固高科技
GOOGOL TECHNOLOGY

扩展功能

GXN 系列运动控制器编程手册

PSO 功能

R1.3

2022 年 4 月

© 2022 固高科技版权所有

版权申明

固高科技股份有限公司

保留所有权力

固高科技股份有限公司（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

固高科技股份有限公司

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@googoltech.com

网址：http://www.googoltech.com.cn

固高科技（海外）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@googoltech.com

info@googoltech.com

網址：http://www.googoltech.com

臺灣固高科技股份有限公司

地址：台中市西屯區福中二路 10 巷 22 號 2 樓

電話：+886-4-23588245

傳真：+886-4-23586495

電子郵件：twinfo@googoltech.com

© 2021 固高科技 版权所有

文档版本

版本号	修订日期
1.0	2019 年 01 月 21 日
1.1	2020 年 01 月 09 日
1.2	2020 年 04 月 29 日
1.3	2020 年 11 月 01 日

© 2021 固高科技 版权所有

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

固高科技股份有限公司

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PSO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《GXN 系列运动控制器编程手册之基本功能》

关于更多的控制器功能，请参见随产品配套的《GXN 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。

产品相关手册及安装文件如驱动程序、dll 文件、例程、Demo 等，请登录固高科技公司网站下载，网址为：www.googoltech.com.cn/pro_view-53.html

© 2021 固高科技 版权所有

目录

© 2021 固高科技 版权所有

一、 指令列表

本章表格中的指令右侧的页码为“**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 GTN_PosCompareClear）均带有超级链接，点击可跳转至指令说明。

指令	说明
GTN_PosCompareClear	清除位置比较输出缓存区数据
GTN_PosCompareStop	停止位置比较输出
GTN_SetPosCompareMode	设置位置比较输出的模式
GTN_GetPosCompareMode	读取位置比较输出的模式
GTN_SetPosComparePsoPrm	设置等间距位置比较输出参数
GTN_GetPosComparePsoPrm	读取等间距位置比较输出参数
GTN_PosCompareStart	启动位置比较输出
GTN_GetTerminalPermitEx	读取硬件通道信号输出类型
GTN_SetTerminalPermitEx	设置硬件通道信号输出类型
GTN_BufPosComparePsoPrm	缓冲区设置 PSO 参数
GTN_BufPosCompareStart	缓冲区启动位置比较或 PSO
GTN_BufPosCompareStop	缓冲区关闭位置比较或 PSO
GTN_SetPosCompareMultiPsoPrm	设置 PSO 多间距参数

二、 重点说明

1、 设置 PSO 输出控制权

GNM-401、GNM-402、GNM-403、GNM-601、GNM-602 都支持 2 路 PSO 功能。

以 GNM-403 模块为例，GNM-403 模块默认上电之后，CN5-pin1/pin6（LASER+/ LASER-）输出控制权为激光的开关光，CN5-pin2/pin7（PWM+/ PWM-）的输出控制权为 PWM 波输出，当需要使用 PSO 功能时，需要调用 GTN_SetTerminalPermitEx 将其中一路输出的控制权切换到 PSO 功能控制。

