AMC4030 运动控制器

编

程

手

册

（2.0 版）

成都福誉科技有限公司

FUYU TECHNOLOGY CO.,LTD.

版 权 申 明

 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

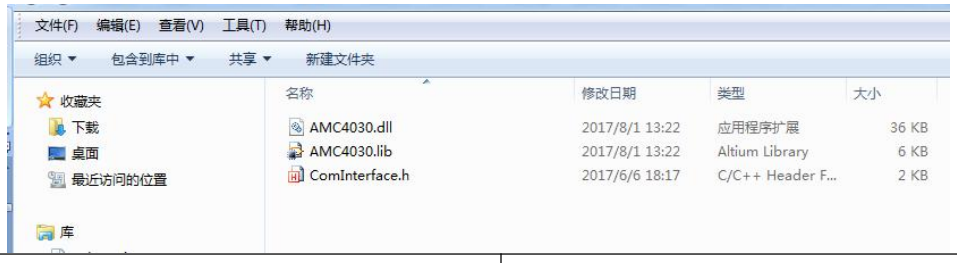
电话：028-65058998

传真：028-65058997

网址：www.fuyuautomation.com

1 概述

利用 AMC4030 的动态链接库（DLL），开发者可以很快开发出 Windows 平台下的运动控制系统。AMC4030 动态链接库是标准的 Windows 32 位动态链接库，选用的开发工具应支持 Windows 标准的 32 位 DLL 调用。我公司提供的开发包内容有：



名称	描述
ComInterface.h	二次开发函数接口定义
AMC4030.lib	二次开发 DLL 的函数库
AMC4030.dll	二次开发 DLL

PC 端程序总体框架

应用程序通过 AMC4030.dll 提供的函数接口进行二次开发，实现自己的运动控制系统。AMC4030 是应用程序与 AMC4030 运动控制器的桥梁，通过 DLL 可以实现和 AMC4030 运动控制器的数据交互和命令传递。

默认约定：

- 除非有特殊说明（详见函数描述），所有函数必须在一个线程中调用。
- 所有函数接口的返回值都有统一的含义（非常重要）

返回值	含义
1	命令执行完毕，控制器已经正确接收，并返回了正确值给服务器端。
-1	命令执行失败，命令被控制判断为非法，控制器拒绝执行此命令（有可能执行此命令的条件不具备）
-2	命令参数检查没通过，命令执行失败
-4	DLL 与控制器通讯超时。
-6	命令执行冲突，表示上一个命令还未完全执行，又有新命令来了。（表现为：一个接口调用还未返回，又开始调用新的接口了。多线程中比较普遍）。

1.1 开发 Visual C++控制程序

（一）说明

用户可以使用 VC6.0 或更高版本，来进行 Windows 平台下运动控制系统开发。Visual C++有两种调用动态链接库的方法，使用的文件略有不同。

（二）动态链接库函数调用方法

（1）隐式调用

隐式调用步骤如下：

（a）启动 Visual C++，新建一个工程；

（b）将开发包里的动态链接库“AMC4030.dll”、“AMC4030.lib”和函数声明文件“ComInterface.h”复制到工程文件中；

（c）选择“Project”菜单下的“Settings”菜单项；

（d）切换到“Link”标签页，在“Object/library modules”栏中输入“AMC4030.lib”文件名；

（e）在应用程序中加入函数库头文件的声明文件“ComInterface.h”；

（f）在应用程序的合适位置调用接口函数。

（2）显式调用

显式调用方法需要调用 Windows API 函数加载和释放动态链接库。方法如下：

（a）调用 Windows API 函数 LoadLibrary()动态加载 DLL；

（b）调用 Windows API 函数 GetProcAddress()取得将要调用的 DLL 中函数的指针；

（c）用函数指针调用 DLL 中函数完成相应功能；

（e）在程序结束时或不再使用 DLL 中函数时，调用 Windows API 函数 FreeLibrary()释放动态链接库。

该方法比较繁琐。我们已经将常用的 DLL 函数封装成类 CAMC4030DLL，并提供该类的源代码。该类含有与 AMC4030DLL.h 函数名及参数相同的成员函数。源代码可向福誉技术支持索要，或自己编写，源码的文件名为 AMC4030DLL.cpp 和 AMC4030DLL.h。开发人员可将其添加进工程，在程序适当地方添加该类的对象，通过对应成员函数来调用 DLL 中的函数。

