固高科技
GOOGOL TECHNOLOGY

扩展功能

GHN 系列运动控制器编程手册

DLM 功能

2016 年 7 月

版权申明

固高科技有限公司（以下简称固高科技）保留在不事先通知的情况下，修改本手册中的产品和产品规格等文件的权力。

固高科技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、特殊的、附带的或相应产生的损失或责任。

固高科技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，不得直接或者间接地复制、制造、加工、使用本产品及其相关部分。

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，固高科技没有义务或责任对由此造成的附带的或相应产生的损失负责。

联系我们

固高科技（深圳）有限公司

地　　址：深圳市高新技术产业园南区深港产学研基地西座二楼 W211 室

电　　话：0755-26970817　26737236　26970824

传　　真：0755-26970821

电子邮件：support@googoltech.com

网　　址：http://www.googoltech.com.cn

固高科技（香港）有限公司

地　　址：香港九龍觀塘偉業街 108 號絲寶國際大廈 10 樓 1008-09 室

電　　話：+(852) 2358-1033

傳　　真：+(852) 2719-8399

電子郵件：info@googoltech.com

網　　址：http://www.googoltech.com

臺灣固高科技股份有限公司

地　　址：台中市西屯區工業區三十二路 86 號 3 樓

電　　話：+886-4-23588245

傳　　真：+886-4-23586495

電子郵件：twinfo@googoltech.com

文档版本

版本号	修订日期
V1.0	2017 年 4 月 17 日
V1.1	2019 年 7 月 30 日

前言

感谢选用固高运动控制器

为回报客户，我们将以品质一流的运动控制器、完善的售后服务、高效的技术支持，帮助您建立自己的控制系统。

固高产品的更多信息

固高科技的网址是 http://www.googoltech.com.cn。在我们的网页上可以得到更多关于公司和产品的信息，包括：公司简介、产品介绍、技术支持、产品最新发布等等。

您也可以通过电话（0755－26970817）咨询关于公司和产品的更多信息。

技术支持和售后服务

您可以通过以下途径获得我们的技术支持和售后服务：

电子邮件：support@googoltech.com；
电话：0755－26970843
发函至：深圳市高新技术产业园南区园深港产学研基地西座二楼 W211 室
固高科技（深圳）有限公司
邮编： 518057

编程手册的用途

用户通过阅读本手册，能够了解运动控制器的功能，掌握函数的用法，熟悉编程实现。最终，用户可以根据自己特定的控制系统，编制用户应用程序，实现控制要求。

编程手册的使用对象

本编程手册适用于具有C语言编程基础或Windows环境下使用动态链接库的基础，同时具有一定运动控制工作经验，对伺服或步进控制的基本结构有一定了解的工程开发人员。

编程手册的主要内容

本手册由四章内容组成，详细介绍了运动控制器的Smart Home功能及编程实现。

相关文件

关于控制器的调试和安装，请参见随产品配套的运动控制器用户手册。
关于控制器基本功能使用，请参见随产品配套的《GTN 系列运动控制器编程手册之基本功能》
关于更复杂的控制器功能，请参见随产品配套的《GTN 系列运动控制器编程手册之高级功能》
关于扩展模块的使用，请参见随产品配套的扩展模块编程手册。

相关手册及控制器适用文档列表见于光盘的 manual 目录下。

目录

一、 指令列表

提示	本章表格中右侧的数字为“页码”，其中指令右侧的为“**三、指令详细说明**”中的对应页码，其他为章节页码，均可以使用“超级链接”进行索引。 本手册中所有字体为蓝色的指令（如 GTN_LoadDlm）均带有超级链接，点击可跳转至指令说明。 只有 GHN 控制卡支持 DLM 功能。