2、 设置 PSO 输出间距

调用等间距输出函数 GTN_SetPosComparePsoPrm 用来设置输出间距，输出间距为 X、Y 轴的合成间距（单位为 Pulse）。

© 2021 固高科技版权所有

三、 例程

```
int _tmain(int argc, _TCHAR* argv[])
{
    //设置位置比较输出参数
    short rtn;
    short posCompareIndex=1;
    short core=1;

    rtn = GTN_Open();
    rtn = GTN_Reset(core);
    rtn = GTN_LoadConfig(core,"GTS800_test.cfg");
    rtn = GTN_ClrSts(core,1,4);
    rtn = GTN_ZeroPos(core,1,4);

    //设置控制权
    short permit;
    permit = 0x2;
    rtn = GTN_SetTerminalPermitEx(core,1,MC_HSO,&permit,2,1);
    //设置位置比较输出模式
    TPosCompareMode prm;
    rtn = GTN_PosCompareClear(core,posCompareIndex);                    //清空位置比较输出数据
    //设置位置比较输出参数
    TPosCompareMode mode;
    rtn = GTN_GetPosCompareMode(core,posCompareIndex,&mode);
    mode.mode = 2;                      //0, fifo 模式;1,  liner 模式;2, 等间距输出模式
    mode.dimension = 2;                 //2 维模式
    mode.sourceMode = 1;                //输出比较源，:编码器,：脉冲计数器
    mode.sourceX = 1;                   //X 轴比较源[1,12]
    mode.sourceY = 2;                   //Y 轴比较源[1,12]
    mode.outputMode = 0;                //0：输出脉冲,：输出电平
    mode.outputCounter = 1;             //保留，需要大于.
    mode.outputPulseWidth = 100; //输出脉冲宽度，单位：us,电平模式下该参数无效
    mode.errorBand = 0;
    rtn = GTN_SetPosCompareMode(core,posCompareIndex,&mode);
    //设置等间距输出相关参数
    TPosComparePsoPrm psoPrm;
    rtn = GTN_GetPosComparePsoPrm(core,posCompareIndex,&psoPrm);
    psoPrm.count = 1;                   //输出个数，暂时保留，使用时可以先写
    psoPrm.syncPos = 1000;              //输出间距，X、Y 的合成距离，单位：Pulse
    rtn = GTN_SetPosComparePsoPrm(core,posCompareIndex,&psoPrm);    //启动位置比较输出
    rtn = GTN_PosCompareStart(core,posCompareIndex);
```


© 2021 固高科技版权所有

```
    //启动 Jog 运动
    TJogPrm jog;
    rtn = GTN_PrfJog(core,1);
    rtn = GTN_GetJogPrm(core,1,&jog);
    jog.acc = 1;
    jog.dec = 1;
    rtn = GTN_SetJogPrm(core,1,&jog);
    rtn = GTN_SetVel(core,1,10);

    rtn = GTN_PrfJog(core,2);
    rtn = GTN_GetJogPrm(core,2,&jog);
    jog.acc = 1;
    jog.dec = 1;
    rtn = GTN_SetJogPrm(core,2,&jog);
    rtn = GTN_SetVel(core,2,10);

    rtn = GTN_Update(core,0x3);

    while(1)
    {
        double pos[2];
        TPosCompareStatus sts;
        rtn = GTN_PosCompareStatus(core , posCompareIndex ,&sts );
        rtn = GTN_GetPrfPos(core,1,pos,2);
        printf("pos1=%lf,pos2=%lf,mode=%d,run=%d,space=%d,pulseCount=%ld,hso=%d,gpo=%d\r",
            pos[0],pos[1],sts.mode,sts.run,sts.space,sts.pulseCount,sts.hso,sts.gpo);
    }

    return 0;
}


//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
```


© 2021 固高科技版权所有

```
        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
        //axisLimitMode[2]=AXIS_LIMIT_MAX_VEL|AXIS_LIMIT_MAX_DV;//轴最大速度和速度最大跳变生效
        //axisFollowMode[2]=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 = GTN_SetupLookAheadCrd(core,crd,machineMode);//设置机床模式
        rtn = GTN_SetAxisFollowModeLa(core,crd,axisFollowMode);//设置跟随模式
        rtn = GTN_SetAxisLimitModeLa(core,crd,axisLimitMode);//设置轴限制模式
        rtn = GTN_SetAxisVelValidModeLa(core,crd,0xF);//设置轴速度有效，按位设置，xF 表示前个轴
        rtn = GTN_InitLookAheadEx(core,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 rtn;
        short posCompareIndex=1;
        short core=1;

        rtn = GTN_Open();
        rtn = GTN_Reset(core);
        rtn = GTN_LoadConfig(core,"GTS800_test.cfg");
        rtn = GTN_ClrSts(core,1,4);
        rtn = GTN_ZeroPos(core,1,4);
        rtn = GTN_AxisOn(core,1);
        rtn = GTN_AxisOn(core,2);
```


