**TwinCAT 3以FeedTable的形式实现NCI插补**

|  |  |
| --- | --- |
|  | 作者：简自豪  职务：技术工程师  邮箱：Yosef45545@163.com  日期：2023-11-7 |
| **摘 要：**  笔者作为倍福用户，在学习倍福的过程中发现虚拟学院中并没有文档是关于TwinCAT 3如何使用FeedTable的形式来实现NCI插补的，因此，借这个客户投稿的机会来补充一下TwinCAT 3 NCI中该功能的一些文档，本文参考了TwinCAT 2以及infosys上的FeedTable资料进行了编写。  FeedTable实际上就是使用自定义的点位段表来实现NCI的插补，而非通过执行G代码文件。该形式是非常便于前期调试NCI时的一种方法，得益于其能够在程序当中直接写入插补指令来达到更改插补路径的效果，从而避免了频繁的修改系统中的G代码文件。 | |
| **附 件：**   |  |  |  | | --- | --- | --- | | 序 号 | 文件名 | 备注 | | 1 | FeedTableNci.tnzip | 例程 | |  |  |  | |  |  |  | |  |  |  | | |
| **历史版本：**   |  |  |  | | --- | --- | --- | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |  |  |  | | |
| **免责声明：**  我们已对本文档描述的内容做测试。但是差错在所难免，无法保证绝对正确并完全满足您的使用需求。本文档的内容可能随时更新，如有改动，恕不事先通知，也欢迎您提出改进建议。 | |
| **参考信息：** | |

目 录

[1. 软硬件版本、FeedTable执行流程 3](#_Toc151649302)

[1.1. 倍福Beckhoff 3](#_Toc151649303)

[1.1.1. 控制器硬件 3](#_Toc151649304)

[1.1.2. 控制软件 3](#_Toc151649305)

[1.2. FeedTable执行流程 3](#_Toc151649306)

[2. 指令介绍 4](#_Toc151649307)

[2.1. FeedTable相关指令介绍 4](#_Toc151649308)

[2.1.1. 结构体NCI\_SingleEntry 4](#_Toc151649309)

[2.1.2. 指令准备功能块FB\_NciFeedTablePreparation 4](#_Toc151649310)

[2.1.3. 执行段表插补指令 FB\_NciFeedTable 5](#_Toc151649311)

[2.2. NCI插补通道相关指令介绍 5](#_Toc151649312)

[2.2.1. 组合通道CfgBuildExt3DGroup、解散通道CfgReconfigGroup 5](#_Toc151649313)

[2.2.2. 插补通道速比ItpSetOverridePercent、插补通道状态ItpGetStateInterpreter 6](#_Toc151649314)

[2.2.3. 插补指令启停ItpStartStopEx 6](#_Toc151649315)

[3. 例程简介 6](#_Toc151649316)

[3.1. 准备工作 6](#_Toc151649317)

[3.1.1. 库文件添加 6](#_Toc151649318)

[3.1.2. 插补通道硬件配置 6](#_Toc151649319)

[3.2. 程序介绍 8](#_Toc151649320)

[3.2.1. A\_NcPTP 8](#_Toc151649321)

[3.2.2. A\_NciChannel 8](#_Toc151649322)

[3.2.3. A\_LineTable 9](#_Toc151649323)

[3.3. 逻辑执行 11](#_Toc151649324)

[3.3.1. HMI 11](#_Toc151649325)

[3.3.2. 执行NCI插补 14](#_Toc151649326)

