

**AMC4030** motion **controller**

**editing**

**Ride.**

**Hand.**

**Book.**

**(Version1.0).**

**Chengdu Fuyu Technology Co., Ltd**

**FUYU TECHNOLOGY CO.,LTD.**

**Copyright statement**

**Chengdu Fuyu Technology Co., Ltd**

**All rights reserved**

**Chengdu Fuyu Technology Co., Ltd. reserves the right to modify the finished products and product specifications in this manual without prior notice.**

**Chengdu Fuyu Technology Co., Ltd. shall not be liable for any direct, indirect, incidental or corresponding product loss or liability abroad as a result of improper use of this manual or this product**

**Chengdu Fuyu Technology Co., Ltd. has patents, copyrights and other intellectual property rights in this product and its software. This product and its related parts may not be reproduced, manufactured, processed or used directly or indirectly without authorization.**

|  |  |
| --- | --- |
| **C:\Users\Administrator\AppData\Local\Temp\0d0a6f46076592e0ff9ca24f81582ff1.gif**  **Attention.** | **Machines in motion are in danger! Users are responsible for designing effective error handling and safety protection mechanisms in the machine, and Chengdu Fuyu Technology Co., Ltd. has no obligation or liability for incidental or corresponding losses** |

**Contact us**

**Chengdu Fuyu Technology Co., Ltd**

**Address: 1455 West Port Avenue, Shuangliu County, Chengdu**

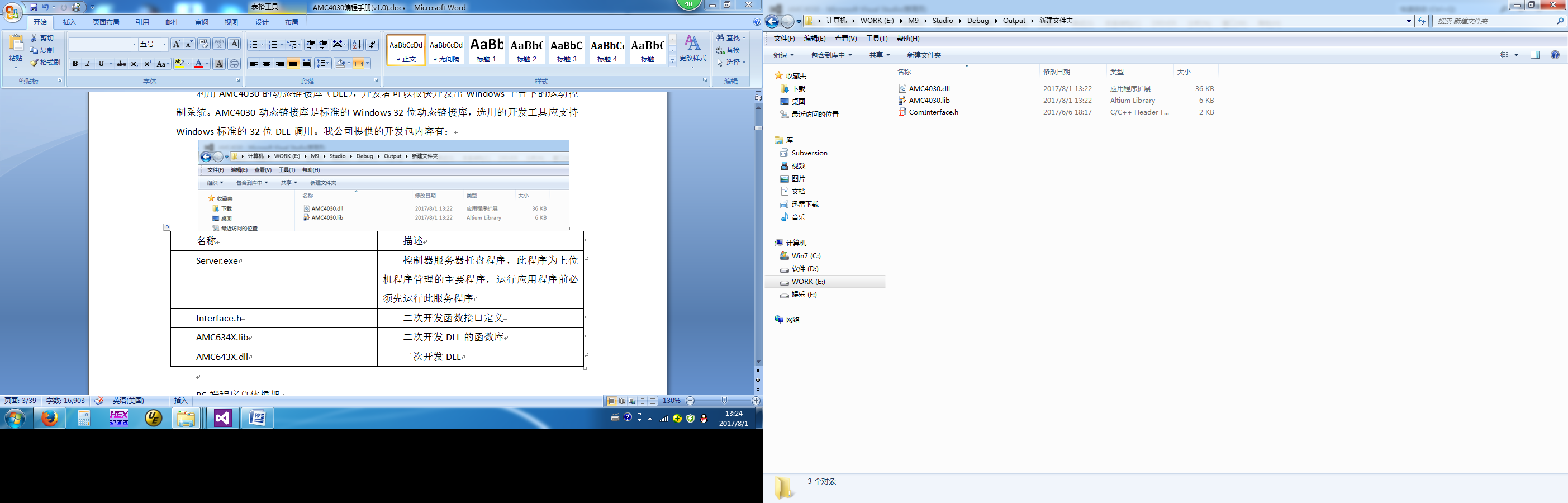
**Tel: 028-65058998**

**Fax: 028-65058997**

**Website: www.fuyuautomation.com**

# 1 Overview

With the AMC4030's Dynamic LinkLibrary (DLL),developers can quickly develop motion control systems under the Windows platform. The AMC4030 Dynamic Link Library is a standard Windows 32-bit dynamic link library, and the development tool of choice should support windows standard 32-bit DLL calls. The development package content provided by our company is as:



|  |  |
| --- | --- |
| Name. | Describe. |
| ComInterface.h | Secondary development function interface definition |
| AMC4030.lib | The library of functions for secondary development of DLLs |
| AMC4030.dll | Secondary development of DLLs |

The overall framework of the PC-side program

The application is secondaryly developed .dllfunction interface provided by the AMC4030, which enables its own motion control system. The AMC4030 is a bridge between the application and the AMC4030 motion controller, which enables data interaction and command delivery with the AMC4030 motion controller via the DLL.