表 1　DLM 模块列表

指令	说明	页码
GTN_LoadDlm	将 DLM 模块下载在运动控制器	16
GTN_RunDlm	执行指定的 DLM	16
GTN_StopDlm	停止指定的 DLM	17
GTN_GetDlmStatus	读取 DLM 状态	15
GTN_SetDlmFunction	设置 DLM 模块挂接函数的参数	17
GTN_GetDlmFunction	读取 DLM 模块挂接函数的参数	15
GTN_DlmCommandInit	DLM 模块传输参数初始化	15
GTN_DlmCommadAdd16	传输 short 类型参数至控制器 DLM 模块	12
GTN_DlmCommadAdd32	传输 long 类型参数至控制器 DLM 模块	13
GTN_DlmCommadAddFloat	传输 float 类型参数至控制器 DLM 模块	13
GTN_DlmCommadAddDouble	传输 double 类型参数至控制器 DLM 模块	13
GTN_SendDlmCommand	将参数发送至控制器 DLM 模块	16
GTN_DlmCommadGet16	从控制器 DLM 模块读取 short 类型参数	13
GTN_DlmCommadGet32	从控制器 DLM 模块读取 long 类型参数	14
GTN_DlmCommadGetFloat	从控制器 DLM 模块读取 float 类型参数	14
GTN_DlmCommadGetDouble	从控制器 DLM 模块读取 double 类型参数	14

二、 例程

DLM 功能具有以下特点：a、C 编写。b、CCES 编辑、编译、调试。c、编译成机器码，直接在 DSP 上执行。d、能灵活嵌入到 DSP 定时中断、背景任务、指令执行。e、代码中仍可以调用 GT 指令。f、运行时动态下载执行。其最大的特点是能将用户的算法，灵活的嵌入到 DSP 定时中断、背景任务中，且代码执行效率与 DSP 中代码的执行效率一样。

1、**例程实现功能**：正弦曲线位置规划，单 Kp 伺服控制跟随。

为了表述方便，直接在 PC 机上调用动态链接库发送指令访问控制器的程序称为“应用程序”，下载到运动控制器上执行的程序称为“DLM 程序”。DLM 程序与应用程序的关系如图二- 1 所示。

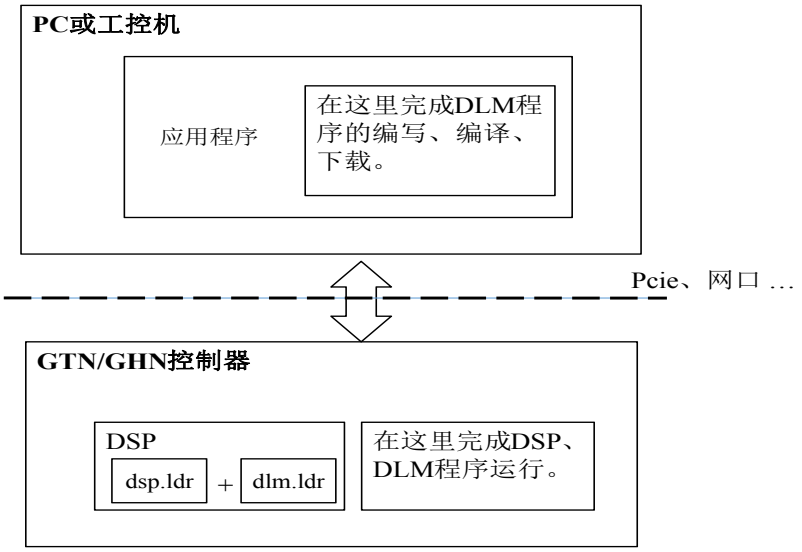

图 1　运动程序与应用程序的关系

2、开发流程：

（1）首先在编程软件 CCES2.5.0 中，编写 DLM 程序，实现正弦曲线形态位置规划器和 Kp 控制器，并生成 dlm.ldr 文件。DLM 程序中挂接的函数有：a、自定义规划器（Profile()），产生正弦曲线形态的规划位置。b、自定义控制器（Servo()），单 Kp 控制，跟随规划位置。c、指令系统（HostCommand()），用于应用程序与 DLM 程序数据交互，如设置 DLM 程序中 Kp 的值。d、背景任务函数（Background()），用于逻辑管理，例程中检测到 GPO Bit 位为 1 时，对应轴开始规划，跟随。