调用步骤如下：

（a）启动 Visual C++，新建一个工程；

（b）将“Interface.h”和函数声明文件 AMC4030DLL.cpp 和 AMC4030DLL.h 复制到工程文件中；

（c）选择“Project”菜单下的“Add To Project”的子菜单“Files”项；

（d）将 AMC4030DLL.cpp 和 AMC4030DLL.h 加入到工程中；

（e）在应用程序中生成 CAMC4030DLL 的一个对象，调用运动函数。

以上在两种方法均为 VC 中调用动态链接库函数的标准方法，若要获得更具体的调用方法和帮助，请参考微软 Visual Stutio 开发文档 MSDN 或相关 VC 参考书籍中相应部分内容。

2 接口开发指南

2.1 系统初始化

2.1.1 创建链接

```
int nRtn;
int nType;
COM_API_SetComType(2);//参数 2：表示建立 USB 通讯链接，只能传递 2.
nRtn = COM_API_OpenLink(3,115200);  //建立连接。返回 1，表示创建链接成功。
```

2.2 接口函数详细说明

2.2.1 COM_API_SetComType ()

函数原型	**int WINAPI COM_API_SetComType(int nType)**
函数功能	设置通讯类型
输入参数	Int nType　　--2:USB 通讯。目前只支持此类通讯。
输出参数	无
返回值	其它：参见返回值统一定义
备注	

2.2.2 Com_API_OpenLink()

函数原型	**int WINAPI Com_API_OpenLink(int nID,int nBound)**
函数功能	建立通讯连接通道
输入参数	Int nID　　--随便填，系统自动处理。 Int nBound　　--波特率固定值 115200
输出参数	
返回值	参见返回值统一定义
备注	在开始发送命令之前，必须调用此接口，以建立硬件通讯通道。

2.2.3 COM_API_GetMachineStatus ()

函数原型	**int WINAPI COM_API_GetMachineStatus(unsgined char* nStatus)**
函数功能	获取当前控制器的机器状态

输入参数	Unsinged char* nStatus --当前机器的状态,将此缓冲转化为机器状态结构体。
	
	typedef struct _IO_STUTAS_
	{
	char NodeId[128];
	unsigned short int NodeIOStatus[128];
	}IO_STATUS, *PIO_STATUS;
	
	//机器状态结构体。
	typedef struct _MACHINE_STATUS_
	{
	IO_STATUS IO_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]; //Rsv[0]是生产序列号
	} MACHINE_STATUS,*PMACHINE_STATUS;

输出参数	无
返回值	参见返回值统一定义
备注	

2.2.4 COM_API_Jog()

函数原型	**int WINAPI COM_API_Jog(int nAxis,float fDis,float Speed)**
函数功能	轴点动（调用此函数后，运动控制器将按照设置的速度，运动指定的距离）
输入参数	int nAxis ---轴号。0：X 轴，1：Y 轴。2：Z 轴。 Float fDis ---运动距离

	Float Speed --运动速度。
输出参数	无
返回值	参见返回值统一定义
备注	

2.2.5 COM_API_Home()

函数原型	**int WINAPI COM_API_Home(int nXAxisSet,int nYAxisSet,int nZAxisSet)**
函数功能	轴回零
输入参数	int nXAxisSet --- 1：X 轴回零，0：X 轴不回零 int nYAxisSet --- 1：Y 轴回零，0：Y 轴不回零 int nZAxisSet --- 1：Z 轴回零，0：Z 轴不回零
输出参数	无
返回值	参见返回值统一定义
备注	

2.2.6 COM_API_StopAll()

函数原型	**int WINAPI COM_API_StopAll()**
函数功能	停止机器的所有运动
输入参数	无
输出参数	无
返回值	参见返回值统一定义
备注	

2.2.7 COM_API_StopAxis()

函数原型	**int WINAPI COM_API_StopAxis(int nXAxisSet,int nYAxisSet,int nZAxisSet)**
函数功能	停止某个轴
输入参数	int nXAxisSet --- 1：停止 X 轴，0：不停止 X 轴 int nYAxisSet --- 1：停止 Y 轴，0：不停止 Y 轴 int nZAxisSet --- 1：停止 Z 轴，0：不停止 Z 轴
输出参数	无
返回值	参见返回值统一定义
备注	

