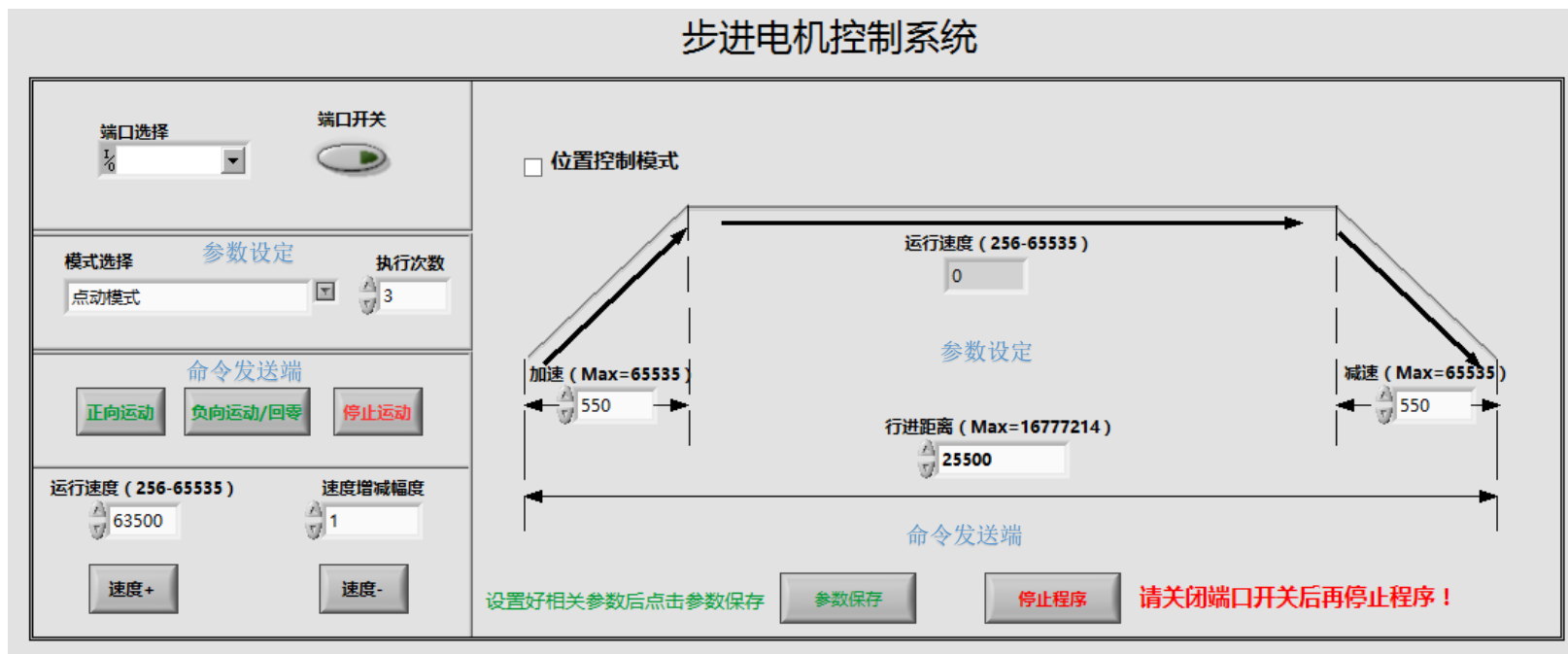
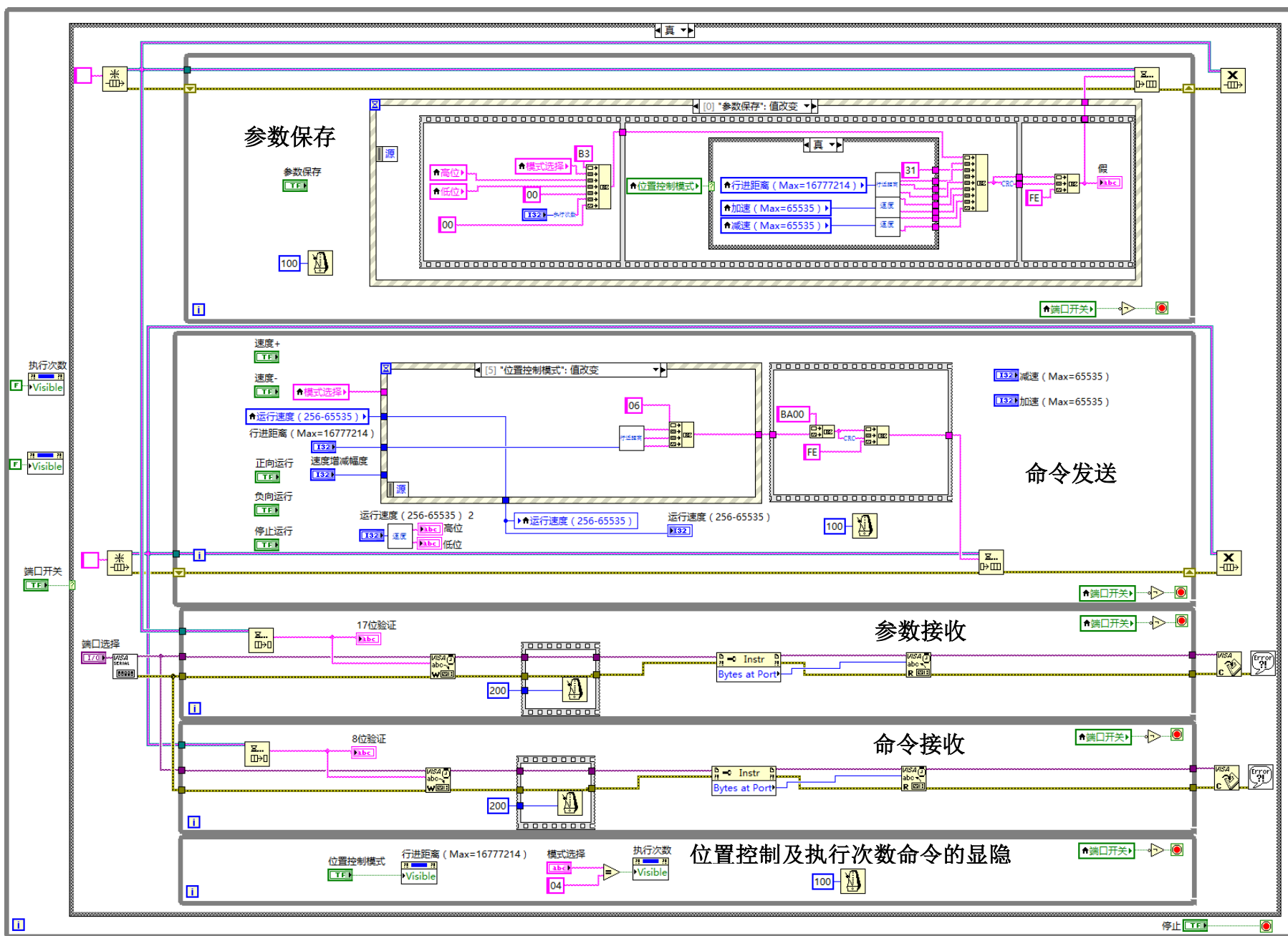


① 前面板用于人机交互，主要分为两部分：参数设置和命令发送。如图？所示。



图？

② 程序框图主要分为三部分：参数保存与接收、命令发送与接收、位置控制及执行次数命令的显隐。如图? 所示。

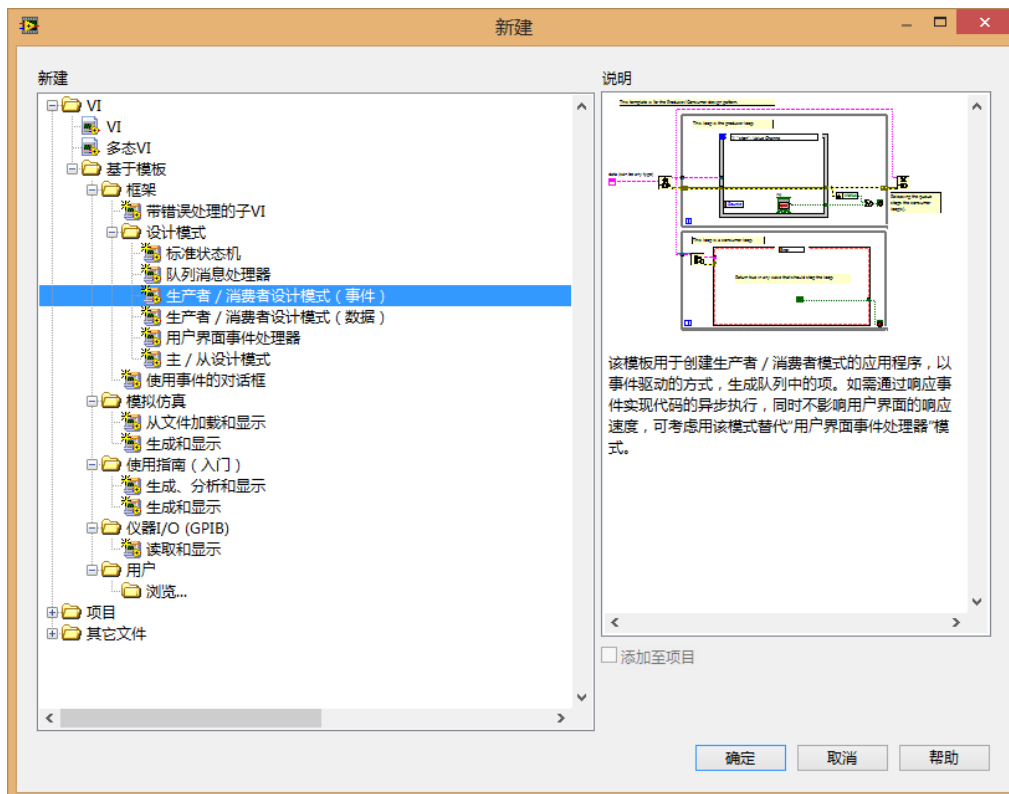


图？

3 操作步骤

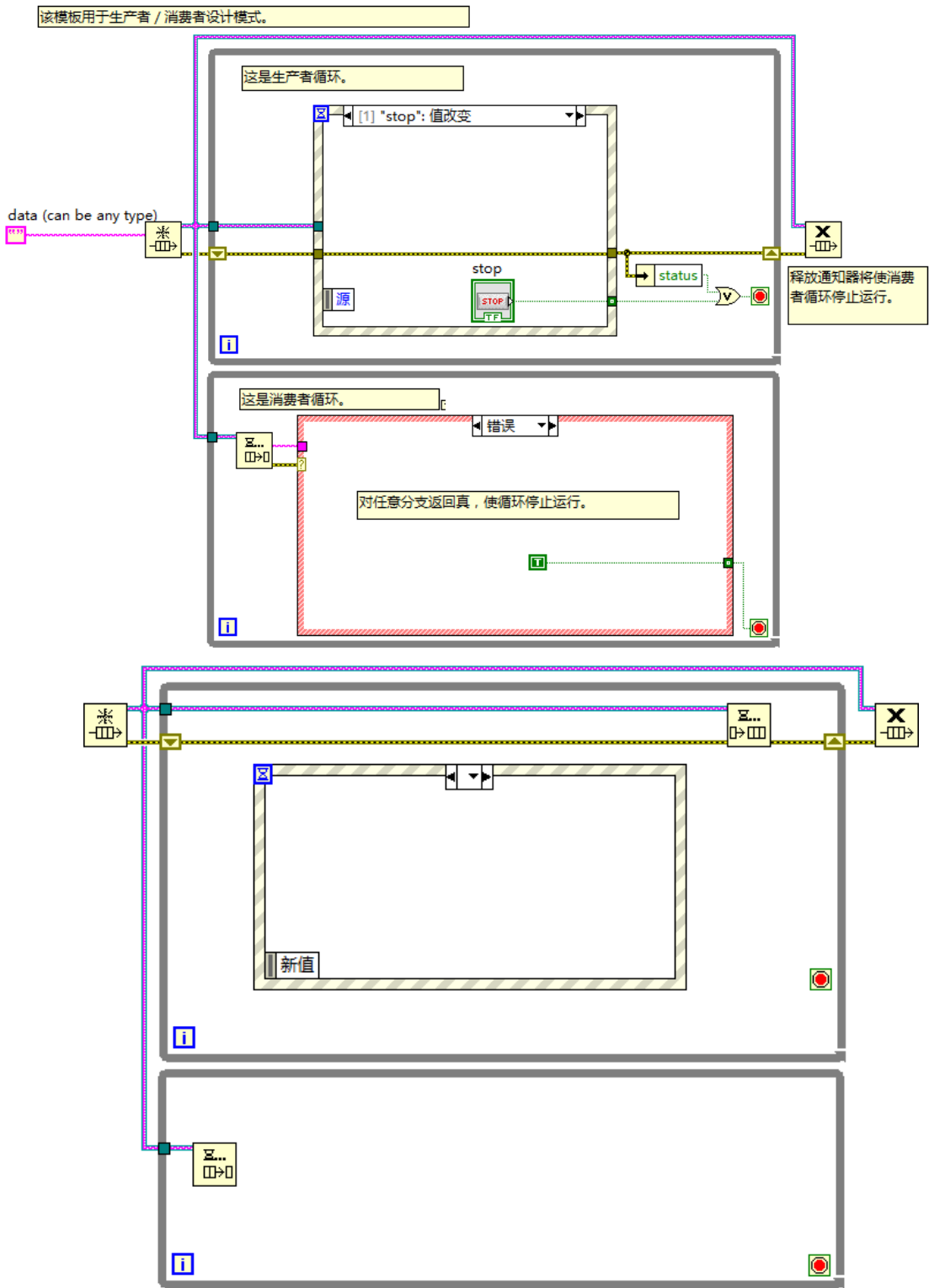
3.1 构建程序框架

(1) 程序框架以系统自带的生产者/消费者设计模式(事件)作为基础，以此来创建多循环框架。(在打开程序后，选择<文件>/<新建>)如图？设计模式所示，选择生产者/消费者设计模式(事件)的设计模式。



图？

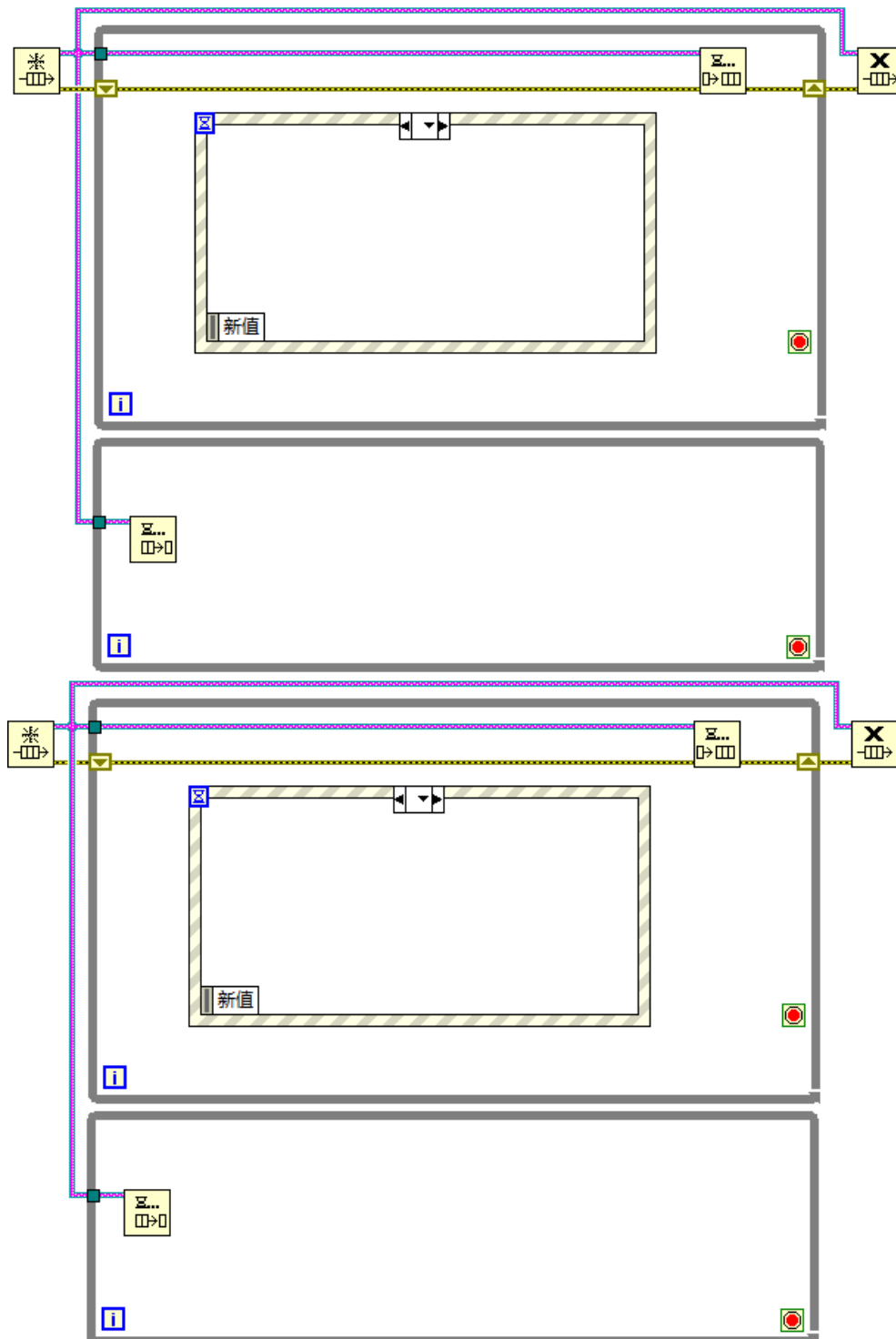
(2) 打开该模式后，删除不必要的说明以及程序控件，清空【事件结构】内部控件。整理前后效果对比如图? 所示。





图?