（2）、通过应用程序调用相关指令下载 dlm.ldr，设置 DLM 程序中需要应用程序动态改变的参数。

DLM 程序解析（工程见“dlm_ghn”）

注意：例程中红色部分不能修改，其他可以根据自己需求和编程习惯自定义。

```
#define GT_API extern "C" short

#include <adi_libldr.h>
#include "gts.h"

GT_API DlmLoad(short id,void *pService);
short HostCommand(short code,long index,TCommandParseItem *pParam,TCommandParseItem *pResult)

short Background(void)
short Profile(short axis,long long *pPrfPos,short *pRun)
short Servo(short control,short *pValue)

short Dlm_SetPid(TCommandParseItem *pParam,TCommandParseItem *pResult)
short Dlm_GetPid(TCommandParseItem *pParam,TCommandParseItem *pResult)

#define AXIS_MAX      8
#define CORE_DLM      2
```

```
float gKp[AXIS_MAX];
short gPrfRunSts[AXIS_MAX];
float gPrfTime[AXIS_MAX];

GT_API DlmInit(short id)
{
    short rtn;
    TDlmFunction dlmFunction;

    unsigned long clock;
    unsigned long loop;
    // 设置 DLM 程序的密码核版本信息
    rtn = GT_SetDlmInfo(0x1234,0x5678,1,0x20170825);

    // 挂接指令函数，在背景任务中触发式执行，用于 DLM 程序与 PC 进行数据交互
    rtn = GT_HookDlmHostCommand(HostCommand);   // HostCommand 为自定义函数
    dlmFunction.enable = 1;
    dlmFunction.function = DLM_FUNCTION_COMMAND;
    dlmFunction.value = 0;        // 对于不同的挂接函数 value 的意义不一样，该函数无意义。
    rtn = GT_SetDlmFunction(id,&dlmFunction);

    // 挂接背景任务函数，在背景任务中循环执行。一般用于逻辑管理，函数中可调用 GT 指令。
    rtn = GT_HookDlmBackground(Background);
    dlmFunction.enable = 1;
    dlmFunction.function = DLM_FUNCTION_BACKGROUND;
    dlmFunction.value = 0;        // 无意义
    rtn = GT_SetDlmFunction(id,&dlmFunction);

    // 挂接伺服任务函数，在中断中定周期执行。可编写自己的控制算法。
    rtn = GT_HookDlmServo(Servo);
    dlmFunction.enable = 1;
    dlmFunction.function = DLM_FUNCTION_SERVO;
    dlmFunction.value = 0xff;    // 按位使能，对应位为 1，表示该轴使用 DLM 中的伺服控制算法。
    rtn = GT_SetDlmFunction(id,&dlmFunction);

    // 挂接规划任务函数，在中断中定周期执行。可编写自己的规划算法。
    rtn = GT_HookDlmProfile(Profile);
    dlmFunction.enable = 1;
    dlmFunction.function = DLM_FUNCTION_PROFILE;
    dlmFunction.value = 0xff;    // 按位使能，对应位为 1，表示该轴使用 DLM 中的规划算法。
    rtn = GT_SetDlmFunction(id,&dlmFunction);

    return CMD_SUCCESS;
}

adi_libldr_Symbol gDlmSymbolTable[] =
```

```
{
        { ADI_LIBLDR_MAGIC_SYMNAME, ADI_LIBLDR_MAGIC_SYMADDR },
        { "DlmLoad.", (const void *)DlmLoad },
        { "DlmInit.", (const void *)DlmInit },
        { 0, 0 }
};

//  自定义指令系统，在背景任务中触发式执行，用于 DLM 程序与应用程序间的数据交互，应用程序调用
//  GTN_SendDlmCommand 一次，就执行一次 HostCommand()函数。接口中 code、index 与应用程序
//  GTN_DlmCommandInit()函数接口中的 code、index 相对应，pParam 为应用程序传递给 DLM 程序的数
//  据，pResult 为 DLM 程序返回给应用程序的数据
short HostCommand(short code,long index,TCommandParseItem *pParam,TCommandParseItem *pResult)
{

    float f1;
    long double d1,d2,sum;

    switch(code)
    {
    case 1:
        return Dlm_GetPid(pParam,pResult);
    case 2:
        return Dlm_SetPid(pParam,pResult);
    default:
        return CMD_ERROR_UNKNOWN;
    }
}

short Dlm_SetPid(TCommandParseItem *pParam,TCommandParseItem *pResult)
{
    short axis;
    float kp;

    // 读取应用程序传递的轴号
    CommandGet16(pParam,&axis);
    axis--;
    // 读取应用程序传递的 kp 值。
    CommandGetFloat(pParam,&kp);
    // 对 DLM 对应轴的 gKp[]变量赋值。
    gKp[axis] = kp;

    return CMD_SUCCESS;
}

short Dlm_GetPid(TCommandParseItem *pParam,TCommandParseItem *pResult)
{
```

```
    short axis;

    // 读取应用程序传递的轴号
    CommandGet16(pParam,&axis);
    axis--;

    // 将 DLM 对应轴的 gKp[]值传递给应用程序
    CommandAddFloat(pResult,gKp[axis]);

    return CMD_SUCCESS;
}

// 挂接背景任务函数，在背景任务中循环执行。一般用于逻辑管理，只有该挂接函数中才能使用 GT 指令。
short Background(void)
{
    short rtn,i;
    short runSts;
    long doValue;

    // 读取 GPO 的状态。
    rtn = GTN_GetDo(CORE_DLM,MC_GPO,&doValue);
    for(i=0;i<AXIS_MAX;i++)
    {
        runSts = (doValue>>i) & 1;
        // 根据 gPrfRunSts 的状态管理规划器、控制的启动停止（在 Profile()、Servo()函数中使用）
        gPrfRunSts[i] = runSts;
    }

    return CMD_SUCCESS;
}

//  自定义规划器，在中断中定周期执行。接口中 axis 为 DSP 传递给 DLM 的轴号，DLM 程序中的规划
//  器状态和计算的规划位置通过接口 pPrfPos、pRun 传递给 DSP。
short Profile(short axis,long long *pPrfPos,short *pRun)
{
    float prfPos;
    float sita;
    axis --;
    if(1 == gPrfRunSts[axis])
    {
        gPrfTime[axis] += 0.00025;
        sita = 100*gPrfTime[axis];

        // 正弦规划
        prfPos = 10000*sin(sita);
```

```
            // 计算出的规划值返回给 DSP
            *pPrfPos = 65536 * prfPos;

            // DLM 中规划标志返回给 DSP，作为 DSP 规划器的运动标志
            *pRun = 1;
        }
        else
        {
            gPrfTime[axis] = 0;

            *pPrfPos = 0;
            *pRun = 0;
        }

        return CMD_SUCCESS;
}

//  自定义控制器，在中断中定周期执行。接口中 control 为 DSP 传递给 DLM 程序的控制轴号，计算的
//  DAC 赋值给*pValue，返回给 DSP。
short Servo(short control,short *pValue)
{
        short rtn;
        long long data64;
        float prfPos,encPos,value;
        control --;

        if(true == gPrfRunSts[control])
        {
            // 读取 DSP 中的规划位置，GetMcVarInt64()为 DSP 与 DLM 的接口函数，直接调用。
            rtn = GetMcVarInt64(CORE_DLM,WATCH_VAR_AXIS_PRF_POS,control+1,&data64);
            prfPos = data64 / 65536.0;

            // 读取 DSP 中的编码器位置
            rtn = GetMcVarInt64(CORE_DLM,WATCH_VAR_ENC_POS,control+1,&data64);
            encPos = data64 / 65536.0;

            // 简单的 kp 运算
            value = gKp[control] * (prfPos - encPos);

            // 计算出的 DAC 值返回给 DSP，通过轴通道输出
            *pValue = (short)value;
        }
        else
        {
             *pValue = 0;
        }
```

```
    return CMD_SUCCESS;
}
```

