固高科技
GOOGOL TECHNOLOGY

基于 Labview 编程的 GSN 卡

初级入门手册

目 录

第 1 章 概述

1.1 术语与缩写解释

术语、缩写	解释
FPGA	即现场可编程门阵列。
PCI-E	外设组件互连标准局部总线。
gLink-IIA	主卡中的第 1 个端口（对应处理器的 core1）
gLink-IIB	主卡中的第 2 个端口（对应处理器的 core2）
Core1	处理器中的第 1 个内核
Core2	处理器中的第 2 个内核

1.2 简介

固高公司生产的 GSN 系列运动控制器，可以实现高速的点位运动控制。其核心由双核处理器 和 FPGA 组成，可以实现高性能的控制计算。它适用领域广泛，包括机器人、数控机床、木工机械、印刷机械、装配生产线、电子加工设备、激光加工设备以及 PCB 钻铣设备等。

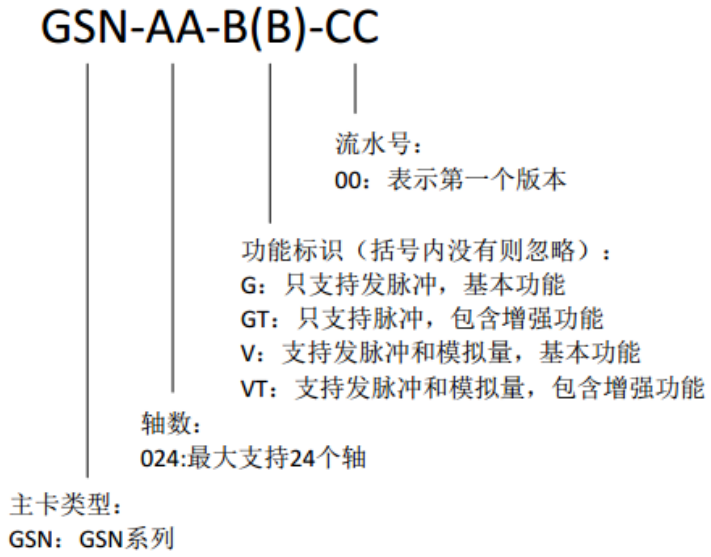
GSN 系列运动控制器以工控机及兼容机为主机，提供标准的 PCI-E 总线接口产品。运动控制器提供 C 语言等函数库和 Windows 动态链接库，实现复杂的控制功能。用户能够将这些控制函数与自己控制系统所需的数据处理、界面显示、用户接口等应用程序模块集成在一起，建造符合特定应用需求的控制系统，以适应各种应用领域的要求。

1.3 型号说明

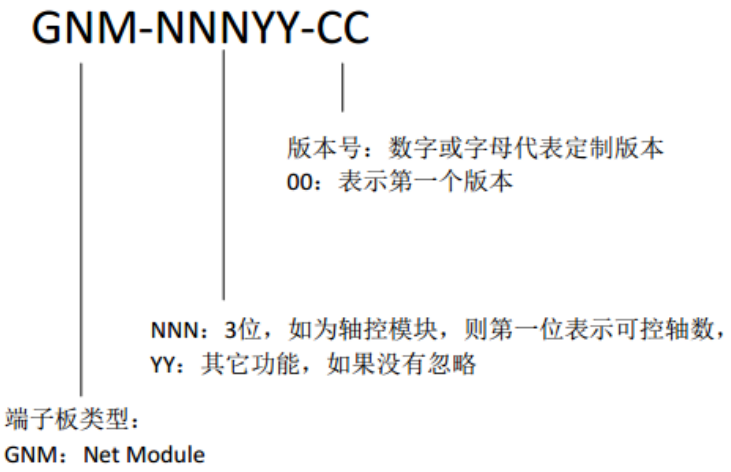
运动控制器包含两个部件：运动控制卡和端子板。每一种运动控制卡和端子板都有自身的型号，不同的型号对应不同的功能，根据实际需要选择相应的运动控制卡。

以下是主卡、轴模块、通讯线缆的型号说明。

主卡：


GSN-AA-B(B)-CC
流水号：
00：表示第一个版本
功能标识（括号内没有则忽略）：
G：只支持发脉冲，基本功能
GT：只支持脉冲，包含增强功能
V：支持发脉冲和模拟量，基本功能
VT：支持发脉冲和模拟量，包含增强功能
轴数：
024:最大支持24个轴
主卡类型：
GSN：GSN系列

轴模块：


GNM-NNNYY-CC
版本号：数字或字母代表定制版本
00：表示第一个版本
NNN：3位，如为轴控模块，则第一位表示可控轴数，
YY：其它功能，如果没有忽略
端子板类型：
GNM：Net Module

通讯线缆：


GN-AAAA-AAAA-CC
线缆长度：
1M5：1.5米
10M：10米
线缆两端接口形式：
DB9F：DB9母头
DB9M：DB9公头
RJ45：RJ45水晶头
GoogolTech Net

第 2 章 快速使用

2.1 开箱检查

打开包装前，请先查看外包装标明的产品型号是否与订购的产品一致。打开包装后，请先戴上固高科技给您配置的防静电手套，然后按照《装箱清单》或订购合同仔细核对配件是否齐备。检查运动控制器的表面是否有机械损坏，如果运动控制器表面有损坏，或产品内容不符合，请不要使用，立即与固高科技或经销商联系。

