TSMaster 应用笔记

AN0006

How to use TOSUN Automation Module

如何使用同星自动化模块编程

作者: Link

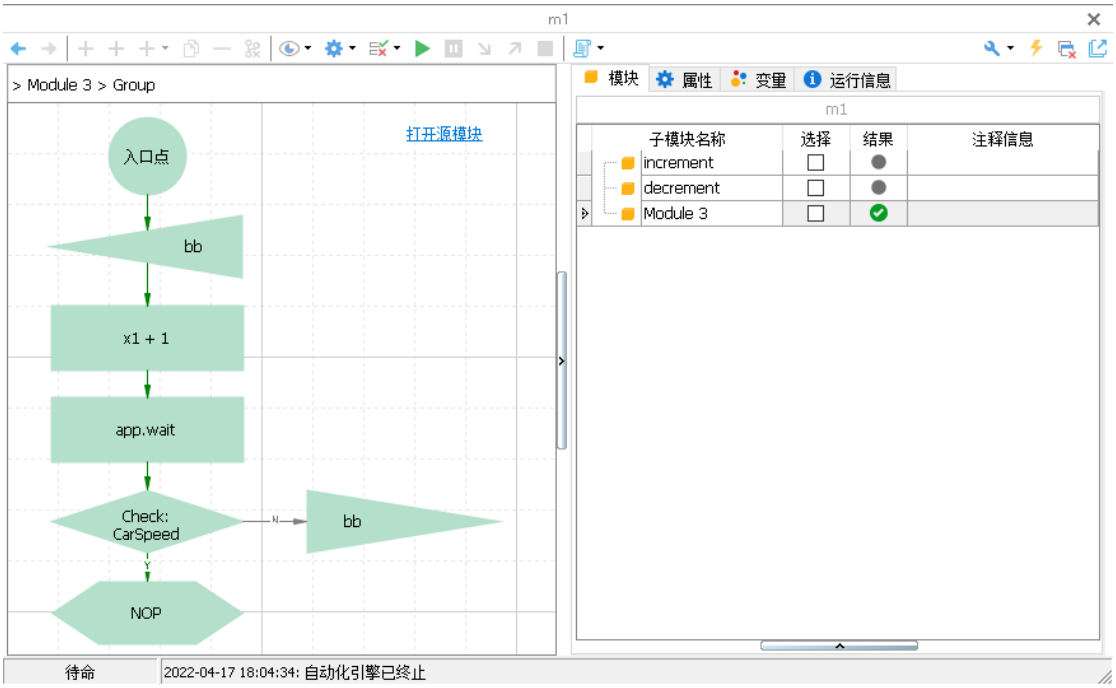
2022-04-17

目录

1 自动化模块用户界面介绍

自动化模块基于同星自主研发的图形编程语言构建，可以在不需要编写代码的情况下，仅用图形化编程就能实现测量、仿真、任意逻辑执行、测试、标定和诊断等等功能。

1.1 自动化模块界面



顶上一行为工具栏，下方为图形化逻辑控制窗口工作区简称流程图，流程图内的逻辑操作由“入口点”和一系列的动作组成，每个动作可以是表达式、信号读写、函数调用、分支或空操作，右侧为每一个动作的属性、变量编辑区。

1.2 自动化模块工具栏

自动化模块工具栏如下图所示：

工具栏中各个图标定义如下：

用于返回上一级模块。

用于进入下一级子模块。

用于调整图形化逻辑控制窗口的大小。

添加一个向下的动作，程序可向下执行。

添加一个向右的动作，程序可向右执行。

添加向下、向右动作组（子模块），插入一个向下、向右跳转或来自跳转的动作。

删除一个选定的动作。

删除一个选定的分支。

导入、导出模块及模块属性设置。

编译、运行模块、编辑后台 C 代码、清除执行状态和清除所有端点操作。

运行模块。

暂停程序执行。

单步运行程序（进入子模块）。

单步运行程序（步过子模块）。

停止程序运行。

打开自动化模块所在的文件夹。

1.3 自动化模块属性窗口

1.3.1 模块页面

显示自动化模块编号及子模块名称，自动化模块编号可以根据需要更改。在模块页面空白处点击鼠标右键可以添加或删除模块及运行模块。

1.3.2 属性页面

先在图形化逻辑控制窗口工作区即流程图选中一个动作，再点击“属性”按钮，则关于该动作的各种属性都可以在这个页面进行设定或调整。

1.3.2.1 名称

名称与流程图内相应的动作一致，可以根据需要进行修改。

1.3.2.2 注释信息

在注释信息栏内添加一些说明的信息，以帮助对动作的理解。

1.3.2.3 动作执行设置

动作执行设置可以选择空操作（NOP）、信号读写、API 调用和表达式。

空操作（NOP）：可以设置向下执行和向右执行，值可以设置 OK 和 NOK。当程序运行至空操作动作时，若该值是 NOK 时这个动作显示红色。

信号读写：用来对已经设置好的本地变量、全局变量、CAN 信号和 LIN 信号进行赋值和读取，点击鼠标右键可以进行变量的添加和删除。

API 调用：可以选择小程序系统函数、小程序库函数和本地小程序内部函数。

表达式：可以按照 C 语言格式来书写表达式，表达式输入和表达式输出可以是已经设置好的变量。

1.3.3 变量页面

在变量窗口点击鼠标右键可以添加或删除变量，变量的类型可以是整形、双精度、字符串、CAN 消息、CANFD 消息和 LIN 消息，变量名和初始值可以根据需要修改设置，变量的当前值是当程序执行到断点后暂停，或是单步调试的过程中才会显示出来。

