

**AMC4030Motion Controller**

**编**

**程**

**手**

**册**

**（1.0版）**

**成都福誉科技有限公司**

**FUYU TECHNOLOGY CO.,LTD.**

**版权申明**

**成都福誉科技有限公司**

**保留所有权利**

**成都福誉科技有限公司保留在不事先通知的情况下，修改本手册中的成品和产品规格等文件的权利。**

**成都福誉科技有限公司不承担由于使用本手册或本产品不当，所造成直接的、间接的、附带的或相应产品的损失或责任**

**成都福誉科技有限公司具有本产品及其软件的专利权、版权和其它知识产权。未经授权，不得直接或间接的复制、制造、加工、使用本产品及其相关部分。**

|  |  |
| --- | --- |
| **C:\Users\Administrator\AppData\Local\Temp\0d0a6f46076592e0ff9ca24f81582ff1.gif**  **注意** | **运动中的机器有危险！使用者有责任在机器中设计有效的出错处理和安全保护机制，成都福誉科技有限公司没有义务或责任对此造成的附带的或相应产生的损失责任** |

**联系我们**

**成都福誉科技有限公司**

**地址：成都市双流县西航港大道中4段1455号**

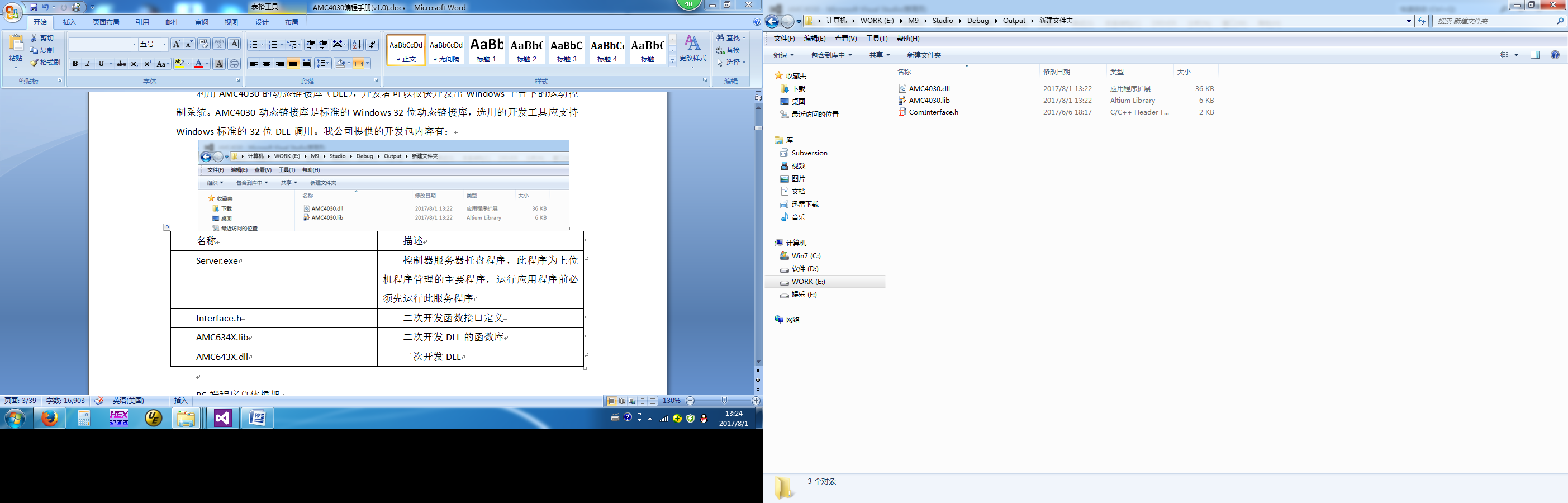
**电话：028-65058998**

**传真：028-65058997**

**网址：www.fuyuautomation.com**

# 1 Overview

With the AMC4030's dynamic link library (DLL), developers can quickly develop motion control systems for the Windows platform. The AMC4030 dynamic link library is a standard Windows 32-bit dynamic link library. The selected development tools should support Windows standard 32-bit DLL calls. The development kit provided by our company has：



|  |  |
| --- | --- |
| Name | Description |
| ComInterface.h | Secondary development function interface definition |
| AMC4030.lib | Secondary development DLL function library |
| AMC4030.dll | Secondary development DLL |

PC-wide program overall framework

The application is redeveloped through the function interface provided by AMC4030.dll to realize its own motion control system. The AMC4030 is the bridge between the application and the AMC4030 motion controller. Data interaction and command transfer with the AMC4030 motion controller can be achieved through the DLL.

