fend2</td><td>2nd axis motion coordinates</td></tr> <tr> <td>fcentre1</td><td>1st axis motion circle center, absolute</td></tr> <tr> <td>fcentre2</td><td>2nd axis motion circle center, absolute</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	imaxaxes	Total axes	piAxislist	Axis list array	fend1	1 st axis motion coordinates	fend2	2 nd axis motion coordinates	fcentre1	1 st axis motion circle center, absolute	fcentre2	2 nd axis motion circle center, absolute
Parameter name	Description																
handle	Link mark																
imaxaxes	Total axes																
piAxislist	Axis list array																
fend1	1 st axis motion coordinates																
fend2	2 nd axis motion coordinates																
fcentre1	1 st axis motion circle center, absolute																
fcentre2	2 nd axis motion circle center, absolute																

	idirection	0-anticlockwise, 1-clockwise
	fADis	1 st axis' ellipse radius, semi-major or semi-minor are OK.
	fBDis	1 st axis' ellipse radius, semi-major or semi-minor are OK. When A=B, it will automatically do arc / spiral.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	- You can set 3rd to do synchronous helical. - Full ellipse can be drawn. - It can only draw an ellipse whose major axis (short axis) is parallel or perpendicular to X	

Instruction 146		ZAux_Direct_MEclipseHelical
Original Format	int32 __stdcall ZAux_Direct_MEclipseHelical(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fend1, float fend2, float fcenter1, float fcenter2, int idirection, float fADis, float fBDis, float fDistance3)	
Description	Interpolation of ellipse + helical	
Input parameters	Parameter name	Description
	handle	Link mark
	imaxaxes	Total axes
	piAxislist	Axis list array
	fend1	1 st axis motion coordinates
	fend2	2 nd axis motion coordinates
	fcentre1	1 st axis motion circle center, which relates to starting point.
	fcentre2	2 nd axis motion circle center, which relates to starting point.
	idirection	0-anticlockwise, 1-clockwise
	fADis	1 st axis' ellipse radius, semi-major or semi-minor are OK.
	fBDis	1 st axis' ellipse radius, semi-major or semi-minor are OK. When A=B, it will automatically do arc / spiral.
	fDistance3	3 rd axis motion coordinates
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	Please make sure the coordinates are correct, otherwise the actual motion trajectory will be wrong.	

Instruction 147	ZAux_Direct_MEclipseHelicalSp	
Original Format	int32 __stdcall ZAux_Direct_MEclipseHelicalSp(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fend1, float fend2, float fcenter1, float fcenter2, int idirection, float fADis, float fBDis, float fDistance3)	
Description	Interpolation of ellipse + helical. Use SP motion commands.	
Input parameters	Parameter name	Description
	handle	Link mark
	imaxaxes	Total axes
	piAxislist	Axis list array
	fend1	1 st axis motion coordinates
	fend2	2 nd axis motion coordinates
	fcentre1	1 st axis motion circle center, which relates to starting point.
	fcertre2	2 nd axis motion circle center, which relates to starting point.
	idirection	0-anticlockwise, 1-clockwise
	fADis	1 st axis' ellipse radius, semi-major or semi-minor are OK.
	fBDis	1 st axis' ellipse radius, semi-major or semi-minor are OK. When A=B, it will automatically do arc / spiral.
	fDistance3	3 rd axis motion coordinates
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	Please make sure the coordinates are correct, otherwise the actual motion trajectory will be wrong.	

Instruction 148	ZAux_Direct_MEclipseHelicalAbs	
Original Format	int32 __stdcall ZAux_Direct_MEclipseHelicalAbs(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fend1, float fend2, float fcenter1, float fcenter2, int idirection, float fADis, float fBDis, float fDistance3)	
Description	To do eclipse interpolation of 1 st axis and 2 nd axis in axis list. It is absolute motion. The 3 rd axis does helical motion synchronously (absolute).	
Input parameters	Parameter name	Description
	handle	Link mark
	imaxaxes	Total axes
	piAxislist	Axis list array
	fend1	1 st axis motion coordinates, absolute

	fend2	2 nd axis motion coordinates, absolute
	fcentre1	1 st axis motion circle center, absolute
	fcentre2	2 nd axis motion circle center, absolute
	idirection	0-anticlockwise, 1-clockwise
	fADis	1 st axis' ellipse radius, semi-major or semi-minor are OK.
	fBDis	1 st axis' ellipse radius, semi-major or semi-minor are OK. When A=B, it will automatically do arc / spiral.
	fDistance3	3 rd axis motion coordinates
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	Please make sure the coordinates are correct, otherwise the actual motion trajectory will be wrong.	

Instruction 149	ZAux_Direct_MEclipseHelicalAbsSp	
Original Format	int32 __stdcall ZAux_Direct_MEclipseHelicalAbsSp(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fend1,float fend2, float fcenter1, float fcenter2, int idirection, float fADis, float fBDis,float fDistance3)	
Description	To do eclipse interpolation of 1 st axis and 2 nd axis in axis list. It is absolute motion. The 3 rd axis does helical motion synchronously (absolute). Use SP motion commands.	
Input parameters	Parameter name	Description
	handle	Link mark
	imaxaxes	Total axes
	piAxislist	Axis list array
	fend1	1 st axis motion coordinates, absolute
	fend2	2 nd axis motion coordinates, absolute
	fcentre1	1 st axis motion circle center, absolute
	fcentre2	2 nd axis motion circle center, absolute
	idirection	0-anticlockwise, 1-clockwise
	fADis	1 st axis' ellipse radius, semi-major or semi-minor are OK.
	fBDis	1 st axis' ellipse radius, semi-major or semi-minor are OK. When A=B, it will automatically do arc / spiral.
	fDistance3	3 rd axis motion coordinates
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	

Details	Please make sure the coordinates are correct, otherwise the actual motion trajectory will be wrong.

Instruction 150	ZAux_Direct_MSpherical	
Original Format	int32 __stdcall ZAux_Direct_MSpherical(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fend1, float fend2, float fend3, float fcenter1, float fcenter2, float fcenter3, int imode, float fcenter4, float fcenter5)	
Description	Space circular interpolation motion, in relative way.	
Input parameters	Parameter name	Description
	handle	Link mark
	imaxaxes	Total axes
	piAxislist	Axis list array
	fend1	1 st axis motion distance 1
	fend2	2 nd axis motion distance 1
	fend3	3 rd axis motion distance 1
	fcenter1	1 st axis motion distance 2
	fcenter2	2 nd axis motion distance 2
	fcenter3	3 rd axis motion distance 2
	imode	Specify previous parameters' functions: 0: use 3 points (current point + middle point + end point) to draw the arc. "distance 1": the end point distance. "distance 2": the middle point distance. 1: use current point + circle center + end point to draw the smallest arc. "distance 1": the end point distance. "distance 2": the circle center distance. 2: use 3 points (current point + middle point + end point) to draw the circle. "distance 1": the end point distance. "distance 2": the middle point distance. 3: use current point + circle center + end point to draw the circle, move the smallest arc. "distance 1": the end point distance "distance 2": the circle center distance
	Fcenter4	4 th axis helical distance
	Fcenter5	5 th axis helical distance
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	space arc	

Details	Please make sure the coordinates are correct, otherwise the actual motion trajectory will be wrong.

Instruction 151	ZAux_Direct_MSphericalSp	
Original Format	int32 __stdcall ZAux_Direct_MSphericalSp(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fend1, float fend2, float fend3, float fcenter1, float fcenter2, float fcenter3, int imode, float fcenter4, float fcenter5)	
Description	Space circular interpolation motion, which is relative motion. SP motion command is used.	
Input parameters	Parameter name	Description
	handle	Link mark
	imaxaxes	Total axes
	piAxislist	Axis list array
	fend1	1 st axis motion distance 1
	fend2	2 nd axis motion distance 1
	fend3	3 rd axis motion distance 1
	fcenter1	1 st axis motion distance 2
	fcenter2	2 nd axis motion distance 2
	fcenter3	3 rd axis motion distance 2
	imode	Specify previous parameters' functions: 0: use 3 points (current point + middle point + end point) to draw the arc. "distance 1": the end point distance. "distance 2": the middle point distance. 1: use current point + circle center + end point to draw the smallest arc. "distance 1": the end point distance. "distance 2": the circle center distance. 2: use 3 points (current point + middle point + end point) to draw the circle. "distance 1": the end point distance. "distance 2": the middle point distance. 3: use current point + circle center + end point to draw the circle, move the smallest arc. "distance 1": the end point distance "distance 2": the circle center distance
	Fcenter4	4 th axis helical distance
	Fcenter5	5 th axis helical distance
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	space arc	

Details	Please make sure the coordinates are correct, otherwise the actual motion trajectory will be wrong.

Instruction 152	ZAux_Direct_MoveSpiral																				
Original Format	int32 __stdcall ZAux_Direct_MoveSpiral(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float centre1, float centre2, float circles, float pitch, float distance3, float distance4)																				
Description	Involute circular interpolation motion, which is relative mode. Start from angle 0 when the initial radius is 0 and spreads directly.																				
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>imaxaxes</td><td>Total axes</td></tr> <tr> <td>piAxislist</td><td>Axis list array</td></tr> <tr> <td>centre1</td><td>1st circle center's relative distance</td></tr> <tr> <td>centre2</td><td>2nd circle center's relative distance</td></tr> <tr> <td>circles</td><td>How many circles, it can be not integer. "-" value means anti-clockwise.</td></tr> <tr> <td>pitch</td><td>Diffusion distance / circle, it can be negative value.</td></tr> <tr> <td>distance3</td><td>3rd axis does spiral, this is the 3rd axis related distance, this axis does not participate in the speed calculation.</td></tr> <tr> <td>distance4</td><td>4th axis does spiral, this is the 4th axis related distance, this axis does not participate in the speed calculation.</td></tr> </table>	Parameter name	Description	handle	Link mark	imaxaxes	Total axes	piAxislist	Axis list array	centre1	1 st circle center's relative distance	centre2	2 nd circle center's relative distance	circles	How many circles, it can be not integer. "-" value means anti-clockwise.	pitch	Diffusion distance / circle, it can be negative value.	distance3	3 rd axis does spiral, this is the 3 rd axis related distance, this axis does not participate in the speed calculation.	distance4	4 th axis does spiral, this is the 4 th axis related distance, this axis does not participate in the speed calculation.
Parameter name	Description																				
handle	Link mark																				
imaxaxes	Total axes																				
piAxislist	Axis list array																				
centre1	1 st circle center's relative distance																				
centre2	2 nd circle center's relative distance																				
circles	How many circles, it can be not integer. "-" value means anti-clockwise.																				
pitch	Diffusion distance / circle, it can be negative value.																				
distance3	3 rd axis does spiral, this is the 3 rd axis related distance, this axis does not participate in the speed calculation.																				
distance4	4 th axis does spiral, this is the 4 th axis related distance, this axis does not participate in the speed calculation.																				
OP Parameters	/																				
Return value	0 = success, not-0 = failure (please check error codes for details).																				
Example	/																				
Details	● Use current point and circle center distance to determine the starting radius. ● If it can't be confirmed, starting from angle 0.																				

Instruction 153	ZAux_Direct_MoveSpiralSp										
Original Format	int32 __stdcall ZAux_Direct_MoveSpiralSp(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float centre1, float centre2, float circles, float pitch, float distance3, float distance4)										
Description	Involute circular interpolation movement, in relative mode. Start from angle 0 when the initial radius is 0 and spreads directly.										
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>imaxaxes</td><td>Total axes</td></tr> <tr> <td>piAxislist</td><td>Axis list array</td></tr> <tr> <td>centre1</td><td>1st circle center's relative distance</td></tr> </table>	Parameter name	Description	handle	Link mark	imaxaxes	Total axes	piAxislist	Axis list array	centre1	1 st circle center's relative distance
Parameter name	Description										
handle	Link mark										
imaxaxes	Total axes										
piAxislist	Axis list array										
centre1	1 st circle center's relative distance										

	centre2	2 nd circle center's relative distance
	circles	How many circles, it can be not integer. "-" value means anti-clockwise.
	pitch	Diffusion distance / circle, it can be negative value.
	distance3	3 rd axis does spiral, this is the 3 rd axis related distance, this axis does not participate in the speed calculation.
	distance4	4 th axis does spiral, this is the 4 th axis related distance, this axis does not participate in the speed calculation.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	- Use current point and circle center distance to determine the starting radius. - If it can't be confirmed, starting from angle 0.	

Instruction 154	ZAux_Direct_MoveSmooth	
Original Format	int32 __stdcall ZAux_Direct_MoveSmooth(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float end1, float end2, float end3, float next1, float next2, float next3, float radius)	
Description	Space linear chamfering interpolation: according to next linear motion absolute coordinates, automatically insert arc at the corner.	
Input parameters	Parameter name	Description
	handle	Link mark
	imaxaxes	Total axes
	piAxislist	Axis list array
	end1	1 st axis absolute coordinates
	end2	2 nd axis absolute coordinates
	end3	3 rd axis absolute coordinates
	Next1	1 st axis' next linear absolute coordinates.
	Next2	2 nd axis' next linear absolute coordinates.
	Next3	3 rd axis' next linear absolute coordinates.
	radius	The radius of inserted arc, if it is too larger, it will zoom out automatically.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	- This command is an early development command and has a limit on the number of axes. It is recommended to use the CORNER_MODE command, which has more functions. - According to the absolute coordinates of the next linear	

	motion, the arc is automatically inserted at the corner. After adding the arc, the end point of the motion will be inconsistent with the end point of the straight line. And when the corner is too large, the arc will not be inserted. Also, when the distance is not enough, the radius will be automatically reduced.

Instruction 155	ZAux_Direct_MoveSmoothSp																						
Original Format	int32 __stdcall ZAux_Direct_MoveSmoothSp(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float end1, float end2, float end3, float next1, float next2, float next3, float radius)																						
Description	Space linear chamfering interpolation: according to next linear motion absolute coordinates, automatically insert arc at the corner.																						
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>imaxaxes</td><td>Total axes</td></tr> <tr> <td>piAxislist</td><td>Axis list array</td></tr> <tr> <td>end1</td><td>1st axis absolute coordinates</td></tr> <tr> <td>end2</td><td>2nd axis absolute coordinates</td></tr> <tr> <td>end3</td><td>3rd axis absolute coordinates</td></tr> <tr> <td>Next1</td><td>1st axis' next linear absolute coordinates.</td></tr> <tr> <td>Next2</td><td>2nd axis' next linear absolute coordinates.</td></tr> <tr> <td>Next3</td><td>3rd axis' next linear absolute coordinates.</td></tr> <tr> <td>radius</td><td>The radius of inserted arc, if it is too larger, it will zoom out automatically.</td></tr> </table>	Parameter name	Description	handle	Link mark	imaxaxes	Total axes	piAxislist	Axis list array	end1	1 st axis absolute coordinates	end2	2 nd axis absolute coordinates	end3	3 rd axis absolute coordinates	Next1	1 st axis' next linear absolute coordinates.	Next2	2 nd axis' next linear absolute coordinates.	Next3	3 rd axis' next linear absolute coordinates.	radius	The radius of inserted arc, if it is too larger, it will zoom out automatically.
Parameter name	Description																						
handle	Link mark																						
imaxaxes	Total axes																						
piAxislist	Axis list array																						
end1	1 st axis absolute coordinates																						
end2	2 nd axis absolute coordinates																						
end3	3 rd axis absolute coordinates																						
Next1	1 st axis' next linear absolute coordinates.																						
Next2	2 nd axis' next linear absolute coordinates.																						
Next3	3 rd axis' next linear absolute coordinates.																						
radius	The radius of inserted arc, if it is too larger, it will zoom out automatically.																						
OP Parameters	/																						
Return value	0 = success, not-0 = failure (please check error codes for details).																						
Example	/																						
Details	- This command is an early development command and has a limit on the number of axes. It is recommended to use the CORNER_MODE command, which has more functions. - According to the absolute coordinates of the next linear motion, the arc is automatically inserted at the corner. After adding the arc, the end point of the motion will be inconsistent with the end point of the straight line. And when the corner is too large, the arc will not be inserted. Also, when the distance is not enough, the radius will be automatically reduced.																						

Instruction 156	ZAux_Direct_McirkTurnabs
Original Format	int32 __stdcall ZAux_Direct_McirkTurnabs (ZMC_HANDLE handle , int tablenum, float refpos1, float refpos2, int mode, float end1, float end2, int maxaxes, int *piAxislist, float * pfDistancelist);
Description	Rotary table arc + helical interpolation movement. Valid in controller versions after 20130901.

Input parameters	Parameter name	Description
	handle	Link mark
	tablenum	TABLE No., saving rotary parameters.
	refpos1	Reference point of 1 st axis, absolute.
	refpos2	Reference point of 2 nd axis, absolute.
	mode	1: reference point is before current point. 2: reference point is after current point. 3: reference point is in the middle, using 3-point to draw one circle.
	end1	1 st axis end point, absolute.
	end2	2 nd axis end point, absolute.
	Maxaxes	The number of motion axes
	piAxislist	Axis list
	pfDistancelist	Rotary axis distance list
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	- Rotary function: the working platform rotates on a plane that is parallel to XY, and the rotation forward must be consistent with the XY forward (right-hand rule). - Rotary parameters: they are stored in the TABLE in sequence, which stores axis R No., the pulses of R-axis in one circle, axis X No., axis Y No., the X circle center, and the Y circle center.	

Instruction 157	ZAux_Direct_MoveTurnabs	
Original Format	int32 __stdcall ZAux_Direct_MoveTurnabs (ZMC_HANDLE handle , int tablenum, int imaxaxes, int *piAxislist, float * pfDistancelist);	
Description	Rotary linear interpolation movement. Valid in controller versions after 20130901.	
Input parameters	Parameter name	Description
	handle	Link mark
	tablenum	TABLE No., saving rotary parameters.
	Maxaxes	The number of motion axes
	piAxislist	Axis No. list
	pfDistancelist	Distance list
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	- Rotary function: the working platform rotates on a plane that is parallel to XY, and the rotation forward must be consistent with the XY forward (right-hand rule). - Rotary parameters: they are stored in the TABLE in sequence, which stores axis R No., the pulses of R-axis in one circle, axis	

	X No., axis Y No., the X circle center, and the Y circle center.

Instruction 158	ZAux_Direct_MultiMove												
Original Format	int32 __stdcall ZAux_Direct_MultiMove(ZMC_HANDLE handle,int iMoveLen, int imaxaxes, int *piAxislist, float *pfDisancelist)												
Description	Several relative multi-axis linear interpolation.												
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iMoveLen</td><td>Filled motion length</td></tr> <tr> <td>Maxaxes</td><td>The number of motion axes</td></tr> <tr> <td>piAxislist</td><td>Axis No. list array</td></tr> <tr> <td>pfDistancelist</td><td>Distance list array</td></tr> </table>	Parameter name	Description	handle	Link mark	iMoveLen	Filled motion length	Maxaxes	The number of motion axes	piAxislist	Axis No. list array	pfDistancelist	Distance list array
Parameter name	Description												
handle	Link mark												
iMoveLen	Filled motion length												
Maxaxes	The number of motion axes												
piAxislist	Axis No. list array												
pfDistancelist	Distance list array												
OP Parameters	/												
Return value	0 = success, not-0 = failure (please check error codes for details).												
Example	For example: write 5 groups of motions one time for axis 0, 1, 2, 3. <pre>int axisList[4]={0,1,2,3} float posList[20] //position defined for(int i=0;i<5;i++) { posList[0+i*4]=100; //axis 0 position posList[0+i*4]=200; //axis 1 position posList[0+i*4]=300 //axis 2 position posList[0+i*4]=400 //axis 3 position } //write 5 groups of motions one time for axis 0, 1, 2, 3 ZAux_Direct_MultiMove (handle, 5, 4, axisList, posList);</pre>												
Details	Send multiple multi-axis linear relative motions at one time.												

Instruction 159	ZAux_Direct_MultiMoveAbs												
Original Format	int32 __stdcall ZAux_Direct_MultiMoveAbs(ZMC_HANDLE handle,int iMoveLen, int imaxaxes, int *piAxislist, float *pfDisancelist)												
Description	Several absolute multi-axis linear interpolations.												
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iMoveLen</td><td>Filled motion length</td></tr> <tr> <td>Maxaxes</td><td>The number of motion axes</td></tr> <tr> <td>piAxislist</td><td>Axis No. list array</td></tr> <tr> <td>pfDistancelist</td><td>Distance list array</td></tr> </table>	Parameter name	Description	handle	Link mark	iMoveLen	Filled motion length	Maxaxes	The number of motion axes	piAxislist	Axis No. list array	pfDistancelist	Distance list array
Parameter name	Description												
handle	Link mark												
iMoveLen	Filled motion length												
Maxaxes	The number of motion axes												
piAxislist	Axis No. list array												
pfDistancelist	Distance list array												
OP Parameters	/												
Return value	0 = success, not-0 = failure (please check error codes for details).												

Example	For example: write 5 groups of motions one time for axis 0, 1, 2, 3. <pre>int axisList[4]={0,1,2,3} float posList[20] //position defined for(int i=0;i<5;i++) { posList[0+i*4]=100+10*i; //axis 0 position posList[1+i*4]=200+10*i; //axis 1 position posList[2+i*4]=300+10*i; //axis 2 position posList[3+i*4]=400+10*i; //axis 3 position } //write 5 groups of motions one time for axis 0, 1, 2, 3 ZAux_Direct_MultiMoveAbs (handle, 5, 4, axisList, posList);</pre>
Details	Send multiple multi-axis linear absolute motions in one time.

Instruction 160	ZAux_Direct_MultiLineN														
Original Format	int32 __stdcall ZAux_Direct_MultiLineN(ZMC_HANDLE handle,int imode,int iMoveLen, int imaxaxes, int *piAxislist, float *pfDisancelist,int *iReBuffLen)														
Description	Several relative multi-axis linear interpolations.														
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>imode</td><td>bit0- bifabs absolute mode bit1- bifsp customized speed mode bit2- bifresume resume motion bit3- bifmovescan switch line motion to galvanometer motion</td></tr> <tr> <td>iMoveLen</td><td>Filled motion length</td></tr> <tr> <td>Maxaxes</td><td>The number of motion axes</td></tr> <tr> <td>piAxislist</td><td>Axis No. list array</td></tr> <tr> <td>pfDistancelist</td><td>Distance list array</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	imode	bit0- bifabs absolute mode bit1- bifsp customized speed mode bit2- bifresume resume motion bit3- bifmovescan switch line motion to galvanometer motion	iMoveLen	Filled motion length	Maxaxes	The number of motion axes	piAxislist	Axis No. list array	pfDistancelist	Distance list array
Parameter name	Description														
handle	Link mark														
imode	bit0- bifabs absolute mode bit1- bifsp customized speed mode bit2- bifresume resume motion bit3- bifmovescan switch line motion to galvanometer motion														
iMoveLen	Filled motion length														
Maxaxes	The number of motion axes														
piAxislist	Axis No. list array														
pfDistancelist	Distance list array														
OP Parameters	/														
Return value	0 = success, not-0 = failure (please check error codes for details).														
Example	For example: write 5 groups of motions one time for axis 0, 1, 2, 3. <pre>int axisList[4]={0,1,2,3} float posList[20] //position defined for(int i=0;i<5;i++) { posList[0+i*4]=100+10*i; //axis 0 position posList[1+i*4]=200+10*i; //axis 1 position posList[2+i*4]=300+10*i; //axis 2 position posList[3+i*4]=400+10*i; //axis 3 position } //write 5 groups of motions one time for axis 0, 1, 2, 3 ZAux_Direct_MultiLineN (handle, 1, 5, 4, axisList, posList);</pre>														

Details	Send continuous small segment interpolation directly.

Instruction 161	ZAux_Direct_SetFhspeed								
Original Format	int32 __stdcall ZAux_Direct_SetFhspeed (ZMC_HANDLE handle , int iaxis, float fValue);								
Description	Set axis holding speed.								
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>fValue</td><td>Speed value that is set</td></tr> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.	fValue	Speed value that is set
Parameter name	Description								
handle	Link mark								
iaxis	Axis No.								
fValue	Speed value that is set								
OP Parameters	/								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	Axis Holding Input								
Details	It can move at this speed only when the corresponding input is in the hold state.								

Instruction 162	ZAux_Direct_GetFhspeed								
Original Format	int32 __stdcall ZAux_Direct_GetFhspeed (ZMC_HANDLE handle , int iaxis, float *pfValue);								
Description	Read axis holding speed.								
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>fValue</td><td>Speed value that is set</td></tr> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.	fValue	Speed value that is set
Parameter name	Description								
handle	Link mark								
iaxis	Axis No.								
fValue	Speed value that is set								
Output parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>fValue</td><td>Speed value that is read.</td></tr> </table>	Parameter name	Description	fValue	Speed value that is read.				
Parameter name	Description								
fValue	Speed value that is read.								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	Axis Holding Input								
Details	It can move at this speed only when the corresponding input is in the hold state.								

Instruction 163	ZAux_Direct_SetEndMoveSpeed								
Original Format	int32 __stdcall ZAux_Direct_SetEndMoveSpeed(ZMC_HANDLE handle, int iaxis, float fValue);								
Description	Set end speed of SP motion (customized speed).								
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>fValue</td><td>Speed value that is set</td></tr> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.	fValue	Speed value that is set
Parameter name	Description								
handle	Link mark								
iaxis	Axis No.								
fValue	Speed value that is set								
OP Parameters	/								
Return value	0 = success, not-0 = failure (please check error codes for details).								

Example	continuous interpolation SP motion
Details	- It will take effect only when SP motion command is used. - Please set a larger value when not in use. - The controller default value is 1000.

Instruction 164	ZAux_Direct_GetEndMoveSpeed	
Original Format	int32 __stdcall ZAux_Direct_GetEndMoveSpeed(ZMC_HANDLE handle, int iaxis, float *fValue);	
Description	Read end speed of SP motion (customized speed).	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
OP Parameters	Parameter name	Description
	fValue	Speed value that is read.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	continuous interpolation SP motion	
Details	- It will take effect only when SP motion command is used. - Please set a larger value when not in use. - The controller default value is 1000.	

Instruction 165	ZAux_Direct_GetForceSpeed	
Original Format	int32 __stdcall ZAux_Direct_GetForceSpeed (ZMC_HANDLE handle , int iaxis, float *pfValue);	
Description	Read SP running speed.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	fValue	SP speed value that is read.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Axis Holding Input	
Details	- This parameter will be carried into the motion buffer. - When FORCE_SPEED is > SPEED, SPEED will also take effect to limit the max speed (after 140716, SPEED does not work). - If FORCE_SPEED has reduced to the corresponding speed when entering this section, please set STARTMOVE_SPEED.	

Instruction 166	ZAux_Direct_SetForceSpeed	
Original Format	int32 __stdcall ZAux_Direct_SetForceSpeed (ZMC_HANDLE handle , int iaxis, float fValue);	
Description	Set SP running speed.	

Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
	fValue	Speed value that is set
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Axis Holding Input	
Details	- This parameter will be carried into the motion buffer. - When FORCE_SPEED is > SPEED, SPEED will also take effect to limit the max speed (after 140716, SPEED does not work). - If FORCE_SPEED has reduced to the corresponding speed when entering this section, please set STARTMOVE_SPEED.	

Instruction 167	ZAux_Direct_SetStartMoveSpeed	
Original Format	int32 __stdcall ZAux_Direct_SetStartMoveSpeed(ZMC_HANDLE handle, int iaxis, float fValue);	
Description	Set the starting speed of SP motion, this parameter will be brought into the motion buffer. The unit is units/s. For more details, please refer to "STARTMOVE_SPEED" in RTBasic Programming Manual .	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
	fValue	Speed value that is set
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	continuous interpolation SP motion	
Details	- It will take effect only when there is SP motion command. - Please set a larger value when not in use. - The controller default value is 1000.	

Instruction 168	ZAux_Direct_GetStartMoveSpeed	
Original Format	int32 __stdcall ZAux_Direct_GetStartMoveSpeed(ZMC_HANDLE handle, int iaxis, float * pfValue);	
Description	Read starting speed of SP motion.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	fValue	SP starting speed value that is read.
Return value	0 = success, not-0 = failure (please check error codes for details).	

Example	continuous interpolation SP motion
Details	- It will take effect only when there is SP motion command. - Please set a larger value when not in use. - The controller default value is 1000.

Instruction 169	ZAux_Direct_GetFholdIn						
Original Format	int32 __stdcall ZAux_Direct_GetFholdIn (ZMC_HANDLE handle , int iaxis, into*piValue);						
Description	Read holding input.						
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.
Parameter name	Description						
handle	Link mark						
iaxis	Axis No.						
Output parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>fValue</td><td>Return to FholdIN input No.</td></tr> </tbody> </table>	Parameter name	Description	fValue	Return to FholdIN input No.		
Parameter name	Description						
fValue	Return to FholdIN input No.						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	Please see instruction 162 ZAux_Direct_SetFhspeed						
Details	- If there is an input signal, the speed of the moving axis is set by FHSPEED. When the input is canceled, the motion running speed will return to the program speed. - When IN is OFF, which means there is signal input. If you need the opposite effect, INVERT_IN can do that, inverting the electric level (except for ECI series controllers). - This parameter only suits to speed control mode (command with SP suffix). Therefore, it does not work if the motion is not velocity controlled (CAMBOX, CONNECT, MOVELINK).						

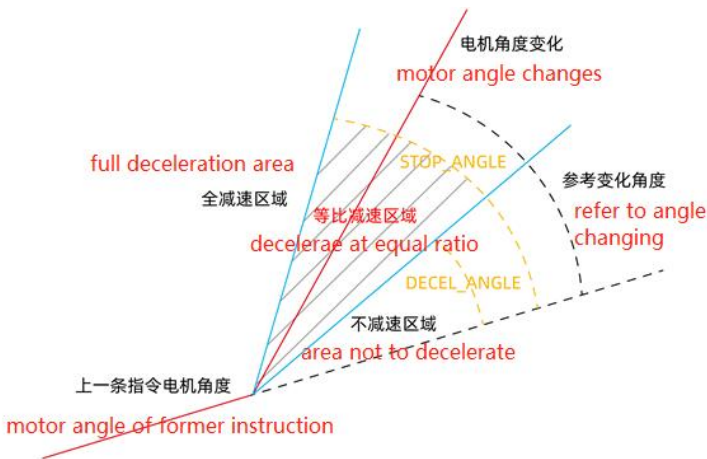
Instruction 170	ZAux_Direct_SetFholdIn								
Original Format	int32 __stdcall ZAux_Direct_SetFholdIn (ZMC_HANDLE handle, int iaxis, int iValue);								
Description	Set hold input No., -1 means invalid.								
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>iValue</td><td>Set Input No.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.	iValue	Set Input No.
Parameter name	Description								
handle	Link mark								
iaxis	Axis No.								
iValue	Set Input No.								
OP Parameters	/								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	Axis Holding Input								
Details	- If there is an input signal, the speed of the moving axis is set by FHSPEED. When the input is canceled, the motion running speed will return to the program speed. - When IN is OFF, which means there is signal input. If you need the opposite effect, INVERT_IN can do that, inverting the								

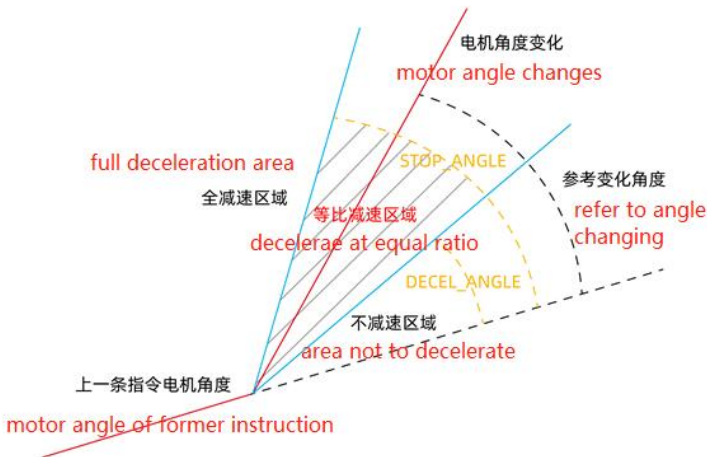
	electric level (except for ECI series controllers). This parameter only suits to speed control mode (command with SP suffix). Therefore, it does not work if the motion is not velocity controlled (CAMBOX, CONNECT, MOVELINK).

Instruction 171	ZAux_Direct_SetCornerMode			
Original Format	int32 __stdcall ZAux_Direct_SetCornerMode(ZMC_HANDLE handle, int iaxis, int iValue);			
Description	Set corner mode.			
Input parameters	Parameter name	Description		
	handle	Link mark		
	iaxis	Axis No.		
	iValue	Mode setting:		
		Bit	Value	Description
		0	1	Reserved
		1	2	Auto corner deceleration
2		4	Reserved	
3	8	Auto small-circle speed limit		
OP Parameters	/			
Return value	0 = success, not-0 = failure (please check error codes for details).			
Example	Corner deceleration			
Details	Model: different bits represent different meanings. Please 6.5.2.2 CORNER_MODE instruction parameter description :			

Instruction 172	ZAux_Direct_GetCornerMode				
Original Format	int32 __stdcall ZAux_Direct_GetCornerMode (ZMC_HANDLE handle, int iaxis, int * piValue);				
Description	Read corner deceleration mode.				
Input parameters	Parameter name		Description		
	handle		Link mark		
	iaxis		Axis No.		
Output parameters	Parameter name		Description		
	iValue		Mode setting:		
			Bit	Value	Description
			0	1	Reserved
			1	2	Auto corner deceleration
			2	4	Reserved
3	8	Auto small-circle speed limit			

Return value	0 = success, not-0 = failure (please check error codes for details).
Example	Corner deceleration
Details	Model: different bits represent different meanings. Please 6.5.2.2 CORNER_MODE instruction parameter description :

Instruction 173	ZAux_Direct_SetDecelAngle	
Original Format	int32 __stdcall ZAux_Direct_SetDecelAngle(ZMC_HANDLE handle, int iaxis, float fValue);	
Description	Set corner deceleration angle, starting deceleration angle, the unit is in radian.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
	fValue	Corner deceleration angle that is set.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Corner deceleration	
Details	- The reference speed of the deceleration corner is based on the FORCE_SPEED, and please set one reasonable FORCE_SPEED. Angle conversion radian formular: $\text{angle value} * (\text{PI}/180)$ - The deceleration angle: the change value of the reference angle of the motor that is relative to the previous movement. <i>[this angle value is not actual trajectory's angle, but the angle converted to the motor transformation. That is, this angle value is only for reference].</i>	
	 - The absolute value is taken when the next interpolation motion track is below. - When a straight line is connected with an arc, the angle is calculated according to the tangent direction of the arc. - Used together with STOP_ANGLE, it will decelerate when the actual motion angle is between DECEL_ANGLE (upper limit) and STOP_ANGLE (lower limit).	

Instruction 174	ZAux_Direct_GetDecelAngle	
Original Format	int32 __stdcall ZAux_Direct_GetDecelAngle (ZMC_HANDLE handle, int iaxis, float *pfValue);	
Description	Read corner starting decelerating angle, the unit is radian.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	fValue	Corner deceleration angle that is read.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Corner deceleration	
Details	● The reference speed of the deceleration corner is based on the FORCE_SPEED, and please set one reasonable FORCE_SPEED. ● Angle conversion radian formular: angle value * (PI/180) ● The deceleration angle: the change value of the reference angle of the motor that is relative to the previous movement. <i>[this angle value is not actual trajectory's angle, but the angle converted to the motor transformation. That is, this angle value is only for reference].</i>	
	 ● The absolute value is taken when the next interpolation motion track is below. ● When a straight line is connected with an arc, the angle is calculated according to the tangent direction of the arc. ● Used together with STOP_ANGLE, it will decelerate when the actual motion angle is between DECEL_ANGLE (upper limit) and STOP_ANGLE (lower limit).	

Instruction 175	ZAux_Direct_SetFullSpRadius
Original Format	int32 __stdcall ZAux_Direct_SetFullSpRadius(ZMC_HANDLE handle, int iaxis, float fValue);
Description	Set small circle speed limit's radius.

Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
	fValue	Small-round radius
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Draw the circle with plane circle center + small-round speed limit	
Details	- When the radius is > the radius of the small circle speed limit, the speed is specified by the program. - When the radius is < the radius of the small circle speed limit, the controller will reduce the speed proportionally.	

Instruction 176	ZAux_Direct_GetFullSpRadius	
Original Format	int32 __stdcall ZAux_Direct_GetFullSpRadius (ZMC_HANDLE handle , int iaxis, float *pfValue);	
Description	Read the min radius of small-circle speed limit.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	fValue	Returned speed limit radius
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Draw the circle with plane circle center + small-round speed limit	
Details	- When the radius is > the radius of the small circle speed limit, the speed is specified by the program. - When the radius is < the radius of the small circle speed limit, the controller will reduce the speed proportionally.	

Instruction 177	ZAux_Direct_GetInterpFactor	
Original Format	int32 __stdcall ZAux_Direct_GetInterpFactor (ZMC_HANDLE handle , int iaxis, int *piValue);	
Description	When reading interpolation, whether the axis participates in the speed calculation, the default (1) means YES. This parameter only works for the 3rd of the line and the spiral.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	fValue	Returned speed calculation mode
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	This parameter only works for the 3rd axis of any axis in line or	

	the spiral. Please restore it in time after motion, otherwise it will cause incorrect following motions. - When some axes do not participate in the speed calculation, firstly calculate axis' sub-velocity and total movement time according to the interpolation motion axes, and then divide the movement distance of the non-participating axes by the total time in turn to obtain corresponding speed of each axis that is excluded in calculation. - Don't set all real motion axes to 0, which will cause the actual speed to be infinite.

Instruction 178	ZAux_Direct_SetInterpFactor								
Original Format	int32 __stdcall ZAux_Direct_SetInterpFactor (ZMC_HANDLE handle , int iaxis, int iValue);								
Description	Set whether the axis participates in the speed calculation, the default (1) means YES. This parameter only works for the 3rd of the line and the spiral.								
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>piValue</td><td>Mode: 0-not to participate, 1-participate</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.	piValue	Mode: 0-not to participate, 1-participate
Parameter name	Description								
handle	Link mark								
iaxis	Axis No.								
piValue	Mode: 0-not to participate, 1-participate								
OP Parameters	/								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	/								
Details	- This parameter only works for the 3rd axis of any axis in line or the spiral. Please restore it in time after motion, otherwise it will cause incorrect following motions. - When some axes do not participate in the speed calculation, firstly calculate axis' sub-velocity and total movement time according to the interpolation motion axes, and then divide the movement distance of the non-participating axes by the total time in turn to obtain corresponding speed of each axis that is excluded in calculation. - Don't set all real motion axes to 0, which will cause the actual speed to be infinite.								

Instruction 179	ZAux_Direct_SetMerge						
Original Format	int32 __stdcall ZAux_Direct_SetMerge(ZMC_HANDLE handle, int iaxis, int iValue)						
Description	Set continuous interpolation.						
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.
Parameter name	Description						
handle	Link mark						
iaxis	Axis No.						

	piValue	0-OFF, 1-ON
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Corner deceleration	
Details	When MERGE = ON (continuous interpolation is opened), but if there is still deceleration between interpolations, possible reasons are as follows: ● Maybe MERGE is not set successfully. ● The controller doesn't support continuous interpolation. ● The CORNER_MODE corner deceleration is set. ● The motion command with SP is used, and ENDMOVE_SPEED and STARTMOVE_SPEED are set. At this time, the speed is determined by these two commands. ● The main axis is switched between multiple interpolations, and the master axis speed parameter is changed. ● The MOVE_DELAY delay command is added them, even if the delay is written to 0, it will also cause deceleration.	

Instruction 180	ZAux_Direct_GetMerge	
Original Format	int32 __stdcall ZAux_Direct_GetMerge(ZMC_HANDLE handle, int iaxis, int *piValue)	
Description	Read continuous interpolation switch state.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	piValue	Continuous interpolation state: 0-OFF, 1-ON
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Corner deceleration	
Details	When MERGE = ON (continuous interpolation is opened), but if there is still deceleration between interpolations, possible reasons are as follows: ● Maybe MERGE is not set successfully. ● The controller doesn't support continuous interpolation. ● The CORNER_MODE corner deceleration is set. ● The motion command with SP is used, and ENDMOVE_SPEED and STARTMOVE_SPEED are set. At this time, the speed is determined by these two commands. ● The main axis is switched between multiple interpolations, and the master axis speed parameter is changed. ● The MOVE_DELAY delay command is added them, even if the delay is written to 0, it will also cause deceleration.	

Instruction 181	ZAux_Direct_SetStopAngle								
Original Format	int32 __stdcall ZAux_Direct_SetStopAngle(ZMC_HANDLE handle, int iaxis, float pfValue);								
Description	Set stop deceleration angle.								
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>pfValue</td><td>Stop deceleration angle</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.	pfValue	Stop deceleration angle
Parameter name	Description								
handle	Link mark								
iaxis	Axis No.								
pfValue	Stop deceleration angle								
OP Parameters	/								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	Corner deceleration								
Details	- The reference speed of the deceleration corner is based on the FORCE_SPEED, and please set one reasonable FORCE_SPEED. Angle conversion radian formular: $\text{angle value} * (\text{PI}/180)$ - The deceleration angle: the change value of the reference angle of the motor that is relative to the previous movement. <i>[this angle value is not actual trajectory's angle, but the angle converted to the motor transformation. That is, this angle value is only for reference].</i> - The absolute value is taken when the next interpolation motion track is below. - When a straight line is connected with an arc, the angle is calculated according to the tangent direction of the arc. - Used together with STOP_ANGLE, it will decelerate when the actual motion angle is between DECEL_ANGLE (upper limit) and STOP_ANGLE (lower limit).								

Instruction 182	ZAux_Direct_GetStopAngle
Original Format	int32 __stdcall ZAux_Direct_GetStopAngle (ZMC_HANDLE handle , int iaxis, float * pfValue);
Description	Set Corner stop deceleration angle, the unit is radian. The reference speed of the deceleration corner is based on the FORCE_SPEED speed, and a reasonable FORCE_SPEED must be set.

Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	pfValue	Returned corner stop angle
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Corner deceleration	
Details	- The reference speed of the deceleration corner is based on the FORCE_SPEED, and please set one reasonable FORCE_SPEED. Angle conversion radian formular: $\text{angle value} * (\text{PI}/180)$ - The deceleration angle: the change value of the reference angle of the motor that is relative to the previous movement. [this angle value is not actual trajectory's angle, but the angle converted to the motor transformation. That is, this angle value is only for reference].	
	- The absolute value is taken when the next interpolation motion track is below. - When a straight line is connected with an arc, the angle is calculated according to the tangent direction of the arc. - Used together with STOP_ANGLE, it will decelerate when the actual motion angle is between DECEL_ANGLE (upper limit) and STOP_ANGLE (lower limit).	

Instruction 183	ZAux_Direct_SetZsmooth	
Original Format	int32 __stdcall ZAux_Direct_SetZsmooth (ZMC_HANDLE handle, int iaxis, float fValue);	
Description	Set deceleration chamfering radius.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
	pfValue	The value is set.
OP Parameters	/	

Return value	0 = success, not-0 = failure (please check error codes for details).
Example	Chamfer
Details	- Automatically calculate the actual corner radius according to corner angle. Limits to 50% when path is exceeded. - At 90°, the corner radius is the set value.

Instruction 184	ZAux_Direct_GetZsmooth	
Original Format	int32 __stdcall ZAux_Direct_GetZsmooth(ZMC_HANDLE handle, int iaxis, float *pfValue);	
Description	Read global chamfer radius.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	pfValue	Returned chamfer radius
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Chamfer	
Details	- Automatically calculate the actual corner radius according to corner angle. Limits to 50% when path is exceeded. - At 90°, the corner radius is the set value.	

Instruction 185	ZAux_Direct_MovePause	
Original Format	int32 __stdcall ZAux_Direct_MovePause(ZMC_HANDLE handle,int iaxis,int imode)	
Description	Pause BASE axes. It is only valid during single-axis or multi-axis interpolation motion. Pause all axes when multi-axis is linked.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
	imode	Pause Modes: ➤ 0: pause current motion. ➤ 1: when current motion is completed and it is ready to execute next motion, pause. ➤ 2: when current motion is completed and it is ready to execute next motion, also when the two commands' MARK are different, pause (this mode also can pause the action that is achieved by several commands when the entire action is completed). ➤ 3: force to pause, it can also enter the

		pause state in IDLE mode, it is valid in firmware version after 20170513.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	axis pause & resume	
Details	- It is only valid during single-axis or multi-axis interpolation motion, and it will pause together when multi-axis is linked. - It can use AXISSTATUS to check whether there is a pause. - When the axis has paused or is not in motion, there is alarm output if this command is called, but it will not affect the program running. - For some motions, they don't support pause, such as VMOVE, synchronous motion instructions, etc.	

Instruction 186	ZAux_Direct_Rapidstop	
Original Format	int32 __stdcall ZAux_Direct_Rapidstop(ZMC_HANDLE handle, int imode)	
Description	Stop all axes (in axis list) immediately, and the interpolation movement also can be stopped if the axis does the interpolation.	
Input parameters	Parameter name	Description
	handle	Link mark
	imode	Modes
		➤ 0: cancel the current motion ➤ 1: cancel motions in buffer ➤ 2: cancel current & buffer motions ➤ 3: immediately interrupt pulse sending
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	- Mode 0~2 deceleration follows the maximum value of FASTDEC and DECEL. - To call absolute position motion after RAPIDSTOP, it must wait for the stop to complete by WAIT IDLE.	

Instruction 187	ZAux_Direct_MoveResume	
Original Format	int32 __stdcall ZAux_Direct_MoveResume (ZMC_HANDLE handle , int iaxis);	
Description	Cancel motion pause.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
OP Parameters	/	

Return value	0 = success, not-0 = failure (please check error codes for details).
Example	axis pause & resume
Details	- When BASE axis / axes paused, it will keep moving from where it is paused. - AXISSTATUS can be used to check whether there is a pause.

Instruction 188	ZAux_Direct_CancelAxisList	
Original Format	int32 __stdcall ZAux_Direct_CancelAxisList(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, int imode)	
Description	Stop all axes (in axis list) immediately, and the interpolation movement also can be stopped if the axis does the interpolation.	
Input parameters	Parameter name	Description
	handle	Link mark
	imaxaxes	Total motion axes
	piAxislist	Axis list array
	imode	Modes ➤ 0: cancel the current motion ➤ 1: cancel motions in buffer ➤ 2: cancel current & buffer motions ➤ 3: stop immediately
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	Mode0~2 deceleration follows the maximum value of FASTDEC and DECEL.	

Instruction 189	ZAux_Direct_Single_Cancel	
Original Format	int32 __stdcall ZAux_Direct_Single_Cancel (ZMC_HANDLE handle, int iaxis, int imode);	
Description	Stop single-axis motion.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
	imode	Modes ➤ 0: cancel the current motion (default) ➤ 1: cancel motions in buffer ➤ 2: cancel current & buffer motions ➤ 3: immediately interrupt pulse sending
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	single-axis continuous motion	
Details	If the specified axis is in the interpolation motion axis list, regardless of the main axis or other axes, stop the	

	interpolation motion of the axis group ● MODE0~2 deceleration follows the maximum value of FASTDEC and DECEL. ● To call one absolute position motion after CANCEL, you must wait for the stop to complete.

Instruction 190	ZAux_Direct_GetRepDist						
Original Format	int32 __stdcall ZAux_Direct_GetRepDist (ZMC_HANDLE handle , int iaxis, float *pfValue);						
Description	Read DPOS and MPOS auto-cycle coordinates according to REP_OPTION.						
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.
Parameter name	Description						
handle	Link mark						
iaxis	Axis No.						
Output parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>pfValue</td><td>Returned cyclic coordinate values</td></tr> </table>	Parameter name	Description	pfValue	Returned cyclic coordinate values		
Parameter name	Description						
pfValue	Returned cyclic coordinate values						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	Coordinate Cycle						
Details	/						

Instruction 191	ZAux_Direct_SetRepDist						
Original Format	int32 __stdcall ZAux_Direct_SetRepDist (ZMC_HANDLE handle , int iaxis, float fValue);						
Description	Automatically cycle DPOS and MPOS coordinates according to REP_OPTION configuration.						
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.
Parameter name	Description						
handle	Link mark						
iaxis	Axis No.						
Output parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>pfValue</td><td>Configured cyclic coordinate values</td></tr> </table>	Parameter name	Description	pfValue	Configured cyclic coordinate values		
Parameter name	Description						
pfValue	Configured cyclic coordinate values						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	Coordinate Cycle						
Details	/						

Instruction 192	ZAux_Direct_GetRepOption
Original Format	int32 __stdcall ZAux_Direct_GetRepOption (ZMC_HANDLE handle , int iaxis, int *piValue);
Description	Read coordinates repeated configurations.

Input parameters	Parameter name	Description		
	handle	Link mark		
	iaxis	Axis No.		
Output parameters	Parameter name	Description		
	pfValue	Returned mode:		
		Bit	Value	Description
		0	1	0: loop range: - REP_DIST ~ + REP_DIST 1: loop range: 0 ~ + REP_DIST
		1	2	1: prohibit the repeated movement of CAMBOX and MOVELINK, and reset as 0 after prohibition takes effect
		2	4	Reserved
4	16	1: do not use REP_DIST 0: use REP_DIST		
Return value	0 = success, not-0 = failure (please check error codes for details).			
Example	Coordinate Cycle			
Details	- It can be used to limit the coordinate cycle range of the cam's main axis to realize the continuity of multiple cam curves. - When absolute mode is used, if the target position is within coordinate cycle range, it can move correctly (relative motion is not affected)			

Instruction 193	ZAux_Direct_SetRepOption			
Original Format	int32 __stdcall ZAux_Direct_SetRepOption (ZMC_HANDLE handle , int iaxis, int iValue);			
Description	Set coordinate loop mode.			
Input parameters	Parameter name	Description		
	handle	Link mark		
	iaxis	Axis No.		
	iValue	Different bits -- different meanings:		
		Bit	Value	Description
		0	1	0: loop range: - REP_DIST ~ + REP_DIST 1: loop range: 0 ~ + REP_DIST

		1	2	1: prohibit the repeated movement of CAMBOX and MOVELINK, and trset to 0 after prohibition takes effect
		2	4	Reserved
		4	16	1: do not use REP_DIST 0: use REP_DIST
OP Parameters	/			
Return value	0 = success, not-0 = failure (please check error codes for details).			
Example	Coordinate Cycle			
Details	- It can be used to limit the coordinate cycle range of the cam's main axis to realize the continuity of multiple cam curves. - When absolute mode is used, if the target position is within coordinate cycle range, it can move correctly (relative motion is not affected)			

Instruction 194	ZAux_Direct_Pitchset																		
Original Format	int32 __stdcall ZAux_Direct_Pitchset(ZMC_HANDLE handle,int iaxis,int iEnable,float StartPos,uint32 maxpoint,float DisOne ,uint32 TablNum,float * pfDisancelist)																		
Description	Set axis' pitch compensation, it is invalid for expanded axes.																		
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>iEnable</td><td>If opens compensation: 1: ON, 0: OFF.</td></tr> <tr> <td>StartPos</td><td>Staring compensation MPOS position</td></tr> <tr> <td>maxpoint</td><td>Total points in compensation area</td></tr> <tr> <td>DisOne</td><td>Each compensation point's gap</td></tr> <tr> <td>TablNum</td><td>TABLE starting address where saves the compensation coordinates.</td></tr> <tr> <td>pfDisancelist</td><td>Area compensation value list</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.	iEnable	If opens compensation: 1: ON, 0: OFF.	StartPos	Staring compensation MPOS position	maxpoint	Total points in compensation area	DisOne	Each compensation point's gap	TablNum	TABLE starting address where saves the compensation coordinates.	pfDisancelist	Area compensation value list
Parameter name	Description																		
handle	Link mark																		
iaxis	Axis No.																		
iEnable	If opens compensation: 1: ON, 0: OFF.																		
StartPos	Staring compensation MPOS position																		
maxpoint	Total points in compensation area																		
DisOne	Each compensation point's gap																		
TablNum	TABLE starting address where saves the compensation coordinates.																		
pfDisancelist	Area compensation value list																		
OP Parameters	/																		
Return value	0 = success, not-0 = failure (please check error codes for details).																		
Example	Pitch Compensation																		
Details	Compensation pulses numbers of each point are stored in TABLE.																		