# 软硬件版本、FeedTable执行流程

## 倍福Beckhoff

### 控制器硬件

编程电脑本地仿真模拟

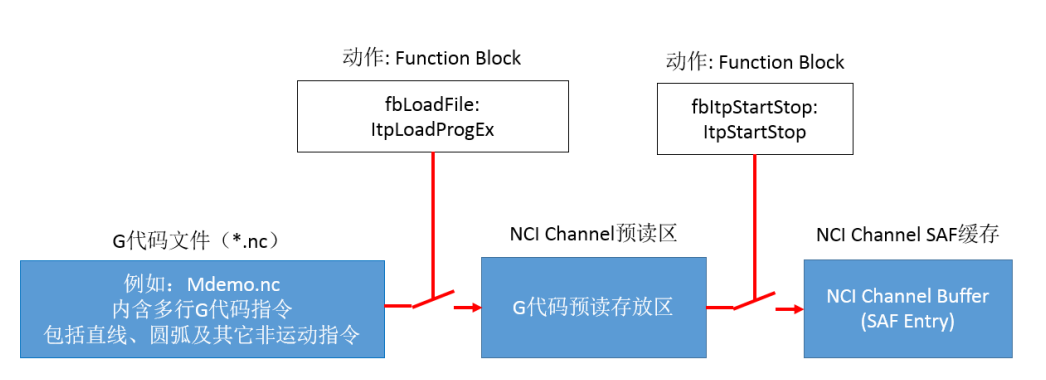
Win10企业版 64位

### 控制软件

TwinCAT 3.1 Build 4024.44版本

## FeedTable执行流程

最快理解FeedTable插补流程的方法便是拿其和G代码插补方式去做对比，我们可以用两张流程图来展示，首先是G代码的形式：



可以看到G代码插补的方式源头是提前编写好的G代码文件，里面包含了直线、圆弧插补以及其他非运动指令例如M参数、道具补偿参数等；接着，通过G代码功能块将其预读到NCI插补通道的存放区，最后通过启停功能块将其写入NCI插补通道SAF的缓存区，同时控制G代码的执行与停止。

接着我们再来看下FeedTable方式的流程图：

图示

描述已自动生成

可以看到FeedTable方式的插补与G代码最大的区别在于NCI指令在写入SAF缓存之前，指令的运转全是在PLC程序当中，而并没有NCI插补通道的介入。NCI指令的源头是PLC程序中的各个数据块，同样可以包括直线、圆弧插补以及其他非运动控制指令；接着通过FeedTable独有的指令填充功能块，将所有的指令存放在一个段表中，这个段表也可以理解为一个预读区，但和G代码的预读取差别在于，FeedTable预读的段表是存放在PLC程序中，而G代码则是NCI插补通道，也正因如此，FeedTable方式的插补，在修改NCI指令上比G代码形式更加灵活多变；最后通过功能块将该段表的指令写入SAF的缓存进行执行。

以上便是两种插补方式的区别。

# 指令介绍

## FeedTable相关指令介绍

在2.1章节中介绍的指令都可以在infosys中查询到相关说明：

<https://infosys.beckhoff.com/content/1033/tcnci/958083979.html?id=5205927589624402180>

### 结构体NCI\_SingleEntry

NCI\_SingleEntry是一个自定义的结构体，创建该结构体是为了便于统一直线和圆弧插补的准备数据，在填充插补指令的过程中总是会从该结构体中提取数据。因此，该结构体只是便于填充数据而创建的结构体，而并不是FeedTable中必须使用的一环。

文本

中度可信度描述已自动生成

### 指令准备功能块FB\_NciFeedTablePreparation

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

该功能块用于填充插补指令，nEntryType引脚用于定义单次指令的类型。插补指令的类型繁多，但最常用的为两种，分别是ST\_NciGeoLine直线插补、以及ST\_NciGeoCirclePlane平面圆弧插补，除此之外，还有些类型在实际使用的过程中也需要注意，分别是ST\_NciGeoCircleCIP空间圆弧插补以及通常在首位指令使用的ST\_NciGeoStart和ST\_NciEndOfTables。引脚则是插补指令的数据来源，在本例程中，数据的来源便是上文提到的自定义结构体NCI\_SingleEntry。而最终数据将会填充到指令段表，也就是stFeedGroupTable这个引脚当中。需要注意的是，该段表能够填充的指令数量并非无限，一张表中最多能够填充100条指令，若在填充满后继续向表中填写数据，功能块便会报错，需新建一张段表或是清空当前段表才能继续填写。清空段表则是通过置位bResetTable引脚，其与bResetAll的区别在于bResetAll还会令一些模态参数恢复默认值。

