

# GTS 系列运动控制器编程手册

---

## DMA 功能

2019 年 3 月

© 2019 固高科技版权所有

# 版权申明

固高科技有限公司

保留所有权力

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：[googol@googletech.com](mailto:googol@googletech.com)

网址：<http://www.googletech.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：[sales@googletech.com](mailto:sales@googletech.com)

[info@googletech.com](mailto:info@googletech.com)

網址：<http://www.googletech.com>

# 文档版本

版本号	修订日期
1.0	2018年10月16日
1.1	2019年03月13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](mailto: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

## 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《GTS 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《GTS 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。



注意

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

版权申明 .....	1
联系我们 .....	1
文档版本 .....	2
前言 .....	3
目录 .....	5
一、 指令列表.....	6
二、 重点说明 .....	6
三、 例程 .....	6
例程 1：未使用前瞻的插补运动 DMA .....	6
例程 2：使用新前瞻的插补运动 DMA .....	9
四、 指令详细说明 .....	12

## 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“四、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 [GT\\_CrdHsOn](#)）均带有超级链接，点击可跳转至指令说明。

表 1 DMA 功能指令列表

指令	说明
<a href="#">GT_CrdHsOn</a>	打开插补 DMA
<a href="#">GT_CrdHsOff</a>	关闭插补 DMA

## 二、重点说明

- 1、DMA 功能为数据段批次压入功能，每次压入的数据段的数量由用户设定，压完数据段之后，再调用 [GT\\_CrdData](#) 指令将剩余数据段压入缓冲区。
- 2、开启 DMA 功能后，[GTN\\_CrdSpace](#) 将查询的为 PC 端的剩余数据段空间（插补共有 4096 段空间）。

## 三、例程

### 例程 1：未使用前瞻的插补运动 DMA

```
short commandhandle(char * command,short returnValue)
```

```
{
    if(0!=returnValue)
    {
        printf("%s=%d\n",command,returnValue);
    }
    return 0;
}
```

```
int main(int argc,char *argv[])
```

```
{
    short rtn, crd,fifo;
    crd = 1;fifo = 1;
    short axisNumber;
    short seg = 0;
    long space;
```

```
bool startflag = true;
bool dataFlag = true;
long xPos,yPos;
double synVel,synAcc,velEnd;
xPos = 0;yPos = 0;synVel = 500;synAcc = 1;velEnd = 0;
short link = 1;
```

```
//此处省略控制器初始化
```

```
rtn = GT_ZeroPos(1,2);
commandhandle("GT_ZeroPos",rtn);
```

```
for(axisNumber=1;axisNumber<=8;axisNumber++)
```

```
{
    rtn = GT_AlarmOff(axisNumber);
    commandhandle("GT_AlarmOff",rtn);
    rtn = GT_LmtsOff(axisNumber);
    commandhandle("GT_LmtsOff",rtn);
    rtn = GT_ClrSts(axisNumber);
    commandhandle("GT_ClrSts",rtn);
    rtn = GT_AxisOn(axisNumber);
    commandhandle("GT_AxisOn",rtn);
}
```

```
TCrdPrm crdPrm;
memset(&crdPrm, 0, sizeof(crdPrm));
crdPrm.dimension = 2;
crdPrm.synVelMax = 3000;
crdPrm.synAccMax = 40;
crdPrm.evenTime = 50;
crdPrm.profile[0] = 1;
crdPrm.profile[1] = 2;
crdPrm.setOriginFlag = 1;
crdPrm.originPos[0] = 0;
crdPrm.originPos[1] = 0;
rtn = GT_SetCrdPrm(crd,&crdPrm);
commandhandle("GT_SetCrdPrm",rtn);
rtn = GT_CrdClear(crd,fifo);
commandhandle("GT_CrdClear",rtn);
```

```
//打开插补DMA功能
```

```
rtn = GT_CrdHsOn(coreTemp,crd,fifo,link,100,0);
commandhandle("GT_CrdHsOn",rtn);
```

```
while(1)
{
```



```
rtn = GT_CrdSpace(coreTemp,crd,&space,fifo);
commandhandle("GT_CrdSpace",rtn);
if((0 != space)&&(dataFlag == true))
{
    seg = seg+1;
    xPos = xPos + 2;
    rtn = GT_LnXY(coreTemp,crd,xPos,yPos,synVel,synAcc,velEnd,fifo);
    commandhandle("GT_CrdStart",rtn);
    SetDlgItemInt(AfxGetApp()->m_pMainWnd->m_hWnd, IDC_EDIT12,seg,false);
    if( 10000 == seg)
    {
        do
        {
            rtn = GT_CrdData(coreTemp,crd,NULL,fifo);
        } while (rtn);
        dataFlag = false;
    }
}

long segment;
rtn = GT_GetRemainderSegNum(coreTemp,crd,&segment,fifo);
commandhandle("GT_GetPrfPos",rtn);

if((true == startflag)&&(0 != segment))
{
    rtn = GT_CrdStart(coreTemp,1,1);
    commandhandle("GT_CrdStart",rtn);
    startflag = false;
}

double prfPos[2];
rtn = GT_GetPrfPos(coreTemp,1,&prfPos,2,0);
commandhandle("GT_GetPrfPos",rtn);
printf("X_Pos = %d    Y_Pos = %d\r\n", prfPos [0], prfPos [1]);
}

return 0;
}
```