AMC4030

Motion Controller

AMC4030.dll

Application

Default convention：

* Unless otherwise specified (see function description), all functions must be called in one thread。
* The return value of all function interfaces has a uniform meaning (very important)

|  |  |
| --- | --- |
| Return value | Meaning |
| 1 | After the command is executed, the controller has received it correctly and returned the correct value to the server.。 |
| -1 | The command execution failed, the command was judged to be illegal, and the controller refused to execute the command (there may be conditions for executing the command) |
| -2 | Command parameter check failed, command execution failed |
| -4 | DLL and controller communication timeout。 |
| -6 | The command execution conflict indicates that the previous command has not been fully executed and a new command has arrived. (Expressed as: an interface call has not been returned, and then start calling the new interface. Multi-threaded is more common)。 |

## 1.1Develop Visual C++ Control Program

（一）Description

Users can use VC6.0 or higher to develop motion control systems for Windows platforms. Visual C++ has two methods for calling dynamic link libraries, and the files used are slightly different.。

（二）Dynamic link library function calling method

（1）Implicit call

The implicit call steps are as follows：

（a）Start Visual C++ and create a new project；

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

（c）Select the "Settings" menu item under the "Project" menu；

（d）Switch to the "Link" tab and enter the "AMC4030.lib" filename in the "Object/library modules" field.；

（e）Add the declaration file "ComInterface.h" of the function library header file to the application.；

（f）Call the interface function at the appropriate place in the application。

（2）Explicit call

Explicitly calling methods need to call the Windows API functions to load and release the dynamic link library. Methods as below：

（a）Call 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）Call the function in the DLL with the 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 used。

This method is cumbersome. We have packaged the commonly used DLL functions into the class CAMC4030DLL and provided the source code for the class. This class contains the same member functions as the AMC4030DLL.h function name and parameters. The source code can be requested from Fitch Technology Support, or you can write it yourself. The source files are named AMC4030DLL.cpp and AMC4030DLL.h. The developer can add it to the project, add the object of the class in the appropriate place of the program, and call the function in the DLL through the corresponding member function.

The calling steps are as follows：

（a）Start Visual C++ and create a new project；

（b）Copy "Interface.h" and function declaration files AMC4030DLL.cpp and AMC4030DLL.h into the project file；

（c）Select the "Files" submenu of "Add To Project" under the "Project" menu.；

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

（e）Generate an object of CAMC4030DLL in the application, call the motion function。

The above two methods are standard methods for calling dynamic link library functions in VC. For more specific calling methods and help, please refer to the corresponding part of Microsoft Visual Stutio development document MSDN or related VC reference books.。

# 2Interface Development Guide

## 2.1System Initialization

### 2.1.1Create link

int nRtn;

int nType;

COM\_API\_SetComType(2);//Parameter 2：Indicates that a USB communication link is established and can only pass 2.

nRtn = COM\_API\_OpenLink(3,115200); //Establish connection。Return 1，Indicates that the link was created successfully。

## 2.2 Interface function details

### 2.2.1 COM\_API\_SetComType ()

|  |  |
| --- | --- |
| Prototype | int WINAPI COM\_API\_SetComType(int nType) |
| Function | **Set the communication type，** |
| Input | Int nType --2:USBcommunication。Currently only supports such communications。 |
| Output | No |
| Return | See return value uniform definition |
| Remarks |  |

### 2.2.2 Com\_API\_OpenLink()

|  |  |
| --- | --- |
| Prototype | int WINAPI Com\_API\_OpenLink(int nID,int nBound) |
| Function | **Establish communication connection channel** |
| Input | Int nID -- Just fill it out，Automatic system processing。  Int nBound -- Fixed baud rate 115200 |
| Output |  |
| Return | See return value uniform definition |
| Remarks | Before starting to send commands，This interface must be called，To establish a hardware communication channel. |