### 执行段表插补指令 FB\_NciFeedTable

表格

中度可信度描述已自动生成

该功能块用于执行段表中填充的插补指令，触发该功能块后stNciToPlc引脚上链接的对应插补通道便会开始执行stFeedGroupTable段表中的指令。bLogFeederEntries如果置True，则代表功能块在TwinCAT\Mc\Nci这个路径下写入了'PlcItpFeed.log'的日志文件，它包含通过ADS发送到NC内核的所有条目。

当段表中的所有指令被发送到NC内核时该引脚便会置位，而当所有指令被执行完成且检测到段表中的ST\_NciEndOfTables类型指令后，bChannelDone引脚便会置位。

## NCI插补通道相关指令介绍

### 组合通道CfgBuildExt3DGroup、解散通道CfgReconfigGroup

文本

描述已自动生成

该功能块若大家经常使用NCI插补的话应该是再熟悉不过了，通过NC轴的ID将一个个单独的PTP轴组合进nGroupId引脚对应的插补通道当中。

文本, 应用程序

描述已自动生成

如需将插补通道的轴重新转换成独立的NC PTP轴，则可以使用该功能块来解散插补通道。

### 插补通道速比ItpSetOverridePercent、插补通道状态ItpGetStateInterpreter

图形用户界面, 文本, 应用程序

描述已自动生成

插补通道同样也有速比设置，使用ItpSetOverridePercent这个Function来设置插补速比，需要注意，该速比必须在插补通道中进行配置或是在程序中进行定义，系统并不会默认设置插补通道的速比为100%。

图形用户界面, 文本, 应用程序

描述已自动生成

在轴进行插补的过程中往往会需要一些标志位来对插补通道的状态进行判断，以便于观察插补逻辑的正常运行以及执行一些其他的工艺指令，因此，可以通过ItpGetStateInterPreter这个Function来实时获取插补通道的当前状态。

### 插补指令启停ItpStartStopEx

图形用户界面, 文本, 应用程序

描述已自动生成

ItpStartStopEx功能块在使用G代码文件方式进行插补时，是用于启动/暂停G代码执行的功能块；而在FeedTable的方式中，启动插补是通过FB\_NciFeedTable功能块，而ItpStartStopEx的作用仅限于在插补指令执行的过程中对其进行暂停。