Instruction 195	ZAux_Direct_Pitchset2
Original Format	int32 __stdcall ZAux_Direct_Pitchset2(ZMC_HANDLE handle,int iaxis,int iEnable,float StartPos,uint32 maxpoint,float DisOne ,uint32 TablNum,float * pfDisancelist ,uint32 RevTablNum,float *

	RevpfDisancelist)																						
Description	Set axis' pitch compensation 2, it is invalid for expanded axes.																						
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>iEnable</td><td>If opens compensation: 1: ON, 0: OFF.</td></tr> <tr> <td>StartPos</td><td>Staring compensation MPOS position</td></tr> <tr> <td>maxpoint</td><td>Total points in compensation area</td></tr> <tr> <td>DisOne</td><td>Each compensation point's gap</td></tr> <tr> <td>TablNum</td><td>TABLE starting address where saves the compensation coordinates.</td></tr> <tr> <td>pfDisancelist</td><td>Area compensation value list</td></tr> <tr> <td>RevTablNum</td><td>TABLE starting address where saves the reverse compensation coordinates.</td></tr> <tr> <td>RevpfDisancelist</td><td>TABLE starting address where saves the reverse area compensation coordinates.</td></tr> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.	iEnable	If opens compensation: 1: ON, 0: OFF.	StartPos	Staring compensation MPOS position	maxpoint	Total points in compensation area	DisOne	Each compensation point's gap	TablNum	TABLE starting address where saves the compensation coordinates.	pfDisancelist	Area compensation value list	RevTablNum	TABLE starting address where saves the reverse compensation coordinates.	RevpfDisancelist	TABLE starting address where saves the reverse area compensation coordinates.
Parameter name	Description																						
handle	Link mark																						
iaxis	Axis No.																						
iEnable	If opens compensation: 1: ON, 0: OFF.																						
StartPos	Staring compensation MPOS position																						
maxpoint	Total points in compensation area																						
DisOne	Each compensation point's gap																						
TablNum	TABLE starting address where saves the compensation coordinates.																						
pfDisancelist	Area compensation value list																						
RevTablNum	TABLE starting address where saves the reverse compensation coordinates.																						
RevpfDisancelist	TABLE starting address where saves the reverse area compensation coordinates.																						
OP Parameters	/																						
Return value	0 = success, not-0 = failure (please check error codes for details).																						
Example	/																						
Details	/																						

Instruction 196	ZAux_Direct_GetPitchStatus	
Original Format	int32 __stdcall ZAux_Direct_GetPitchStatus(ZMC_HANDLE handle,int iaxis,int * IfEnable,float * PitchDist)	
Description	Read axis' pitch compensation state, it is invalid in expanded axes.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	iEnable	If opens compensation: 1: ON, 0: OFF.
	PicthDist	Compensation distance
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	/	

Instruction 197	ZAux_Direct_GetIn	
Original Format	int32 __stdcall ZAux_Direct_GetIn(ZMC_HANDLE handle,int ionum,uint32 *piValue)	
Description	Read input state, please refer to IN command in RTBasic Manual .	
Input parameters	Parameter name	Description
	handle	Link mark
	ionum	Input No.

Output parameters	Parameter name	Description
	piValue	Get input state values
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	IO reading and setting.	
Details	- If INVERT_IN is used to invert the level, what is read is the state after INVERT_IN flipping. - The IO channel No. of the ZIO expansion board is related to the dial code. The initial value is (16 + dial code combination value * 16). For EIO bus expansion, it uses the NODE_IO command, which only can be set to a multiple of 8. Please refer to the hardware manual for details. - The IO mapping No. must > the controller itself max IO No., and it cannot coincide with the controller No.	

Instruction 198	ZAux_Direct_SetOp	
Original Format	int32 __stdcall ZAux_Direct_SetOp(ZMC_HANDLE handle, int ionum, uint32 iValue);	
Description	Open output, please refer to OP command in RTBasic Manual	
Input parameters	Parameter name	Description
	handle	Link mark
	ionum	Input No.
	piValue	Output port's state values that are set.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	IO reading and setting.	
Details	- The IO channel No. of the ZIO expansion board is related to the dial code. The initial value is (16 + dial code combination value * 16). For EIO bus expansion, it uses the NODE_IO command, which only can be set to a multiple of 8. Please refer to the hardware manual for details. - The IO mapping No. must > the controller itself max IO No., and it cannot coincide with the controller No.	

Instruction 199	ZAux_Direct_GetOp	
Original Format	int32 __stdcall ZAux_Direct_GetOp(ZMC_HANDLE handle, int ionum, uint32 *piValue);	
Description	Read output, please refer to OP command in RTBasic Manual	
Input parameters	Parameter name	Description
	handle	Link mark
	ionum	output No.
Output parameters	Parameter name	Description
	piValue	Get output state values
Return value	0 = success, not-0 = failure (please check error codes for details).	

Example	IO reading and setting.
Details	- The IO channel No. of the ZIO expansion board is related to the dial code. The initial value is (16 + dial code combination value * 16). For EIO bus expansion, it uses the NODE_IO command, which only can be set to a multiple of 8. Please refer to the hardware manual for details. - The IO mapping No. must > the controller itself max IO No., and it cannot coincide with the controller No. 32 outputs can be operated at most.

Instruction 200	ZAux_GetModbusIn								
Original Format	int32 __stdcall ZAux_GetModbusIn (ZMC_HANDLE handle,int ionumfirst, int ionumend, uint8 * pValueList);								
Description	Rapidly read current multiple input states.								
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>ionumfirst</td><td>Input starting No.</td></tr> <tr> <td>ionumend</td><td>Input end No.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	ionumfirst	Input starting No.	ionumend	Input end No.
Parameter name	Description								
handle	Link mark								
ionumfirst	Input starting No.								
ionumend	Input end No.								
Output parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>pValueList</td><td>Bit state, it is store bit by bit, one unit8 corresponds to 8 inputs states, 8 or above can be read.</td></tr> </tbody> </table>	Parameter name	Description	pValueList	Bit state, it is store bit by bit, one unit8 corresponds to 8 inputs states, 8 or above can be read.				
Parameter name	Description								
pValueList	Bit state, it is store bit by bit, one unit8 corresponds to 8 inputs states, 8 or above can be read.								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	Multiple IO reading								
Details	The state obtained by the Modbus method is the state before inversion. That is, if INVERT_IN is used to invert, the read state may be wrong.								

Instruction 201	ZAux_GetModbusOut								
Original Format	int32 __stdcall ZAux_GetModbusOut (ZMC_HANDLE handle,int ionumfirst, int ionumend, uint8 * pValueList);								
Description	Rapidly read multiple current output states. The state obtained by the Modbus method is the state before inversion.								
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>ionumfirst</td><td>OUT starting No.</td></tr> <tr> <td>ionumend</td><td>OUT end No.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	ionumfirst	OUT starting No.	ionumend	OUT end No.
Parameter name	Description								
handle	Link mark								
ionumfirst	OUT starting No.								
ionumend	OUT end No.								
Output parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>pValueList</td><td>Bit state, it is store bit by bit, one unit8 corresponds to 8 inputs states, 8 or above can be read.</td></tr> </tbody> </table>	Parameter name	Description	pValueList	Bit state, it is store bit by bit, one unit8 corresponds to 8 inputs states, 8 or above can be read.				
Parameter name	Description								
pValueList	Bit state, it is store bit by bit, one unit8 corresponds to 8 inputs states, 8 or above can be read.								
Return value	0 = success, not-0 = failure (please check error codes for details).								

Example	Multiple IO reading

Instruction 202	ZAux_Direct_SetOutMulti										
Original Format	int32 __stdcall ZAux_SetOutMulti (ZMC_HANDLE handle, uint16 iofirst, uint16 ioend, uint32 *istate);										
Description	Set multiple outputs.										
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>ionumfirst</td><td>OUT starting No.</td></tr> <tr> <td>ionumend</td><td>OUT end No.</td></tr> <tr> <td>istate</td><td>Output state, istate is set bit by bit, one UNIT corresponds to 32 outputs. 32 or above can be set.</td></tr> </table>	Parameter name	Description	handle	Link mark	ionumfirst	OUT starting No.	ionumend	OUT end No.	istate	Output state, istate is set bit by bit, one UNIT corresponds to 32 outputs. 32 or above can be set.
Parameter name	Description										
handle	Link mark										
ionumfirst	OUT starting No.										
ionumend	OUT end No.										
istate	Output state, istate is set bit by bit, one UNIT corresponds to 32 outputs. 32 or above can be set.										
OP Parameters	/										
Return value	0 = success, not-0 = failure (please check error codes for details).										
Example	Multiple IO reading										
Details	/										

Instruction 203	ZAux_Direct_GetOutMulti								
Original Format	int32 __stdcall ZAux_direct_GetOutMulti (ZMC_HANDLE handle, uint16 iofirst, uint16 ioend, uint32 *istate);								
Description	Read multiple output states.								
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iofirst</td><td>IO starting No.</td></tr> <tr> <td>ioend</td><td>IO end No.</td></tr> </table>	Parameter name	Description	handle	Link mark	iofirst	IO starting No.	ioend	IO end No.
Parameter name	Description								
handle	Link mark								
iofirst	IO starting No.								
ioend	IO end No.								
Output parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>istate</td><td>Read IO state bit by bit, one UNIT corresponds to 32 outputs. 32 or above can be read.</td></tr> </table>	Parameter name	Description	istate	Read IO state bit by bit, one UNIT corresponds to 32 outputs. 32 or above can be read.				
Parameter name	Description								
istate	Read IO state bit by bit, one UNIT corresponds to 32 outputs. 32 or above can be read.								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	Multiple IO reading								
Details	/								

Instruction 204	ZAux_Direct_GetInMulti								
Original Format	int32 __stdcall ZAux_Direct_GetInMulti (ZMC_HANDLE handle, int startio, int endio, int32 *piValue);								
Description	Read multiple inputs states.								
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>Handle</td><td>Link mark</td></tr> <tr> <td>lofirst</td><td>IO starting No.</td></tr> <tr> <td>loend</td><td>IO end No.</td></tr> </table>	Parameter name	Description	Handle	Link mark	lofirst	IO starting No.	loend	IO end No.
Parameter name	Description								
Handle	Link mark								
lofirst	IO starting No.								
loend	IO end No.								

Output parameters	Parameter name	Description
	Istate	Get IO state bit by bit, one UNIT corresponds to 32 inputs. 32 or above can be read.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Multiple IO reading	
Details	/	

Instruction 205	ZAux_Direct_SetInvertIn	
Original Format	int32 __stdcall ZAux_Direct_SetInvertIn(ZMC_HANDLE handle, int ionum, int biflInvert)	
Description	Invert the input state, please refer to INVERT_IN command in RTBasic Programming Manual .	
Input parameters	Parameter name	Description
	Handle	Link mark
	Ionum	Input No.
	BiflInvert	State: 0 – OFF, 1 – ON
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Position limit settings	
Details	For ZMC series controllers, it considers that there is signal input when the IN is OFF (it is opposite for ECI series controllers).	

Instruction 206	ZAux_Direct_GetInvertIn	
Original Format	int32 __stdcall ZAux_Direct_GetInvertIn(ZMC_HANDLE handle, int ionum, int * piValue)	
Description	Read input "invert" state.	
Input parameters	Parameter name	Description
	Handle	Link mark
	Ionum	Input No.
Output parameters	Parameter name	Description
	piValue	Returned inverse input state. 0: OFF, 1: ON.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Position limit settings	
Details	For ZMC series controllers, it considers that there is signal input when the IN is OFF (it is opposite for ECI series controllers).	

Instruction 207	ZAux_Direct_GetAD	
Original Format	int32 __stdcall ZAux_Direct_GetAD(ZMC_HANDLE handle, int ionum, float *pfValue);	
Description	Read analog input values. please refer to AIN command in RTBasic	

	Programming Manual.	
Input parameters	Parameter name	Description
	Handle	Link mark
	Ionum	AIN No.
Output parameters	Parameter name	Description
	pfValue	Returned analog values
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	AD/DA Setting & Getting	
Details	● For 12-bit, the scale value range 0~4095 corresponds to 0~10V voltage. ● For 16-bit, the scale value range 0~65536 corresponds to 0~10V voltage. ● The AD channel No. of the ZAI/O expansion board is related to the dial code. The initial value is "8 + dial code combination value * 8". ● The AD expanded by ZMIO bus expansion only can be set to a multiple of 8 through "NODE_AIO". Please refer to the user manual for details. ● AIO mapping No. must be greater than the controller itself maximum AIO No., and they can't be the same.	

Instruction 208	ZAux_Direct_SetDA	
Original Format	int32 __stdcall ZAux_Direct_SetDA(ZMC_HANDLE handle, int ionum, float fValue);	
Description	Set analog output values. please refer to AOUT command in RTBasic Programming Manual .	
Input parameters	Parameter name	Description
	Handle	Link mark
	Ionum	DA output No.
	fValue	Set the value.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	AD/DA Setting & Getting	
Details	- For 12-bit, the scale value range 0~4095 corresponds to 0~10V voltage. - For 16-bit, the scale value range 0~65536 corresponds to 0~10V voltage.	

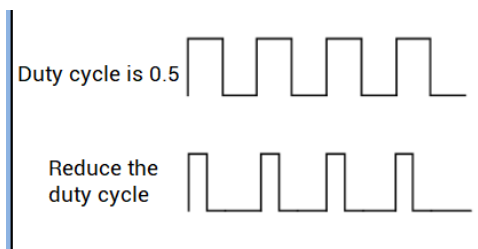
Instruction 209	ZAux_Direct_GetDA
Original Format	int32 __stdcall ZAux_Direct_GetDA(ZMC_HANDLE handle, int ionum, float *pfValue);
Description	Read analog output values. please refer to AOUT command in

	RTBasic Programming Manual.	
Input parameters	Parameter name	Description
	Handle	Link mark
	Ionum	DA output No.
Output parameters	Parameter name	Description
	piValue	Returned output values
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	AD/DA Setting & Getting	
Details	- For 12-bit, the scale value range 0~4095 corresponds to 0~10V voltage. - For 16-bit, the scale value range 0~65536 corresponds to 0~10V voltage.	

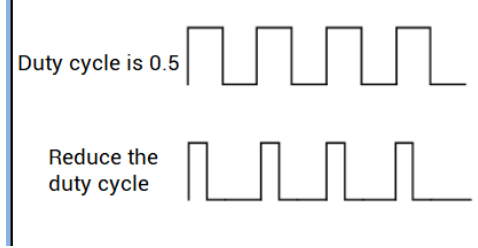
Instruction 210	ZAux_Direct_SetPwmFreq	
Original Format	int32 __stdcall ZAux_Direct_SetPwmFreq(ZMC_HANDLE handle, int ionum, float fValue);	
Description	Set PWM frequency, please refer to "PWM_DREQ" command in RTBasic Programming Manual.	
Input parameters	Parameter name	Description
	Handle	Link mark
	Ionum	PWM No.
	fValue	Configured value.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	PWM Usage	
Details	- PWM only can be turned off by setting the duty cycle to 0. - Don't set the frequency to 0, the PWM frequency must be adjusted before the PWM switch.	

Instruction 211	ZAux_Direct_GetPwmFreq	
Original Format	int32 __stdcall ZAux_Direct_GetPwmFreq(ZMC_HANDLE handle, int ionum, float *pfValue);	
Description	Read PWM frequency.	
Input parameters	Parameter name	Description
	Handle	Link mark
	Ionum	PWM No.
Output parameters	Parameter name	Description
	fValue	Configured value.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	PWM Usage	
Details	- PWM only can be turned off by setting the duty cycle to 0. - Don't set the frequency to 0, the PWM frequency must be	

	adjusted before the PWM switch.

Instruction 212	ZAux_Direct_SetPwmDuty								
Original Format	int32 __stdcall ZAux_Direct_SetPwmDuty(ZMC_HANDLE handle, int ionum, float fValue);								
Description	Set PWM duty cycle, please refer to "PWM_DUTY" command in RTBasic Programming Manual .								
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Handle</td><td>Link mark</td></tr> <tr> <td>ionum</td><td>PWM No.</td></tr> <tr> <td>fValue</td><td>Configured value.</td></tr> </tbody> </table>	Parameter name	Description	Handle	Link mark	ionum	PWM No.	fValue	Configured value.
Parameter name	Description								
Handle	Link mark								
ionum	PWM No.								
fValue	Configured value.								
OP Parameters	/								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	PWM Usage								
Details	- PWM only can be turned off by setting the duty cycle to 0. - Don't set the frequency to 0, the PWM frequency must be adjusted before the PWM switch. - The duty cycle: the ratio of the valid level to the entire period, and in one cycle, output the valid level at first, then output invalid level.  - The actual output of the PWM is controlled by the output port, so please open output to make PWM output, otherwise the output will be shielded. Also, you could open PWM function at first, then open OP to achieve laser power's first pulse suppression function.								

Instruction 213	ZAux_Direct_GetPwmDuty						
Original Format	int32 __stdcall ZAux_Direct_GetPwmDuty(ZMC_HANDLE handle, int ionum, float *pfValue);						
Description	Read PWM duty cycle.						
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>ionum</td><td>PWM No.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	ionum	PWM No.
Parameter name	Description						
handle	Link mark						
ionum	PWM No.						
Output parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>fValue</td><td>Configured value.</td></tr> </tbody> </table>	Parameter name	Description	fValue	Configured value.		
Parameter name	Description						
fValue	Configured value.						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	PWM Usage						

Details	● PWM only can be turned off by setting the duty cycle to 0. ● Don't set the frequency to 0, the PWM frequency must be adjusted before the PWM switch. ● The duty cycle: the ratio of the valid level to the entire period, and in one cycle, output the valid level at first, then output invalid level.  ● The actual output of the PWM is controlled by the output port, so please open output to make PWM output, otherwise the output will be shielded. Also, you could open PWM function at first, then open OP to achieve laser power's first pulse suppression function.

Instruction 214	ZAux_Direct_SetFsLimit	
Original Format	int32 __stdcall ZAux_Direct_SetFsLimit(ZMC_HANDLE handle, int iaxis, float fValue)	
Description	Set axis forward soft position limit. Set a bigger value to cancel it. The unit is units.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
	fValue	Configured forward position limit value.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Position limit settings	
Details	- When FS_LIMIT > REP_DIST, this parameter is invalid, the forward soft limit is prohibited. - If you want to cancel the soft position limit, you could set one bigger value for FS_LIMIT (default value: 200000000), don't modify REP_DIST value. - The soft limit cannot be used as a reference signal when homing by DATUM. - Before setting the limit, please check whether acceleration, deceleration, fast deceleration are set or not. If there is no setting, it will not stop when the limit is met.	

Instruction 215	ZAux_Direct_GetFsLimit
Original Format	int32 __stdcall ZAux_Direct_GetFsLimit(ZMC_HANDLE handle, int iaxis, float *fValue)

Description	Read the forward soft position limit value.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	fValue	Returned positive position limit value
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Position limit settings	
Details	- When FS_LIMIT > REP_DIST, this parameter is invalid, the forward soft limit is prohibited. - If you want to cancel the soft position limit, you could set one bigger value for FS_LIMIT (default value: 200000000), don't modify REP_DIST value. - The soft limit cannot be used as a reference signal when homing by DATUM. - Before setting the limit, please check whether acceleration, deceleration, fast deceleration are set or not. If there is no setting, it will not stop when the limit is met.	

Instruction 216	ZAux_Direct_SetRsLimit	
Original Format	Set axis reverse soft position limit. Set a bigger value to cancel it. The unit is units.	
Description	Set the negative soft limit of the axis, and set a larger value when canceling the soft limit. the unit is units.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
	fValue	Configured negative position limit value.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Position limit settings	
Details	- When FS_LIMIT > REP_DIST, this parameter is invalid, the forward soft limit is prohibited. - If you want to cancel the soft position limit, you could set one bigger value for FS_LIMIT (default value: 200000000), don't modify REP_DIST value. - The soft limit cannot be used as a reference signal when homing by DATUM. - Before setting the limit, please check whether acceleration, deceleration, fast deceleration are set or not. If there is no setting, it will not stop when the limit is met.	

Instruction 217	ZAux_Direct_GetRsLimit	
Original Format	int32 __stdcall ZAux_Direct_GetRsLimit(ZMC_HANDLE handle, int iaxis, float *fValue)	
Description	Read the inverse soft position limit value.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
Output parameters	Parameter name	Description
	fValue	Returned negative position limit value
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Position limit settings	
Details	- When FS_LIMIT > REP_DIST, this parameter is invalid, the forward soft limit is prohibited. - If you want to cancel the soft position limit, you could set one bigger value for FS_LIMIT (default value: 200000000), don't modify REP_DIST value. - The soft limit cannot be used as a reference signal when homing by DATUM. - Before setting the limit, please check whether acceleration, deceleration, fast deceleration are set or not. If there is no setting, it will not stop when the limit is met.	

Instruction 218	ZAux_Direct_SetFwdIn	
Original Format	int32 __stdcall ZAux_Direct_SetFwdIn(ZMC_HANDLE handle, int iaxis, int iValue)	
Description	Set axis forward hard position limit. Set as -1 to cancel it. Please refer to "FWD_IN" command in RTBasic Programming Manual .	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
	iValue	IO No., -1: cancel the setting.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Position limit settings	
Details	- Before setting the limit, please check whether acceleration, deceleration, fast deceleration are set or not. If there is no setting, it will not stop when the limit is met. - When input is OFF, it considers there is signal input. INVERT_IN can be used to make the level invert (for ECI series, the usage is opposite). - Position limits must be set, otherwise, it will keep moving when meets the position limit.	

Instruction 219	ZAux_Direct_GetFwdIn						
Original Format	int32 __stdcall ZAux_Direct_GetFwdIn (ZMC_HANDLE handle, int iaxis, int * piValue)						
Description	Read axis forward hard position limit IO No.						
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.
Parameter name	Description						
handle	Link mark						
iaxis	Axis No.						
Output parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>piValue</td><td>Returned corresponding IN No., -1 means not to set.</td></tr> </table>	Parameter name	Description	piValue	Returned corresponding IN No., -1 means not to set.		
Parameter name	Description						
piValue	Returned corresponding IN No., -1 means not to set.						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	Position limit settings						
Details	- Before setting the limit, please check whether acceleration, deceleration, fast deceleration are set or not. If there is no setting, it will not stop when the limit is met. - When input is OFF, it considers there is signal input. INVERT_IN can be used to make the level invert (for ECI series, the usage is opposite). - Position limits must be set, otherwise, it will keep moving when meets the position limit.						

Instruction 220	ZAux_Direct_SetRevIn								
Original Format	int32 __stdcall ZAux_Direct_SetRevIn(ZMC_HANDLE handle, int iaxis,int iValue);								
Description	Set axis forward hard position limit. Set as -1 to cancel it. Please refer to "REV_IN" command in RTBasic Programming Manual .								
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>iValue</td><td>IO No., -1: cancel the setting.</td></tr> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.	iValue	IO No., -1: cancel the setting.
Parameter name	Description								
handle	Link mark								
iaxis	Axis No.								
iValue	IO No., -1: cancel the setting.								
OP Parameters	/								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	Position limit settings								
Details	- Before setting the limit, please check whether acceleration, deceleration, fast deceleration are set or not. If there is no setting, it will not stop when the limit is met. - When input is OFF, it considers there is signal input. INVERT_IN can be used to make the level invert (for ECI series, the usage is opposite). - Position limits must be set, otherwise, it will keep moving when meets the position limit.								

Instruction 221	ZAux_Direct_GetRevIn	
Original Format	int32 __stdcall ZAux_Direct_GetRevIn(ZMC_HANDLE handle, int iaxis,int* piValue);	
Description	Read axis inverse hard position limit IO No.	
Input parameters	Parameter name	Description
	Handle	Link mark
	laxis	Axis No.
Output parameters	Parameter name	Description
	piValue	Returned corresponding IN No., -1 means not to set.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Position limit settings	
Details	● Before setting the limit, please check whether acceleration, deceleration, fast deceleration are set or not. If there is no setting, it will not stop when the limit is met. ● When input is OFF, it considers there is signal input. INVERT_IN can be used to make the level invert (for ECI series, the usage is opposite). ● Position limits must be set, otherwise, it will keep moving when meets the position limit.	

Instruction 222	ZAux_Direct_SetAlmIn	
Original Format	int32 __stdcall ZAux_Direct_SetAlmIn(ZMC_HANDLE handle, int iaxis,int iValue)	
Description	Set axis alarm signal. Set as -1 to cancel it. Please refer to "ALM_IN" command in RTBasic Programming Manual .	
Input parameters	Parameter name	Description
	Handle	Link mark
	laxis	Axis No.
	iValue	IO No., -1 means cancelling the setting.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Alarm IN Settings	
Details	- Before setting the axis alarm signal, please check whether acceleration, deceleration, fast deceleration are set or not. If there is no setting, it will not stop when the limit is met. - When input is OFF, it considers there is signal input. INVERT_IN can be used to make the level invert (for ECI series, the usage is opposite).	

Instruction 223	ZAux_Direct_GetAlmIn
Original Format	int32 __stdcall ZAux_Direct_GetAlmIn(ZMC_HANDLE handle, int iaxis,int * piValue)

Description	Read alarm IO No. of selected axis.	
Input parameters	Parameter name	Description
	Handle	Link mark
	Iaxis	Axis No.
Output parameters	Parameter name	Description
	piValue	Returned corresponding IN No., -1 means not to set.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Alarm IN Settings	
Details	- Before setting the axis alarm signal, please check whether acceleration, deceleration, fast deceleration are set or not. If there is no setting, it will not stop when the limit is met. - When input is OFF, it considers there is signal input. INVERT_IN can be used to make the level invert (for ECI series, the usage is opposite).	

Instruction 224	ZAux_Direct_SetVrf	
Original Format	int32 __stdcall ZAux_Direct_SetVrf(ZMC_HANDLE handle,int vrstartnum, int numes, float *pfValue);	
Description	Set data in VR.	
Input parameters	Parameter name	Description
	handle	Link mark
	vrstartnum	VR starting No.
	numes	The number of writings
	pfValue	Data list
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	VR Register Usage	
Details	- It is power-down storage register 32-bit floating type, and the floating number is saved, they share same space with VR_INT and VRSTRING.	

Instruction 225	ZAux_Direct_GetVrf	
Original Format	int32 __stdcall ZAux_Direct_GetVrf(ZMC_HANDLE handle, int vrstartnum, int numes, float *pfValue)	
Description	Read data in VR.	
Input parameters	Parameter name	Description
	handle	Link mark
	vrstartnum	VR starting No.
	numes	The number of readings
Output parameters	Parameter name	Description
	pfValue	Data list

Return value	0 = success, not-0 = failure (please check error codes for details).
Example	VR Register Usage
Details	- It is power-down storage register 32-bit floating type, and the floating number is saved, they share same space with VR_INT and VRSTRING.

Instruction 226	ZAux_Direct_GetVrInt								
Original Format	int32 __stdcall ZAux_Direct_GetVrInt (ZMC_HANDLE handle ,int vrstartnum, int numes, int * piValue);								
Description	Read VR data in integer. The firmware version should be above 150401.								
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>vrstartnum</td><td>VR starting No.</td></tr> <tr> <td>numes</td><td>The number of readings</td></tr> </table>	Parameter name	Description	handle	Link mark	vrstartnum	VR starting No.	numes	The number of readings
Parameter name	Description								
handle	Link mark								
vrstartnum	VR starting No.								
numes	The number of readings								
Output parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>pfValue</td><td>Data list</td></tr> </table>	Parameter name	Description	pfValue	Data list				
Parameter name	Description								
pfValue	Data list								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	/								
Details	/								

Instruction 227	ZAux_Direct_SetTable										
Original Format	int32 __stdcall ZAux_Direct_SetTable(ZMC_HANDLE handle,int tabstart, int numes, float *pfValue)										
Description	Write data into TABLE.										
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>tablestart</td><td>TABLE starting No.</td></tr> <tr> <td>numes</td><td>The number of writings</td></tr> <tr> <td>pfValue</td><td>Data list</td></tr> </table>	Parameter name	Description	handle	Link mark	tablestart	TABLE starting No.	numes	The number of writings	pfValue	Data list
Parameter name	Description										
handle	Link mark										
tablestart	TABLE starting No.										
numes	The number of writings										
pfValue	Data list										
OP Parameters	/										
Return value	0 = success, not-0 = failure (please check error codes for details).										
Example	TABLE Register Usage										
Details	- It is a super large array that comes with the controller. - The data type is 32-bit floating point (if you need 64-bit floating, please use ZMC4XX series and above). - It will not be saved when power off.										

Instruction 228	ZAux_Direct_GetTable
Original Format	int32 __stdcall ZAux_Direct_GetTable(ZMC_HANDLE handle, int tabstart, int numes, float *pfValue)

Description	Read TABLE data.	
Input parameters	Parameter name	Description
	handle	Link mark
	tablestart	TABLE No.
	numes	The number of readings
Output parameters	Parameter name	Description
	pfValue	Data list
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	TABLE Register Usage	
Details	- It is a super large array that comes with the controller. - The data type is 32-bit floating point (if you need 64-bit floating, please use ZMC4XX series and above). - It will not be saved when power off.	

Instruction 229	ZAux_Modbus_Set0x	
Original Format	int32 __stdcall ZAux_Modbus_Set0x(ZMC_HANDLE handle, uint16 start, uint16 inum, uint8* pdata)	
Description	Set modbus bit register.	
Input parameters	Parameter name	Description
	handle	Link mark
	start	MODBUS_BIT starting No.
	inum	The number of writings
	pdata	Data list
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	MODBUS Register Usage	
Details	/	

Instruction 230	ZAux_Modbus_Get0x	
Original Format	int32 __stdcall ZAux_Modbus_Get0x(ZMC_HANDLE handle, uint16 start, uint16 inum, uint8* pdata)	
Description	Read modbus bit register.	
Input parameters	Parameter name	Description
	handle	Link mark
	start	MODBUS_BIT starting No.
	inum	The number of readings
Output parameters	Parameter name	Description
	pdata	Data list
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	MODBUS Register Usage	
Details	/	

Instruction 231	ZAux_Modbus_Set4x										
Original Format	int32 __stdcall ZAux_Modbus_Set4x(ZMC_HANDLE handle, uint16 start, uint16 inum, uint16* pdata)										
Description	Set modbus word register.										
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>start</td><td>MODBUS_REG starting No.</td></tr> <tr> <td>inum</td><td>The number of writings</td></tr> <tr> <td>pdata</td><td>Data list</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	start	MODBUS_REG starting No.	inum	The number of writings	pdata	Data list
Parameter name	Description										
handle	Link mark										
start	MODBUS_REG starting No.										
inum	The number of writings										
pdata	Data list										
OP Parameters	/										
Return value	0 = success, not-0 = failure (please check error codes for details).										
Example	MODBUS Register Usage										
Details	MODBUS word register, this function corresponds to modbus_reg, sharing with other MODBUS register space, please refer to 10.3 .										

Instruction 232	ZAux_Modbus_Get4x								
Original Format	int32 __stdcall ZAux_Modbus_Get4x(ZMC_HANDLE handle, uint16 start, uint16 inum, uint16* pdata)								
Description	Read modbus word register.								
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>start</td><td>MODBUS_REG starting No.</td></tr> <tr> <td>inum</td><td>The number of readings</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	start	MODBUS_REG starting No.	inum	The number of readings
Parameter name	Description								
handle	Link mark								
start	MODBUS_REG starting No.								
inum	The number of readings								
Output parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>pdata</td><td>Data list</td></tr> </tbody> </table>	Parameter name	Description	pdata	Data list				
Parameter name	Description								
pdata	Data list								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	MODBUS Register Usage								
Details	MODBUS word register, this function corresponds to modbus_reg, sharing with other MODBUS register space, please refer to 10.3 .								

Instruction 233	ZAux_Modbus_Set4x_Long										
Original Format	int32 __stdcall ZAux_Modbus_Set4x_Long (ZMC_HANDLE handle, uint16 start, uint16 inum, int32 * pdata);										
Description	Set MODBUS_LONG.										
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>start</td><td>MODBUS_LONG starting No.</td></tr> <tr> <td>inum</td><td>The number of writings</td></tr> <tr> <td>pdata</td><td>Data list</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	start	MODBUS_LONG starting No.	inum	The number of writings	pdata	Data list
Parameter name	Description										
handle	Link mark										
start	MODBUS_LONG starting No.										
inum	The number of writings										
pdata	Data list										
OP Parameters	/										
Return value	0 = success, not-0 = failure (please check error codes for details).										

Example	/
Details	- One MODBUS_LONG occupies two MODBUS_REG addresses, for example, MODBUS_LONG (0) occupies MODBUS_REG(0) and MODBUS_REG(1). - MODBUS word register, this function corresponds to modbus_long, sharing with other MODBUS register space, please refer to 10.3.

Instruction 234	ZAux_Modbus_Get4x_Long	
Original Format	int32 __stdcall ZAux_Modbus_Get4x_Long (ZMC_HANDLE handle, uint16 start, uint16 inum, int32 * pdata);	
Description	Read MODBUS_LONG.	
Input parameters	Parameter name	Description
	handle	Link mark
	start	MODBUS_LONG starting No.
	inum	The number of readings.
Output parameters	Parameter name	Description
	pdata	Data list
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	- One MODBUS_LONG occupies two MODBUS_REG addresses, for example, MODBUS_LONG (0) occupies MODBUS_REG(0) and MODBUS_REG(1). - MODBUS word register, this function corresponds to modbus_long, sharing with other MODBUS register space, please refer to 10.3.	

Instruction 235	ZAux_Modbus_Set4x_String	
Original Format	int32 __stdcall ZAux_Modbus_Set4x_String (ZMC_HANDLE handle , uint16 start,uint16 inum, char * pdata);	
Description	Set modbus_string.	
Input parameters	Parameter name	Description
	handle	Link mark
	start	Modbus starting address
	inum	Length
	pdata	Written character string.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	MODBUS word register, this function corresponds to modbus_string, sharing with other MODBUS register space, please refer to 10.3 .	

Instruction 236	ZAux_Modbus_Get4x_String								
Original Format	int32 __stdcall ZAux_Modbus_Get4x_String (ZMC_HANDLE handle, uint16 start, uint16 inum, char * pldata);								
Description	Read modbus_string.								
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>start</td><td>Modbus starting address</td></tr> <tr> <td>inum</td><td>Length</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	start	Modbus starting address	inum	Length
Parameter name	Description								
handle	Link mark								
start	Modbus starting address								
inum	Length								
Output parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>pdata</td><td>Read character string.</td></tr> </tbody> </table>	Parameter name	Description	pdata	Read character string.				
Parameter name	Description								
pdata	Read character string.								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	/								
Details	MODBUS word register, this function corresponds to modbus_string, sharing with other MODBUS register space, please refer to 10.3 .								

Instruction 237	ZAux_Modbus_Set4x_Float										
Original Format	int32 __stdcall ZAux_Modbus_Set4x_Float (ZMC_HANDLE handle, uint16 start, uint16 inum, float * pldata);										
Description	Set MODBUS_IEEE										
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>start</td><td>MODBUS_IEEE starting No.</td></tr> <tr> <td>inum</td><td>The number of writings</td></tr> <tr> <td>pdata</td><td>Data list</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	start	MODBUS_IEEE starting No.	inum	The number of writings	pdata	Data list
Parameter name	Description										
handle	Link mark										
start	MODBUS_IEEE starting No.										
inum	The number of writings										
pdata	Data list										
OP Parameters	/										
Return value	0 = success, not-0 = failure (please check error codes for details).										
Example	/										
Details	- One MODBUS_IEEE occupies two MODBUS_REG addresses, for example, MODBUS_IEEE (0) occupies MODBUS_REG(0) and MODBUS_REG(1). - MODBUS word register, this function corresponds to modbus_ieee, sharing with other MODBUS register space, please refer to 10.3.										

Instruction 238	ZAux_Modbus_Get4x_Float
Original Format	int32 __stdcall ZAux_Modbus_Get4x_Float(ZMC_HANDLE handle, uint16 start, uint16 inum, float * pldata);
Description	Read MODBUS_IEEE.

Input parameters	Parameter name	Description
	handle	Link mark
	start	MODBUS_IEEE starting No.
	inum	The number of readings
Output parameters	Parameter name	Description
	pdata	Data list
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	- One MODBUS_IEEE occupies two MODBUS_REG addresses, for example, MODBUS_IEEE (0) occupies MODBUS_REG(0) and MODBUS_REG(1). - MODBUS word register, this function corresponds to modbus_ieee, sharing with other MODBUS register space, please refer to 10.3.	

Instruction 239	ZAux_WriteUFile	
Original Format	int32 __stdcall ZAux_WriteUFile(const char *sFilename, float *pVarlist, int inum)	
Description	Store variables list with float format into file, and it is consistent with controller USB file's format.	
Input parameters	Parameter name	Description
	sFilename	File name.
	pVarlist	Data list.
	inum	Data length.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	File reading and writing test	
Details	/	

Instruction 240	ZAux_ReadUFile	
Original Format	int32 __stdcall ZAux_ReadUFile(const char *sFilename, float *pVarlist, int*pinum)	
Description	Read variables list with float format, and it is consistent with controller USB file's format.	
Input parameters	Parameter name	Description
	sFilename	File name.
	inum	Data length.
Output parameters	Parameter name	Description
	pVarlist	Data list.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	File reading and writing test	
Details	/	

Instruction 241	ZAux_FlashWritef										
Original Format	int32 __stdcall ZAux_FlashWritef(ZMC_HANDLE handle, uint16 uiflashid, uint32 uinumes, float *pflue)										
Description	Write float data into Flash block.										
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link handle</td></tr> <tr> <td>uiflashid</td><td>FLASH block No.</td></tr> <tr> <td>uinumes</td><td>The number of writings</td></tr> <tr> <td>pflue</td><td>Data list</td></tr> </table>	Parameter name	Description	handle	Link handle	uiflashid	FLASH block No.	uinumes	The number of writings	pflue	Data list
Parameter name	Description										
handle	Link handle										
uiflashid	FLASH block No.										
uinumes	The number of writings										
pflue	Data list										
OP Parameters	/										
Return value	0 = success, not-0 = failure (please check error codes for details).										
Example	Flash reading and writing test										
Details	- The internal FLASH is saved in sequence, and reading sequence must be consistent with storage sequence. - The internal FLASH has a limit on the number of storage times, so do not operate in cycle. - Be careful not to operate the FLASH during the motion, it will affect the execution of the motion.										

Instruction 242	ZAux_FlashReadf								
Original Format	int32 __stdcall ZAux_FlashReadf(ZMC_HANDLE handle, uint16 uiflashid, uint32 uibuffnum, float *pflue, uint32* puinumesread)								
Description	Read user data in float type from Flash block.								
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link handle</td></tr> <tr> <td>uiflashid</td><td>FLASH block No.</td></tr> <tr> <td>uibuffnum</td><td>The number of buffer variables</td></tr> </table>	Parameter name	Description	handle	Link handle	uiflashid	FLASH block No.	uibuffnum	The number of buffer variables
Parameter name	Description								
handle	Link handle								
uiflashid	FLASH block No.								
uibuffnum	The number of buffer variables								
Output parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>Pflue</td><td>Data list.</td></tr> <tr> <td>puinumesread</td><td>The number of variables that is read.</td></tr> </table>	Parameter name	Description	Pflue	Data list.	puinumesread	The number of variables that is read.		
Parameter name	Description								
Pflue	Data list.								
puinumesread	The number of variables that is read.								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	Flash reading and writing test								
Details	- The internal FLASH is saved in sequence, and reading sequence must be consistent with storage sequence. - The internal FLASH has a limit on the number of storage times, so do not operate in cycle. - Be careful not to operate the FLASH during the motion, it will affect the execution of the motion.								

Instruction 243	ZAux_BusCmd_GetNodeNum
Original Format	int32 __stdcall ZAux_BusCmd_GetNodeNum(ZMC_HANDLE handle,int slot,int *piValue);

Description	Read the number of nodes.	
Input parameters	Parameter name	Description
	handle	Link handle
	slot	Slot No.
Output parameters	Parameter name	Description
	pivalue	Returned number of nodes.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	bus initialization (zmotiontools method)	
Details	/	

Instruction 244	ZAux_BusCmd_GetNodeInfo	
Original Format	int32 __stdcall ZAux_BusCmd_GetNodeInfo(ZMC_HANDLE handle, int slot, int node, int sel, int *piValue);	
Description	Read node information.	
Input parameters	Parameter name	Description
	handle	Link handle
	slot	Slot No.
	node	Node No.
	sel	Information No.
Output parameters	Parameter name	Description
	pivalue	Returned parameter information.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	bus initialization (zmotiontools method)	
Details	Sel value	Description
	0	EtherCAT node manufacturer Number
	1	EtherCAT node device Number
	2	EtherCAT node version
	3	EtherCAT node ALIAS alias
	4	Reserved
	10	EtherCAT node IN numbers
	11	EtherCAT node OUT numbers
	12	EtherCAT node AIN numbers
	13	EtherCAT node AOUT numbers
	14	Reserved

Instruction 245	ZAux_BusCmd_GetNodeStatus	
Original Format	int32 __stdcall ZAux_Direct_GetNodeStatus (ZMC_HANDLE handle, uint32 slot , uint32 node , uint32 *nodestatus);	
Description	Read node bus state.	

Input parameters	Parameter name	Description
	handle	Link handle
	slot	Slot No.
	node	Node No.
Output parameters	Parameter name	Description
	nodestatus	nodestatus processes bit by bit. Bit0: whether the node exists. Bit1: communication state. Bit2: node state.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	● Nodestatus: When the value is 1: bit 0 is 1, bit 1 and bit 2 are 0, the device communicates normally. When the value is 3, bit 0 and bit 1 are 1, bit 2 is 0, the device communicates abnormally.	
	BIT	Meaning
	0	Indicate whether node exists or not. 1-YES
	1	Communication state. 1-error.
	2	Node state. 1-error.

Instruction 246	ZAux_BusCmd_DriveClear	
Original Format	int32 __stdcall ZAux_BusCmd_DriveClear(ZMC_HANDLE handle, int iaxis, uint32 mode);	
Description	Clear bus servo alarm.	
Input parameters	Parameter name	Description
	handle	Link handle
	iaxis	Axis No.
	mode	0- clear current alarm 1- clear history alarm 2- clear external input alarm
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	When there is no error of the driver, it will warn 6015 when using the command, but it will not influence program execution.	

Instruction 247	ZAux_BusCmd_InitBus
Original Format	int32 __stdcall ZAux_BusCmd_InitBus(ZMC_HANDLE handle);
Description	Bus initialization.

Input parameters	Parameter name	Description
	handle	Link handle
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	bus initialization (zmotiontools method)	
Details	- It is valid in the controller whose bus parameters have been configured by "Zmotion tools" tool software. - Or you could download Basic file into controller through ZAux_BasDown function.	

Instruction 248	ZAux_BusCmd_GetInitStatus	
Original Format	int32 __stdcall ZAux_BusCmd_GetInitStatus(ZMC_HANDLE handle,int *piValue);	
Description	Get bus initialization state.	
Input parameters	Parameter name	Description
	handle	Link handle
Output parameters	Parameter name	Description
	pivalue	Returned state, 0: fail to initialize, 1: success to initialize.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	bus initialization (zmotiontools method)	
Details	- It is valid in the controller whose bus parameters have been configured by "Zmotion tools" tool software. - Or you could download Basic file into controller through ZAux_BasDown function.	

Instruction 249	ZAux_Direct_SetErrormask	
Original Format	int32 __stdcall ZAux_Direct_SetErrormask (ZMC_HANDLE handle, int iaxis , int iValue);	
Description	Set error mark.	
Input parameters	Parameter name	Description
	handle	Link handle
	iaxis	Axis No.
	iValue	The value that is set.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	To do "and" operation with AXISSTATUS to decide which errors' WDOG need to be closed.	

Instruction 250	ZAux_Direct_GetErrormask	
Original Format	int32 __stdcall ZAux_Direct_GetErrormask (ZMC_HANDLE handle, int iaxis , int * piValue);	
Description	Get error mark.	
Input parameters	Parameter name	Description
	handle	Link handle
	iaxis	Axis No.
	iValue	The value that is set.
Output parameters	Parameter name	Description
	pivalue	Returned mark value
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	To do "and" operation with AXISSTATUS to decide which errors' WDOG need to be closed.	

Instruction 251	ZAux_BusCmd_SDOWrite		
Original Format	int32 __stdcall ZAux_BusCmd_SDOWrite(ZMC_HANDLE handle, uint32 slot, uint32 node, uint32 index, uint32 subindex, uint32 type, int value);		
Description	Write SDO through device No. (node) and slot No.		
Input parameters	Parameter name	Description	
	handle	Link handle	
	slot	Slot No.	
	node	Node No.	
	index	Object dictionary No.	
	subindex	Object dictionary 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	
value	Written data value		
OP Parameters	/		
Return value	0 = success, not-0 = failure (please check error codes for details).		
Example	Max torque of SDO writing and reading.		
Details	● Before execution, please do collection and scan the bus. ● It only can write the data dictionary that can be written.		

Instruction 252	ZAux_BusCmd_SDORead																												
Original Format	int32 __stdcall ZAux_BusCmd_SDORead(ZMC_HANDLE handle, uint32 slot, uint32 node, uint32 index, uint32 subindex, uint32 type, int *value);																												
Description	Read SDO through device No. (node) and slot No.																												
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link handle</td></tr> <tr> <td>slot</td><td>Slot No.</td></tr> <tr> <td>node</td><td>Node No.</td></tr> <tr> <td>index</td><td>Object dictionary No.</td></tr> <tr> <td>subindex</td><td>Object dictionary sub No.</td></tr> <tr> <td>type</td><td>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> </tbody> </table> </td></tr> </tbody> </table>	Parameter name	Description	handle	Link handle	slot	Slot No.	node	Node No.	index	Object dictionary No.	subindex	Object dictionary sub 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> </tbody> </table>	1	boolean	2	integer 8	3	integer 16	4	integer 32	5	unsigned 8	6	unsigned 16	7	unsigned 32
Parameter name	Description																												
handle	Link handle																												
slot	Slot No.																												
node	Node No.																												
index	Object dictionary No.																												
subindex	Object dictionary sub 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> </tbody> </table>	1	boolean	2	integer 8	3	integer 16	4	integer 32	5	unsigned 8	6	unsigned 16	7	unsigned 32														
1	boolean																												
2	integer 8																												
3	integer 16																												
4	integer 32																												
5	unsigned 8																												
6	unsigned 16																												
7	unsigned 32																												
Output parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>value</td><td>data value that is read.</td></tr> </tbody> </table>	Parameter name	Description	value	data value that is read.																								
Parameter name	Description																												
value	data value that is read.																												
Return value	0 = success, not-0 = failure (please check error codes for details).																												
Example	Max torque of SDO writing and reading.																												
Details	Before execution, please do collection and scan the bus.																												

Instruction 253	ZAux_BusCmd_RtexWrite																
Original Format	int32 __stdcall ZAux_BusCmd_RtexWrite(ZMC_HANDLE handle, uint32 iaxis, uint32 ipara, float value);																
Description	Set RTEX parameter information.																
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link handle</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>ipara</td><td>Servo parameter No. <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Parameter type * 256 + parameter No. (Pr7.20 = 7 * 256 + 20)</td></tr> <tr> <td>2</td><td>Special parameter = 128, current parameter is written into EEPROM (now, value = 1)</td></tr> <tr> <td>3</td><td>Special parameter = 4000h + homing clamp mode + (set</td></tr> </tbody> </table> </td></tr> </tbody> </table>	Parameter name	Description	handle	Link handle	iaxis	Axis No.	ipara	Servo parameter No. <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Parameter type * 256 + parameter No. (Pr7.20 = 7 * 256 + 20)</td></tr> <tr> <td>2</td><td>Special parameter = 128, current parameter is written into EEPROM (now, value = 1)</td></tr> <tr> <td>3</td><td>Special parameter = 4000h + homing clamp mode + (set</td></tr> </tbody> </table>	Value	Description	1	Parameter type * 256 + parameter No. (Pr7.20 = 7 * 256 + 20)	2	Special parameter = 128, current parameter is written into EEPROM (now, value = 1)	3	Special parameter = 4000h + homing clamp mode + (set
Parameter name	Description																
handle	Link handle																
iaxis	Axis No.																
ipara	Servo parameter No. <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Parameter type * 256 + parameter No. (Pr7.20 = 7 * 256 + 20)</td></tr> <tr> <td>2</td><td>Special parameter = 128, current parameter is written into EEPROM (now, value = 1)</td></tr> <tr> <td>3</td><td>Special parameter = 4000h + homing clamp mode + (set</td></tr> </tbody> </table>	Value	Description	1	Parameter type * 256 + parameter No. (Pr7.20 = 7 * 256 + 20)	2	Special parameter = 128, current parameter is written into EEPROM (now, value = 1)	3	Special parameter = 4000h + homing clamp mode + (set								
Value	Description																
1	Parameter type * 256 + parameter No. (Pr7.20 = 7 * 256 + 20)																
2	Special parameter = 128, current parameter is written into EEPROM (now, value = 1)																
3	Special parameter = 4000h + homing clamp mode + (set																

			value * 256), use the homing clamp function of the driver.				
	value	Parameter values					
OP Parameters	/						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	RTEX parameters related information.						
Details	1. Servo Parameters						
	Parameter	Function		Value			
	Pr0.00	Set motor rotary direction		0: CW, 1: CCW			
	Pr0.01	Set control mode.		0: half closed-loop control			
	Pr0.08	The number of pulses for motor in one cycle.		0-8388608 (according to actual motor)			
	Pr0.09	Set gear ratio numerator		0-1073741824			
	Pr0.10	Set gear ratio denominator.		1-1073741824			
	Pr4.01	Set forward position limit signal		OFF commonly: 00818181h (8487297) ON commonly: 00010101h (65793)			
	Pr4.02	Set inverse position limit signal		OFF commonly: 00828282h (8553090) ON commonly: 00020202h (131586)			
	Pr4.03	Set Home signal		OFF commonly: 00A2A2A2h (10658466) ON commonly: 00222222h (2236962)			
	Torque	/		/			
	Pr0.13	First torque limit		0-500%			
	Pr5.21	Select torque limit, when in torque control, it is fixed as Pr0.13 (first torque limit)		As shown in below.			
		Set value	TL_SW=0		TL_SW=1		
			-	+	-	+	
			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	Second torque limit		0-500%			
	Pr5.25	Forward torque limit		0-500%			
	Pr5.26	Inverse torque limit		0-500%			
	Speed	/		/			
	Pr3.12	Set acceleration time		0~10000ms (reach 1000.r/min)			
	Pr3.13	Set deceleration time					
	Pr3.14	Set S accel & decel		0~1000ms			
	Pr3.17	Select speed limit.		Set			
Mode selection of speed limit		SL_SW					

	value during torque control.	value	0	1
		0	Pr3.21	
		1	Pr3.21	Pr3.22
		When it is set to 1, select according to RTEX communication "SL_SW".		
Pr3.21	Speed limit value 1	0~20000r/min		
Pr3.22	Speed limit value 2	0~20000r/min		

2. RTEX Communication Parameter

Pr7.20	Rtex communication period	-1: take effect Pr7.91 setting. 3: 0.5ms 6: 1.0ms
Pr7.21	Rtex command 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 clamp mode

typecode	Descriptions
\$50	Position clamp state monitor.
\$51	Position clamp 1 is ON.
\$52	Position clamp 2 is ON.
\$53	Position clamp 1 and 2 are ON.
\$54	Position clamp 1 is OFF.
\$58	Position clamp 2 is OFF.
\$5c	Position clamp 1 and 2 are OFF.