GSN 系列运动控制器产品清单（详细请参考《装箱清单》）：

(1) GSN-A00 控制器主卡，数量 1 块；

(2) GNM 网络端子板模块，数量 N 块（根据客户需求配置）；

(3) 9PIN 连接电缆，数量 N+1 条（由端子板数量决定，一块端子板配 2 条线缆）；

(4) 配套光盘，数量 1 张；

(5) 保修卡，数量 1 张；

(6) 合格证，数量 1 张；

2.2 安装场所

控制器须远离大功率，强电磁干扰的商用电器和环境。

2.3 准备工作

在安装之前，请先准备好以下物品：

(1) 具有PCI-E接口以及安装了Windows操作系统(Windows 98, Windows XP, Win 7 均可）的计算机。

(2) +24V 直流电源（不允许使用+12V 直流电源代替）。

(3) 步进电机或伺服电机。

(4) 驱动器和驱动器电源。

(5) 端子板轴信号接口到驱动器轴接口之间的连接线缆(需要您根据驱动器的型号，制作与运动控制器端子板轴信号相匹配的线缆)。

(6) 原点开关、正/负限位开关(用户根据系统需要自行选择)。

(7) 万用表。

2.4 安装步骤

2.4.1 步骤 1：将运动控制器插入计算机

(1) 关断计算机电源。

(2) 打开计算机机箱，选择一条空闲的 PCI-E 插槽，用螺丝刀卸下对应插槽的挡板条。

(3) 将运动控制器可靠地插入该槽。

(4) 拧紧其上的固定螺丝。

(5) 卸下临近插槽的一条挡板条，用螺丝将转接板固定在机箱该插槽上。

(6) 盖上计算机机盖，打开 PC 电源，启动计算机。

2.4.2 步骤 2：安装运动控制器驱动程序

硬件安装好后，启动计算机，windows 自动检测到运动控制器，打开设备管理器，如图 2-1 所示。

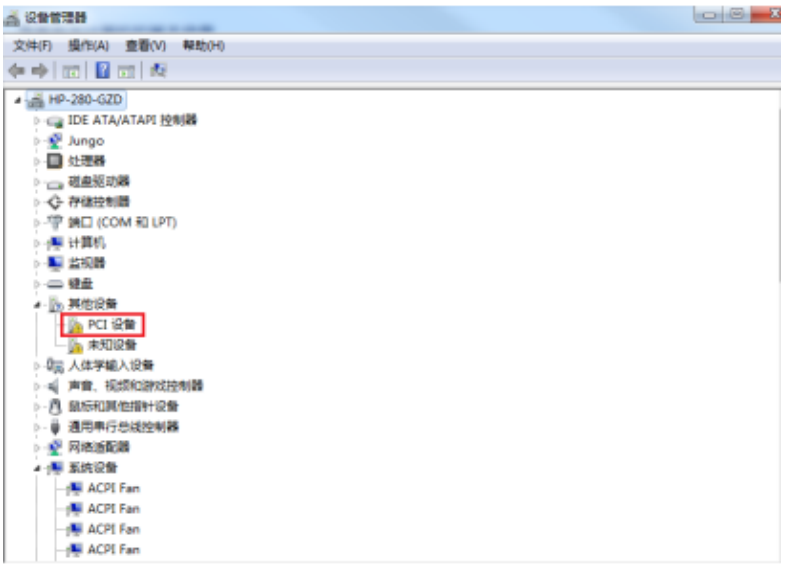


图 2-1

将产品的配套光盘放入光驱。如果没有配置光驱的工控机可以将光盘中的驱动程序先拷贝到电脑中(驱动程序在光盘中的路径为 \chinese\windows\Driver

或者\english\windows\Driver)。

选中”PCI 设备”，点击鼠标右键。如图 2-2 控制器驱动程序安装界面及图所示，点击“更新驱动程序软件(P)…”。

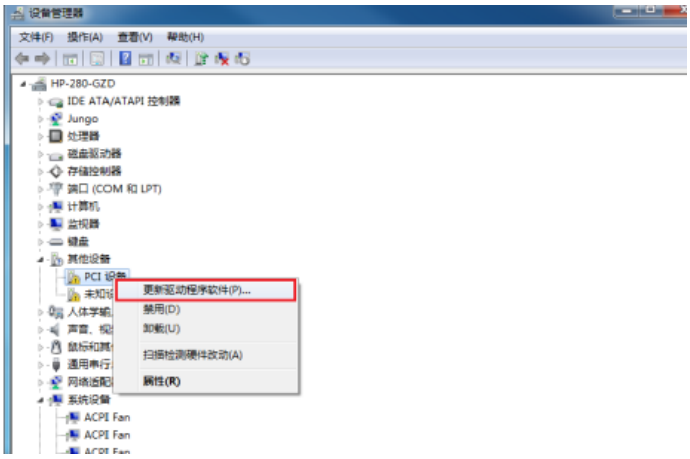


图 2-2

选择“浏览计算机以查找驱动程序软件(R)”。界面如下图所示。

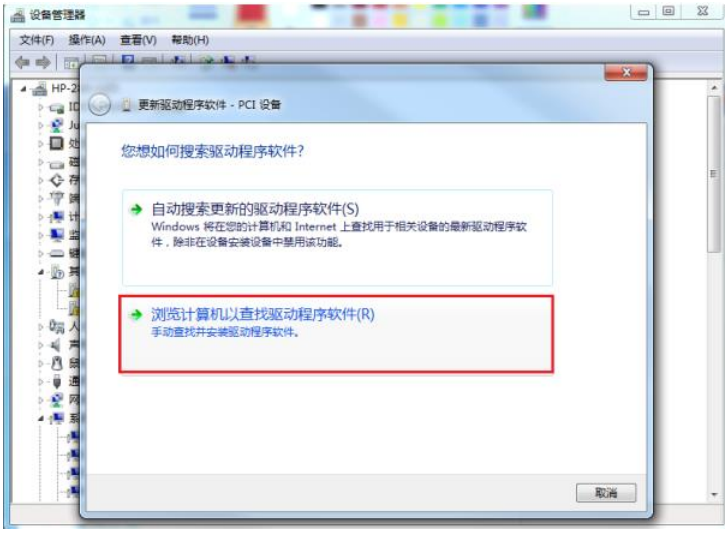


图 2-3

点击“浏览”，选择驱动程序所在路径。 点击下一步。

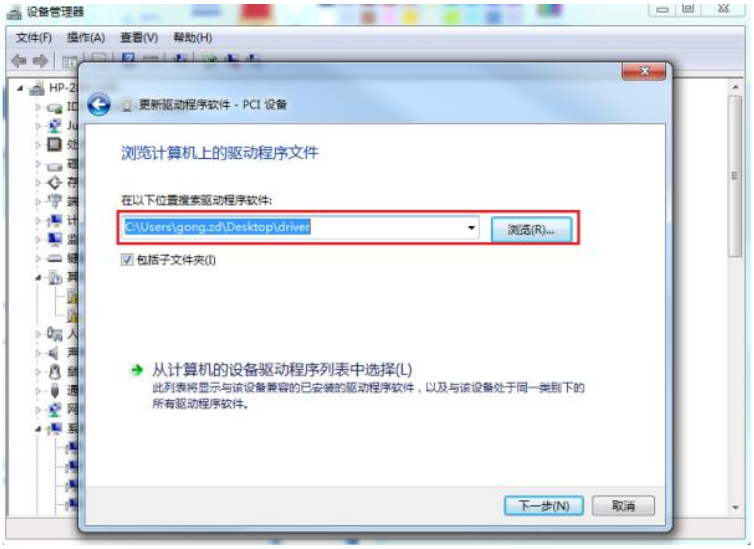


图 2-4

如图 2-5 所示，勾选 “始终信任来自”固高科技(深圳)有限公司的软件(A)”， 点击安装。

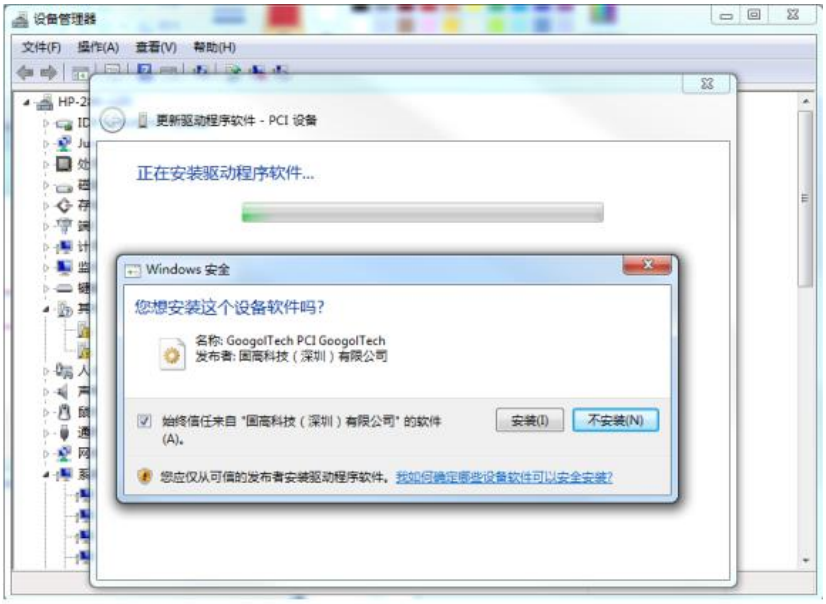


图 2-5

安装成功后，打开设备管理器，可以看到驱动已经安装成功，如图 2-6 所示。

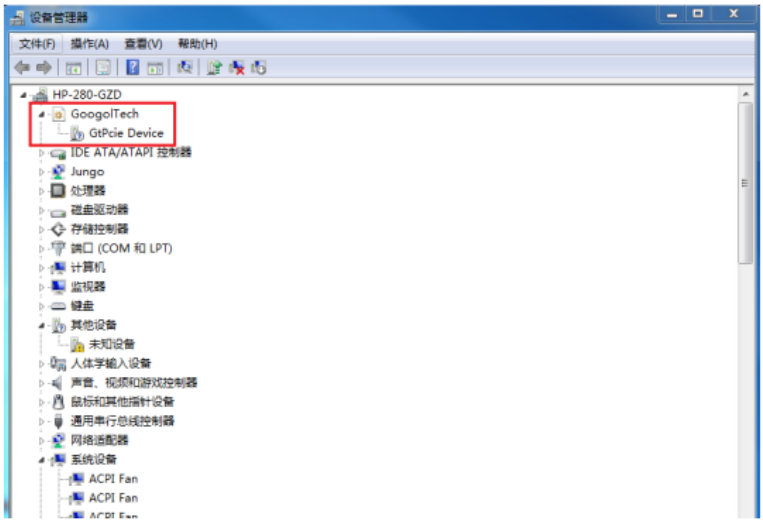


图 2-6

2.4.3 步骤 3：建立主机和运动控制器的通讯

使用光盘里附带的 MotionStudio 系统调试软件，测试主机是否和运动控制器建立了联系；详细的操作过程，请参照 MotionStudio 的帮助文档。

如果 MotionStudio 能正常工作，证明运动控制器通讯正常。否则会提示错误信息“打开板卡失败”，确定问题所在，排除故障后重新测试。

2.4.4 步骤 4：连接电机和驱动器

在驱动器没有与运动控制器连接之前，连接驱动器与电机。用户必须仔细地阅读驱动器的说明书，正确连接。按照驱动器说明书的要求测试驱动器与电机，确保其工作正常。

2.4.5 步骤 5：连接运动控制器和端子板

关闭计算机电源，取出产品附带的电缆。连接主卡的 gLink-IIA/gLink-IIB 和端子板的 CN1/CN2（其中 CN1 和 CN2 可以随意配置为一个输入和一个输出），如下图所示。如果不需要构成环形网络（冗余备份）只需要连接 gLink-IIA 与端子板接口，网络即可正常工作；如果需要构成环网，则需要把最后一个模块的空余接口接到 gLink-IIB，从而构成环形网络拓扑结构。

在连接模块时，必须有一个端口和主卡的 gLink-IIA 接口连接才能正常工作。推荐使用 gLink-IIA 接口作为起始连接端口，最后一个轴模块的输出端口连接在主卡的 gLink-IIB，形成闭环通讯模式。如图 2-7 所示。

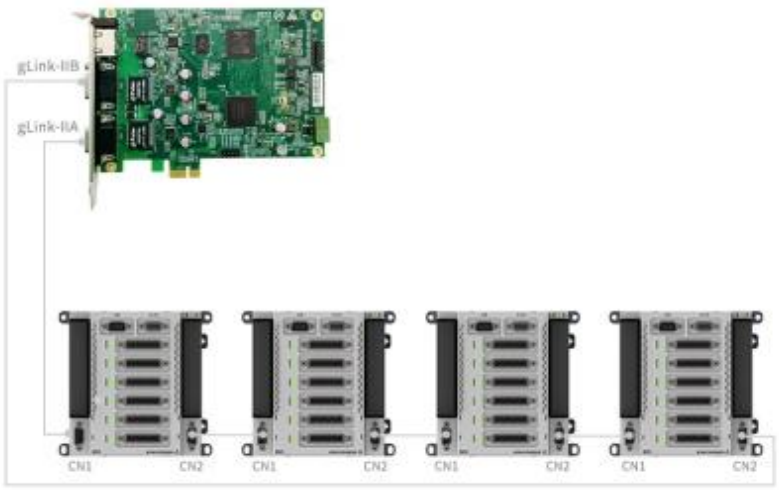


图 2-7

2.4.6 步骤 6：连接驱动器、系统输入/输出和端子板

端子板上标有“24”的端子接外部电源+24V，标有”0v”的接外部电源地，标有” PE”的保护地。用户可以根据需求连接电源。

2.4.7 控制器状态检测

通过主卡左侧的 4 个指示灯判断 处理器、 FPGA 和网络工作状态

(1) 1：闪烁时，表示 处理器 工作正常；其它状态表示处理器工作不正常。

(2) 2：闪烁时，表示 FPGA 工作正常；其它状态表示 FPGA 工作不正常。

(3) A：快速闪烁时(周期为 100ms)，表示 gLink-IIA 口与模块正常通信，所接模块正常工作；不亮时，说明 gLink-IIA 口未接模块或者 gLink-IIA 口通信失败。

(4) B：快速闪烁时(周期为 100ms)，表示 gLink-IIB 口与模块正常通信，所接模块正常工作；不亮时，说明 gLink-IIB 口未接模块或者 gLink-IIB 口通信失败。