2.2.8 COM_API_SetOutputBit ()

函数原型	**int WINAPI COM_API_SetOutputBit(int OutputID,int nStatus)**
函数功能	设置输出口状态
输入参数	int OutputID --- 输出口序号 1~4

	int nStatus --- 输出口状态，1：低电平，0：高电平
输出参数	无
返回值	参见返回值统一定义
备注	

2.2.9 COM_API_GetLastError ()

函数原型	**int WINAPI COM_API_GetLastError(unsigned int* dwErr)**
函数功能	获取上一次产生的错误代码
输入参数	无
输出参数	unsigned int * dwErr ---错误码
返回值	参见返回值统一定义
备注	

2.2.10 COM_API_DowloadSystemCfg()

函数原型	**int WINAPI COM_API_DowloadSystemCfg(char* iniPath)**
函数功能	下载配置文件到控制器
输入参数	char* iniPath --- 配置文件的 PC 端存放路径
输出参数	无
返回值	参见返回值统一定义
备注	

2.2.11 COM_API_UploadSystemCfg()

函数原型	**int WINAPI COM_API_UploadSystemCfg(char* iniPath)**
函数功能	将控制器内的配置信息生成配置文件至 PC 上
输入参数	char* iniPath --- 配置文件的 PC 端存放路径
输出参数	无
返回值	参见返回值统一定义
备注	

2.2.12 COM_API_FastLine3 ()

函数原型	**int WINAPI COM_API_FastLine3 (float Speed,float Acc,float XDis,float YDis,float ZDis)**
函数功能	使最多三个轴进行联立运动（所有轴同时启动同时停止）
输入参数	float Speed --- 所有运动轴的合成速度 float Acc --- 所有运动轴的合成加速度 float XDis --- X 轴的运动距离 float YDis --- Y 轴的运动距离

	float ZDis --- Z 轴的运动距离
输出参数	无
返回值	参见返回值统一定义
备注	

2.2.13 COM_API_IndependentJog ()

函数原型	**int WINAPI COM_API_IndependentJog(float XSpeed,float XAcc,float XDis,float YSpeed,float YAcc,float YDis,float ZSpeed,float ZAcc,float ZDis)**
函数功能	使最多三个轴进行独立运动（所有轴同时启动不同时停止）
输入参数	float XSpeed --- X 轴的运动速度 float XAcc --- X 轴的运动加速度 float XDis --- X 轴的运动距离 float YSpeed --- Y 轴的运动速度 float YAcc --- Y 轴的运动加速度 float YDis --- Y 轴的运动距离 float ZSpeed --- Z 轴的运动速度 float ZAcc --- Z 轴的运动加速度 float ZDis --- Z 轴的运动距离
输出参数	无
返回值	参见返回值统一定义
备注	

2.2.14 COM_API_PauseAll ()

函数原型	**int WINAPI COM_API_PauseAll ()**
函数功能	暂停机器的所有运动
输入参数	无
输出参数	无
返回值	参见返回值统一定义
备注	

2.2.15 COM_API_ResumeAll ()

函数原型	**int WINAPI COM_API_ResumeAll()**
函数功能	恢复机器在暂停前的所有运动
输入参数	无
输出参数	无
返回值	参见返回值统一定义
备注	

2.2.16 COM_API_SendData ()

函数原型	**int WINAPI COM_API_SendData(int nLen,unsigned char* pData)**
函数功能	发送一帧数据或命令到控制器
输入参数	int nLen ---发送数据的长度。 Unsinged char* pData ---需要发送的数据。
输出参数	无
返回值	参见返回值统一定义
备注	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。

2.2.17 COM_API_ReadData ()

函数原型	**int WINAPI COM_API_ReadData(int nLen,unsgined char* pInput,unsigned char* pOutput)**
函数功能	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
输入参数	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
输出参数	无
返回值	参见返回值统一定义
备注	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。

3 系统配置文件开发指南

系统配置文件是运动控制器能正常工作的最重要的文件，也是应用程序和控制器参数交互的纽带，应用程序根据自己工艺的需要，生成系统配置文件，然后通过文件下载接口将系统配置文件下载到运动控制器中，运动控制器读取系统配置文件中的参数，然后执行新的系统配置参数。由于系统配置文件由应用程序和控制器共同维护，所以配置文件的格式和参数名称以及参数在文件中的位置不可随意修改，否则控制器将无法正确解析出某些参数。系统配置文件的后缀名必须“ini”，文件名称由数字+字母组成，且长度不要超过 8 字节。在系统配置参数中有的配置参数可能无意义，或则应用程序使用不到，请不要随意修改它们。

系统配置文件格式说明

```
[Head]                          //系统文件头起始端
MachineType=4030                //控制器的型号
```

```
Version=1000                    //配置文件版本
[HeadEnd]                       //系统文件头结束
[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。
```