4. Set value

Set value = clamp 1 setting + (\$10 * clamp 2 setting)

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LATCH_SEL2				LATCH_SEL1			

Clamp 1 / 2 setting	Description
\$0	Phase Z signal trigger
\$1	Logic rising edge of EXT1
\$2	Logic rising edge of EXT2
\$3	Logic rising edge of EXT3
\$9	Logic falling edge of EXT1
\$10	Logic falling edge of EXT2
\$11	Logic falling edge of EXT3

Instruction 254	ZAux_BusCmd_RtexRead
Original Format	int32 __stdcall ZAux_BusCmd_RtexRead(ZMC_HANDLE handle, uint32 iaxis, uint32 ipara, float *value);

Description	Read RTEX parameter information.		
Input parameters	Parameter name	Description	
	handle	Link handle	
	iaxis	Axis No.	
	ipara	Servo parameter No.	
		Value	Description
		1	Parameter type * 256 + parameter No. (Pr7.20 = 7 * 256 + 20)
		2	Special parameter = 128, current parameter is written into EEPROM (now value = 1)
3	Special parameter = 4000h + homing clamp mode + (set value * 256), use the homing clamp function of the driver.		
value	Parameter values		
Output parameters	Parameter name	Description	
	value	Data that is read.	
Return value	0 = success, not-0 = failure (please check error codes for details).		
Example	RTEX parameters related information.		
Details	1. Servo Parameters		
	Parameter	Function	Value
	Pr0.00	Set motor rotary direction	0: CW, 1: CCW
	Pr0.01	Set control mode.	0: half closed-loop control
	Pr0.08	The number of pulses for motor in one cycle.	0-8388608 (according to actual motor)
	Pr0.09	Set gear ratio numerator	0-1073741824
	Pr0.10	Set gear ratio denominator.	1-1073741824
	Pr4.01	Set forward position limit signal	OFF commonly: 00818181h (8487297) ON commonly: 00010101h (65793)
	Pr4.02	Set inverse position limit signal	OFF commonly: 00828282h (8553090) ON commonly: 00020202h (131586)
	Pr4.03	Set Home signal	OFF commonly: 00A2A2A2h (10658466) ON commonly: 00222222h (2236962)
	Torque	/	/
	Pr0.13	First torque limit	0-500%
	Pr5.21	Select torque limit, when in torque control, it is fixed as Pr0.13 (first torque limit)	As shown in below.

	<table><tr><td rowspan="2">Set value</td><td colspan="2">TL_SW=0</td><td colspan="2">TL_SW=1</td></tr><tr><td>-</td><td>+</td><td>-</td><td>+</td></tr><tr><td>0, [1]</td><td colspan="4">Pr0.13</td></tr><tr><td>2</td><td>Pr5.22</td><td>Pr0.13</td><td>Pr5.22</td><td>Pr0.13</td></tr><tr><td>3</td><td colspan="2">Pr0.13</td><td colspan="2">Pr5.22</td></tr><tr><td>4</td><td>Pr5.22</td><td>Pr0.13</td><td>Pr5.22</td><td>Pr5.25</td></tr></table>	Set value	TL_SW=0		TL_SW=1		-	+	-	+	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
	Set value		TL_SW=0		TL_SW=1																									
		-	+	-	+																									
	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	Second torque limit		0-500%																											
Pr5.25	Forward torque limit		0-500%																											
Pr5.26	Inverse torque limit		0-500%																											
Speed	/		/																											
Pr3.12	Set acceleration time		0~10000ms (reach 1000.r/min)																											
Pr3.13	Set deceleration time																													
Pr3.14	Set S accel & decel		0~1000ms																											
Pr3.17	Select speed limit. Mode selection of speed limit value during torque control.		Set value	SL_SW																										
				0	1																									
			0	Pr3.21																										
			1	Pr3.21	Pr3.22																									
		When it is set to 1, select according to RTEX communication "SL_SW".																												
Pr3.21	Speed limit value 1		0~20000r/min																											
Pr3.22	Speed limit value 2		0~20000r/min																											

2. RTEX Communication Parameter

Pr7.20	Rtex communication period	-1: take effect Pr7.91 setting. 3: 0.5ms 6: 1.0ms
Pr7.21	Rtex command 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 clamp mode

typecode	Descriptions
\$50	Position clamp state monitor.
\$51	Position clamp 1 is ON.
\$52	Position clamp 2 is ON.
\$53	Position clamp 1 and 2 are ON.
\$54	Position clamp 1 is OFF.
\$58	Position clamp 2 is OFF.
\$5c	Position clamp 1 and 2 are OFF.

	4. Set value							
	Set value = clamp 1 setting + (\$10 * clamp 2 setting)							
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	LATCH_SEL2				LATCH_SEL1			
	Clamp 1 / 2 setting				Description			
	\$0				Phase Z signal trigger			
	\$1				Logic rising edge of EXT1			
	\$2				Logic rising edge of EXT2			
	\$3				Logic rising edge of EXT3			
	\$9				Logic falling edge of EXT1			
	\$10				Logic falling edge of EXT2			
	\$11				Logic falling edge of EXT3			

Instruction 255	ZAux_BusCmd_SDORedAxis		
Original Format	int32 __stdcall ZAux_BusCmd_SDORedAxis (ZMC_HANDLE handle, uint32 iaxis,uint32 index,uint32 subindex ,uint32 type ,int32 * value)		
Description	Read SDO through axis No.		
Input parameters	Parameter name	Description	
	handle	Link handle	
	iaxis	Axis No.	
	index	Object dictionary No.	
	subindex	Object dictionary 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	
Output parameters	Parameter name	Description	
	value	data value that is read.	
Return value	0 = success, not-0 = failure (please check error codes for details).		
Example	Int32 iValue; ZAux_BusCmd_SDORedAxis(0,0x6061,0,2,&iValue); //get axis 0's control mode.		
Details	- Before execution, please do collection and scan the bus. - It only can read the data dictionary that can be read.		

Instruction 256	ZAux_BusCmd_SDOWriteAxis	
Original	int32 __stdcall ZAux_BusCmd_SDOWriteAxis (ZMC_HANDLE	

Format	handle, uint32 iaxis, uint32 index,uint32 subindex ,uint32 type ,int32 value)																												
Description	Set SDO through axis No.																												
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link handle</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>index</td><td>Object dictionary No.</td></tr> <tr> <td>subindex</td><td>Object dictionary sub No.</td></tr> <tr> <td>type</td><td>Data type: <table> <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> </table> </td></tr> <tr> <td>value</td><td>data value that is written.</td></tr> </table>	Parameter name	Description	handle	Link handle	iaxis	Axis No.	index	Object dictionary No.	subindex	Object dictionary sub No.	type	Data type: <table> <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> </table>	1	boolean	2	integer 8	3	integer 16	4	integer 32	5	unsigned 8	6	unsigned 16	7	unsigned 32	value	data value that is written.
Parameter name	Description																												
handle	Link handle																												
iaxis	Axis No.																												
index	Object dictionary No.																												
subindex	Object dictionary sub No.																												
type	Data type: <table> <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> </table>	1	boolean	2	integer 8	3	integer 16	4	integer 32	5	unsigned 8	6	unsigned 16	7	unsigned 32														
1	boolean																												
2	integer 8																												
3	integer 16																												
4	integer 32																												
5	unsigned 8																												
6	unsigned 16																												
7	unsigned 32																												
value	data value that is written.																												
OP Parameters	/																												
Return value	0 = success, not-0 = failure (please check error codes for details).																												
Example	<pre>ZAux_BusCmd_SDOWriteAxis(0,0x6060,0,2,8); //axis 0 control mode is 8, position control</pre>																												
Details	- Before execution, please do collection and scan the bus. - It only can read the data dictionary that can be read.																												

Instruction 257	ZAux_BusCmd_NodePdoWrite																												
Original Format	int32 __stdcall ZAux_BusCmd_NodePdoWrite (ZMC_HANDLE handle, uint32 inode,uint32 index,uint32 subindex ,uint32 type ,int value)																												
Description	Set PDO.																												
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link handle</td></tr> <tr> <td>inode</td><td>Device No.</td></tr> <tr> <td>index</td><td>Object dictionary No.</td></tr> <tr> <td>subindex</td><td>Object dictionary sub No.</td></tr> <tr> <td>type</td><td>Data type: <table> <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> </table> </td></tr> <tr> <td>value</td><td>data value that is written.</td></tr> </table>	Parameter name	Description	handle	Link handle	inode	Device No.	index	Object dictionary No.	subindex	Object dictionary sub No.	type	Data type: <table> <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> </table>	1	boolean	2	integer 8	3	integer 16	4	integer 32	5	unsigned 8	6	unsigned 16	7	unsigned 32	value	data value that is written.
Parameter name	Description																												
handle	Link handle																												
inode	Device No.																												
index	Object dictionary No.																												
subindex	Object dictionary sub No.																												
type	Data type: <table> <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> </table>	1	boolean	2	integer 8	3	integer 16	4	integer 32	5	unsigned 8	6	unsigned 16	7	unsigned 32														
1	boolean																												
2	integer 8																												
3	integer 16																												
4	integer 32																												
5	unsigned 8																												
6	unsigned 16																												
7	unsigned 32																												
value	data value that is written.																												

OP Parameters	/
Return value	0 = success, not-0 = failure (please check error codes for details).
Example	<pre>ZAux_BusCmd_NodePdoWrite(0,0x6040,0,3,15); //Set device 0 control word to 15</pre>
Details	- For non-axis and IO devices, use this command to read and write PDO, such as power supply devices. - For axis and IO type devices that already can access PDO by axis parameters and IO commands can't use this anymore. - When the bus is turned on, it will automatically read current PDO list and the current value of the writable PDO. - The PDO list or the current value of the related data dictionary can be modified through SDO before calling "start bus". Please start it, then you can modify it, that is, use SOD_START to start to SAFEOP at first, then set the initial PDO state, and then start to OP.

Instruction 258	ZAux_BusCmd_NodePdoRead		
Original Format	int32 __stdcall ZAux_BusCmd_NodePdoRead(ZMC_HANDLE handle,uint32 inode,uint32 index,uint32 subindex ,uint32 type ,int* ivalue)		
Description	Read PDO.		
Input parameters	Parameter name	Description	
	handle	Link handle	
	inode	Device No.	
	index	Object dictionary No.	
	subindex	Object dictionary 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	
	value	data value that is written.	
Output parameters	Parameter name	Description	
	value	Read data value.	
Return value	0 = success, not-0 = failure (please check error codes for details).		
Example	Int32 iValue; ZAux_BusCmd_NodePdoRead(0,0x6041,0,3,&iValue); //get device 0's state word		
Details	For non-axis and IO devices, use this command to read and write PDO, such as power supply devices.		

	For axis and IO type devices that already can access PDO by axis parameters and IO commands can't use this anymore.

Instruction 259	ZAux_BusCmd_GetDriveTorque						
Original Format	int32 __stdcall ZAux_BusCmd_GetDriveTorque (ZMC_HANDLE handle , int iaxis,int *piValue);						
Description	Read current bus torque, it needs to set corresponding DRIVE_PROFILE type.						
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link handle</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </table>	Parameter name	Description	handle	Link handle	iaxis	Axis No.
Parameter name	Description						
handle	Link handle						
iaxis	Axis No.						
Output parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>piValue</td><td>Current torque.</td></tr> </table>	Parameter name	Description	piValue	Current torque.		
Parameter name	Description						
piValue	Current torque.						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	Get current torque of current bus drive						
Details	The correct ATYPE (set to 65/66/67) and DRIVE_PROFILE must be set before reading.						

Instruction 260	ZAux_BusCmd_SetMaxDriveTorque								
Original Format	int32 __stdcall ZAux_BusCmd_SetMaxDriveTorque (ZMC_HANDLE handle, int iaxis , int piValue);								
Description	Set max torque of current bus drive, it needs to set corresponding DRIVE_PROFILE type.								
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link handle</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>fValue</td><td>Set max torque limit</td></tr> </table>	Parameter name	Description	handle	Link handle	iaxis	Axis No.	fValue	Set max torque limit
Parameter name	Description								
handle	Link handle								
iaxis	Axis No.								
fValue	Set max torque limit								
OP Parameters	/								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	/								
Details	The correct ATYPE (set to 65/66/67) and DRIVE_PROFILE must be set before reading.								

Instruction 261	ZAux_BusCmd_GetMaxDriveTorque						
Original Format	int32 __stdcall ZAux_BusCmd_GetMaxDriveTorque (ZMC_HANDLE handle, int iaxis , int *piValue);						
Description	Get max torque of current bus torque, it needs to set corresponding DRIVE_PROFILE type.						
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link handle</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </table>	Parameter name	Description	handle	Link handle	iaxis	Axis No.
Parameter name	Description						
handle	Link handle						
iaxis	Axis No.						

Output parameters	Parameter name	Description
	piValue	Obtained max torque.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Get current torque of current bus drive	
Details	The correct ATYPE (set to 65/66/67) and DRIVE_PROFILE must be set before reading.	

Instruction 262	ZAux_Direct_SetDAC	
Original Format	int32 __stdcall ZAux_Direct_SetDAC (ZMC_HANDLE handle, int iaxis , float fValue);	
Description	The servo axis DA directly controls the torque. ps: for the bus driver, it needs to set corresponding DRIVE_PROFILE type, and ATYPE needs to be set as speed mode.	
Input parameters	Parameter name	Description
	handle	Link handle
	iaxis	Axis No.
	fValue	Set analog output value.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	- The unit: the scale of the DA module, 12-bit or 16-bit. - Under speed control, please check drive specific unit. - Under torque control, the unit is one thousandth, when it is 1000, which means 100% torque.	

Instruction 263	ZAux_Direct_GetDAC	
Original Format	int32 __stdcall ZAux_Direct_GetDAC (ZMC_HANDLE handle, int iaxis , float *fValue);	
Description	Read servo axis DA (the torque is controlled directly) value. ps: for the bus driver, it needs to set corresponding DRIVE_PROFILE type, and ATYPE needs to be set as speed mode.	
Input parameters	Parameter name	Description
	handle	Link handle
	iaxis	Axis No.
	fValue	Set analog output value.
Output parameters	Parameter name	Description
	fValue	Analog returned value.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	- The unit: the scale of the DA module, 12-bit or 16-bit. - Under speed control, please check drive specific unit.	

	Under torque control, the unit is one thousandth, when it is 1000, which means 100% torque.

Instruction 264	ZAux_Execute								
Original Format	int32 __stdcall ZAux_Execute(ZMC_HANDLE handle, const char *pszCommand, char *psResponse, uint32 uiResponseLength)								
Description	Send character string command to controller in buffer way (it will be automatically blocked when there is no buffer). Note: for ECI low series controllers, if "print" is added in Basic program, and "ZAux_Execute" command is also called in PC program, they will affect each other, causing some printed contents lost, therefore, don't recommend to add "print" in basic program, or try to stagger their calling time.								
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link handle</td></tr> <tr> <td>pszCommand</td><td>Command character string.</td></tr> <tr> <td>uiResponseLength</td><td>Returned character string length.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link handle	pszCommand	Command character string.	uiResponseLength	Returned character string length.
Parameter name	Description								
handle	Link handle								
pszCommand	Command character string.								
uiResponseLength	Returned character string length.								
Output parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>psResponse</td><td>Returned character string.</td></tr> </tbody> </table>	Parameter name	Description	psResponse	Returned character string.				
Parameter name	Description								
psResponse	Returned character string.								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	Use online command function								
Details	Upper computer calls unencapsulated Basic instruction.								

Instruction 265	ZAux_DirectCommand								
Original Format	int32 __stdcall ZAux_DirectCommand(ZMC_HANDLE handle, const char *pszCommand, char *psResponse, uint32 uiResponseLength)								
Description	Send character string commands to controller in direct way (it doesn't enter the buffer area, but some commands are not supported currently).								
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link handle</td></tr> <tr> <td>pszCommand</td><td>Command character string.</td></tr> <tr> <td>uiResponseLength</td><td>Returned character string length.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link handle	pszCommand	Command character string.	uiResponseLength	Returned character string length.
Parameter name	Description								
handle	Link handle								
pszCommand	Command character string.								
uiResponseLength	Returned character string length.								
Output parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>psResponse</td><td>Returned character string.</td></tr> </tbody> </table>	Parameter name	Description	psResponse	Returned character string.				
Parameter name	Description								
psResponse	Returned character string.								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	Use online command function								
Details	Upper computer calls unencapsulated Basic instruction.								

Instruction 266	ZAux_Pause				
Original Format	int32 __stdcall ZAux_Pause(ZMC_HANDLE handle)				
Description	Pause BASIC program that is running inside the controller.				
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link handle</td></tr> </table>	Parameter name	Description	handle	Link handle
Parameter name	Description				
handle	Link handle				
OP Parameters	/				
Return value	0 = success, not-0 = failure (please check error codes for details).				
Example	/				
Details	When the task is paused, please resume it at first, then it can keep executing.				

Instruction 267	ZAux_Resume				
Original Format	int32 __stdcall ZAux_Resume(ZMC_HANDLE handle)				
Description	Continue to run controller internal BASIC program.				
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link handle</td></tr> </table>	Parameter name	Description	handle	Link handle
Parameter name	Description				
handle	Link handle				
OP Parameters	/				
Return value	0 = success, not-0 = failure (please check error codes for details).				
Example	/				
Details	When the task is paused, please resume it at first, then it can keep executing.				

Instruction 268	ZAux_BasDown								
Original Format	int32 __stdcall ZAux_BasDown(ZMC_HANDLE handle,const char *Filename,uint32 run_mode)								
Description	Generate one single basic file (.bas) as ZAR file and download into controller for execution.								
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link handle</td></tr> <tr> <td>Filename</td><td>BAS file name that is with path.</td></tr> <tr> <td>run_mode</td><td>Download mode, 0-RAM, 1-ROM.</td></tr> </table>	Parameter name	Description	handle	Link handle	Filename	BAS file name that is with path.	run_mode	Download mode, 0-RAM, 1-ROM.
Parameter name	Description								
handle	Link handle								
Filename	BAS file name that is with path.								
run_mode	Download mode, 0-RAM, 1-ROM.								
OP Parameters	/								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	Download & edit controller program								
Details	.bas file, it can directly run Zmotion control card's Basic file.								

Instruction 269	ZAux_ZarDown
Original	int32 __stdcall ZAux_ZarDown(ZMC_HANDLE handle,const char

Format	*Filename,uint32 run_mode)	
Description	Download ZAR program into controller for operation.	
Input parameters	Parameter name	Description
	handle	Link handle
	Filename	ZAR file name that is with path.
	run_mode	Download mode, 0-RAM, 1-ROM.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Download & edit controller program	
Details	Zar file, it can directly run Zmotion motion control card program. It is the file that is generated after encryption.	

Instruction 270	ZAux_Direct_GetVariableInt	
Original Format	int32 __stdcall ZAux_Direct_GetVariableInt (ZMC_HANDLE handle , const char * pname, int * piValue);	
Description	Read Basic integer global variables, or axis parameters.	
Input parameters	Parameter name	Description
	handle	Link handle
	pname	Global variables name / axis parameter name of customized axis No., DPOS(0).
Output parameters	Parameter name	Description
	piValue	Returned value.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	<pre>int piValue; ZAux_Direct_GetVariableInt (handle , "DPOS (0) ", &piValue); //get axis 0 DPOS value in integer type</pre>	
Details	- Global variable: the name of the global variable that needs to be set in the RTSys software, and then set and read the value of the global variable through the host computer. Does not support setting and reading the Basic structures.	

Instruction 271	ZAux_Direct_GetVariablef	
Original Format	int32 __stdcall ZAux_Direct_GetVariablef (ZMC_HANDLE handle , const char * pname, float* pfValue);	
Description	Read Basic floating global variables, or axis parameters.	
Input parameters	Parameter name	Description
	handle	Link handle
	pname	Global variables name / axis parameter name with custom axis No., DPOS(0).
Output parameters	Parameter name	Description
	pfValue	Returned value.

Return value	0 = success, not-0 = failure (please check error codes for details).
Example	Float pfValue; ZAux_Direct_GetVariablef(handle, "DPOS (0)", &pfValue); // get axis 0 DPOS value in float type
Details	- Global variable: the name of the global variable that needs to be set in the RTSys software, and then set and read the value of the global variable through the host computer. Does not support setting and reading the Basic structures.

Instruction 272	ZAux_Direct_SetUserArray												
Original Format	int32 __stdcall ZAux_Direct_SetUserArray(ZMC_HANDLE handle, char* arrayname, int arraystart, int numes, float * pfValue);												
Description	Set Basic global array.												
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link handle</td></tr> <tr> <td>arrayname</td><td>Array name.</td></tr> <tr> <td>arraystart</td><td>Array starting element.</td></tr> <tr> <td>numes</td><td>The number of arrays</td></tr> <tr> <td>pfValue</td><td>The value that is set.</td></tr> </table>	Parameter name	Description	handle	Link handle	arrayname	Array name.	arraystart	Array starting element.	numes	The number of arrays	pfValue	The value that is set.
Parameter name	Description												
handle	Link handle												
arrayname	Array name.												
arraystart	Array starting element.												
numes	The number of arrays												
pfValue	The value that is set.												
OP Parameters	/												
Return value	0 = success, not-0 = failure (please check error codes for details).												
Details	- Global variable: the name of the global variable that needs to be set in the RTSys software, and then set and read the value of the global variable through the host computer. Does not support setting and reading the Basic structures.												

Instruction 273	ZAux_Direct_GetUserArray										
Original Format	int32 __stdcall ZAux_Direct_GetUserArray(ZMC_HANDLE handle, char* arrayname, int arraystart, int numes, float *pfValue);										
Description	Read Basic global array.										
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link handle</td></tr> <tr> <td>arrayname</td><td>Array name.</td></tr> <tr> <td>arraystart</td><td>Array starting element.</td></tr> <tr> <td>numes</td><td>The number of arrays</td></tr> </table>	Parameter name	Description	handle	Link handle	arrayname	Array name.	arraystart	Array starting element.	numes	The number of arrays
Parameter name	Description										
handle	Link handle										
arrayname	Array name.										
arraystart	Array starting element.										
numes	The number of arrays										
Output parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>pfValue</td><td>Read array element's values.</td></tr> </table>	Parameter name	Description	pfValue	Read array element's values.						
Parameter name	Description										
pfValue	Read array element's values.										
Return value	0 = success, not-0 = failure (please check error codes for details).										
Details	- Global variable: the name of the global variable that needs to be set in the RTSys software, and then set and read the value of the global variable through the host computer. Does not support setting and reading the Basic structures.										

Instruction 274	ZAux_Direct_SetUserVar								
Original Format	int32 __stdcall ZAux_Direct_SetUserVar(ZMC_HANDLE handle,char *varname , float pfValue);								
Description	Set Basic customized global variables.								
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link handle</td></tr> <tr> <td>varname</td><td>Variables' name.</td></tr> <tr> <td>pfValue</td><td>The value that is set.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link handle	varname	Variables' name.	pfValue	The value that is set.
Parameter name	Description								
handle	Link handle								
varname	Variables' name.								
pfValue	The value that is set.								
OP Parameters	/								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	Download & edit controller program								
Details	- Global variable: the name of the global variable that needs to be set in the RTSys software, and then set and read the value of the global variable through the host computer. Does not support setting and reading the Basic structures.								

Instruction 275	ZAux_Direct_GetUserVar						
Original Format	int32 __stdcall ZAux_Direct_GetUserVar(ZMC_HANDLE handle,char *varname , float *pfValue);						
Description	Read Basic customized global variables.						
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Handle</td><td>Link handle</td></tr> <tr> <td>varname</td><td>Variables' name.</td></tr> </tbody> </table>	Parameter name	Description	Handle	Link handle	varname	Variables' name.
Parameter name	Description						
Handle	Link handle						
varname	Variables' name.						
Output parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>pfValue</td><td>Returned value</td></tr> </tbody> </table>	Parameter name	Description	pfValue	Returned value		
Parameter name	Description						
pfValue	Returned value						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	Download & edit controller program						
Details	- Global variable: the name of the global variable that needs to be set in the RTSys software, and then set and read the value of the global variable through the host computer. Does not support setting and reading the Basic structures.						

Instruction 276	ZAux_SetTimeOut						
Original Format	int32 __stdcall ZAux_SetTimeOut(ZMC_HANDLE handle, uint32 timems)						
Description	Command's delay waiting time.						
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link handle</td></tr> <tr> <td>timems</td><td>Waiting time MS.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link handle	timems	Waiting time MS.
Parameter name	Description						
handle	Link handle						
timems	Waiting time MS.						
OP Parameters	/						
Return value	0 = success, not-0 = failure (please check error codes for details).						

Instruction 277	ZAux_TransStringToInt	
Original Format	int32 __stdcall ZAux_TransStringToInt (const char * pstringin, int inues, int * pivlaue);	
Description	Convert character string into int.	
Input parameters	Parameter name	Description
	pstringin	Character string of data
	inumes	The number of data converting.
Output parameters	Parameter name	Description
	pivalue	Converted data
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	/	

Instruction 278	ZAux_TransStringtoFloat	
Original Format	int32 __stdcall ZAux_TransStringtoFloat (const char * pstringin, int inues, int * pfvlaue);	
Description	Convert character string into float.	
Input parameters	Parameter name	Description
	pstringin	Character string of data
	inumes	The number of data converting.
Output parameters	Parameter name	Description
	pfvalue	Converted data
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	/	

Instruction 279	*PZMCAutoUpCallBack	
Original Format	typedef void (*PZMCAutoUpCallBack)(ZMC_HANDLE handle, int32 itypecode, int32 idatalength, uint8 *pdata);	
Description	Call back function format of active reporting.	
Input parameters	Parameter name	Description
	handle	Link handle
	itypecode	Upload type code
	idatalength	Data length
	pdata	Data pointer
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Active reporting.	
Details	/	

Instruction 280	ZAux_SetAutoUpCallBack						
Original Format	int32 __stdcall ZAux_SetAutoUpCallBack (ZMC_HANDLE handle, PZMCAutoUpCallBack pcallback);						
Description	Controller reports automatically.						
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link handle</td></tr> <tr> <td>pcallback</td><td>Function format</td></tr> </tbody> </table>	Parameter name	Description	handle	Link handle	pcallback	Function format
Parameter name	Description						
handle	Link handle						
pcallback	Function format						
OP Parameters	/						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	Automatically report						
Details	Controller auto-report function: controller sends messages to PC.						

Instruction 281	ZAux_CycleUpEnable										
Original Format	int32 __stdcall ZAux_CycleUpEnable(ZMC_HANDLE handle, uint32 cycleindex, float fintervalms, const char* psetesname);										
Description	Enable cyclic reporting.										
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link handle</td></tr> <tr> <td>cycleindex</td><td>Report channel No., 0-max value (PS: ECI series max values = 0, ZMC4XX series max values = 1)</td></tr> <tr> <td>fintervalms</td><td>Reporting gap time, the unit is ms. And it can't be lower than controller bus communication period.</td></tr> <tr> <td>psetesname</td><td>Select reporting parameters: Parameter 1: for example: IDLE: report IDLE of default axis. Parameter 2(index): for example: IDLE(1): report IDLE of axis 1. Parameter 3(index,numes): for example: IDLE(0,3): report IDLE of axis 0-2.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link handle	cycleindex	Report channel No., 0-max value (PS: ECI series max values = 0, ZMC4XX series max values = 1)	fintervalms	Reporting gap time, the unit is ms. And it can't be lower than controller bus communication period.	psetesname	Select reporting parameters: Parameter 1: for example: IDLE: report IDLE of default axis. Parameter 2(index): for example: IDLE(1): report IDLE of axis 1. Parameter 3(index,numes): for example: IDLE(0,3): report IDLE of axis 0-2.
Parameter name	Description										
handle	Link handle										
cycleindex	Report channel No., 0-max value (PS: ECI series max values = 0, ZMC4XX series max values = 1)										
fintervalms	Reporting gap time, the unit is ms. And it can't be lower than controller bus communication period.										
psetesname	Select reporting parameters: Parameter 1: for example: IDLE: report IDLE of default axis. Parameter 2(index): for example: IDLE(1): report IDLE of axis 1. Parameter 3(index,numes): for example: IDLE(0,3): report IDLE of axis 0-2.										
OP Parameters	/										
Return value	0 = success, not-0 = failure (please check error codes for details).										
Example	Periodically report										
Detail 1	- The number of bytes uploaded by each channel cannot exceed 1000 bytes. - The space occupied by each parameter is different. One IO only occupies one bit. For ZMC4XX series products, they use the double data type by default. For ZMC3XX series and XPLC series products, they use the float data type by default. - It can upload periodically only when Ethernet is connected. - Both zmotion.dll and controller firmware should support this										

	function. It can reduce PC active polling time by setting parameters that are read frequently in advance to make it report actively and periodically.
Detail 2	Valid Parameters: ✓ VP_SPEED (command current speed) ✓ MPOS (feedback / measurement position) ✓ DPOS (command / demand position), ✓ MSPEED (feedback current speed) ✓ AXISSTATUS (axis status) ✓ FE (following error) ✓ MOVES_BUFFERED (current buffer number) ✓ IDLE (motion state) ✓ MOVE_CURMARK (current movement mark) ✓ MARK (latch trigger) ✓ VECTOR_BUFFERED (buffer remain distance) ✓ MTYPE (current motion type) ✓ DRIVER_FE (driver's current follow-up error) ✓ DRIVER_TORCH (driver current torque) ✓ DRIVER_STATUS (driver status) ✓ N (input), OUT (output) ✓ AIN (analog input), AOUT (analog output), ✓ TABLE (register). Below parameters are also supported for ZMC4XX series: ✓ REMAIN_BUFFER1: the remaining number of linear motion buffers, in the case of MTYPE=1.

Instruction 282	ZAux_CycleUpDisable						
Original Format	int32 __stdcall ZAux_CycleUpDisable(ZMC_HANDLE handle, uint32 cycleindex);						
Description	Close periodic-reporting enable.						
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link handle</td></tr> <tr> <td>cycleindex</td><td>Report channel No., 0-max value-1</td></tr> </table>	Parameter name	Description	handle	Link handle	cycleindex	Report channel No., 0-max value-1
Parameter name	Description						
handle	Link handle						
cycleindex	Report channel No., 0-max value-1						
OP Parameters	/						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	Periodically report						
Detail	- Both zmotion.dll and controller firmware should support this function. - It can reduce PC active polling time by setting parameters that are read frequently in advance to make it report actively and periodically.						

Instruction 283	ZAux_CycleUpGetRecvTimes
Original Format	uint32 __stdcall ZAux_CycleUpGetRecvTimes(ZMC_HANDLE handle, uint32 cycleindex);
Description	Obtain the number of packages that is received by periodical

	reporting.	
Input parameters	Parameter name	Description
	handle	Link handle
	cycleindex	Report channel No., 0-max value-1
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Periodically report	
Detail	● Both zmotion.dll and controller firmware should support this function. ● Exceed overflow, for debugging.	

Instruction 284	ZAux_CycleUpForceOnce	
Original Format	int32 __stdcall ZAux_CycleUpForceOnce(ZMC_HANDLE handle, uint32 cycleindex);	
Description	Report once compulsively.	
Input parameters	Parameter name	Description
	handle	Link handle
	cycleindex	Report channel No., 0-max value-1
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Periodically report	
Detail	It is called when idle may be wrong, and after motion command.	

Instruction 285	ZAux_CycleUpReadBuff	
Original Format	uint32 __stdcall ZAux_CycleUpReadBuff(ZMC_HANDLE handle, uint32 cycleindex, const char *psetname, uint32 isetindex, double *pvalue);	
Description	Read contents from periodical reporting, the data type is floating.	
Input parameters	Parameter name	Description
	handle	Link handle
	cycleindex	Report channel No., 0-max value-1
	psetname	Parameter name
	isetindex	Parameter No.
Output parameters	Parameter name	Description
	pvalue	Read parameter content from periodical reporting.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Periodically report	
Detail	- Both zmotion.dll and controller firmware should support this. - It can reduce PC active polling time by setting parameters that are read frequently in advance to make it report actively and periodically.	

Instruction 286	ZAux_CycleUpReadBuffInt	
Original Format	uint32 __stdcall ZAux_CycleUpReadBuffInt(ZMC_HANDLE handle, uint32 cycleindex, const char *psetname, uint32 isetindex, int32*pvalue);	
Description	Read contents from periodical reporting, the data type is integer.	
Input parameters	Parameter name	Description
	handle	Link handle
	cycleindex	Report channel No., 0-max value-1
	psetname	Parameter name
	isetindex	Parameter No.
Output parameters	Parameter name	Description
	pvalue	Read parameter content in periodical reporting.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Periodically report	
Detail	- Both zmotion.dll and controller firmware should support this function. - It can reduce PC active polling time by setting parameters that are read frequently in advance to make it report actively and periodically.	

Instruction 287	ZAux_SetTraceFile	
Original Format	int32 __stdcall ZAux_SetTraceFile(int bifTofile,const char *pFilePathName)	
Description	Print out the called PC command to the specified file.	
Input parameters	Parameter name	Description
	bifTofile	Print output mode. 0- OFF 1- Only error command is output 2- Only motion and parameter configuration commands are output 3- All commands are output
	pFilePathName	File name is with absolute path. This is reserved parameter, which means it doesn't take effect currently, any character string can be filled in.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Debug and print information	
Detail	- The file is saved under \ZmotionLog\ of the current program path, and it is saved as a log. Note: this command is only used during debugging. When not in use, remember to turn off the print output mode, otherwise	

it will affect the communication efficiency.

Instruction 288	ZAux_Trigger	
Original Format	int32 __stdcall ZAux_Trigger(ZMC_HANDLE handle)	
Description	Trigger the oscilloscope (it is valid in firmware after 150723).	
Input parameters	Parameter name	Description
	handle	Link handle
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Oscilloscope Usage	
Detail	It needs to use RTSys software to open oscilloscope.	

Instruction 289	ZAux_Direct_GetCreep	
Original Format	int32 __stdcall ZAux_Direct_GetCreep(ZMC_HANDLE handle, int iaxis, float *pfValue);	
Description	Read 2 times homing creep speed.	
Input parameters	Parameter name	Description
	handle	Link handle
	iaxis	Axis No.
Output parameters	Parameter name	Description
	*pfValue	Returned creep speed.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	float fValue; ZAux_Direct_GetCreep(handle, 0, &fValue); //read axis 0's homing creep speed	
Detail	During the homing, after finding the origin, it is the creep speed when leave the origin.	

Instruction 290	ZAux_Direct_Base	
Original Format	int32 __stdcall ZAux_Direct_Base(ZMC_HANDLE handle, int imaxaxes, int *piAxislist)	
Description	Set controller's default axis No.	
Input parameters	Parameter name	Description
	handle	Link handle
	imaxaxes	The number of participated axes
	puAxislist	Axis No. array
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Detail	/	

Instruction 291	ZAux_Direct_Backlash	
Original Format	int32 __stdcall ZAux_Direct_Backlash(ZMC_HANDLE handle, int iaxis, bool enable, float dist, float speed, float accel)	
Description	Set axis backlash, but it is invalid in expanded axis.	
Input parameters	Parameter name	Description
	handle	Link handle
	iaxis	Axis No. that is participated in motion.
	enable	Enable switch: 0-OFF, 1-ON
	dist	Compensation distance, unit: units
	speed	Compensation speed of backlash
	accel	Compensation acceleration of backlash
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Backlash Compensation	
Detail	/	

Instruction 292	ZAux_LocalRTManage	
Original Format	uint32 __stdcall ZAux_LocalRTManage(int32 isel, int32 icode)	
Description	Control (start & stop) RT soft kernel console.	
Input parameters	Parameter name	Description
	isel	Reserved
	icode	1 – start RT, 2 – stop RT
OP Parameters	/	
Return value	Error code	
Example	start & stop RT.	
Detail	/	

Instruction 293	ZAux_LocalRTState	
Original Format	uint32 __stdcall ZAux_GetLocalRTState(int32 isel)	
Description	Read the RT soft kernel console state.	
Input parameters	Parameter name	Description
	isel	Reserved
OP Parameters	0—no motion, 1 – running, others – error codes	
Return value	Error code	
Example	start & stop RT.	
Detail	/	

Instruction 294	ZAux_Direct_MoveModify2	
Original	int32 __stdcall ZAux_Direct_MoveModify2(ZMC_HANDLE handle, int	

Format	imaxaxes, int* piAxislist, float* pfDisance)										
Description	Forcefully stop the original motion, then position the device at the new target location using the original speed and acceleration. When there is no preceding motion, it works the same as MOVEABS, but the motion on each axis is independent and does not enter the motion buffer. For continuous interpolation, MOVEMODIFY2 will disrupt the speed continuity. And for multi-axis motion, it is not necessarily linear interpolation when MOVEMODIFY2 is used.										
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Reserved</td></tr> <tr> <td>imaxaxes</td><td>Axis numbers</td></tr> <tr> <td>piAxislist</td><td>Axis No. list</td></tr> <tr> <td>pfDistance</td><td>Absolute distance list</td></tr> </table>	Parameter name	Description	handle	Reserved	imaxaxes	Axis numbers	piAxislist	Axis No. list	pfDistance	Absolute distance list
Parameter name	Description										
handle	Reserved										
imaxaxes	Axis numbers										
piAxislist	Axis No. list										
pfDistance	Absolute distance list										
OP Parameters	/										
Return value	Error code										
Example	/										
Detail	Same as ZAux_Direct_MoveModify. ZAux_Direct_MoveModify2: multi-axis ZAux_Direct_MoveModify: single-axis										

Instruction 295	ZAux_Execute_GetMoveTime								
Original Format	int32 __stdcall ZAux_Execute_GetMoveTime(ZMC_HANDLE handle, int iaxis, float *fValue)								
Description	Read motion remain time.								
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Connection handle</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>fValue</td><td>Motion remain time, the unit is ms</td></tr> </table>	Parameter name	Description	handle	Connection handle	iaxis	Axis No.	fValue	Motion remain time, the unit is ms
Parameter name	Description								
handle	Connection handle								
iaxis	Axis No.								
fValue	Motion remain time, the unit is ms								
OP Parameters	/								
Return value	Error code								
Example	<pre> ZAux_Direct_SetAxisMode(handle, 0, 2^6); //set axismode ZAux_Direct_SetVpMode(handle,0,7); //set vpmode ZAux_Direct_Single_MoveAbs(handle,0,1000); //set the motion, 1000 float fValue=0; ZAux_Execute_GetMoveTime(handle, 0, &fValue);//get remain time </pre>								
Detail	- The "Execute" encapsulation command should not be used frequently. - It only works when bit 6 of AxisMode is 1, VpMode = 7.								

Instruction 296	ZAux_Execute_GetPreCalTime
Original Format	int32 __stdcall ZAux_Execute_GetPreCalTime(ZMC_HANDLE handle,float dis, float *fValue)

Description	Estimate the time it takes for point motion, in milliseconds.	
Input parameters	Parameter name	Description
	handle	Connection handle
	dis	Axis motion distance
	fValue	Estimate the point motion time (ms).
OP Parameters	/	
Return value	Error code	
Example	<pre> ZAux_Direct_SetAxisMode(handle, 0, 2^6); //set axismode ZAux_Direct_SetVpMode(handle,0,7); //set vpmode ZAux_Direct_Single_MoveAbs(handle,0,1000); //set motion, 1000 float fValue=0; ZAux_Execute_GetPreCalTime (handle, 0, &fValue);//get the time </pre>	
Detail	- The "Execute" encapsulation command should not be used frequently. - It only works when bit 6 of AxisMode is 1, VpMode = 7.	

Instruction 297	ZAux_Direct_SetVpMode	
Original Format	int32 __stdcall ZAux_Direct_SetVpMode(ZMC_HANDLE handle, int iaxis, int mode)	
Description	Select acceleration and deceleration curve types.	
Input parameters	Parameter name	Description
	handle	Connection handle
	iaxis	Axis No.
	mode	Select accel & decel curve types:
	0	Default value, use SRAMP to set S curve.
	4	Max acceleration at start-up, acceleration gradually decreases to 0 upon reaching top speed.
	6	New type: SS curve, jerk continuous curve, it will add 87% deceleration time compared to T-shape deceleration. Under this mode, it uses mode 0 when accelerating, then this is valid when decelerating, which is convenient for continuous small-segment interpolation.
	7	New type: SS curve, jerk continuous curve. If parameters are modified dynamically, or continuous interpolation is opened, it may cause discontinuous jerk. And now it

		will switch to mode 0, so set a reasonable value for SRAMP.	
Return value	Error code		

Instruction 298	ZAux_Direct_SetAxisMode	
Original Format	int32 __stdcall ZAux_Direct_SetAxisMode(ZMC_HANDLE handle, int iaxis, int mode)	
Description	It is valid in firmware version after 20170616.	
Input parameters	Parameter name	Description
	handle	Connection handle
	iaxis	Axis No.
	mode	set mode of "connect (motion hold)", enter the value: ➤ BIT1=1: prevent limit & soft limit from causing CONNECT to exit. ➤ BIT1=0: after meets the position limit, the "CONNECT" connection of master axis and slave axis will break, and after cleared the position limit alarm, only operate the spindle, and the slave axis will not follow. ➤ BIT1=1: after meets the position limit, the "CONNECT" connection will hold, and after cleared the position limit alarm, the slave axis still is connected. ➤ BIT5=0: (default), when the spindle is with encoder, it will track MPOS priority. ➤ BIT5=1: set the DPOS of the spindle that tracks gear or cam motion, the following commands are involved: CAMBOX, CONNECT, MOVELINK, MOVESLINK, MOVESYNC, and HW_PSWITCH2. ➤ BIT6: 0 (default), 1 -- use the algorithm that can predict the exact time, now it is currently effective when VP_MODE=7. ➤ BIT7: 0 (default), 1 -- when MOVEMODIFY is used, try to avoid inverse running, it will use a greater deceleration.
OP Parameters	/	

Return value	Error code
Example	/
Details	/

Instruction 299	ZAux_Direct_SetTableDouble										
Original Format	int32 __stdcall ZAux_Direct_SetTableDouble(ZMC_HANDLE handle,int tabstart, int numes, double *pfValue)										
Description	Write into TABLE.										
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Connection handle</td></tr> <tr> <td>tabstart</td><td>TABLE starting No.</td></tr> <tr> <td>numes</td><td>The number of writings</td></tr> <tr> <td>pfValue</td><td>The data value</td></tr> </table>	Parameter name	Description	handle	Connection handle	tabstart	TABLE starting No.	numes	The number of writings	pfValue	The data value
Parameter name	Description										
handle	Connection handle										
tabstart	TABLE starting No.										
numes	The number of writings										
pfValue	The data value										
OP Parameters	/										
Return value	Error code										
Example	Double value=1.01 ZAux_Direct_SetTableDouble(handle,0, 1, &value)										
Details	/										

Instruction 300	ZAux_Direct_GetTableDouble										
Original Format	int32 __stdcall ZAux_Direct_GetTableDouble(ZMC_HANDLE handle, int tabstart, int numes, double *pfValue)										
Description	Read TABLE, it can read multiple at one time.										
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Connection handle</td></tr> <tr> <td>tabstart</td><td>TABLE starting No.</td></tr> <tr> <td>numes</td><td>The number of readings</td></tr> <tr> <td>pfValue</td><td>It must assign the space when there is not one single TABLE.</td></tr> </table>	Parameter name	Description	handle	Connection handle	tabstart	TABLE starting No.	numes	The number of readings	pfValue	It must assign the space when there is not one single TABLE.
Parameter name	Description										
handle	Connection handle										
tabstart	TABLE starting No.										
numes	The number of readings										
pfValue	It must assign the space when there is not one single TABLE.										
OP Parameters	/										
Return value	Error code										
Example	Double value=1.01 ZAux_Direct_GetTableDouble(handle,0, 1, &value)										
Details	/										

Instruction 301	ZAux_Direct_MSphericalABSSp
Original Format	int32 __stdcall ZAux_Direct_MSphericalABSSp(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float fend1, float fend2, float fend3, float fcenter1, float fcenter2, float fcenter3, int imode, float fcenter4, float fcenter5)
Description	Space arc + helical interpolation, absolute motion, and SP command is used.

It is valid in controller firmware version after 20130901.																											
Input parameters	<table><tr><th>Parameter name</th><th>Description</th></tr><tr><td>handle</td><td>Link mark</td></tr><tr><td>imaxaxes</td><td>Total axes</td></tr><tr><td>piAxislist</td><td>Axis No. list</td></tr><tr><td>fend1</td><td>1st axis motion distance 1</td></tr><tr><td>fend2</td><td>2nd axis motion distance 1</td></tr><tr><td>fend3</td><td>3rd axis motion distance 1</td></tr><tr><td>fcenter1</td><td>1st axis motion distance 2</td></tr><tr><td>fcenter2</td><td>2nd axis motion distance 2</td></tr><tr><td>fcenter3</td><td>3rd axis motion distance 2</td></tr><tr><td>imode</td><td>Specify previous parameters' functions: 0: use 3 points (current point + middle point + end point) to draw the arc. "distance 1": the end point distance. "distance 2": the middle point distance. 1: use current point + circle center + end point to draw the smallest arc. "distance 1": the end point distance. "distance 2": the circle center distance. 2: use 3 points (current point + middle point + end point) to draw the circle. "distance 1": the end point distance. "distance 2": the middle point distance. 3: use current point + circle center + end point to draw the circle, move the smallest arc. "distance 1": the end point distance "distance 2": the circle center distance</td></tr><tr><td>Fcenter4</td><td>4th axis helical distance</td></tr><tr><td>Fcenter5</td><td>5th axis helical distance</td></tr></table>	Parameter name	Description	handle	Link mark	imaxaxes	Total axes	piAxislist	Axis No. list	fend1	1 st axis motion distance 1	fend2	2 nd axis motion distance 1	fend3	3 rd axis motion distance 1	fcenter1	1 st axis motion distance 2	fcenter2	2 nd axis motion distance 2	fcenter3	3 rd axis motion distance 2	imode	Specify previous parameters' functions: 0: use 3 points (current point + middle point + end point) to draw the arc. "distance 1": the end point distance. "distance 2": the middle point distance. 1: use current point + circle center + end point to draw the smallest arc. "distance 1": the end point distance. "distance 2": the circle center distance. 2: use 3 points (current point + middle point + end point) to draw the circle. "distance 1": the end point distance. "distance 2": the middle point distance. 3: use current point + circle center + end point to draw the circle, move the smallest arc. "distance 1": the end point distance "distance 2": the circle center distance	Fcenter4	4 th axis helical distance	Fcenter5	5 th axis helical distance
	Parameter name	Description																									
	handle	Link mark																									
	imaxaxes	Total axes																									
	piAxislist	Axis No. list																									
	fend1	1 st axis motion distance 1																									
	fend2	2 nd axis motion distance 1																									
	fend3	3 rd axis motion distance 1																									
	fcenter1	1 st axis motion distance 2																									
	fcenter2	2 nd axis motion distance 2																									
	fcenter3	3 rd axis motion distance 2																									
imode	Specify previous parameters' functions: 0: use 3 points (current point + middle point + end point) to draw the arc. "distance 1": the end point distance. "distance 2": the middle point distance. 1: use current point + circle center + end point to draw the smallest arc. "distance 1": the end point distance. "distance 2": the circle center distance. 2: use 3 points (current point + middle point + end point) to draw the circle. "distance 1": the end point distance. "distance 2": the middle point distance. 3: use current point + circle center + end point to draw the circle, move the smallest arc. "distance 1": the end point distance "distance 2": the circle center distance																										
Fcenter4	4 th axis helical distance																										
Fcenter5	5 th axis helical distance																										
OP Parameters	/																										
Return value	Error code																										
Example	space arc																										
Details	/																										

Instruction 302	ZAux_Direct_GetBusAxisEnable
Original Format	int32 __stdcall ZAux_Direct_GetBusAxisEnable(ZMC_HANDLE handle, int iaxis, int *piValue)
Description	Read bus axis enable state.

Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
	pfValue	Returned enable state
OP Parameters	/	
Return value	Error code	
Example	<pre>int piValue=0; int ret=ZAux_Direct_GetBusAxisEnable(handle, 0, &piValue);</pre>	
Details	Directly obtain state word's enable state, not the axisEnable value.	