### 2.2.3 COM\_API\_GetMachineStatus ()

|  |  |
| --- | --- |
| Prototype | int WINAPI COM\_API\_GetMachineStatus(unsgined char\* nStatus) |
| Function | **Get the machine state of the current controller。** |
| Input | Unsinged char\* nStatus--当前机器的状态,将此缓冲转化为机器状态结构体。  //机器状态结构体。  typedef struct \_MACHINE\_STATUS\_  {  uint32\_t dwWorkStatus; //D0:暂停中；D1：加工；D2：点动中；D3：回零中。D4:有报警。  uint8\_t dwHomeDone; //是否正确回零。D0:X轴正确回过零；D1:Y轴正确回过零；D2:Z轴正确回过零；  uint8\_t nID; //机器逻辑编号。预留多卡共用识别标示。  uint16\_t FirmVer; //固件版本。  int32\_t nPos[3]; //轴的当前位置，此值放大了100000倍，以解决浮点传输问题。  uint32\_t RealSpeed[3]; //轴的当前速度。此值放大100000倍，以解决浮点传输问题。  uint32\_t nAlmCode; //报警编码  uint16\_t dwInputStatus; //输入口状态D0-IN1; D1-IN2; D3-IN3; D4-IN4; D5-ORG1; D6-ORG2; D7-ORG3;  uint16\_t dwOutputStatus; //输出口状态。  uint32\_t Rsv[4]; //  } MACHINE\_STATUS,\*PMACHINE\_STATUS; |
| Output | 无 |
| Return | 参见返回值统一定义 |
| Remarks |  |

### 2.2.4 COM\_API\_ReadFileData ()

|  |  |
| --- | --- |
| Prototype | int WINAPI COM\_API\_ReadFileData(int nSrc,int StartAdd,int len,unsigned char\* pOutput) |
| Function | **Read some data in the file，Start reading from the specified location。** |
| Input | int nSrc --- 1：参数文件；2：加工文件（未实现）；  int StartAdd ---读取的文件起始位置。  int len ---读取的长度，最长不超过500字节。  unsigned char\* pOutput ----存放数据的缓冲地址。 |
| Output | 无 |
| Return | 参见返回值统一定义 |
| Remarks | 此接口用来实现读取运动控制器中的某个文件到PC端。读取一个文件时，必须多次调用，传递不同的读取起始地址，然后把他们组合起来，形成一个完整的文件。 |

### 2.2.5 COM\_API\_WriteFileData ()

|  |  |
| --- | --- |
| Prototype | int WINAPI COM\_API\_WriteFileData(int nSrc,int StartAdd,int len,unsigned char\* pInput) |
| Function | **Part of the data，Write to the specified location of the specified file。** |
| Input | int nSrc --- 1：参数文件；2：加工文件（未实现）；  int StartAdd ---要写入文件的起始位置。  int len ---长度，最长不超过500字节。  unsigned char\* pInput ----要写入的数据。 |
| Output | 无 |
| Return | 参见返回值统一定义 |
| Remarks | 此接口用来实现将PC端文件写入到控制器中。写入一个文件时，必须多次调用，传递不同的起始地址，然后把他们组合起来，形成一个完整的文件。 |

### 2.2.6 COM\_API\_Jog()

|  |  |
| --- | --- |
| Prototype | int WINAPI COM\_API\_Jog(int nAxis,float fDis,float Speed) |
| Function | **Axis jog。After calling this function，The motion controller will follow the set speed，Movement specified distance。** |
| Input | int nAxis --- Axis number。0：X axis，1：Y axis。2：Z axis。  Float fDis --- Moving distance  Float Speed -- Movement speed。 |
| Output | No |
| Return | See return value uniform definition |
| Remarks |  |

### 2.2.7 COM\_API\_StopAll()

|  |  |
| --- | --- |
| Prototype | int WINAPI COM\_API\_StopAll() |
| Function | 停止机器所有运动 |
| Input |  |
| Output | 无 |
| Return | 参见返回值统一定义 |
| Remarks |  |