© 2021 固高科技版权所有

```
//设置控制权
short pPermit[16];
short permit;
rtn = GTN_GetTerminalPermitEx(core,1,MC_HSO,&permit,1,1);
permit = 0x2;
rtn = GTN_SetTerminalPermitEx(core,1,MC_HSO,&permit,1,1);

rtn = GTN_PosCompareStop(core,posCompareIndex);
rtn = GTN_PosCompareClear(core,posCompareIndex);                    //清空位置比较输出数据
//设置位置比较输出参数
TPosCompareMode mode;
rtn = GTN_GetPosCompareMode(core,posCompareIndex,&mode);
mode.mode = 3;
mode.dimension = 2;                 //2 维模式
mode.sourceMode = 0;                //输出比较源，:编码器,：脉冲计数器
mode.sourceX = 1;                   //X 轴比较源[1,12]
mode.sourceY = 2;                   //Y 轴比较源[1,12]
mode.outputMode = 0;                //0：输出脉冲,：输出电平
mode.outputCounter = 1;            //保留，需要大于.
mode.outputPulseWidth = 10; //输出脉冲宽度，单位：.1ms,电平模式下该参数无效
mode.errorBand = 100;
rtn = GTN_SetPosCompareMode(core,posCompareIndex,&mode);
if (rtn)
{
    printf("GTN_SetPosCompareMode=%d\n",rtn);
}
//设置等间距输出相关参数
TPosComparePsoPrm psoPrm;
rtn = GTN_GetPosComparePsoPrm(core,posCompareIndex,&psoPrm);
psoPrm.count = 1;                   //输出个数，暂时保留，使用时可以先写
psoPrm.syncPos = 10;               //输出间距，X、Y 的合成距离，单位：Pulse
rtn = GTN_SetPosComparePsoPrm(core,posCompareIndex,&psoPrm);

//启动位置比较输出
rtn = GTN_PosCompareStart(core,posCompareIndex);

//建立插补坐标系
TCrdPrm crdPrm;
rtn = GTN_GetCrdPrm(core,1,&crdPrm);
crdPrm.dimension=2; // 坐标系为二维坐标系
crdPrm.synVelMax=500; // 最大合成速度：pulse/ms
crdPrm.synAccMax=1; // 最大加速度：pulse/ms^2
crdPrm.evenTime = 50; // 最小匀速时间：ms
crdPrm.profile[0] = 1; // 规划器对应到 X 轴
crdPrm.profile[1] = 2; // 规划器对应到 Y 轴
```


© 2021 固高科技版权所有

```
    crdPrm.setOriginFlag = 1;
    crdPrm.originPos[0] = 0;
    crdPrm.originPos[1] = 0;

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

    double synVel=10*(1000/1000),synAcc=10*(1000000/1000);//单位换算，此处分别为 mm/s 和 mm/s^2
    InitialNewLookAhead(core,1,0,200);    //初始化前瞻

    //压入插补数据
    rtn = GTN_LnXYEx(core,1,2,0,synVel,synAcc,0,0,0);
    rtn = GTN_BufPosCompareStartEx(core,1,0,posCompareIndex);          //开启 PSO
    rtn = GTN_LnXYEx(core,1,4,0,synVel,synAcc,0,0,0);
    rtn = GTN_BufPosCompareStopEx(core,1,0,posCompareIndex);           //关闭 PSO
    rtn = GTN_LnXYEx(core,1,2,0,synVel,synAcc,0,0,0);
    while(1)
    {
        rtn=GTN_CrdDataEx(core,1,NULL,0);
        if (0 == rtn)
        {
            break;
        }
    }

    //启动插补
    rtn=GTN_CrdStart(core,1,0);
    //读取状态
    short run;
    long seg;
    do{
        double pos[4];
        double encPos[4];
        TPosCompareStatus sts;
        rtn = GTN_CrdStatus(core,1,&run,&seg);
        rtn = GTN_PosCompareStatus(core , posCompareIndex ,&sts );
        rtn = GTN_GetPrfPos(core,1,pos,4);
        rtn = GTN_GetEncPos(core,1,encPos,4);

printf("pos1=%lf,encPos1=%lf,mode=%d,run=%d,space=%d,pulseCount=%ld,hso=%d,gpo=%d\r",
                pos[0],pos[1],sts.mode,sts.run,sts.space,sts.pulseCount,sts.hso,sts.gpo);
    }while(1 == run);

    return 0;
}
```