应用程序解析（工程见“02_TestVs2010”）

```
#define AXIS            1
#define CORE            2

void Dlm_SetKp(short axis,float kp);
void Dlm_GetKp(short axis,float *pKp);

void Main（void）
{
    short rtn;
    short id;
    float m_DlmKp;

    rtn = GTN_Open();
    rtn = GTN_Reset(CORE);
    rtn = GTN_AlarmOff(CORE,AXIS);
    rtn = GTN_LmtsOff(CORE,AXIS,-1);
    rtn = GTN_ClrSts(CORE,AXIS,8);
    // 闭环模式下才能使用 DLM 中的伺服控制算法。
    rtn = GTN_CtrlMode(CORE,AXIS,0);

    // vender/module 必须和 DLM 程序 GT_SetDlmInf()指令的 vender/module 一致，才能正常跑起来
    long vender = 0x1234;
    long module = 0x5678;
    rtn = GTN_LoadDlm(CORE,vender,module,"dlm.ldr",&id);
    rtn = GTN_RunDlm(CORE,id);

    // 读取 DLM 伺服挂接函数 Servo()中 gKp[]的值
    Dlm_GetKp(AXIS,&m_DlmKp)
    m_DlmKp = 1;
    // 设置 DLM 伺服挂接函数 Servo()中 gKp[]的值
    Dlm_SetKp(AXIS,m_DlmKp);

    rtn = GTN_AxisOn(CORE,AXIS);

    // 逻辑管理，GPO 各 Bit 位为 1 时，对应的轴开始做规划、伺服。
    rtn = GTN_SetDo(CORE,MC_GPO,0);
}
```

```
void Dlm_SetKp(short axis,float kp)
{
    short rtn;
    short code, index;
    short returnValue;
    code = 2;
    index = 1;
    rtn = GTN_DlmCommandInit(CORE,code,index);
    rtn = GTN_DlmCommandAdd16(CORE,axis);
    rtn = GTN_DlmCommandAddFloat(CORE,kp);
    //  应用程序将 code,index,axis,kp 传递给 DLM 程序，触发式执行，应用程序调用
    GTN_SendDlmCommand 一次，
    //  执行一次 DLM 程序中的 HostCommand()函数
    rtn = GTN_SendDlmCommand(CORE,0,&returnValue);

}

void Dlm_GetKp(short axis,float *pKp)
{
    short rtn;
    short code,index;
    short returnValue;

    code = 2;
    index = 1;
    rtn = GTN_DlmCommandInit(CORE,code,index);

    rtn = GTN_DlmCommandAdd16(CORE,axis);

    //  PC 将 code,index,axis 传递给 DLM 程序，触发式执行。
    rtn = GTN_SendDlmCommand(CORE,0,&returnValue);

    //  DLM 程序执行 HostCommand()函数后，返回 DLM 程序中 kp 的值
    rtn = GTN_DlmCommandGetFloat(CORE,pKp);
}
```