### 2.2.8 COM\_API\_StopAxis()

|  |  |
| --- | --- |
| Prototype | int WINAPI COM\_API\_StopAxis(int nXAxisSet,int nYAxisSet,int nZAxisSet) |
| Function | Stop an axis。 |
| Input | int nXAxisSet --- 1：Stop X axis，0：Non-stop X axis  int nYAxisSet --- 1：Stop Y axis，0：Non-stop Y axis  int nZAxisSet --- 1：Stop Z axis，0：Non-stop Z axis |
| Output | No |
| Return | See return value uniform definition |
| Remarks |  |

### 2.2.9 COM\_API\_Home()

|  |  |
| --- | --- |
| Prototype | int WINAPI COM\_API\_Home(int nXAxisSet,int nYAxisSet,int nZAxisSet) |
| Function | Axis return zero |
| Input | int nXAxisSet --- 1：X axis return zero，0：X axis does not return zero  int nYAxisSet --- 1：Y axis return zero，0：Y axis does not return zero  int nZAxisSet --- 1：Z axis return zero，0：Z axis does not return zero |
| Output | No |
| Return | See return value uniform definition |
| Remarks |  |

### 2.2.10 COM\_API\_SetOutputBit ()

|  |  |
| --- | --- |
| Prototype | int WINAPI COM\_API\_SetOutputBit(int OutputID,int nStatus) |
| Function | 设置输出口状态 |
| Input | int OutputID --- 输出口序号1~4  int nStatus --- 输出口状态，1：低电平，0：高电平 |
| Output | 无 |
| Return | 参见返回值统一定义 |
| Remarks |  |

### 2.2.11 COM\_API\_GetLastError ()

|  |  |
| --- | --- |
| Prototype | int WINAPI COM\_API\_GetLastError(unsigned int\* dwErr) |
| Function | 获取上一次产生的错误代码。 |
| Input |  |
| Output | unsigned int \* dwErr ---错误码 |
| Return | 参见返回值统一定义 |
| Remarks |  |

### 2.2.12 COM\_API\_SendData ()

|  |  |
| --- | --- |
| Prototype | int WINAPI COM\_API\_SendData(int nLen,unsigned char\* pData) |
| Function | **Send a frame of data or commands to the controller** |
| Input | int nLen --- Length of data sent。  Unsinged char\* pData --- Data to send。 |
| Output | No |
| Return | See return value uniform definition |
| Remarks | This interface is a universal data sending interface. Developers can send command data formats to their own organization and send them to the control. The command organization format needs to be verified and verified by our technical support staff. Developers are advised not to use this interface. |

### 2.2.13 COM\_API\_ReadData ()

|  |  |
| --- | --- |
| Prototype | int WINAPI COM\_API\_ReadData(int nLen,unsgined char\* pInput,unsigned char\* pOutput) |
| Function | 通用发送数据接口函数，可绕过API直接发送API接口指令到控制器 |
| Input | Int nLen --- 发送数据的长度  unsgined char\* pInput ---发送数据缓冲  unsigned char\* pOutput ---接收数据缓冲 |
| Output | 无 |
| Return | 参见返回值统一定义 |
| Remarks | 此接口为通用数据发送接口，开发者可通过自己组织发送命令数据格式，发送到控制。命令组织格式需要和我司技术支持人员沟通核实。建议开发人员不要使用此接口。 |

### 2.2.14 COM\_API\_DowloadSystemCfg()

|  |  |
| --- | --- |
| Prototype | int WINAPI COM\_API\_DowloadSystemCfg(char\* iniPath) |
| Function | 下载配置文件到控制器。 |
| Input | i char\* iniPath --- 配置文件的PC端存放路径。 |
| Output | 无 |
| Return | 参见返回值统一定义 |
| Remarks |  |

# 3 System Profile Development Guide

The system configuration file is the most important file for the motion controller to work properly. It is also the link between the application and the controller parameters. The application generates the system configuration file according to the needs of its own process, and then downloads the system configuration file through the file download interface. In the motion controller, the motion controller reads the parameters in the system configuration file and then executes the new system configuration parameters. Since the system configuration file is jointly maintained by the application and the controller, the format and parameter names of the configuration file and the location of the parameters in the file cannot be modified at will, otherwise the controller will not be able to correctly parse out some parameters. The system configuration file must have the suffix name "ini", the file name consists of numbers + letters, and the length should not exceed 8 bytes. Some of the configuration parameters in the system configuration parameters may be meaningless, or the application may not use them. Please do not modify them at will.  
System configuration file format description