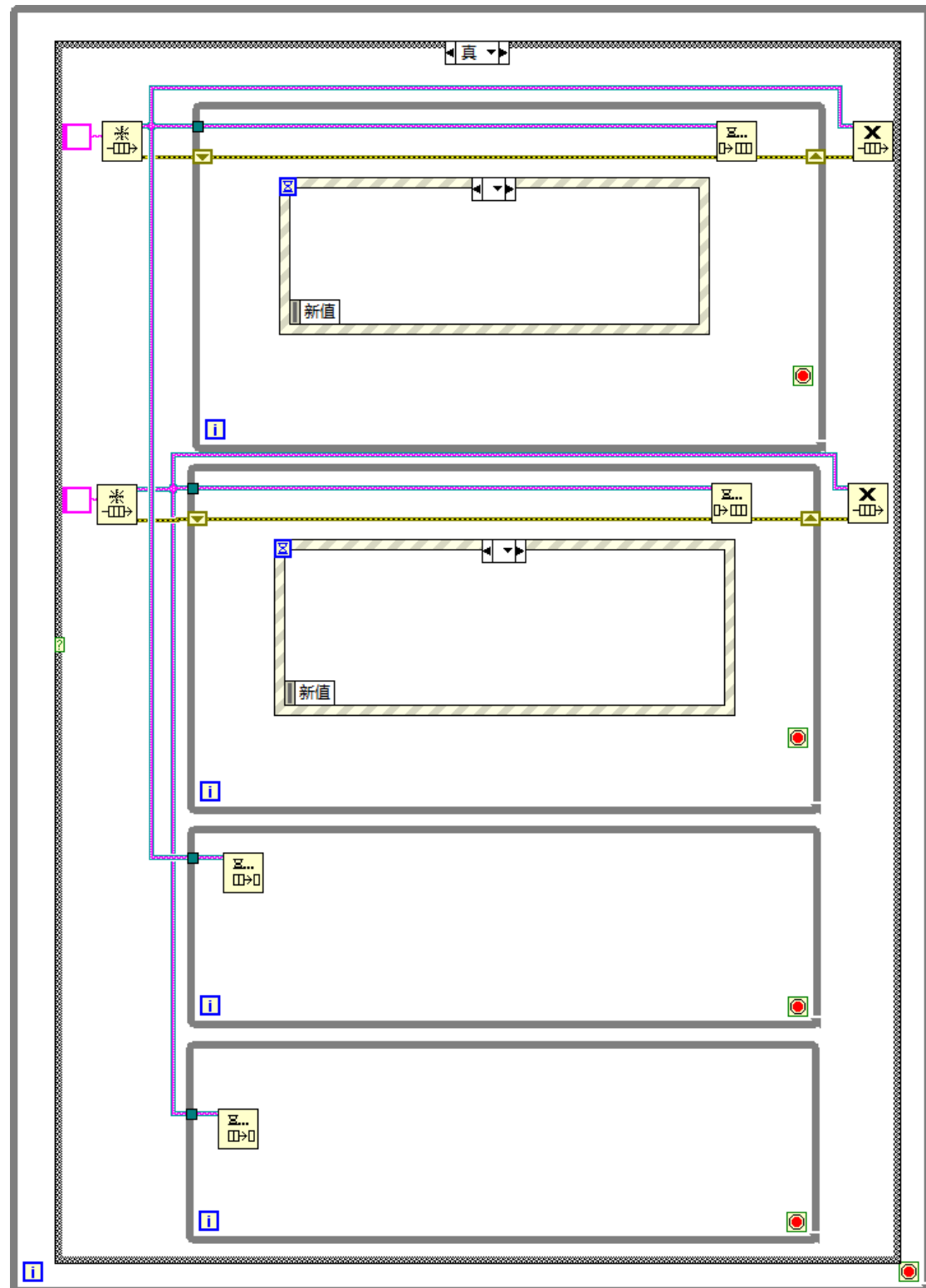
(3) 前面提到程序由参数保存与接收、命令发送与接收和位置控制及执行次数命令的显隐三大部分组成因此还需创建一个阵列结构和一个【While 循环】。

在程序框图中，使用快捷键<Ctrl+a>选中整个程序，按下<Ctrl>的同时，拖动程序，复制出一个新的程序框。如图? 所示。



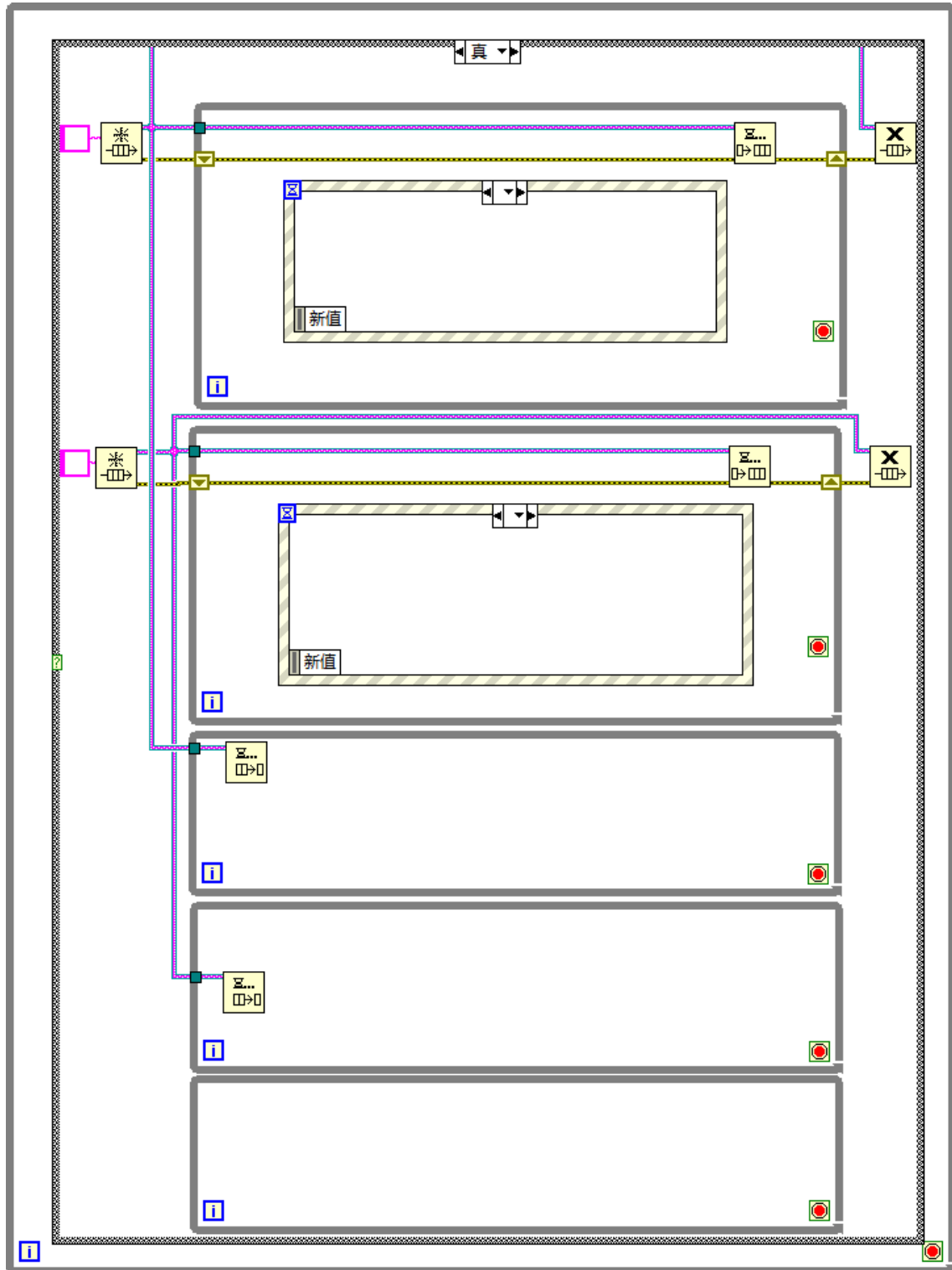
调整框图间的顺序，在空白处创建【字符串常量】，右击【字符串常量】选择<十六进制显示>，并与【获取整列引用】 创建的“元素数据类型”连接，.

最终整理出的结果如图? 所示。



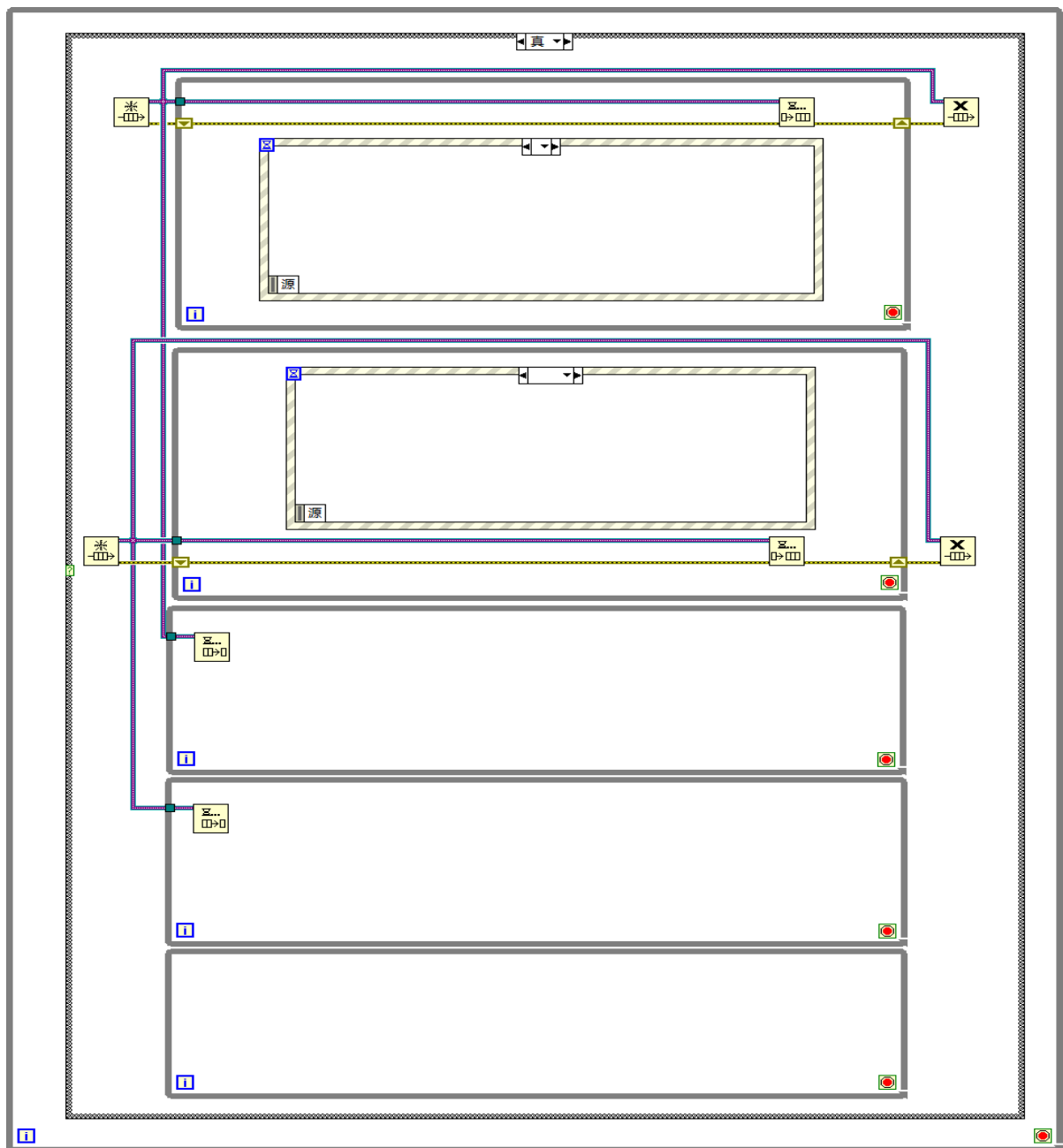
图？

图? 所示。



图？

(5) 考虑到有一个端口打开功能，在主循环外创建【条件事件】。再在【条件事件】外创建【While 循环】。最终结果如图? 所示。



图?

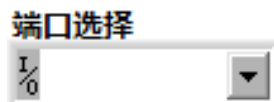
(6) 使用快捷键<Ctrl+s>快速保存文件，并命名为“主程序”。

3.2 编辑前面板

以下操作皆为在前面板下进行的操作。

(1) 创建【VISA 资源名称】(【控件】/【新式】/【I/O】/【VISA 资源名称】)

并将<标签>改名为“端口选择”。如图? 所示。



图?

(2) 创建【开关按钮】并将<标签>改名为“端口开关”。 如图? 所示。

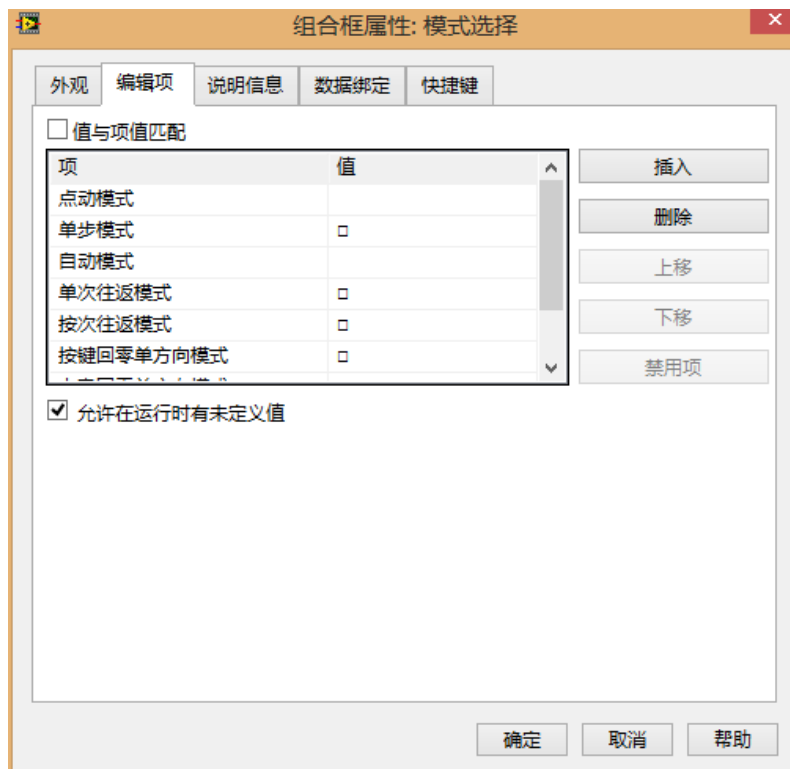


图?