2.5 软件调试

电脑桌面上打开 MotionStudio，如下图所示。

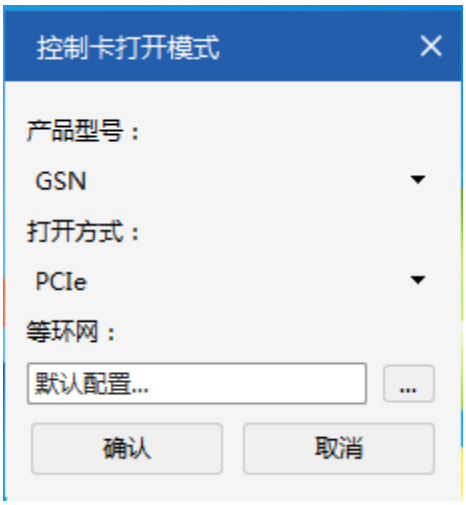


图 2-8

产品型号：GSN，打开方式：PCIe，等环网：默认配置，点击确认，进入到如图 2-9 的软件界面。

图 2-9

在导航窗口可以看到，实际控制卡所接的轴模块情况，这里 GSN 卡接了一个 GNM-601-A00（关于轴模块可以了解对应的手册和说明书），窗口上显示的是核 1 接了一个 6 轴模块，与实际情况相同。菜单栏点击电气调试，导航窗口点击“1 轴”，然后再点击出现的窗口中的“寻找模块”按钮，软件就会检测到模块上所接的轴情况，如图 2-10 所示。

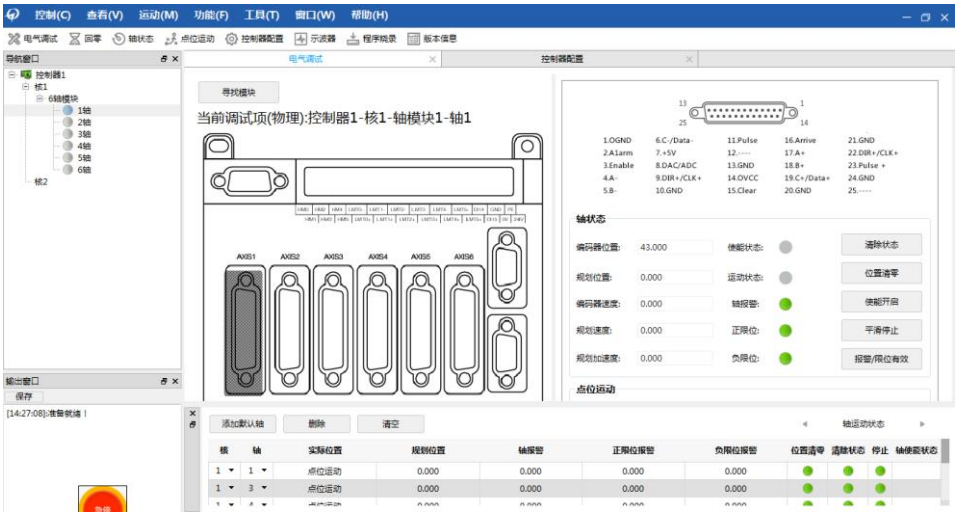


图 2-10

在左侧可以看到轴接口的引脚定义以及轴状态，在调试时可作为参考。同时还有：清除状态、位置清零、使能开启、平滑停止、报警/限位有效，这几个按钮，调试时可以进行相关操作。

点击菜单栏的控制器配置，进入控制器配置界面。

图 2-11

控制器配置界面中，可以对控制器进行一系列的配置，包括：轴 IO、停止类型、当量变换、编码器、脉冲输出、闭环控制、闭环输出、数字输入、数字输出。在轴 IO 配置中，可以为每个轴的轴报警、正负限位索引号按照使用需求来配置。这里使用的是六轴模块，在调试时将各轴报警和限位置为 none，如图所示。

编码器配置页面，可以进行编码器反转的配置。待一些基本配置配置完后，可以将配置加载到控制器中和写入配置文件。如下图所示。也可以从文件和控制器进行加载。

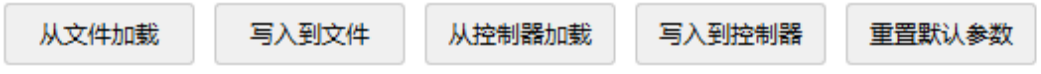


图 2-12

调试软件还可进行点位运以及回零，菜单栏点击点位运动，出现点位运动窗口。可以选择核号、轴号、步长、速度、加速度、减速度等，然后就可以启动点位运动。

图 2-13

第 3 章 基于 Labview2014 软件对 GSN 卡编程

3.1 导入共享链接库

打开 Labview 软件（以 Labview2014 为例），点击工具—导入—共享库。出现导入共享库的窗口。

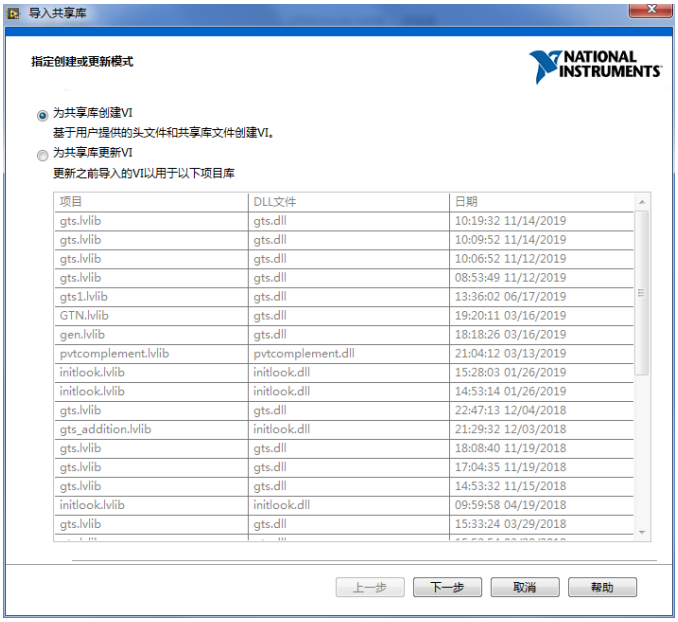


图 3-1

选择为共享库创建 VI，点击下一步。在“选择共享库及头文件”中，按照链接库所在位置，添加 gts.dll 文件，添加后，头文件会自动显示出来。

注意，在添加库时，要区分开 32 位库和 64 位库，32 位的系统就要选择 32 位的库，64 位的库就要选择 64 位的库，不能选错，否则在编程运行的时候程序会报错。

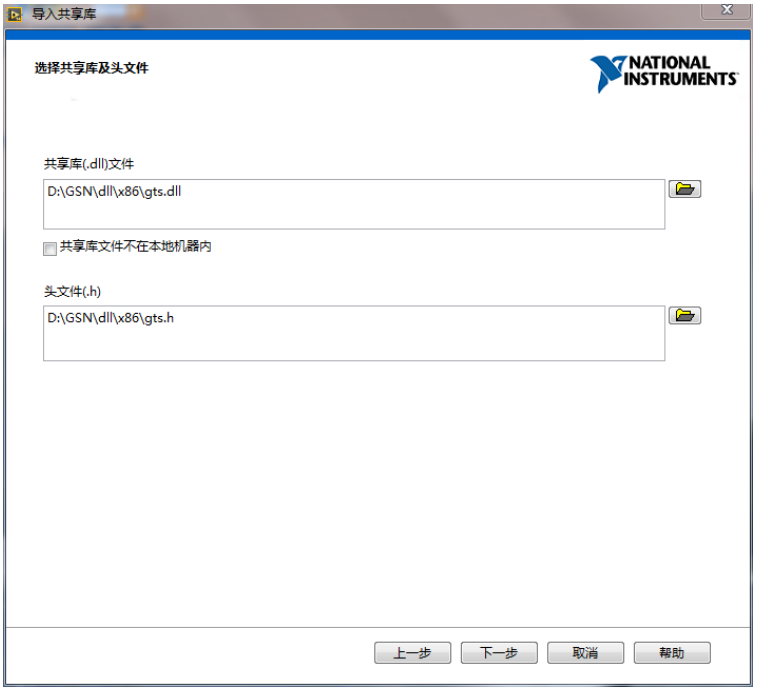


图 3-2

然后继续点下一步，这时需要稍微等待一会，软件会不断的寻找识别共享库中的函数，全部识别后，选择“全部勾选”，再点击下一步。如图 3-3 所示。

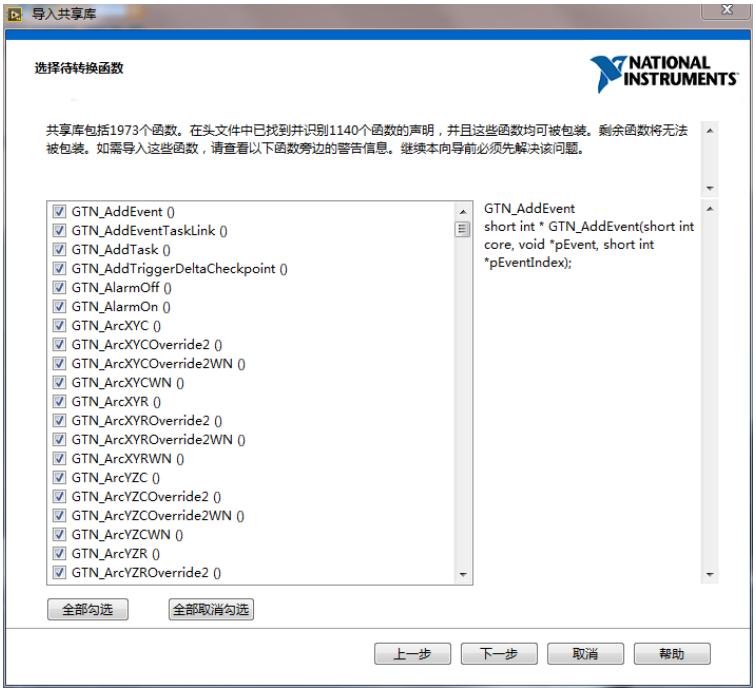


图 3-3

选择错误处理模式里，下拉列表里选择“简易错误处理”，在点击下一步，等候一会，界面加载完后，继续点击下一步，最终会出现一个“正在生成”的界面，稍等一段时间就会生成成功。如图 3-4、图 3-5、图 3-6 所示。