# 例程简介

## 准备工作

### 库文件添加

在使用FeedTable前，首先需要在References文件夹中添加需要用到的3个库文件，详见下图。

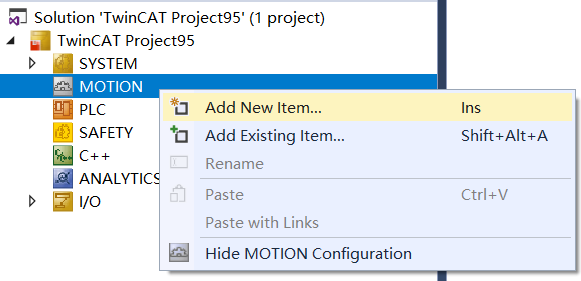
文本

描述已自动生成

### 插补通道硬件配置

无论是使用G代码还是FeedTable的形式，都需要先在Motion中创建一根插补通道才能在程序中使用功能块将轴添加进插补轴组。

首先，右键Motion选项卡，点击Add New Item，创建一个NC task。

图形用户界面, 文本, 应用程序

描述已自动生成

随后右键Axes，点击Add New Item，添加三根插补轴X、Y、Z。

图形用户界面, 应用程序

描述已自动生成图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

最后，右键NC-Task 1SAF再次点击添加新项，选择NC Channel（for Interpolation）创建一根插补通道。

图形用户界面, 应用程序

描述已自动生成图形用户界面, 应用程序

描述已自动生成

上述项目创建完成后我们便完成了插补通道的硬件配置。

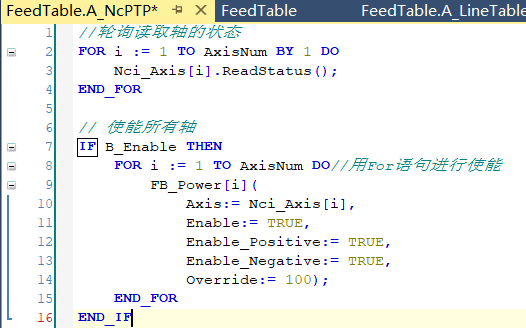
图形用户界面, 文本, 应用程序

描述已自动生成

## 程序介绍

本例程较为简单，结合了一个简单的HMI界面和XYZ Scope，可以演示FeedTable方式下的直线和圆弧插补，主程序共分成了3个Action。

### A\_NcPTP



该Action中的程序非常简单，用于对所有参与插补的轴进行使能。

### A\_NciChannel

前文说到，无论是通过G代码还是FeedTable的形式进行插补，都需要提前进行插补通道的配置。在完成3.1章节的硬件配置后，接下来可以通过PLC程序将PTP轴加入插补通道，而使用到的便是第2章节中介绍的几个功能块。

首先使用两个获取ID的Function获取插补通道的轴组ID以及通道ID，以便后续使用。

文本

描述已自动生成

获取ID完成后需要设定插补通道的速比。



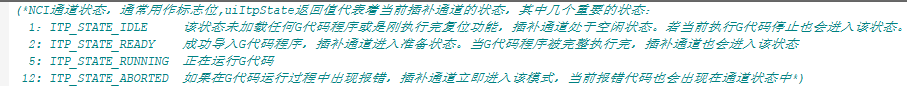
设定完成后，便可以调用功能块将所有的PTP轴加入插补通道进行使用了。

文本

描述已自动生成

除此之外，为了表现出FeedTable方式的插补与G代码的不同之处，笔者还监控了NCI的通道状态。该状态在G代码方式的插补下，会根据G代码的装载、执行而变化为不同的状态机，但FeedTable形式并未经过此流程，因此，该状态机的值始终为1：





### A\_LineTable

此Action是主程序中最重要的一环，整个FeedTable的流程都将在整个Action中执行完毕。

首先是NCI插补指令的填写，程序中声明了一个名为Nci\_Command的变量，其类型是第2章节中介绍的NCI\_SingleEntry的结构体，在该变量中我们可以填写单次的NCI插补指令：



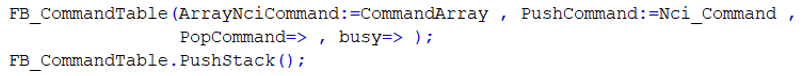
单次的插补指令填写完成后我们便需要填充到ST\_NciFeedGroupTable的指令段表中，这时候我们就需要用到FB\_NciFeedTablePreparation功能块，例程中单次指令只设定成了两种常用的形式，分别是直线插补以及平面圆弧插补，通过结构体中的iPlane的值来进行区分：



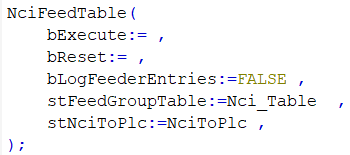
图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

填充的过程中有两行与队列相关的程序是用来在HMI中便于观察填写的指令段表：



在所有的指令编写完成后，便可以调用并触发FB\_NciFeedTable功能块将其写入SAF缓存并执行：



在段表中的指令执行完成后，储存在段表中的指令并不会自动清除，若段表中上一次的指令仍然保留，再次执行该功能块会重新从段表的首行命令开始执行，并不会直接执行新写入的指令。因此，想要NCI继续新的插补指令，必须复位段表：