Instruction 303	ZAux_Scope	
Original Format	int32 __stdcall ZAux_Scope(ZMC_HANDLE handle,int iEnable,int iPeriod,int iStartTab,int iStopTab, char * ParaNamep)	
Description	Capture the data	
Input parameters	Parameter name	Description
	handle	Link mark
	iEnable	Whether to enable
	iPeriod	Sampling period, take controller period as the unit
	iStartTab	TABLE starting position
	iStopTab	TABLE end position
	iParaNum	Data source (data type) numbers, max: 8
	ParaNamep	Data sour character name
OP Parameters	/	
Return value	Error code	
Example	Data sampling	
Details	/	

Instruction 304	ZAux_GetScopeLength	
Original Format	int32 __stdcall ZAux_GetScopeLength(ZMC_HANDLE handle,int *ilen)	
Description	Get the oscilloscope's sampling length	
Input parameters	Parameter name	Description
	handle	Link mark
Output Parameters	Parameter name	Description
	ilen	Data length
Return value	Error code	
Example	Data sampling	
Details	/	

Instruction 305	ZAux_Direct_MoveOpDelay								
Original Format	int32 __stdcall ZAux_Direct_MoveOpDelay(ZMC_HANDLE handle, float uTime, int nAxis)								
Description	Open / close OP in advance.								
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>uTime</td><td>Delay time, ms</td></tr> <tr> <td>nAxis</td><td>Axis No.</td></tr> </table>	Parameter name	Description	handle	Link mark	uTime	Delay time, ms	nAxis	Axis No.
Parameter name	Description								
handle	Link mark								
uTime	Delay time, ms								
nAxis	Axis No.								
OP Parameters	/								
Return value	Error code								
Example	<pre> ZAux_Direct_SetParam(handle," AXIS_ZSET",0, 19); //open precision output ZAux_Direct_MoveOpDelay(handle, 100, 0); //output after 100ms (delay) ZAux_Direct_Single_Move(handle, 0, 100); //single-axis motion, 100 ZAux_Direct_MoveOp(handle,0,0, 1); //axis 0 moves 100 in forward, then set OUT 0 as 1 after 100ms. ZAux_Direct_Single_Move(handle, 0, 100); //single-axis motion, 100 </pre>								
Details	- It is valid in motion controllers that support HW function, and HW port should be used. - When MOVE_OP precision function is used, it can adjust real OP triggering time, in ms. And it supports decimal part. The real longest delay time is 100ms. Negative value means it can open OP in advance. Generally, for stepper, 2ms early output, for servo, 20ms early output.								

Instruction 306	ZAux_DownMem3FileRam												
Original Format	int32 __stdcall ZAux_DownMem3FileRam(ZMC_HANDLE handle, const char* pBuffer, uint32 buffsize, char* File3Name, int32 ifile3num)												
Description	Package the character string motion, and send it for execution.												
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>pBuffer</td><td>Motion character string</td></tr> <tr> <td>buffsize</td><td>Character string size</td></tr> <tr> <td>File3Name</td><td>Generated file 3 file name</td></tr> <tr> <td>ifile3num</td><td>Select which file 3</td></tr> </table>	Parameter name	Description	handle	Link mark	pBuffer	Motion character string	buffsize	Character string size	File3Name	Generated file 3 file name	ifile3num	Select which file 3
Parameter name	Description												
handle	Link mark												
pBuffer	Motion character string												
buffsize	Character string size												
File3Name	Generated file 3 file name												
ifile3num	Select which file 3												
OP Parameters	/												
Return value	Error code												
Example	File 3 Usage												
Details	/												

Instruction 307	ZAux_Run3FileRam
Original Format	int32 __stdcall ZAux_Run3FileRam(ZMC_HANDLE handle, int32 ifile3num, int32 itasknum)

Description	Run the file 3 motion.	
Input parameters	Parameter name	Description
	handle	Link mark
	ifile3num	File 3 file No.
	itaskum	Motion task No.
OP Parameters	/	
Return value	Error code	
Example	File 3 Usage	
Details	/	

Instruction 308	ZAux_Down3FileRom	
Original Format	int32 __stdcall ZAux_Down3FileRom(ZMC_HANDLE handle, const char* pfilename, const char* pfilenameinControl, int32 ifile3num)	
Description	Download file 3 file into RAM	
Input parameters	Parameter name	Description
	handle	Link mark
	pfilename	PC file name: PC path + file name (Z3P / NC)
	pfilenameinControl	Filename (Z3P/NC) that is saved into controller.
	ifile3num	File 3 file No.
OP Parameters	/	
Return value	Error code	
Example	File 3 Usage	
Details	/	

Instruction 309	ZAux_3FileRamGetRemainSpace	
Original Format	int32 __stdcall ZAux_3FileRamGetRemainSpace(ZMC_HANDLE handle, int32 ifile3num, uint32 *premainbyte)	
Description	Dynamically download file 3 file, read writable space.	
Input parameters	Parameter name	Description
	handle	Link mark
	ifile3num	File 3 file No.
OP Parameters	Parameter name	Description
	premainbyte	Byte numbers
Return value	Error code	
Example	File 3 Usage	
Details	/	

Instruction 310	ZAux_3FileRamDownBegin	
Original	int32 __stdcall ZAux_3FileRamDownBegin(ZMC_HANDLE handle,	

Format	int32 ifile3num,uint32 *premainbyte,char*File3Name)	
Description	Dynamically download file 3 file, start "downloading".	
Input parameters	Parameter name	Description
	handle	Link mark
	ifile3num	File 3 file No.
	File3Name	Set Z3P file name, it won't run automatically, so please call "ZAux_Run3FileRam".
OP Parameters	Parameter name	Description
	premainbyte	Remain control's byte numbers
Return value	Error code	
Example	File 3 Usage	
Details	/	

Instruction 311	ZAux_3FileRamDownPart	
Original Format	int32 __stdcall ZAux_3FileRamDownPart(ZMC_HANDLE handle,int32 ifile3num, const char* pBuffer, uint32 bufsize, uint32 *premainbyte)	
Description	Dynamically download Z3P file, each download must contain a complete single line or multiple lines. If the download is incomplete, it ends with a newline character (\n). Once the download is complete, it ends with a null terminator (\0). To restart the program by calling ZMC_Run3FileRam again.	
Input parameters	Parameter name	Description
	handle	Link mark
	ifile3num	File 3 file No.
	File3Name	Set Z3P file name, it won't run automatically, so please call "ZAux_Run3FileRam".
OP Parameters	Parameter name	Description
	premainbyte	Remain control's byte numbers
Return value	Error code	
Example	File 3 Usage	
Details	/	

Instruction 312	ZAux_3FileRamDownEnd	
Original Format	int32 __stdcall ZAux_3FileRamDownEnd(ZMC_HANDLE handle, int32 ifile3num)	
Description	Dynamically download Z3P file, when downloading ends, it will automatically add \0 (after called that, if you call ZMC_3FileRamDownPart again, there will be an error).	

Input parameters	Parameter name	Description
	handle	Link mark
	ifile3num	File 3 file No.
OP Parameters		
Return value	Error code	
Example	File 3 Usage	
Details	/	

Instruction 313	ZAux_Direct_MoveOpAdist	
Original Format	int32 __stdcall ZAux_Direct_MoveOpAdist(ZMC_HANDLE handle, int iaxis,float distance)	
Description	Turn off the glue in advance. It is mainly for ZAux_Direct_MoveOp to output one certain of distance in advance ➤ 0 (default): invalid. ➤ Positive value: early to output assigned vector distance ➤ Negative value: delay to output assigned vector distance Notes: a) It is only valid in one single OP of ZAux_Direct_MoveOp, and operations are done in sequence for each axis, if the first command that changes the position is not completed, following operations will not executed. b) It will be automatically cleared to 0 after MOVEOPADIST is brought into buffer, then other ZAux_Direct_MoveOp will not be affected. c) For HW controllers, they can use AXIS_ZSET to open HW precision output, which is same as ZAux_Direct_MoveOp. This command only supports MOVE interpolation type motions, and MOVESCAN motions, and can make MOVE output across small line segments in advance, but not support cam, MOVE_PT, etc.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
	distance	OP No.
OP Parameters	/	
Return value	Error code	
Example	<pre>ZAux_Direct_Single_Move(handle, 0, 100); //single-axis motion, 100 ZAux_Direct_MoveOp(handle, 0,0, 1); //axis 0 moves 100 in forward, after that, set out0 as 1 ZAux_Direct_Single_Move(handle, 0, 100); //single-axis motion, 100 ZAux_Direct_MoveOpAdist(handle, 0,10.0);//early 10 unit to close OUT0 ZAux_Direct_MoveOp(handle, 0,0, 0); //axis 0 moves 200 in forward, after that, set out0 as 0</pre>	
Details	Valid in ZMC4XX series motion controllers, and the firmware	

	should be "fast" version after 190201.

Instruction 314	ZAux_Direct_GetAxisStatus2	
Original Format	int32 __stdcall ZAux_Direct_GetAxisStatus2(ZMC_HANDLE handle, int iaxis,int homeln ,int *piValue,int *idle,int *bHomestaus,int *BusEnableStaus)	
Description	Read bus axis state.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
	homeln	Origin mapping IO No. (if it is <0, then homestatus will not take effect).
OP Parameters	Parameter name	Description
	piValue	Axis state returned value
	Idle	0 – running, others – no motion
	bHomestatus	1 – yes, 0 – no
	BusEnableStatus	Bus axis enable state
Return value	Error code	
Example	int iaxis=0; //axis No. int homeln=0; //origin No. int cAxisStatus; //get the axis state int cIdle; //get the command motion state int cHomestaus; //get the origin state int cBusEnableStaus;//get the bus axis state ZAux_Direct_GetAxisStatus2(handle, iaxis, homeln,&cAxisStatus,&cIdle,&cHomestaus,&cBusEnableStaus);	
Details	/	

Instruction 315	ZAux_Direct_GetHomeStatus2	
Original Format	int32 __stdcall ZAux_Direct_GetHomeStatus2(ZMC_HANDLE handle,uint32 iaxis,uint32 * homestatus)	
Description	Homing completion state 2.	
Input parameters	Parameter name	Description
	handle	Link mark
	iaxis	Axis No.
OP Parameters	Parameter name	Description
	homestatus	Homing completion mark: ➤ 0 – homing abnormal ➤ 1 – homing success ➤ 2 – it is homing
Return value	Error code	
Example	int ret=0; uint32 mHomeStatus=0; ret=ZAux_Direct_Single_Datum(handle,0,14);	

	<pre>//control axis 0 to do negative limit homing sleep(1); ZAux_Direct_GetHomeStatus2(handle,0,&mHomeStatus); While(mHomeStatus==2) { ZAux_Direct_GetHomeStatus2(handle,0,&mHomeStatus); } If (mHomeStatu) { Printf("home ok"); }else { Printf(home error); } }</pre>
Details	/

Instruction 316	ZAux_BusCmd_GetHomeStatus2						
Original Format	int32 __stdcall ZAux_BusCmd_GetHomeStatus2(ZMC_HANDLE handle,uint32 iaxis,uint32 * homestatus)						
Description	Homing completion state 2.						
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark	iaxis	Axis No.
Parameter name	Description						
handle	Link mark						
iaxis	Axis No.						
OP Parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>homestatus</td><td> Homing completion mark: ➤ 0 – homing abnormal ➤ 1 – homing success ➤ 2 – it is homing </td></tr> </tbody> </table>	Parameter name	Description	homestatus	Homing completion mark: ➤ 0 – homing abnormal ➤ 1 – homing success ➤ 2 – it is homing		
Parameter name	Description						
homestatus	Homing completion mark: ➤ 0 – homing abnormal ➤ 1 – homing success ➤ 2 – it is homing						
Return value	Error code						
Example	<pre>int ret=0; uint32 mHomeStatus=0; ret=ZAux_BusCmd_Datum(handle,0,-1); //call drive default homing mode sleep(1); ZAux_BusCmd_GetHomeStatus2(handle,0,&mHomeStatus); While(mHomeStatus==2) { ZAux_BusCmd_GetHomeStatus2(handle,0,&mHomeStatus); } If (mHomeStatu) { Printf("home ok"); }else { Printf(home error); } }</pre>						
Details	/						

Instruction 317	ZAux_BusCmd_SDOWrite2																				
Original Format	int32 __stdcall ZAux_BusCmd_SDOWrite2(ZMC_HANDLE handle,uint32 slot,uint32 node,uint32 index,uint32 subindex ,uint32 type ,char * in_string)																				
Description	Set node SDO parameters (be compatible with new functions)																				
Input parameters	Parameter name	Description																			
	handle	Link handle																			
	slot	Slot No., default is 0																			
	node	Node No.																			
	index	Object dictionary No. (note: the is data in decimal system)																			
	subindex	Object dictionary sub No. (note: the is data in decimal system)																			
	type	Data type: <table><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>flot</td></tr><tr><td>9</td><td>string</td></tr></table>		1	boolean	2	integer 8	3	integer 16	4	integer 32	5	unsigned 8	6	unsigned 16	7	unsigned 32	8	flot	9	string
	1	boolean																			
	2	integer 8																			
	3	integer 16																			
4	integer 32																				
5	unsigned 8																				
6	unsigned 16																				
7	unsigned 32																				
8	flot																				
9	string																				
in_string	Data value (write in character string)																				
OP Parameters	/																				
Return value	Error code																				
Example	/																				
Details	Compared to ZAux_BusCmd_SDOWrite, this adds “flot” and “string” functions.																				

Instruction 318	ZAux_BusCmd_SDORead2	
Original Format	int32 __stdcall ZAux_BusCmd_SDORead2(ZMC_HANDLE handle,uint32 slot,uint32 node,uint32 index,uint32 subindex ,uint32 type ,char * Out_string)	
Description	Read node SDO parameters (be compatible with new functions)	
Input parameters	Parameter name	Description
	handle	Link handle
	slot	Slot No., default is 0
	node	Node No.
	index	Object dictionary No. (note: the is data in decimal system)
	subindex	Object dictionary sub No. (note: the is data in decimal system)

	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
	in_string	Data value (write in character string)
Output Parameters	Parameter name	Description
	Out_string	Returned value in character string
Return value	Error code	
Example	/	
Details	/	

Instruction 319	ZAux_Up3File	
Original Format	int32 __stdcall ZAux_Up3File(ZMC_HANDLE handle, const char* pfilename, const char* pfilenameinControl)	
Description	Upload file 3 file to the local directly.	
Input parameters	Parameter name	Description
	handle	Link mark
	pfilename	Upload the file to local (path + filename)
	pfilenameinControl	File 3 filename (path + name) in controller
OP Parameters	/	
Return value	Error code	
Example	/	
Details	/	

Instruction 320	ZAux_Up3FileToMem	
Original Format	int32 __stdcall ZAux_Up3FileToMem(ZMC_HANDLE handle, char* pBuffer, uint32 bufsize, const char* pfilenameinControl, uint32* puifilesize)	
Description	Upload file 3 file to the controller file memory's buffer.	
Input parameters	Parameter name	Description
	handle	Link mark
	pbuffer	Assigned buff address
	bufsize	The length of buff
	pfilenameinControl	Controller filename
	puifilesize	File size

OP Parameters	/
Return value	Error code
Example	/
Details	/

Instruction 321	ZAux_BusCmd_NodePdoReadFloat		
Original Format	int32 __stdcall ZAux_BusCmd_NodePdoReadFloat(ZMC_HANDLE handle,uint32 inode,uint32 index,uint32 subindex , uint32 type,float* ivalue)		
Description	Read PDO in flot.		
Input parameters	Parameter name	Description	
	handle	Link mark	
	inode	Device No.	
	index	Object dictionary No	
	subindex	Object dictionary 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	
Output Parameters	Parameter name	Description	
	value	Read data value	
Return value	0 = success, not-0 = failure (please check error codes for details).		
Example	/		
Details	/		

Instruction 322	ZAux_BusCmd_NodePdoWriteFloat		
Original Format	int32 __stdcall ZAux_BusCmd_NodePdoWriteFloat(ZMC_HANDLE handle,uint32 inode,uint32 index,uint32 subindex, uint32 type,float value)		
Description	Set PDO in flot.		
Input parameters	Parameter name	Description	
	handle	Link mark	
	inode	Device No.	
	index	Object dictionary No.	
	subindex	Object dictionary 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	
	value	The data value			
OP Parameters					
Return value	0 = success, not-0 = failure (please check error codes for details).				
Example	/				
Details	/				

Instruction 323	ZAux_ZpjDown	
Original Format	int32__stdcall ZAux_ZpjDown(ZMC_HANDLE handle,const char *ZpjName,const char *ZarName,const char *pPass, uint32 uid,uint32 run_mode)	
Description	Download ZAR program into controller for operation.	
Input parameters	Parameter name	Description
	handle	Link mark
	ZpjName	ZPJ file source path
	ZarName	Generated ZAR file path
	pPass	Software password, bind APP_PASS, when there is no password, pPass=NULL
	unit32 uid	Bind with the controller unique ID, 0 – not to bind.
	run_mode	Downloading mode, 0 – RAM, 1 – ROM
OP Parameters		
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Char * ZpjFileName= "C:\\Users\\Administrator\\Desktop\\Frame11Demo\\ZmcZpj.zpj"; //ZPJ path Char *ZarFileName= "C:\\Users\\Administrator\\Desktop\\Frame11Demo\\ZmcZpj.zar; //generated ZAR path name Int iret=ZAux_ZpjDown(g_handle,ZpjFileName,m_ZarFileName,NULL,0,1); //download ZPJ	
Details	Zar file, it can directly run Zmotion motion control card program. It is the file that is generated after encryption.	

Instruction 324	ZAux_Down3FileRom					
Original Format	int32__stdcall ZAux_Down3FileRom(ZMC_HANDLE handle, const char* pfilename, const char* pfilenameinControl);					
Description	Run the downloaded file.					

Input parameters	Parameter name	Description
	handle	Link mark
	pfilename	File 3 name
	pfilenameinControl	It supports NC filename, it must be with name, otherwise it can't be known the file type: basic file or NC file.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	/	

Instruction 325	ZAux_Check3File	
Original Format	int32__stdcall ZAux_Check3File(ZMC_HANDLE handle, const char* pfilenameinControl, uint8 *pblfExist, uint32 *pFileSize)	
Description	Check whether there is Z3P file (file 3) in the controller.	
Input parameters	Parameter name	Description
	handle	Link mark
	pfilenameinControl	Z3P file name
	pblfExist	Whether there is mark
	pFileSize	File size
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	Zar file, it can directly run Zmotion motion control card program. It is the file that is generated after encryption.	

Instruction 326	ZAux_Delete3File	
Original Format	int32__stdcall ZAux_Delete3File(ZMC_HANDLE handle, const char* pfilenameinControl)	
Description	Delete the Z3P file (file 3) of the controller.	
Input parameters	Parameter name	Description
	handle	Link mark
	pfilenameinControl	Z3P file name
	pblfExist	Whether there is mark
	pFileSize	File size
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	Zar file, it can directly run Zmotion motion control card program. It is the file that is generated after encryption.	

Instruction 327	ZAux_FindFirst3File	
Original Format	int32_stdcall ZAux_FindFirst3File(ZMC_HANDLE handle, char* pfilenameinControl, uint32 *pFileSize)	
Description	Get the first Z3P file of the controller. If it is empty, which means the file doesn't exist.	
Input parameters	Parameter name	Description
	handle	Link mark
	pfilenameinControl	Obtained Z3P filename, if it is empty, which means the file doesn't exist.
	pFileSize	File size
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	Zar file, it can directly run Zmotion motion control card program. It is the file that is generated after encryption.	

Instruction 328	ZAux_FindNext3File	
Original Format	int32__stdcall ZAux_FindNext3File(ZMC_HANDLE handle, char* pfilenameinControl, uint32 *pFileSize)	
Description	Get the next Z3P file of the controller. If it is empty, which means the file doesn't exist.	
Input parameters	Parameter name	Description
	handle	Link mark
	pfilenameinControl	Obtained Z3P filename, if it is empty, which means the file doesn't exist.
	pFileSize	File size
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	Zar file, it can directly run Zmotion motion control card program. It is the file that is generated after encryption.	

Instruction 329	ZAux_GetCur3File	
Original Format	int32__stdcall ZAux_GetCur3File(ZMC_HANDLE handle, char* pfilenameinControl, uint32 *pFileSize)	
Description	Get the current Z3P file of the controller. If it is empty, which means the file doesn't exist.	
Input parameters		
	Parameter name	Description
	handle	Link mark
	pfilenameinControl	Obtained Z3P filename, if it is empty, which means the file doesn't exist.
	pFileSize	File size

OP Parameters	/
Return value	0 = success, not-0 = failure (please check error codes for details).
Example	/
Details	Zar file, it can directly run Zmotion motion control card program. It is the file that is generated after encryption.

Instruction 330	ZAux_RemoveAll3Files	
Original Format	int32__stdcall ZAux_RemoveAll3Files(ZMC_HANDLE handle)	
Description	Delete all Z3P files of the controller.	
Input parameters	Parameter name	Description
	handle	Link mark
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	Zar file, it can directly run Zmotion motion control card program. It is the file that is generated after encryption.	

Chapter XIV Instruction Returned Values

14.1. Instruction Returned Value

User can send the command through calling function that is encapsulated by dynamical library in program, then, motion controller will move according to received motion commands.

After the instruction is accepted by motion controller, return instruction execution result for the host computer that sends the command to indicate whether the instruction is executed correctly.

14.2. Return Values Details (Error Codes)

Error Code	Meaning	Example
210	File is too large	
212	State error	It is non-suspended state in Resume
213	File downloading and uploading errors, and package loss	When PC function is called, return this error.
214	The length of downloaded file is checked that is wrong.	
215	Buffer length or storage is not enough	When the character string command is over long, return this error.
217	Functions are not supported or prohibited by controllers.	
218	Called and transferred parameters are wrong.	
219	Download appears conflict, multiple files are downloaded simultaneously.	
220	Filename error, there are special characters.	
221	Filename error, more than the length.	
222	File does not exist	
223	Password protection limits.	
224	Password protection limits 2.	
260	Hardware error	
261	Disk is not formatted	
262	RTC error	
263	NORFLASH error	

264	RAM error	
265	NANDFLASH error	
266	USB Drive error	
267	FPGA error	
268	Ethernet hardware error	
271	Backup power supply error	
272	Sub-card does not exist	
273	File is missing	
274	System File Error	
275	No master, generating from sub-card	
276	Program file correction error	
277	Program file error causes it can't start	
278	ZAR check, APPPASS error	
279	ZAR check, ID error	
280	BAS file exceeds the maximum number	
281	Sub-card ID conflicts, or multi-main conflict	
282	Unsupported functions	
284	The controller does not match with zar	
285	Image File Error	
286	Font file error	
288	Controller appears above errors, it will cause alarm when ON next time.	
1000	Motion module returns wrong offset	
1002	No motion buffer	
1004	Slave axis is moving	
1005	Motion functions that are not supported	
1006	Arc position error	
1007	Parameter ellipse AB errors	
1008	The motion module input parameter error	
1009	It is in motion process, which means it can't be operated	
1010	Pause and others are operated repeatedly	
1011	IDLE can't do pause, this kind of operation.	
1012	The current movement doesn't support pause	
1013	Pause point can't be found	
1014	ATYPE are not supported	
1015	ATYPE of ZCAN conflicts	
1016	Functions are not supported by axis	
1017	FRAME correction data error	
1018	FRAME correction data is not enough	
1019	Data meeting conditions in FRAME correction data is not enough	
1020	FRAME auxiliary parameter correction data is not enough	

1021	The interval of FRAME correction data is too small, smaller than the number of joint axes	
1022	FRAME coordinate error	
1023	Coordinate can't be modified compulsively under FRAME status	
1024	FRAME abnormal inverse solution	
1025	Not FRAME status	
1026	FRAME HAND error	
1027	Pose can't be switched in interpolation	
1028	Axis equivalent of special joint axis and virtual axis are the same.	
1030	CORNERMODE 7-bit is set well but it does not support this motion	
1031	CORNERMODE 7-bit is set well but it is not in FRAME status	
1032	AXIS_ADDRESS error	
2000	ZBASIC module offset.	
2021	Manually stop	
2022	This task stops caused by errors in other tasks	
2023	Attempt to modify as read-only parameter	
2024	Array is over bound	
2025	The number of variables is more than controller specifications	
2026	The number of arrays is more than controller specifications	
2027	The number array space is more than controller specifications	
2028	The number of SUB is more than controller specifications	
2029	Identifier naming error	
2030	Identifier name is too long	
2031	No right parenthesis	
2032	Unknown characters	
2033	Expressions encountered unknown name	
2034	SUB can't be used in expression	
2043	Unknown command identifier, and it is the first identifier name of current line.	
2044	Stack Overflow	
2045	Math expression is too complex, different controllers have different specifications	
2046	End reference numerals are not found	
2047	Command does not have return value, not be used for expression calculation.	
2048	Function must return a value, no need at the beginning of the line.	
2049	Special instruction must be in a	

	separate line	
2050	Parameters or arrays need to be indexed.	
2051	Variables can not use index	
2052	Redefine array, and the length is inconsistent.	
2053	The "length" parameter of array definition is wrong, it is negative or too large.	
2054	Identifiers have been defined as the SUB process, they can not do others	SUB process mark symbol can't be defined again.
2055	Identifiers have been defined as parameters, can not do others.	
2056	Identifiers reserved and can not be used.	
2057	Unrecognizable character appears	Such as, "&" can't be identified
2058	SUB calling is out stake repeatedly	
2060	Syntax format error	There is no TO after FOR
2062	Function parameters range Error	When the task No. exceeds the range, it will return this error. Auto run task No. error is the error code.
2063	Too many function parameters	Like, CONNECT(1,1,1)
2064	Too less function parameters	Like, CONNECT(1)
2065	Lack operands	
2066	Lack operands behind operators	
2067	Lack operands before operators	
2068	Unknown operators	
2069	Lack binary operators	An operator is needed between two instructions that are in same line.
2070	CALL must call SUB	
2072	Require assignment symbol	It will report error when use space between data instead of comma.
2073	Empty file	
2074	SUB-defined identifier name conflicts.	
2075	Task to be opened has been running.	
2076	Multiple parameters are separated by commas.	
2077	Parentheses are not matched, no left parenthesis.	
2078	"IF" nesting too much	
2079	"Loops" nested too much.	
2080	The number of interpolation axes is too less	
2081	CONST constant can not be modified.	It will report error when defined constant data is assigned again.
2082	Command can not be sent from the PC online.	

2083	Too many SUB-defined parameters.	
2084	SUB with parameters, they can not be used GOTO statement.	
2085	Local identifier defined too much	
2086	LOCAL variable names and variable names or other identifiers file name conflict.	
2087	LOCAL does not support the array definition.	
2088	GSUB defined parameters letters repeated.	
2089	GSUB defined parameters can only be a single letter.	
2090	Modify read-only parameter is not allowed	
2091	GSUB_IFPARA function uses the wrong occasion.	
2092	Division by zero	
2093	Over buffer	
2094	Command-line blocking time is too long.	
2095	Parameter same name	
2096	Values are used without initialization	
2097	Axis No. conflicts	
2099	internal error	
2100	The number of SCANEDGE is too many	
2101	ZINDEX type mismatch	
2901	System errors, defined identifiers are too many, including variables, arrays, process, process parameters and so on.	
3201	Over buffer	
3202	File ends abnormally	
3204	Internal state error	
3205	Unsupported functions	
3206	Internal call parameters error	
3231	Resource is not enough	
3242	"os" error	
3243	U disk is not inserted	
3244	File is opened repeatedly	
3245	File is oversize	
3248	Filename error	
3249	Filename is over long	
3250	File doesn't exist	
3301	Three points in a circular arc line.	
3302	Two parallel straight lines, there is no intersection.	
3401	MODBUS master station parameter is wrong, and the normal length is over	
3402	Message Response Timeout	
3407	MODBUS return parameter error	

3408	MODBUS return is not supported	
3421	MODBUS slave station returns invalid function codes	
3422	MODBUS slave station returns address space error	
3423	MODBUS slave station returned wrong length data	
3424	MODBUS slave station returned length is too long	
3501	ZCAN return without sub card	
3502	No relative axis when ZCAN returns sub card	
Error: 4000-4500 PLC module		
4002	Parameter error	
4003	Unknown type	
4004	Unknown function	
4005	Push too many STL	
4006	Push too many stakes	
4007	The program is too complex, and BLOCK is too many	
4008	No push BLOCK	
4009	No push STL	
4010	No push	
4014	File content errors	
4015	RET must be behind STL	
4016	Over range	
4017	Below range	
4018	L is not defined	
4019	Not support G code function	
4020	GOTO can not cross PLC and BASIC	
4021	PLC is only one main task	
4022	Grammatical errors	
4023	FOR NEXT errors, mismatched	
4024	FOR NEXT error, no NEXT	
4026	FOR MC mix	
4027	FOR STL mix	
4030	Must be used in PLC main task	
4031	Must be used in interrupt	
4032	The number of parameters is less	
4033	The number of parameters is more	
4034	To be a multiple of 8	
4035	The register indicates an error	
4036	Register type error	
4037	More than the number of LVs	
4038	Read-only	
4500-5000 PLC PC side error		
4503	Not enough memory	
4504	Return to the bus cable	

4505	Reflux	
4506	AND type can not be directly connected to the bus cable	
4510	Dangling	
4511	The far right must be the type of output	
Error: 5000-5500 HMI module		
5000	No. of LCD is wrong	
5002	LCD No. conflicts	
5003	Object that is not supported	
5004	Memory is not enough	
5005	Control Hierarchy Error	
5006	Window NO. exceeds	
5007	Invalid window number	
5010	Object attribute is missing	
5011	There are several display elements on input window.	
5012	ACTION type error	
5013	Too many events.	
5014	Fail to return to the previous window	
5015	Basic window can't close	
5016	Font is not found in the related character	
5017	It must be used in HMI tasks	
5020	Control ID Conflict	
5021	Error No. LCD	
5022	Available LCD can't be found	
5023	LCD does not open	
5024	LCD has no data	
5025	Program reset	
5026	LCD has been opened	
5027	Not a network LCD	
5028	It does not support compression	
5029	Color depth is not supported	
5030	Unsupported Data Types	
5031	Device No. error	
5032	LCD_SEL can not be used	
5033	REDRAW can't be set in DRAW stage	
5034	DRAW function only can be called in DRAW stage	
5035	Operation can't be called in DRAW stage	
5036	Internal LCD resolution is fixed	
5037	LCD resolution exceeds	
5038	Library filename error	
5039	Too many characters	
5501-5599	PC side PLC file compile errors	
5501-5599: PC-side PLC file compilation error		
5503	Memory is not enough	

5504	Return to bus wire	
5505	Reflux	
5506	AND type instruction can not be directly connected to the bus	
5510	Right is hanging in the air, there is no output instruction	
5511	The far right type of instruction is not output	
5512	The far right can not be connected together	
5513	Output type of instruction must be the rightmost	
5514	It does not support the type of instruction	
5517	No value of register	
5518	DOT value exceeds the range	
5519	Index register exceeds the range	
5520	Excessive number of characters	
5521	Register type error	
5522	Register value error	
5523	Excessive number of registers	
5524	The number of registers is too less	
5525	Use STL wrongly	
5526	Use RET wrongly	
5527	Repeat RET	
5528	Position error of END or LBL	
5529	Function can't be directly connected to the bus	
5530	Out the stake without pushing the stake	
5531	Too many MPP	
5532	Using the wrong type register	
5533	ANB error, insufficient number of blocks	
5534	ORB error, insufficient number of blocks	
5535	ANB error, it can't be combined after output operation	
5536	ORB error, it can't be combined after output operation	
5537	AND directly connects to bus	
5538	OR directly connects to the bus	
5539	OR can't be behind OUT	
5540	STL can not be shared with MC	
5541	MC can not directly access the bus	
5542	_@ register needs brackets	
5543	Notes error	
5544	Ladder excessive number of columns	
5545	Output types can not be directly connected to bus	
6000: ECAT bus error		
6000	ECAT module error, SLOT No. error	

6001	Internal error, function is not supported.	
6005	Parameter error	
6006	The number of device types that are supported is over the limit	
6009	NODE operation times exceeds	
6012	Insufficient resources	
6013	Slave device response timeout	
6014	Buffer is not enough	
6015	Response packet error WKC	
6016	Long answer of SDO	
6017	SDO wrong response	
6018	SDO response data length error	
6019	WKC timeout	
6020	STATE switch timeout	
6021	SDO ABORT, drive return error	Data dictionary write & read error or unsupported data of drive is written.
6023	NODE PROFILE error	
6024	Axis PROFILE error	
6025	The number of axes exceeds	
6029	PDO list length exceeds the number of system specification	
6031	The number of devices exceeds	
6042	Device does not support	
6045	E-mail timeout	
6047	Data type error	
6049	Sub module is not supported by module	
6050	Submodule numbers exceeds the limit	
6051	Unknown submodule by module	
6208	RTEX drive ID conflict	
6209	Scan timeout	Usually, net cable appears problem
6210	RTEX fails to initialize	
6211	RTEX scan result error	
20000: PC side error		
20000	There is wrong offset on PC side	
20002	Wrong parameters	
20003	time out	Fifo buffer may be blocked
20006	Operating system error	
20007	Fail to open serial	
20008	Fail to turn Ethernet on	
20009	Handle Error	
20010	Send Error	
20011	File Error	
20012	File length error	
20013	File name is too long	
20014	file does not exist	
20015	ZLB library file errors	

20016	File does not compile	General PLC file needs to compile
20020	Firmware file does not match	
20021	Unsupported functions	
20030	Input buffer length is not enough	
20100	Response buffer length is not enough	
30000	Above 30000 are the error codes generated from ZAUX auxiliary library.	
30001	No respond.	
30002	Respond error.	
30003	Parameter error.	
30004	Unsupported	
30005	Parameter error.	

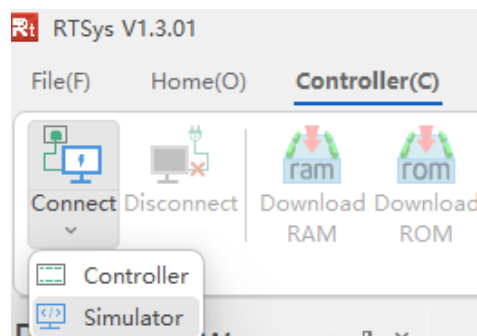
14.3. Others: Quick Start / Common Problems

14.3.1. Quick Start

Simulation Connection

Step 1: open RTSys software (it can be downloaded from [Zmotion website](#) or [contact us](#))

Step 2: Connect to Simulator.



Step 3: Connect to simulator through ethernet, IP address is 127.0.0.1.

```
// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s return code is %d\n", command, ret);
    }
}
```

```

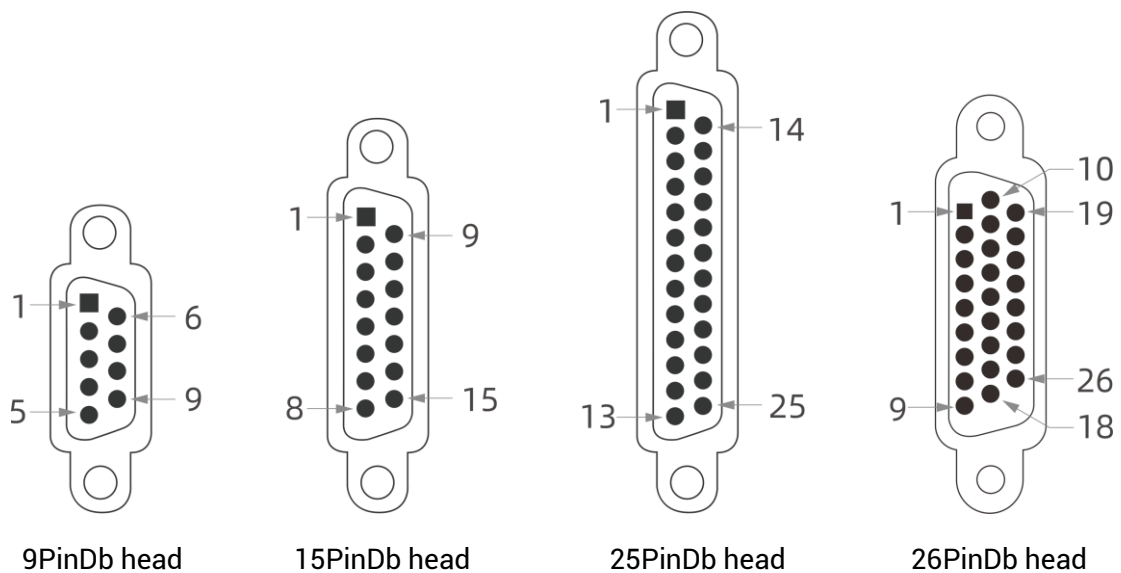
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"127.0.0.1";    //controller IP address, simulator IP address, it needs
    to open the simulator in RTSys
    ZMC_HANDLE handle = NULL;                //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to Controller!\n");
        handle = NULL;
        getchar();
        return -1;
    }
    printf("Success to Controller!\n");

    //obtain motion type}
    Sleep(2000);
    ret = ZAux_Close(handle); //close the connection
    commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
    printf("connection closed!\n");
    handle = NULL;
    return 0;
}

```

Pulse Servo Diver Wiring



Please refer to specific user manual.

14.3.2. Common Problems

Not to Connect to Controller?

The default IP address of the controller is 192.168.0.11, please make PC IP address consistent with controller IP segment (first 3 segments are same, the last one is different), and then connect.

For some laptops, their wireless connection conflicts with the Ethernet. At this time, please don't try to connect through wireless.

- (1) check PC IP, it should be in same segment with controller IP.
- (2) check controller PC, you can use the serial port to connect, then check it.
- (3) check the PC firewall settings.
- (4) Ping the controller IP to see if the controller can be pinged through. If the Ping fails, check the physical interface or the cable.

If the PC cannot connect to the internet and the controller at the same time

➤ Method 1

Modify controller IP, it should be on the same segment as the PC and the gateway.

➤ Method 2

Modify the subnet mask of the PC so that the IP address of the gateway and the controller are in the same network segment, for example, change it to: 255.255.128.0

Motion Command Called, Axis Not Move?

➤ Possibility 1:

There is limit or alarm input, or the level configuration of the limit alarm is wrong.

- For special signals, they are normally closed by default, but for most of Chinese products, they are normally opened. They can be switched (inverted) by INVERT_IN command.

➤ Possibility 2:

Axis type ATYPE is configured incorrectly.

How to Calculate IO Expansion Module No.?

Depend on module ID, and the ID is determined by the dial switch.

- For ZIO1608:

ID0 - Input 16 – 31 & Output 16-23

ID1 - Input 32 - 47 & Output 32-39

And so on.

All Commands can be Sent Directly by PC?

Nope.

Some commands such as WAIT and CAN are restricted from being sent from the PC, because it will cause "PC commands blocked".

What's more, it also can't define a variable array through the PC command.

How to Stop the Motion Immediately?

There is **no absolute immediate stop**.

- You can set a large value for "fastdel" to make it stop rapidly when meets the limit or triggers the rapidstop.
- If it moves again after stopped, it is best to add the judgment of wait idle
- rapidstop ignores axes in BASE, it stops all axes

How to Achieve Handwheel?

It can use "connect" command to achieve that, and there is reference handwheel routine, please go to Zmotion website – technical support / download or contact us.

How to Access Some System States Directly By MODBUS?

IO, position etc. can be accessed through special MODBUS registers.

- IN: start from BIT 10000
- OP: start from BIT 20000.

Each axis of the word register occupies two words, in float format.

Motion is still Running After Cancel?

- "CANCEL" is to cancel the current motion (default), that is, it won't cancel the buffered motion. You could use "cancel (2)" to cancel all motions.
- Check whether the acceleration and deceleration are set.
 - Even though there is no absolute immediate stop, you could set a large value for "fastdel", to make it stop rapidly when it meets the position limit or when in rapidstop.

- If the deceleration or rapid deceleration is not set, then it can't stop.

How to Achieve Power Failure Storage?

The VR data will be saved when the power is turned off.
For other data, you could save the data into flash, then read out when the power is turned on (*note: the flash should not be read and written frequently, because there is a limit on the number of reads and writes*).

How to Check Errors when Running?

- Do not restart. Please connect it and download it by RTSys (select "attach current program"), then program inner state and error positions can be known.
- Also, you could enter "?*task" in command window to view error positions.

Overshoot after Position Limit, How to Do?

- Increase FASTDEC, or enlarge induction block sensing range of position limit signal.
- Check whether the "invert" is set for position limit.

MERGE=ON, But no Continuous Interpolation?

- Check whether MERGE is set on the first axis of BASE in MOVESP instruction, or there are other axes parameters.
- If the motion command is a small line segment, check whether the motion time is too short, if yes, PC command sending speed is slower than motion speed.

Sensor Signal Unstable, the Signal is Affected, How to Do?

- Check whether the power supply of the IO sensor is not the same as the IO power supply of the controller. At this time, the baselines of the two need to be connected.
- If there is drive IO mapping, the IO signal jumps, please detect whether IO mapping is repeated.

When Homing and It is near the Origin, Inverse-Find Slowly?

The special protection is set for the situation when the servo motor may cross the origin. Then, the controller may decelerate to stop when detects the origin, but after completely stopped, the origin cannot be detected, so it is mistakenly thought that the axis has passed the origin sensing range.

- Solution: when homing, set LSPEED as 0 to make a certain deceleration distance.

Not Invert Level (0X System IN State Register on HMI)?

Because it reads the original state. INVERT command is only valid in IN Reading.

Not Execute Full-Circle Interpolation?

Full circle interpolation only can be achieved by relative circular interpolation.

How to Check Error when ZAR File Downloading Error?

Download ZAR file by RTSys, corresponding error information will be shown.

Not Move after PC Command Sent?

- Stop command is called continuously for controller BASIC program.
- Check the wiring.
- Check the axis state, whether there is the error.

Zmotion PC Programming Manual

[Advanced Functions]

Chapter I Instruction List

Chapter II Axis Superposition / Electronic Cam	
2.1 Motion Superposition	
ZAux_Direct_GetAddax	Read superposing axis
ZAux_Direct_Single_Addax	Axis superposition motion
2.2 Electronic Cam / Synchronous Tracking	
ZAux_Direct_Cam	Electronic cam, synchronous motion
ZAux_Direct_Cambox	Following cam table, synchronous motion
ZAux_Direct_Movelink	Special cam, synchronous motion
ZAux_Direct_Moveslink	Special cam 2, synchronous motion

Chapter III High-Speed Latch	
3.1 High-Speed Latch	
ZAux_Direct_Regist	Position latch
ZAux_Direct_CycleRegist	Continuous position latch
ZAux_Direct_SetCloseWin	Set the end coordinates range for latch triggering
ZAux_Direct_GetCloseWin	Read the end coordinates range for latch triggering
ZAux_Direct_GetRegPos	Read latch measurement position (MPOS)
ZAux_Direct_GetRegPosB	Read latch measurement position (MPOS)
ZAux_Direct_SetOpenWin	Set the starting coordinates range for latch triggering
ZAux_Direct_GetOpenWin	Read the starting coordinates range for latch triggering
ZAux_Direct_GetMark	Read encoder latch returned state
ZAux_Direct_GetMarkB	Read encoder latch B returned state

Chapter IV Position Comparison Output	
4.1 Position Comparison Output	
ZAux_Direct_Pswitch	Soft position comparison output
ZAux_Direct_HwPswitch	Hard position comparison output
ZAux_Direct_GetHwPswitchBuff	Hard position comparison output remain buffers
ZAux_Direct_HwPswitch2	Bus hard position comparison output (this is also valid in pulse axis and bus axis)
ZAux_Direct_HwPswitch2_2	Hard position comparison output 2
ZAux_Direct_HwTimer	Hard timer is used to restore the electric level after one certain of time after hard position comparison output.
ZAux_Direct_MoveHwPswitch2	Hard position comparison output, in buffer way
ZAux_Direct_MoveHwTimer	Hard timer is used to restore the electric level after one certain of time after hard position

	comparison output, in buffer way
ZAux_Direct_SetHwPs2AxisNum	Set axis No. of hard comparison output
ZAux_Direct_GetHwPs2AxisNum	Get axis No. of hard comparison output
ZAux_Direct_GetHwPs2Counts	Get compared points of hard comparison output

Chapter V PT Motion	
5.1 PT Motion	
ZAux_Direct_MultiMovePt	Multiple relative PT motions
ZAux_Direct_MultiMovePtAbs	Multiple absolute PT motions
ZAux_Direct_MultiMovePvt	Multiple relative PVT motions

Chapter VI Robot Motion Control	
6.1 Manipulator Models	
ZAux_Direct_Connframe	Build robot inverse kinematics
ZAux_Direct_Connreframe	Build robot forward kinematics
ZAux_Direct_FrameRotate	Rotate robot coordinate system
ZAux_Frame_Rotate	Transform workpiece coordinate system
ZAux_Frame_Rotate2	Calculate coordinate system
ZAux_World_DPos	World coordinate system
ZAux_Direct_FrameStatus	Robot axis state
ZAux_Direct_FrameTrans2	Transform forward mode and inverse mode
ZAux_Direct_FrameCal	Correct parameters
ZAux_Direct_Mover_C	Make joint-axis move plane circular interpolation, relative
ZAux_Direct_Mover_CABS	Make joint-axis move plane circular interpolation, absolute
ZAux_Direct_Mover_C3	Make joint-axis move space circular interpolation, relative
ZAux_Direct_Mover_C3ABS	Make joint-axis move space circular interpolation, absolute
ZAux_Direct_Mover_L	Make joint-axis move linear interpolation, relative
ZAux_Direct_Mover_LABS	Make joint-axis move linear interpolation, absolute

Chapter II Axis Superposition (Electronic Cam)

2.1. Motion Superposition

2.1.1. Instruction List

Commands	Descriptions
ZAux_Direct_GetAddax	Read superposing axis
ZAux_Direct_Single_Addax	Axis superposition motion

2.1.2. Principle & Description

■ Function Definition:

Motion superposition, which superimposes the motion of one axis to another axis.

■ Function Command:

ZAux_Direct_Single_Addax.

And the number of pulses is superposed, not UNITS.

➤ Usage:

motion distance of superimposing axis * pulse amount of superimposing axis / pulse amount of superimposed axis = motion distance of superimposed axis.

➤ Example:

suppose the pulse amount of axis A is 100, the pulse amount of axis B is 50, and the superimposing axis movement is 100

- ✧ Superimpose the movement of axis A to axis B. At this time, axis A moves 100, then axis B moves $100 \times 100 / 50 = 200$.
- ✧ The movement of axis B is superimposed on axis A. At this time, axis B moves 100, then axis A moves $100 \times 50 / 100 = 50$.

➤ Notes:

- The axes cannot be superimposed on each other at the same time. That is, after A is superimposed on B, B can no longer be superimposed on A.
- Serial superposition is valid -- A is superimposed to B, B is superimposed to C.
- Parallel superposition is valid, -- A superimposes to B and C at the same time.

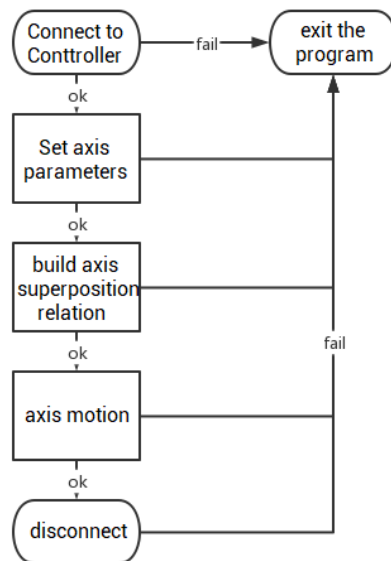
- When superimposing, the speed changes from the superimposed axis, and the acceleration & deceleration are determined according to the accel & decel of the superimposing axis and the ratio of the units of the two axes.
- **ZAux_Direct_Single_Addax function does not work when the current state of the axis is under forward solution (forward kinematic) or inverse solution (inverse kinematic).**

2.1.3. Routine

Motion Superposition

The pulse amount of axis 0 is 100, the pulse amount of axis 1 is 50.

Superpose axis 0 on axis 1, axis 0 moves 100.



```
// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"

void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail! return code is %d\n", command, ret);
    }
}
```

```
int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"127.0.0.1"; //controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to Controller! \n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to Controller!\n");

    ZAux_Direct_SetAtype(handle, 0, 1); //set axis 0 axis type as 1
    ZAux_Direct_SetUnits(handle, 0, 100); //set axis 0 pulse amount as 100
    ZAux_Direct_SetAtype(handle, 1, 1); //set axis 1 axis type as 1
    ZAux_Direct_SetUnits(handle, 1, 50); //set axis 1 pulse amount as 50

    ZAux_Direct_SetSpeed(handle, 0, 200); //set axis 0 speed as 200units/s
    ZAux_Direct_SetAccel(handle, 0, 2000); //set axis 0 acceleration as 2000units/s/s
    ZAux_Direct_SetDecel(handle, 0, 2000); //set axis 0 deceleration as 2000units/s/s
    ZAux_Direct_SetSramp(handle, 0, 200); //set axis 0 S curve time as 200ms

    ret = ZAux_Direct_SetDpos( handle, 0, 0); //clear axis command position to 0
    commandCheckHandler("ZAux_Direct_SetDpos", ret); //if it executes successfully
    ret = ZAux_Direct_SetMpos( handle, 0, 0); //clear encoder feedback position to 0
    commandCheckHandler("ZAux_Direct_SetMpos", ret); //if it executes successfully

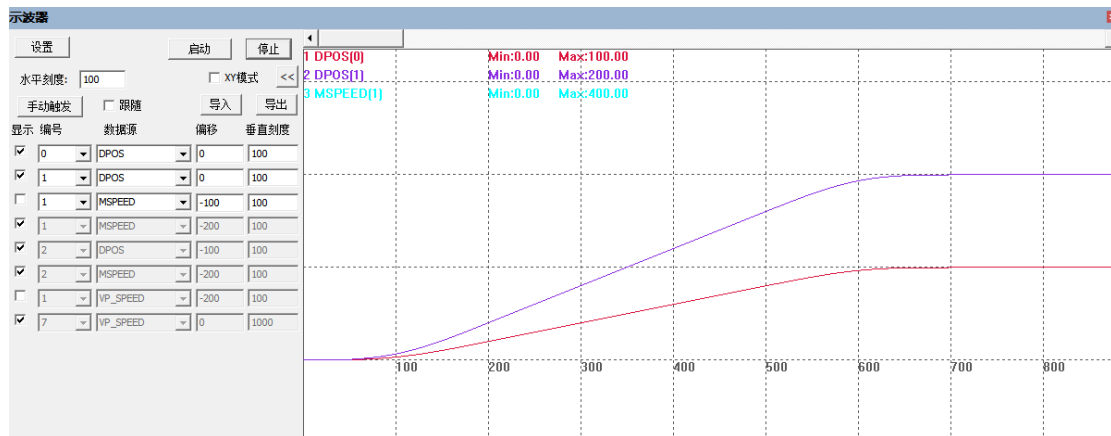
    ZAux_Direct_Single_Addax(handle,1,0); //axis 0 pulse is superposed on axis 1

    ZAux_Trigger(handle);
    ZAux_Direct_Single_Move(handle,0,100); //axis 0 moves 100
    float IDLE;
    while (1) //wait for axis 0 to complete the motion
    {
        Sleep(100);
        ZAux_Direct_GetParam(handle,"IDLE",0,&IDLE);
        if (IDLE<0)break;
    }
    ZAux_Direct_Single_Addax(handle,1,-1); //cancel superposition

    ret = ZAux_Close(handle); //close the connection
    commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
    printf("connection closed!\n");

    handle = NULL;
    return 0;
}
```

➤ Waveform:



2.2. Electronic Cam & Synchronous Track

2.2.1. Instruction List

Commands	Descriptions
ZAux_Direct_Cam	Electronic cam, synchronous motion
ZAux_Direct_Cambox	Following cam table, synchronous motion
ZAux_Direct_Movelink	Special cam, synchronous motion
ZAux_Direct_Moveslink	Special cam 2, synchronous motion

2.2.2. Principle & Description

(1) Electronic Cam

➤ Definition

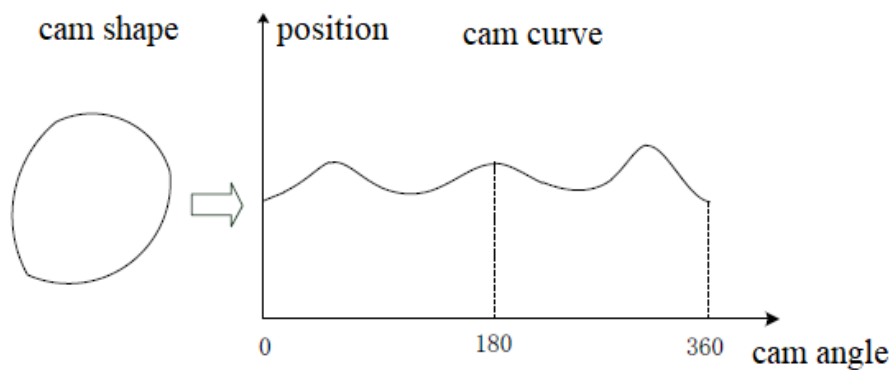
Electronic cam is used to simulate the mechanical-cam through constructed cam curve, in this way, it can achieve the software system that can do relative motion between cam axes and master axes, which is similar with of mechanical cam system. Namely, achieve mechanical-cam through “controller controls servo motor”, no need to install other mechanical structures.