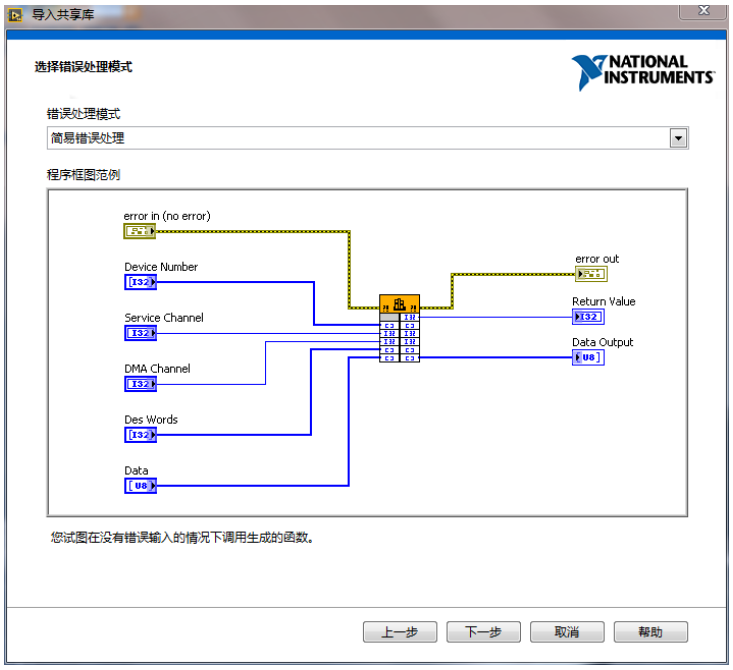


图 3-4

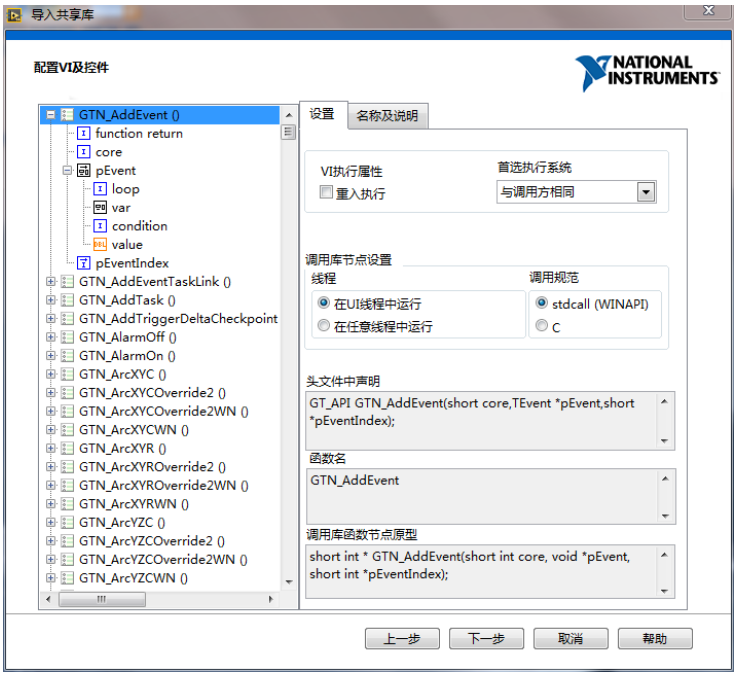


图 3-5

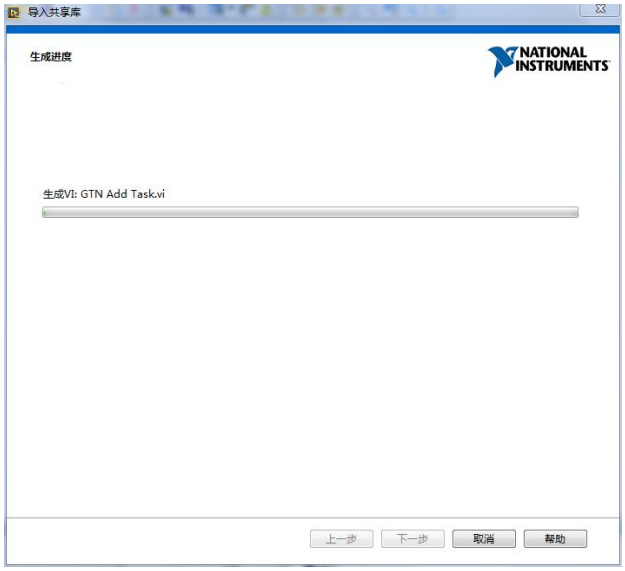


图 3-6

3.2 调用 VI 库

打开 Labview2014 软件，点击文件，创建项目，然后新建 VI，将窗口从前面板切换到程序框图窗口，右击鼠标出现函数选板，点击“选择 VI...”。

图 3-7

选择之前导入共享库生成 VI 的文件夹，就可以调用选择 VI 使用了。文件夹内包含了运动控制功能将会使用到的 VI。

3.3 指令返回值

每个调用到的 VI 都有几个数据接口，这些接口在编写程序时用来配置数据，同时每个 VI 都会有一个返回值接口，每个返回值都有其表示的含义，用户编程时可观察返回值来检验程序。下表是指令返回值的说明以及有问题的情形下的解决方法。

返回值	说明	出现的可能原因	解决措施
0	指令执行正常	/	/
-2	读取数据长度错误	PCI 通信异常，通信区被破坏，数据丢失或包含了杂数据	PCI 通信异常的原因很多，需要根据具体问题具体分析。 可以先尝试更换 PCI 插槽，清除控制卡PCI上金手指的灰尘，实在不行只能寄回工厂对硬件进行检测。
-3	读取数据校验和错误	数据在 PCI 传输过程中受到干扰，电平发生改变，导致数据被更改，因此是无效数据	
-4	写入数据块错误	Windows API 函数向PCI写或读数据出错，例如 PCI 通信异常，或者没有创建 PCI 通信就向通信区读写数据	
-5	读取数据块错误		
-6	打开/关闭设备错误	打开卡时，没有控制卡、或者控制卡数量超过最大设定值（4 张卡）、创建 PCI 通信区失败 关闭卡时，已经没有卡、或者关闭 PCI 通信区失败	/
-7	DSP 忙	调用 GT 指令后，DSP 仍然在处理，不再接收新指令	/
-8	多线程资源忙	GT 指令在线程里执行超时才返回，有可能是 PCI 通信异常，导致 GT 指令无法及时返回	/
1	错误调用指令	相关的指令没有调用，该指令的执行条件不满足	/
7	参数错误	输入指令的参数出错	/

8	DSP 固件不支持该指令	DSP 固件不支持该指令对应的功能	/

第 4 章 运动例程编写

4.1 开卡程序编写

使用运动控制卡，程序的开始的部分是开卡以及一系列的初始化动作。

每一个运动程序都会包含开卡、伺服使能、停止、关闭伺服等一些基本的 VI。当一个 VI 被调用到程序框图中后，VI 图标中会有数据接口，这些数据接口都是用来输入输出数据的，具体的数据设置需根据指令说明和用户需求来配置。

在前面板上，右键，控件选板上：新式—布尔—开关按钮，点击，移动鼠标将其放到前面板上合适的位置。这样的开关按钮需要六个，六个放置好后，右键开关按钮，点击属性，属性界面中外观页下，将标签处名字修改，五个开关按钮依次修改为：开卡、上使能、停止运动、关闭使能、位置清零、关卡。如图 4-1。

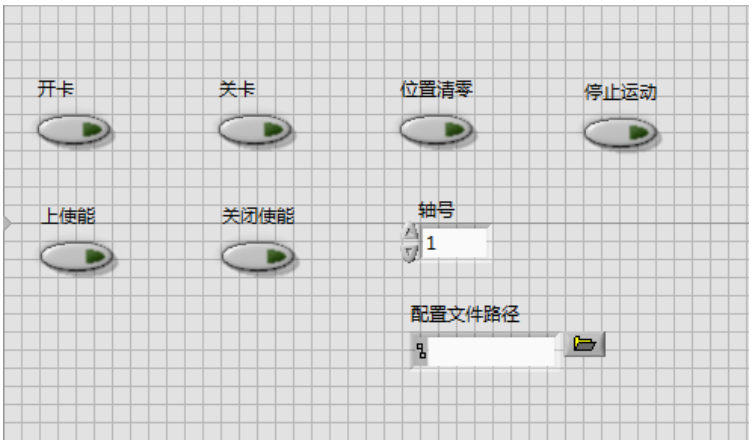


图 4-1

右键，控件选板上，新式—字符串与路径—文件路径输入控件，点击，移动鼠标放置到前面板上，这个控件的目的是选择配置文件的路径，加载配置文件。右击属性，标签将此控件名称改为：配置文件路径。

右键，控件选板—新式—下拉列表与枚举—枚举，点击，放置到前面板上，将其名称改为：轴号。并且，在其属性界面，编辑项中，插入项，插入 1 到 8 八个数据，代表的 1 到 8 轴（轴数的选择可根据需求来插入）。如图 4-2 所示。