© 2021 固高科技版权所有

四、指令详细说明

指令 1　GTN_PosCompareClear

指令原型	short GTN_PosCompareClear(short core, short posCompareIndex);		
指令说明	清除位置比较输出缓存区数据。		
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	核号，正整数，取值范围[1,1]。		
posCompareIndex	位置比较索引，正整数，取值范围[1,8]。		
指令返回值	请参照指令返回值列表。		
相关指令	无		
指令示例	例程		

指令 2　GTN_PosCompareStop

指令原型	short GTN_PosCompareStop(short core, short posCompareIndex);		
指令说明	停止位置比较输出。		
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	核号，正整数，取值范围[1,1]。		
posCompareIndex	位置比较索引，正整数，取值范围[1,8]。		
指令返回值	请参照指令返回值列表。		
相关指令	GTN_PosCompareStart		
指令示例	例程		

指令 3　GTN_SetPosCompareMode

指令原型	short GTN_SetPosCompareMode(short core,short posCompareIndex,TPosCompareMode *pMode);		
指令说明	设置位置比较输出模式。		
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	核号，正整数，取值范围[1,1]。		
posCompareIndex	位置比较索引，正整数，取值范围[1,8]。		

pMode	设置位置比较输出模式
	typedef struct
	{
	short mode; // 0：FIFO 模式，1：liner模式，2：PSO立即模式，3：PSO等		

© 2021 固高科技版权所有

	待到位触发模式
	short dimension; //1：1D，2：2D
	short sourceMode; // 位置比较源，0：编码器； 1：脉冲计数器
	short sourceX; // X 轴比较源 [1,12]
	short sourceY; // Y 轴比较源[1,12]
	short outputMode; // 输出模式：0:脉冲 1:电平，2:电平自动翻转
	short outputCounter; // (保留)
	unsigned short outputPulseWidth; // 输出脉冲宽度，单位为 1us，电平模式该参数无效
	unsigned short errorBand; // 二维位置比较输出误差带
	} TPosCompareMode;
	➢ 对于PSO立即模式（mode=2），PSO的开关立即执行；PSO等待到位触发模式（mode=3），PSO的开关会根据设置的误差带，以及当前的编码器/脉冲计数器是否到位来决定是否执行PSO开关操作。对于PSO立即模式，PSO开关立即执行，对于伺服滞后系统，电机还没有到位就执行PSO开关，会影响加工效果；对于PSO等待到位触发模式，能保证电机到位才执行PSO开关，能给保证加工效果。
	➢ fifo模式下：输出模式可选为：脉冲、电平
	➢ liner模式下：输出模式可选为：脉冲、电平自动翻转
	➢ PSO立即模式下：输出模式可选为：脉冲
	➢ PSO等待到位触发模式：输出模式可选为：脉冲

指令返回值	请参照指令返回值列表。
相关指令	GTN_GetPosCompareMode
指令示例	例程

指令 4 GTN_GetPosCompareMode

指令原型	short GTN_GetPosCompareMode(short core,short posCompareIndex,TPosCompareMode *pMode);		
指令说明	读取位置比较输出模式。		
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	核号，正整数，取值范围[1,1]。		
posCompareIndex	位置比较索引，正整数，取值范围[1,8]。		
pMode	读取输出模式信息，详细信息请参照指令 GTN_SetPosCompareMode 中的参数 3。		
指令返回值	请参照指令返回值列表。		
相关指令	GTN_SetPosCompareMode		
指令示例	例程		