(3) 创建【组合框】(【控件】/【新式】/【字符串与路径】/【组合框】)并将<标签>改名为“模式选择”。右击该控件，选择<属性>/<编辑项>插入下列项和值，将“值与项值匹配”勾去，并依次创建项和它所对应的值，结果如图? 所示。

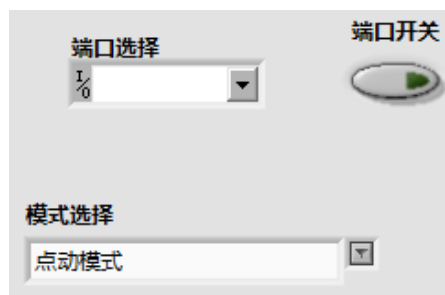
项	值(16 进制)
点动模式	00
单步模式	01
自动模式	02
单次往返模式	03
按次往返模式	04
按键回零单方向模式	05
上电回零单方向模式	06
上电运行单方向模式	07

16 进制(HEX)可由【字符串常量】生成，创建【字符串常量】，右击【字符串常量】选择<十六进制显示>，输入 00 00，再次右击【字符串常量】选择<正常显示> 00，将生成后的值复制粘贴至属性界面中对应的“值”位置即可，余下部分以此类推。



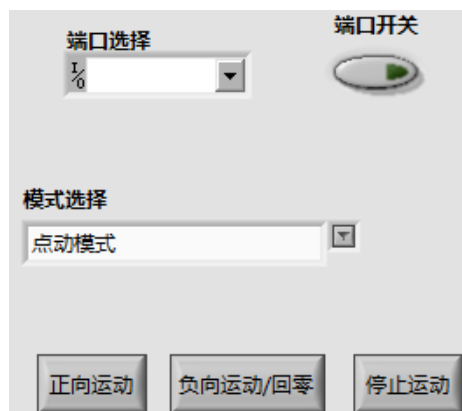
图？

创建完成后结果如图？所示。



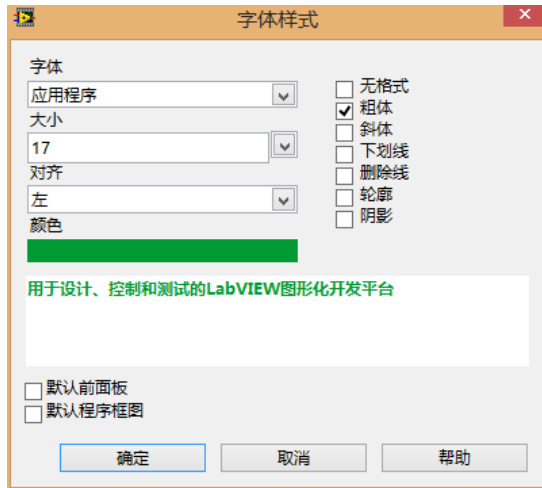
图？

(4) 在前面板创建 3 个【确定按钮】，去除<标签>(右击控件<显示项>/<标签>)。依次改名为正向运动、负向运动/回零、停止运动，如图？所示。



图？

编辑控件的颜色使用快捷键<Ctrl+0>，弹出“字体样式窗口”，选择相应“颜色”，如图？所示。效果如图？所示。

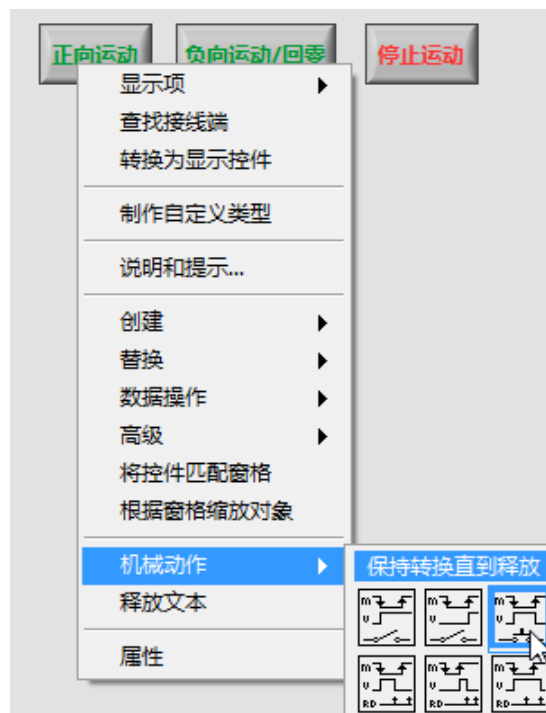


图？



图？

依次右击 3 个控件，选择<机械动作>/<保持转换直到释放>，如图？所示



图？

(5) 创建【文本下拉列表】(【下拉列表与枚举】/【文本下拉列表】)，并将<标签>改名为“速度增减幅度”。右击该控件，选择<属性>/<编辑项>插入下列项和值，如图？所示。效果如图？所示。

项	值
1	1
10	10
20	20
50	50
100	100
200	200
500	500
1000	1000
2000	2000
5000	5000
10000	10000



图？



图？

(6) 在前面板创建 2 个【确定按钮】，去除<标签>(右击控件<显示项>/<标签>），依次改名为速度+、速度-。如图？所示。



图？

(7) 在前面板创建【数值输入控件】并将<标签>改名为“运行速度(256-65535)”，。如图？所示。



图？

(8) 在前面板创建【系统复选框】(【控件】/【系统】/【布尔】/【系统复选框】)，删除“OFF/NO”并将<标签>改名为“位置控制模式”。如图？所示。



图7

(9) 在前面板创建 4 个【数值输入控件】并将<标签>分别改名为“加速(Max=65535)”、“减速(Max=65535)”、“行进距离(Max=16777214)”、“执行次数”。如图7 所示。



图8

(10) 在前面板创建【数值显示控件】并将<标签>改名为“运行速度(256-65535)”。如图7 所示。





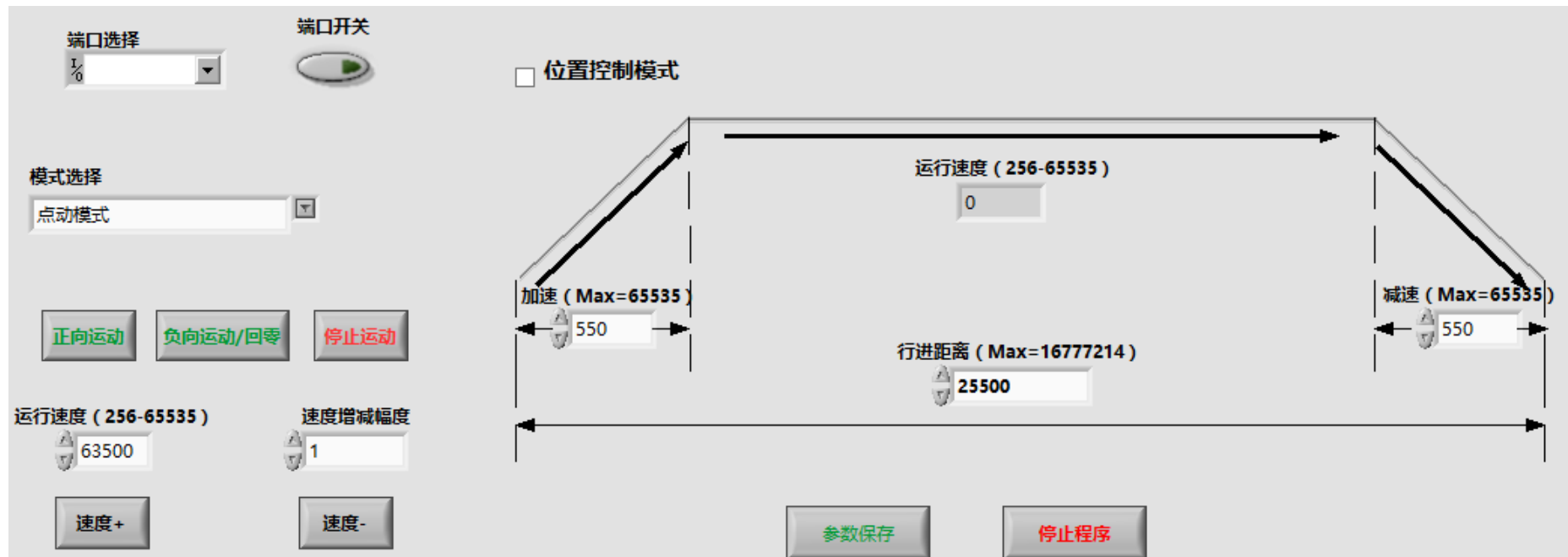
图7

(11) 在前面板创建【确认按钮】和【停止按钮】去除各自<标签>依次改名“参数保存”和“停止程序”，改变颜色。如图8所示。



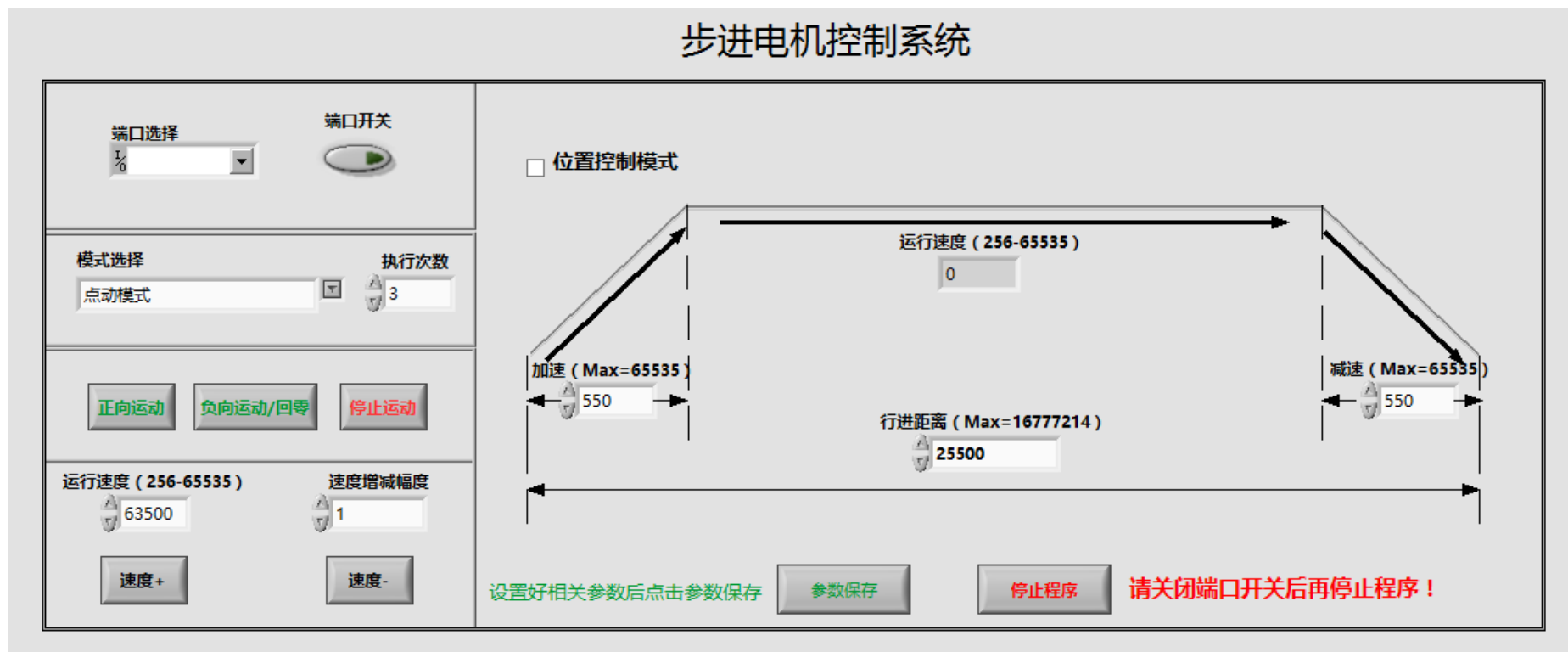
图8

(12) 在前面板创建加速、减速以及匀速运动的效果图。创建【带箭头粗线】和【细线】（【控件】/【新式】/【修饰】/【带箭头粗线】|【细线】）构建框架，编辑各个控件的位置，结果如图？所示。



图？

(13) 在前面板创建【平面框】(【控件】/【新式】/【修饰】/【平面框】)编辑各个控件的位置并添加必要的说明，前面板最终构架如图?所示。

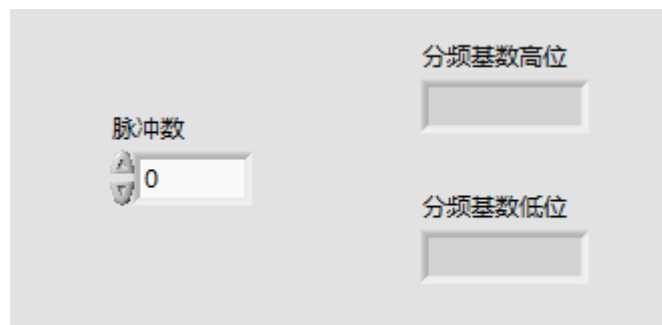


图?

3.3 编辑程序框图

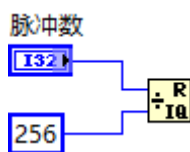
(1) 创建速度子 VI(为了使程序不显得杂乱, 因而选择创建一系列的子 VI)

① 新建一个 VI,在前面板创建【数值输入控件】改名为“脉冲数”和 2 个【字符串显示控件】右击 2 个【字符串显示控件】选择<十六进制显示>, 分别改名为“分频基数低位”和“分频基数高位”。如图? 所示。



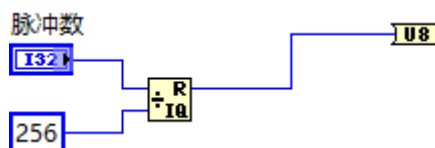
图?

② 在程序框图创建【商与余数】(【函数】/【编程】/【数值】/【商与余数】), 将【数值输入控件】与【商与余数】的 x 端相连。右击【商与余数】的 y 端, 选择<创建>/<常量>, 常量值为 256。如图? 所示。



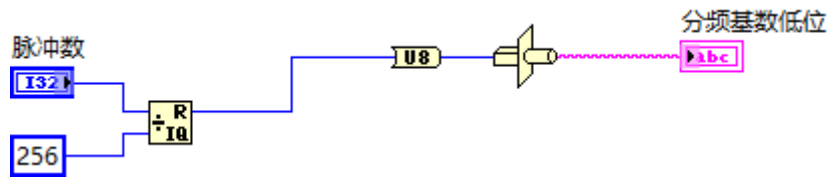
图?

③ 在程序框图创建【转换为无符号单字节整型】(【函数】/【编程】/【数值】/【转换】/【转换为无符号单字节整型】)将它与【商与余数】的 R 端相连。如图? 所示。



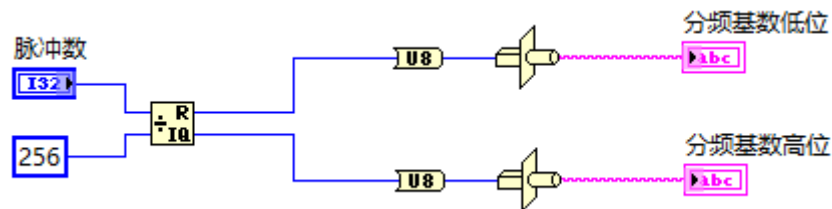
图?

④ 在程序框图创建【强制转换类型】(【函数】/【编程】/【数值】/【数据操作】/【强制转换类型】)将它和【转换为无符号单字节整型】相连, 最后链接至 【分频基数低位】。如图? 所示。



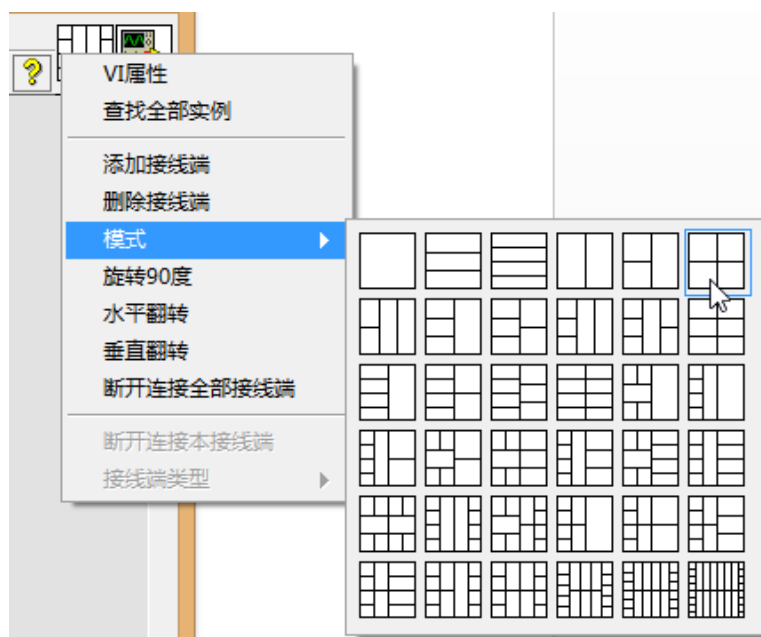
图？

⑤ 重复③、④的操作步骤，创建与【分频基数高位】的连接。如图？所示。

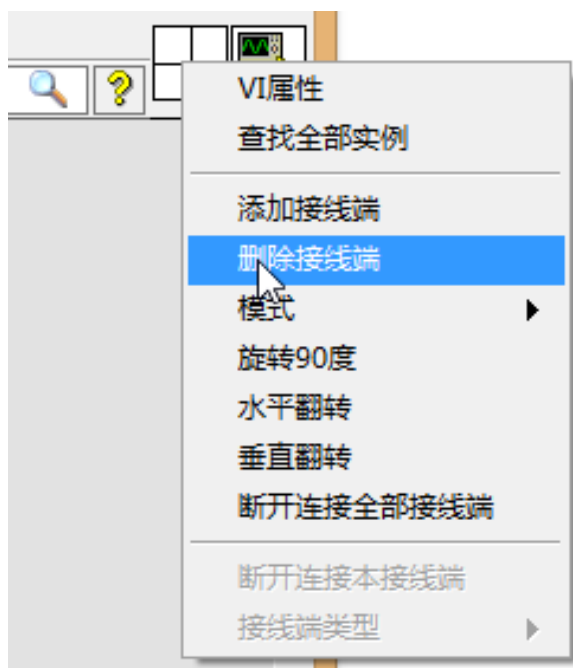


图？

⑥ 在前面板的连线框图中右击选择如图？所示连线模式。以连线框图的中心线段为分界线。左边为输入控件，右边为输出控件，可根据 VI 内的输入和输出控件的数目编辑框图。此 VI 只有一个输入控件，所以在选择好的连线模式框图的左边，右击选择删除接线端。如图？所示。