在属性页面中，可以对控件的各种属性进行设置，常用的包括：数据类型、

显示格式、编辑项等。数据类型选项中，可以根据使用需要设置控件中数据的类型，显示格式选项下，可以设置不同的数据显示格式，来满足不同的函数及数据要求。

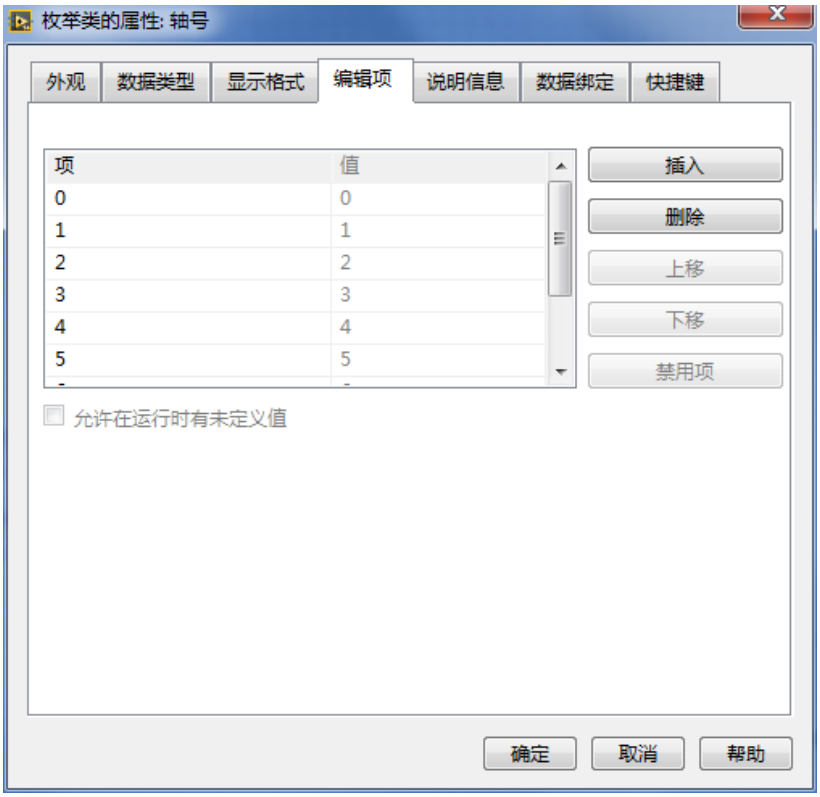


图 4-2

将界面切换到程序框图窗口，右键，函数选板—编程—结构—while 循环，点击在程序框图中新建一个 while 循环的框，然后再点击结构—事件结构，将事件结构放到 while 循环中。

右键单击事件结构的编辑器标签，点击添加事件分支，在事件源中选择开卡，事件选择鼠标，鼠标按下，然后点击确定，这样就添加好了开卡事件。如图 4-3、图 4-4。

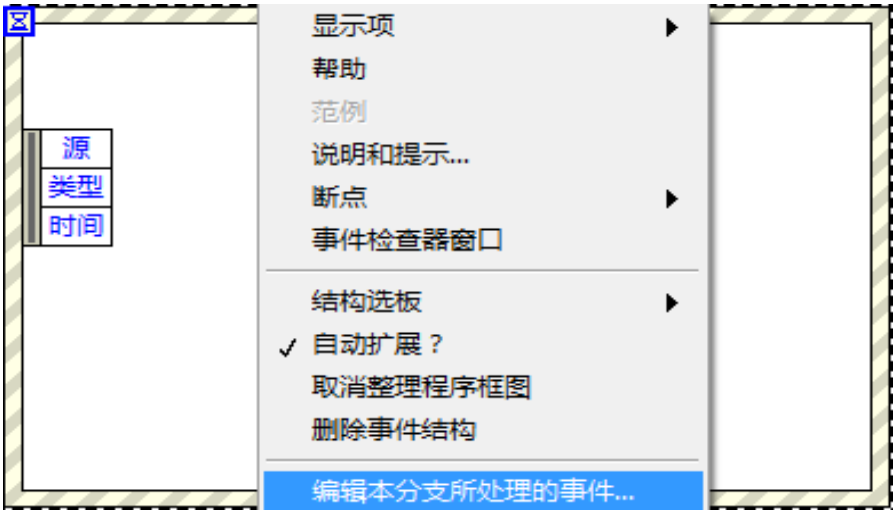


图 4-3

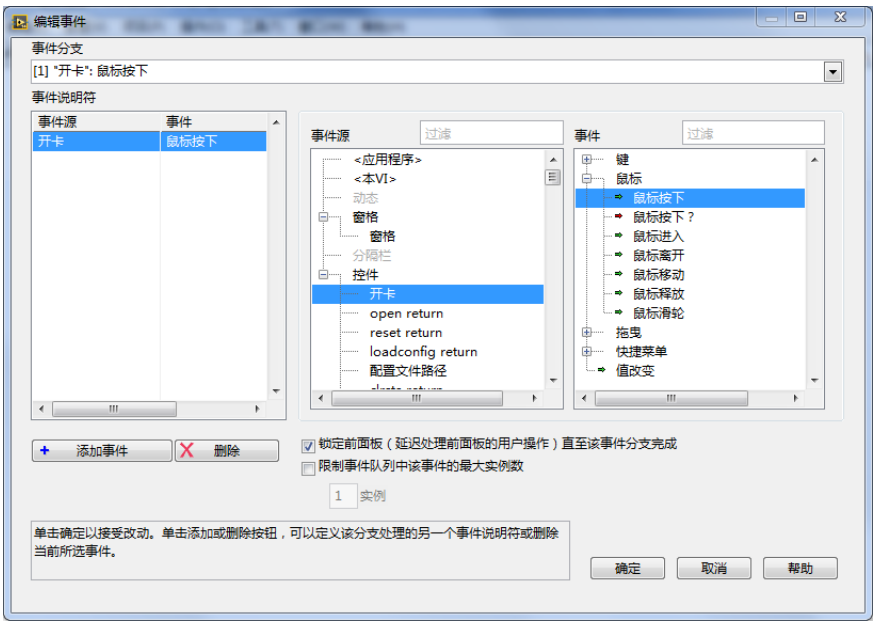


图 4-4

将“开卡”移动到开卡这个事件框图中，点击右键选择 VI，这个事件下需要使用到：GTN Open.vi,GTN Reset.vi,GTN load Config.vi,GTN Clr Sts.vi，从库中选出这四个 VI 放到开卡这个事件结构中。每一个 VI 都有数据接口和错误输入输出接口。将鼠标放到接口端就会显示出接口信息。将前一个 VI 的错误输出连到下一个 VI 的错误输入上，如下图。

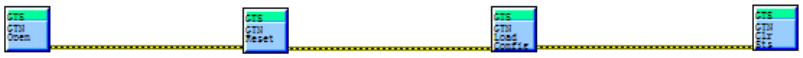


图 4-5

每一个 VI 都有一个返回值，返回值用来显示指令执行的具体情况，每一个使用到的 VI 最好都要加上一个返回值显示。前面板上，右键—控件选板—新式—数值—数值显示控件，移动鼠标放置到前面板上，并通过右键属性窗口，将此控件标签改为 open return。相应的为其他几个使用到的 VI 也都配上一个数值显示控件并修改标签名称。控件添加放置好后如图 4-6 所示。

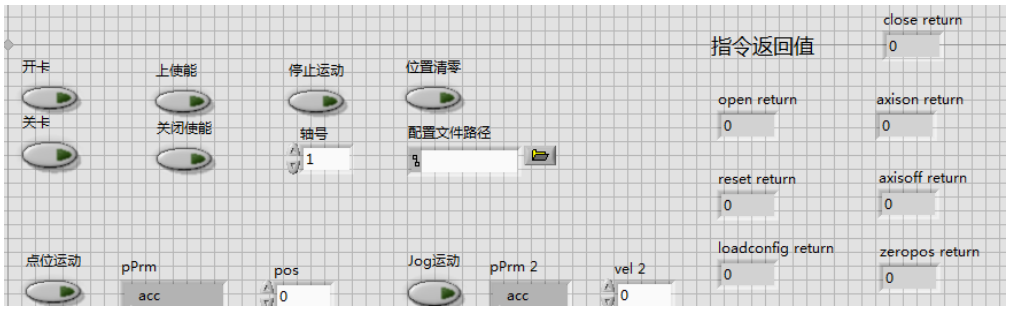


图 4-6

将窗口切换到程序框图界面，可以看到在前面板上新建的数值显示控件的接线端。按照名称定义将其连到对应的 VI 上。连接好后，完成的界面如下图。

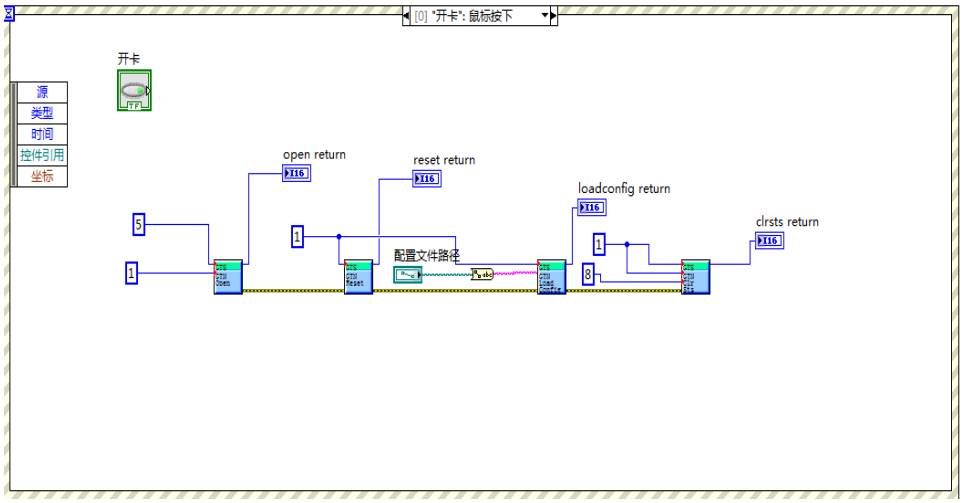


图 4-7

其中 GTN Load Config（加载配置文件）VI 的文件输入端的数据类型是字符串类型，因此需要一个函数将文件的路径转换成相应的字符串。该函数在函数选板—编程—字符串—路径/数组/字符串转换—路径至字符串转换。这样开卡这个时间的程序就可以了。

4.2 点位运动程序编写

运动模式是指规划一个或多个轴运动的方式。运动控制器支持的运动模式有点位运动模式、Jog 运动模式、电子齿轮（即 Gear）运动模式和插补运动模式。每一个轴在规划静止时都可以设置为点位运动。在点位运动模式下，各轴可以独立设置目标位置、目标速度、加速度、减速度、起跳速度、平滑时间等运动参数，能够独立运动或停止。

要进行点位运动，步骤是：开卡—复位—加载配置文件—清除状态，其中配置文件用 Motionstudio 软件生成并放到程序所使用的位置。点位运动的步骤是：设置指定轴为点位运动模式—读取点位运动模式下的运动参数—设置目标位置—设置目标速度—启动点位运动。在启动点位运动之前，需要给轴上使能。

前面板上，添加一个布尔按钮开关，标签名称改为上使能。界面切换到程序框图，在事件结构中添加事件分支，选择事件源：上使能。事件选择：鼠标按下。上使能是给所选的轴上使能。轴号通过之前的枚举控件（标签名称为轴号）来选择。如图 4-8 所示。