指令 5 GTN_SetPosComparePsoPrm

指令原型	short GTN_SetPosComparePsoPrm(short core,short posCompareIndex, TPosComparePsoPrm *pPrm);
指令说明	设置等间距位置比较输出参数。

© 2021 固高科技版权所有

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	核号，正整数，取值范围[1,1]。		
posCompareIndex	位置比较索引，正整数，取值范围[1,8]。		

pPrm	设置等间距输出间距
	typedef struct
	{
	unsigned long count; // 保留，使用时设置大于0的数
	unsigned short hso; // 保留
	unsigned short gpo; // 保留
	long startSyncPos; // 保留
	long SyncPos; // 输出间距，X、Y轴的合成间距。单位：Pulse
	long time; // 保留
	} TPosComparePsoPrm;		

指令返回值	请参照指令返回值列表。		
相关指令	GTN_GetPosComparePsoPrm		
指令示例	例程		

指令 6　GTN_GetPosComparePsoPrm

指令原型	short GTN_SetPosComparePsoPrm(short core,short posCompareIndex, TPosComparePsoPrm *pPrm);		
指令说明	读取等间距位置比较输出参数。		
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	核号，正整数，取值范围[1,1]。		
posCompareIndex	位置比较索引，正整数，取值范围[1,8]。		
pPrm	读取输出模式信息，详细信息请参照指令 GTN_SetPosComparePsoPrm 中的参数 3。		
指令返回值	请参照指令返回值列表。		
相关指令	GTN_SetPosComparePsoPrm		
指令示例			

指令 7　GTN_PosCompareStart

指令原型	short GTN_PosCompareStart(short core,short posCompareIndex);		
指令说明	启动位置比较输出。		
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	核号，正整数，取值范围[1,1]。		
posCompareIndex	位置比较索引，正整数，取值范围[1,8]。		

© 2021 固高科技版权所有

指令返回值	请参照指令返回值列表。
相关指令	GTN_PosCompareStop
指令示例	例程

指令 8　GTN_GetTerminalPermitEx

指令原型	short GTN_GetTerminalPermitEx(short core, short station, short dataType, short *pPermit, short index, short count);		
指令说明	读取硬件通道输出信号类型。		
指令类型	立即指令。	章节页码	5
指令参数	该指令共有 6 个参数，参数的详细信息如下。		
Core	核号，正整数，取值范围[1,2]。		
station	模块站号。		
dataType	请参考 GTN_SetTerminalPermitEx 的参数 3。		
pPermit	按位读取硬件输出通道信号输出的类型，详细说明请参考 GTN_SetTerminalPermitEx 的参数 3。		
index	读取的起始硬件信号输出通道索引。		
count	读取硬件信号输出通道的个数。		
指令返回值	请参照指令返回值列表。		
相关指令	GTN_SetTerminalPermitEx		
指令示例	例程		

指令 9　GTN_SetTerminalPermitEx

指令原型	short GTN_SetTerminalPermitEx(short core, short station, short dataType, short *pPermit, short index, short count);		
指令说明	设置硬件通道输出信号类型。		
指令类型	立即指令。	章节页码	5
指令参数	该指令共有 6 个参数，参数的详细信息如下。		
Core	核号，正整数，取值范围[1,2]。		
station	模块站号。		
dataType	需要设置控制权的硬件资源类型 MC_GPO　(12)　// 通用数字量输出 MC_HSO　(18)　// 高速 IO 输出		
pPermit	按位设置硬件输出通道信号输出的类型, 从 bit0~bit15 按位表示对应信号类型输出，0：无效，1：有效。 Bit0:通用 GPO Bit1:第一路位置比较输出 Bit2:第二路位置比较输出 Bit3:激光输出 Bit4：PWM 输出 Bit5~bit15:对于 403 模块保留		
index	需要设置控制权的起始硬件通道。		
count	需要设置控制权的硬件通道个数。		