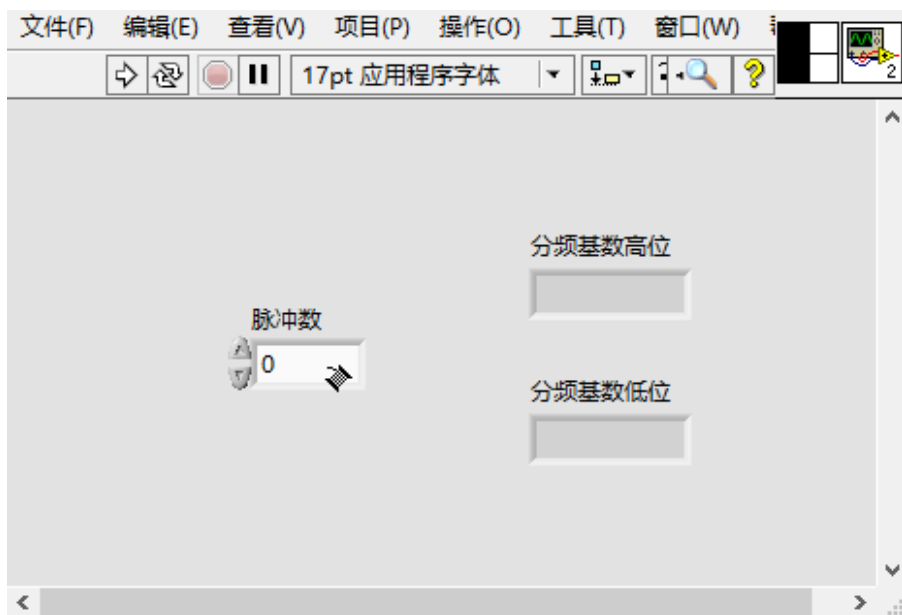


图？



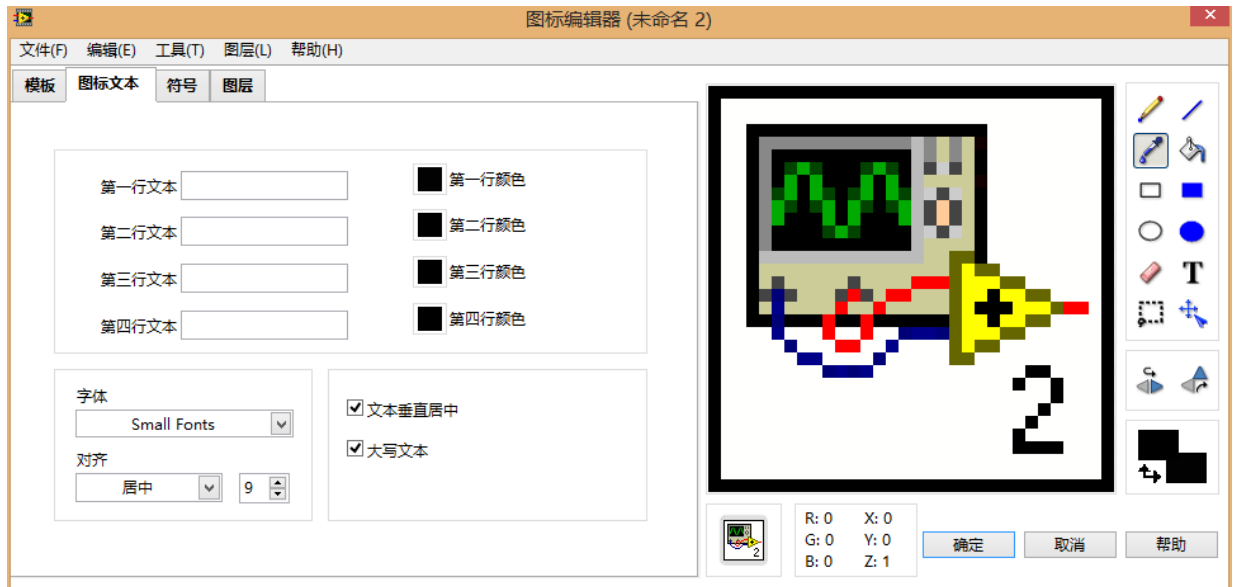
图？

⑦ 在前面板的连线框图中，将连线框图中的输入与输出格与 VI 中的输入和输出控件一一对应连接起来。如图？所示



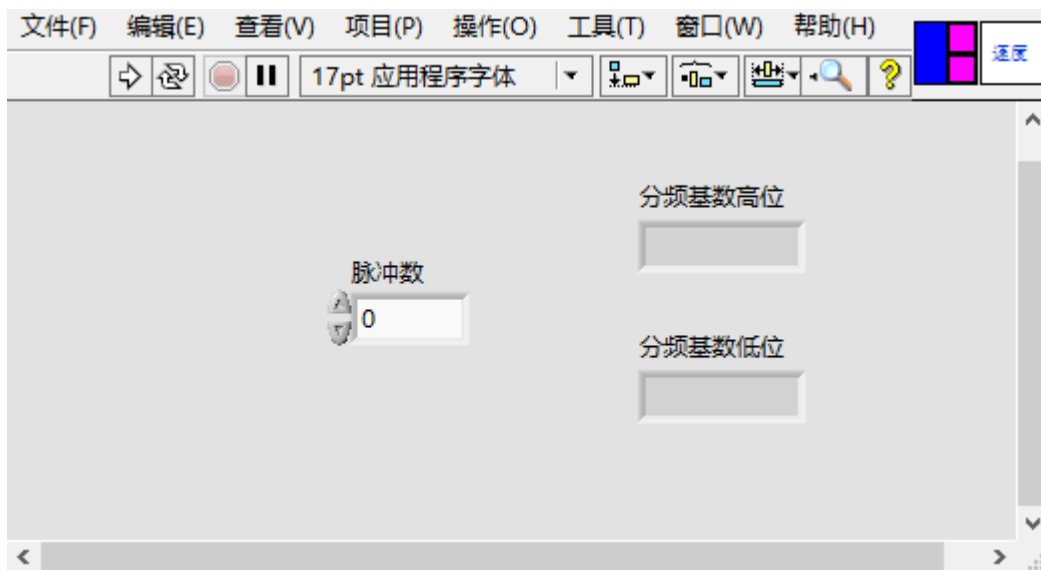
图？

⑧ 在前面板右击 VI 图标选择<编辑图标>。弹出如图？所示的对话框。



图？

使用快捷键<Ctrl+u>清除用户图层，再在“图标文本”项目下的“第一行文本”输入“速度”并改变字体颜色。完成后单击确定即可。完成后如图？所示。



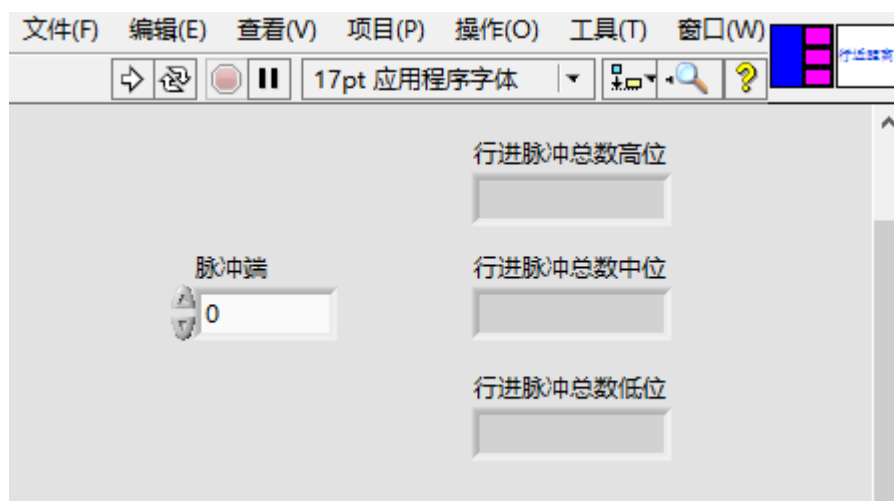
图？

⑨ 使用快捷键<Ctrl+s>将 VI 快速保存，并命名为“速度子 VI”。

(2) 创建行进距离子 VI

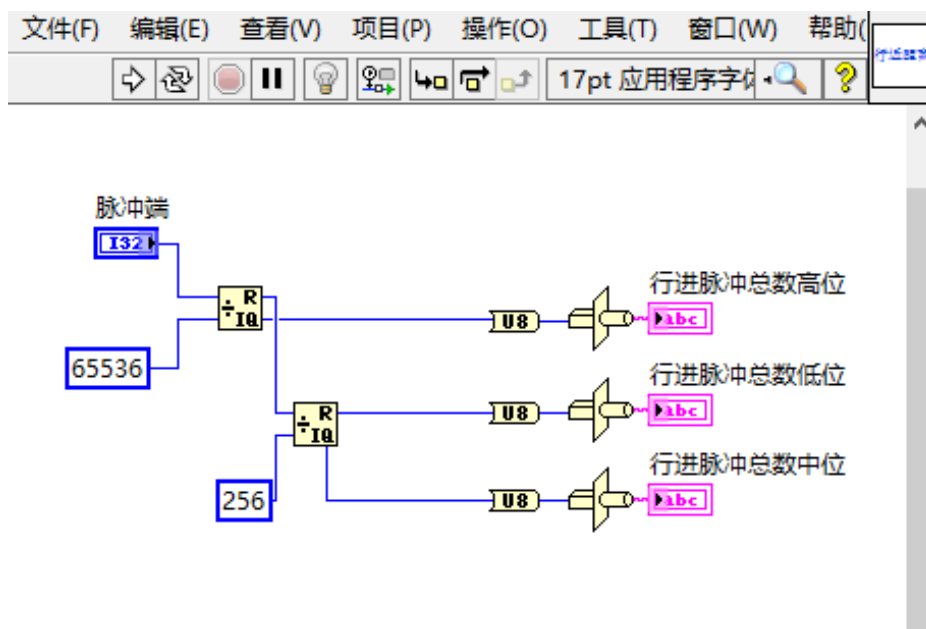
因与(1)中创建“速度子 VI”所用的控件和方法一致，故只列出最终效果图，详细步骤不再赘述。

① 行进距离子 VI 前面板如图？所示。



图？

② 行进距离子 VI 程序框图如图 7-1-10 所示。

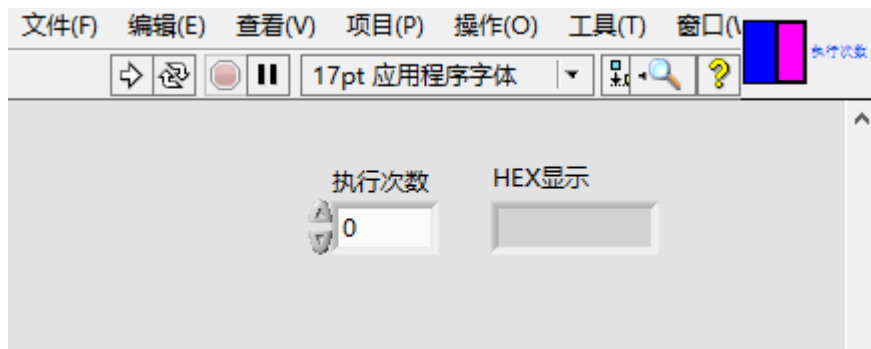


图？

(3) 创建执行次数子 VI

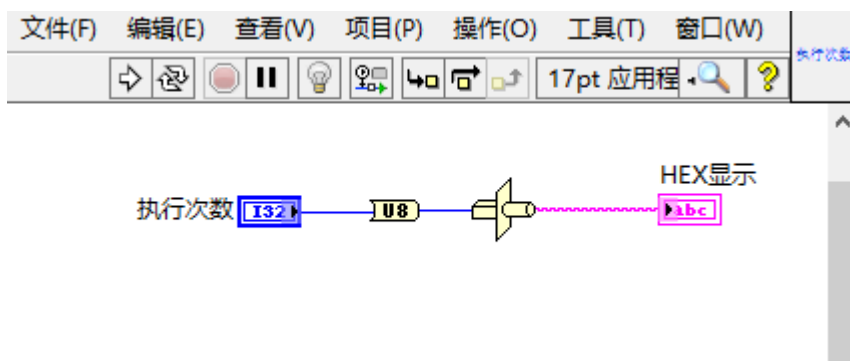
因与(1)创建“速度子 VI”所用的控件和方法一致，故只列出最终效果图，详细步骤不再赘述。

① 执行次数子 VI 前面板如图 7-1-10 所示 (HEX 显示即为 16 进制显示)。



图？

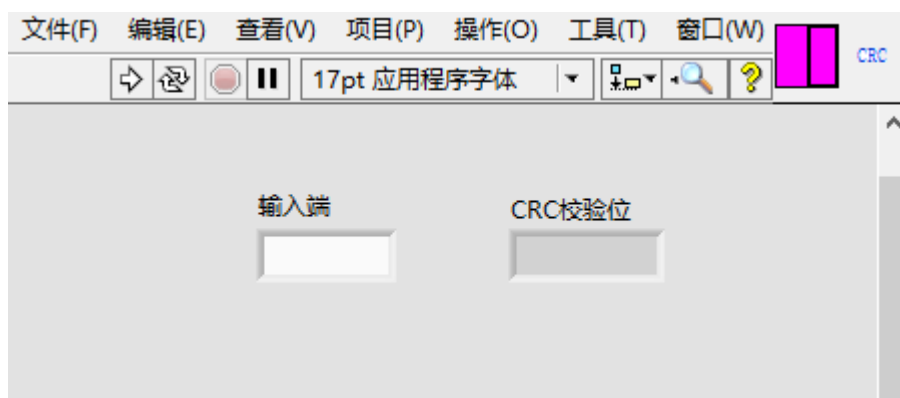
②执行次数子 VI 程序框图如图？所示。



图？

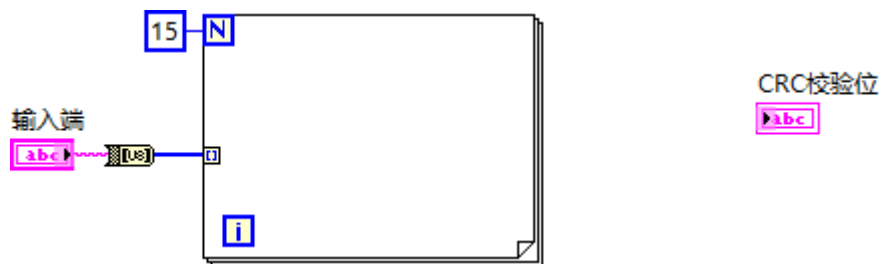
(4) 15 位 CRC 校验子 VI

① 新建一个 VI,在前面板创建【字符串输入控件】改名为“输入端”和【字符串显示控件】改名为“CRC 校验位”，分别右击【字符串输入控件】和【字符串显示控件】选择<十六进制显示>。编辑连线模式和图标，整理后如图？所示。



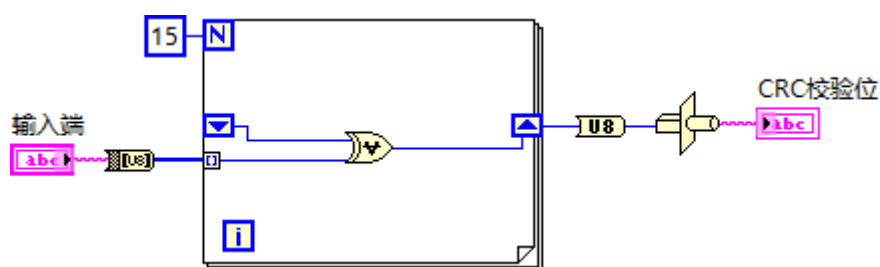
图？

② 在程序框图中创建【For 循环】并设定循环次数为 15、【字符串至字节数组转换】(【函数】/【编程】/【字符串】/【路径/数组...】/【字符串至字节数组转换】)