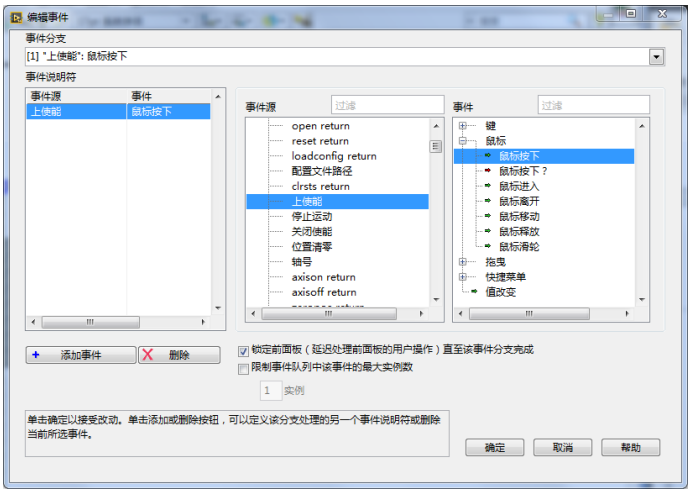


图 4-8

在程序框图界面中，把前面板上使能的接线端拉到上使能事件分支中，右键—函数选板—选择 VI—GTN Axis On，放置到程序框图中。

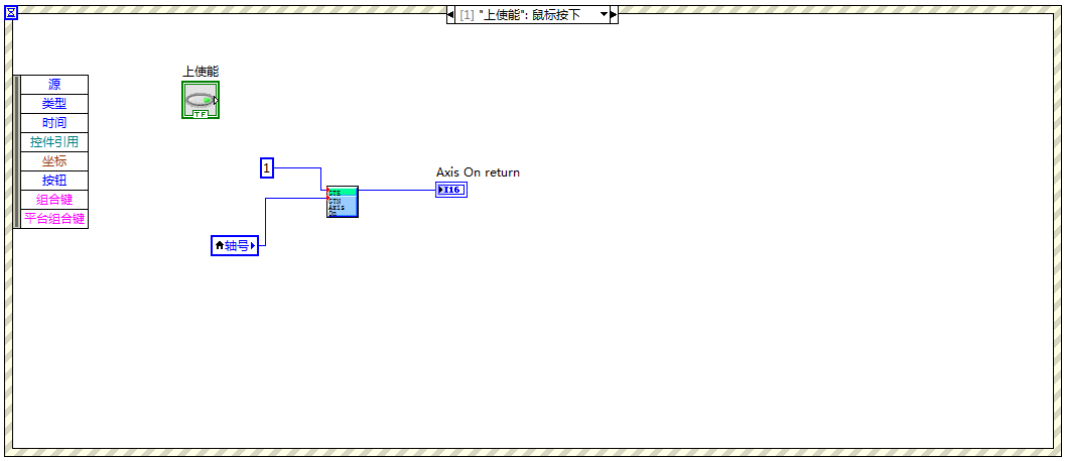


图 4-9

鼠标放到 VI 右边第一个接口是：core，即核号，第二个接口是：axis，即轴号，第三个接口是：错误输入。这里 core 接数值常量 1，代表核 1，鼠标右击—函数选板—结构—局部变量，放置到程序框图中，右键选择装换为读取，点击图标的中间，就会显示出可以选择的局部变量，选择轴号。同时和前面的操作相同，给这个 VI 接上一个返回值显示，名称为：Axis On return。完成之后如图 4-9 所示。

点位运动的步骤：GTN Prf Trap（设置指定轴为点位运动）—GTN Get Trap Prm（读取点位运动模式下的运动参数）—GTN Set Trap Prm（设置点位运动模式下的运动参数）—GTN Set Pos（设置目标位置）—GTN Set Vel（设置目标速度）—GTN Updata（启动点位运动）。在程序框图中按顺序放置点位运动步骤用到的 VI，将前一个的错误输出连到下一个 VI 的错误输入上。

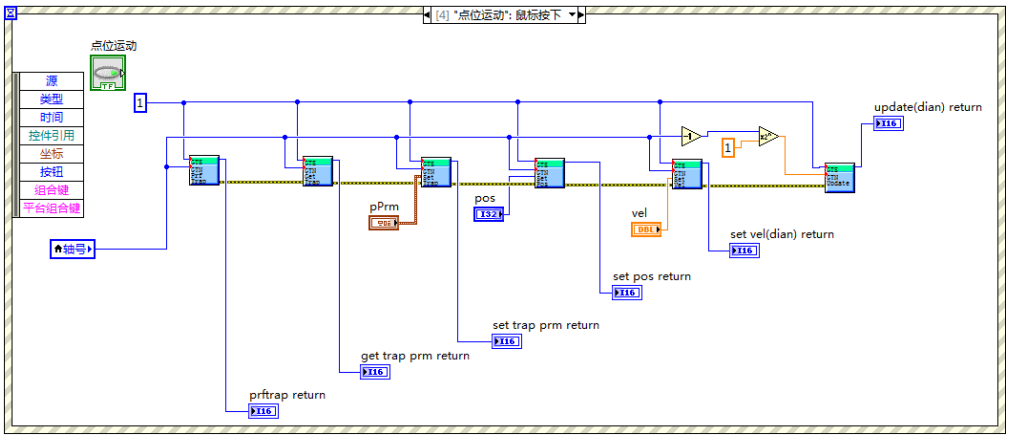


图 4-10

其中 GTN Set Trap 右侧的第三个接口，是 pPrm，是一个簇数组输入。pPrm 中存放着点位运动的加速度、减速度、起跳速度、平滑时间。这些数据在启动点位运动之前就要设定好。

前面板上右键—控件选板—新式—数组、矩阵与簇—簇，放置到前面板上，最好放到点位运动的开关按钮的旁边。右键属性将标签名称改为：pPrm。

然后再新建四个数值显示控件放到簇中，四个数值显示控件名称分别为:acc、dec、velStart、smoothTime。其中 acc、dec、velStart 是 double 型数值，smoothTime 是 short 型数值，右键 smoothTime 控件，选择属性，在属性窗口数据类型选项下，点击表示法选择 I16 型。还需要新建两个数值输入控件，一个 pos，是点位运动的目标位置，另一个是 vel，是点位运动的目标速度。

完成之后的程序框图如图 4-10 所示。

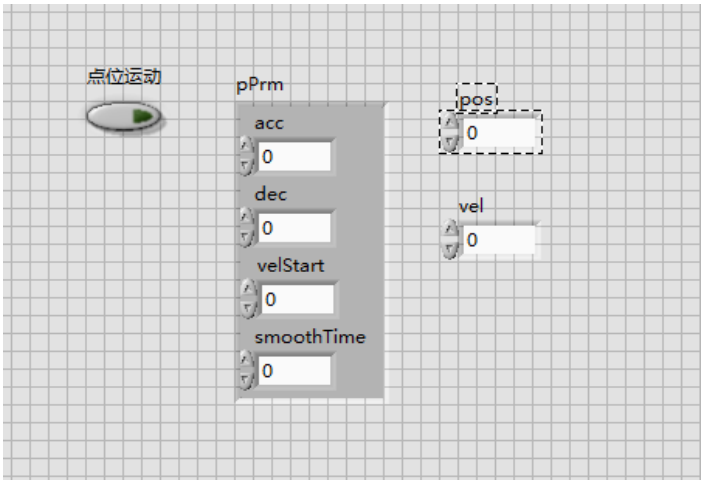


图 4-11

pos 为 32 位的整型数值，vel 是 double 型数值，通过属性窗口将数值类型进行更改。

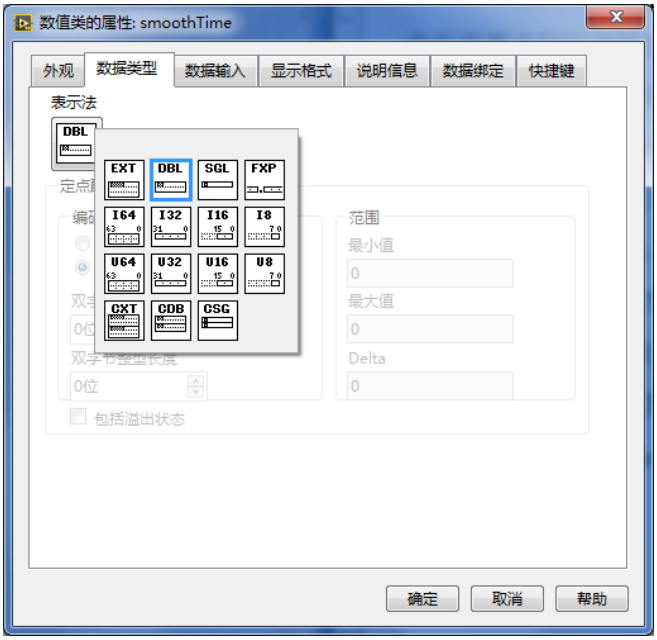


图 4-12

vel 和 pos 数值可以通过按钮上下选择，也可以直接手动输入数值。最后一步启动点位运动 GTN Updata 的右边第二个接口是 mask，按位指示需要启动点位运动或 Jog 运动的轴号。当 bit 位为 1 时表示启动对应的轴，一共从 0 到 11 位。对于每一个执行的 VI 都需要接上一个数值显示控件来显示返回值。如图 4-13 所示，点位运动的程序。