AMC4030.dll

Application.

AMC4030

Motion controller

Default convention:

* Unless there is a special description (see function description), all functions must be called in one thread.
* The return values of all function interfaces have a uniform meaning (very important).

|  |  |
| --- | --- |
| Returns a value | Meaning. |
| 1 | The command is executed, the controller has received it correctly, and the correct value has been returned to the server side. |
| -1 | The command failed to execute, the command was determined to be illegal by the control, and the controller refused to execute the command (conditions for the possible execution of the command are not available) |
| -2 | The command parameter check failed and the command execution failed |
| -4 | The DLL's communication timed out with the controller. |
| -6 | A command execution conflict indicates that the last command has not yet been fully executed and a new command has arrived. (As shown: an interface call has not yet returned, and the new interface has started to be called again.) More common in multithreaded). |

## 1.1 Develop the Visual C++ control program

(i) Description

Users can use VC6.0 or later for motion control system development under the Windows platform. There are two ways to call a dynamic link library, using slightly different files.

(2) Dynamic link library function call method

(1) implicitly calls

The implicit call steps are as follows:

(a) to start the Visual C +and create a newproject;

(b) will develop the dynamic link library"AMC4030.dll", "AMC4030.lib"and the function declaration file "ComInterface. h"Copy to the project file;

(c)Select theSettingsmenu itemunder theProjectmenu;

(d) switch totheLinktabandenter the"AMC4030.lib"file name in theObject/library modulescolumn;

(e )Add the declaration file "Com Interface . . . to thelibrary header file"intheapplication h"; 。

(f )Calls the interface function at the appropriate place in the application.

(2) explicitly calls

Explicitly calling methods requires calling Windows API functions to load and release dynamic link libraries. Here's how:

(a)calls the Windows API function LoadLibrary() to dynamically load the DLL;

(b)Call the Windows API function GetProcAddress() to get a pointer to the function in the DLL to be called;

(c )to call the function in the DLL with a function pointer to complete the corresponding function;

(e)Call the Windows API function FreeLibrary() to release the dynamic link library at the end of the program or when the function in the DLL is no longer in use.

This method is more cumbersome. We've encapsulated commonly used DLL functions as CLASS CAMC4030DLLandprovided source code for that class. This class contains with AMC4030DLL. . h Member functions with the same function name and parameters. The source code can be requested from Fukuyu Technical Support, or written by itself, and the source code is named AMC4030DLL.cpp and AMC4030DLL.h. Developers can add it to the project, add objects of the class where the program is appropriate, and call functions in the DLL by corresponding member functions.

The call steps are as follows:

(a) to start the Visual C +and create a newproject;

(b) will be "Interface. . h" and function declaration files AMC4030DLL.cpp and AMC4030DLL.h are copied to the engineering file;

(c)Selectthe "Files"item ofthesub-menuof Add To ProjectundertheProjectmenu;

(d)Add AMC4030DLL .cpp and AMC4030DLL.h to the project;

(e) an object that generates CAMC4030DLL in the application, calling the motion function.

Both of these methods are standard methods for calling dynamic link library functions in a VC, and for more specific call methods and help, refer to the Microsoft Visual Stutio development documentation MSDN or the corresponding section of the relevant VC reference book.

# 2 Interface development guide

## 2.1 System initialization

### 2.1.1 Create a link

int nRtn;

int nType;

COM\_API\_SetComType(2); // Parameter 2:indicates that a USB communication link can only be passed 2.

nRtn COM\_API\_OpenLink(3,115200); Returns 1,indicating that the link was created successfully.