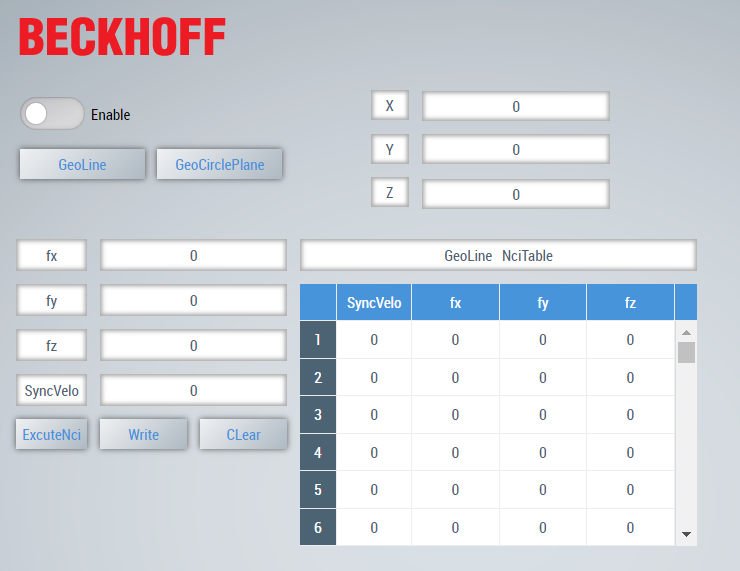
若在填写指令的过程中发现指令填写错误，则需要手动清空段表重新填写：

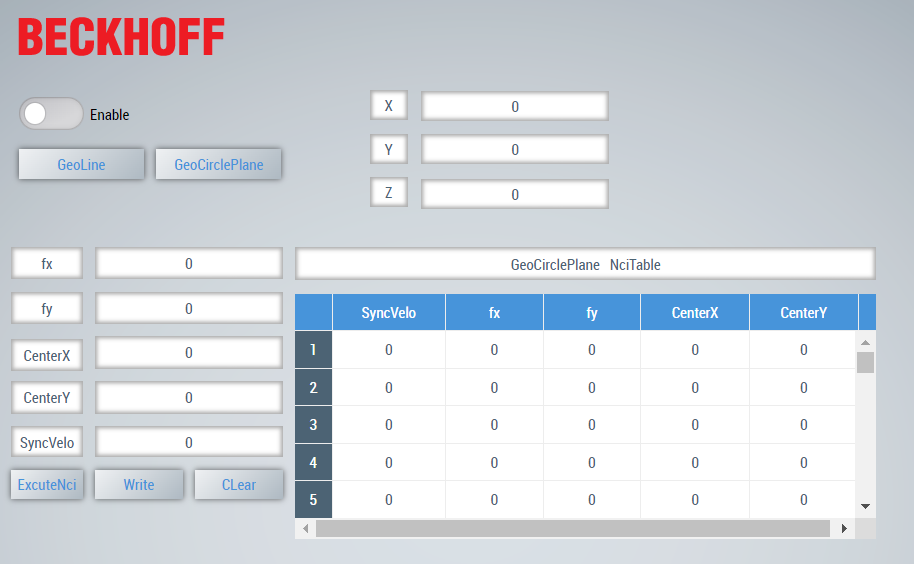


## 逻辑执行

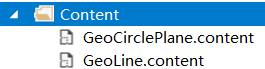
### HMI

在执行逻辑之前，我们先来看一下例程中HMI界面的使用，该界面是为了便于写入插补指令而制作的界面，由两个子界面构成：





首先，子页面的切换由左上方的两个button来进行切换，切换的同时程序会自动判断写入的指令是直线插补形式还是平面圆弧插补形式：