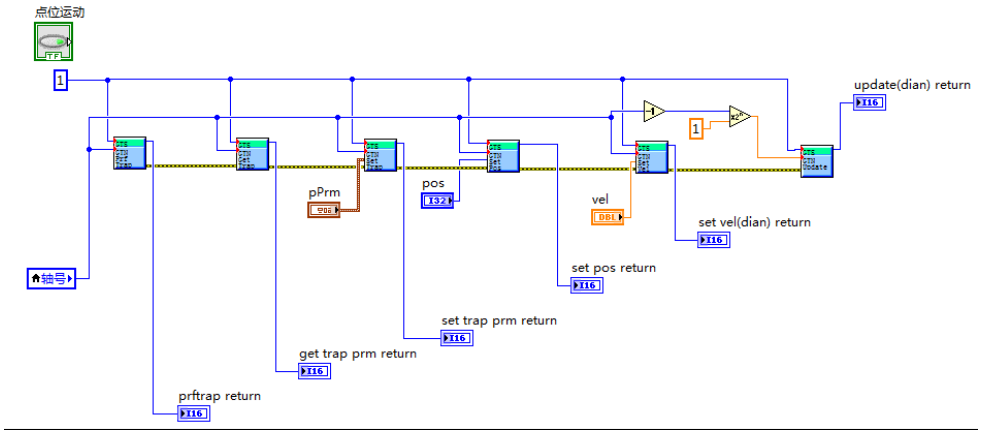


图 4-13

在设置或切换运动模式时，要保证轴处于规划静止状态。如果轴正在运动，请先调用 GTN Stop 指令停止一个或多个轴的运动，也就是前面板上的停止运动按钮，然后再切换到想要的运动模式下。

4.3 Jog 运动程序编写

在 Jog 运动模式下，各轴可以独立设置目标速度、加速度、减速度、平滑系数等运动参数，能够独立运动或停止。调用 GTN_Update 指令启动 Jog 运动以后，按照设定的加速度加速到目标速度后保持匀速运动，在运动过程中可以随时修改目标速度。

前面板上，添加一个布尔开关按钮，标签名称改成 Jog 运动。Jog 运动的步骤：GTN Prf Jog（设定指定轴为 Jog 运动模式）—GTN Get Jog Prm（读取 Jog 运动模式下的运动参数）—GTN Set Jog Prm（设置 Jog 运动模式下的运动参数）—GTN Set Vel（设置目标速度）—GTN Update（启动 Jog 运动）。如图 4-14 所示，将所需的 VI 一次添加到程序框图上，将前一个 VI 的错误输出接到下一个 VI 的错误输入上，各个 VI 的 core 和 profile 都接上正确的输入数值。

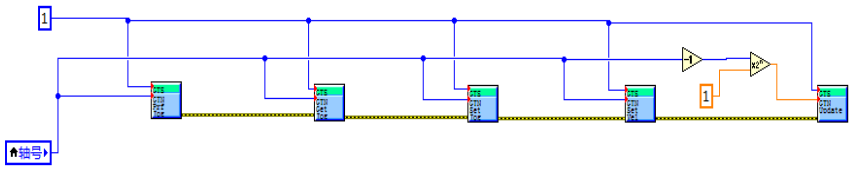


图 4-14

其中 GTN Set Jog Prm（设置 Jog 运动模式下的运动参数）VI 的右侧第二个接口是一个簇数组类型的输入，这个簇数组中包含：acc、dec、smooth，分别代表的意思是：点位运动的加速度、点位运动减速度、平滑系数。其中平滑系数的取值范围为：[0,1)，平滑系数的数值越大，加减速过程越平稳。

在前面板上添加一个簇控件，便签名称改为：pPrm2（与点位运动的簇名称区别开）。并在簇中加入三个数值显示控件，数值类型均为 double 型，标签名称依次是：acc、dec、smooth。将 Jog 运动布尔开关按钮的接线端拉到 Jog 运动事件结构分支中，和前面的做法一致，为每个 VI 接上一个数值显示控件，用来显示指令返回值。图 4-15 为接好的程序图。

设置速度和启动运动的返回值显示控件的名称要与点位运动那里的两个数值显示控件名称区别开来。

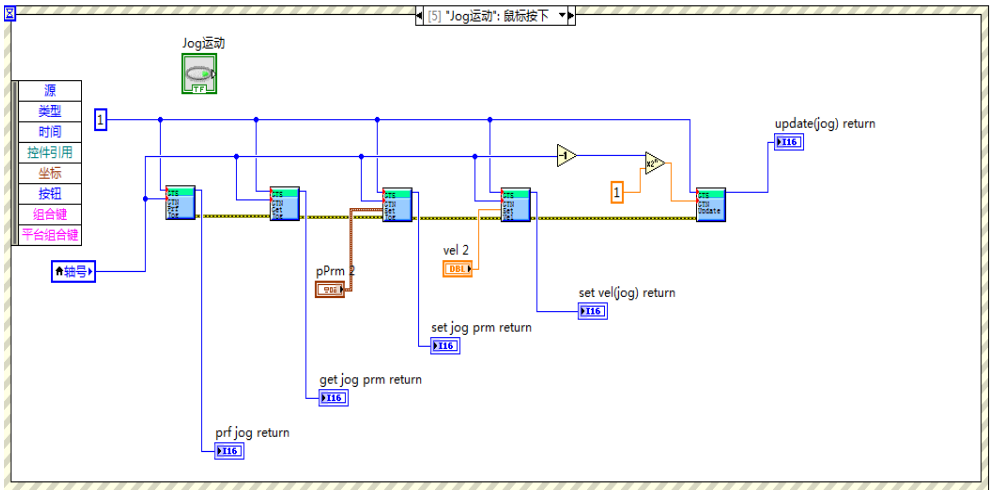


图 4-15

Jog 运动事件分支的事件是鼠标按下，事件中的程序当鼠标点击 Jog 运动按钮按下时就会执行，当松开鼠标时运动就要停止。右键点击时间结构标签编辑器，选择添加时间分支。时间源：Jog 运动，事件：鼠标释放。如图 4-16。

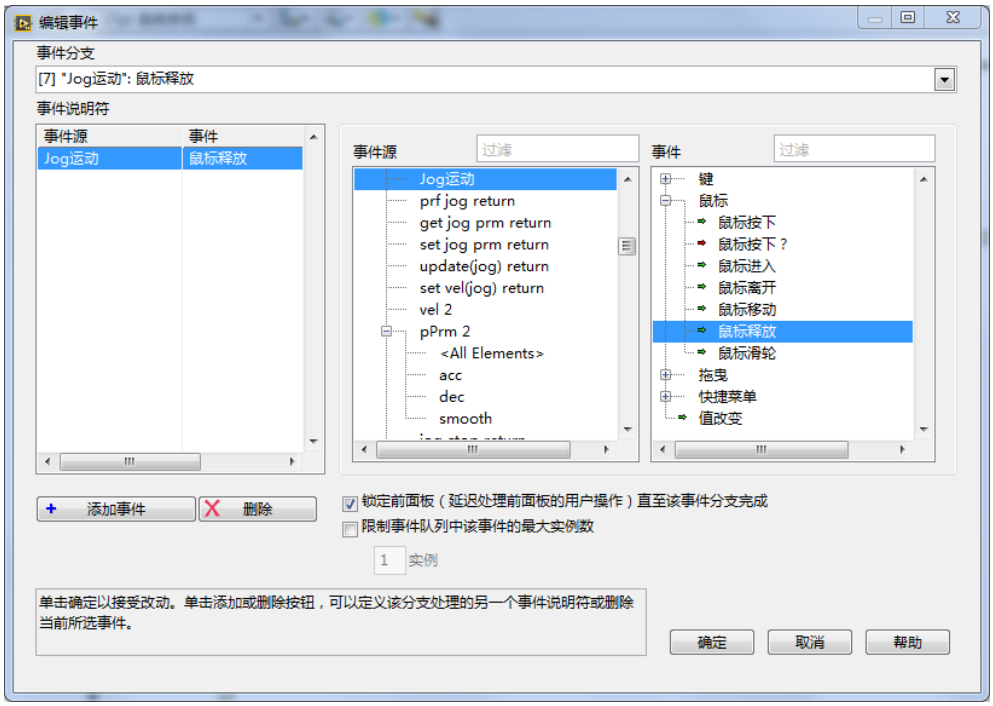


图 4-16

鼠标释放后轴就停止运动，在 Jog 运动鼠标释放事件分支中，右键选择 VI，选择 GTN Stop（停止一个或多个轴的规划运动，停止坐标系运动）VI，添加到程序框图上。左侧三个接线端（错误输入不接）按照指令说明及使用需求连接好相应的数据。图 4-17 为已经写好的程序。

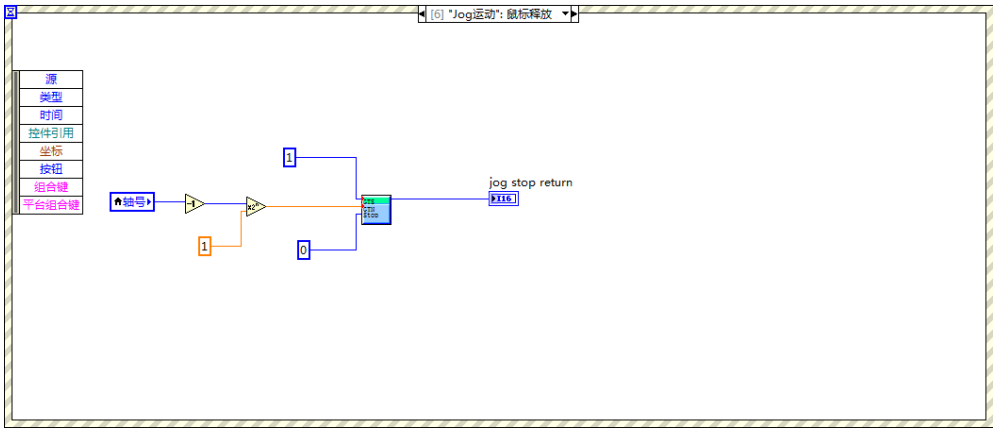


图 4-17

4.4 位置速度实时显示程序

当轴在运动时，希望观察到轴的一些数据，这些数据包括：（轴）实际位置、规划位置、实际速度、规划位置、轴状态。

前面板上，新建四个数值显示控件，标签名称分别为：实际位置、实际速度、规划位置、规划速度。

以 1 轴为例，新建一个 while 循环，循环条件改为真（T）时连续。右键选择 VI，选择添加四个 VI：GTN Get Enc Pos、GTN Get Prf Pos、GTN Get Enc Vel、GTN Get Prf Vel，表示的是：读取实际位置、读取规划位置、读取实际速度、读取规划速度。添加到程序框图上后，按照指令说明和使用需求，各个接口都接上相应的数据，并和所对应的数值显示控件相连。下图所示，为 1 轴的位置速度实时显示。