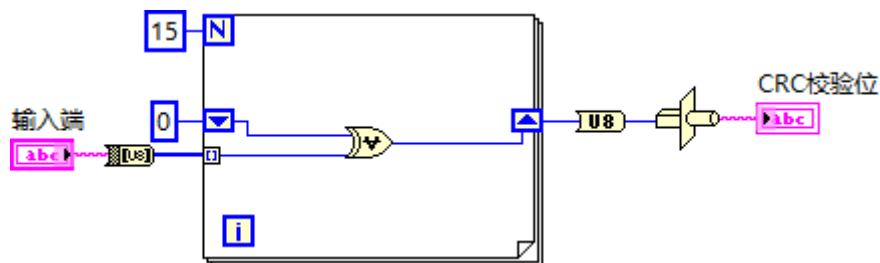
图？

- ③ 创建【异或】(【函数】/【编程】/【布尔】/【异或】)、【转换为无符号单字节整型】、【强制转换类型】等控件。并在【For 循环】中添加“移位寄存器”(右击【For 循环】选择<添加移位寄存器>)。连线方式如图？所示。



图？

- ④ 选中【For 循环】左边的移位寄存器，右击选择<创建>/<常量>



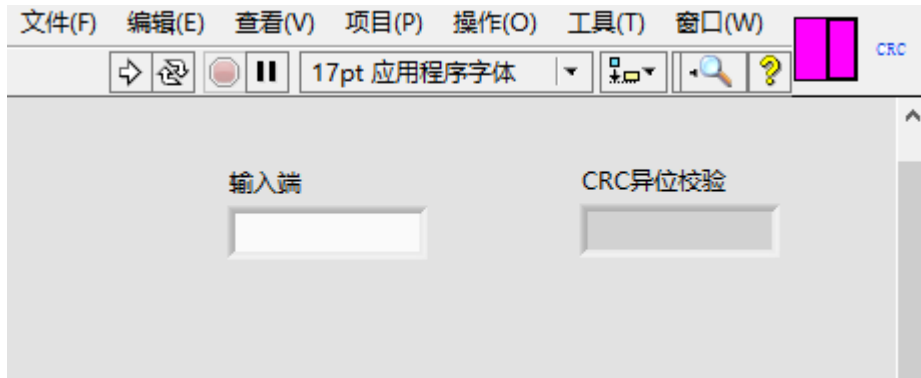
图？

- ⑤ 使用快捷键<Ctrl+s>将 VI 快速保存，并命名为“15 位 CRC 校验子 VI”。

(5) 创建 6 位 CRC 校验子 VI

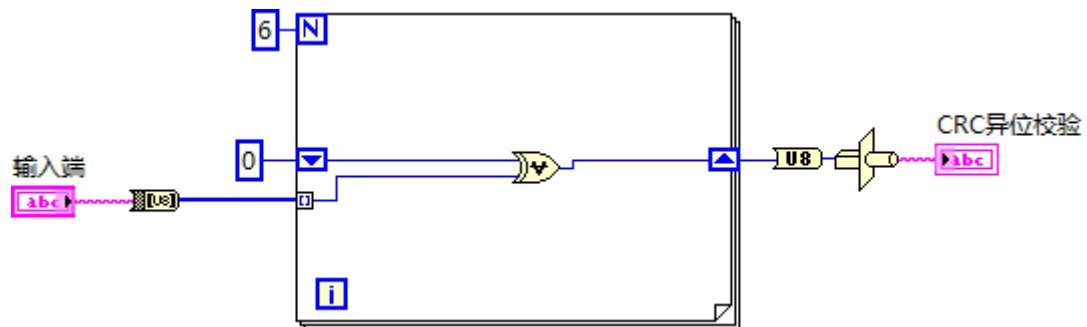
因与(4)“15 位 CRC 校验子 VI”所用的控件和方法一致，故只列出最终效果图，详细步骤不再赘述。

- ① 6 位 CRC 校验子 VI 前面板，如图？所示。



图?

② 6 位 CRC 校验子 VI 程序框图，如图? 所示。

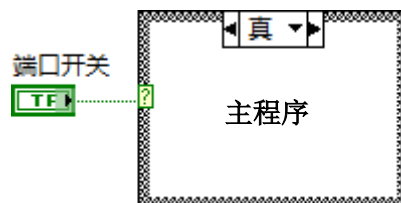


图?

(6) 外部【条件结构】处理

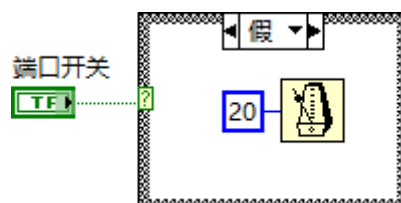
① 将【端口开关】与外层【条件结构】的“分支选择器”相连。

② 在“真”分支的框图内创建主程序结构。如图?所示。



图?

③ 在“假”分支的框架内创建延时，创建【等待下一个整数倍毫秒】设置等待时间为【20ms】，如图? 所示。

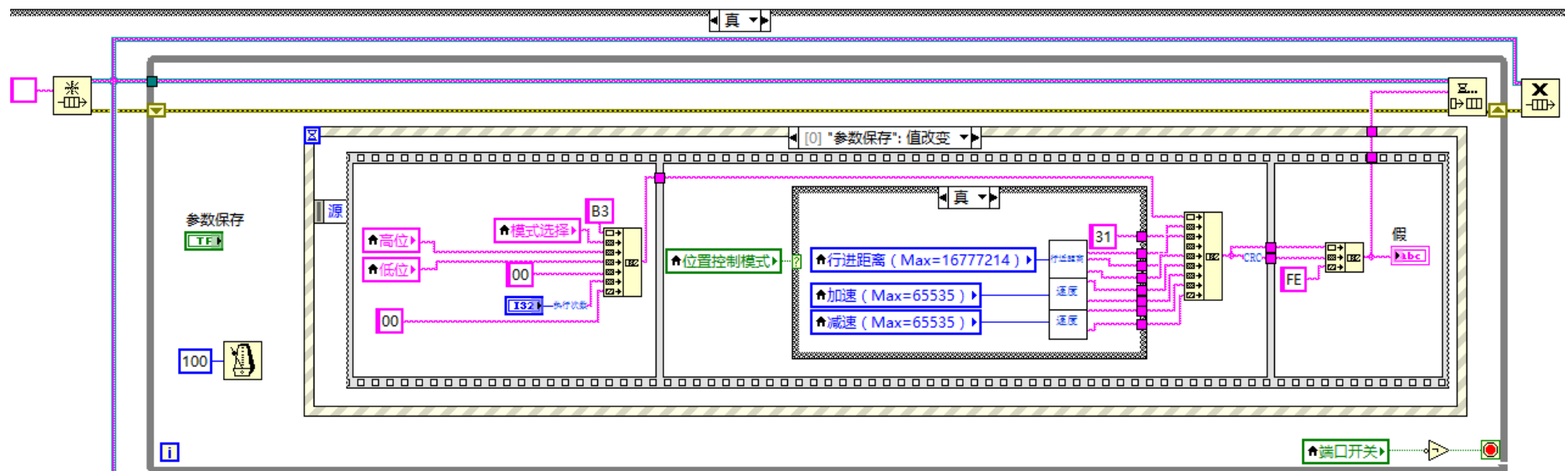


图?

(7) 设计参数发送与接收部分

该部分由两个小部分组成。第一部分为参数发送部分，第二部分为参数接收部分。

① 创建参数发送部分，如图？所示。

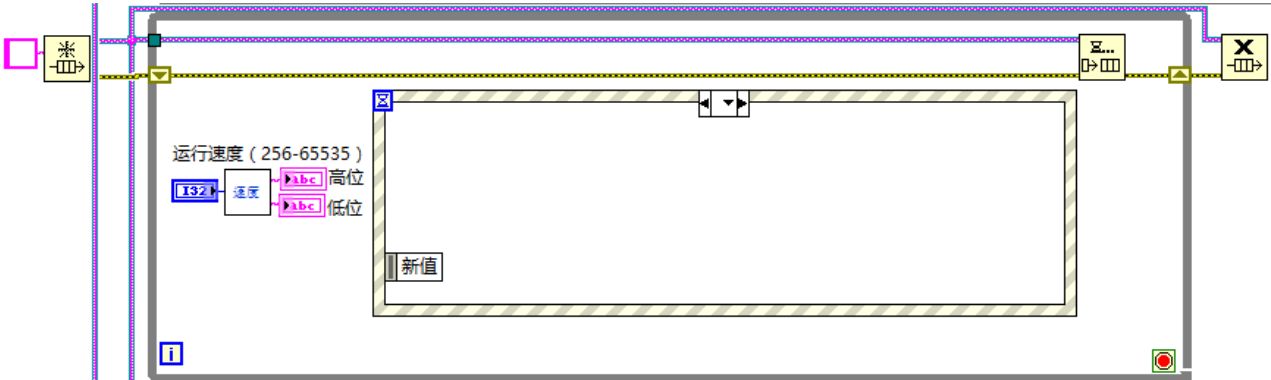


图？

a 在第 2 个【While 循环】中(自上而下)导入速度子 VI，在主程序的程序框图空白处右击，选择<选择 VI>，在先前保存的位置找到“速度子 VI”。

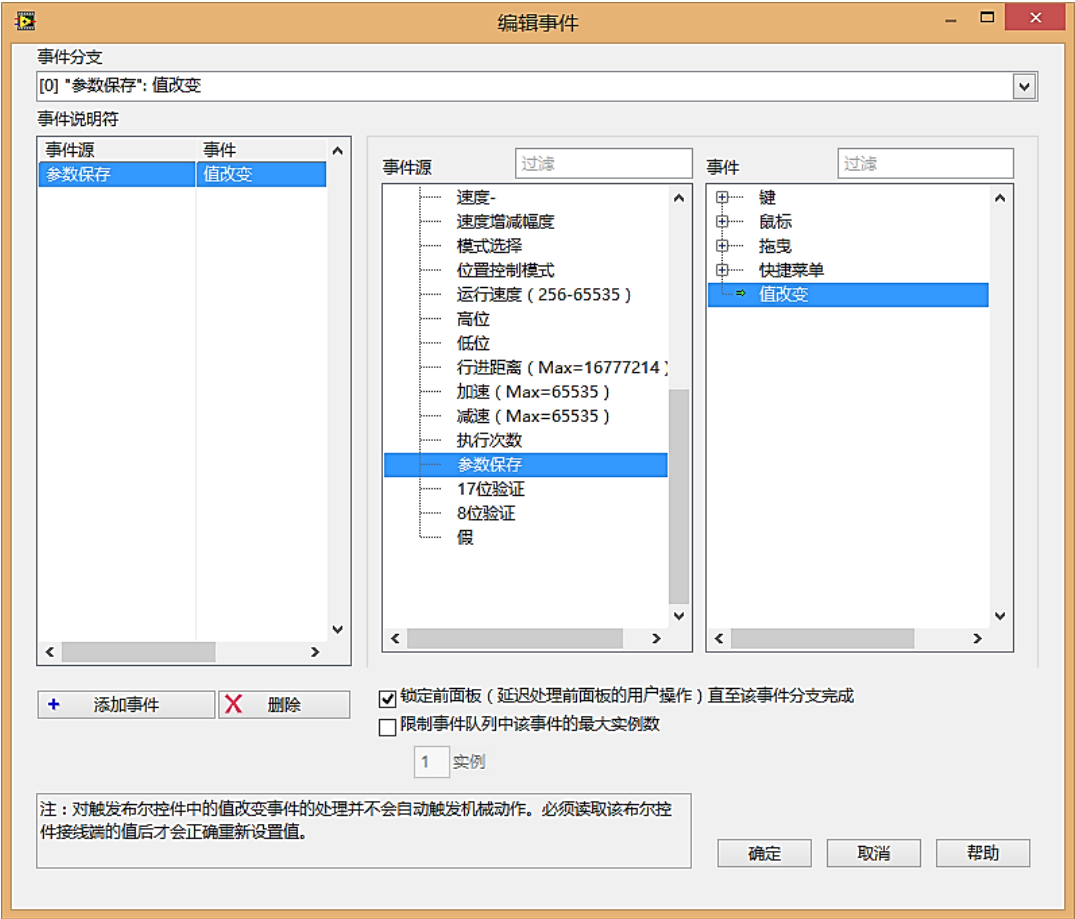
b 将【运行速度】与【速度子 VI】相连。在【速度子 VI】的输出端创建 2 个【字符串显示控件】为简化改名为“高位”和“低位”，结果

如图? 所示。



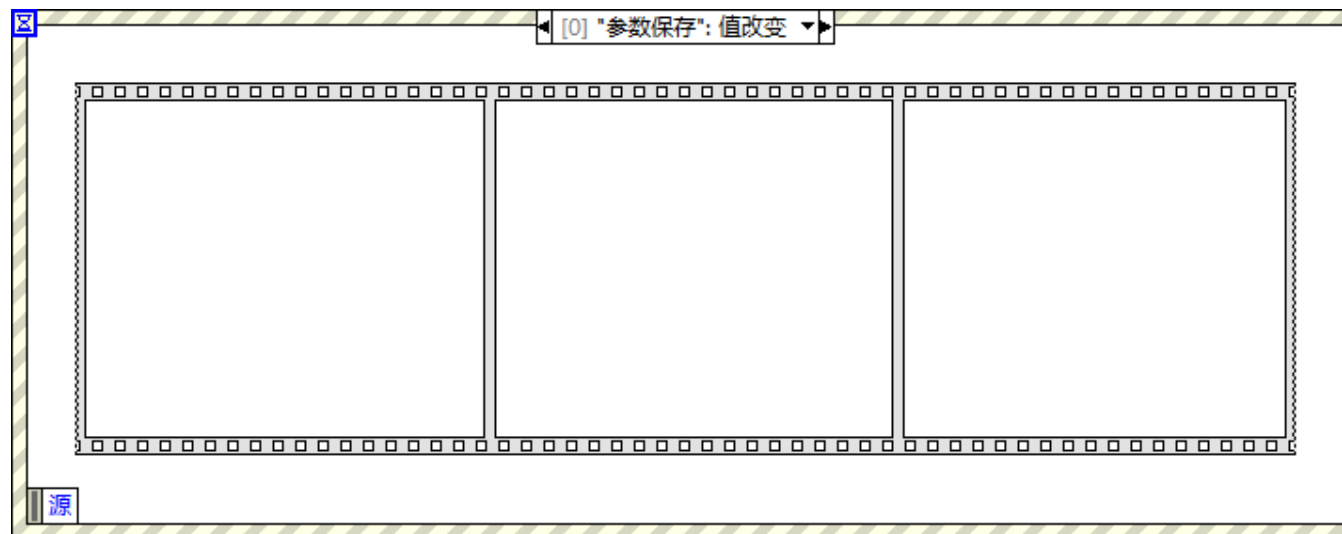
图?

c 选中第 1 个【While 循环】(自上而下)中的【事件结构】中“[0]”分支，右击<编辑本分支处理的事件...>，在“事件源”一列中选择“参数保存”，“事件”一列中选择“值改变”。如图? 。



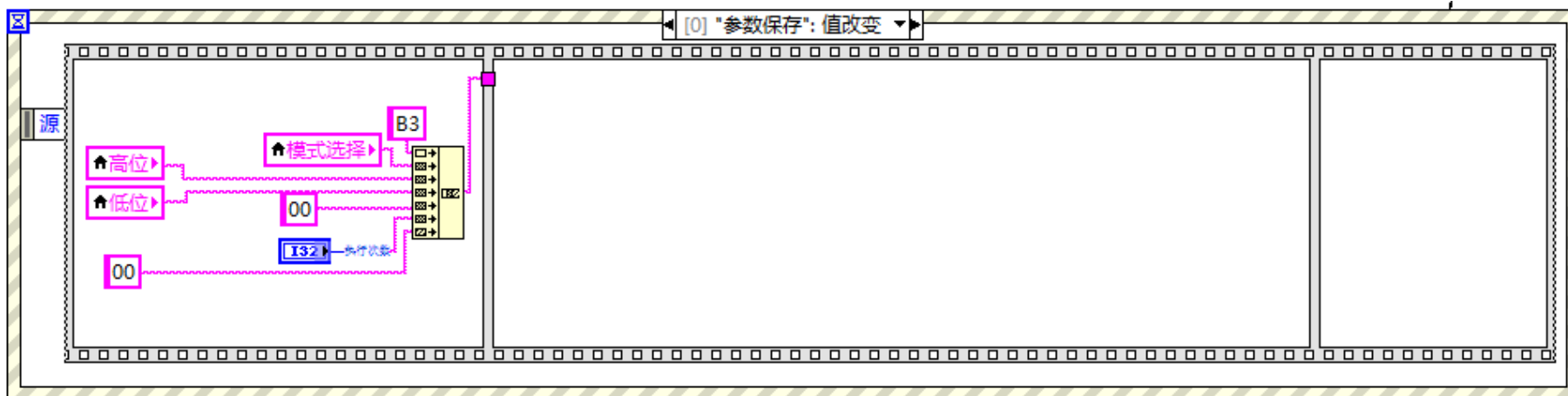
图?

在【事件结构】的[0]分支中里，创建 3 段【平铺式顺序结构】结果如图? 所示。



图?