## 2.2 Interface functions are detailed

### 2.2.1 COM\_API\_SetComType ()

|  |  |
| --- | --- |
| The function prototype | int WINAPI COM\_API\_SetComType(int nType) |
| Function function | **Set the type of communication,** |
| Enter the parameters | Int nType --2:USB communication. Currently, only such communications are supported. |
| The output parameters | No. |
| Returns a value | Other: See Return Value Uniform Definition |
| Note. |  |

### 2.2.2 Com\_API\_OpenLink()

|  |  |
| --- | --- |
| The function prototype | int WINAPI Com\_API\_OpenLink(int nID,int nBound) |
| Function function | Establish a communication connection channel |
| Enter the parameters | Int nID -- fill it in, the system handles it automatically.  Int nBound - Baud rate fixed value 115200 |
| The output parameters |  |
| Returns a value | See Return Value Uniform Definition |
| Note. | Before you can start sending commands, you must call this interface to establish a hardware communication channel. |

### 2.2.3 COM\_API\_GetMachineStatus ()

|  |  |
| --- | --- |
| The function prototype | int WINAPI COM\_API\_GetMachineStatus(unsgined char\* nStatus) |
| Function function | Gets the machine status of the current controller. |
| Enter the parameters | Unsinged char\*nStatus-- the currentstate of the machine, converting this buffer into a machine state structure.  The machine state structure.  typedef struct \_MACHINE\_STATUS\_  {  uint32\_t dwWorkStatus; //D0: Pause;D1:Processing;D2:Dot;D3:Back toZero. D4: There is an alarm.  uint8\_t dwHomeDone; // Correct back to zero. D0:X-axis correctly back to zero;D1:Y axis correctly back to zero;D2:Z axis correctly back to zero;  uint8\_t nID; // Machine logic number. Reserved multi-card share identification markings.  uint16\_t FirmVer; // Firmware version.  int32\_t nPos[3]; The current position of the // axis, which is magnified 100,000 times to resolve floating-point transmission issues.  uint32\_t RealSpeed[3]; // The current speed of the axis. This value is magnified 100,000 times to resolve floating-point transmission issues.  uint32\_t nAlmCode; // Alarm code  uint16\_t dwInputStatus; // Input port status D0-IN1; D1-IN2; D3-IN3; D4-IN4; D5-ORG1; D6-ORG2; D7-ORG3;  uint16\_t dwOutputStatus; // Export status.  uint32\_t Rsv[4]; //  } MACHINE\_STATUS,\*PMACHINE\_STATUS; |
| The output parameters | No. |
| Returns a value | See Return Value Uniform Definition |
| Note. |  |

### 2.2.4 COM\_API\_ReadFileData ()

|  |  |
| --- | --- |
| The function prototype | int WINAPI COM\_API\_ReadFileData(int nSrc,int StartAdd,int len,unsigned char\* pOutput) |
| Function function | Read a portion of the data in the file, starting at the specified location. |
| Enter the parameters | int nSrc --- 1:parameterfile;2: processing file (not implemented);  The int StartAdd --- the file that was read.  Int len --- length of the read, up to a maximum of 500 bytes.  Unsigned char-pOutput ---- buffered address where data is stored. |
| The output parameters | No. |
| Returns a value | See Return Value Uniform Definition |
| Note. | This interface is used to implement reading a file in the motion controller to the PC side. When you read a file, you must call multiple times, pass different read start addresses, and then combine them to form a complete file. |

### 2.2.5 COM\_API\_WriteFileData ()

|  |  |
| --- | --- |
| The function prototype | int WINAPI COM\_API\_WriteFileData(int nSrc,int StartAdd,int len,unsigned char\* pInput) |
| Function function | Write a portion of the data to the specified location of the specified file. |
| Enter the parameters | int nSrc --- 1:parameterfile;2: processing file (not implemented);  The int StartAdd --- where you want to write to the file.  The int len --- length of up to 500 bytes.  The unsigned char\*pInput ---- the data to be written. |
| The output parameters | No. |
| Returns a value | See Return Value Uniform Definition |
| Note. | This interface is used to implement writing PC-side files to the controller. When you write to a file, you must call multiple times , pass different start addresses, and then combine them to form a complete file. |

### 2.2.6 COM\_API\_Jog()

|  |  |
| --- | --- |
| The function prototype | int WINAPI COM\_API\_Jog(int nAxis,float fDis,float Speed) |
| Function function | Pivot point. After this function is called, the motion controller will move the specified distance at the set speed. |
| Enter the parameters | int nAxis ---轴号。 0:X轴,1:Y轴。 2:Z轴。  Float fDis --- Sport distance  Float Speed -- speed of motion. |
| The output parameters | No. |
| Returns a value | See Return Value Uniform Definition |
| Note. |  |

### 2.2.7 COM\_API\_StopAll()

|  |  |
| --- | --- |
| The function prototype | int WINAPI COM\_API\_StopAll() |
| Function function | Stop all movement of the machine |
| Enter the parameters |  |
| The output parameters | No. |
| Returns a value | See Return Value Uniform Definition |
| Note. |  |