➤ Theory

Electronic cam actually belongs to multi-axes synchronous motion, which means the motion consists of one spindle and one slave axis / multiple axes. It is developed on

the basis of mechanical cam. Then it is usually used for periodical curve motions.

As following graphic, according to cam contour, one motion trajectory that is between one rotating angle and processing position can be gained, and it can be seen it is one arc, which is decomposed as countless linear trajectories, and then one path (approaching arc motion) is formed. In real application, enter this path parameters into motion command for controlling the axis to move target trajectory.



Put one motion program of electronic-cam into controller, then feedback the position signal to controller through encoder. When position signals are received and processed by the controller, then all are sent to servo driver. Then, the driver controls multi axes to do synchronous motion for completing preset trajectory.

➤ Advantages

There are many advantages of electronic cam.

- promote machine control precision: make control distance longer, low error ratio, promote the reliability.
- simplify institution and make it more flexible, make debug and maintain easier.
- electronic cam is achieved by software, when you set motion related parameters, motion curve also can be changed, which is very flexible.
- Easy to install, no need extra mechanical components -- no tool wear situation.

(2) Synchronous Follow

Synchronous follow is a kind of electronic cam application, which is similar with "shearing", both need position synchronization between the mater axis and the slave axis, the difference is that, in "shearing", feeding material (master) is continuous, but in "synchronous follow", feeding material is random (materials appear randomly), and it usually will add sensor as signal to trigger synchronization, the signal comes each time, synchronization motion will only happen once. Then it will go back to starting position

after achieving set distance, and follow again until next trigger. This is commonly applied in assembly line, such as, synchronous spraying, container material injection, material clamping, etc.

(3) Automatic Cam

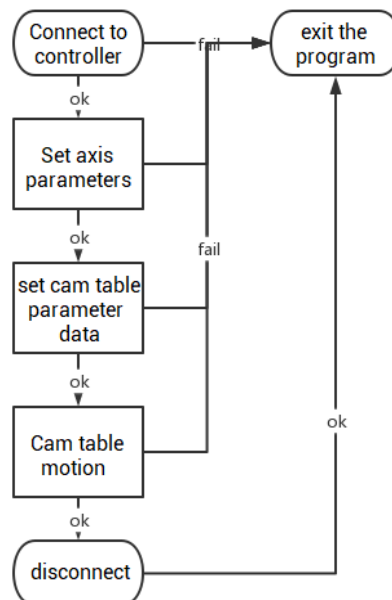
Automatic cam is a kind of cam motion. It is mainly aimed at the master-slave follow-up movement between two axes. And the motion can be built only by setting several related parameters. Please note the position relationship is not stored in the TABLE. The following distance and speed change process of each segment are set by command parameters. And the slave axis' speed is automatically calculated to match spindle during motion. Common motion processes include following acceleration, deceleration, and synchronization.

Commands: MOVELINK, MOVESLINK, FLEXLINK, etc.

Applications: chasing shear, flying shear, and wheel cutting, etc.

2.2.3. Routines

Cam Table



```
// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
```

```
#include "zaxdll2.h"
#include <math.h>
#define PI 3.1415
void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail! return code is %d\n", command, ret);
        Sleep(2000);
        exit(0);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1"; // simulator IP address
    //char *ip_addr = (char *) "192.168.1.11"; // controller IP address
    ZMC_HANDLE handle = NULL; // link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); // connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to Controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to Controller!\n");

    ret = ZAux_Direct_SetAtype(handle, 0, 1); // set axis 0 axis type as 1
    commandCheckHandler("ZAux_Direct_SetAtype", ret); // if it executes successfully
    ret = ZAux_Direct_SetUnits(handle, 0, 100); // set axis 0 pulse amount as 100
    commandCheckHandler("ZAux_Direct_SetUnits", ret); // if it executes successfully
    ret = ZAux_Direct_SetSpeed(handle, 0, 100); // set axis 0 speed as 100 units/s
    commandCheckHandler("ZAux_Direct_SetSpeed", ret); // if it executes successfully
    ret = ZAux_Direct_SetAccel(handle, 0, 1000); // set axis 0 acceleration as 1000 units/s/s
    commandCheckHandler("ZAux_Direct_SetAccel", ret); // if it executes successfully
    ret = ZAux_Direct_SetDecel(handle, 0, 1000); // set axis 0 deceleration as 1000 units/s/s
    commandCheckHandler("ZAux_Direct_SetDecel", ret); // if it executes successfully

    ret = ZAux_Direct_SetDpos(handle, 0, 0); // clear axis command position to 0
    commandCheckHandler("ZAux_Direct_SetDpos", ret); // if it executes successfully
    ret = ZAux_Direct_SetMpos(handle, 0, 0); // clear encoder feedback position to 0
    commandCheckHandler("ZAux_Direct_SetMpos", ret); // if it executes successfully

    int TableStartPoint = 10; // Table starting position
    int TableEndPoint = 10 + 1024 - 1; // Table end position
    float ftablemulti = 100; // position ratio
    float ReferDistance = 1000; // refer to movement distance
    int i, j;

    float setTableValue[1024]; // cam table data

    // write sine curve data
    for (j = 0; j < 4; j++)
    {
        for (i = 0; i < 256; i++)
        {
            setTableValue[i + j * 256] = (sin(((2 * PI) / 34) * i)) * 100;
        }
    }

    ZAux_Direct_SetTable(handle, TableStartPoint, 1024, setTableValue); // write cam table data into
    Table register value
```

```

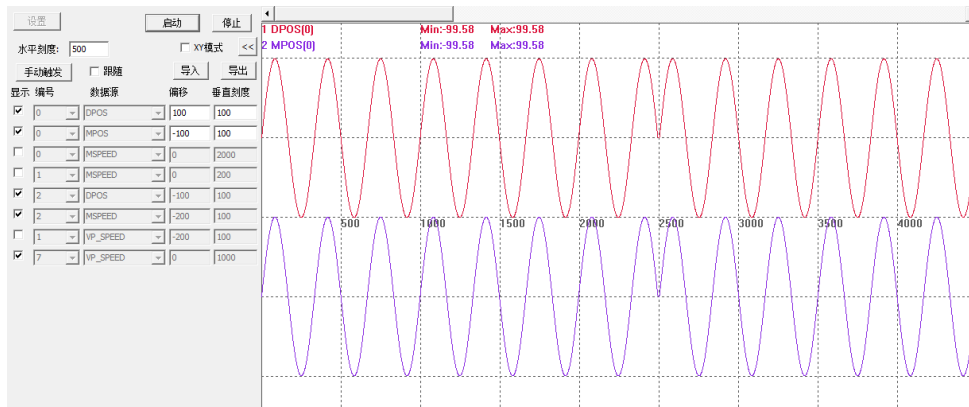
ZAux_Trigger(handle);
ret = ZAux_Direct_Cam( handle, 0, TableStartPoint, TableEndPoint, ftablemulti, ReferDistance);
//cam table motion
commandCheckHandler("ZAux_Direct_Cam", ret);
float IDLE;
while (1)//wait for axis 0 to complete the motion
{
    Sleep(100);
    ZAux_Direct_GetParam(handle,"IDLE",0,&IDLE);
    if (IDLE<0)break;
}

Sleep(6000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
printf("connection closed!\n");

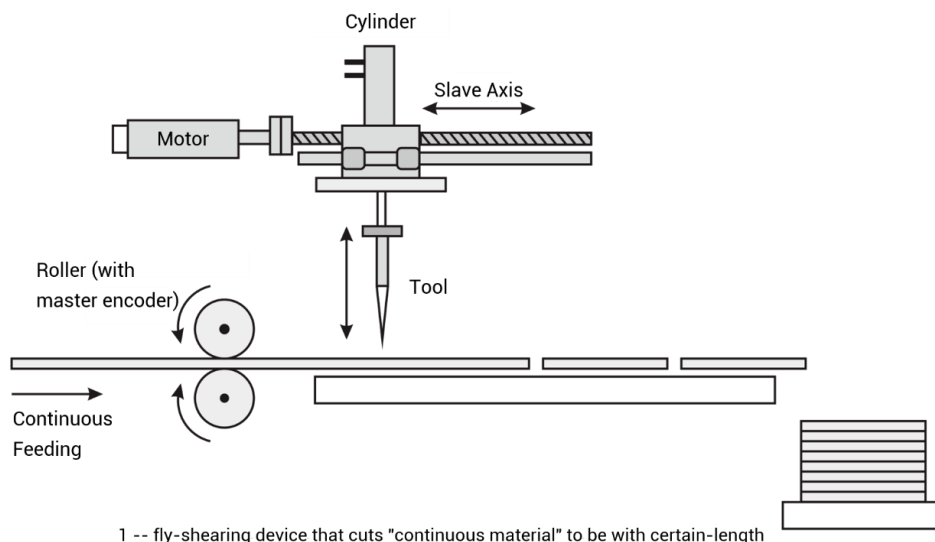
handle = NULL;
return 0;
}

```

➤ Waveform:



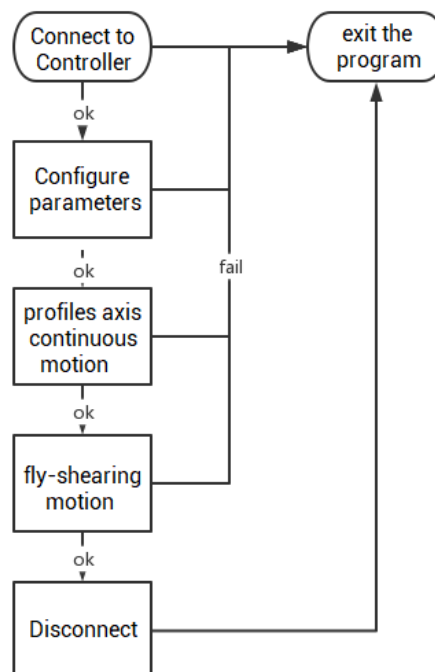
Fly-Shearing



- The profile continues to move, and the workbench stops at first.
- Until the profile continues to move for a certain distance, the workbench starts to accelerate.
- When the speed of the workbench is consistent with the profile, then switch on the tool to cut down
- The tool will back up after cutting, and the workbench starts to decelerate, then backs to the starting point.

[Repeat this process, to obtain the profile of the set length]

Example: the length of the profile is 4m, the bench motion distance is 1m, using axis 1 (basic axis = profile transmission) and axis 0 (following axis = shearing bench), and OUT0 is used to control the tool. The flying shear part program is as follows:



```
// test1.cpp: define the entry point of control panel application program
//
```

```
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"
```

```
void commandCheckHandler(const char *command, int ret)
```

```
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
        Sleep(2000);
        exit(0);
    }
}
```

```
int _tmain(int argc, _TCHAR* argv[])
{
```

```

char *ip_addr = (char *)"127.0.0.1"; //simulator IP address
//char *ip_addr = (char *)"192.168.0.11"; //controller IP address
ZMC_HANDLE handle = NULL; //link handle
int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
if (ERR_SUCCESS != ret)
{
    printf("Fail to Controller!\n");
    handle = NULL;
    Sleep(2000);
    return -1;
}
printf("Success to Controller!\n");

int FirstAxis = 0, SecondAxis = 1; //define axis 0, axis 1
int AType = 1; //set axis type
float DposValue = 0; //set DPOS
float SpeedValue = 3; //set the value of speed, profile running speed is 1m/s,60m/min
float AccelValue = 10; //set acceleration
float DecelValue = 10; //set deceleration
float UnitsValue = 10000; //set units value

ret = ZAux_Direct_SetAtype(handle, FirstAxis, AType); //set axis 0 type
commandCheckHandler("ZAux_Direct_SetAtype", ret);
ret = ZAux_Direct_SetAtype(handle, SecondAxis, AType); //set axis 1 type
commandCheckHandler("ZAux_Direct_SetAtype", ret);
ret = ZAux_Direct_SetSpeed( handle, FirstAxis, SpeedValue); //set axis 0 speed
commandCheckHandler("ZAux_Direct_SetSpeed", ret);
ret = ZAux_Direct_SetSpeed(handle, SecondAxis, SpeedValue); //set axis 1 speed
commandCheckHandler("ZAux_Direct_SetSpeed", ret);
ret = ZAux_Direct_SetDpos( handle, FirstAxis, DposValue); //set axis 0 DPOS
commandCheckHandler("ZAux_Direct_SetDpos", ret);
ret = ZAux_Direct_SetDpos(handle, SecondAxis, DposValue); //set axis 1 DPOS
commandCheckHandler("ZAux_Direct_SetDpos", ret);
ret = ZAux_Direct_SetUnits(handle, FirstAxis, UnitsValue); //set axis 0 UNITS
commandCheckHandler("ZAux_Direct_SetUnits", ret);
ret = ZAux_Direct_SetUnits(handle, SecondAxis, UnitsValue); //set axis 1 UNITS
commandCheckHandler("ZAux_Direct_SetUnits", ret);
ret = ZAux_Direct_SetAccel(handle, FirstAxis, AccelValue); //set axis 0 acceleration
commandCheckHandler("ZAux_Direct_SetAccel", ret);
ret = ZAux_Direct_SetAccel(handle, SecondAxis, AccelValue); //set axis 1 acceleration
commandCheckHandler("ZAux_Direct_SetAccel", ret);
ret = ZAux_Direct_SetDecel(handle, FirstAxis, DecelValue); //set axis 0 deceleration
commandCheckHandler("ZAux_Direct_SetDecel", ret);
ret = ZAux_Direct_SetDecel(handle, SecondAxis, DecelValue); //set axis 1 deceleration
commandCheckHandler("ZAux_Direct_SetDecel", ret);
ret = ZAux_Direct_SetSramp(handle, FirstAxis, 400); //set axis 0 Sramp
commandCheckHandler("ZAux_Direct_SetDecel", ret);
ret = ZAux_Direct_SetSramp(handle, SecondAxis, 400); //set axis 1 Sramp
commandCheckHandler("ZAux_Direct_SetDecel", ret);

ret = ZAux_Trigger( handle); //trigger the oscilloscope
commandCheckHandler("ZAux_Direct_SetDecel", ret);

ret = ZAux_Direct_Single_Vmove(handle, SecondAxis, 1); //profile continuous motion
commandCheckHandler("ZAux_Direct_Single_Vmove", ret);

int i;
for (i=0;i<3;i++)//fly-shearing three times
{
    ret = ZAux_Direct_Movelink( handle, FirstAxis, 0 , 1 , 0 , 0 , SecondAxis,8, 0); //before 1m of
the profile motion, the workbench is static.
    commandCheckHandler("ZAux_Direct_Movelink1", ret);
    ret = ZAux_Direct_Movelink( handle, FirstAxis, 0.4 , 0.8 , 0.8 , 0 , SecondAxis, 8,
0); //workbench is in acceleration stage: profile moves 0.8m, workbench accelerates 0.4m.

```

```

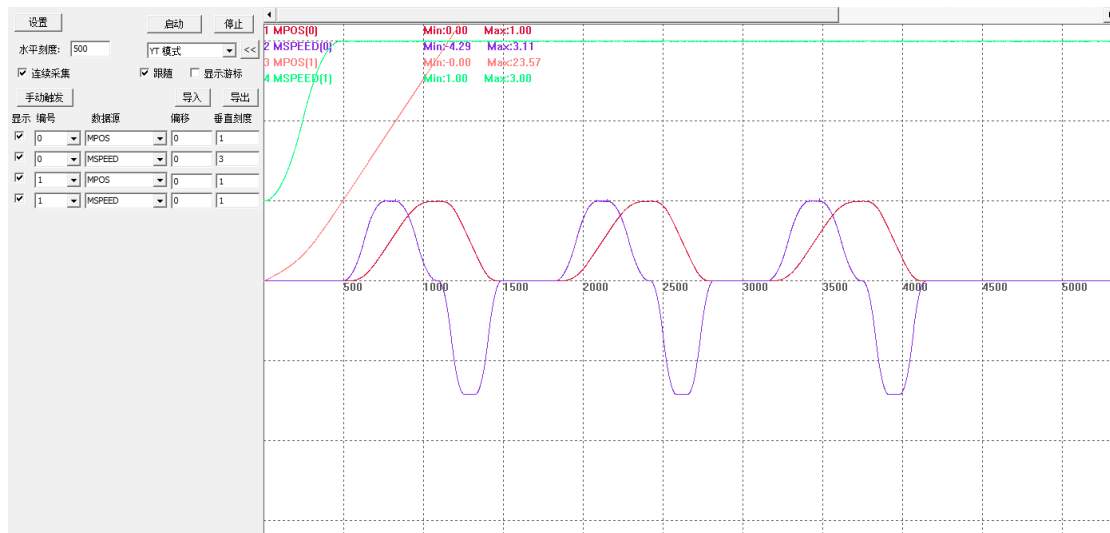
commandCheckHandler("ZAux_Direct_Movelink2", ret);
ret = ZAux_Direct_Movelink( handle, FirstAxis, 0.2 , 0.2 , 0 , 0 , SecondAxis, 8,
0);//synchronous follow 0.2m
commandCheckHandler("ZAux_Direct_Movelink3", ret);
ZAux_Direct_MoveOp2(handle, FirstAxis,0,1,1000);//operate OP to make tool down to cut, lift
back after 1s (the time needs to be calculated)
commandCheckHandler("ZAux_Direct_MoveOp2", ret);
ret = ZAux_Direct_Movelink( handle, FirstAxis, 0.4 , 0.8 , 0 , 0.8, SecondAxis, 8,
0);//workbench is in deceleration stage: profile moves 0.8m, workbench accelerates 0.4m.
commandCheckHandler("ZAux_Direct_Movelink2", ret);
ret = ZAux_Direct_Movelink( handle, FirstAxis, -1 , 1.2 , 0.5 , 0.5, SecondAxis, 8,
0);//workbench goes to starting point: profile moves 1.2m, workbench moves 1m inversely.
commandCheckHandler("ZAux_Direct_Movelink2", ret);
}

Sleep(5000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret); //if it executes successfully
printf("connection closed!\n");

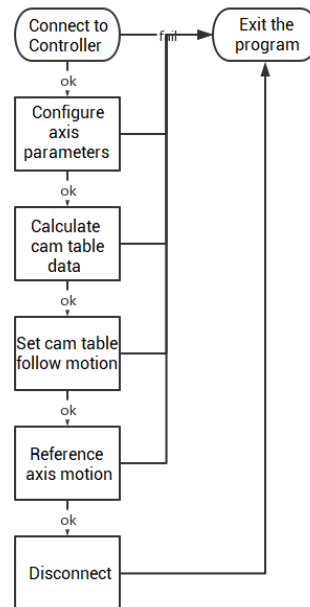
handle = NULL;
return 0;
}

```

➤ Waveform:



Electronic Cam Synchronous Motion (Follow Cam Table)



```
// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"
#include <math.h>
void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
        Sleep(2000);
        exit(0);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"127.0.0.1"; //simulator IP address
    //char *ip_addr = (char *)"192.168.0.11"; //controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to Controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to Controller!\n");

    int firstAxis = 0, secondAxis = 1; //define axis 0 and axis 1
    int AType = 1; //set axis type
    int DposValue = 0; //set DPOS
```

```

int UnitValue = 100;           //set pulse amount (UNITS)
int SpeedValue = 200;          //set speed
int AccelValue = 2000;         //set acceleration
int DecelValue = 2000;         //set deceleration

ret = ZAux_Direct_SetAtype(handle, firstAxis, AType);           //set axis 0 type
commandCheckHandler("ZAux_Direct_SetAtype", ret);
ret = ZAux_Direct_SetAtype(handle, secondAxis, AType);          //set axis 1 type
commandCheckHandler("ZAux_Direct_SetAtype", ret);
ret = ZAux_Direct_SetDpos(handle, firstAxis, DposValue);         //set axis 0 DPOS value
commandCheckHandler("ZAux_Direct_SetDpos", ret);
ret = ZAux_Direct_SetDpos(handle, secondAxis, DposValue);        //set axis 1 DPOS value
commandCheckHandler("ZAux_Direct_SetDpos", ret);
ZAux_Direct_SetUnits(handle, firstAxis, UnitValue);             //set axis 0 UNITS
commandCheckHandler("ZAux_Direct_SetUnits", ret);
ZAux_Direct_SetUnits(handle, secondAxis, UnitValue);            //set axis 1 UNITS
commandCheckHandler("ZAux_Direct_SetUnits", ret);
ZAux_Direct_SetSpeed(handle, firstAxis, SpeedValue);            //set axis 0 speed
commandCheckHandler("ZAux_Direct_SetSpeed", ret);
ZAux_Direct_SetSpeed(handle, secondAxis, SpeedValue);           //set axis 1 speed
commandCheckHandler("ZAux_Direct_SetSpeed", ret);
ZAux_Direct_SetAccel(handle, firstAxis, AccelValue);             //set axis 0 acceleration
commandCheckHandler("ZAux_Direct_SetAccel", ret);
ZAux_Direct_SetAccel(handle, secondAxis, AccelValue);           //set axis 1 acceleration
commandCheckHandler("ZAux_Direct_SetAccel", ret);
ZAux_Direct_SetDecel(handle, firstAxis, DecelValue);            //set axis 0 deceleration
commandCheckHandler("ZAux_Direct_SetDecel", ret);
ZAux_Direct_SetDecel(handle, secondAxis, DecelValue);           //set axis 1 deceleration
commandCheckHandler("ZAux_Direct_SetDecel", ret);

//{calculate Table data
float deg;
float rad;
float x;
int stepdeg = 2;
float PI = 3.1416;

uint32 IN_Status;        //receive state value of input
int piValue;              //read motion status of axis 1
int i,j;
int TableStartPoint = 10;           //Table starting position
int TableEndPoint = 10+1024-1;      //Table end position
float setTableValue[1024]; //cam table data

//write sine curve data
for (j=0;j<4;j++)
{
    for (i=0;i<256;i++)
    {
        setTableValue[i+j*256]=(sin(((2*PI)/34)*i))*100;
    }
}

ZAux_Direct_SetTable(handle,TableStartPoint,1024,setTableValue); //write cam table data into
//calculate Table data}

ret = ZAux_Trigger(handle);
commandCheckHandler("ZAux_Trigger", ret);

ret = ZAux_Direct_Cambox(handle, firstAxis, TableStartPoint, TableEndPoint, 100, 1500, 1, 2, 100);
//when reference axis 1 moves to position 100, following axis 0 opens cam table motion

ret = ZAux_Direct_Single_Move(handle, secondAxis, 3000);

```



```

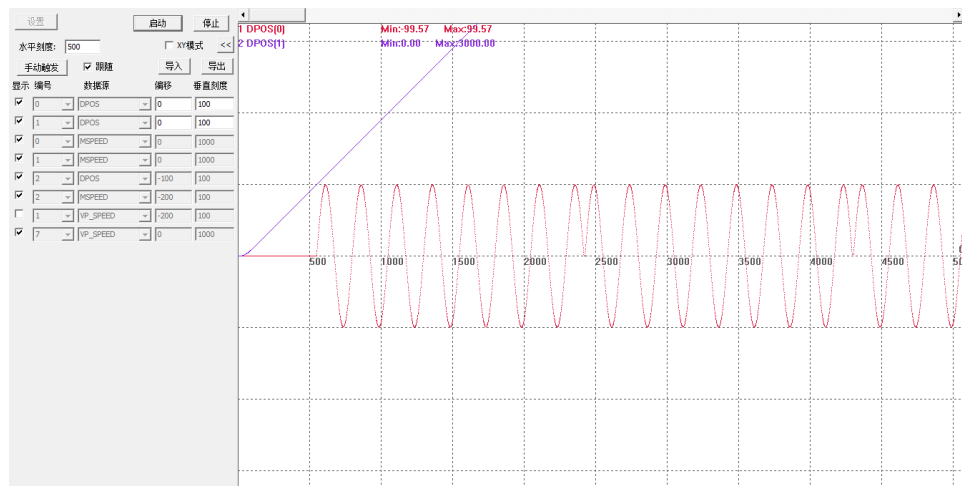
commandCheckHandler("ZAux_Direct_Single_Move", ret);

Sleep(5000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
printf("connection closed!\n");

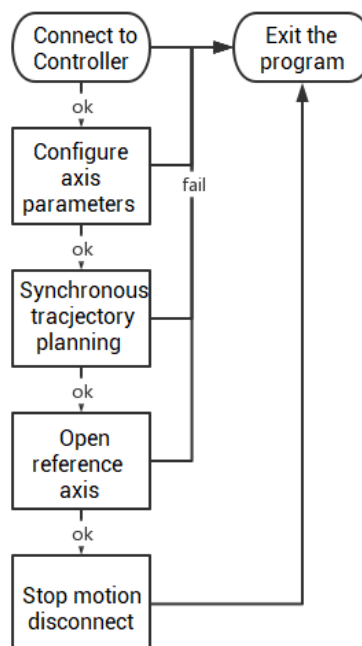
handle = NULL;
return 0;
}

```

➤ Waveform:



Electronic Cam Synchronous Motion (MOVESLINK)



```

// test1.cpp: define the entry point of control panel application program
//

```

```
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"
#include <math.h>
void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
        Sleep(2000);
        exit(0);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"127.0.0.1"; //simulator IP address
    //char *ip_addr = (char *)"192.168.0.11"; //controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to Controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to Controller!\n");

    int FirstAxis = 0, SecondAxis = 1; //define axis 0 and axis 1
    int AType = 1; //set axis type
    float DposValue = 0; //set DPOS
    float SpeedValue = 100; //set speed
    float AccelValue = 2000; //set acceleration
    float DecelValue = 2000; //set deceleration
    float UnitsValue = 100; //set Units value

    char psRespons[1024]; //used for array that is received controller model
    int length = 1024; //the controller side length must > received character string length
    float DistanceValue = 150;

    float fDistance = 50;
    float fLinkDis = 100;
    float startsp = 0;
    float endsp = 1;
    int iLinkaxis = SecondAxis;
    int ioption = 2;
    float flinkstartpos = 50;

    ret = ZAux_Direct_SetAtype(handle, FirstAxis, AType); //set axis 0 type
    commandCheckHandler("ZAux_Direct_SetAtype", ret);
    ret = ZAux_Direct_SetAtype(handle, SecondAxis, AType); //set axis 1 type
    commandCheckHandler("ZAux_Direct_SetAtype", ret);
    ret = ZAux_Direct_SetDpos(handle, FirstAxis, DposValue); //set axis 0 DPOS
    commandCheckHandler("ZAux_Direct_SetDpos", ret);
    ret = ZAux_Direct_SetDpos(handle, SecondAxis, DposValue); //set axis 1 DPOS
    commandCheckHandler("ZAux_Direct_SetDpos", ret);
    ret = ZAux_Direct_SetUnits(handle, FirstAxis, UnitsValue); //set axis 0 UNITS
    commandCheckHandler("ZAux_Direct_SetUnits", ret);
    ret = ZAux_Direct_SetUnits(handle, SecondAxis, UnitsValue); //set axis 1 UNITS
    commandCheckHandler("ZAux_Direct_SetUnits", ret);
}
```

```

ret = ZAux_Direct_SetAccel(handle, FirstAxis, AccelValue); //set axis 0 acceleration
commandCheckHandler("ZAux_Direct_SetAccel", ret);
ret = ZAux_Direct_SetAccel(handle, SecondAxis, AccelValue); //set axis 1 acceleration
commandCheckHandler("ZAux_Direct_SetAccel", ret);
ret = ZAux_Direct_SetDecel(handle, FirstAxis, DecelValue); //set axis 0 deceleration
commandCheckHandler("ZAux_Direct_SetDecel", ret);
ret = ZAux_Direct_SetDecel(handle, SecondAxis, DecelValue); //set axis 1 deceleration
commandCheckHandler("ZAux_Direct_SetDecel", ret);

ret = ZAux_Trigger(handle); //trigger the oscilloscope
commandCheckHandler("ZAux_Trigger", ret);

ret = ZAux_Direct_Moveslink(handle, 0, 250, 500, 0, 1, 1, 2, 1000); //when axis 1 is at position
1000-1500, axis 0 accelerates to synchronize with axis 1, the acceleration process motion is all 250.
commandCheckHandler("ZAux_Direct_Moveslink", ret);
ret = ZAux_Direct_Moveslink(handle, 0, 500, 500, 1, 1, 1, 2, 1500); //when axis 1 is at position
1500-2000, axis 0 synchronizes with axis 1 to move 500.
commandCheckHandler("ZAux_Direct_Moveslink", ret);
ret = ZAux_Direct_Moveslink(handle, 0, 250, 500, 1, 0, 1, 2, 2000); //when axis 1 is at position
2000-2500, axis 0 and axis 1 decrease to 0, the deceleration process all moves 250.
commandCheckHandler("ZAux_Direct_Moveslink", ret);

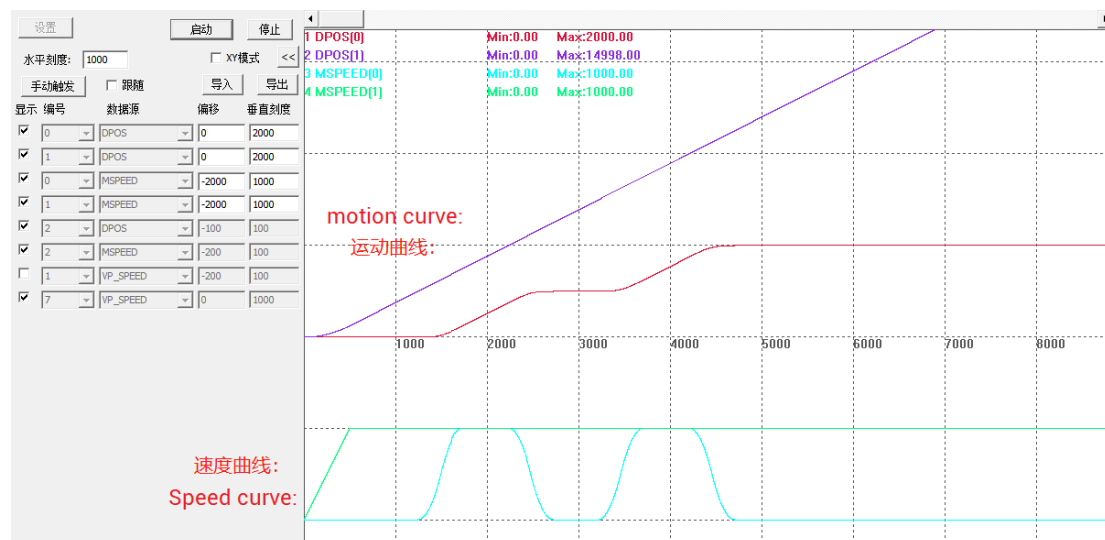
ret = ZAux_Direct_Moveslink(handle, 0, 250, 500, 0, 1, 1, 2, 3000); //when axis 1 is at position
3000-3500, axis 0 accelerates to synchronize with axis 1, the acceleration process motion is all 250.
commandCheckHandler("ZAux_Direct_Moveslink", ret);
ret = ZAux_Direct_Moveslink(handle, 0, 500, 500, 1, 1, 1, 2, 3500); //when axis 1 is at position
3500-4000, axis 0 synchronizes with axis 1 to move 500.
commandCheckHandler("ZAux_Direct_Moveslink", ret);
ret = ZAux_Direct_Moveslink(handle, 0, 250, 500, 1, 0, 1, 2, 4000); //when axis 1 is at position
4000-4500, axis 0 and axis 1 decrease to 0, the deceleration process all moves 250.
commandCheckHandler("ZAux_Direct_Moveslink", ret);

ZAux_Direct_Single_Vmove(handle, 1, 1); //open axis 1
Sleep(15000);
ZAux_Direct_Single_Cancel(handle, 1, 2); //stop axis 1
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret); //if it executes successfully
printf("connection closed!\n");

handle = NULL;
return 0;
}

```

➤ Waveform:



Belt Synchronous Follow Motion (MOVESYNC)

```
// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zaxdll2.h"
#include <math.h>
void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
        Sleep(2000);
        exit(0);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"127.0.0.1"; //simulator IP address
    //char *ip_addr = (char *)"192.168.0.11"; //controller IP address
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to Controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to Controller!\n");

    int FirstAxis = 0, SecondAxis = 1; //define axis 0 axis 1
    int AType = 1; //set axis type
    float DposValue = 0; //set DPOS
    float SpeedValue = 100; //set speed
    float AccelValue = 2000; //set acceleration
    float DecelValue = 2000; //set deceleration
    float UnitsValue = 100; //set Units

    ret = ZAux_Direct_SetAtype(handle, FirstAxis, AType); //set axis 0 type
    commandCheckHandler("ZAux_Direct_SetAtype", ret);
    ret = ZAux_Direct_SetAtype(handle, SecondAxis, AType); //set axis 1 type
    commandCheckHandler("ZAux_Direct_SetAtype", ret);
    ret = ZAux_Direct_SetDpos(handle, FirstAxis, DposValue); //set axis 0 DPOS
    commandCheckHandler("ZAux_Direct_SetDpos", ret);
    ret = ZAux_Direct_SetDpos(handle, SecondAxis, DposValue); //set axis 1 DPOS
    commandCheckHandler("ZAux_Direct_SetDpos", ret);
    ret = ZAux_Direct_SetUnits(handle, FirstAxis, UnitsValue); //set axis 0 UNITS
    commandCheckHandler("ZAux_Direct_SetUnits", ret);
    ret = ZAux_Direct_SetUnits(handle, SecondAxis, UnitsValue); //set axis 1 UNITS
    commandCheckHandler("ZAux_Direct_SetUnits", ret);
    ret = ZAux_Direct_SetAccel(handle, FirstAxis, AccelValue); //set axis 0 acceleration
    commandCheckHandler("ZAux_Direct_SetAccel", ret);
    ret = ZAux_Direct_SetAccel(handle, SecondAxis, AccelValue); //set axis 1 acceleration
    commandCheckHandler("ZAux_Direct_SetAccel", ret);
    ret = ZAux_Direct_SetDecel(handle, FirstAxis, DecelValue); //set axis 0 deceleration
    commandCheckHandler("ZAux_Direct_SetDecel", ret);
    ret = ZAux_Direct_SetDecel(handle, SecondAxis, DecelValue); //set axis 1 deceleration
```

```

commandCheckHandler("ZAux_Direct_SetDecel", ret);

ret = ZAux_Trigger(handle);           //trigger the oscilloscope
commandCheckHandler("ZAux_Trigger", ret);

ZAux_Direct_Single_Vmove(handle, 1,1); //open axis 1, simulate belt axis motion

ZAux_Direct_MoveWait(handle,0,"MPOS",1,1,500);//when axis 1 moves to 500, open
synchronous motion
float fpos = 500;
ZAux_Direct_MoveSync(handle,0,0,1000,1,1,&FirstAxis,&fpos);//the belt axis moves to 1000, the
following axis 0 moves to 500, and the speed has been synchronized, the synchronization time is
automatically calculated according to the acceleration and deceleration

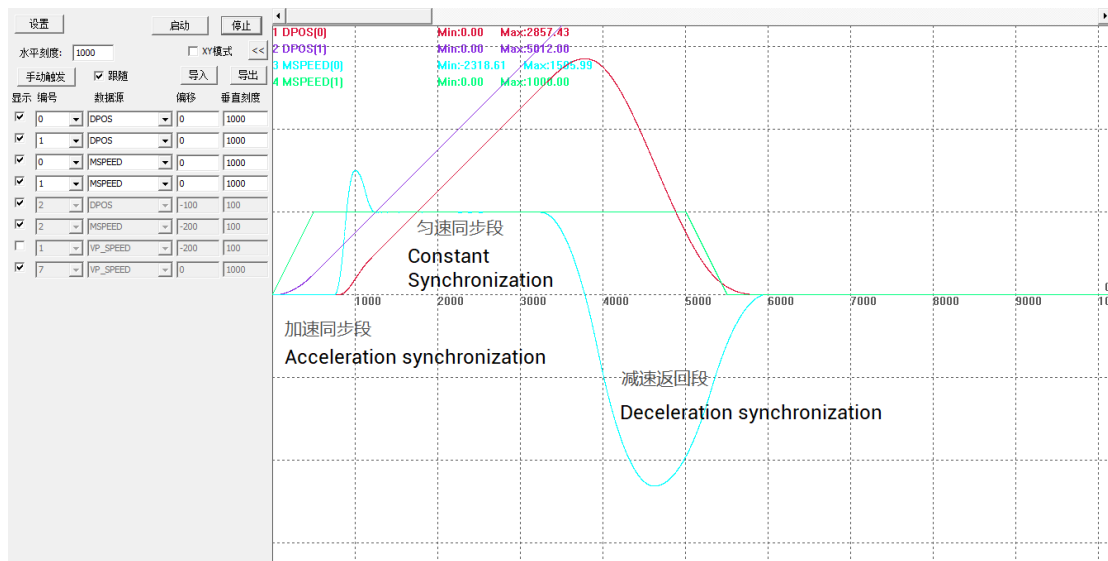
ZAux_Direct_MoveSync(handle,0,2000,1000,1,1,&FirstAxis,&fpos);//continue to synchronize 2s
fpos=0;
ZAux_Direct_MoveSync(handle,-1,0,300,1,1,&FirstAxis,&fpos);//following axis 0 stops
synchronization, returns to initial position

Sleep(5000);
ZAux_Direct_Single_Cancel(handle, 1,2);//stop axis 1
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
printf("connection closed!\n");

handle = NULL;
return 0;
}

```

➤ Waveform:



Chapter III Position Latch

3.1. High-Speed Latch

3.1.1. Instruction List

Commands	Descriptions
ZAux_Direct_Regist	Position latch
ZAux_Direct_CycleRegist	Continuous position latch
ZAux_Direct_SetCloseWin	Set the end coordinates range for latch triggering
ZAux_Direct_GetCloseWin	Read the end coordinates range for latch triggering
ZAux_Direct_GetRegPos	Read latch measurement position (MPOS)
ZAux_Direct_GetRegPosB	Read latch measurement position (MPOS)
ZAux_Direct_SetOpenWin	Set the starting coordinates range for latch triggering
ZAux_Direct_GetOpenWin	Read the starting coordinates range for latch triggering
ZAux_Direct_GetMark	Read encoder latch returned state
ZAux_Direct_GetMarkB	Read encoder latch B returned state

3.1.2. Principle & Description

➤ **Definition:**

Some Zmotion motion controllers support latch function, and there are encoder axis latch, bus axis latch, and even virtual axis latch, pulse axis latch.

- Encoder Axis Latch: most of controllers support that, for previous series controllers and cards, they only can use encoder axis latch.
- Virtual-Axis Latch & Pulse-Axis Latch: they are valid in ZMC4XX series controllers and above (latest version).
- EtherCAT Bus Axis Latch: it only can use drive latch, that is, using drive IO to achieve latch.
- RTEX Bus Axis Latch: it only can use controller latch.

➤ **Syntax:**

Use **ZAux_Direct_Regist** to latch axis' measurement feedback position.

When latching occurs, axis status `ZAux_Direct_GetMark` will become ON, at the same time, the latched position will be saved into `ZAux_Direct_GetRegPos`.

Each axis has R0, R1 and EZ signals to trigger latch action. When 2 signals are used, the latch value from the second signal will be used in `ZAux_Direct_GetMarkB` and `ZAux_Direct_GetRegPosB` to latch.

1) Single-time latch mode: `ZAux_Direct_Regist`

- ✓ When input signal R0 in controller is mapped into IN0 by default (the controller supports IN latch).
- ✓ When input signal R1 in controller is mapped into IN1 by default (the controller supports IN latch).
- ✓ When input signal R2 in controller is mapped into IN2 by default (the controller supports IN latch, it is valid when $IN > 2$).
- ✓ When input signal R3 in controller is mapped into IN3 by default (the controller supports IN latch, it is valid when $IN > 2$).

Value	Descriptions
1	Save absolute position in REG_POS when meet Z pulse rising edge.
2	Save absolute position in REG_POS when meet Z pulse falling edge.
3	Save absolute position in REG_POS when meet IN signal R0 rising edge.
4	Save absolute position in REG_POS when meet IN signal R0 falling edge.
6	Save absolute position in REG_POS when meet IN signal R0 rising edge. Save absolute position in REG_POSB when meet Z pulse rising edge.
7	Save absolute position in REG_POS when meet IN signal R0 rising edge. Save absolute position in REG_POSB when meet Z pulse falling edge.
8	Save absolute position in REG_POS when meet IN signal R0 falling edge. Save absolute position in REG_POSB when meet Z pulse rising edge.
9	Save absolute position in REG_POS when meet IN signal R0 falling edge. Save absolute position in REG_POSB when meet Z pulse falling edge.
10	Save absolute position in REG_POS when meet IN signal R0 rising edge. Save absolute position in REG_POSB when meet IN signal R1 rising edge.
11	Save absolute position in REG_POS when meet IN signal R0 rising edge. Save absolute position in REG_POSB when meet IN signal R1 falling edge.
12	Save absolute position in REG_POS when meet IN signal R0 falling edge. Save absolute position in REG_POSB when meet IN signal R1 rising edge.
13	Save absolute position in REG_POS when meet IN signal R0 falling edge. Save absolute position in REG_POSB when meet IN signal R1 falling edge.
<i>Below modes are only valid in firmware after 1500804, each latch channel is separate, and it supports 4-channel latch.</i>	
14	Save absolute position in REG_POSB when meet IN signal R1 rising edge
15	Save absolute position in REG_POSB when meet IN signal R1 falling edge
16	Save absolute position in REG_POSB when meet Z pulse rising edge.
17	Save absolute position in REG_POSB when meet Z pulse falling edge.
18	Save absolute position in REG_POSC when meet IN signal R2 rising edge

19	Save absolute position in REG_POSC when meet IN signal R2 falling edge
20	Save absolute position in REG_POSD when meet IN signal R3 rising edge
21	Save absolute position in REG_POSD when meet IN signal R3 falling edge
Notes: For pulse axis type, usually it uses R0, R1 and Z pulse signals to latch. For bus axis type, it uses R2 and R3 to latch.	

2) **Continuous Latch: ZAux_Direct_CycleRegist** (ZMC_HANDLE handle,int iaxis, int imode,int iTabStart,int iTabNum)

Mode + 100 means continuous latch, and latched data are saved in TABLE register.

2 channels are latched continuously and separately, then to achieve continuous latch of up and down edge.

ECI: valid in firmware above 20150829

ZMC4XX controllers: valid in firmware above 20170523

■ **Parameters:**

- handle: connection handle
- iaxis: controller axis
- imode: latch method
- iTabStart: continuous latched values are saved in TABLE, the first TABLE element saves the number of latched, followings save latched coordinates, the max storage numbers = numes-1, and it will cycle-write if exceeds limit.
- iTabNum: occupied number of TABLE

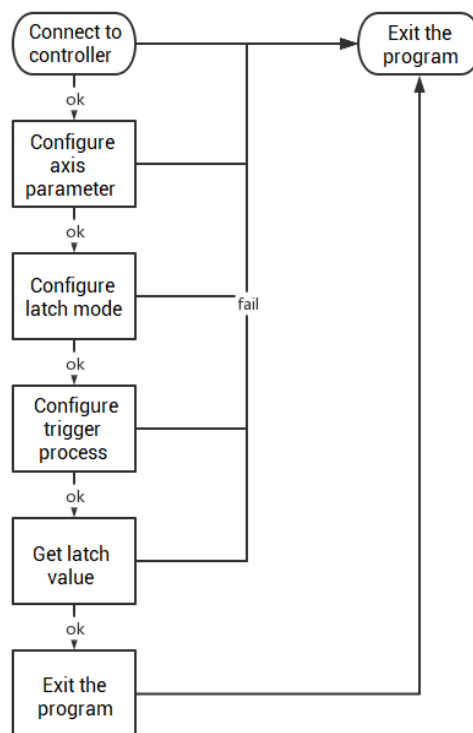
Value	Descriptions
101	Save absolute position in REG_POS when meet Z pulse rising edge.
102	Save absolute position in REG_POS when meet Z pulse falling edge.
103	Save absolute position in REG_POS when meet IN signal R0 rising edge.
104	Save absolute position in REG_POS when meet IN signal R0 falling edge.
114	Save absolute position in REG_POSB when meet IN signal R1 rising edge
115	Save absolute position in REG_POSB when meet IN signal R1 falling edge
116	Save absolute position in REG_POSB when meet Z pulse rising edge.
117	Save absolute position in REG_POSB when meet Z pulse falling edge.
123	Save absolute position in REG_POSB when meet IN signal R0 rising edge.
124	Save absolute position in REG_POSB when meet IN signal R0 falling edge.
133	Save absolute position in REG_POS when meet IN signal R0 rising edge. Next time, save when meeting falling edge. Switch between rising edge and falling edge in cycle.
134	Save absolute position in REG_POS when meet IN signal R0 falling edge. Next time, save when meeting rising edge. Switch between rising edge and falling edge in cycle.
135	Save absolute position in REG_POSB when meet IN signal R1 rising edge.

	Next time, save when meeting falling edge. Switch between rising edge and falling edge in cycle.
136	Save absolute position in REG_POSB when meet IN signal R1 falling edge. Next time, save when meeting rising edge. Switch between rising edge and falling edge in cycle.