三、 指令详细说明

指令 1　GTN_DlmCommadAdd16

指令原型	shortGTN_DlmCommandAdd16(short core,shortdata)		
适用板卡	GTN 系列产品		
指令说明	传输 short 类型参数至控制器 DLM 模块		
指令类型	立即指令	章节页码	5

指令参数	该指令共有 2 个参数，参数的详细信息如下。
core	内核，正整数，取值范围[2,2]
data	参数数值
指令返回值	请参照《GTN 系列运动控制器编程手册之基本功能.docx》中的第 3 章
相关指令	无
指令示例	应用程序

指令 2　GTN_DlmCommadAdd32

指令原型	shortGTN_DlmCommandAdd32 (short core,longdata)		
适用板卡	GTN 系列产品		
指令说明	传输 long 类型参数至控制器 DLM 模块		
指令类型	立即指令	章节页码	5
指令参数	该指令共有 2 个参数，参数的详细信息如下。		
core	内核，正整数，取值范围[2,2]		
data	参数数值		
指令返回值	请参照《GTN 系列运动控制器编程手册之基本功能.docx》中的第 3 章		
相关指令	无		
指令示例	无。		

指令 3　GTN_DlmCommadAddDouble

指令原型	shortGTN_DlmCommandAddDouble(short core,doubledata)		
适用板卡	GTN 系列产品		
指令说明	传输 double 类型参数至控制器 DLM 模块		
指令类型	立即指令	章节页码	5
指令参数	该指令共有 2 个参数，参数的详细信息如下。		
core	内核，正整数，取值范围[2,2]		
data	参数数值		
指令返回值	请参照《GTN 系列运动控制器编程手册之基本功能.docx》中的第 3 章		
相关指令	无		
指令示例	应用程序		

指令 4　GTN_DlmCommadAddFloat

指令原型	shortGTN_DlmCommandAddFloat(short core,floatdata)		
适用板卡	GTN 系列产品		
指令说明	传输 float 类型参数至控制器 DLM 模块		
指令类型	立即指令	章节页码	5
指令参数	该指令共有 2 个参数，参数的详细信息如下。		
core	内核，正整数，取值范围[2,2]		
data	参数数值		
指令返回值	请参照《GTN 系列运动控制器编程手册之基本功能.docx》中的第 3 章		
相关指令	无		
指令示例	无。		