### 2.2.8 COM\_API\_StopAxis()

|  |  |
| --- | --- |
| The function prototype | int WINAPI COM\_API\_StopAxis(int nXAxisSet,int nYAxisSet,int nZAxisSet) |
| Function function | Stop an axis. |
| Enter the parameters | Int nXAxisSet --- 1:Stop X-axis,0:Donot stop X-axis  Int nYAxisSet --- 1:Stop Y-axis, 0:Do notstop Y-axis  Int nZAxisSet --- 1:Stop Z-axis,0:Donot stop Z-axis |
| The output parameters | No. |
| Returns a value | See Return Value Uniform Definition |
| Note. |  |

### 2.2.9 COM\_API\_Home()

|  |  |
| --- | --- |
| The function prototype | int WINAPI COM\_API\_Home(int nXAxisSet,int nYAxisSet,int nZAxisSet) |
| Function function | Axis back to zero |
| Enter the parameters | Int nXAxisSet --- 1:X-axis back to zero,0:X-axis does not return to zero  Int nYAxisSet --- 1:Y-axis back to zero,0:Y-axis does not return to zero  Int nZAxisSet --- 1:Z-axis back to zero,0:Z-axis does not return to zero |
| The output parameters | No. |
| Returns a value | See Return Value Uniform Definition |
| Note. |  |

### 2.2.10 COM\_API\_SetOutputBit ()

|  |  |
| --- | --- |
| The function prototype | int WINAPI COM\_API\_SetOutputBit(int OutputID,int nStatus) |
| Function function | Set the exit status |
| Enter the parameters | Int OutputID --- export serial numbers 1 to 4  int nStatus --- export status,1:low,0:high |
| The output parameters | No. |
| Returns a value | See Return Value Uniform Definition |
| Note. |  |

### 2.2.11 COM\_API\_GetLastError ()

|  |  |
| --- | --- |
| The function prototype | int WINAPI COM\_API\_GetLastError(unsigned int\* dwErr) |
| Function function | Get the error code that was generated last time. |
| Enter the parameters |  |
| The output parameters | Unsigned int s dwErr --- error code |
| Returns a value | See Return Value Uniform Definition |
| Note. |  |

### 2.2.12 COM\_API\_SendData ()

|  |  |
| --- | --- |
| The function prototype | int WINAPI COM\_API\_SendData(int nLen,unsigned char\* pData) |
| Function function | Send a frame of data or command to the controller |
| Enter the parameters | The length of the data --- the int nLen data.  Unsinged char\*pData --- data that needs to be sent. |
| The output parameters | No. |
| Returns a value | See Return Value Uniform Definition |
| Note. | This interface is a common data sending interface that developers can send command data formats to control through their own organizations. Command organization format needs to communicate with our technical support staff to verify. Developers are advised not to use this interface. |

### 2.2.13 COM\_API\_ReadData ()

|  |  |
| --- | --- |
| The function prototype | int WINAPI COM\_API\_ReadData(int nLen,unsgined char\* pInput,unsigned char\* pOutput) |
| Function function | A common send data interface function that bypasses the API and sends API interface instructions directly to the controller |
| Enter the parameters | Int nLen --- the length of the data sent  Unsgined char\*pInput --- send data buffers  unsigned char\*pOutput --receives data buffering |
| The output parameters | No. |
| Returns a value | See Return Value Uniform Definition |
| Note. | This interface is a common data sending interface that developers can send command data formats to control through their own organizations. Command organization format needs to communicate with our technical support staff to verify. Developers are advised not to use this interface. |

### 2.2.14 COM\_API\_DowloadSystemCfg()

|  |  |
| --- | --- |
| The function prototype | int WINAPI COM\_API\_DowloadSystemCfg(char\* iniPath) |
| Function function | Download the profile to the controller. |
| Enter the parameters | i char\*iniPath --- path to the PC side of the profile. |
| The output parameters | No. |
| Returns a value | See Return Value Uniform Definition |
| Note. |  |

# 3 System profile development guide

The system profile is the most important file for the motion controller to work properly, and it is also the link between the application and the controller parameter interaction, the application generates the system profile according to the needs of its own process, then downloads the system profile to the motion controller through the file download interface, the motion controller reads the parameters in the system configuration file, and then executes the new system configuration parameters. Because the system profile is maintained jointly by the application and the controller, the format and parameter names of the configuration file, as well as the position of the parameters in the file, cannot be modified at will, otherwise the controller will not be able to resolve some parameters correctly. The suffix name of the system profile must be"ini"and the file name must be made up of numbers and letters and should not be longer than 8 bytes. Configuration parameters in system configuration parameters may not make sense, or the application may not be able to use them, please do not modify them at will.