© 2021 固高科技版权所有

指令返回值	请参照指令返回值列表。
相关指令	GTN_GetTerminalPermitEx
指令示例	例程

指令 10　GTN_BufPosComparePsoPrm

指令原型	short GTN_ BufPosComparePsoPrm(short core, short crd, short posCompareIndex, TPosComparePsoPrm *pPrm,short fifo);		
指令说明	缓冲区设置PSO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 5 个参数，参数的详细信息如下。		
core	核号，正整数，取值范围[1,1]。		
crd	插补坐标系索引，取值范围[1,2]。		
posCompareIndex	位置比较索引，正整数，取值范围[1,8]。		

pPrm	typedef struct
	{
	unsigned long count; //保留，使用时大于 0 即可
	unsigned short hso; //保留
	unsigned short gpo; //保留
	long startPosX; //保留
	long startPosY; //保留
	long syncPos; //输出间距，X、Y 轴的合成间距。单位：Pulse
	long time; //保留
	short reserve[20]; //保留
	}TPosComparePsoPrm;		

fifo	插补缓冲区 FIFO，取值范围[0,1]。		
指令返回值	请参照指令返回值列表。		
相关指令	无		
指令示例	无		

指令 11　GTN_BufPosCompareStart

指令原型	short GTN_BufPosCompareStart(short core,short crd, ,short fifo,short posCompareIndex);		
指令说明	缓冲区启动位置比较或PSO。		
指令类型	缓存区指令。	章节页码	5
指令参数	该指令共有 4 个参数，参数的详细信息如下。		
core	核号，正整数，取值范围[1,1]。		
crd	插补坐标系索引，取值范围[1,2]。		
fifo	插补缓冲区 FIFO，取值范围[0,1]。		
posCompareIndex	位置比较索引，正整数，取值范围[1,8]。		
指令返回值	请参照指令返回值列表。		
相关指令	对应前瞻指令 GTN_BufPosCompareStartEx，参数与本指令相同。		

© 2021 固高科技版权所有

指令示例	例程

指令 12　GTN_BufPosCompareStop

指令原型	short GTN_BufPosCompareStop(short core,short crd, ,short fifo,short posCompareIndex);		
指令说明	缓冲区停止位置比较或PSO。		
指令类型	缓存区指令。	章节页码	5
指令参数	该指令共有 4 个参数，参数的详细信息如下。		
core	核号，正整数，取值范围[1,1]。		
crd	插补坐标系索引，取值范围[1,2]。		
fifo	插补缓冲区 FIFO，取值范围[0,1]。		
posCompareIndex	位置比较索引，正整数，取值范围[1,8]。		
指令返回值	请参照指令返回值列表。		
相关指令	对应前瞻指令 GTN_BufPosCompareStopEx，参数与本指令相同。		
指令示例	例程		

指令 13　GTN_SetPosCompareMultiPsoPrm

指令原型	short GTN_SetPosCompareMultiPsoPrm (short core,short posCompareIndex, TPosCompareMultiPsoPrm *pPrm);		
指令说明	设置PSO多间距参数。		
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	核号，正整数。		
PosCompareIndex	位置比较索引，正整数，取值范围[1,8]。		

pPrm	typedef struct
	{
	unsigned long count; //保留，使用时大于即可
	unsigned short hso; //保留
	unsigned short gpo; //保留
	long startPosX; //保留
	long startPosY; //保留
	long time; //保留
	short multiNumber; //存在几种间距
	short pad1;
	long syncPosArray[256]; //按照数组排列每种间距合成长度。单位：Pulse
	} TPosCompareMultiPsoPrm;		

© 2021 固高科技版权所有

	
	例如存在两种间距 S1，S2 则
	multiNumber = 2;
	syncPosArray[0] = S1;
	syncPosArray[1] = S2;
	该指令需要在 GTN_SetPosCompareMode 之后调用，与 GTN_SetPosComparePsoPrm 单间距不能混用。

指令返回值	请参照指令返回值列表。
相关指令	无
指令示例	无

© 2021 固高科技版权所有