3.1.3. Routines

Position Latch

Axis 0 moves continuously, when IN0 senses the signal, latching axis 0 encoder position value at high speed.



```
// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"
#include <math.h>
void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
        Sleep(2000);
        exit(0);
    }
}
```

```
int _tmain(int argc, _TCHAR* argv[])
{
    //char *ip_addr = (char *)"127.0.0.1"; //simulator IP address
    char *ip_addr = (char *)"192.168.0.11"; //real controller IP address is 192.168.0.11 by default
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to Controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to Controller!\n");

    int iaxis = 0;
    float DposValue = 0;
    float MposValue = 0;
    int mode = 3; //wait for the absolute position of R0 rising edge
    int piValue;
    float pfValue;
    ret = ZAux_Direct_SetDpos( handle, iaxis, DposValue); //clear axis 0 dpos to 0
    commandCheckHandler("ZAux_Direct_SetDpos", ret);

    ret = ZAux_Direct_SetMpos( handle, iaxis, MposValue); //clear axis 0 mpos to 0
    commandCheckHandler("ZAux_Direct_SetMpos", ret);

    ret = ZAux_Direct_Single_Vmove( handle, iaxis, 1); //axis 0 moves forward
    commandCheckHandler("ZAux_Direct_Single_Move", ret);

    ret = ZAux_Direct_Regist( handle, iaxis, mode); //set single-time latch, latch mode: trigger
    //latching when IN0 is rising edge
    commandCheckHandler("ZAux_Direct_Regist", ret);

    ret = ZAux_Direct_MoveWait(handle, 1, "Mpos", iaxis, 1, 432); //in axis 1 buffer, add the block (here
    //axis 1 buffer is occupied) that executes the next command until axis 0 MPOS is larger than 432
    commandCheckHandler("ZAux_Direct_Regist", ret);

    /*after MPOS equals to 432, write rising edge into axis 1 buffer, this routine connects ZMC
    series controllers, low level is valid level, so the rising edge is formed when IN(0) is set from 1 to 0. (if
    it is ECI series, it is opposite, from 0 to 1)

    ret = ZAux_Direct_MoveOp(handle, 1, 0, 1); //write that OUT0 is set to 1 into axis 1 buffer (here,
    the used controller has been connected OUT0 to IN0 directly)
    commandCheckHandler("ZAux_Direct_Regist", ret);
    ret = ZAux_Direct_MoveOp(handle, 1, 0, 0); //write that OUT0 is set to 0 into axis 1 buffer (here,
    the used controller has been connected OUT0 to IN0 directly)
    commandCheckHandler("ZAux_Direct_Regist", ret);

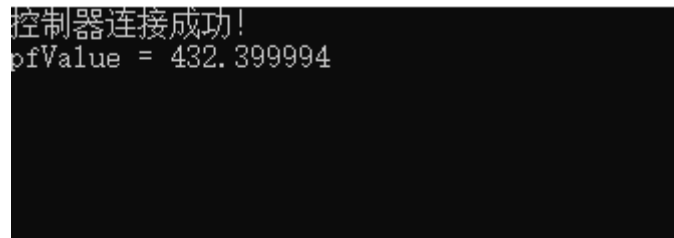
    while (1)
    {
        Sleep(2000);
        ZAux_Direct_GetMark(handle, iaxis, &piValue);

        if (piValue == -1) //trigger latching
        {
            ret = ZAux_Direct_GetRegPos(handle, 0, &pfValue);
            commandCheckHandler("ZAux_Direct_GetRegPos", ret);
            printf("pfValue = %f\n", pfValue);
            break;
        }
    }
}
```

```
ret = ZAux_Direct_Single_Cancel( handle, 0, 2); //stop moving
commandCheckHandler("ZAux_Direct_Single_Cancel", ret);
Sleep(5000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
printf("connection closed!\n");

handle = NULL;
return 0;
}
```

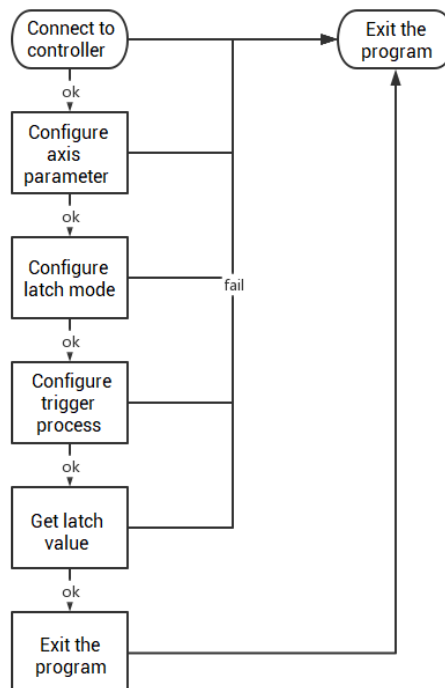
➤ Output Result: Success to Connect!



```
控制器连接成功!
pfValue = 432.399994
```

Continuous Position Latch

Axis 0 moves continuously, when IN0 senses the signal, latching axis 0 encoder position value at high speed continuously.



```
// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"
#include <math.h>
void commandCheckHandler(const char *command, int ret)
{
```

```
if (ret)//it is not 0, fail
{
    printf("%s fail!return code is %d\n", command, ret);
    Sleep(2000);
    exit(0);
}

int _tmain(int argc, _TCHAR* argv[])
{
    //char *ip_addr = (char *)"127.0.0.1"; //simulator IP address
    char *ip_addr = (char *)"192.168.0.11"; //real controller IP address is 192.168.0.11 by default
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to Controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to Controller!\n");

    int iaxis = 0;
    float DposValue = 0;
    float MposValue = 0;
    int mode = 103; //wait for absolute position of R0 rising edge
    float pfValues[10];
    float pfValue;
    int AType = 1; //set axis type
    int UnitValue = 100; //set UNITS
    int SpeedValue = 200; //set speed
    int AccelValue = 2000; //set acceleration
    int DecelValue = 2000; //set deceleration

    ret = ZAux_Direct_SetAtype(handle, iaxis, AType); //set axis 0 type
    commandCheckHandler("ZAux_Direct_SetAtype", ret);

    ZAux_Direct_SetUnits(handle, iaxis, UnitValue); //set axis 0 pulse amount
    commandCheckHandler("ZAux_Direct_SetUnits", ret);

    ZAux_Direct_SetSpeed(handle, iaxis, SpeedValue); //set axis 0 speed
    commandCheckHandler("ZAux_Direct_SetSpeed", ret);

    ZAux_Direct_SetAccel(handle, iaxis, AccelValue); //set axis 0 acceleration
    commandCheckHandler("ZAux_Direct_SetAccel", ret);

    ZAux_Direct_SetDecel(handle, iaxis, DecelValue); //set axis 0 deceleration
    commandCheckHandler("ZAux_Direct_SetDecel", ret);
    ret = ZAux_Direct_SetDpos( handle, iaxis, DposValue); //clear axis 0 dpos to 0
    commandCheckHandler("ZAux_Direct_SetDpos", ret);

    ret = ZAux_Direct_SetMpos( handle, iaxis, MposValue); //clear axis 0 mpos to 0
    commandCheckHandler("ZAux_Direct_SetMpos", ret);

    ret = ZAux_Direct_Single_Vmove( handle, iaxis, 1); //axis 0 moves forward
    commandCheckHandler("ZAux_Direct_Single_Move", ret);

    ret = ZAux_Direct_CycleRegist( handle, iaxis, mode, 0, 100); //set continuous latch, latch mode:
    trigger latching when IN0 is in rising edge, latch value saves table(0)-table(99)
    commandCheckHandler("ZAux_Direct_CycleRegist", ret);

    ret = ZAux_Direct_MoveWait(handle, 1, "Mpos", iaxis, 1, 432); //in axis 1 buffer, add the block (here
    axis 1 buffer is occupied) that executes the next command until axis 0 MPOS is larger than 432
```

```
commandCheckHandler("ZAux_Direct_MoveWait", ret);

/*after MPOS equals to 432, write rising edge into axis 1 buffer, this routine connects ZMC
series controllers, low level is valid level, so the rising edge is formed when IN(0) is set from 1 to 0. (if
it is ECI series, it is opposite, from 0 to 1)*/
int i;

for (i=0;i<10;i++)//trigger 1 latch every 1 second, trigger 10 times
{
    ret = ZAux_Direct_MoveOp(handle, 1,0,1);//write that OUT0 is set to 1 into axis 1 buffer
    (here, the used controller has been connected OUT0 to IN0 directly)
    commandCheckHandler("ZAux_Direct_MoveOp ", ret);
    ret = ZAux_Direct_MoveOp(handle, 1,0,0);///write that OUT0 is set to 0 into axis 1 buffer
    (here, the used controller has been connected OUT0 to IN0 directly)
    commandCheckHandler("ZAux_Direct_MoveOp ", ret);
    ret = ZAux_Direct_MoveDelay(handle, 1,1000);///write that OUT0 is set to 0 into axis 1
    buffer (here, the used controller has been connected OUT0 to IN0 directly)
    commandCheckHandler("ZAux_Direct_MoveDelay", ret);
}

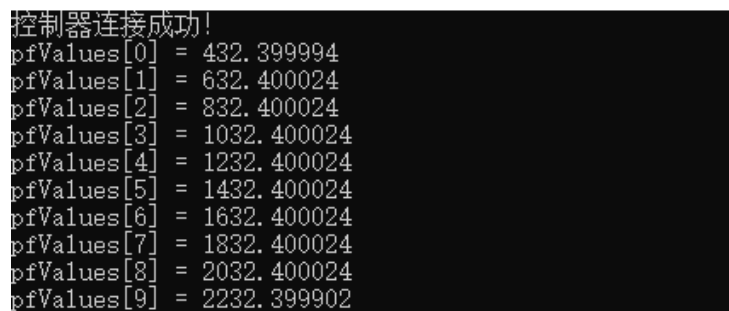
while (1)
{
    Sleep(2000);
    ZAux_Direct_GetTable(handle, 0, 1,&pfValue);

    if (pfValue == 10)//exit the cycle when 10 times latches are triggered.
    {
        ret = ZAux_Direct_GetTable(handle, 1, 10,pfValues);
        commandCheckHandler("ZAux_Direct_GetTable", ret);
        for (i=0;i<10;i++)//trigger 1 latch every 1 second, trigger 10 times
        {
            printf("pfValues[%d] = %f\n", i,pfValues[i]);
        }
        break;
    }
}

ret = ZAux_Direct_Single_Cancel( handle, 0, 2);//stop moving
commandCheckHandler("ZAux_Direct_Single_Cancel", ret);
Sleep(5000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
printf("connection closed!\n");

handle = NULL;
return 0;
}
```

3) Output Result: Success to Connect!



```
控制器连接成功!
pfValues[0] = 432.399994
pfValues[1] = 632.400024
pfValues[2] = 832.400024
pfValues[3] = 1032.400024
pfValues[4] = 1232.400024
pfValues[5] = 1432.400024
pfValues[6] = 1632.400024
pfValues[7] = 1832.400024
pfValues[8] = 2032.400024
pfValues[9] = 2232.399902
```

Chapter IV Position Comparison Output

4.1. Position Comparison Output

4.1.1. Instruction List

Commands	Descriptions
ZAux_Direct_Pswitch	Soft position comparison output
ZAux_Direct_HwPswitch	Hard position comparison output
ZAux_Direct_GetHwPswitchBuff	Hard position comparison output remain buffers
ZAux_Direct_HwPswitch2	Bus hard position comparison output (this is also valid in pulse axis and bus axis)
ZAux_Direct_HwPswitch2_2	Hard position comparison output 2
ZAux_Direct_HwTimer	Hard timer is used to restore the electric level after one certain of time after hard position comparison output.
ZAux_Direct_MoveHwPswitch2	Hard position comparison output, in buffer way
ZAux_Direct_MoveHwTimer	Hard timer is used to restore the electric level after one certain of time after hard position comparison output, in buffer way
ZAux_Direct_SetHwPs2AxisNum	Set axis No. of hard comparison output
ZAux_Direct_GetHwPs2AxisNum	Get axis No. of hard comparison output
ZAux_Direct_GetHwPs2Counts	Get compared points of hard comparison output

4.1.2. Principle & Description

1. Soft Position Comparison Output

Operate related OUT according to axis position, when several *ZAux_Direct_Pswitch* are used to operate one output, please make the sequence by No.

2. Precision Position Output

Only controllers that support the hardware comparison output function can achieve precise output, command: *ZAux_Direct_SetParam* (handle, "AXIS_ZSET", axis No., mode)

The instruction sets whether to enable precise output.

For normal output, please wait for one controller cycle before execution. For the precise output, it can respond within one pulse sent by the motor, which greatly improves the dispensing accuracy. And the *ZAux_Direct_SetParam*

(handle,"MOVEOP_DELAY",axis No., delay time) can be used to adjust the response time (advance or delay).

Mode:

Bit 0	1- current motion speed uses interpolation speed by default. 0- current motion speed uses single-axis speed.
Bit 1	1- use ZAux_Direct_MoveOp precision output function. 0- use ZAux_Direct_MoveOp original method.
Bit 4	1- for axis that is with encoder, use encoder position ZAux_Direct_MoveOp precision method.

For multi-axis interpolation, set master axis:

```
ZAux_Direct_SetParam(handle,"AXIS_ZSET",0,19) //open axis 0 precision output
```

3. Hard Position Comparison Output

Through setting feedback position table, write into TABLE. In motion, compare values in table to operate corresponding outputs.

There are 1024 HW buffers, and 1024 HW commands can be called continuously.

After calling HW instruction, which won't be affected by following coordinates. And please ensure called coordinates in TABLE are correct. Therefore, it's better to modify coordinates manually to avoid random clashing caused by HW commands and coordinates cyclic auto modification.

Due to auto-cycle-coordinates configuration is not controlled by program, HW is in the front or behind can't be determined, which means coordinates in TABLE also can't be determined.

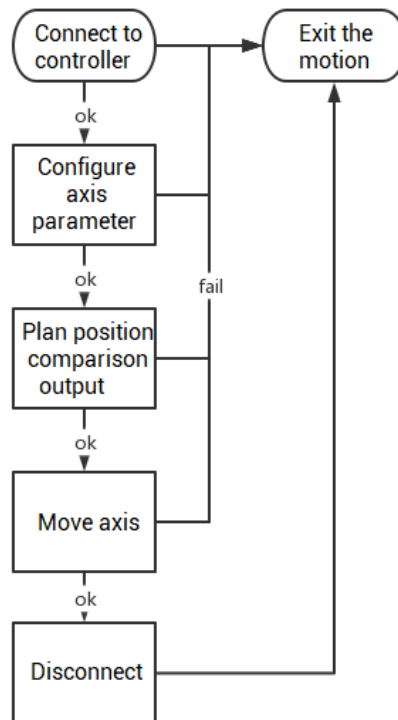
For bus axes, ZAux_Direct_HwPswitch2 command should be used.

(Note: it is valid in the controller that supports PSO function, ZMC 4xx series products & the latest firmware).

4.1.3. Routines

Soft Position Comparison Output

Axis 0 moves continuously, then opens OP0 at the position of 100-200.



```

// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"
#include <math.h>
void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
        Sleep(2000);
        exit(0);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1"; //simulator IP address
    //char *ip_addr = (char *) "192.168.0.11"; //real controller IP address is 192.168.0.11 by default
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to Controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to Controller!\n");

    int iaxis = 0;
    float DposValue = 0;
    float MposValue = 0;
    int AType = 1; //set axis type
    int UnitValue = 100; //set UNITS
  
```



```

int SpeedValue = 100;           //set speed
int AccelValue = 2000;          //set acceleration
int DecelValue = 2000;          //set deceleration

ret = ZAux_Direct_SetAtype(handle, iaxis, AType); //set axis 0 type
commandCheckHandler("ZAux_Direct_SetAtype", ret);
ZAux_Direct_SetUnits(handle, iaxis, UnitValue); //set axis 0 pulse amount
commandCheckHandler("ZAux_Direct_SetUnits", ret);
ZAux_Direct_SetSpeed(handle, iaxis, SpeedValue); //set axis speed
commandCheckHandler("ZAux_Direct_SetSpeed", ret);
ZAux_Direct_SetAccel(handle, iaxis, AccelValue); //set axis 0 acceleration
commandCheckHandler("ZAux_Direct_SetAccel", ret);
ZAux_Direct_SetDecel(handle, iaxis, DecelValue); //set axis 0 deceleration
commandCheckHandler("ZAux_Direct_SetDecel", ret);
ret = ZAux_Direct_SetDpos( handle, iaxis, DposValue); //clear axis 0 dpos to 0
commandCheckHandler("ZAux_Direct_SetDpos", ret);
ret = ZAux_Direct_SetMpos( handle, iaxis, MposValue); //clear axis 0 mpos to 0
commandCheckHandler("ZAux_Direct_SetMpos", ret);

float GetValue;
uint32 piValue;
ret = ZAux_Direct_Pswitch(handle,0,1,iaxis,0,1,100,200); //operate the output in the range of
DPOS 0-201 of axis 0
commandCheckHandler("ZAux_Direct_Pswitch",ret);

ZAux_Trigger(handle); //open the oscilloscope
ret = ZAux_Direct_Single_Vmove( handle, iaxis, 1); //axis 0 moves forward
commandCheckHandler("ZAux_Direct_Single_Move", ret);

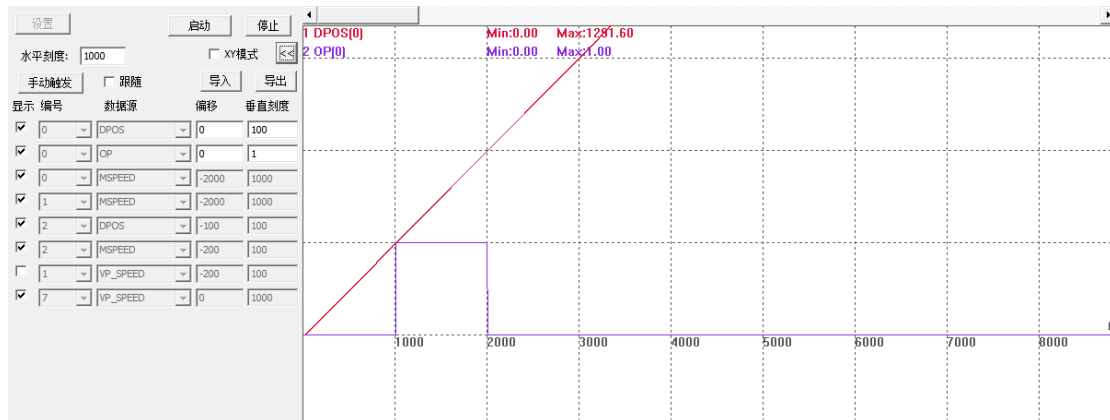
while (1)
{
    Sleep(200);
    ret = ZAux_Direct_GetOp(handle,iaxis,&piValue); //get OUT1 status, 1 = ON, 0 = OFF.
    commandCheckHandler("ZAux_Direct_GetDA",ret);
    printf("OP status piValue=%d\n", piValue);
    ret = ZAux_Direct_GetDpos(handle,iaxis,&GetValue); //get current position
    printf("position value GetValue=%f\n", GetValue);
    if (GetValue>300) //exit print when position is bigger than 300
    {
        break;
    }
}

ret = ZAux_Direct_Single_Cancel( handle, 0, 2); //stop moving
commandCheckHandler("ZAux_Direct_Single_Cancel", ret);
Sleep(5000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ; //if it executes successfully
printf("connection closed!\n");

handle = NULL;
return 0;
}

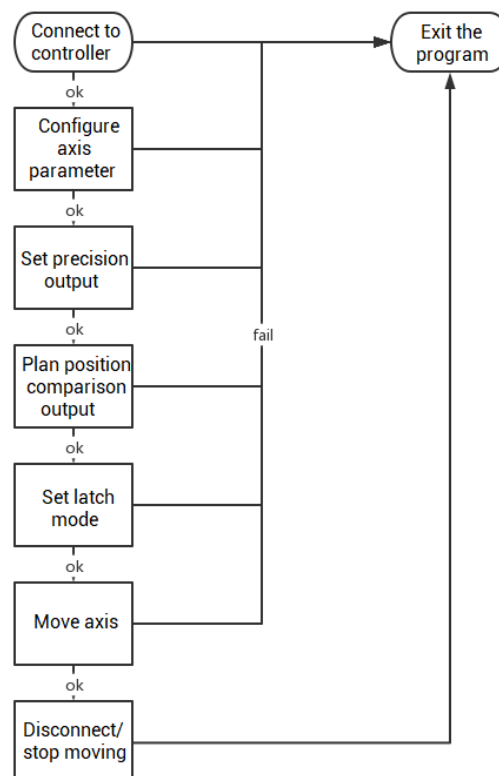
```

➤ Waveform:



Fly-Shooting (ZAux_Direct_HwPswitch)

Assume axis 0 is the conveyor belt axis of continuous motion, it needs to trigger camera to shoot when axis 0 is at the position of 100, 150, 200, 250 and 300, and to latch actual position value of encoder when shooting.



```
// test1.cpp: define the entry point of control panel application program
//
```

```
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
```

```
#include "zauxdll2.h"
#include <math.h>
void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
        Sleep(2000);
        exit(0);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    //char *ip_addr = (char *)"127.0.0.1"; //simulator IP address
    char *ip_addr = (char *)"192.168.0.11"; //real controller IP address is 192.168.0.11 by default
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to Controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to Controller!\n");

    int iaxis = 0;
    float DposValue = 0;
    float MposValue = 0;
    int AType = 1; //set axis type
    int UnitValue = 100; //set UNITS
    int SpeedValue = 50; //set speed
    int AccelValue = 2000; //set acceleration
    int DecelValue = 2000; //set deceleration

    ret = ZAux_Direct_SetAtype(handle, iaxis, AType); //set axis 0 type
    commandCheckHandler("ZAux_Direct_SetAtype", ret);
    ZAux_Direct_SetUnits(handle, iaxis, UnitValue); //set axis 0 units
    commandCheckHandler("ZAux_Direct_SetUnits", ret);
    ZAux_Direct_SetSpeed(handle, iaxis, SpeedValue); //set axis 0 speed
    commandCheckHandler("ZAux_Direct_SetSpeed", ret);
    ZAux_Direct_SetAccel(handle, iaxis, AccelValue); //set axis 0 acceleration
    commandCheckHandler("ZAux_Direct_SetAccel", ret);
    ZAux_Direct_SetDecel(handle, iaxis, DecelValue); //set axis 0 deceleration
    commandCheckHandler("ZAux_Direct_SetDecel", ret);
    ret = ZAux_Direct_SetDpos( handle, iaxis, DposValue); //clear axis 0 dpos to 0
    commandCheckHandler("ZAux_Direct_SetDpos", ret);
    ret = ZAux_Direct_SetMpos( handle, iaxis, MposValue); //clear axis 0 mpos to 0
    commandCheckHandler("ZAux_Direct_SetMpos", ret);

    ret = ZAux_Direct_SetParam(handle, "AXIS_ZSET", 0, 19); //open axis 0 precision output
    commandCheckHandler("ZAux_Direct_SetParam", ret);

    float TableValue[10]={100,101,150,151,200,201,250,251,300,301}; //set the output points
    ret = ZAux_Direct_SetTable(handle, 0, 10, TableValue ); //set Table value
    commandCheckHandler("ZAux_Direct_SetTable", ret);

    ret = ZAux_Direct_HwPswitch(handle, 0, 2, 0, 0, 0); //stop and delete uncompleted comparison
    point (unfinished comparison points should be deleted)
    commandCheckHandler("ZAux_Direct_HwPswitch", ret);
    ret = ZAux_Direct_HwPswitch(handle, 0, 1, 1, 0, 0, 10); //configure position comparison output
    commandCheckHandler("ZAux_Direct_HwPswitch", ret);
}
```

```

ret = ZAux_Direct_Regist( handle, iaxis, 4); //set single-time latch, latch mode: trigger latching
when IN0 is in falling edge
commandCheckHandler("ZAux_Direct_Regist", ret);

ZAux_Trigger(handle); //open oscilloscope
ret = ZAux_Direct_Single_Vmove( handle, iaxis, 1); //axis 0 moves forward
commandCheckHandler("ZAux_Direct_Single_Move", ret);

float GetValue, pfValue;
int piValue;

while (1)
{
    Sleep(200);
    ZAux_Direct_GetMark(handle, iaxis, &piValue); //get whether the latch is triggered

    if (piValue == -1) //trigger latching
    {
        ZAux_Direct_GetRegPos(handle, 0, &pfValue);
        printf("pfValue = %f\n", pfValue);
        ret = ZAux_Direct_Regist( handle, iaxis, 4); //set single-time latch, latch mode: trigger
        latching when IN0 is in falling edge
        commandCheckHandler("ZAux_Direct_Regist", ret);
    }

    ret = ZAux_Direct_GetDpos(handle, iaxis, &GetValue); //get position value

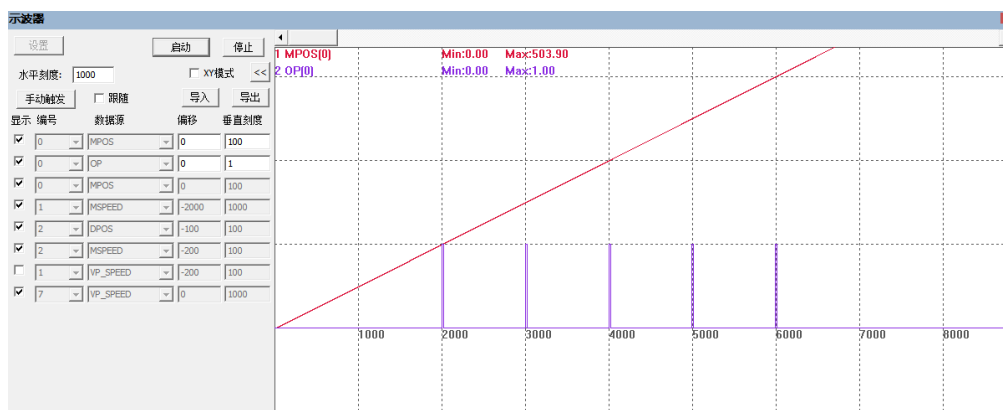
    if (GetValue > 500) //exit when position is larger than 500
    {
        printf("position value GetValue = %f\n", GetValue);
        break;
    }
}

ret = ZAux_Direct_Single_Cancel( handle, 0, 2); //stop moving
commandCheckHandler("ZAux_Direct_Single_Cancel", ret);
Sleep(5000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret); //if it executes successfully
printf("connection closed!\n");

handle = NULL;
return 0;
}

```

➤ Waveform:



Fly-Shooting (1 axis + multi-camera) (ZAux_Direct_HwPswitch2)

Assume axis 0 is the conveyor belt axis of continuous motion, it needs to trigger camera 1, camera 2, camera 3 and camera 4 to shoot. Camera 1, camera 2, camera 3 and camera 4 correspond to OUT0, OUT1, OUT2 and OUT3 respectively.

```
// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"
#include <math.h>
void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail! return code is %d\n", command, ret);
        Sleep(2000);
        exit(0);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    //char *ip_addr = (char *) "127.0.0.1"; // simulator IP address
    char *ip_addr = (char *) "192.168.0.11"; // real controller IP address is 192.168.0.11 by default
    ZMC_HANDLE handle = NULL; // link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); // connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to Controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to Controller!\n");

    int iaxis = 0;
    float DposValue = 0;
    float MposValue = 0;
    int AType = 1; // set axis type
    int UnitValue = 100; // set UNITS
    int SpeedValue = 50; // set speed
    int AccelValue = 2000; // set acceleration
    int DecelValue = 2000; // set deceleration

    ret = ZAux_Direct_SetAtype(handle, iaxis, AType); // set axis 0 type
    commandCheckHandler("ZAux_Direct_SetAtype", ret);
    ZAux_Direct_SetUnits(handle, iaxis, UnitValue); // set axis 0 UNITS
    commandCheckHandler("ZAux_Direct_SetUnits", ret);
    ZAux_Direct_SetSpeed(handle, iaxis, SpeedValue); // set axis 0 speed
    commandCheckHandler("ZAux_Direct_SetSpeed", ret);
    ZAux_Direct_SetAccel(handle, iaxis, AccelValue); // set axis 0 acceleration
    commandCheckHandler("ZAux_Direct_SetAccel", ret);
    ZAux_Direct_SetDecel(handle, iaxis, DecelValue); // set axis 0 deceleration
    commandCheckHandler("ZAux_Direct_SetDecel", ret);
    ret = ZAux_Direct_SetDpos(handle, iaxis, DposValue); // clear axis 0 dpos to 0
    commandCheckHandler("ZAux_Direct_SetDpos", ret);
}
```

```

ret = ZAux_Direct_SetMpos( handle, iaxis, MposValue); //clear axis 0 mpos to 0
commandCheckHandler("ZAux_Direct_SetMpos", ret);

ZAux_Direct_SetParam(handle,"HW_PS2AXISNUM",1,0);//use axis 1 buffer, compare axis 0 pos
commandCheckHandler("ZAux_Direct_SetParam",ret);

ZAux_Direct_SetParam(handle,"HW_PS2AXISNUM",2,0);//use axis 2 buffer, compare axis 0 pos
commandCheckHandler("ZAux_Direct_SetParam",ret);

ZAux_Direct_SetParam(handle,"HW_PS2AXISNUM",3,0);//use axis 3 buffer, compare axis 0 pos
commandCheckHandler("ZAux_Direct_SetParam",ret);

uint32 uValue=0;

ZAux_Direct_SetOutMulti(handle,0,3,&uValue);//initialize OUT0-3 status
commandCheckHandler("ZAux_Direct_SetOutMulti",ret);

float TableValue[8]={100,101,150,151,200,201,250,251};//set output points
ret =ZAux_Direct_SetTable(handle,0,8,TableValue );//set Table value
commandCheckHandler("ZAux_Direct_SetTable",ret);

ret = ZAux_Direct_HwPswitch2(handle,0,2,0,0,0,0,0); //stop and delete uncompleted
comparison point (unfinished comparison points should be deleted)
commandCheckHandler("ZAux_Direct_HwPswitch",ret);

ret = ZAux_Direct_HwPswitch2(handle,0,1,0,1,0,1,-1,0);//set position comparison output, axis 0,
mode 1, OUT0, output 1, output positions are saved into table(0)-table(1), -1: no direction used
commandCheckHandler("ZAux_Direct_HwPswitch",ret);

ret = ZAux_Direct_HwPswitch2(handle,1,2,0,0,0,0,0,0); //stop and delete uncompleted
comparison point (unfinished comparison points should be deleted)
commandCheckHandler("ZAux_Direct_HwPswitch",ret);

ret = ZAux_Direct_HwPswitch2(handle,1,1,1,1,2,3,-1,0);//set position comparison output, mode
1, axis 1, OUT1, output 1, output positions are saved into table(2)-table(3), -1: no direction used
commandCheckHandler("ZAux_Direct_HwPswitch",ret);

ret = ZAux_Direct_HwPswitch2(handle,2,2,0,0,0,0,0,0); //stop and delete uncompleted
comparison point (unfinished comparison points should be deleted)
commandCheckHandler("ZAux_Direct_HwPswitch",ret);

ret = ZAux_Direct_HwPswitch2(handle,2,1,2,1,4,5,-1,0);//set position comparison output, axis 2,
mode 1, OUT2, output 1, output positions are saved into table(4)-table(5), -1: no direction used
commandCheckHandler("ZAux_Direct_HwPswitch",ret);

ret = ZAux_Direct_HwPswitch2(handle,3,2,0,0,0,0,0,0); //stop and delete uncompleted
comparison point (unfinished comparison points should be deleted)
commandCheckHandler("ZAux_Direct_HwPswitch",ret);

ret = ZAux_Direct_HwPswitch2(handle,3,1,3,1,6,7,-1,0);//set position comparison output, axis 3,
mode 1, OUT1, output 1, output positions are saved into table(6)-table(7), -1: no direction used
commandCheckHandler("ZAux_Direct_HwPswitch",ret);

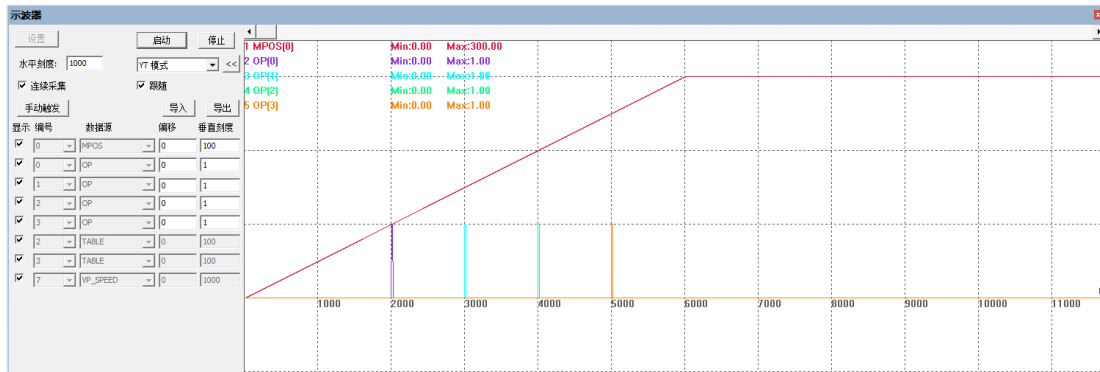
ZAux_Trigger(handle);//open the oscilloscope
ret = ZAux_Direct_Single_Move( handle, iaxis, 300);//axis 0 moves forward
commandCheckHandler("ZAux_Direct_Single_Move", ret);

Sleep(5000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
printf("connection closed!\n");

handle = NULL;
return 0;

```

➤ Waveform:



Fly-Shooting (2D Comparison Output)

There are 2 axes (axis 0 & axis 1), using them to build one plane coordinate system, and the camera is triggered to take pictures at positions (100, 100), (200, 200), (300, 300). It is valid in firmware version 4 series 170706 and above.

```
// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"
#include <math.h>
void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail! return code is %d\n", command, ret);
        Sleep(2000);
        exit(0);
    }
}

/*parameters:
    mode        comparison mode of 25, 26, 2D comparison
    Opnum        corresponding output
    Opstate      the output state of the first comparison point
    maxerr       pulse deviation of comparison position of each axis, it starts to compare after
entering the deviation range.
    num          the number of comparison points saved in TABLE
    tablepos     Table No. where saves the first comparison point coordinates

    when it is used together with hwtimer, hwtimer can be adjusted dynamically.
    ModePara1: pulse time
    ModePara2: the number of pulses
    ModePara3: pulse period
*/
int32 __stdcall ZAux_Direct_HwPswitch2_2D(ZMC_HANDLE handle, int Axisnum, int Mode, int
Opnum, int Opstate, int maxerr, int num, int tablepos, float ModePara1=0, float ModePara2=0, float
ModePara3=0)
```

```
{
    if(0 > Axisnum || Axisnum > MAX_AXIS_AUX)
    {
        return ERR_AUX_PARAERR;
    }
    char cmdbuff[2048];
    char cmdbuffAck[2048];
    //generate the command
    switch(Mode)
    {
        case 25:
            sprintf(cmdbuff, "HW_PSWITCH2(%d,%d,%d,%d,%d,%d) AXIS(%d)", Mode, Opnum, Opstate,
maxerr,num,tablepos,Axisnum);
            break;
        case 26:
            sprintf(cmdbuff, "HW_PSWITCH2(%d,%d,%d,%d,%d,%d,%f,%f,%f) AXIS(%d)", Mode, Opnum,
Opstate, maxerr,num,tablepos,ModePara1,ModePara2,ModePara3,Axisnum);
            break;
    }
    //call the command to execute the function
    return ZAux_Execute(handle, cmdbuff, cmdbuffAck,2048);
}
int _tmain(int argc, _TCHAR* argv[])
{
    //char *ip_addr = (char *)"127.0.0.1"; //simulator IP address
    char *ip_addr = (char *)"192.168.0.11"; //real controller address is 192.168.0.11 by default
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to Controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to Controller!\n");

    int iaxis = 0;
    float DposValue = 0;
    float MposValue = 0;
    int AType = 1; //set axis type
    int UnitValue = 100; //set UNITS
    int SpeedValue = 50; //set speed
    int AccelValue = 2000; //set acceleration
    int DecelValue = 2000; //set deceleration

    ret = ZAux_Direct_SetAtype(handle, iaxis, AType); //set axis 0 type
    commandCheckHandler("ZAux_Direct_SetAtype", ret);
    ZAux_Direct_SetUnits(handle, iaxis, UnitValue); //set axis 0 UNITS
    commandCheckHandler("ZAux_Direct_SetUnits", ret);
    ZAux_Direct_SetSpeed(handle, iaxis, SpeedValue); //set axis 0 speed
    commandCheckHandler("ZAux_Direct_SetSpeed", ret);
    ZAux_Direct_SetAccel(handle, iaxis, AccelValue); //set axis 0 acceleration
    commandCheckHandler("ZAux_Direct_SetAccel", ret);
    ZAux_Direct_SetDecel(handle, iaxis, DecelValue); //set axis 0 deceleration
    commandCheckHandler("ZAux_Direct_SetDecel", ret);
    ret = ZAux_Direct_SetDpos( handle, 0, 0); //clear axis 0 dpos to 0
    commandCheckHandler("ZAux_Direct_SetDpos", ret);
    ret = ZAux_Direct_SetMpos( handle, 0, 0); //clear axis 0 mpos to 0
    commandCheckHandler("ZAux_Direct_SetMpos", ret);

    ret = ZAux_Direct_SetAtype(handle, 1, AType); //set axis 1 type
    commandCheckHandler("ZAux_Direct_SetAtype", ret);
}
```



```

ZAux_Direct_SetUnits(handle, 1, UnitValue);           //set axis 1 UNITS
commandCheckHandler("ZAux_Direct_SetUnits", ret);
ZAux_Direct_SetSpeed(handle, 1, SpeedValue);          //set axis 1 speed
commandCheckHandler("ZAux_Direct_SetSpeed", ret);
ZAux_Direct_SetAccel(handle, 1, AccelValue);          //set axis 1 acceleration
commandCheckHandler("ZAux_Direct_SetAccel", ret);
ZAux_Direct_SetDecel(handle, 1, DecelValue);          //set axis 1 deceleration
commandCheckHandler("ZAux_Direct_SetDecel", ret);
ret = ZAux_Direct_SetDpos( handle, 1, DposValue); //clear axis 1 dpos to 0
commandCheckHandler("ZAux_Direct_SetDpos", ret);
ret = ZAux_Direct_SetMpos( handle, 1, MposValue); //clear axis 1 mpos to 0
commandCheckHandler("ZAux_Direct_SetMpos", ret);

ZAux_Direct_SetOp(handle,0,0); //initialize OUT0 state
commandCheckHandler("ZAux_Direct_SetOp",ret);

float TableValue[12]={100,100,102,102,200,200,202,202,300,300,302,302}; //planbit output, each
2 table saves one point
ret =ZAux_Direct_SetTable(handle,10,12,TableValue ); //set Table value
commandCheckHandler("ZAux_Direct_SetTable",ret);

ret = ZAux_Direct_HwPswitch2(handle,0,2,0,0,0,0,0); //stop and delete uncompleted
comparison point (unfinished comparison points should be deleted)
commandCheckHandler("ZAux_Direct_HwPswitch2",ret);
ret = ZAux_Direct_HwPswitch2(handle,1,2,0,0,0,0,0); //stop and delete uncompleted
comparison point (unfinished comparison points should be deleted)
commandCheckHandler("ZAux_Direct_HwPswitch2",ret);

ret = ZAux_Direct_HwPswitch2_2D(handle,0,25,0,1,10,6,10);
commandCheckHandler("ZAux_Direct_HwPswitch2_2D",ret);

int iaxislist[2]={0,1};
float dposlist[2]={350,350};

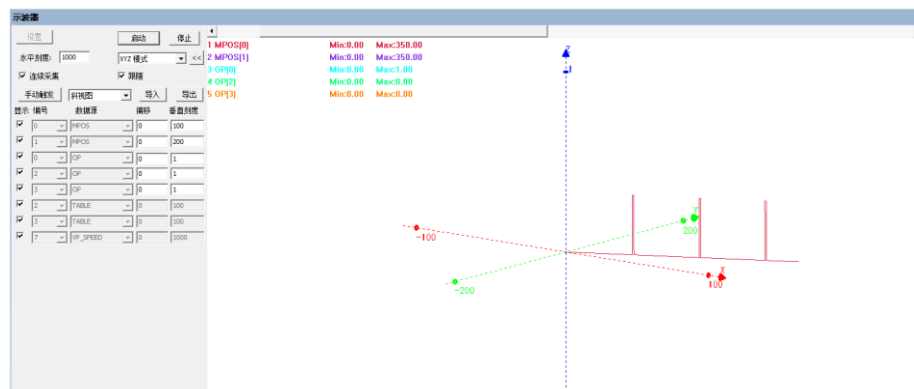
ZAux_Trigger(handle); //open the oscilloscope
ret = ZAux_Direct_Move( handle,2,iaxislist, dposlist); //axis 0 moves forward
commandCheckHandler("ZAux_Direct_Move", ret);

Sleep(15000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ; //if it executes successfully
printf("connection closed!\n");

handle = NULL;
return 0;
}

```

➤ Waveform:



Vector Position Comparison Output

```
// test1.cpp: define the entry point of control panel application program
//physical wiring is required, waveform will appear by connecting OUT0 to IN0.

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"
#include <math.h>
void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
        Sleep(2000);
        exit(0);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    //char *ip_addr = (char *)"127.0.0.1"; //simulator IP address
    char *ip_addr = (char *)"192.168.0.147"; //real controller address is 192.168.0.11 by default
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to Controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to Controller!\n");

    int iaxis[2] = {0,1};
    float DposValue = 0;
    float MposValue = 0;
    int AType = 1; //set axis type
    int UnitValue = 100; //set UNITS
    int SpeedValue = 50; //set speed
    int AccelValue = 2000; //set acceleration
    int DecelValue = 2000; //set deceleration

    ret = ZAux_Direct_SetAtype(handle, 0, AType); //set axis 0 type
    commandCheckHandler("ZAux_Direct_SetAtype", ret);
    ZAux_Direct_SetUnits(handle, 0, UnitValue); //set axis 0 UNITS
    commandCheckHandler("ZAux_Direct_SetUnits", ret);
    ZAux_Direct_SetSpeed(handle, 0, SpeedValue); //set axis 0 speed
    commandCheckHandler("ZAux_Direct_SetSpeed", ret);
    ZAux_Direct_SetAccel(handle, 0, AccelValue); //set axis 0 acceleration
    commandCheckHandler("ZAux_Direct_SetAccel", ret);
    ZAux_Direct_SetDecel(handle, 0, DecelValue); //set axis 0 deceleration
    commandCheckHandler("ZAux_Direct_SetDecel", ret);
    ret = ZAux_Direct_SetDpos( handle, 0, DposValue); //clear axis 0 dpos to 0
    commandCheckHandler("ZAux_Direct_SetDpos", ret);
    ret = ZAux_Direct_SetMpos( handle, 0, MposValue); //clear axis 0 mpos to 0
    commandCheckHandler("ZAux_Direct_SetMpos", ret);
}
```

```

ret = ZAux_Direct_SetAtype(handle, 1, AType);           //set axis 1 type
commandCheckHandler("ZAux_Direct_SetAtype", ret);
ZAux_Direct_SetUnits(handle, 1, UnitValue);             //set axis 1 UNITS
commandCheckHandler("ZAux_Direct_SetUnits", ret);
ZAux_Direct_SetSpeed(handle, 1, SpeedValue);            //set axis 1 speed
commandCheckHandler("ZAux_Direct_SetSpeed", ret);
ZAux_Direct_SetAccel(handle, 1, AccelValue);            //set axis 1 acceleration
commandCheckHandler("ZAux_Direct_SetAccel", ret);
ZAux_Direct_SetDecel(handle, 1, DecelValue);            //set axis 1 deceleration
commandCheckHandler("ZAux_Direct_SetDecel", ret);
ret = ZAux_Direct_SetDpos( handle, 1, DposValue); //clear axis 1 dpos to 0
commandCheckHandler("ZAux_Direct_SetDpos", ret);
ret = ZAux_Direct_SetMpos( handle, 1, MposValue); //clear axis 1 mpos to 0
commandCheckHandler("ZAux_Direct_SetMpos", ret);

ret = ZAux_Direct_SetParam( handle, "VECTOR_MOVED",0,0); // 'set vector starting position
commandCheckHandler("ZAux_Direct_SetParam", ret);

float TableValue[10]={100,101,200,201,300,301,400,401,498,499}; //
ret =ZAux_Direct_SetTable(handle,0,10,TableValue ); //set Table value
commandCheckHandler("ZAux_Direct_SetTable",ret);

ret = ZAux_Direct_HwPswitch2(handle,0,2,0,0,0,0,0); //stop and delete uncompleted
comparison point (unfinished comparison points should be deleted)
commandCheckHandler("ZAux_Direct_HwPswitch2",ret);

ret = ZAux_Direct_HwPswitch2(handle,0,3,0,1,0,10,0,0); //configure position comparison output,
master axis is axis 0, mode 3, OUT0, output status 1
commandCheckHandler("ZAux_Direct_HwPswitch2",ret);

ZAux_Trigger(handle); //open the oscilloscope
float pos[2]={300,400};

ret = ZAux_Direct_Move( handle, 2,iaxis,pos); //axis 0 moves forward
commandCheckHandler("ZAux_Direct_Move", ret);

Sleep(10000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ; //if it executes successfully
printf("connection closed!\n");

handle = NULL;
return 0;
}

```

➤ Waveform:



Period Comparison Mode (Distance Reset)

```
// test1.cpp: define the entry point of control panel application program
// this routine uses ZMC4 series controllers, and physical wiring is required, waveform will appear by
// connecting OUT0 to IN0

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zaxdll2.h"
#include <math.h>
void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
        Sleep(2000);
        exit(0);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    //char *ip_addr = (char *)"127.0.0.1"; //simulator IP address
    char *ip_addr = (char *)"192.168.0.11"; //real controller IP address is 192.168.0.11 by default
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to Controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to Controller!\n");

    int iaxis[2] = {0,1};
    float DposValue = 0;
    float MposValue = 0;
    int AType = 1; //set axis type
    int UnitValue = 100; //set UNITS
    int SpeedValue = 50; //set speed
    int AccelValue = 2000; //set acceleration
    int DecelValue = 2000; //set deceleration

    ret = ZAux_Direct_SetAtype(handle, 0, AType); //set axis 0 type
    commandCheckHandler("ZAux_Direct_SetAtype", ret);
    ZAux_Direct_SetUnits(handle, 0, UnitValue); //set axis 0 UNITS
    commandCheckHandler("ZAux_Direct_SetUnits", ret);
    ZAux_Direct_SetSpeed(handle, 0, SpeedValue); //set axis 0 speed
    commandCheckHandler("ZAux_Direct_SetSpeed", ret);
    ZAux_Direct_SetAccel(handle, 0, AccelValue); //set axis 0 acceleration
    commandCheckHandler("ZAux_Direct_SetAccel", ret);
    ZAux_Direct_SetDecel(handle, 0, DecelValue); //set axis 0 deceleration
    commandCheckHandler("ZAux_Direct_SetDecel", ret);
    ret = ZAux_Direct_SetDpos( handle, 0, DposValue); //clear axis 0 dpos to 0
    commandCheckHandler("ZAux_Direct_SetDpos", ret);
    ret = ZAux_Direct_SetMpos( handle, 0, MposValue); //clear axis 0 mpos to 0
```

```

commandCheckHandler("ZAux_Direct_SetMpos", ret);

ret = ZAux_Direct_SetAtype(handle, 1, AType); //set axis 1 type
commandCheckHandler("ZAux_Direct_SetAtype", ret);
ZAux_Direct_SetUnits(handle, 1, UnitValue); //set axis 1 UNITS
commandCheckHandler("ZAux_Direct_SetUnits", ret);
ZAux_Direct_SetSpeed(handle, 1, SpeedValue); //set axis 1 speed
commandCheckHandler("ZAux_Direct_SetSpeed", ret);
ZAux_Direct_SetAccel(handle, 1, AccelValue); //set axis 1 acceleration
commandCheckHandler("ZAux_Direct_SetAccel", ret);
ZAux_Direct_SetDecel(handle, 1, DecelValue); //set axis 1 deceleration
commandCheckHandler("ZAux_Direct_SetDecel", ret);
ret = ZAux_Direct_SetDpos( handle, 1, DposValue); //clear axis 1 dpos to 0
commandCheckHandler("ZAux_Direct_SetDpos", ret);
ret = ZAux_Direct_SetMpos( handle, 1, MposValue); //clear axis 1 mpos to 0
commandCheckHandler("ZAux_Direct_SetMpos", ret);

ret = ZAux_Direct_SetParam( handle, "VECTOR_MOVED", 0, 0); // 'set vector starting position
commandCheckHandler("ZAux_Direct_SetParam", ret);

ret = ZAux_Direct_HwPswitch2(handle, 0, 2, 0, 0, 0, 0, 0); //stop and delete uncompleted
comparison point (unfinished comparison points should be deleted)
commandCheckHandler("ZAux_Direct_HwPswitch2", ret);
//start comparing from position 100, compare 4 times, the period distance is 100, output valid
distance 20.

ret = ZAux_Direct_HwPswitch2(handle, 0, 5, 0, 1, 100, 4, 100, 20); //configure position comparison
output, master axis is axis 0, mode 5
commandCheckHandler("ZAux_Direct_HwPswitch2", ret);

ZAux_Trigger(handle); //open the oscilloscope
float pos[2] = {300, 400};

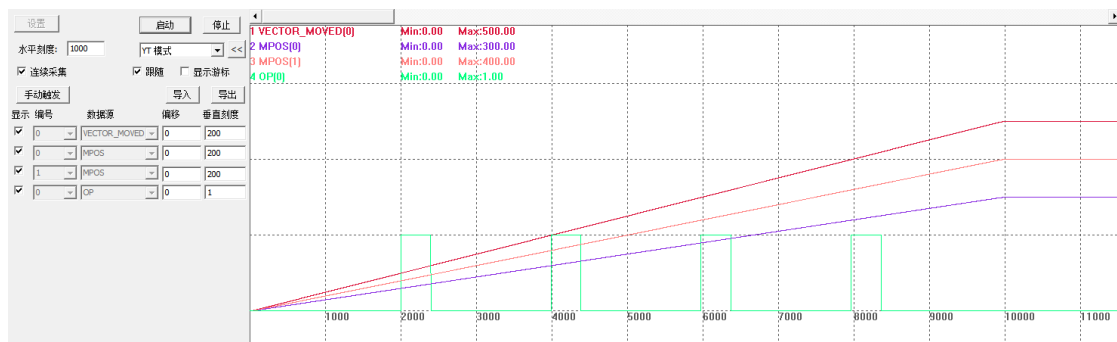
ret = ZAux_Direct_Move( handle, 2, iaxis, pos); //axis 0 moves forward
commandCheckHandler("ZAux_Direct_Move", ret);

Sleep(10000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret); //if it executes successfully
printf("connection closed!\n");

handle = NULL;
return 0;
}

```

➤ Waveform:



Period Comparison Mode (Time Reset)

```
// test1.cpp: define the entry point of control panel application program
// this routine uses ZMC4 series controllers, and physical wiring is required, waveform will appear
// by connecting OUT0 to IN0

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zaxdll2.h"
#include <math.h>
void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
        Sleep(2000);
        exit(0);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    //char *ip_addr = (char *)"127.0.0.1"; //simulator IP address
    char *ip_addr = (char *)"192.168.0.147"; //real controller address is 192.168.0.11 by default
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to Controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to Controller!\n");

    int iaxis[2] = {0,1};
    float DposValue = 0;
    float MposValue = 0;
    int AType = 1; //set axis type
    int UnitValue = 100; //set UNITS
    int SpeedValue = 50; //set speed
    int AccelValue = 2000; //set acceleration
    int DecelValue = 2000; //set deceleration

    ret = ZAux_Direct_SetAtype(handle, 0, AType); //set axis 0 type
    commandCheckHandler("ZAux_Direct_SetAtype", ret);
    ZAux_Direct_SetUnits(handle, 0, UnitValue); //set axis 0 UNITS
    commandCheckHandler("ZAux_Direct_SetUnits", ret);
    ZAux_Direct_SetSpeed(handle, 0, SpeedValue); //set axis 0 speed
    commandCheckHandler("ZAux_Direct_SetSpeed", ret);
    ZAux_Direct_SetAccel(handle, 0, AccelValue); //set axis 0 acceleration
    commandCheckHandler("ZAux_Direct_SetAccel", ret);
    ZAux_Direct_SetDecel(handle, 0, DecelValue); //set axis 0 deceleration
    commandCheckHandler("ZAux_Direct_SetDecel", ret);
    ret = ZAux_Direct_SetDpos( handle, 0, DposValue); //clear axis 0 dpos to 0
    commandCheckHandler("ZAux_Direct_SetDpos", ret);
    ret = ZAux_Direct_SetMpos( handle, 0, MposValue); //clear axis 0 mpos to 0
    commandCheckHandler("ZAux_Direct_SetMpos", ret);

    ret = ZAux_Direct_SetAtype(handle, 1, AType); //set axis 1 type
```

```

commandCheckHandler("ZAux_Direct_SetAtype", ret);
ZAux_Direct_SetUnits(handle, 1, UnitValue); //set axis 1 UNITS
commandCheckHandler("ZAux_Direct_SetUnits", ret);
ZAux_Direct_SetSpeed(handle, 1, SpeedValue); //set axis 1 speed
commandCheckHandler("ZAux_Direct_SetSpeed", ret);
ZAux_Direct_SetAccel(handle, 1, AccelValue); //set axis 1 acceleration
commandCheckHandler("ZAux_Direct_SetAccel", ret);
ZAux_Direct_SetDecel(handle, 1, DecelValue); //set axis 1 deceleration
commandCheckHandler("ZAux_Direct_SetDecel", ret);
ret = ZAux_Direct_SetDpos( handle, 1, DposValue); //clear axis 1 dpos to 0
commandCheckHandler("ZAux_Direct_SetDpos", ret);
ret = ZAux_Direct_SetMpos( handle, 1, MposValue); //clear axis 1 mpos to 0
commandCheckHandler("ZAux_Direct_SetMpos", ret);

ret = ZAux_Direct_SetParam( handle, "VECTOR_MOVED", 0, 0); //set vector starting position
commandCheckHandler("ZAux_Direct_SetParam", ret);

float TableValue[10]={100,101,200,201,300,301,400,401,498,499}; //plan bit output point
ret = ZAux_Direct_SetTable(handle, 0, 10, TableValue ); //set Table value
commandCheckHandler("ZAux_Direct_SetTable", ret);

ret = ZAux_Direct_HwPswitch2(handle, 0, 2, 0, 0, 0, 0, 0); //stop and delete uncompleted
comparison point (unfinished comparison points should be deleted)
commandCheckHandler("ZAux_Direct_HwPswitch2", ret);

ret = ZAux_Direct_HwPswitch2(handle, 0, 6, 0, 1, 100, 3, 100, 0); //configure position comparison
output, master axis 0, mode 3, OUT0, output status 1, start to compare from position 100, compare
3 times, the period distance is 100, the output valid time is determined according to HW_TIMER.
commandCheckHandler("ZAux_Direct_HwPswitch2", ret);

ret = ZAux_Direct_HwTimer(handle, 2, 1000000, 500000, 1, 0, 0); //period is 1s, when output
becomes ON, after 500ms, it becomes OFF.
commandCheckHandler("ZAux_Direct_HwTimer", ret);

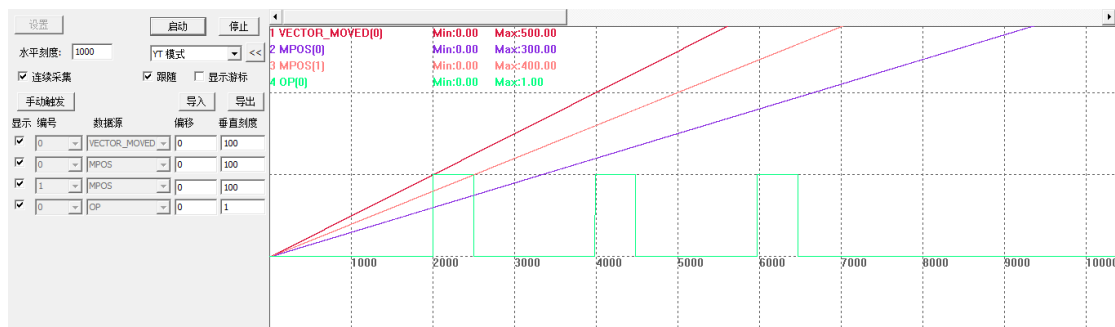
ZAux_Trigger(handle); //open the oscilloscope
float pos[2]={300,400};
ret = ZAux_Direct_Move( handle, 2, iaxis, pos); //vector position moves 500
commandCheckHandler("ZAux_Direct_Move", ret);

Sleep(10000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) //if it executes successfully
printf("connection closed!\n");

handle = NULL;
return 0;
}

```

4) Waveform:



Chapter V PT Motion

5.1. PT Motion

5.1.1. Instruction List

Commands	Descriptions
ZAux_Direct_MultiMovePt	Multiple relative PT motions
ZAux_Direct_MultiMovePtAbs	Multiple absolute PT motions
ZAux_Direct_MultiMovePvt	Multiple relative PVT motions

5.1.2. Principle & Description

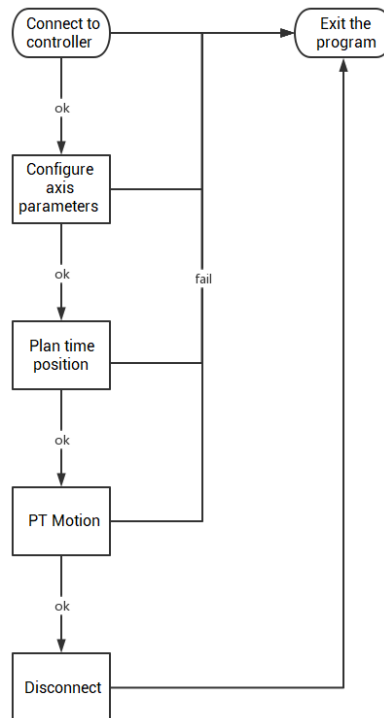
- 1) One certain of distance that will be moved by motor in set time.
- 2) The acceleration, speed and deceleration of PT movement are all planned according to the set time and position.
- 3) Generally, the PC calculates the corresponding coordinates every cycle, and then transmits them to the controller.
- 4) Motion speed = (moving distance / time length) * 1000 units/ms.
- 5) Do not move a large distance in a very short time, the pulse frequency will be too high, and the motor will stall. You could make them as several small segments and send them repeatedly.