指令 5　GTN_DlmCommadGet16

指令原型	shortGTN_DlmCommandGet16(short core,short*pValue)		
适用板卡	GTN 系列产品		
指令说明	从控制器 DLM 模块 short 类型参数		
指令类型	立即指令	章节页码	5
指令参数	该指令共有 2 个参数，参数的详细信息如下。		
core	内核，正整数，取值范围[2,2]		
pValue	参数数值		
指令返回值	请参照《GTN 系列运动控制器编程手册之基本功能.docx》中的第 3 章		
相关指令	无		
指令示例	无。		

指令 6　GTN_DlmCommadGet32

指令原型	shortGTN_DlmCommandGet32(short core,long*pValue)		
适用板卡	GTN 系列产品		
指令说明	从控制器 DLM 模块 long 类型参数		
指令类型	立即指令	章节页码	5
指令参数	该指令共有 2 个参数，参数的详细信息如下。		
core	内核，正整数，取值范围[2,2]		
pValue	参数数值		
指令返回值	请参照《GTN 系列运动控制器编程手册之基本功能.docx》中的第 3 章		
相关指令	无		
指令示例	无。		

指令 7　GTN_DlmCommadGetDouble

指令原型	shortGTN_DlmCommandGetDouble (short core,double*pValue)		
适用板卡	GTN 系列产品		
指令说明	从控制器 DLM 模块 double 类型参数		
指令类型	缓存区指令	章节页码	5
指令参数	该指令共有 2 个参数，参数的详细信息如下。		
core	内核，正整数，取值范围[2,2]		
pValue	参数数值		
指令返回值	请参照《GTN 系列运动控制器编程手册之基本功能.docx》中的第 3 章		
相关指令	无		
指令示例	无。		

指令 8　GTN_DlmCommadGetFloat

指令原型	shortGTN_DlmCommandGetFloat(short core,float*pValue)		
适用板卡	GTN 系列产品		
指令说明	从控制器 DLM 模块 float 类型参数		
指令类型	立即指令	章节页码	5
指令参数	该指令共有 2 个参数，参数的详细信息如下。		
core	内核，正整数，取值范围[2,2]		
pValue	参数数值		

指令返回值	请参照《GTN 系列运动控制器编程手册之基本功能.docx》中的第 3 章
相关指令	无
指令示例	无。

指令 9　GTN_DlmCommandInit

指令原型	shortGTN_DlmCommandInit(short core,shortcode,long index)		
适用板卡	GTN 系列产品		
指令说明	DLM 模块参数初始化		
指令类型	立即指令	章节页码	5
指令参数	该指令共有 3 个参数，参数的详细信息如下。		
core	内核，正整数，取值范围[2,2]		
code	指令字		
index	指令索引		
指令返回值	请参照《GTN 系列运动控制器编程手册之基本功能.docx》中的第 3 章		
相关指令	无		
指令示例	应用程序		

指令 10　GTN_GetDlmFunction

指令原型	shortGTN_GetDlmFunction (short core,short id, TDlmFunction *pFunction)		
适用板卡	GTN 系列产品		
指令说明	读取 DLM 模块挂接函数的参数		
指令类型	立即指令	章节页码	5
指令参数	该指令共有 3 个参数，参数的详细信息如下。		
core	内核，正整数，取值范围[2,2]		
id	DLM 模块序号		
pFunction	见指令 GTN_SetDlmFunction 参数 3		
指令返回值	请参照《GTN 系列运动控制器编程手册之基本功能.docx》中的第 3 章		
相关指令	无		
指令示例	无。		

指令 11　GTN_GetDlmStatus

指令原型	shortGTN_GetDlmStatus(short core,short id,TDlmStatus *pStatus)		
适用板卡	GTN 系列产品		
指令说明	读取 DLM 模块状态		
指令类型	立即指令	章节页码	5
指令参数	该指令共有 3 个参数，参数的详细信息如下。		
core	内核，正整数，取值范围[2,2]		
id	DLM 模块号，取值范围：[0, 1)		

pStatus	DLM 模块状态，定义如下。
	typedef struct TDlmStatus
	{
	long version;//版本号
	long date;//日期（年月日）
	short enable;//使能状态,1:使能 0:未使能		

	long function;//DLM模块执行位置信息，按位表示
	//背景任务：DLM_FUNCTION_BACKGROUND 2
	//接收传输参数：DLM_FUNCTION_COMMMAND 3
	} TDlmStatus;