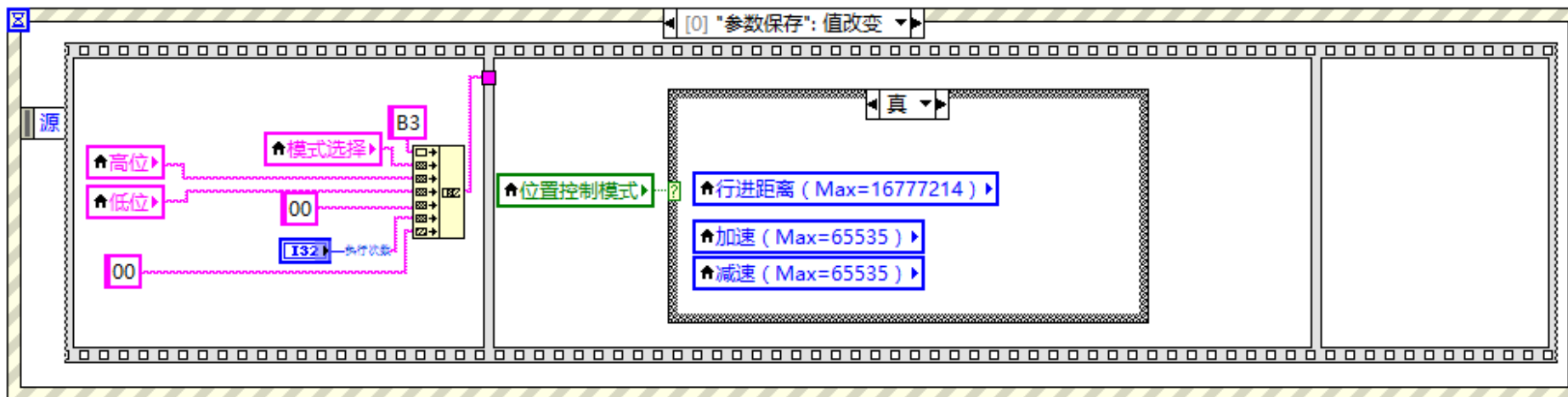
- d 分别右击【高位】、【低位】、【模式选择】。依次选择<创建>/<局部变量>，分别右击 3 个【局部变量】选择<转化为写入>。
- e 创建 3 个【字符串常量】(【函数】/【编程】/【字符串】/【字符串常量】)，依次右击 3 个【字符串常量】选择<十六进制显示>，并输入 B3、00、00。
- d 创建【执行次数子 VI】。
- f 创建【连接字符串】(【函数】/【编程】/【字符串】/【连接字符串】)。
- g 在【平铺式顺序结构】的第一个顺序中，连接各个控件,结果如图?所示。



图？

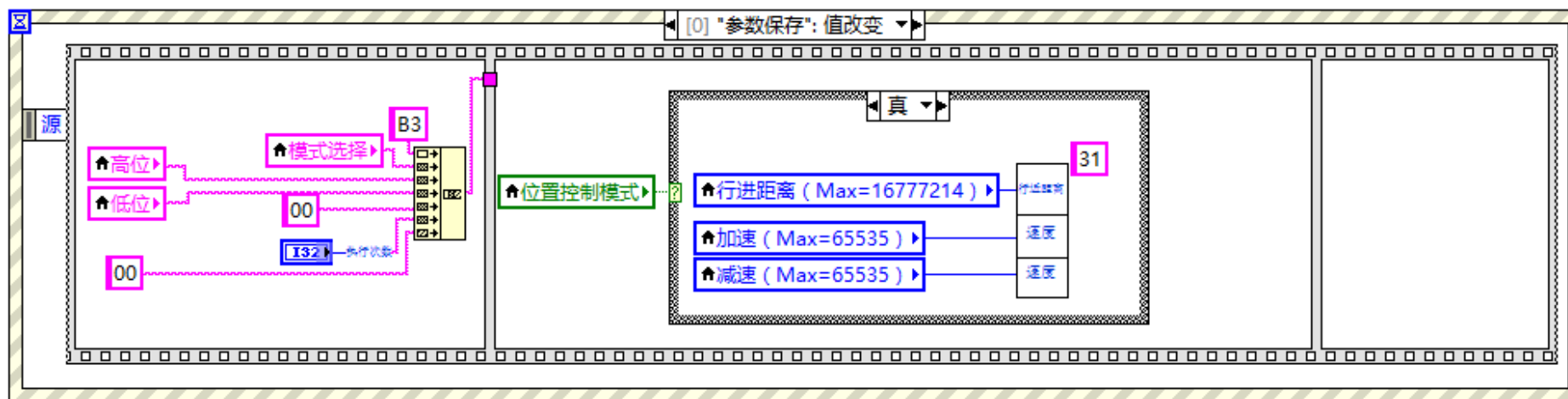
h 在【平铺式顺序结构】的第二个顺序中，创建一个【条件结构】。

j 分别右击【位置控制模式】、【行进距离】、【加速】、【减速】依次选择<创建>/<局部变量>，分别右击 4 个【局部变量】选择<转化为写入>。如图？所示。



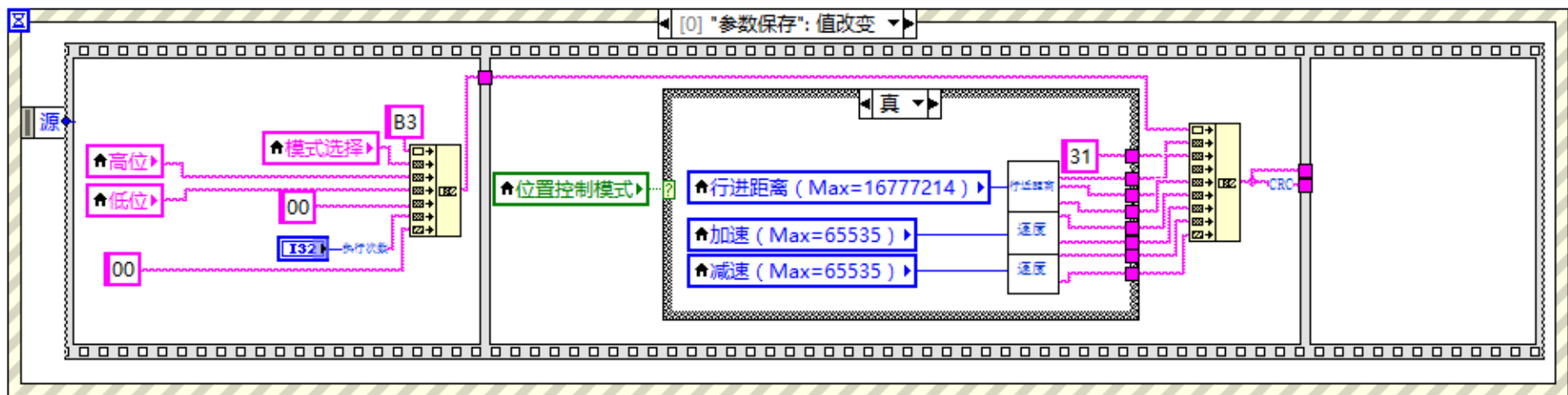
图？

k 将【位置控制模式局部变量】与【条件结构】相连。在【条件结构】的“真”分支中创建【行进距离子 VI】和 2 个【速度子 VI】，以及一个值为 31(16 进制的【字符串常量】)。如图？所示。



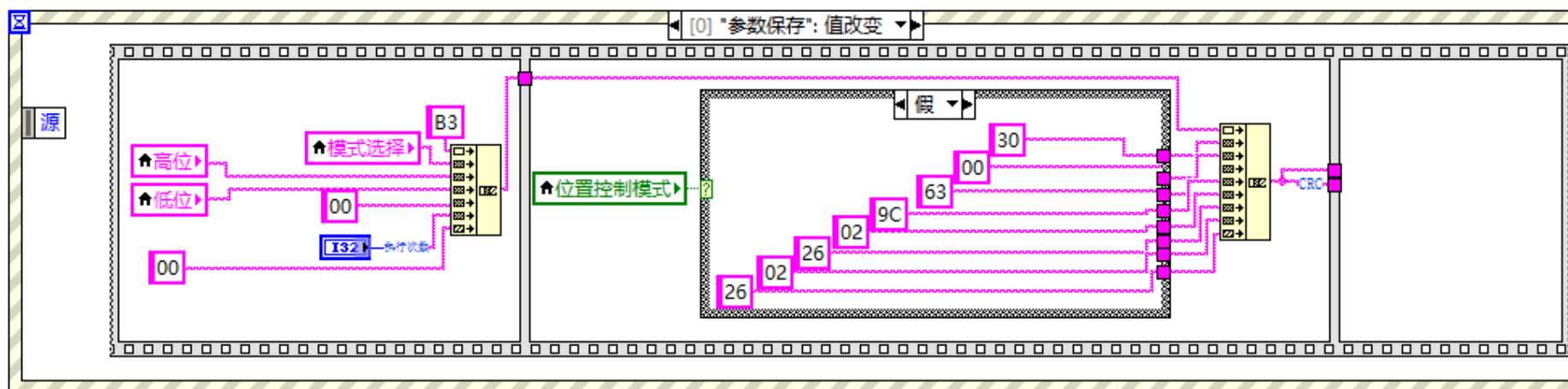
图？

1 创建【连接字符串】(【函数】/【编程】/【字符串】/【连接字符串】)和【15 位异位校验子 VI】，连接相关端口，结果如图？所示。



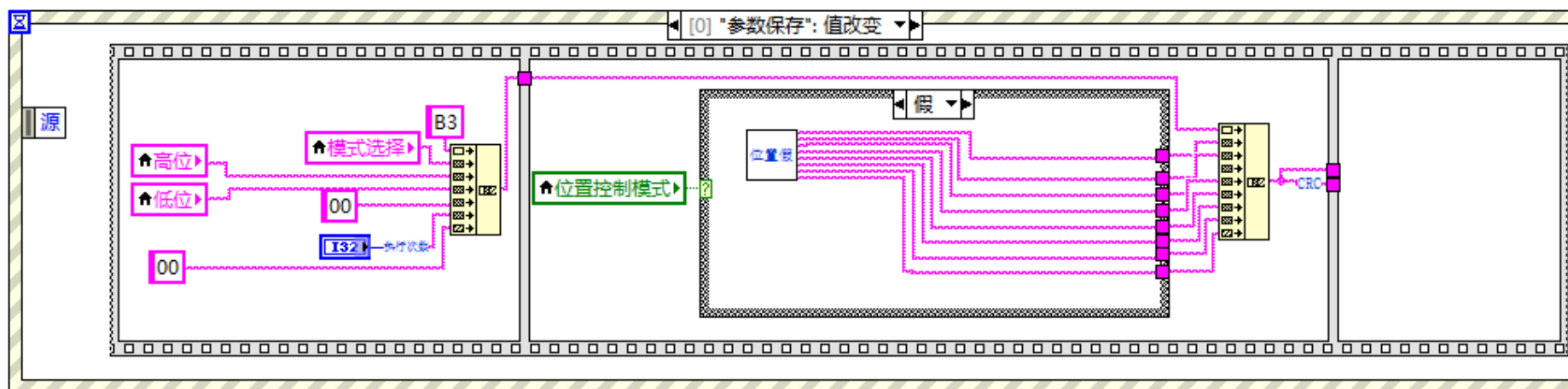
图？

m 在【条件结构】的“假”分支中创建 8 个【字符串常量】，右击这 8 个【字符串常量】选择<十六进制显示>，并依次赋值 30、00、63、9C、02、26、02、26。如图？所示。



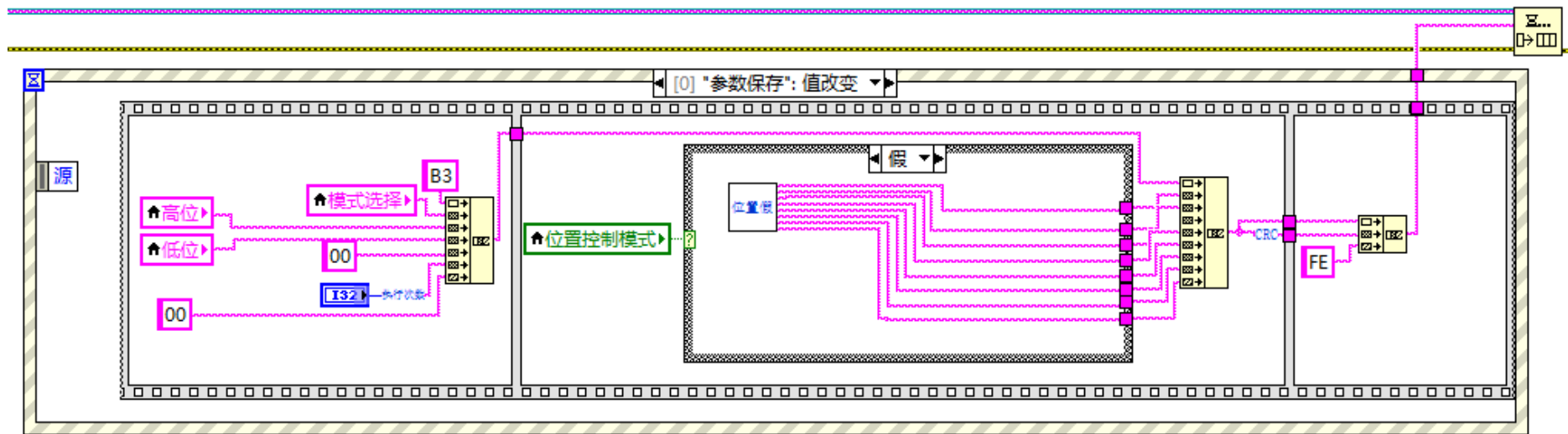
图?

为了使界面显得简化，也可创建一个子 VI。如图 7-10 所示。



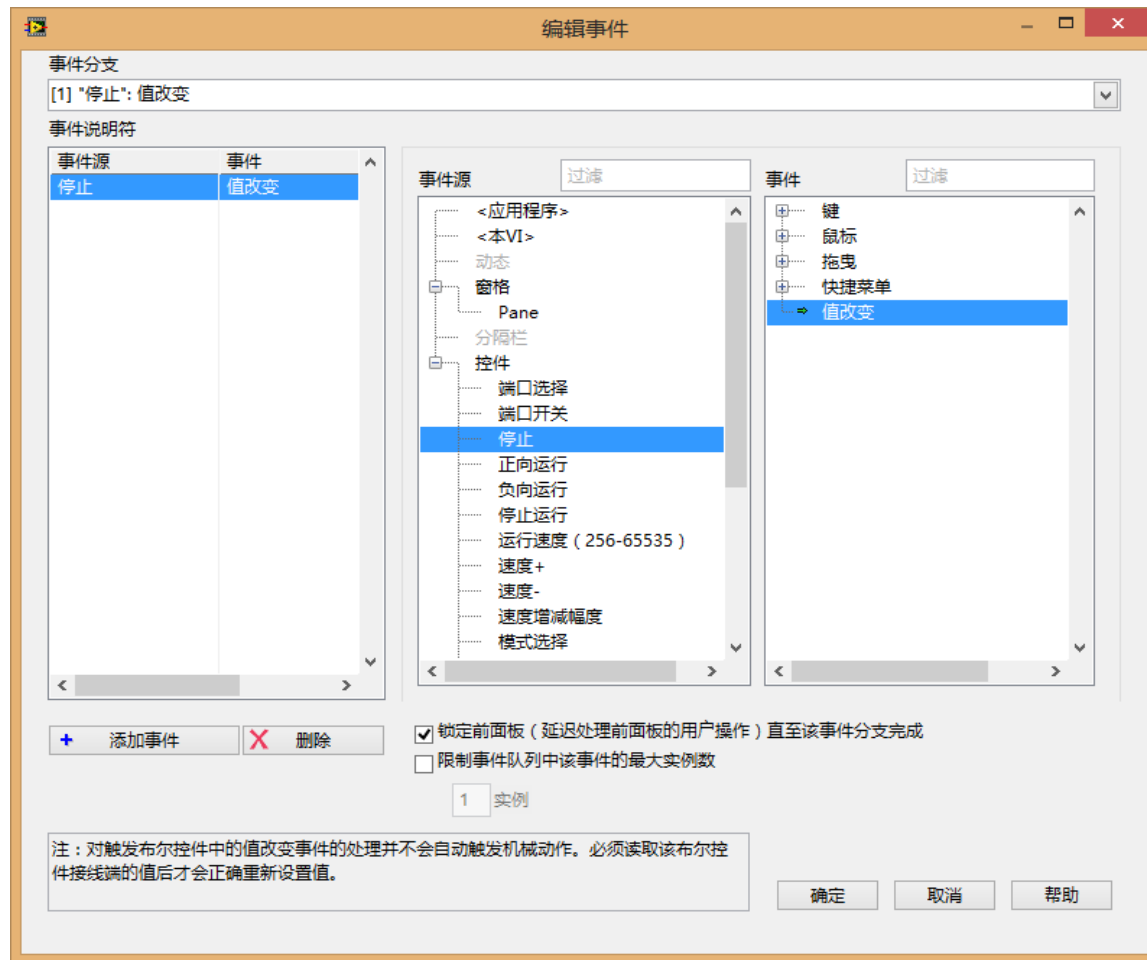
图?