Note: when using this command, fast deceleration or deceleration speed need to be configured, otherwise, it can't stop if encounters abnormal situation.

5.1.3. Routines

PT Motion

Axis 0 moves from -100 to 0 in the first 1s, then it moves to 100 starting from 1s.
Axis 1 moves to -100, and finally axis 0 and axis 1 return to absolute position 0.



```

// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"
#include <math.h>
#define PI 3.1415
void commandCheckHandler(const char *command, int ret)
{
    if (ret)//it is not 0, fail
    {
        printf("%s fail!return code is %d\n", command, ret);
        Sleep(2000);
        exit(0);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *)"127.0.0.1"; //simulator IP address
    //char *ip_addr = (char *)"192.168.1.11"; //real controller IP address is 192.168.0.11 by default
    ZMC_HANDLE handle = NULL; //link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
    if (ERR_SUCCESS != ret)
    {
        printf("Fail to Controller!\n");
        handle = NULL;
        Sleep(2000);
        return -1;
    }
    printf("Success to Controller!\n");

    int iaxis[2] = {0,1};

    float DposValue = 0;
    float MposValue = 0;
  
```

```

int AType = 1; //set axis type
int UnitValue = 100; //set UNITS

ret = ZAux_Direct_SetAtype(handle, iaxis[0], AType); //set axis 0 type
commandCheckHandler("ZAux_Direct_SetAtype", ret);
ZAux_Direct_SetUnits(handle, iaxis[0], UnitValue); //set axis UNITS
commandCheckHandler("ZAux_Direct_SetUnits", ret);
ZAux_Direct_SetFastDec(handle, iaxis[0], 10000); //set fast deceleration
commandCheckHandler("ZAux_Direct_SetFastDec", ret);

commandCheckHandler("ZAux_Direct_SetDecel", ret);
ret = ZAux_Direct_SetDpos( handle, iaxis[0],0); //set axis 0 DPOS to -100
commandCheckHandler("ZAux_Direct_SetDpos", ret);
ret = ZAux_Direct_SetMpos( handle, iaxis[0],0); // set axis 0 MPOS to -100
commandCheckHandler("ZAux_Direct_SetMpos12", ret);
ret = ZAux_Direct_SetAtype(handle, iaxis[1], AType); //set axis 1 type
commandCheckHandler("ZAux_Direct_SetAtype", ret);
ZAux_Direct_SetUnits(handle, iaxis[1], UnitValue); //set axis 1 UNITS
commandCheckHandler("ZAux_Direct_SetUnits", ret);
ZAux_Direct_SetFastDec(handle, iaxis[1], 10000); //set fast deceleration
commandCheckHandler("ZAux_Direct_SetFastDec", ret);

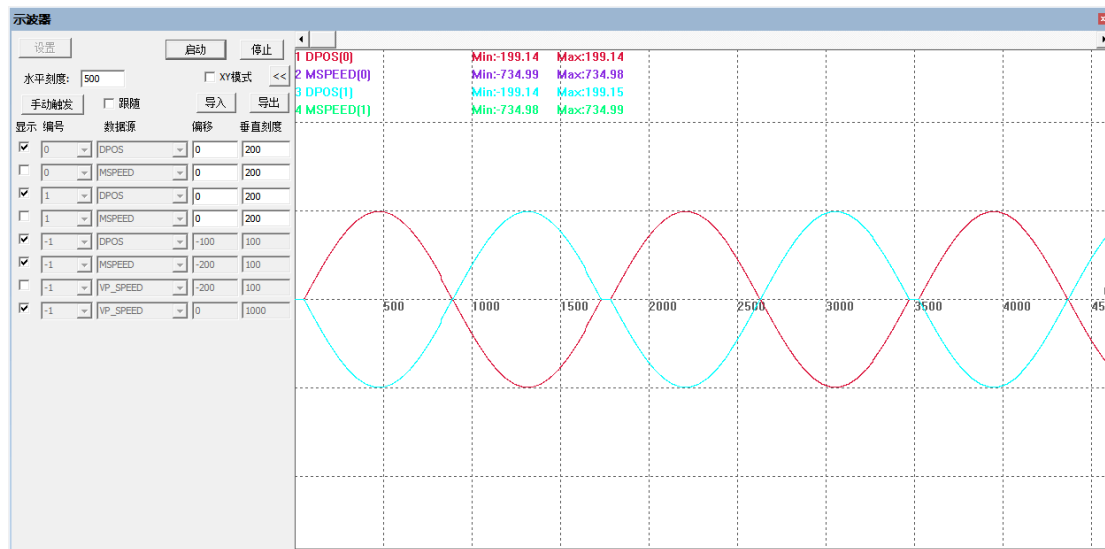
ret = ZAux_Direct_SetDpos( handle, iaxis[1], DposValue); //clear axis 1 dpos to 0
commandCheckHandler("ZAux_Direct_SetDpos", ret);
ret = ZAux_Direct_SetMpos( handle, iaxis[1], MposValue); //clear axis 1 mpos to 0
commandCheckHandler("ZAux_Direct_SetMpos", ret);

uint32 Tims[35]; //time planning of relative or absolute PT motion
float DposList[35]; //distance planning of relative PT motion
float DposList1[35]; //distance planning of relative PT motion
int i;
for ( i=0;i<35;i++)
{
    Tims[i]=50;
    DposList[i]=(sin(((2*PI)/34)*i))*200;
    DposList1[i]=(cos(((2*PI)/34)*i+0.5*PI))*200;
}
printf("DposList[i]=%f\n",DposList[34]);
ZAux_Trigger(handle);
for ( i=0;i<3;i++)
{
    ret = ZAux_Direct_MultiMovePtAbs(handle, 35,1, iaxis,Tims,DposList); //35*50ms, move
one sine curve with a period of y=sin(x)
    commandCheckHandler("ZAux_Direct_MultiMovePtAbs", ret);
    ret = ZAux_Direct_MultiMovePtAbs(handle, 35,1, &iaxis[1],Tims,DposList1); //35*50ms,
move one sine curve with a period of y=cos(x+0.5pi)
    commandCheckHandler("ZAux_Direct_SetMpos", ret);
    Sleep(34*50); //send once with a gap of 34*50ms
}
Sleep(5000);
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ; //if it executes successfully
printf("connection closed!\n");

handle = NULL;
return 0;
}

```

➤ Waveform:



PVT Motion

Axis 0 does forward and inverse motion between 0-200, the motion position curve will be $x = 100 * \cos(X + \pi) + 100$, and the speed curve will be $v = 100 * \sin(X + \pi)$.

🌈 Program Steps:

Connect to Controller (OpenEth) → set axis parameters (UNITS, ACCEL, SPEED, etc.) → set the path (plan position and speed) → execute the trajectory → end.

```
// test1.cpp: define the entry point of control panel application program
//
#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"
#include <math.h>
#define PI 3.141592654
void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail! return code is %d\n", command, ret);
        Sleep(2000);
        exit(0);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1"; // simulator IP address
    // char *ip_addr = (char *) "192.168.1.11"; // real controller IP address is 192.168.0.11 by default
```

```
ZMC_HANDLE handle = NULL;           //link handle
int ret = ZAux_OpenEth(ip_addr, &handle); //connect to controller
if (ERR_SUCCESS != ret)
{
    printf("%s fail!return code is %d\n", command, ret);
    Sleep(2000);
    exit(0);
}
printf("Success to Controller!\n");

int iaxis[2] = {0,1};

float DposValue = 0;
float MposValue = 0;
int AType = 1;           //set axis type
int UnitValue = 100;     //set UNITS

/*****set axis basic parameters*****/
ret = ZAux_Direct_SetAtype(handle, iaxis[0], AType); //set axis 0 type
commandCheckHandler("ZAux_Direct_SetAtype", ret);

ZAux_Direct_SetUnits(handle, iaxis[0], UnitValue); //set axis 0 UNITS
commandCheckHandler("ZAux_Direct_SetUnits", ret);

ZAux_Direct_SetFastDec(handle, iaxis[0], 1000000); //set fast deceleration
commandCheckHandler("ZAux_Direct_SetFastDec", ret);

ret = ZAux_Direct_SetDpos(handle, iaxis[0], 0); //set axis 0 DPOS as 0
commandCheckHandler("ZAux_Direct_SetDpos", ret);

ret = ZAux_Direct_SetMpos(handle, iaxis[0], 0); //set axis 0 MPOS as 0
commandCheckHandler("ZAux_Direct_SetMpos12", ret);

ZAux_Direct_SetMerge(handle, iaxis[0], 1);
commandCheckHandler("ZAux_Direct_SetMerge", ret);

/*****set motion trajectory*****/
// 'period = 2π/|ω|
// 'x=A*cos (ωx+ψ) +C
// 'obtain the slope, that is, the speed:
// 'v=-A*ω*SIN (ωx+ψ)

const int dlen=2*PI/0.01+1; //apart it as small segments

uint32 timelist[dlen];
float DposList[dlen]; //PVT motion distance planning
float SpeedList[dlen]; // PVT motion speed planning
int i;
int temp=0;
float x1=0.00;
x1=0.00;
//example: make one position curve "x=100*cos(X+π)+100", then the speed v=-100*SIN(X+π),
the period will be 2π
for (i=0; i<dlen; i++)
{
    DposList[i]=100*cos(x1+PI)+100;
    SpeedList[i]=-100*sin(x1+PI);
    timelist[i]=10;
    x1=x1+0.01;
}
```

```

i=0;

ZAux_Trigger(handle);
while(1)
{
    if ((i+1)*25>dlen)
    {
        ret=ZAux_Direct_MultiMovePvtAbs(handle,dlen-
i*25,1,iaxis,timelist,DposList+25*i,SpeedList+25*i);
        commandCheckHandler("ZAux_Direct_MultiMovePvtAbs", ret);
        i=0;
        temp=temp+1;
        if (temp>2)
        {
            printf("3 periods all executed!\n");
            break;
        }
    }else
    {
        ret=ZAux_Direct_MultiMovePvtAbs(handle,25,1,iaxis,timelist,DposList+25*i,SpeedList+25*i);
        commandCheckHandler("ZAux_Direct_MultiMovePvtAbs", ret);
        i=i+1;
    }
}

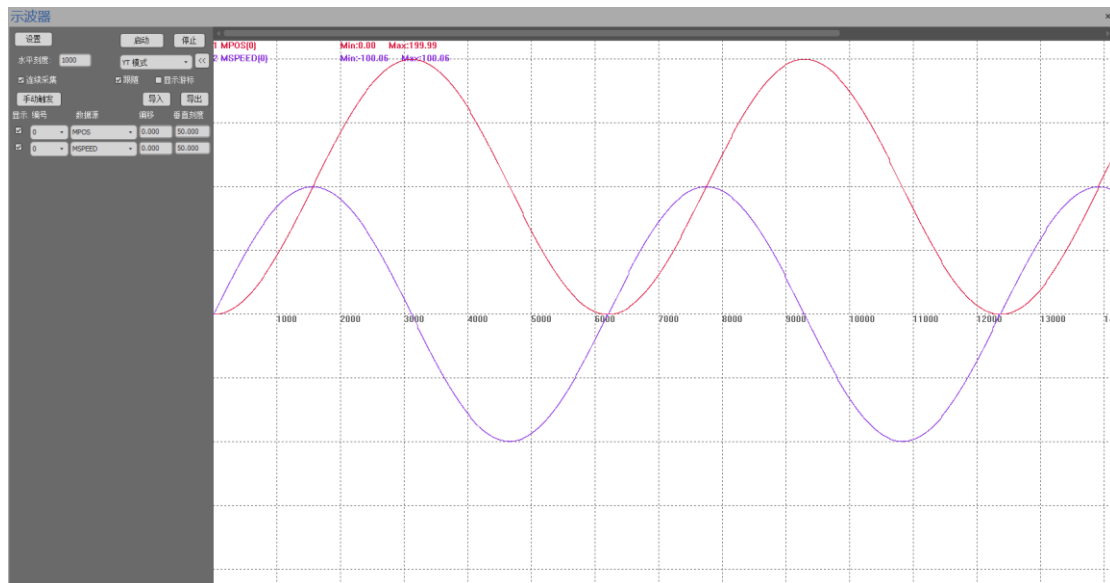
Sleep(5000);
ret = ZAux_Close(handle); //close connection
commandCheckHandler("ZAux_Close", ret) ;// if it executes successfully
printf("connection closed!\n");

handle = NULL;
return 0;
}

```

➤ Results:

Position Curve: $x=100 \cdot \cos(X+\pi)+100$; Speed Curve: $v=-100 \cdot \sin(X+\pi)$



Chapter VI Robot Motion Control

6.1. Robot Model

6.1.1. Instruction List

Commands	Descriptions
ZAux_Direct_Connframe	Build robot inverse kinematics
ZAux_Direct_Connreframe	Build robot forward kinematics
ZAux_Direct_FrameRotate	Rotate robot coordinate system
ZAux_Frame_Rotate	Transform workpiece coordinate system
ZAux_Frame_Rotate2	Calculate coordinate system
ZAux_World_DPos	World coordinate system
ZAux_Direct_FrameStatus	Robot axis state
ZAux_Direct_FrameTrans2	Transform forward mode and inverse mode
ZAux_Direct_FrameCal	Correct parameters
ZAux_Direct_Mover_C	Make joint-axis move plane circular interpolation, relative
ZAux_Direct_Mover_CABS	Make joint-axis move plane circular interpolation, absolute
ZAux_Direct_Mover_C3	Make joint-axis move space circular interpolation, relative
ZAux_Direct_Mover_C3ABS	Make joint-axis move space circular interpolation, absolute
ZAux_Direct_Mover_L	Make joint-axis move linear interpolation, relative
ZAux_Direct_Mover_LABS	Make joint-axis move linear interpolation, absolute

6.1.2. Principle & Description

In Zmotion, the number of robotic arm's motors define the name of robot. For example, if there are 6 motors, it is called 6-joint robot. And the axis that controls the motor motion is called the **joint axis** of robot. And the axis that controls robot terminal to do linear motion under space coordinate system is called **virtual axis**.

They are indicated by **ZAux_Direct_Connreframe (inverse mode)** or **ZAux_Direct_Connframe (forward mode)**. And both motions can be controlled by motion

commands (all are valid), but two different types axes can't move at the same time.

Use **ZAux_Direct_Single_Cancel** to cancel robot mode.

➤ **Inverse Kinematic:**

Use **ZAux_Direct_Connframe** to build inverse kinematics. It is used for joint axis. At this time, virtual axis can do linear, circular, spherical motions, and the joint axis will automatically move to corresponding joint axis position.

➤ **Forward Kinematic:**

Use **ZAux_Direct_Connreframe** to build forward kinematics. It is used for virtual axis. Now, joint axis can do all kinds of motions, but actual trajectory is not linear or circular.

This mode is usually used to manually adjust joint position or upper point homing.

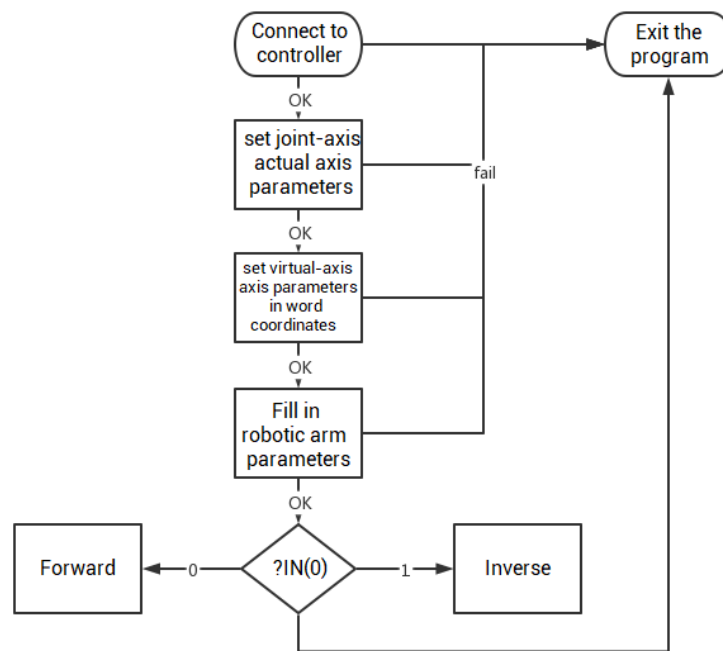
➤ **3 Corner Modes of Kinematics:**

Corner Mode of Terminal Axis under Inverse Kinematics ZAux_Direct_SetCornerMod (ZMC_HANDLE handle, int iaxis, int iValue)	Connection Speed of Joint-Axis ZAux_Direct_SetC lutchRate	Motion Mode Descriptions
BIT7 = 1 of iValue value	Invalid	Hybrid mode: speeds and accelerations of virtual axis and joint axis take effect at the same time. In this mode, all virtual axes automatically join in motion all the time, but it is invalid in motion superposition of virtual axis. For version above 150718, it supports linear motion, but doesn't support continuous interpolation. And it can't change the acceleration and deceleration while moving.
BIT 7 = 0 of iValue value	0	Smooth mode: joint axis uses itself speed and acceleration, it will do speed planning, and the trajectory will deform in high-speed.
	Non - 0	Force mode: joint axis plans speed and acceleration fully according to virtual axis'.

6.1.3. Routine

Build Robotic Arm Model

This routine mainly demonstrates the routine of the standard scara manipulator. Modify the robot type through the value corresponding to the robot type. For the robot type, please refer to the manual. Then manually move the corresponding virtual axis in RTSys (ZRobotView software can be download from website).



```

// test1.cpp: define the entry point of control panel application program
//

#include "stdafx.h"
#include <windows.h>
#include "zmotion.h"
#include "zauxdll2.h"
#include <math.h>
void commandCheckHandler(const char *command, int ret)
{
    if (ret) // it is not 0, fail
    {
        printf("%s fail! return code is %d\n", command, ret);
        Sleep(2000);
        exit(0);
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    char *ip_addr = (char *) "127.0.0.1"; // simulator IP address
    //char *ip_addr = (char *) "192.168.1.11"; // real controller address is 192.168.0.11 by default
    ZMC_HANDLE handle = NULL; // link handle
    int ret = ZAux_OpenEth(ip_addr, &handle); // connect to controller
    if (ERR_SUCCESS != ret)

```



```
{
    printf("Fail to Controller!\n");
    handle = NULL;
    Sleep(2000);
    return -1;
}
printf("Success to Controller!\n");

int JointAxisList[4] = { 0, 1, 2, 3 };// //define the type of joint-axis

//set axis type
ret = ZAux_Direct_SetAtype(handle, JointAxisList[0], 1);
commandCheckHandler("ZAux_Direct_SetAtype", ret);

ret = ZAux_Direct_SetAtype(handle, JointAxisList[1], 1);
commandCheckHandler("ZAux_Direct_SetAtype", ret);

ret = ZAux_Direct_SetAtype(handle, JointAxisList[2], 1);
commandCheckHandler("ZAux_Direct_SetAtype", ret);

ret = ZAux_Direct_SetAtype(handle, JointAxisList[3], 1);
commandCheckHandler("ZAux_Direct_SetAtype", ret);

//set UNITS
ret = ZAux_Direct_SetUnits(handle, JointAxisList[0], 3600 * 2 / 360);
commandCheckHandler("ZAux_Direct_SetUnits", ret);

ret = ZAux_Direct_SetUnits(handle, JointAxisList[1], 3600 * 2 / 360);
commandCheckHandler("ZAux_Direct_SetUnits", ret);

ret = ZAux_Direct_SetUnits(handle, JointAxisList[2], 3600 * 2 / 360);
commandCheckHandler("ZAux_Direct_SetUnits", ret);

ret = ZAux_Direct_SetUnits(handle, JointAxisList[3], 3600 * 2 / 1.5);
commandCheckHandler("ZAux_Direct_SetUnits", ret);

//set DPOS
ret = ZAux_Direct_SetDpos(handle, JointAxisList[0], 0);
commandCheckHandler("ZAux_Direct_SetUnits", ret);

ret = ZAux_Direct_SetDpos(handle, JointAxisList[1], 0);
commandCheckHandler("ZAux_Direct_SetUnits", ret);

ret = ZAux_Direct_SetDpos(handle, JointAxisList[2], 0);
commandCheckHandler("ZAux_Direct_SetUnits", ret);

ret = ZAux_Direct_SetDpos(handle, JointAxisList[3], 0);
commandCheckHandler("ZAux_Direct_SetUnits", ret);

//set acceleration
ret = ZAux_Direct_SetAccel(handle, JointAxisList[0], 1000);
commandCheckHandler("ZAux_Direct_SetAccel", ret);

ret = ZAux_Direct_SetAccel(handle, JointAxisList[1], 1000);
commandCheckHandler("ZAux_Direct_SetAccel", ret);

ret = ZAux_Direct_SetAccel(handle, JointAxisList[2], 1000);
commandCheckHandler("ZAux_Direct_SetAccel", ret);

ret = ZAux_Direct_SetAccel(handle, JointAxisList[3], 1000);
commandCheckHandler("ZAux_Direct_SetAccel", ret);

//set deceleration
ret = ZAux_Direct_SetDecel(handle, JointAxisList[0], 1000);
```

```
commandCheckHandler("ZAux_Direct_SetDecel", ret);

ret = ZAux_Direct_SetDecel(handle, JointAxisList[1], 1000);
commandCheckHandler("ZAux_Direct_SetDecel", ret);

ret = ZAux_Direct_SetDecel(handle, JointAxisList[2], 1000);
commandCheckHandler("ZAux_Direct_SetDecel", ret);

ret = ZAux_Direct_SetDecel(handle, JointAxisList[3], 1000);
commandCheckHandler("ZAux_Direct_SetDecel", ret);

//set connection ratio
ret = ZAux_Direct_SetClutchRate(handle, JointAxisList[0], 0);
commandCheckHandler("ZAux_Direct_SetClutchRate", ret);

ret = ZAux_Direct_SetClutchRate(handle, JointAxisList[1], 0);
commandCheckHandler("ZAux_Direct_SetClutchRate", ret);

ret = ZAux_Direct_SetClutchRate(handle, JointAxisList[2], 0);
commandCheckHandler("ZAux_Direct_SetClutchRate", ret);

ret = ZAux_Direct_SetClutchRate(handle, JointAxisList[3], 0);
commandCheckHandler("ZAux_Direct_SetClutchRate", ret);

int VirtualAxisList[4] = { 6, 7, 8, 9 };// //define the type of virtual-axis
float TableList[7] = { 10, 10, 3600 * 2, 3600 * 2, 3600 * 2, 0, 0 };
float GetTableList[8];
ret = ZAux_Direct_GetTable(handle, 0, 8, GetTableList);

//set axis type
ret = ZAux_Direct_SetAtype(handle, VirtualAxisList[0], 0);
commandCheckHandler("ZAux_Direct_SetAtype", ret);

ret = ZAux_Direct_SetAtype(handle, VirtualAxisList[1], 0);
commandCheckHandler("ZAux_Direct_SetAtype", ret);

ret = ZAux_Direct_SetAtype(handle, VirtualAxisList[2], 0);
commandCheckHandler("ZAux_Direct_SetAtype", ret);

ret = ZAux_Direct_SetAtype(handle, VirtualAxisList[3], 0);
commandCheckHandler("ZAux_Direct_SetAtype", ret);

//set system table value
ret = ZAux_Direct_SetTable(handle, 0, 8, TableList);
commandCheckHandler("ZAux_Direct_SetAtype", ret);

//set UNITS
ret = ZAux_Direct_SetUnits(handle, VirtualAxisList[0], 1000);
commandCheckHandler("ZAux_Direct_SetUnits", ret);

ret = ZAux_Direct_SetUnits(handle, VirtualAxisList[1], 1000);
commandCheckHandler("ZAux_Direct_SetUnits", ret);

ret = ZAux_Direct_SetUnits(handle, VirtualAxisList[2], 3600 * 2 / 360);
commandCheckHandler("ZAux_Direct_SetUnits", ret);

ret = ZAux_Direct_SetUnits(handle, VirtualAxisList[3], 1000);
commandCheckHandler("ZAux_Direct_SetUnits", ret);

//set speed
ret = ZAux_Direct_SetSpeed(handle, VirtualAxisList[0], 200);
commandCheckHandler("ZAux_Direct_SetSpeed", ret);

ret = ZAux_Direct_SetSpeed(handle, VirtualAxisList[1], 200);
```

```

commandCheckHandler("ZAux_Direct_SetSpeed", ret);

ret = ZAux_Direct_SetSpeed(handle, VirtualAxisList[2], 200);
commandCheckHandler("ZAux_Direct_SetSpeed", ret);

ret = ZAux_Direct_SetSpeed(handle, VirtualAxisList[3], 200);
commandCheckHandler("ZAux_Direct_SetSpeed", ret);

//set acceleration
ret = ZAux_Direct_SetAccel(handle, VirtualAxisList[0], 1000);
commandCheckHandler("ZAux_Direct_SetAccel", ret);

ret = ZAux_Direct_SetAccel(handle, VirtualAxisList[1], 1000);
commandCheckHandler("ZAux_Direct_SetAccel", ret);

ret = ZAux_Direct_SetAccel(handle, VirtualAxisList[2], 1000);
commandCheckHandler("ZAux_Direct_SetAccel", ret);

ret = ZAux_Direct_SetAccel(handle, VirtualAxisList[3], 1000);
commandCheckHandler("ZAux_Direct_SetAccel", ret);

//set deceleration
ret = ZAux_Direct_SetDecel(handle, VirtualAxisList[0], 1000);
commandCheckHandler("ZAux_Direct_SetDecel", ret);

ret = ZAux_Direct_SetDecel(handle, VirtualAxisList[1], 1000);
commandCheckHandler("ZAux_Direct_SetDecel", ret);

ret = ZAux_Direct_SetDecel(handle, VirtualAxisList[2], 1000);
commandCheckHandler("ZAux_Direct_SetDecel", ret);

ret = ZAux_Direct_SetDecel(handle, VirtualAxisList[3], 1000);
commandCheckHandler("ZAux_Direct_SetDecel", ret);

uint32 GetValue = 0; //1 means IN is ON, 0 means OFF
int Jogmaxaxes = 4; //the number of joint-axis
int frame = 1; //robot type, 1-standard SCARA
int Virmaxaxes = 4; //the number of virtual-axis
int piValue; //define the variable to get the motion buffer, -1 means that there is no remaining
motion buffer at present, 0 means that there are remaining motion buffers

//uint32 GetValue = 0; //1 means IN is ON, 0 means OFF
while (1)
{
    //ZAux_Direct_GetIn(ZMC_HANDLE handle, int ionum, uint32 *piValue); get the status
    value of input
    ret = ZAux_Direct_GetIn(handle, 0, &GetValue); //1 means IN is ON, 0 means OFF
    commandCheckHandler("ZAux_Direct_GetIn", ret);
    //printf("GetValue 1 means IN is ON, 0 means OFF GetValue = %d \n", GetValue);

    if (GetValue == 1)
    {
        ret = ZAux_Direct_Connframe(handle, Jogmaxaxes, JointAxisList, frame, 0,
        Virmaxaxes, VirtualAxisList);
        commandCheckHandler("ZAux_Direct_Connreframe", ret);
        printf("open inverse\n");
        Sleep(1000);

        ret = ZAux_Direct_GetLoaded(handle, 0, &piValue);
        commandCheckHandler("ZAux_Direct_GetLoaded", ret);
        printf("piValue = %d\n", piValue);
        //printf("piValue means current motion buffers, -1 means no motion buffer, 0 means
        remaining buffers piValue = %d\n", piValue);
        //0 means no motion, -1 means with motion
    }
}

```

```

//wait for the number of tasks in the motion buffer to complete
while (piValue == -1)
{
    ret = ZAux_Direct_GetLoaded(handle, 0, &piValue);
    commandCheckHandler("ZAux_Direct_GetLoaded", ret);
    printf("piValue = %d\n", piValue);
    Sleep(1000);
    if (piValue == 0)
    {
        printf("inverse---while loop piValue = %d\n ", piValue);
        break;
    }
}

}
else if (GetValue == 0)
{
    ret = ZAux_Direct_Connreframe(handle, Virmaxaxes, VirtualAxisList, frame, 0,
Jogmaxaxes, JointAxisList);
    commandCheckHandler("ZAux_Direct_Connreframe", ret);
    printf("open forward\n");
    Sleep(1000);

    ret = ZAux_Direct_GetLoaded(handle, 0, &piValue);
    commandCheckHandler("ZAux_Direct_GetLoaded", ret);
    printf("piValue = %d\n", piValue);

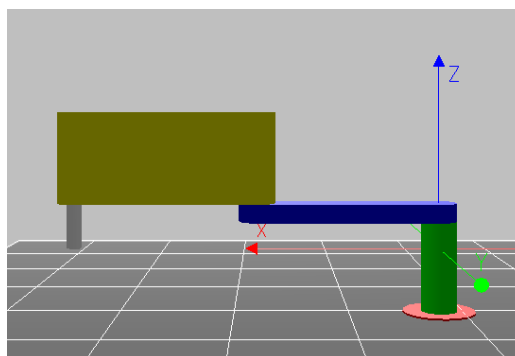
    //wait for the number of tasks in the motion buffer to complete
    while (piValue == -1)
    {
        if (piValue == 0)
        {
            printf("forward---while loop piValue = %d\n ", piValue);
            break;
        }
    }
}

}
ret = ZAux_Close(handle); //close the connection
commandCheckHandler("ZAux_Close", ret) ;//if it executes successfully
printf("connection closed!\n");

handle = NULL;
return 0;
}

```

➤ Simulator Model Building:



Chapter VII Instruction Details

Instruction 1	ZAux_Direct_GetAddax	
Original Format	int32 __stdcall ZAux_Direct_GetAddax (ZMC_HANDLE handle , int iaxis, int * piValue);	
Description	Read superposition axis.	
Input parameters	Parameter name	Description
	handle	Link handle
	iaxis	Axis No.
Output parameters	Parameter name	Description
	piValue	Read axis' superposition axis No., when there is no superposition axis, it is -1.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	<pre>Int iValue; ZAux_Direct_GetAddax(handle, 0, &iValue); //read axis 0's superposition axis No.</pre>	
Details	➤ <u>Usage:</u> motion distance of superimposing axis * pulse amount of superimposing axis / pulse amount of superimposed axis = motion distance of superimposed axis. ➤ <u>Example:</u> suppose the pulse amount of axis A is 100, the pulse amount of axis B is 50, and the superimposing axis movement is 100 ✧ Superimpose the movement of axis A to axis B. At this time, axis A moves 100, then axis B moves $100 \times 100 / 50 = 200$. ✧ The movement of axis B is superimposed on axis A. At this time, axis B moves 100, then axis A moves $100 \times 50 / 100 = 50$. ➤ <u>Notes:</u> ● The axes cannot be superimposed on each other at the same time. That is, after A is superimposed on B, B can no longer be superimposed on A. ● Serial superposition is valid -- A is superimposed to B, B is superimposed to C. ● Parallel superposition is valid, -- A superimposes to B and C at the same time. ● When superimposing, the speed changes from the superimposed axis, and the acceleration & deceleration are determined according to the accel & decel of the	

	superimposing axis and the ratio of the units of the two axes. ● ZAux_Direct_Single_Addax function does not work when the current state of the axis is under forward solution (ZAux_Direct_Connreframe) or inverse solution (ZAux_Direct_Connframe).

Instruction 2	ZAux_Direct_Single_Addax	
Original Format	int32 __stdcall ZAux_Direct_Single_Addax (ZMC_HANDLE handle , int iaxis, int iaddaxis);	
Description	Axis superposition motion, iaddaxis motion is superposed into iaxis, and the number of pulses is superposed by ADDAX instruction.	
Input parameters	Parameter name	Description
	handle	Link handle
	iaxis	Superposed axis No.
	iaddaxis	Superposition axis No.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Motion Superposition	
Details	➤ <u>Usage:</u> motion distance of superimposing axis * pulse amount of superimposing axis / pulse amount of superimposed axis = motion distance of superimposed axis.	
	➤ <u>Example:</u> suppose the pulse amount of axis A is 100, the pulse amount of axis B is 50, and the superimposing axis movement is 100	
	✧ Superimpose the movement of axis A to axis B. At this time, axis A moves 100, then axis B moves 100*100/50=200.	
	✧ The movement of axis B is superimposed on axis A. At this time, axis B moves 100, then axis A moves 100*50/100=50.	
	➤ <u>Notes:</u> ● The axes cannot be superimposed on each other at the same time. That is, after A is superimposed on B, B can no longer be superimposed on A. ● Serial superposition is valid -- A is superimposed to B, B is superimposed to C. ● Parallel superposition is valid, -- A superimposes to B and C at the same time. ● When superimposing, the speed changes from the	

	superimposed axis, and the acceleration & deceleration are determined according to the accel & decel of the superimposing axis and the ratio of the units of the two axes. ● ZAux_Direct_Single_Addax function does not work when the current state of the axis is under forward solution (ZAux_Direct_Connreframe) or inverse solution (ZAux_Direct_Connframe).

Instruction 3	ZAux_Direct_Cam														
Original Format	int32 __stdcall ZAux_Direct_Cam(ZMC_HANDLE handle,int iaxis, int istartpoint, int iendpoint, float ftablemulti, float fDistance)														
Description	Electronic cam, it determines axis motion according to TABLE data.														
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link handle</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> <tr> <td>istartpoint</td><td>TABLE No. where saves starting point.</td></tr> <tr> <td>iendpoint</td><td>TABLE No. where saves end point.</td></tr> <tr> <td>ftablemulti</td><td>The position multiplies this ratio, usually it is the pulse amount.</td></tr> <tr> <td>fDistance</td><td>The distance of reference motion, which is used to calculate total motion time.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link handle	iaxis	Axis No.	istartpoint	TABLE No. where saves starting point.	iendpoint	TABLE No. where saves end point.	ftablemulti	The position multiplies this ratio, usually it is the pulse amount.	fDistance	The distance of reference motion, which is used to calculate total motion time.
Parameter name	Description														
handle	Link handle														
iaxis	Axis No.														
istartpoint	TABLE No. where saves starting point.														
iendpoint	TABLE No. where saves end point.														
ftablemulti	The position multiplies this ratio, usually it is the pulse amount.														
fDistance	The distance of reference motion, which is used to calculate total motion time.														
OP parameters	/														
Return value	0 = success, not-0 = failure (please check error codes for details).														
Example	Electronic Cam														
Details	● 2 or more CAM commands can use the same TABLE data area. ● The total motion time is determined by the speed and the 4th parameter, and the actual motion speed is matched automatically according to TABLE trajectory and time. ● Please set TABLE data manually, <u>the first data is the index, which is recommended to be 0.</u> ● ATBLE data * table multiplier this ratio = sent pulse amounts. ● Please ensure the distance parameter transferred by the instruction * units should be the integer type pulse, otherwise, the floating type will cause small error in motion.														

Instruction 4	ZAux_Direct_Cambox
Original Format	int32 __stdcall ZAux_Direct_Cambox(ZMC_HANDLE handle,int iaxis,int istartpoint, int iendpoint, float ftablemulti, float fDistance, int ilinkaxis, int iooption, float flinkstartpos)
Description	Electronic cam, it determines axis motion according to TABLE data. And the TABLE data corresponds to motion trajectory position, which is relative to motion starting point. Then, the following axis

	moves according to reference axis.	
Input parameters	Parameter name	Description
	handle	Link handle
	iaxis	Axis No.
	istartpoint	TABLE No. where saves starting point.
	iendpoint	TABLE No. where saves end point.
	ftablemulti	The position multiples this ratio, usually it is the pulse amount.
	fDistance	The distance of reference motion, which is used to calculate total motion time.
	llinkaxis	Reference axis.
	ioption	The connection method of reference axis. (different binary values represent different meanings) ➤ Bit 0: the connection (between current axis & reference axis) starts when the reference axis "regist" event is triggered. ➤ Bit 1: the connection (between current axis & reference axis) starts when the reference axis arrived set absolute position. ➤ Bit 2: automatically repeat continuous bi-directional motion (REP_OPTION=1 to cancel repeat). ➤ Bit 5: it is connected only when the reference axis is in forward motion. ➤ Bit 8: the connection (between current axis & reference axis) starts when the reference axis "MARKB" event is triggered. Please use the latest firmware.
	Flinkstartpos	When bit 1 of the ioption is set to 1, which means the absolute position at which the connection starts.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Electronic Cam Synchronous Motion	
Details	● 2 or more CAMBOX can use the same TABLE data area. ● The total motion time is determined by the speed and the 4th parameter, and the actual motion speed is matched automatically according to TABLE trajectory and time. ● Please set TABLE data manually, the first data is the index.	

	which is recommended to be 0. ● ATBLE data * table multiplier this ratio = sent pulse amounts. ● Please ensure the distance parameter transferred by the instruction * units should be the integer type pulse, otherwise, the floating type will cause small error in motion.

Instruction 5	ZAux_Direct_Movelink	
Original Format	int32 __stdcall ZAux_Direct_Movelink(ZMC_HANDLE handle, int iaxis, float fDistance, float fLinkDis, float fLinkAcc, float fLinkDec, int iLinkaxis, int ioption, float flinkstartpos)	
Description	Automatic cam. It is the customized cam motion, you could set needed acceleration and deceleration.	
Input parameters	Parameter name	Description
	handle	Link handle
	iaxis	Axis No.
	fDistance	It is the current axis' moved distance during the motion. Unit: user unit.
	fLinkDis	It is the reference axis' moved forward distance during connection. Unit: units.
	fLinkAcc	It is the reference axis' moved forward distance when the current axis is in acceleration. Unit: units.
	fLinkDec	It is the reference axis' moved forward distance when the current axis is in deceleration. Unit: units.
	iLinkaxis	Reference axis No.
	ioption	The connection method of reference axis. (different binary values represent different meanings) ➤ Bit 0: the connection precisely starts when the reference axis "regist" event is triggered. ➤ Bit 1: the connection starts when the reference axis arrived one absolute position. ➤ Bit 2: only when BIT2 is set, then it will automatically execute in cycle by MOVELINK, and it can be inverse. ➤ Bit 3: when BIT3 is set, it will use S curve acceleration and deceleration. Valid in firmware after 20170502. ➤ Bit 5: it is connected only when the reference axis is in forward motion. ➤ Bit 8: the connection precisely starts

		when the reference axis "MARKB" event is triggered. Please use the latest firmware.
	Flinkstartpos	When bit 1 of the ioption is set to 1, which means the absolute position at which the connection starts.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Fly-shearing	
Details	- The connected axis is the reference axis, then the connecting axis is the following axis. - There are 3 phases of following axis distance for reference axis motion: acceleration, constant speed and deceleration. - In acceleration and deceleration stages, the basic axis distance must be two times of following axis motion distance for matching speed. - Please ensure the distance parameter transferred by the instruction * units should be the integer type pulse, otherwise, the floating type will cause small error in motion.	

Instruction 6	ZAux_Direct_Moveslink	
Original Format	int32 __stdcall ZAux_Direct_Moveslink(ZMC_HANDLE handle, int iaxis, float fDistance, float fLinkDis, float startsp, float endsp, int iLinkaxis, int ioption, float flinkstartpos)	
Description	Automatic cam 2. It is used for customized cam motion, which automatically plans the middle curve, no need to calculate cam table.	
Input parameters	Parameter name	Description
	handle	Link handle
	iaxis	Axis No.
	fDistance	It is the current axis' moved distance during the motion. Unit: user unit.
	fLinkDis	It is the reference axis' moved forward distance during connection. Unit: units.
	startsp	It is the speed velocity between following axis and reference axis when starts. Unit: units / units. Negative value: following axis does inverse motion.
	endsp	It is the speed velocity between following axis and reference axis when ends. Unit: units / units. Negative value: following axis does inverse motion.

		When startsp = endsp = fdistance / flink dist, it is in constant speed.
	iLinkaxis	Reference axis No.
	ioption	The connection method of reference axis. (different binary values represent different meanings) ➤ Bit 0: the connection precisely starts when the reference axis "regist" event is triggered. ➤ Bit 1: the connection starts when the reference axis arrived one absolute position. ➤ Bit 2: only when BIT2 is set, then it will automatically execute in cycle by MOVELINK, and it can be inverse. ➤ Bit 3: when BIT3 is set, it will use S curve acceleration and deceleration. Valid in firmware after 20170502. ➤ Bit 5: it is connected only when the reference axis is in forward motion. Bit 8: the connection precisely starts when the reference axis "MARKB" event is triggered. Please use the latest firmware.
	Flinkstartpos	When bit 1 of the ioption is set to 1, which means the absolute position at which the connection starts.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Electronic Cam Synchronous Motion	
Details	● The connected axis is the reference axis, then the connecting axis is the following axis. ● There are 3 phases of following axis distance for reference axis motion: acceleration, constant speed and deceleration. ● In acceleration and deceleration stages, the basic axis distance must be two times of following axis motion distance for matching speed. ● Please ensure the distance parameter transferred by the instruction * units should be the integer type pulse, otherwise, the floating type will cause small error in motion.	
Instruction 7	ZAux_Direct_Regist	
Original Format	int32 __stdcall ZAux_Direct_Regist(ZMC_HANDLE handle,int iaxis, int imode)	

Description	Do position latch at one time (single)		
Input parameters	Parameter name	Description	
	handle	Link mark.	
	iaxis	Axis No.	
	imode	Value	Descriptions
		1	Save absolute position in REG_POS when meet Z pulse rising edge.
		2	Save absolute position in REG_POS when meet Z pulse falling edge.
		3	Save absolute position in REG_POS when meet IN signal R0 rising edge.
		4	Save absolute position in REG_POS when meet IN signal R0 falling edge.
		6	Save absolute position in REG_POS when meet IN signal R0 rising edge. Save absolute position in REG_POSB when meet Z pulse rising edge.
		7	Save absolute position in REG_POS when meet IN signal R0 rising edge. Save absolute position in REG_POSB when meet Z pulse falling edge.
		8	Save absolute position in REG_POS when meet IN signal R0 falling edge. Save absolute position in REG_POSB when meet Z pulse rising edge.
		9	Save absolute position in REG_POS when meet IN signal R0 falling edge. Save absolute position in REG_POSB when meet Z pulse falling edge.
		10	Save absolute position in REG_POS when meet IN signal R0 rising edge. Save absolute position in REG_POSB when meet IN signal R1 rising edge.
		11	Save absolute position in REG_POS when meet IN signal R0 rising edge. Save absolute position in REG_POSB when meet IN signal R1 falling edge.
		12	Save absolute position in REG_POS when meet IN signal R0 falling edge. Save absolute position in REG_POSB when meet IN signal R1 rising edge.
		13	Save absolute position in REG_POS when meet IN signal R0 falling edge. Save absolute position in REG_POSB when meet IN signal R1 falling edge.
		14	Save absolute position in REG_POSB when meet IN signal R1 rising edge
		15	Save absolute position in REG_POSB when meet IN signal R1 falling edge
		16	Save absolute position in REG_POSB

			when meet Z pulse rising edge.
		17	Save absolute position in REG_POSB when meet Z pulse falling edge.
		18	Save absolute position in REG_POSC when meet IN signal R2 rising edge
		19	Save absolute position in REG_POSC when meet IN signal R2 falling edge
		20	Save absolute position in REG_POSD when meet IN signal R3 rising edge
		21	Save absolute position in REG_POSD when meet IN signal R3 falling edge
OP Parameters	/		
Return value	0 = success, not-0 = failure (please check error codes for details).		
Example	Position Latch		
Details	- Rising & falling edges refer to controller inner state. Please set correct method according to external input signal needs. ✓ rising edge trigger: the latch is triggered when the external input changes to OFF from ON. ✓ falling edge trigger: the latch is triggered when the external input changes to ON from OFF. ✓ For pulse axis type, R0, R1, and Z pulse are used to latch. ✓ For bus axis type, R2 and R3 are used to latch. - Valid latch types: ✓ Support encoder latch, and it can achieve virtual axis latch and pulse axis latch for ZMC4XX series or above (latest firmware). ✓ Support EtherCAT latch: it only can use drive latch – using drive IO to latch. ✓ Support RTECH latch: using controller latch - Some ZMC4XX controllers & above can do 4-channel latching. ✓ ZMC432-V2: 2 latch channels. ✓ ZMC432N: 4 latch channels. ✓ ZMC412: 8 latch channels. ✓ EtherCAT drive latch and controller latch can be used at the same time, but it needs 4-channel. ✓ 4-channel: use REG_INPUTS to latch to corresponding channels (MARK, MARKB, MARKC, and MARKD). - When latching is made, axis status MARK will be set as ON, and latched position will be saved into REG_POS. - Input signals R0, R1, EZ of each axis can be used to “latch”. And when two signals are used, the second one uses MARKB and REG_POSB. - Generally, R0 and R0 correspond to IN 0 and IN 1. Please check user manual – general IN.		

Instruction 8	ZAux_Direct_CycleRegist		
Original Format	int32 __stdcall ZAux_Direct_CycleRegist(ZMC_HANDLE handle,int iaxis, int imode,int iTabStart,int iTabNum)		
Description	Position continuous latch.		
Input parameters	Parameter name	Description	
	handle	Link mark.	
	iaxis	Axis No.	
	imode	Value	Descriptions
		101	Save absolute position in REG_POS when meet Z pulse rising edge.
		102	Save absolute position in REG_POS when meet Z pulse falling edge.
		103	Save absolute position in REG_POS when meet IN signal R0 rising edge.
		104	Save absolute position in REG_POS when meet IN signal R0 falling edge.
		114	Save absolute position in REG_POSB when meet IN signal R1 rising edge
		115	Save absolute position in REG_POSB when meet IN signal R1 falling edge
		116	Save absolute position in REG_POSB when meet Z pulse rising edge.
		117	Save absolute position in REG_POSB when meet Z pulse falling edge.
		123	Save absolute position in REG_POSB when meet IN signal R0 rising edge.
		124	Save absolute position in REG_POSB when meet IN signal R0 falling edge.
		133	Save absolute position in REG_POS when meet IN signal R0 rising edge. Next time, save when meeting falling edge. Switch between rising edge and falling edge in cycle.
		134	Save absolute position in REG_POS when meet IN signal R0 falling edge. Next time, save when meeting rising edge. Switch between rising edge and falling edge in cycle.
		135	Save absolute position in REG_POSB when meet IN signal R1 rising edge. Next time, save when meeting falling edge. Switch between rising edge and falling edge in cycle.
		136	Save absolute position in REG_POSB when meet IN signal R1 falling edge. Next time, save when meeting rising edge.

			Switch between rising edge and falling edge in cycle.
OP Parameters	/		
Return value	0 = success, not-0 = failure (please check error codes for details).		
Example	Continuous position latch		
Details	- Latch results are stored in TABLE. - Latch two-channel continuously & respectively, which means it can achieve continuous latch for upper and bottom edge. ECL: valid in 20150829 firmware and above. ZMC4XX controllers: valid in 20170523 firmware and above.		

Instruction 9	ZAux_Direct_GetCloseWin	
Original Format	int32 __stdcall ZAux_Direct_GetCloseWin(ZMC_HANDLE handle,int iaxis, float * pfValue);	
Description	Read end coordinates range of latching. Reserved	
Input parameters	Parameter name	Description
	handle	Link mark.
	iaxis	Axis No.
Output parameters	Parameter name	Description
	pfValue	Returned range value
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	Reversed function	

Instruction 10	ZAux_Direct_SetCloseWin	
Original Format	int32 __stdcall ZAux_Direct_SetCloseWin(ZMC_HANDLE handle,int iaxis, float fValue);	
Description	Set end coordinates range of latching. Reserved	
Input parameters	Parameter name	Description
	handle	Link mark.
	iaxis	Axis No.
	iValue	Range value that is set.
Output parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	Reversed function	

Instruction 11	ZAux_Direct_GetRegPosB	
Original Format	int32 __stdcall ZAux_Direct_GetRegPosB (ZMC_HANDLE handle, int iaxis, float *pfValue);	
Description	Return to measurement position (ZAux_Direct_GetMpos) of latch register 2 (ZAux_Direct_GetMarkB). Unit: pulse amount	
Input parameters	Parameter name	Description
	handle	Link mark.
	iaxis	Axis No.
Output parameters	Parameter name	Description
	pfValue	Returned coordinate position
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	● Default: encoder latch only. ● For ZMC4XX controllers, pulse and virtual latching can be used.	