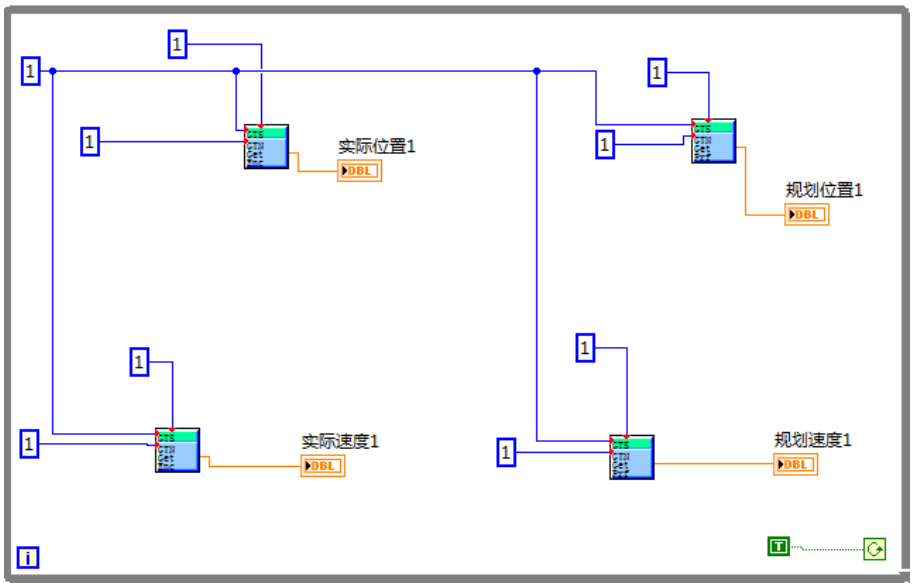


图 4-18

4.5 停止运动程序编写

在前面板上已经新建了停止运动的布尔按钮开关，程序框图中新建一个 while 循环，在 while 循环中添加一个条件结构，将停止运动的布尔按钮开关的接线端放入 while 循环中。

条件结构“真”条件下，添加 GTN Stop VI。停止运动的连线端接到条件结构的条件选择器。core 接数值常量 1，mask 接数值常量 255，option 接数值常量 0，返回值输出也需接上一个数值显示控件。编辑好的程序框图如下。当程序运行，控制轴运动时，某些时刻或测试需要停止轴运动的，按下停止运动按钮，所有的轴停止运动。

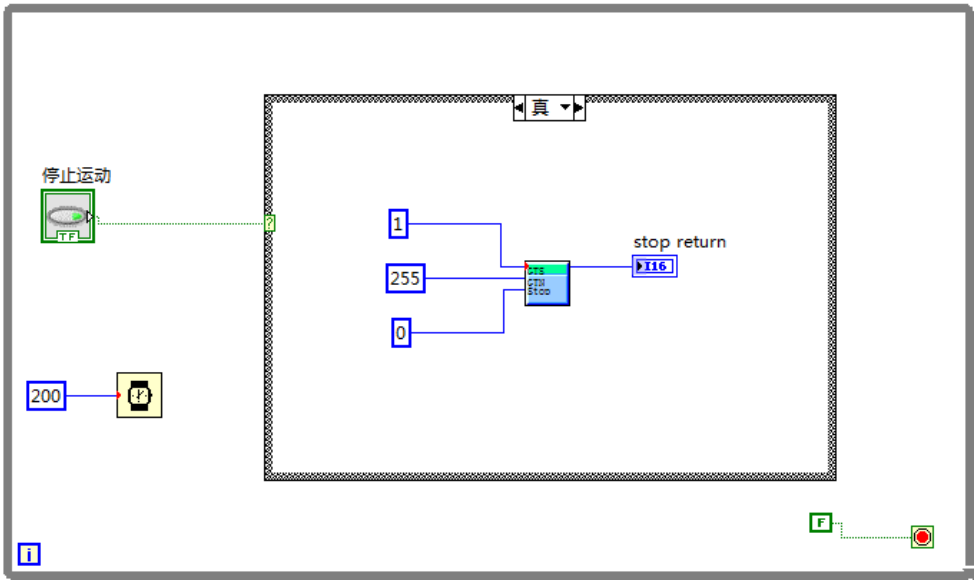


图 4-19

4.6 回原点运动程序编写

回零使用 Smart Home 功能来实现，Smart Home 是运动控制器的“Home/Index 回原点”和“自动回原点”的优化和扩展。Smart Home 仍然采用高速硬件捕获机制实现回原点，把原来较为繁琐的回零过程固化到控制器，只需要调用简单指令就能够实现回原点，简化了用户程序。此外，Smart Home 汲取了工控界较为常见的回原点方式，集成到控制器给予实现，具体包括：限位回原点、限位+Home 回原点、限位+Index 回原点、限位+Home+Index 回原点、Home 回原点、Home+Index 回原点、Index 回原点。

每一种方式都包含正向和负向两种方法，正向指规划位置为正数的方向，负方向指规划位置为负数的方向，例如“限位+Home 回原点”即包括了“正限位+Home 回原点”和“负限位+Home 回原点”；每一种回原点方式，都可以通过设置偏移量使得最终电机停止的位置离原点位置有一个偏移量；每种回原点方式可能由于设定的搜索距离、电机意外停止等因素而找不到原点，大部分异常情况都可以通过查看回原点状态来进行辨识。此外，Smart Home 功能支持各轴单独回原点，互不影响，即可以实现多轴同时回原点。

程序前面板上，新建一个布尔开关按钮控件，标签名称为：回原点。新建四个数值输入控件，名称分别为：回零方式、回零方向、回零高速、回零低速。回零方式和回零方向数据类型为 I16，回零高速和回零低速数据类型为 double 型。

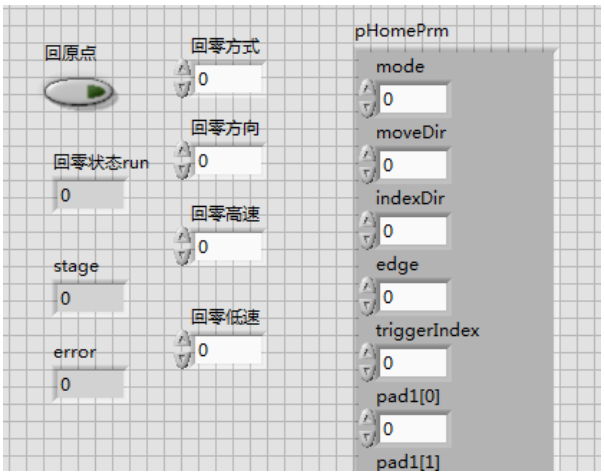


图 4-20

切换到程序框图界面，在事件结构中添加事件分支，事件源选择控件下的回原点，事件选择鼠标—鼠标按下，确定建立回原点事件分支。

需要用到四个 VI：GTN Get Home Prm（读取设置到控制器的 Smart Home 回原点参数）、GTN Go Home（启动 Smart Home 实现各种方式回原点）、GTN Get Home Status（获取 Smart Home 回原点状态）、GTN Zero Pos（位置清零）。选择 VI 将这 4 个 VI 添加到程序框图中。使用 Smart Home 回零完成后不会自动清零位置，需要调用 GT_ZeroPos 把当前位置清零来确定零点。GTN Go Home 的左侧第三个接口输入的是一个簇数组，里面存放的是回原点的各种参数。双击 GTN Go Home 的 VI 图标，进入到 GTN Go Home VI 的前面板窗口，就可以看到 pHomePrm 簇数组和 pHomePrm out 簇数组。在主程序的前面板窗口，创建这样的簇数组或者复制添加。回原点事件分支中，将 pHomePrm 按名称捆绑显示出前四个选择项，并将对应的数值输入控件的接线端连接好。

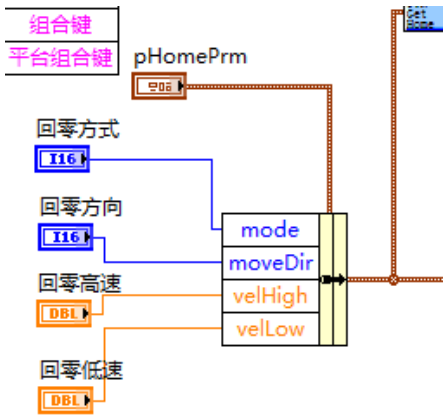


图 4-21

GTN Get Home Status 的 pHomeStatus out 接口按名称解除捆绑，显示出 run、stage、error，run 用来判断回零是否完成，stage 表示回原点运动的阶段，error 表示回原点过程发生的错误。创建对应的显示控件与 run、stage、error 连接。

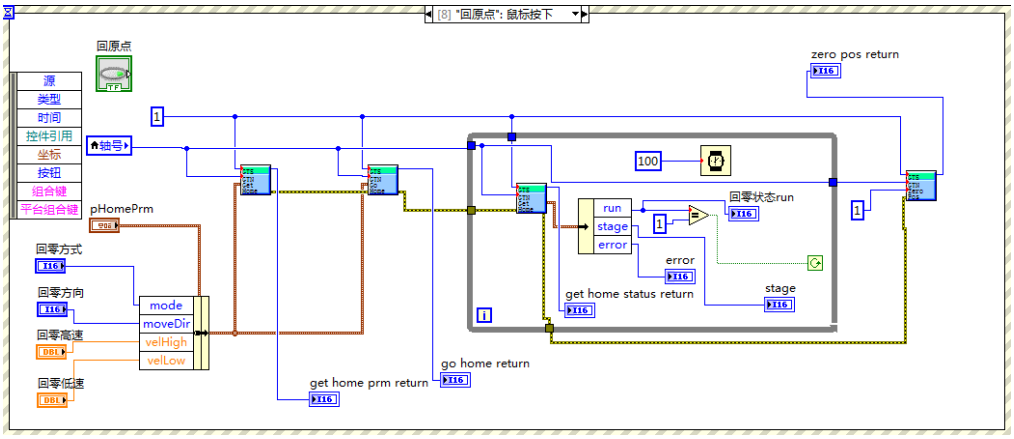


图 4-22