n 在【平铺式顺序结构】的第三个顺序中，创建【连接字符串】和【字符串常量】，选择<十六进制显示>，并赋值 FE。连接相关线段，并将【连接字符串】的输出端连接至【元素入阵列】的“元素”端，结果如图? 所示



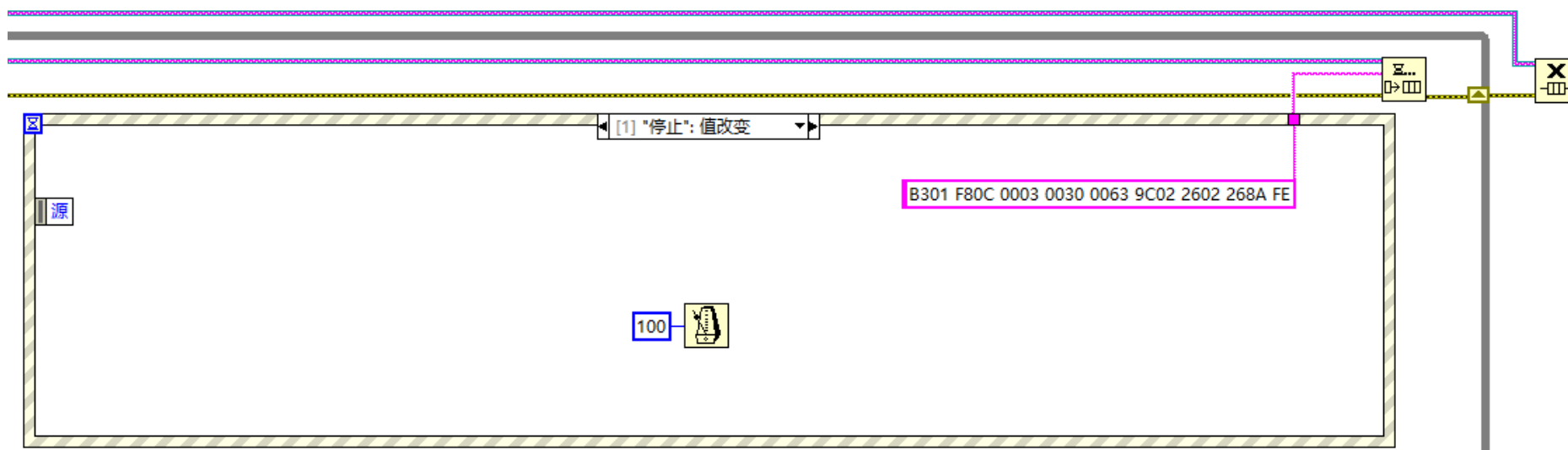
图?

o 选择【事件结构】“[1]”分支右击<编辑本分支处理的事件...>，在“事件源”一列中选择“停止”，“事件”一列中选择“值改变”。如图?



如图？

在该分支下创建一个【字符串常量】，右击这个【字符串常量】选择<十六进制显示>，并赋 B3 01 F8 0C 00 03 00 30 00 63 9C 02 26 02 26 8A FE。连接至【元素入阵列】的“元素”端。创建相应延时。如图？所示。



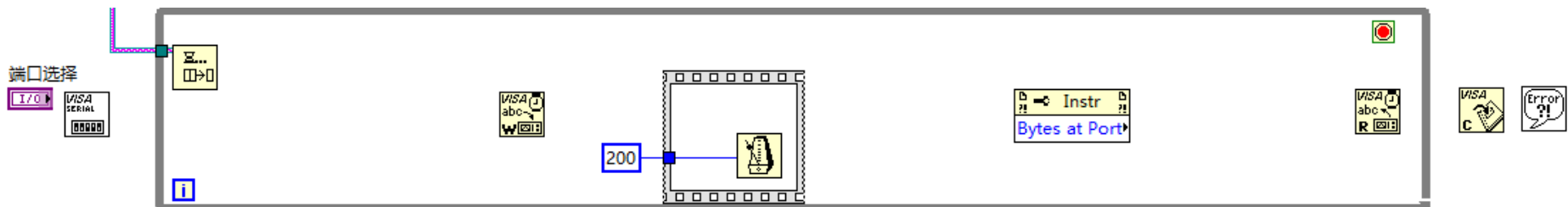
② 参数接收部分(对于 VISA 的说明，详细部分请参见附录部分)

a 在第 3 个【While 循环】(自上而下)中依次创建【VISA 配置串口】(【函数】/【数据通信】/【协议】/【串口】/【VISA 配置串口】)、【VISA 写入】、【VISA 读取】、【VISA 读取】、【VISA 串口字节数】、【VISA 关闭】、【简易错误处理器】(【函数】/【对话框与用户界面】/【简易错误处理器】)等控件。如图？所示。



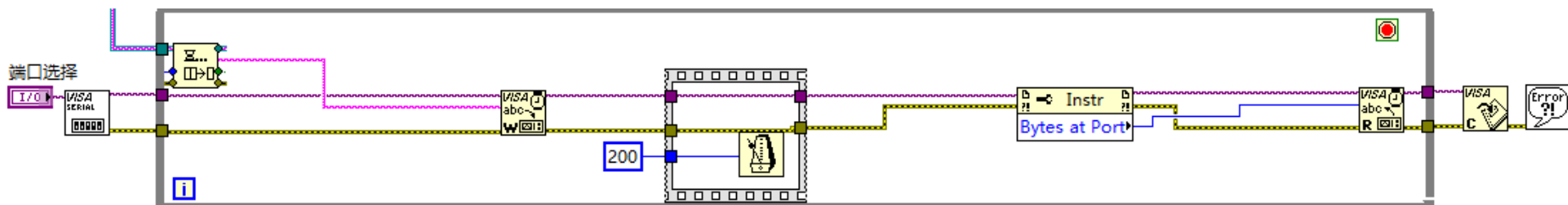
图？

b 为了使是数据能够读取完全，因此需要创建延时。本【While 循环】中创建由【平铺式顺序结构】和【等待下一个整数倍毫秒】设置等待时间为【200ms】构成。如图？所示。



图？

c 依次连接相关端口，如图？所示。



图?

(8) 设计命令发送与接收部分

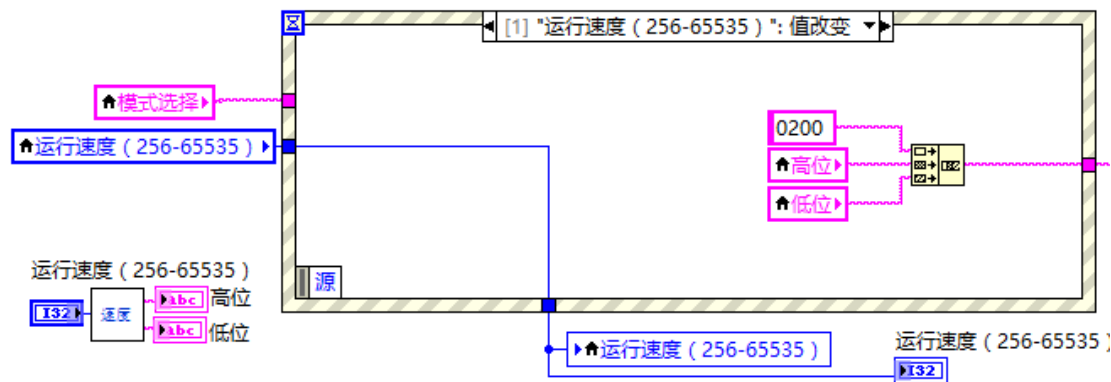
该部分由两个小部分组成。第一部分为命令发送部分，第二部分为命令接收部分。

①命令发送部分

a 选中第 2 个【While 循环】(自上而下)中的【事件结构】中的“[0]”分支，右击<编辑本分支处理的事件...>，在“事件源”一列中选择“模式选择”，“事件”一列中选择“值改变”。如图? 所示。

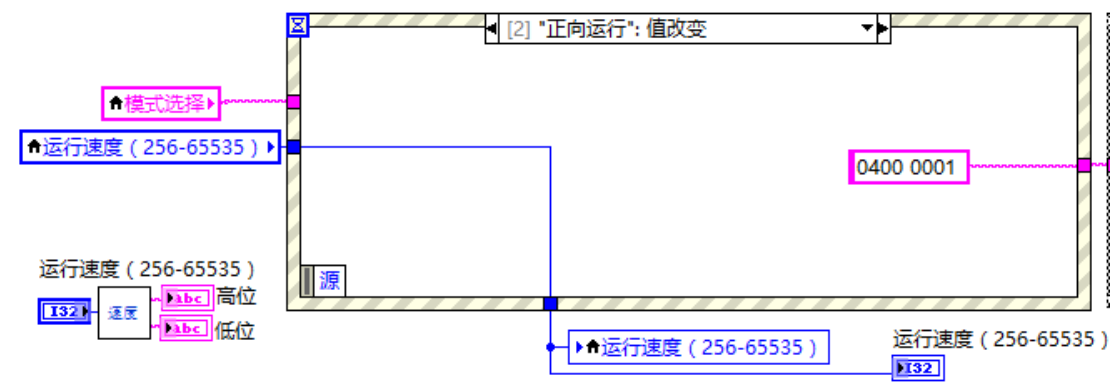
选择“值改变”。

f 在【事件结构】中的“[1]”分支“运行速度”中创建 1 个【字符串常量】，以十六进制显示，并赋值为 01 00，创建【高位局部变量】、【低位局部变量】，转为写入，再创建一个【连接字符串】。详细连线方式如图 ？所示。



图？

g 在【事件结构】中的“[2]”分支“正向运行”中创建 1 个【字符串常量】，以十六进制显示，并赋值为 04 00 00 01。详细连线方式如图 ？所示。



图？

h 在【事件结构】中的“[3]”分支“负向运行”中创建 1 个【字符串常量】，以十六进制显示，并赋值为 04 00 00 02。详细连线方式如图 ？所示。

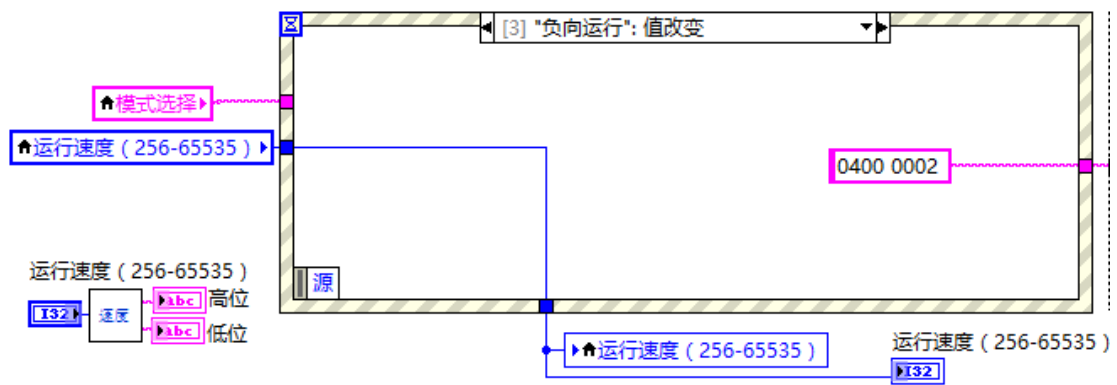


图 3-10

i 在【事件结构】中的“[4]”分支“停止运动”中创建 1 个【字符串常量】，以十六进制显示，并赋值为 04 00 00 03。详细连线方式如图 3-11 所示。

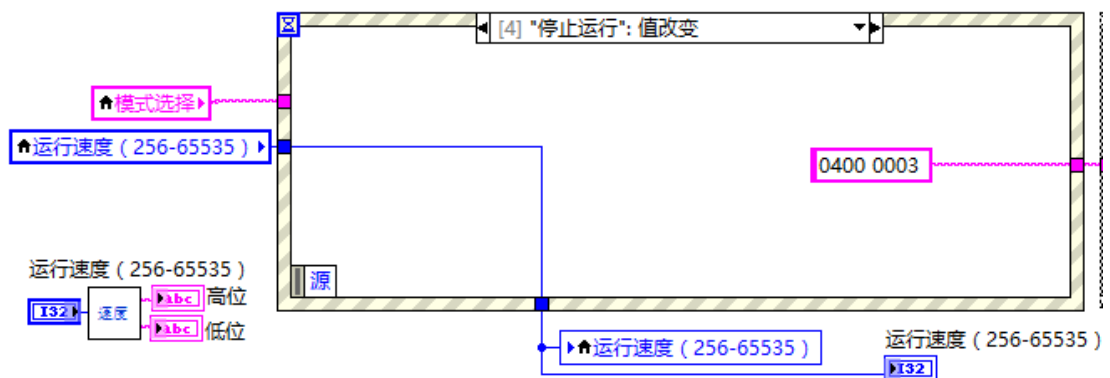


图 3-11

j 在【事件结构】中的“[5]”分支“位置控制”中创建 1 个【字符串常量】，以十六进制显示，并赋值为 06，创建【行进距离子 VI】和【连接字符串】，将【行进距离】与【行进距离子 VI】相连。详细连线方式如图 3-12 所示。

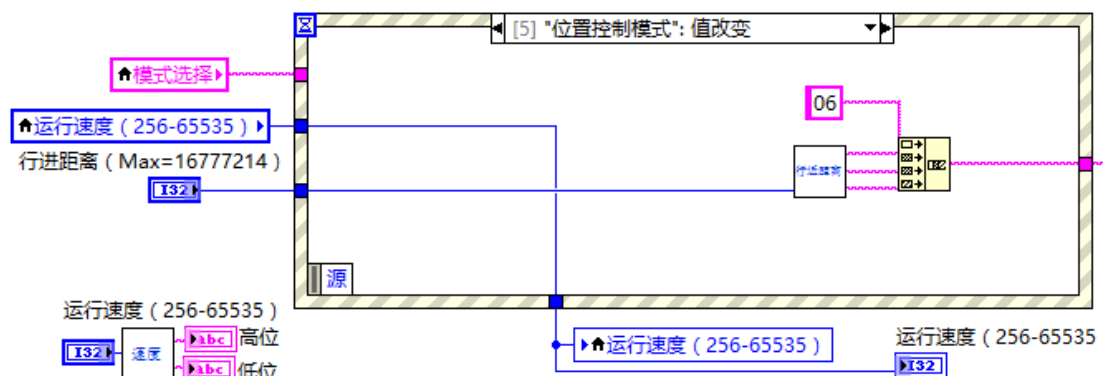
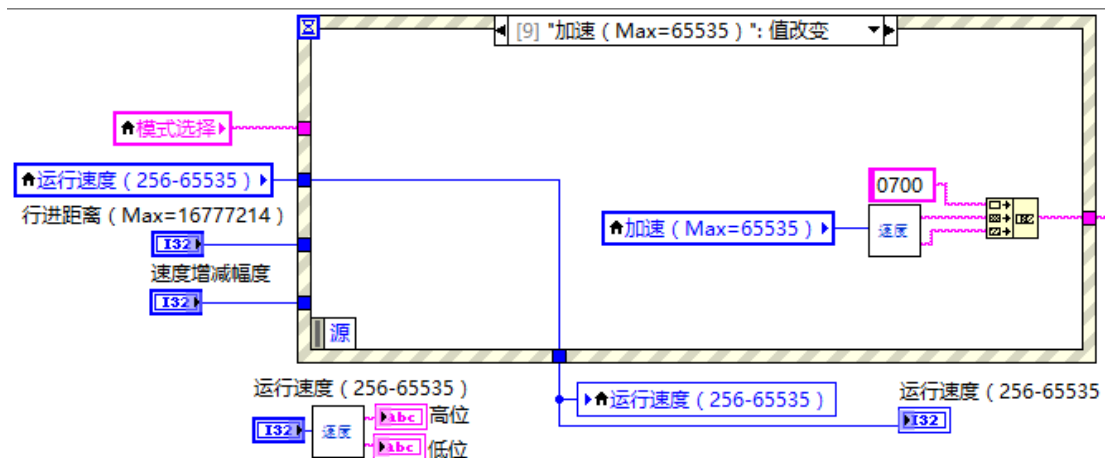