## 例程 2：使用新前瞻的插补运动 DMA

//新前瞻初始化

```
void InitialNewLookAhead(short core,short crd,short fifo,int lookAheadNum)
```

```
{
```

```
    short rtn;
```

```
    EMachineMode machineMode; //机床类型
```

```
    EVelSettingDef velDefineMode; //速度定义模式
```

```
    int axisLimitMode[8]; //轴限制模式
```

```
    EWorkLimitMode workLimitMode; //工件坐标系限制模式
```

```
    int axisFollowMode[8]; //轴跟随模式
```

```
    TLookAheadParameter lookAheadPara; //前瞻参数
```

```
    machineMode = NORMAL_THREE_AXIS; //标准三轴机床
```

```
    velDefineMode = NORMAL_DEF_VEL; //输入速度为三轴合成速度
```

```
    workLimitMode = WORK_LIMIT_VALID; //工件坐标系限制生效
```

```
    for (int i=0;i<8;++i)
```

```
    {
```

```
        axisLimitMode[i] = AXIS_LIMIT_NONE; //轴限制不生效
```

```
        axisFollowMode[i] = 0; //非跟随轴
```

```
    }
```

```
    //坐标系第4轴为跟随轴并限制轴运动能力
```

```
    axisLimitMode[3]=AXIS_LIMIT_MAX_VEL|AXIS_LIMIT_MAX_DV; //轴最大速度和速度  
最大跳变生效
```

```
    axisFollowMode[3]=1; //跟随轴
```

```
    lookAheadPara.lookAheadNum = lookAheadNum;
```

```
    lookAheadPara.time = 1; //时间常数
```

```
    lookAheadPara.radiusRatio = 50; //曲率参数
```

```
    for (int i=0;i<8;++i)
```

```
    {
```

```
        lookAheadPara.vMax[i] = 5000; //轴最大速度限制
```

```

    lookAheadPara.aMax[i] = 100;//轴最大加速度限制
    lookAheadPara.DVMax[i] = 500;//轴跳变速度限制
    lookAheadPara.axisRelation[i] = i+1; //坐标系轴与前瞻轴一一映射
    lookAheadPara.scale[i] = 1000; //脉冲当量
}

rtn = GT_SetupLookAheadCrd(crd,machineMode);//设置机床模式
rtn = GT_SetAxisLimitModeLa(crd,axisLimitMode);//设置轴限制模式
rtn = GT_SetAxisVelValidModeLa(crd,0xF);//设置轴速度有效,按位设置,0xF表示前4个轴
rtn = GT_InitLookAheadEx(crd,&lookAheadPara,fifo,0); //设置前瞻参数 (需要放在最后设置)
}

int _tmain(int argc, _TCHAR* argv[])
{
    short i;
    short rtn;
    short core = 1;
    rtn = GT_Open();
    rtn = GT_Reset(core);
    for (i=1;i<5;i++)
    {
        rtn = GT_AlarmOff(i);
        rtn = GT_LmtsOff(i);
    }
    rtn = GT_ClrSts(1,4);
    rtn = GT_ZeroPos(1,4);
    TCrdPrm crdPrm;
    rtn = GT_GetCrdPrm(1,&crdPrm);
    crdPrm.dimension=2; // 坐标系为二维坐标系
    crdPrm.synVelMax=500; // 最大合成速度: 500pulse/ms
    crdPrm.synAccMax=1; // 最大加速度: 1pulse/ms^2
    crdPrm.evenTime = 50; // 最小匀速时间: 50ms
    crdPrm.profile[0] = 1; // 规划器1对应到X轴
    crdPrm.profile[1] = 2; // 规划器2对应到Y轴

    rtn = GT_SetCrdPrm(1, &crdPrm);//建立1号坐标系, 设置坐标系参数
    rtn = GT_CrdClear(1, 0);//清除此缓存区

    //打开插补DMA功能
    rtn = GT_CrdHsOn(1,0,1,200,0);

    double synVel=100.0*(1000/1000),synAcc=10*(1000000/1000);//单位换算,此处分别为mm/s
    和mm/s^2
    //前瞻初始化

```

```
InitialNewLookAhead(crd,0,300);

int count = 1000;
double x = 0;
double y = 0;
for(i=0;i<count;i++)
{
    x += 0.01;
    y += 0;
    rtn += GT_LnXYZEx(1,x,y,0,synVel,synAcc,0,0,0);
}
//压入数据并启动运动
while(1)
{
    rtn=GT_CrdDataEx(1,NULL,0);
    if (rtn==0)
    {
        break;
    }
}
rtn=GT_CrdStart(1,0);
return 0;
}
```

## 四、指令详细说明

### 指令 1 GT\_CrdHsOn

指令原型	GT_CrdHsOn(short crd,short fifo,short link=1,unsigned short threshold=200,short lookAheadInMc=0)		
指令说明	打开插补 DMA		
指令类型	立即指令，调用后立即生效。	章节页码	6
指令参数	该指令共有 6 个参数，参数的详细信息如下。		
crd	坐标系号，取值范围[1,2]		
fifo	FIFO号，取值范围[0,1]		
link	默认取 1，取值范围[1,4]		
threshold	阈值，PC 机指令达到该阈值时自动启动 DMA 发送数据		
lookAheadInMc	默认取 0		

### 指令 2 GT\_CrdHsOff

指令原型	GT_CrdHsOff(short crd,short fifo);		
指令说明	关闭插补 DMA		
指令类型	立即指令，调用后立即生效。	章节页码	6
指令参数	该指令共有 2 个参数，参数的详细信息如下。		
crd	坐标系号，取值范围[1,2]		

**fifo**

FIFO号，取值范围[0,1]