Instruction 12	ZAux_Direct_GetRegPos	
Original Format	int32 __stdcall ZAux_Direct_GetRegPosB (ZMC_HANDLE handle, int iaxis, float *pfValue);	
Description	Return to measurement position (ZAux_Direct_GetMpos) of latch register 1 (ZAux_Direct_GetMark).	
Input parameters	Parameter name	Description
	handle	Link mark.
	iaxis	Axis No.
Output parameters	Parameter name	Description
	pfValue	Latched coordinate position
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Position Latch	
Details	● Default: encoder latch only. ● For ZMC4XX controllers, pulse and virtual latching can be used.	

Instruction 13	ZAux_Direct_GetOpenWin	
Original Format	int32 __stdcall ZAux_Direct_GetOpenWin (ZMC_HANDLE handle, int iaxis, float *pfValue);	
Description	Read starting coordinates range of latching. Reserved	
Input parameters	Parameter name	Description
	handle	Link mark.
	iaxis	Axis No.

Output parameters	Parameter name	Description
	pfValue	Returned range value
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	Reversed function	

Instruction 14	ZAux_Direct_SetOpenWin	
Original Format	int32 __stdcall ZAux_Direct_SetOpenWin (ZMC_HANDLE handle, int iaxis, float *pfValue);	
Description	Set starting coordinates range of latching. Reserved	
Input parameters	Parameter name	Description
	handle	Link mark.
	iaxis	Axis No.
	iValue	The range value that is set.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	Reversed function	

Instruction 15	ZAux_Direct_GetMark	
Original Format	iint32 __stdcall ZAux_Direct_GetMark(ZMC_HANDLE handle,int iaxis, int *piValue);	
Description	Read returned state of latch event 0.	
Input parameters	Parameter name	Description
	handle	Link mark.
	iaxis	Axis No.
Output parameters	Parameter name	Description
	piValue	Modes, 0-doesn't occur, 1-occur.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Position latch	
Details	- When "latch" is executing, piValue becomes true, that is, return -1. After completed, it becomes false, return 0. - Corresponding latch channel R0. - Controller IN0 corresponds to latch channel R0 by default, IN1 corresponds to latch channel R1.	

Instruction 16	ZAux_Direct_GetMarkB	
Original Format	int32 __stdcall ZAux_Direct_GetMarkB(ZMC_HANDLE handle,int iaxis, int *piValue);	

Description	Read returned state of latch event 1.	
Input parameters	Parameter name	Description
	handle	Link mark.
	iaxis	Axis No.
Output parameters	Parameter name	Description
	piValue	Modes, 0-doesn't occur, 1-occur.
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Position latch	
Details	- When "latch" is executing, piValue becomes true, that is, return -1. After completed, it becomes false, return 0. - Corresponding latch channel R0. - Controller IN0 corresponds to latch channel R0 by default, IN1 corresponds to latch channel R1.	

Instruction 17	ZAux_Direct_Pswitch	
Original Format	int32 __stdcall ZAux_Direct_Pswitch(ZMC_HANDLE handle,int num, int enable,int axisnum,int outnum,int outstate,float setpos,float resetpos)	
Description	Set soft comparison output.	
Input parameters	Parameter name	Description
	handle	Link mark.
	num	Software comparer No., different modes are with different comparers.
	enable	Enable comparer: 0-OFF, 1-ON.
	axisnum	Motion axis No. of comparison output.
	outnum	Output No. to be operated.
	outstate	Output state, 1-ON, 0-OFF.
	setpos	Starting position of comparison output
	resetpos	Ending position of comparison output
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Software position comparison	
Details	- If multiple PSWITCH commands are used to operate one output, please do it in sequence. - For pulse motor, MPOS (measurement position) is compared only when ATYPE = 4 (factory ATYPE is 1 / 4, which compares DPOS – target position).	

Instruction 18	ZAux_Direct_HwPswitch	
Original Format	int32 __stdcall ZAux_Direct_HwPswitch(ZMC_HANDLE handle,int Axisnum,int Mode, int Direction, int Reserve, int Tablestart, int Tableend)	

Description	Set hardware position comparison output. ZMC4XX series products support pulse axis and encoder axis.	
Input parameters	Parameter name	Description
	handle	Link mark.
	Axisnum	Comparison output axis No.
	Mode	Comparer operations: 1 = open comparer, 2 = stop and delete uncompleted point
	Direction	Comparison direction: 0-negative, 1-positive.
	Reserve	Reserved
	Tablestart	TABLE starting address
	Tableend	TABLE ending address
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Fly-shooting	
Details	● If HW_PSWITCH has not compared all the points, please set mode as 2 -- HW_PSWITCH(2): to stop & delete unfinished comparison points. ● The default axes 0-5 correspond to OP1, OP1, OP2, OP3, OP0, OP1 respectively. There are 4 comparison outputs in total. ● Two HW_PSWITCH instructions can be used continuously, and how many commands called can be known by PC command. ● Each trigger of comparison point will invert the OP level. ● There are 1024 HW buffers, so 1024 HW commands can be called continuously. ● After calling HW instruction, which won't be affected by following coordinates. And please ensure called coordinates in TABLE are correct. Therefore, it's better to modify coordinates manually to avoid random clashing caused by HW commands and coordinates cyclic auto modification. ● Due to auto-cycle-coordinates configuration is not controlled by program, HW is in the front or behind can't be determined, which means coordinates in TABLE also can't be determined. ● This command is only valid in pulse axis. For bus axis, please use the HW_PSWITCH2. ● For pulse motor, MPOS (measurement position) is compared only when ATYPE = 4 (factory ATYPE is 1 / 4, which compares DPOS – target position).	

Instruction 19	ZAux_Direct_HwPswitch2
Original Format	int32 __stdcall ZAux_Direct_HwPswitch2(ZMC_HANDLE handle,int Axisnum,int Mode, int Opnum, int Opstate, float ModeParal, float ModePara2, float Mode Para3 , float ModePara4)

Description	Hardware position comparison output 2 Valid in ZMC4XX products whose firmware 20170513 or above. And it is valid in ZMC306E/ZMC306N.	
Input parameters	Parameter name	Description
	handle	Link mark.
	Axisnum	Comparison output axis No.
	Mode	Mode 1 - enable comparator ✓ ModePara1 = TABLE No. where saves first comparison coordinates ✓ ModePara2 = TABLE No. where saves end comparison coordinates ✓ ModePara3= The direction of the first point, 0-negative, 1-positive, -1-do not use the direction ✓ ModePara4 = Reserved
		Mode 2 - stop and delete uncompleted comparison points ✓ ModePara1 = Reserved ✓ ModePara2 = Reserved ✓ ModePara3 = Reserved ✓ ModePara4 = Reserved
		Mode 3 - vector comparison ✓ ModePara1 = spindle No. ✓ ModePara2 = Reserved ✓ ModePara3 = Reserved ✓ ModePara4 = Reserved
		Mode 4 - vector comparison, single comparison point ✓ ModePara1 = compare point coordinates ✓ ModePara2 = Reserved ✓ ModePara3 = Reserved ✓ ModePara4 = Reserved
		Mode 5 - vector comparison mode, periodic pulse mode ✓ ModePara1 = compare point coordinates ✓ ModePara2 = repeat cycle, compare twice in one cycle, first outputs the valid state, then outputs the invalid state

		✓ ModePara3 = periodic distance, output Opstate for each distance, output valid state distance (ModePara4) & restore to invalid state ✓ ModePara4 = output valid state distance, (ModePara1-ModePara4) bit invalid state distance Mode 6 - vector comparison, periodic mode, this mode is used together with HW_TIMER ✓ ModePara1 = compare point coordinates ✓ ModePara2 = repeat cycle, a coordinate is only compared once ✓ ModePara3 = cycle distance, output every time this distance ✓ ModePara4 = Reserved Mode 8 - open comparer, single-axis comparison, same as MODE1, but it won't invert signal. Also, it can be used together with HW_TIMER. Generally, used for single-axis fly-shooting (trigger each position after one certain of time) ✓ ModePara1 = TABLE No. where saves first comparison coordinates ✓ ModePara2 = TABLE No. where saves end comparison coordinates ✓ ModePara3= The direction of the first point, 0-negative, 1-positive, -1-do not use the direction ✓ ModePara4 = Reserved
	Opnum	Output No.
	Opstate	Output state of the first comparison point. 0-OFF, 1-ON.
	ModePara1	Multiple function parameter.
	ModePara2	Multiple function parameter.
	ModePara3	Multiple function parameter.
	ModePara4	Multiple function parameter.
	OP Parameters /	
	Return value	0 = success, not-0 = failure (please check error codes for details).
	Example	Fly-shooting (one axis & multi-camera)

Details	● Generally, there are 4 comparison outputs (OUT0, OUT1, OUT2, OUT3) for ZMC4XX controllers. ● When comparing the main axis that is with encoder input, the encoder position is automatically used to trigger. You also can use MOVEOP_DELAY to adjust the exact time of output. ● HW_PSWITCH2 and MOVE_OP (precision output) use the same hardware resources, so it is not recommended to use them on the same channel at the same time, but they can be used on different channels at the same time. ● It only can be compared once in each system-cycle, the system cycle can be checked through SERVO_PERIOD. ● Do not modify the TABLE position data until all comparison points are completed. ● This is valid in both pulse axis and bus axis. ● For pulse motor, MPOS (measurement position) is compared only when ATYPE = 4 (factory ATYPE is 1 / 4, which compares DPOS – target position).

Instruction 20	ZAux_Direct_GetHwPswitchBuff						
Original Format	int32 __stdcall ZAux_Direct_GetHwPswitchBuff(ZMC_HANDLE handle, int axisnum, int *buff)						
Description	Read remain buffers of hardware position comparison output. It is valid in 4xx series pulse axis and encoder axis.						
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark.</td></tr> <tr> <td>Axisnum</td><td>Axis No. of comparison output</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark.	Axisnum	Axis No. of comparison output
Parameter name	Description						
handle	Link mark.						
Axisnum	Axis No. of comparison output						
Output parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Buff</td><td>Remaining buffer</td></tr> </tbody> </table>	Parameter name	Description	Buff	Remaining buffer		
Parameter name	Description						
Buff	Remaining buffer						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	/						
Details	/						

Instruction 21	ZAux_Direct_HwTimer						
Original Format	int32 __stdcall ZAux_Direct_HwTimer(ZMC_HANDLE handle, int mode, int cyclonetime, int optime, int reptimes, int opstate, int opnum)						
Description	Hardware timer is used to restore electric level after one certain of time of hardware comparison output, which is valid in 4xx series products.						
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark.</td></tr> <tr> <td>mode</td><td>0-OFF, 2-ON</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark.	mode	0-OFF, 2-ON
Parameter name	Description						
handle	Link mark.						
mode	0-OFF, 2-ON						

	cyclonetime	Period time, unit: us
	optime	Valid time, unit: us
	reptime	Repeat times
	opstate	It starts timing when output becomes the state that is not the state.
	opnum	OP No. (must support comparison out)
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	<pre>//period is adjusted as 2, output twice. ZAux_Trigger(g_handle); ZAux_Direct_SetOp(g_handle, 0, 0); ZAux_Direct_HwTimer(g_handle, 2, 1000000, 500000, 2, 0, 0); //after OUT0 became ON, hardware timer is triggered to do timing, after 500ms, it will switch to off. ZAux_Direct_SetOp(g_handle, 0, 1);</pre>	
Details	● There is only one HW_TIMER, and each call will forcibly stop the previous call. ● The HW_TIMER of each OUT of ZMC420SCAN is independent. ● Valid in some ZMC3 series, 4 series and above products. ● OP and MOVE_OP operations will turn off the ongoing HW_TIMER pulse, so that HW_TIMER can be used like PWM: the OP outputs, to turn on the pulse output, and the next OP outputs, to turn off the pulse output. When MOVE_OP is used for precise output, precise PWM output infinite pulse function output can be achieved.	

Instruction 22	ZAux_Direct_MultiMovePt	
Original Format	int32 __stdcall ZAux_Direct_MultiMovePt(ZMC_HANDLE handle,int iMoveLen, int imaxaxes, int *piAxislist, uint32 *pTicklist, float *pfDisancelist)	
Description	Set the motion distance for diver motor within the specified ticks. Ticks: the time, about 1ms	
Input parameters	Parameter name	Description
	handle	Link mark.
	iMoveLen	Filled motion numbers
	imaxaxes	Total axes that participated in motion.
	piAxislist	Axis No. list
	pTicklist	Ticks time list
	pfDistancelist	Motion distance list
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	PT motion	
Details	● Generally, the PC calculates the corresponding coordinates	

	every cycle, and then transmits them to the controller. ● Speed during motion = (motion distance/time) * 1000 units/s. ● Do not move a large distance in a very short time, the pulse frequency will be too high, then the motor will stall. It can be broken down into small segments and sent repeatedly.

Instruction 23	ZAux_Direct_MultiMovePtAbs														
Original Format	int32 __stdcall ZAux_Direct_MultiMovePtAbs(ZMC_HANDLE handle, int iMoveLen, int imaxaxes, int *piAxislist, uint32 *pTicklist, float *pfDisancelist)														
Description	Set the absolute motion distance for diver motor within the specified ticks. Ticks: the time, about 1ms														
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark.</td></tr> <tr> <td>iMoveLen</td><td>Filled motion numbers</td></tr> <tr> <td>imaxaxes</td><td>Total axes that participated in motion.</td></tr> <tr> <td>piAxislist</td><td>Axis No. list</td></tr> <tr> <td>pTicklist</td><td>Ticks time list</td></tr> <tr> <td>pfDistancelist</td><td>Motion distance list</td></tr> </table>	Parameter name	Description	handle	Link mark.	iMoveLen	Filled motion numbers	imaxaxes	Total axes that participated in motion.	piAxislist	Axis No. list	pTicklist	Ticks time list	pfDistancelist	Motion distance list
Parameter name	Description														
handle	Link mark.														
iMoveLen	Filled motion numbers														
imaxaxes	Total axes that participated in motion.														
piAxislist	Axis No. list														
pTicklist	Ticks time list														
pfDistancelist	Motion distance list														
OP Parameters	/														
Return value	0 = success, not-0 = failure (please check error codes for details).														
Example	PT motion														
Details	● Generally, the PC calculates the corresponding coordinates every cycle, and then transmits them to the controller. ● Speed during motion = (motion distance/time) * 1000 units/s. ● Do not move a large distance in a very short time, the pulse frequency will be too high, then the motor will stall. It can be broken down into small segments and sent repeatedly.														

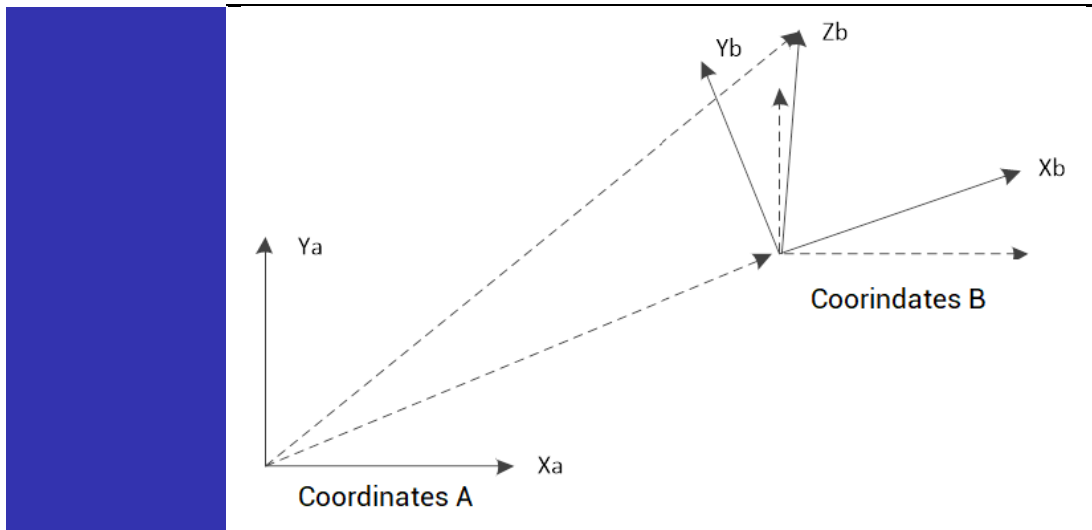
Instruction 24	ZAux_Direct_Connframe																
Original Format	int32 __stdcall ZAux_Direct_Connframe(ZMC_HANDLE handle, int Jogmaxaxes, int *JogAxislist, int frame, int tablenum, int Virmaxaxes, int *VirAxislist)																
Description	Build robotic arm inverse kinematic.																
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark.</td></tr> <tr> <td>Jogmaxaxes</td><td>The number of joint axes</td></tr> <tr> <td>JogAxislist</td><td>The list of joint-axis</td></tr> <tr> <td>frame</td><td>Robotic arm type, 1-scara, 3-Palletizing</td></tr> <tr> <td>tablenum</td><td>TABLE position where starts to save robot related parameters.</td></tr> <tr> <td>Virmaxaxes</td><td>Virtual axes numbers in joint motion</td></tr> <tr> <td>VirAxislist</td><td>Virtual axes list in joint motion</td></tr> </table>	Parameter name	Description	handle	Link mark.	Jogmaxaxes	The number of joint axes	JogAxislist	The list of joint-axis	frame	Robotic arm type, 1-scara, 3-Palletizing	tablenum	TABLE position where starts to save robot related parameters.	Virmaxaxes	Virtual axes numbers in joint motion	VirAxislist	Virtual axes list in joint motion
Parameter name	Description																
handle	Link mark.																
Jogmaxaxes	The number of joint axes																
JogAxislist	The list of joint-axis																
frame	Robotic arm type, 1-scara, 3-Palletizing																
tablenum	TABLE position where starts to save robot related parameters.																
Virmaxaxes	Virtual axes numbers in joint motion																
VirAxislist	Virtual axes list in joint motion																

OP Parameters	/
Return value	0 = success, not-0 = failure (please check error codes for details).
Example	Build Robotic Arm
Details	- When the connection speed is 0, the motion speed and acceleration of the joint coordinate system are limited by parameters such as SPEED. - When there is an error such as joint axis alarm, this movement will be cancelled by CANCEL. Do not CANCEL this movement during the high-speed motion of the virtual axis. - When this command is LOAD, it will automatically make the virtual axis' coordinates consistent with joint axis. Therefore, after called, it needs WAIT LOADED before starting to move. - Do not modify the coordinates of the virtual axis during the connection, or don't call motions (like, DATUM) that may change the coordinates, which will cause the joint axis to quickly move to the new virtual position. After CONNFRAME takes effect, the MTYPE of the joint axis is 33. At this time, the joint axis cannot be moved directly, and the joint axis must be moved indirectly by running the virtual axis. When the joint axis needs to be moved directly, please call CANCEL to cancel the CONNFRAME at first and then move the joint axis. - When both virtual axis and actual axis are rotating axes (end rotary axis), their pulse amount (UNITS) should be consistent.

Instruction 25	ZAux_Direct_Connreframe	
Original Format	int32 __stdcall ZAux_Direct_Connreframe(ZMC_HANDLE handle, int Virmaxaxes, int *VirAxislist, int frame, int tablenum, int Jogmaxaxes, int *JogAxislist)	
Description	Build robotic arm forward kinematic.	
Input parameters	Parameter name	Description
	handle	Link mark.
	Virmaxaxes	The number of joint axes
	VirAxislist	The list of joint-axis
	frame	Robotic arm type, 1-scara, 3-Palletizing
	tablenum	TABLE position where starts to save robot related parameters.
	Jogmaxaxes	Virtual axes numbers in joint motion
	JogAxislist	Virtual axes list in joint motion
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Build Robotic Arm	

Details	This is the opposite instruction of inverse kinematic motion. - When virtual axis CAONNREFRAME is in LOAD, joint axis CONNFRAME motion will be CANCEL automatically. - When joint axis CONNFRAME is in LOAD, virtual axis CONNREFRAME motion will be CANCEL automatically.

Instruction 26	ZAux_Direct_FrameRotate	
Original Format	int32 __stdcall ZAux_Direct_FrameRotate(ZMC_HANDLE handle,int iaxis, float *pfRotatePara)	
Description	Rotate robotic arm coordinates, to translate and rotate the workpiece coordinate system.	
Input parameters	Parameter name	Description
	handle	Link mark.
	iaxis	Axis No., joint axis / virtual axis
	pfRotateParat[0]	The translation distance of coordinate system B along with X^
	pfRotateParat[1]	The translation distance of coordinate system B along with Y^
	pfRotateParat[2]	The translation distance of coordinate system B along with Z^
	pfRotateParat[3]	The rotary radian of coordinate system B along with X^
	pfRotateParat[4]	The rotary radian of coordinate system B along with Y^
	pfRotateParat[5]	The rotary radian of coordinate system B along with Z^
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	- At present, only FRAME6 mode's pose can be rotated at the same time. For others, XYZ virtual axes can be rotated, but the pose axis can't be rotated. - After rotation, the virtual axis WORLD_DPOS indicates that the world coordinates will not change, and the virtual axis DPOS indicates that the workpiece coordinates will change. When using, the mechanical connection to the controller is needed. - The BASE axis can be the virtual axis or the joint axis, if BASE axes don't connect one robot, it will report an error of 1025. - Under multiple manipulator superposition, identify which current robot mode is according to BASE axis, for example: if axis_1 in BASE (axis_1, axis_2) is the manipulator axis of mode 1, and axis_2 is the manipulator axis of mode 2, then the manipulator of mode 1 performs coordinate calculations, namely, they are calculated in the order of the BASE axis.	



Instruction 27	ZAux_Direct_MoveSync	
Original Format	int32 __stdcall ZAux_Direct_MoveSync(ZMC_HANDLE handle,float imode,int synctime, float syncposition, int syncaxis, int imaxaxes, int *piAxislist, float *pfDisancelist)	
Description	Synchronous motion, the object on the belt follows, this motion is non-interpolation motion, which means it can't ensure the trajectory is a straight line.	
Input parameters	Parameter name	Description
	handle	Link mark.
	imode	➤ imode = -1: end mode, move to the specified absolute position, if there are other ZAux_Direct_MoveSync behind, this will be overwritten, and the belt axis is invalid in this mode ➤ imode = -2: force to end, the original ZAux_Direct_MoveSync is forced to stop when called, and it will arrive to specified end position. if there are other ZAux_Direct_MoveSync behind, this will be overwritten, and the belt axis is invalid in this mode ➤ imode = 0: the first axis (in following-axis list) follows ➤ imode = 10: the second axis (in following-axis list) follows ➤ imode = 20: the third axis (in following-axis list) follows ➤ imode = 0+angle: angle: belt rotation angle, angle = positive rotation angle between the belt and the 1/2 axis in the list of following axes. For

		example, imode = $\pi/4$, the belt is in the direction of 45 degrees. imode = $\pi/2$, the belt is in the y direction. imode = π, the belt is in the negative x direction; imode = $(\pi \times 1.75)$, the belt is in the direction of -45 degrees ➤ imode=1000+tableindex, the tracking ratio of the axis No. list is stored in the TABLE in turn, so that 3D belt tracking can be realized when the belt is inclined. (Firmware version 20180209 adds this function) (tableindex: table register starting No.)
	synctime	Synchronization time, in ms unit. The movement is completed within the specified time, and the speed of the axis is consistent with that of the object on the belt axis when it is completed. 0 indicates that the synchronization time is estimated based on the velocity and acceleration of the motion axis, which may not be accurate.
	syncposition	The belt axis position when an object is sensed. It supports belt axis coordinate cycle, but when the command is called, ensure that there is no coordinate modification or cycle operation between the parameter position and the current belt axis position, so this command should not be near the coordinate cycle point when calling.
	syncaxis	Belt axis No.
	imaxaxes	The number of total slave axes that participate in synchronization.
	piAxislist	Following axis No. list
	pfDistancelist	Slave axis absolute coordinates position when belt axis object is sensed.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	Belt-axis Synchronization Motion Following	

Instruction 28	ZAux_Direct_MultiMovePvt
Original Format	int32 __stdcall ZAux_Direct_MultiMovePvtAbs(ZMC_HANDLE handle,int iMoveLen, int imaxaxes, int *piAxislist, uint32

	*pTicklist, float *pfDisancelist,float *pSpeedlist)	
Description	Set the motion distance for diver motor within the specified ticks, and it can assign the end speed, the speed of small-segment will be planned automatically to make it continuous according to former speed and end speed.	
Input parameters	Parameter name	Description
	handle	Link mark.
	iMoveLen	Filled motion numbers
	imaxaxes	Total axes that participated in motion.
	piAxislist	Axis No. list
	pTicklist	Ticks time list
	pfDistancelist	Motion distance list
	pSpeedlist	End speed list
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	PT motion	
Details	● Generally, the PC calculates the corresponding coordinates every cycle, and then transmits them to the controller. ● Do not move a large distance in a very short time, the pulse frequency will be too high, then the motor will stall. It can be broken down into small segments and sent repeatedly.	

Instruction 29	ZAux_Frame_Rotate	
Original Format	int32 __stdcall ZAux_Frame_Rotate(ZMC_HANDLE handle,int imaxaxes,int *piAxislist,float distance0,float distance1,float distance2,float Radian0,float Radian1,float Radian2)	
Description	Transform the workpiece coordinate system.	
Input parameters	Parameter name	Description
	handle	Link mark.
	imaxaxes	Total axes that participated in motion.
	piAxislist	Axis No. list
	distance0	The translation distance of coordinate system B along with X^
	distance1	The translation distance of coordinate system B along with Y^
	distance2	The translation distance of coordinate system B along with Z^
	Radian0	The rotary radian of coordinate system B along with X^
	Radian1	The rotary radian of coordinate system B along with Y^
	Radian2	The rotary radian of coordinate system B along with Z^

OP Parameters	/
Return value	0 = success, not-0 = failure (please check error codes for details).
Example	/
Details	- At present, only FRAME6 mode's pose can be rotated at the same time. For others, XYZ virtual axes can be rotated, but the pose axis can't be rotated. - After rotation, the virtual axis WORLD_DPOS indicates that the world coordinates will not change, and the virtual axis DPOS indicates that the workpiece coordinates will change. When using, the mechanical connection to the controller is needed.

Instruction 30	ZAux_Frame_Rotate2	
Original Format	int32 __stdcall ZAux_Frame_Rotate2(ZMC_HANDLE handle,int *piAxislist,int imaxaxes,int tablein,int tableout,int dir,float *distance,float *radian)	
Description	Calculate coordinate system transformation.	
Input parameters	Parameter name	Description
	handle	Link mark.
	piAxislist	Axis No. list
	imaxaxes	Total axes that participated in motion.
	tablein	TABLE position where saves filled coordinates before transformation.
	tableout	TABLE position where saves output coordinates after transformation.
	dir	Transformation direction: 0: from DPOS to WORLD_DPOS 1: from WORLD_DPOS to DPOS
	distance	The translation distance of coordinate system B along with X^, Y^, Z^.
	radian	The rotary radian of coordinate system B along with X^, Y^, Z^.
	OP Parameters	/
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	● Manually calculate the coordinates value after rotated. ● When using, the mechanical connection to the controller is needed. ● The BASE axis can be the virtual axis or the joint axis, if BASE axes don't connect one robot, it will report an error of 1025. ● Under multiple manipulator superposition, identify which current robot mode is according to BASE axis, for example: if axis_1 in BASE (axis_1, axis_2) is the manipulator axis of mode 1, and axis_2 is the manipulator axis of mode 2, then the	

	manipulator of mode 1 performs coordinate calculations, namely, they are calculated in the order of the BASE axis.

Instruction 31	ZAux_Direct_FrameStatus																
Original Format	int32 __stdcall ZAux_Direct_FrameStatus(ZMC_HANDLE handle,int axis,int piValue)																
Description	Set the robotic arm pose.																
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark.</td></tr> <tr> <td>iMoveLen</td><td>Filled motion numbers</td></tr> <tr> <td>imaxaxes</td><td>Total axes that participated in motion.</td></tr> <tr> <td>piAxislist</td><td>Axis No. list</td></tr> <tr> <td>pTicklist</td><td>Ticks time list</td></tr> <tr> <td>pfDistancelist</td><td>Motion distance list</td></tr> <tr> <td>pSpeedlist</td><td>End speed list</td></tr> </table>	Parameter name	Description	handle	Link mark.	iMoveLen	Filled motion numbers	imaxaxes	Total axes that participated in motion.	piAxislist	Axis No. list	pTicklist	Ticks time list	pfDistancelist	Motion distance list	pSpeedlist	End speed list
Parameter name	Description																
handle	Link mark.																
iMoveLen	Filled motion numbers																
imaxaxes	Total axes that participated in motion.																
piAxislist	Axis No. list																
pTicklist	Ticks time list																
pfDistancelist	Motion distance list																
pSpeedlist	End speed list																
OP Parameters	/																
Return value	0 = success, not-0 = failure (please check error codes for details).																
Example	/																
Details	● Assign the current robotic arm pose. ✓ When it is not in manipulator status, it will return -1. FRAME_TRANS2 command will use this pose. ✓ There are multiple poses for SCARA, SCARA-type, 6 DOF structures. ✓ SCRAR -- left hand pose value is 1, right hand pose value is 0.																

Instruction 32	ZAux_Direct_FrameTrans2												
Original Format	int32 __stdcall ZAux_Direct_FrameTrans2(ZMC_HANDLE handle,int *piAxislist,int imaxaxes,int tablein,int tableout,int dirmode)												
Description	Transform forward kinematics and inverse kinematics.												
Input parameters	<table> <tr> <th>Parameter name</th><th>Description</th></tr> <tr> <td>handle</td><td>Link mark.</td></tr> <tr> <td>piAxislist</td><td>Axis No. list</td></tr> <tr> <td>imaxaxes</td><td>Total axes that participated in motion.</td></tr> <tr> <td>tablein</td><td>TABLE index No., it starts to save data continuous from this index. For forward mode, it enters joint coordinates; for inverse mode, it enters virtual coordinates, then adds the pose.</td></tr> <tr> <td>tableout</td><td>TABLE index No., it starts to save data continuous from this index. For forward mode, it outputs virtual coordinates, and the pose is added at last; for inverse mode, it outputs joint coordinates.</td></tr> </table>	Parameter name	Description	handle	Link mark.	piAxislist	Axis No. list	imaxaxes	Total axes that participated in motion.	tablein	TABLE index No., it starts to save data continuous from this index. For forward mode, it enters joint coordinates; for inverse mode, it enters virtual coordinates, then adds the pose.	tableout	TABLE index No., it starts to save data continuous from this index. For forward mode, it outputs virtual coordinates, and the pose is added at last; for inverse mode, it outputs joint coordinates.
Parameter name	Description												
handle	Link mark.												
piAxislist	Axis No. list												
imaxaxes	Total axes that participated in motion.												
tablein	TABLE index No., it starts to save data continuous from this index. For forward mode, it enters joint coordinates; for inverse mode, it enters virtual coordinates, then adds the pose.												
tableout	TABLE index No., it starts to save data continuous from this index. For forward mode, it outputs virtual coordinates, and the pose is added at last; for inverse mode, it outputs joint coordinates.												

	dirmode	Modes: ➤ Mode 0 – inverse mode. From virtual axis to joint axis, no pose, auto-use current pose. ➤ Mode 1 – forward mode. From joint axis to virtual axis, no pose output. ➤ Mode 2 – inverse mode. Enter the virtual axis coordinates, and add the pose at last. ➤ Mode 3 – forward mode. Enter the virtual axis coordinates, the last output position is "pose".
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	● It must be used when forward / inverse mode is built. ● And it must use correct BASE axis type: under inverse mode, please use virtual axes for BASE; under forward mode, please use joint axes for BASE. ● Data also should be entered in correct sequence, which is consistent with "?*frame".	

Instruction 33	ZAux_Direct_FrameCal	
Original Format	int32 __stdcall ZAux_Direct_FrameCal(ZMC_HANDLE handle,int *piAxislist,int imaxaxes,int tablein,int space,int groups,int tableaux,int zeroout,int tableout2)	
Description	Correct the parameters.	
Input parameters	Parameter name	Description
	handle	Link mark.
	piAxislist	Axis No. list
	imaxaxes	Total axes that participated in motion.
	tablein	TABLE index No., which starts to save each point's joint coordinates in order, and spans "space" between points.
	space	How many TABLE elements between 2 points.
	tableaux	TABLE No. of auxiliary parameters, it is used for some FRAME structures.
	zeroout	TABLE No. of joint axis absolute coordinates that is according to calculated theoretical origin.
	tableout2	The position where saves calculated robot parameters. If not to fill in, it will

		use original parameters.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details	● The robot's parameters are automatically corrected based on the coordinates and characteristics of the robot's joints during teaching. ✓ Obtain the joint axis coordinates that are stored in tablein. Then control the robot terminal point to the correction point when the robot is connected by using current origin and robot parameters. At last, get the correction point's joint-axis coordinates. ✓ Calculate theoretical origin's joint-axis coordinates through correcting the deviation between current origin position and theoretical origin. ✓ Calculate theoretical mechanical parameters values through correcting some robot parameters. ✓ This is only used for calculation: returned value -1 means success, 0 means failure. ✓ And axis / axes in BASE must be under FRAME status.	

Instruction 34	ZAux_Direct_MoveHwPswitch2	
Original Format	int32 __stdcall ZAux_Direct_MoveHwPswitch2(ZMC_HANDLE handle,int Axisnum,int Mode, int Opnum , int Opstate, float ModePara1, float ModePara2,float ModePara3,float ModePara4,float HwTimerOpenTime,float HwTimerCycTime,int HwTimerReptimes)	
Description	Hard comparison output, in motion buffer way.	
Input parameters	Parameter name	Description
	handle	Link mark.
	Axisnum	Comparison output axis No.
	Mode	Mode 1 - enable comparator ✓ ModePara1 = TABLE No. where saves first comparison coordinates ✓ ModePara2 = TABLE No. where saves end comparison coordinates ✓ ModePara3= The direction of the first point, 0-negative, 1-positive, -1-do not use the direction ✓ ModePara4 = Reserved Mode 2 - stop and delete uncompleted comparison points ✓ ModePara1 = Reserved

		✓ ModePara2 = Reserved ✓ ModePara3 = Reserved ✓ ModePara4 = Reserved Mode 3 - vector comparison ✓ ModePara1 = spindle No. ✓ ModePara2 = Reserved ✓ ModePara3 = Reserved ✓ ModePara4 = Reserved Mode 4 - vector comparison, single comparison point ✓ ModePara1 = compare point coordinates ✓ ModePara2 = Reserved ✓ ModePara3 = Reserved ✓ ModePara4 = Reserved Mode 5 - vector comparison mode, periodic pulse mode ✓ ModePara1 = compare point coordinates ✓ ModePara2 = repeat cycle, compare twice in one cycle, first outputs the valid state, then outputs the invalid state ✓ ModePara3 = periodic distance, output Opstate for each distance, output valid state distance (ModePara4) & restore to invalid state ✓ ModePara4 = output valid state distance, (ModePara1-ModePara4) bit invalid state distance Mode 6 - vector comparison, periodic mode, this mode is used together with HW_TIMER ✓ ModePara1 = compare point coordinates ✓ ModePara2 = repeat cycle, a coordinate is only compared once ✓ ModePara3 = cycle distance, output every time this distance ✓ ModePara4 = Reserved

		Mode 8 – open comparer, single-axis comparison, same as MODE1, but it won't invert signal. Also, it can be used together with HW_TIMER. Generally, used for single-axis fly-shooting (trigger each position after one certain of time) ✓ ModePara1 = TABLE No. where saves first comparison coordinates ✓ ModePara2 = TABLE No. where saves end comparison coordinates ✓ ModePara3= The direction of the first point, 0-negative, 1-positive, -1-do not use the direction ✓ ModePara4 = Reserved
	Opnum	Output No.
	Opstate	Output state of the first comparison point. 0-OFF, 1-ON.
	ModePara1	Multiple function parameter.
	ModePara2	Multiple function parameter.
	ModePara3	Multiple function parameter.
	ModePara4	Multiple function parameter.
	HwTimerOpenTime	Valid time, unit: us
	HwTimerCycTime	Period time, unit: us
	HwTimerRepTimes	Repeat times.
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details		

Instruction 35	ZAux_Direct_HwPswitch2_2	
Original Format	int32 ZAux_Direct_HwPswitch2_2(ZMC_HANDLE handle, int *AxisList,int Axisnum,int Mode,int Opnum , int Opstate,int maxerr,int TmodePara1, int TmodePara2,bool ModeFlag, float ModePara1, float ModePara2,float ModePara3)	
Description	Hard comparison output 2, this is compatible with ZAux_Direct_HwPswitch2. It is valid in ZMC4XX series & the firmware above 20170513. When it is used together with HW_TIMER, HW_TIMER parameters can be adjust dynamically.	
Input parameters	Parameter name	Description
	handle	Link mark.
	AxisList	Axis array
	Axisnum	Axis numbers

	Mode	✚ [2D comparison: mode 25 and mode 26, each 2 TABLE save one point] ✚ [3D comparison: mode 35 and mode 36, each 3 TABLE save one point] ✚ When there are several comparison points, it will invert the state after each output. ✚ Under 2D and 3D, don't enter 0 for "maxerr" parameter. Mode 7 – use together with HW_TIMER Write comparison points into TABLE, and coordinates all refer to vector coordinates. After arrived one TABLE position, OP will be triggered. At this time, OP pulse width and comparison times of each trigger are controlled by HW_TIMER. After next TABLE, OP is triggered again. Mode 25 – 2D comparison Write comparison points into TABLE (2 continuous TABLE data = one 2D coordinate). After arrived one TABLE position, OP will be inverted (in below example, blue one = OP is ON), this is valid in all kinds of interpolations. Comparison points must be correct, otherwise following points will be affected. Mode 26 – 2D comparison, use together with HW_TIMER Write comparison points into TABLE (2 continuous TABLE data = one 2D coordinate). After arrived one TABLE position, OP will be triggered. At this time, OP invert times and invert periods of each comparison point are controlled by HW_TIMER. After next TABLE, OP is triggered again. Similar to mode 7 and mode 36. Mode 35 – 3D comparison Write comparison points into TABLE (3

		continuous TABLE data = one 3D coordinate). After arrived one TABLE position, OP will be inverted (in below example, blue one = OP is ON), this is valid in all kinds of interpolations. Comparison points must be correct, otherwise following points will be affected. Mode 36 – 3D comparison, use together with HW_TIMER Write comparison points into TABLE (3 continuous TABLE data = one 3D coordinate). After arrived one TABLE position, OP will be triggered. At this time, OP invert times and invert periods of each comparison point are controlled by HW_TIMER. After next TABLE, OP is triggered again. Similar to mode 7 and mode 26.
	Opnum	Output No.
	Opstate	Output state of the first comparison point. 0-OFF, 1-ON.
	maxerr	Pulse deviation of each axis, it starts to compare after entered this range.
	TmodePara1	For mode 7: it is TABLE No. where saves the first comparison point VECTOR_MOVED. For mode 25, 26, 35, 36, it is the number of comparison points saved in TABLE.
	TmodePara2	For mode 7: it is TABLE No. where saves the first comparison point VECTOR_MOVED. For mode 25, 26, 35, 36, it is TABLE No. where saved the first comparison point.
	ModeFlag	Whether to open "dynamically adjust HWTIMER parameter"
	ModePara1	Pulse time
	ModePara2	Pulses
	ModePara3	Pulse period
OP Parameters	/	
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	

Instruction 36	ZAux_Direct_SetHwPs2AxisNum								
Original Format	int32 __stdcall ZAux_Direct_SetHwPs2AxisNum(ZMC_HANDLE handle,int iaxis1, int iaxis2)								
Description	Set the real axis No. for HW_PSWITCH2, default -1 = not change. Used to reuse axis HW_PSWITCH2 (not used), to assign the current motion spindle. It can do multi-comparison for current spindle at the same time.								
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark.</td></tr> <tr> <td>iaxis1</td><td>Occupied buffer axis No.</td></tr> <tr> <td>iaxis2</td><td>Bound real axis No.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark.	iaxis1	Occupied buffer axis No.	iaxis2	Bound real axis No.
Parameter name	Description								
handle	Link mark.								
iaxis1	Occupied buffer axis No.								
iaxis2	Bound real axis No.								
OP Parameters	/								
Return value	0 = success, not-0 = failure (please check error codes for details).								
Example	/								
Details									

Instruction 37	ZAux_Direct_GetHwPs2AxisNum						
Original Format	int32 __stdcall ZAux_Direct_GetHwPs2AxisNum(ZMC_HANDLE handle,int iaxis1, int* iaxis2)						
Description	Get the real axis No. of HW_PSWITCH2.						
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark.</td></tr> <tr> <td>iaxis1</td><td>Occupied buffer axis No.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark.	iaxis1	Occupied buffer axis No.
Parameter name	Description						
handle	Link mark.						
iaxis1	Occupied buffer axis No.						
OP Parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark.</td></tr> <tr> <td>iaxis2</td><td>Bound real axis No.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark.	iaxis2	Bound real axis No.
Parameter name	Description						
handle	Link mark.						
iaxis2	Bound real axis No.						
Return value	0 = success, not-0 = failure (please check error codes for details).						
Example	/						
Details							

Instruction 38	ZAux_Direct_GetHwPs2Counts						
Original Format	int32 __stdcall ZAux_Direct_GetHwPs2Counts(ZMC_HANDLE handle,int iaxis, int* ivalue)						
Description	Get how many points that had been compared by HW_PSWITCH2. For HW_PSWITCH(2), all points are cleared to 0.						
Input parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>handle</td><td>Link mark.</td></tr> <tr> <td>iaxis</td><td>Axis No.</td></tr> </tbody> </table>	Parameter name	Description	handle	Link mark.	iaxis	Axis No.
Parameter name	Description						
handle	Link mark.						
iaxis	Axis No.						
OP Parameters	<table border="1"> <thead> <tr> <th>Parameter name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>ivalue</td><td>Real compared points</td></tr> </tbody> </table>	Parameter name	Description	ivalue	Real compared points		
Parameter name	Description						
ivalue	Real compared points						

Return value	0 = success, not-0 = failure (please check error codes for details).
Example	/
Details	

Instruction 39	ZAux_World_DPos	
Original Format	int32 __stdcall ZAux_World_DPos(ZMC_HANDLE handle,int axis,float*wdpos)	
Description	World coordinate system. The virtual axis refers to world coordinate system values. When there is no rotation, it is same as DPOS.	
Input parameters	Parameter name	Description
	handle	Link mark.
	iaxis	Axis No.
OP Parameters	Parameter name	Description
	wdpos	Corresponding world coordinate system
Return value	0 = success, not-0 = failure (please check error codes for details).	
Example	/	
Details		

Instruction 40	ZAux_Direct_Mover_C	
Original Format	int32 __stdcall ZAux_Direct_Mover_C(ZMC_HANDLE handle,int imaxaxes,int *piAxislist, float end1,float end2,float centre1,float centre2,int mode,float distance0, float distance1, float distance2)	
Description	Make joint-axis to move one plane circular interpolation directly, relative distance.	
Input parameters	Parameter name	Description
	handle	Link mark
	imaxaxes	Total axes
	piAxislist	Axis list array
	end1	1 st axis motion distance 1, relative
	end2	2 nd axis motion distance 1, relative
	centre1	1 st axis motion distance 2, relative
	centre2	2 nd axis motion distance 2, relative
	mode	Specify previous parameters' functions: 0: use 3 points (current point + middle point + end point) to draw the arc. "distance 1": the end point distance. "distance 2": the middle point distance. 1: use current point + circle center + end point to draw the smallest arc. "distance 1": the end point distance.

		"distance 2": the circle center distance. 2: use 3 points (current point + middle point + end point) to draw the circle. "distance 1": the end point distance. "distance 2": the middle point distance. 3: use current point + circle center + end point to draw the circle, move the smallest arc. "distance 1": the end point distance "distance 2": the circle center distance	
	distance0	4 th axis motion distance	
	distance1	5 th axis motion distance	
	distance2	6 th axis motion distance	
OP Parameters			
Return value	Refer to error code		
Example	/		
Details			

Instruction 41	ZAux_Direct_Mover_CABS		
Original Format	int32 __stdcall ZAux_Direct_Mover_CABS(ZMC_HANDLE handle,int imaxaxes,int *piAxislist, float end1,float end2,float centre1,float centre2,int mode,float distance0, float distance1, float distance2)		
Description	Make joint-axis to move one plane circular interpolation directly, absolute distance.		
Input parameters	Parameter name	Description	
	handle	Link mark	
	imaxaxes	Total axes	
	piAxislist	Axis list array	
	end1	1 st axis motion distance 1, absolute	
	end2	2 nd axis motion distance 1, absolute	
	centre1	1 st axis motion distance 2, absolute	
	centre2	2 nd axis motion distance 2, absolute	
	mode	Specify previous parameters' functions: 0: use 3 points (current point + middle point + end point) to draw the arc. "distance 1": the end point distance. "distance 2": the middle point distance. 1: use current point + circle center + end point to draw the smallest arc. "distance 1": the end point distance. "distance 2": the circle center distance.	

		2: use 3 points (current point + middle point + end point) to draw the circle. "distance 1": the end point distance. "distance 2": the middle point distance. 3: use current point + circle center + end point to draw the circle, move the smallest arc. "distance 1": the end point distance "distance 2": the circle center distance	
	distance0	4 th axis motion distance	
	distance1	5 th axis motion distance	
	distance2	6 th axis motion distance	
OP Parameters			
Return value	Refer to error code		
Example	/		
Details			

Instruction 42	ZAux_Direct_Mover_C3		
Original Format	int32 __stdcall ZAux_Direct_Mover_C3(ZMC_HANDLE handle,int imaxaxes,int *piAxislist,float end1,float end2,float end3,float centre1,float centre2,float centre3,int mode, float distance0, float distance1, float distance2)		
Description	Make joint-axis to move one space circular interpolation directly, relative distance.		
Input parameters	Parameter name	Description	
	handle	Link mark	
	imaxaxes	Total axes	
	piAxislist	Axis list array	
	end1	1 st axis motion distance 1, relative	
	end2	2 nd axis motion distance 1, relative	
	end3	3 rd axis motion distance 1, relative	
	centre1	1 st axis motion distance 2, relative	
	centre2	2 nd axis motion distance 2, relative	
	centre3	3 rd axis motion distance 2, relative	
	mode	Specify previous parameters' functions: 0: use 3 points (current point + middle point + end point) to draw the arc. "distance 1": the end point distance. "distance 2": the middle point distance. 1: use current point + circle center + end point to draw the smallest arc. "distance 1": the end point distance.	

		"distance 2": the circle center distance. 2: use 3 points (current point + middle point + end point) to draw the circle. "distance 1": the end point distance. "distance 2": the middle point distance. 3: use current point + circle center + end point to draw the circle, move the smallest arc. "distance 1": the end point distance "distance 2": the circle center distance	
	distance0	4 th axis motion distance	
	distance1	5 th axis motion distance	
	distance2	6 th axis motion distance	
OP Parameters			
Return value	Refer to error code		
Example	/		
Details			

Instruction 43	ZAux_Direct_Mover_C3ABS		
Original Format	<pre>int32 __stdcall ZAux_Direct_Mover_C3ABS(ZMC_HANDLE handle,int imaxaxes,int *piAxislist,float end1,float end2,float end3,float centre1,float centre2,float centre3,int mode, float distance0, float distance1, float distance2)</pre>		
Description	Make joint-axis to move one space circular interpolation directly, absolute distance.		
Input parameters	Parameter name	Description	
	handle	Link mark	
	imaxaxes	Total axes	
	piAxislist	Axis list array	
	end1	1 st axis motion distance 1, absolute	
	end2	2 nd axis motion distance 1, absolute	
	end3	3 rd axis motion distance 1, absolute	
	centre1	1 st axis motion distance 2, absolute	
	centre2	2 nd axis motion distance 2, absolute	
	centre3	3 rd axis motion distance 2, absolute	
	mode	Specify previous parameters' functions: 0: use 3 points (current point + middle point + end point) to draw the arc. "distance 1": the end point distance. "distance 2": the middle point distance. 1: use current point + circle center + end point to draw the smallest arc.	

		"distance 1": the end point distance. "distance 2": the circle center distance. 2: use 3 points (current point + middle point + end point) to draw the circle. "distance 1": the end point distance. "distance 2": the middle point distance. 3: use current point + circle center + end point to draw the circle, move the smallest arc. "distance 1": the end point distance "distance 2": the circle center distance	
	distance0	4 th axis motion distance	
	distance1	5 th axis motion distance	
	distance2	6 th axis motion distance	
OP Parameters			
Return value	Refer to error code		
Example	/		
Details			

Instruction 44	ZAux_Direct_Mover_L	
Original Format	int32 __stdcall ZAux_Direct_Mover_L(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float *pfDisancelist);	
Description	Make joint axis do linear interpolation, relative distance. That is, move the manipulator joint axis (terminal) to assigned coordinates by linear motion. Note: this is used under forward mode, if you operate the joint axis, it may change the pose, so please make sure the start point's pose is consistent with end point's pose, otherwise it will report an error.	
Input parameters	Parameter name	Description
	handle	Link mark
	imaxaxes	Total axes
	piAxislist	Axis No. list
	pfDistancelist	Distance list
Return value	Refer to error code	

Instruction 45	ZAux_Direct_Mover_LABS		
Original Format	int32 __stdcall ZAux_Direct_Mover_LABS(ZMC_HANDLE handle, int imaxaxes, int *piAxislist, float *pfDisancelist);		
Description	Make joint axis do linear interpolation, absolute distance. That is, move the manipulator joint axis (terminal) to assigned coordinates by linear motion. Note: this is used under forward mode, if you operate the joint axis,		

	it may change the pose, so please make sure the start point's pose is consistent with end point's pose, otherwise it will report an error.	
Input parameters	Parameter name	Description
	handle	Link mark
	imaxaxes	Total axes
	piAxislist	Axis No. list
	pfDistancelist	Distance list
Return value	Refer to error code	

Instruction 46	ZAux_Direct_MoveHwTimer	
Original Format	int32__stdcall ZAux_Direct_MoveHwTimer(ZMC_HANDLE handle,int mode,int cyclonetime,int optime,int reptimes,int opstate,int opnum)	
Description	Hardware timer is used to restore electric level after one certain of time of hardware comparison output (in buffer way), which is valid in 4xx series products.	
Input parameters	Parameter name	Description
	handle	Link mark.
	mode	0-OFF, 2-ON
	cyclonetime	Period time, unit: us
	optime	Valid time, unit: us
	reptime	Repeat times
	opstate	It starts timing when output becomes the state that is not the state.
	opnum	OP No. (must support comparison out)
Return value	0 = success, not-0 = failure (please check error codes for details).	
Details	● There is only one HW_TIMER, and each call will forcibly stop the previous call. ● The HW_TIMER of each OUT of ZMC420SCAN is independent. ● Valid in some ZMC3 series, 4 series and above products. ● OP and MOVE_OP operations will turn off the ongoing HW_TIMER pulse, so that HW_TIMER can be used like PWM: the OP outputs, to turn on the pulse output, and the next OP outputs, to turn off the pulse output. When MOVE_OP is used for precise output, precise PWM output infinite pulse function output can be achieved.	