图 3-12

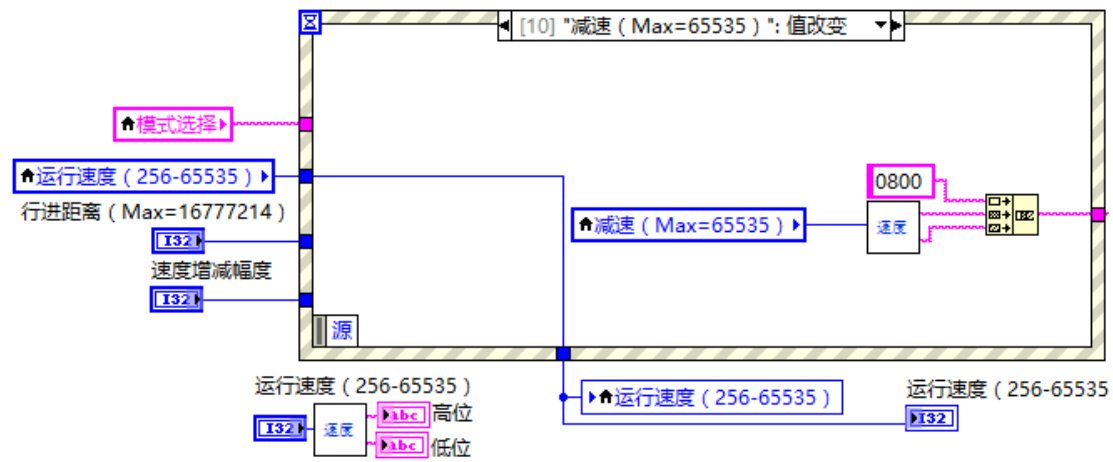
k 在【事件结构】中的“[6]”分支“速度+”中创建 1 个【字符串常量】，以十六进

[illegible]

n 在【事件结构】中的“[9]”分支“加速”中创建 1 个【字符串常量】，以十六进制显示，并赋值为 07 00，创建【加速局部变量】、【速度子 VI】和【连接字符串】，详细连线方式如图 7-10 所示。



o 在【事件结构】中的“[10]”分支“减速”中创建 1 个【字符串常量】，以十六进制显示，并赋值为 08 00，创建【减速局部变量】、【速度子 VI】和【连接字符串】。详细连线方式如图 7-10 所示。



图？

p 在【事件结构】位置后创建【平铺式顺序结构】，在【平铺式顺序结构】内，创建 2 个【字符串常量】，以十六进制显示，并依次赋值为 BA 00、FE。创建 2 个【连接字符串】，最终的输出连接至该【While 循环】中的【元素如阵列】的“元素”端。详细连线方式如图 ？所示。

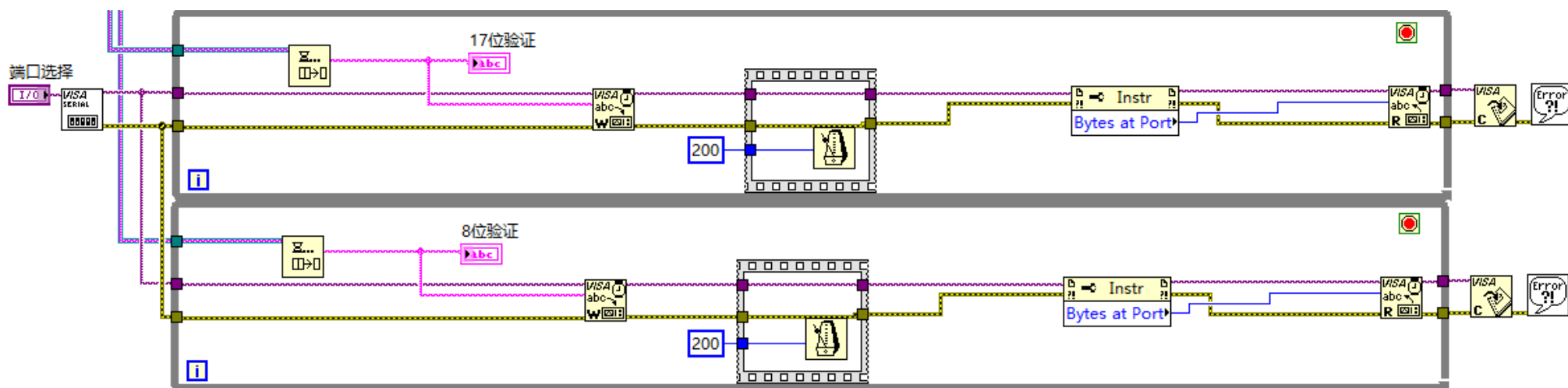


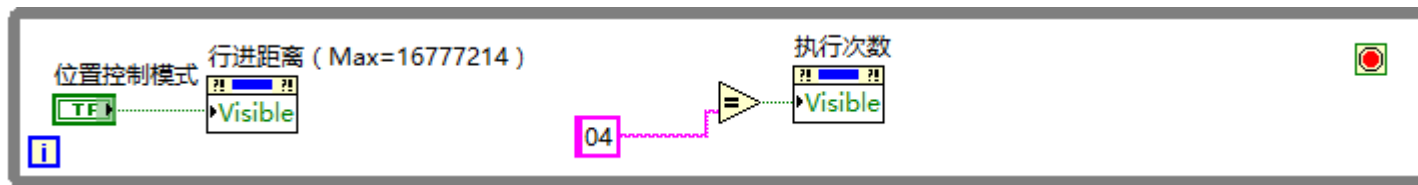
图7

(9) 位置控制及执行次数命令的显隐

① 在最底层的【While 循环】中，在【行进距离】和【执行次数】上右击选择<创建>/<属性节点>/<可见>。如图8所示。

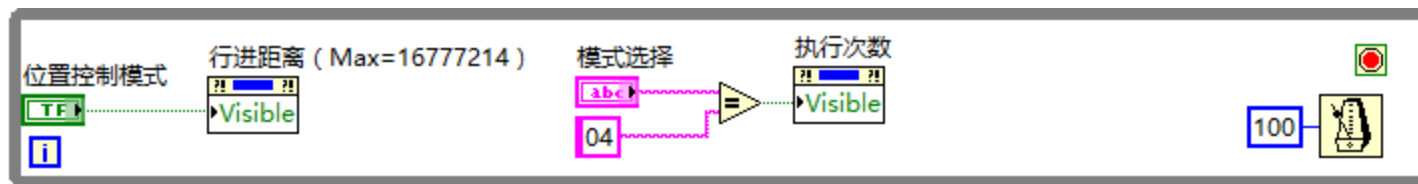


② 创建【字符串常量】，以十六进制显示，并赋值为 04。创建一个【等于】控件。如图9所示。



图？

③ 创建相应的延时，详细连线步骤如图？所示。



图？

④ 考虑到【行进距离】和【执行次数】在程序开始的显隐性。在最外层【While】创建它们的【可见属性节点】并用【假常量】与它们相连，并将最外层【While】的“循环条件”与【停止】控件相连。如图？所示。

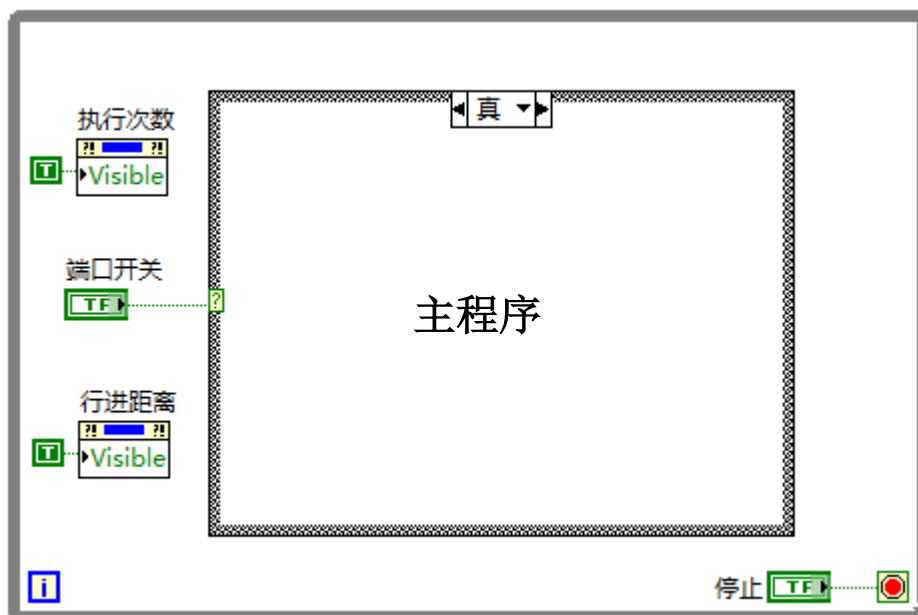


图 3-10

(10) 细节完善

① 程序中【While 循环】缺少停止控件。创建 5 个【端口开关局部变量】和 5 个【非】控件(【函数】/【布尔】/【非】)，在主程序的 5 个【While 循环】中，创建如图 3-11 的停止连线方式。



图 3-11

② 在没有设置延时的【While 循环】中(除了“参数接收”和“命令接收”的两个循环)创建【等待下一个整数倍毫秒】设置等待时间为【100ms】，如图 3-12 所示。

3.4 附录

(1) 指令(16 进制)

17 位 16 进制的来源，每款步进电机控制器都有自己的指令。以本次实验的步进电机驱动器(便能 BE-1108 型)为例。列出 17 位指令的具体含义，方便理解。

此指令即为参数发送的编程依据。

指令：B3 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 FE

(指令值为 16 进制数，通讯波特率 9600，数据位 8，无效验位，停止位 1)

第一位——为数据头 B3:储存参数指令 B9: 储存参数并改变控制编号指令

第二位——模式 00 :点动模式 01:单步模式 02:自动模式 03:单次往返 04:按次

往返 05:按键回零单方向 06:上电回零单方向 07: 上电运行单方向

第三位——分频基数高位

第四位——分频基数低位 (最大值为 65535,当设置为 00 时,启动外部电位器调速)

第五位——控制器编号-地址位(最大值为 254)

第六位——执行次数(提示: 只在 04:按次往返 模式时有效)

第七位——控制按钮值 01:正运行 02:负运行 03:停止

第八位——急停开关默认状态、限位开关默认状态、启动位置控制使能、启动上电回零使能 本字节拆分数据位 8 位 11111111 (1: 代表常开、使能启动 0: 代表常闭、不启动) 如例: 11111111 (例中全部有效位) 代表: 依次排序从左至右位, 启动 0.5 倍频率输出、启动上电回零使能、急停常开、限位常开、启动 0.2 倍频率输出、启动单开关触发、启动输入开关失效、启动位置控制使能、 例中串口应发数据: FF

第九位——行进脉冲总数高位

第十位——行进脉冲总数中位

第十一位——行进脉冲总数低位 (最大值为 16777214)

第十二位——加速脉冲数高位

第十三位——加速脉冲数低位 (最大值为 65535)

第十四位——减速脉冲数高位

第十五位——减速脉冲数低位 (最大值为 65535)

第十六位——CRC 校验位(第一位到第十五位值逐位异或,例: $j=a^b \wedge c^d \wedge e^f \wedge g^h$)

第十七位——FE 数据尾

(2) 实时指令(16 进制)

此部分即为命令发送的编程依据

BA 02 03 04 05 06 07 FE 第一位——为数据头 BA

第二位——控制器编号 第三位——数据修改项 01: 模式 02: 分频基数 03:

控制输出口 04 运行命令 05: 启动位置控制 06: 脉冲总数 07: 加速脉冲 08:

减速脉冲 09: 设置输出模拟电压 第四位 第五位 第六位——数据位 如果需要

修改的数据位为 3 位时 (如: 脉冲位置数据)分别对应 高、中、低位 如果数据位。如果要修改的数据位为 2 位时 (如: 加减速 模拟输出电压)第四位发 0 即可, 第五位, 六位 分别对应数据的高位和低位数据。如果要修改的数据位 1 位时(如: 模式)第四位第五位发 0 即可, 第六位发数据

第七位——CRC 校验位(第一位到第六位值逐位异或,例: $j=a^b^c$)

第八位——FE 数据尾

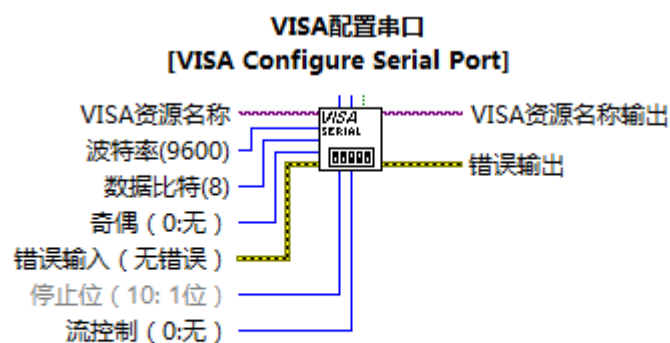
(3) 操作说明

- ① 端口选择
- ② 打开端口开关
- ③ 进行参数设置
- ④ 单击保存参数(每一次调整参数后都需要保存参数)
- ⑤ 执行命令
- ⑥ 关闭端口开关
- ⑦ 停止程序

(4) 有关 VISA 串口通信的简介

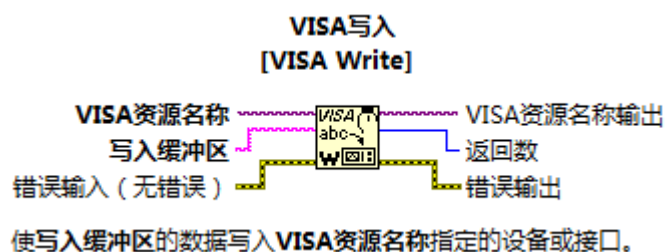
- ① 串口通信中经常使用的控件, 有如下几个。

a

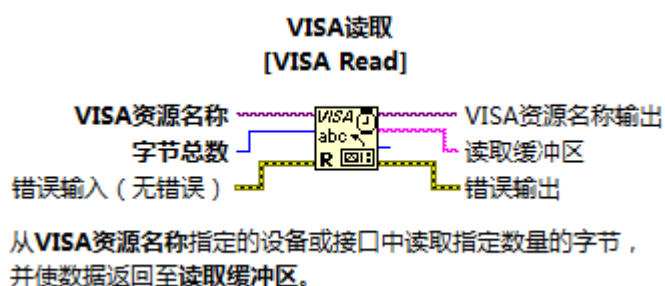


使VISA资源名称指定的串口按特定设置初始化。通过连线数据至VISA资源名称输入端可确定要使用的多态实例, 也可手动选择实例。

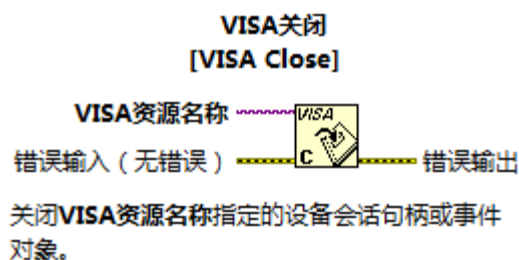
b



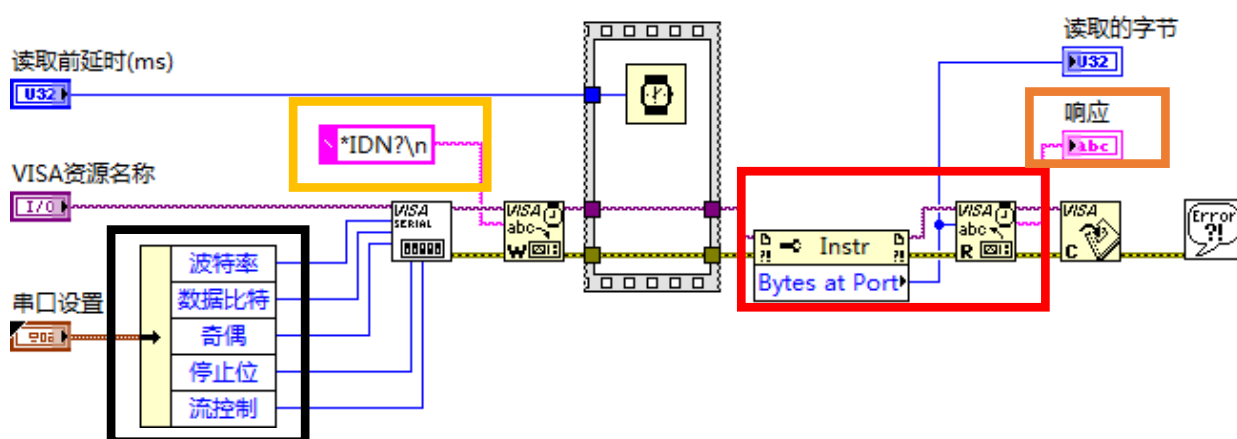
c



d



②主要使用的结构为



本结构为：LabView 中自带案例，打开方法为:在工具栏中<帮助>/<查找范例>/<硬件输入与输出>/<串口>/<简单串口>

a 红色框部分，主要用于接收【VISA 读取】的数据。是 VISA 通信中最为常见

的结构。

b 延时的设置是为了防止【VISA 写入】的数据量太大，读取部分跟不上，出现数据丢失的情况。

c 黑色框部分为串口的一些基本设置。

d 黄色框部分为串口指令的发送部分，通常为第三方所需的指令。在本教程中即为参数发送及命令发送所需的指令。

⑤ 棕色框部分为串口接收后的响应部分。在本教程中即为控制器的反馈部分。

综上，对于 VISA 串口通信，大家只需知道基本结构，即可完成大部分的任务。主要部分为数据写入过程中的指令部分，这需要大家对于一些基础知识能够很好的掌握。有条件的情况下，可选择一下简单的硬件进行尝试。同时也可以对 LabView 自带的案例进行深入了解和学习。