[Head] // System file header start

MachineType=4030 // Controller model

Version=1000 // Profile version

[HeadEnd] // System file header ends

[MachineParam] //机器参数段开始

fTimerPeriod=1.000000 //机器插补周期

fWorkPrecision=0.005000 //圆弧的拆分精度

fArcCheckPrecision=0.010000 //圆弧检测精度，用来检测3点是否能够成一个圆。

fMinLen=0.200000 //线段的最小长度，如果线段小于此值，则将被合并

fMaxFeedSpeed=48000.000000 //系统的最大进给速度（mm/min)

nAccelType=1 //系统加速度类型为S型，必须为1。(未使用）

fMaxAccelSpeed=2000.000000 //系统的拐弯加速度（mm/S2)

fAccelSpeed=3000.000000 //系统的加工加速度（mm/S2)

fJAccelSpeed=200000.000000 //系统的加加速度（mm/S3)

fFastAccelSpeed=4000.000000 //系统的空程加速度（mm/S2)

ControlFlag=1 //控制字。D0为1时，表示停止后机器自动回停靠点。

wHomePowerOn=0 //开机回原点。1为开机自动回原点。

[XAxisParam] //X轴参数段开始

nPulseFactorUp=10000.000000 //驱动器转一圈需要的脉冲数

nPulseFactorDown=31.550000 //驱动器转一圈，机器运动的距离数。

nPulseLogic=1 //驱动器脉冲的有效逻辑（高电平有效还是低电平有效）

fMaxSpeed=24000.000000 //轴的最大运动速度（mm/min）

nHomeDir=1 //回零方向

fMaxPos=1300.000000 //轴的最大行程

nEnableBacklash=0 //轴的反向间隙补偿

fBacklashLen=0.000000 //反向间隙补偿距离

fBacklashSpeed=600.000000 //反向间隙补偿的速度

fHomeSpeed=3600.000000 //轴的回零速度

fHomeCheckDis=10.000000 //无意义

fHomeZeroSpeed=600.000000 //无意义

fHomeOrgSpeed=300.000000 //无意义

fHomePosOffset=10.000000 //轴回原点后偏离原点开关的位置。

[YAxisParam]

nPulseFactorUp=10000.000000 //驱动器转一圈需要的脉冲数

nPulseFactorDown=31.550000 //驱动器转一圈，机器运动的距离数。

nPulseLogic=1 //驱动器脉冲的有效逻辑（高电平有效还是低电平有效）

fMaxSpeed=24000.000000 //轴的最大运动速度（mm/min）

nHomeDir=1 //回零方向

fMaxPos=1300.000000 //轴的最大行程

nEnableBacklash=0 //轴的反向间隙补偿

fBacklashLen=0.000000 //反向间隙补偿距离

fBacklashSpeed=600.000000 //反向间隙补偿的速度

fHomeSpeed=3600.000000 //轴的回零速度

fHomeCheckDis=10.000000 //无意义

fHomeZeroSpeed=600.000000 //无意义

fHomeOrgSpeed=300.000000 //无意义

fHomePosOffset=10.000000 //轴回原点后偏离原点开关的位置。

[ZAxisParam]

nPulseFactorUp=10000.000000 //驱动器转一圈需要的脉冲数

nPulseFactorDown=31.550000 //驱动器转一圈，机器运动的距离数。

nPulseLogic=1 //驱动器脉冲的有效逻辑（高电平有效还是低电平有效）

fMaxSpeed=24000.000000 //轴的最大运动速度（mm/min）

nHomeDir=1 //回零方向

fMaxPos=1300.000000 //轴的最大行程

nEnableBacklash=0 //轴的反向间隙补偿

fBacklashLen=0.000000 //反向间隙补偿距离

fBacklashSpeed=600.000000 //反向间隙补偿的速度

fHomeSpeed=3600.000000 //轴的回零速度

fHomeCheckDis=10.000000 //无意义

fHomeZeroSpeed=600.000000 //无意义

fHomeOrgSpeed=300.000000 //无意义

fHomePosOffset=10.000000 //轴回原点后偏离原点开关的位置。