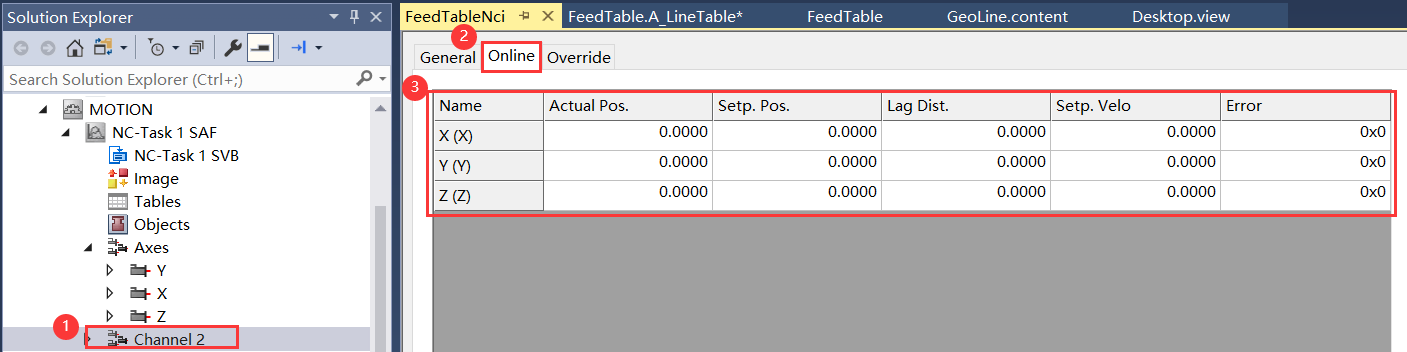


图标

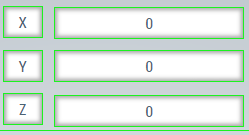
描述已自动生成

左上角的ToggleSwitch控件用于轴的使能，使能的同时，所有PTP轴将会被加入到NCI插补通道中，可以通过NCI通道中是否出现三根PTP轴来判断轴是否加入成功：

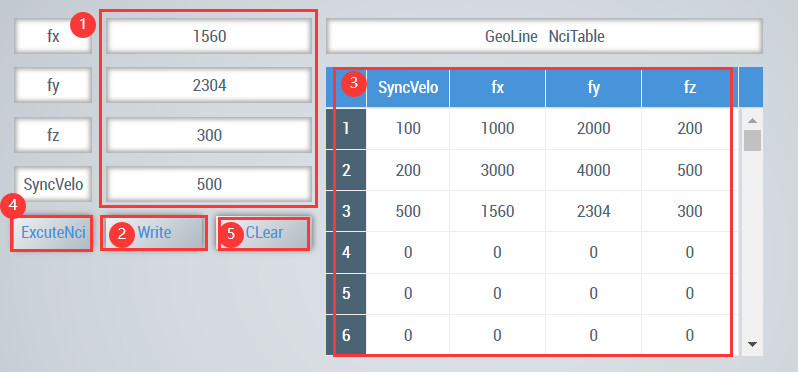




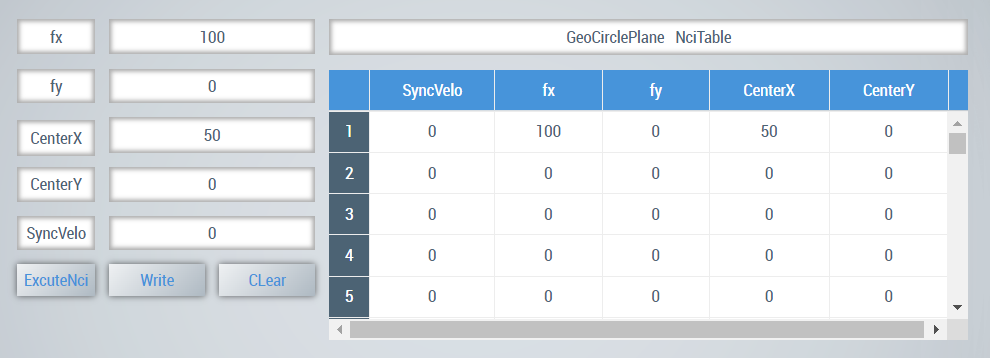
右侧的X Y Z 3个TextBlock可以实时观察三根插补轴所在的ActPos：