指令返回值	请参照《GTN 系列运动控制器编程手册之基本功能.docx》中的第 3 章
相关指令	无
指令示例	应用程序

指令 12 GTN_LoadDlm

指令原型	shortGTN_LoadDlm(short core,long vender,long module,char *fileName,short *pId)		
适用板卡	GTN 系列产品		
指令说明	下载 DLM 模块到运动控制器		
指令类型	立即指令	章节页码	5
指令参数	该指令共有 5 个参数，参数的详细信息如下。		
core	内核，正整数，取值范围[2,2]		
vender	校验码低 32 位		
module	校验码高32位		
fileName	固件名称		
pId	DLM模块序号，当前只支持1个DLM模块，默认值为0		
指令返回值	请参照《GTN 系列运动控制器编程手册之基本功能.docx》中的第 3 章		
相关指令	无		
指令示例	应用程序		

指令 13 GTN_RunDlm

指令原型	shortGTN_RunDlm(short core, shortid)		
适用板卡	GTN 系列产品		
指令说明	运行 DLM 模块		
指令类型	立即指令	章节页码	5
指令参数	该指令共有 2 个参数，参数的详细信息如下。		
core	内核，正整数，取值范围[2,2]		
id	DLM 模块号，取值范围：[0, 1)		
指令返回值	请参照《GTN 系列运动控制器编程手册之基本功能.docx》中的第 3 章		
相关指令	无		
指令示例	应用程序		

指令 14 GTN_SendDlmCommand

指令原型	shortGTN_SendDlmCommand (short core,short id,short*pReturnValue)		
适用板卡	GTN 系列产品		
指令说明	将参数发送至控制器 DLM 模块		
指令类型	立即指令	章节页码	5
指令参数	该指令共有 2 个参数，参数的详细信息如下。		
core	内核，正整数，取值范围[2,2]		
id	DLM 模块序号		
pReturnValue	DLM 模块返回值		

	COMMAND_ERROR_DLM_COMMAND 200 //DLM 指令集嵌套 COMMNAD_ERROR_PARSE 9 //指令译码错误 其他可参照指令返回值
指令返回值	请参照《GTN 系列运动控制器编程手册之基本功能.docx》中的第 3 章
相关指令	无
指令示例	应用程序

指令 15 GTN_SetDlmFunction

指令原型	shortGTN_SetDlmFunction (short core,short id,TDlmFunction *pFunction)		
适用板卡	GTN 系列产品		
指令说明	设置 DLM 模块挂接函数的参数		
指令类型	立即指令	章节页码	5
指令参数	该指令共有 3 个参数，参数的详细信息如下。		
core	内核，正整数，取值范围[2,2]		
id	DLM 模块序号		

pFunction	type struct TDlmFunction
	{
	short function;//挂接函数的类型，
	//#define DLM_FUNCTION_COMMAND 3//传输数据参数类型
	// #define DLM_FUNCTION_BACKGROUND 2 //挂接在背景任务
	short enable;//挂接函数使能状态。1：使能 0：未使能
	short value; //挂接函数参数
	}TDlmFunction		

指令返回值	请参照《GTN 系列运动控制器编程手册之基本功能.docx》中的第 3 章		
相关指令	无		
指令示例	无。		

指令 16 GTN_StopDlm

指令原型	shortGTN_StopDlm(short core, shortid)		
适用板卡	GTN 系列产品		
指令说明	停止 DLM 模块		
指令类型	立即指令	章节页码	5
指令参数	该指令共有 2 个参数，参数的详细信息如下。		
core	内核，正整数，取值范围[2,2]		
id	DLM 模块号，取值范围：[0, 1)		
指令返回值	请参照《GTN 系列运动控制器编程手册之基本功能.docx》中的第 3 章		
相关指令	无		
指令示例	无。		