Description of the system profile format

[Head] // The starting end of the system file header

MachineTypeis themodel number of the 4030 // controller

Version s 1000 // profile version

[HeadEnd] 。 // The system file header ends

[MachineParam] 。 // The machine parameter segment begins

fTimerPeriods 1.000000 // Machine interpolation cycle

fWorkPrecision

fArcCheckPrecisions.010000 // Arc detection accuracy to detect whether 3 points can be made into a circle.

The minimum length of the fMinLen.200000 // segment, if the segment is less than this value, is merged

fMaxFeedSpeed s48000.000000 // Maximum feed speed(mm/min) of the system

nAccelTypes1 // System acceleration type is Type S and must be 1. (not used).

fMaxAccelSpeed s 2000.00000 // The bend acceleration of the system(mm/S2).

fAccelSpeed s 3000.00000 // Process acceleration of the system(mm/S2).

fJAccelSpeed s 200000.00000 // The acceleration of the system(mm/S3).

fFastAccelSpeed s 4000.00000// The system's air-range acceleration(mm/S2).

ControlFlags1 // Control word. A D0 of 1 indicates that the machine automatically returns to the stop after stopping.

wHomePowerOns0 // Power back to origin. 1 is the power-on automatic back to the origin.

[XAxisParam] 。 The //X axis parameter segment begins

nPulseFactorUps 10000.000000 // Number of pulses required for a turn of the drive

nPulseFactorDown . .. 31.550000 // Drive turn around, the distance from which the machine is moving.

nPulseLogics 1 // efficient logic for driver pulses (high or low).

fMaxSpeed s24000.00000 // the maximum moving speed(mm/min) of theaxis

nHomeDir=1 // Zero direction

fMaxPoss 1300.000000 // axis maximum travel

nEnableBacklashs0 // axis reverse gap compensation

fBacklashLen.000000 // Reverse gap compensation distance

fBacklashSpeed s600.000000 // Reverse gap compensation speed

fHomeSpeed s3600.000000 // axis of zero-back speed

fHomeCheckDiss 10.000000 // Meaningless

fHomeZeroSpeeds 600.000000 // meaningless

fHomeOrgSpeeds 300.000000 // Meaningless

fHomePosOffsets 10.000000 // The axis deviates from the position of the origin switch after returning to the origin.

[YAxisParam]

nPulseFactorUps 10000.000000 // Number of pulses required for a turn of the drive

nPulseFactorDown . .. 31.550000 // Drive turn around, the distance from which the machine is moving.

The valid logic of the nPulseLogic.1 // driver pulse (high or low) is valid

fMaxSpeed s24000.00000 // the maximum moving speed(mm/min) of theaxis

nHomeDir=1 // Zero direction

fMaxPoss 1300.000000 // axis maximum travel

nEnableBacklashs0 // axis reverse gap compensation

fBacklashLen.000000 // Reverse gap compensation distance

fBacklashSpeed s600.000000 // Reverse gap compensation speed

fHomeSpeed s3600.000000 // axis of zero-back speed

fHomeCheckDiss 10.000000 // Meaningless

fHomeZeroSpeeds 600.000000 // meaningless

fHomeOrgSpeeds 300.000000 // Meaningless

fHomePosOffsets 10.000000 // The axis deviates from the position of the origin switch after returning to the origin.

[ZAxisParam]

nPulseFactorUps 10000.000000 // Number of pulses required for a turn of the drive

nPulseFactorDown . .. 31.550000 // Drive turn around, the distance from which the machine is moving.

The valid logic of the nPulseLogic.1 // driver pulse (high or low) is valid

fMaxSpeed s24000.00000 // the maximum moving speed(mm/min) of theaxis

nHomeDir=1 // Zero direction

fMaxPoss 1300.000000 // axis maximum travel

nEnableBacklashs0 // axis reverse gap compensation

fBacklashLen.000000 // Reverse gap compensation distance

fBacklashSpeed s600.000000 // Reverse gap compensation speed

fHomeSpeed s3600.000000 // axis of zero-back speed

fHomeCheckDiss 10.000000 // Meaningless

fHomeZeroSpeeds 600.000000 // meaningless

fHomeOrgSpeeds 300.000000 // Meaningless

fHomePosOffsets 10.000000 // The axis deviates from the position of the origin switch after returning to the origin.