界面的下半部分便是填写NCI指令的地方，我们可以通过左侧4个TextBlock填写单次NCI指令的参数，随后点击Write进行写入，写入完成后可以看到右侧的指令队列便会显示出刚刚填写的数据。在所有的指令填写完成后，我们便可以点击ExcuteNci触发插补，在插补完成后便可以点击Clear清空段表：



而平面圆弧插补和直线插补的区别仅在填写的部分参数不同，使用流程上并无区别：



### 执行NCI插补

现在我们可以激活硬件配置来看一下FeedTable的效果。

首先测试直线插补，我们填写需要执行的参数，同时，使用XYZ Scope来看一下其执行效果。

填写的参数是一个正方形的4个点：



最终的执行效果：

图表

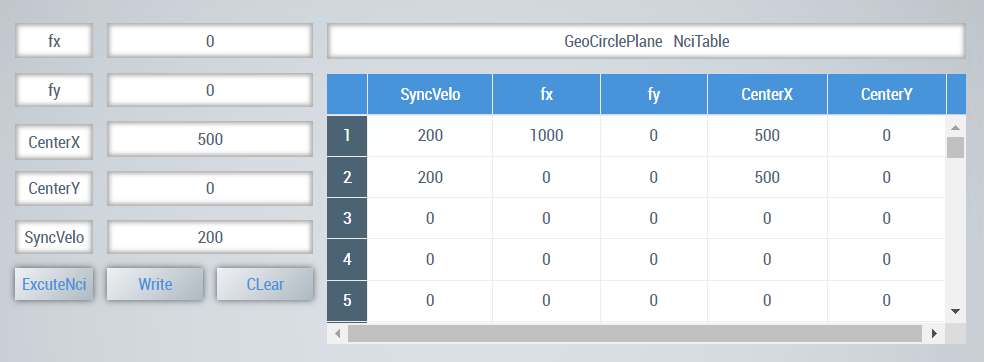
描述已自动生成

同时，我们可以通过功能块中的BlockNo引脚知道当前正在执行第几行NCI指令：



执行完成后，指令便会自动清除，点击Clear清空HMI中的段表，便可以继续填写后续需要执行的指令。

接下来我们再来测试下平面圆弧插补。我们填写平面圆弧插补指令，让其在Scope中画出一个圆形：



可以发现平面圆弧插补执行也成功。

图示

描述已自动生成

**上海（ 中国区总部）**

中国上海市静安区汶水路 299 弄 9号（市北智汇园）

电话: 021-66312666

**北京分公司**

北京市西城区新街口北大街 3 号新街高和大厦 407 室

电话: 010-82200036 邮箱: beijing@beckhoff.com.cn

**广州分公司**

广州市天河区珠江新城珠江东路32号利通广场1303室

电话: 020-38010300/1/2 邮箱: guangzhou@beckhoff.com.cn

**成都分公司**

成都市锦江区东御街18号 百扬大厦2305 室

电话: 028-86202581 邮箱: chengdu@beckhoff.com.cn

|  |  |
| --- | --- |
| 请用微信扫描二维码  通过公众号与技术支持交流 | 倍福官方网站：  https://www.beckhoff.com.cn  在线帮助系统：  https://infosys.beckhoff.com/index\_en.htm |
| 倍福虚拟学院：  https://tr.beckhoff.com.cn/ |
| 招贤纳士：job@beckhoff.com.cn  技术支持：support@beckhoff.com.cn  产品维修：service@beckhoff.com.cn  方案咨询：sales@beckhoff.com.cn |
|  |