1.3.4 运行信息页面

当程序执行到结束时，用鼠标左键选中一个动作，则运行信息窗口显示该动作的有关运行时间或函数调用、表达式计算结果等信息。

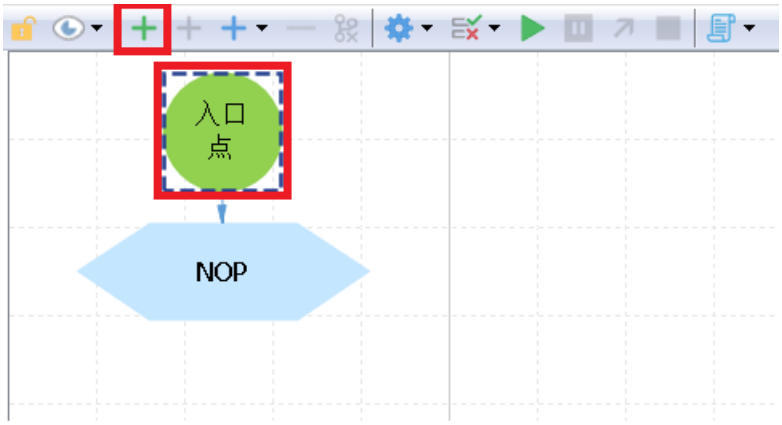
2 自动化模块基本操作及语法介绍

2.1 新建自动化模块

单击“仿真”->“自动化模块”->“添加自动化模块”就完成了一个新建的自动化模块。

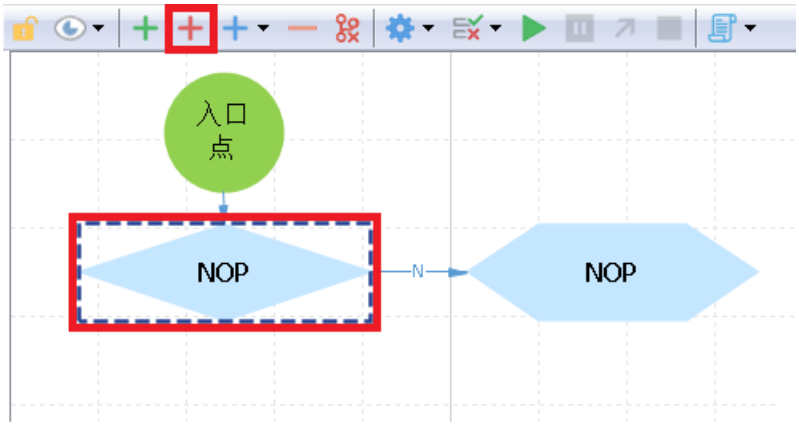
2.1.1 添加向下动作：

先单击“入口点”，再单击“添加向下动作”按钮就在下方添加好了一个新的动作，动作的名字默认为 NOP，可以在“属性”页面修改名字。



2.1.2 添加向右动作：

先单击“NOP”动作，再单击“添加向右动作”按钮就该动作的右方添加好了一个新的动作，动作名默认为 NOP。



2.2 变量

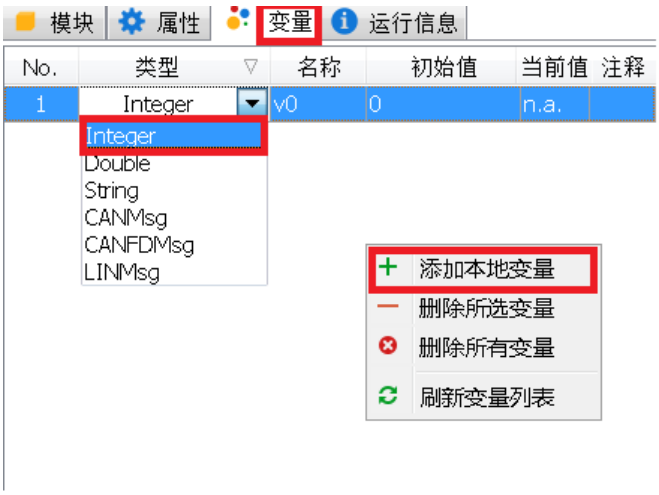
变量的类型有以下几种：

- Integer（整形）：数据范围从-2147483648 到 2147483647，占 4 字节。
- Double（双精度）：数据范围从-1.79×10^328 到 1.79×10^308，占 8 字节。
- String（字符串）：Unicode 字符集，可变字长。
- CANMsg（标准 CAN 总线数据）：数据长度 1 到 8 字节。
- CANFDMsg（FDCAN 总线数据）：数据长度 1 到 64 字节。
- LINMsg（LIN 总线数据）：数据长度 1 到 8 字节。

2.2.1 创建变量

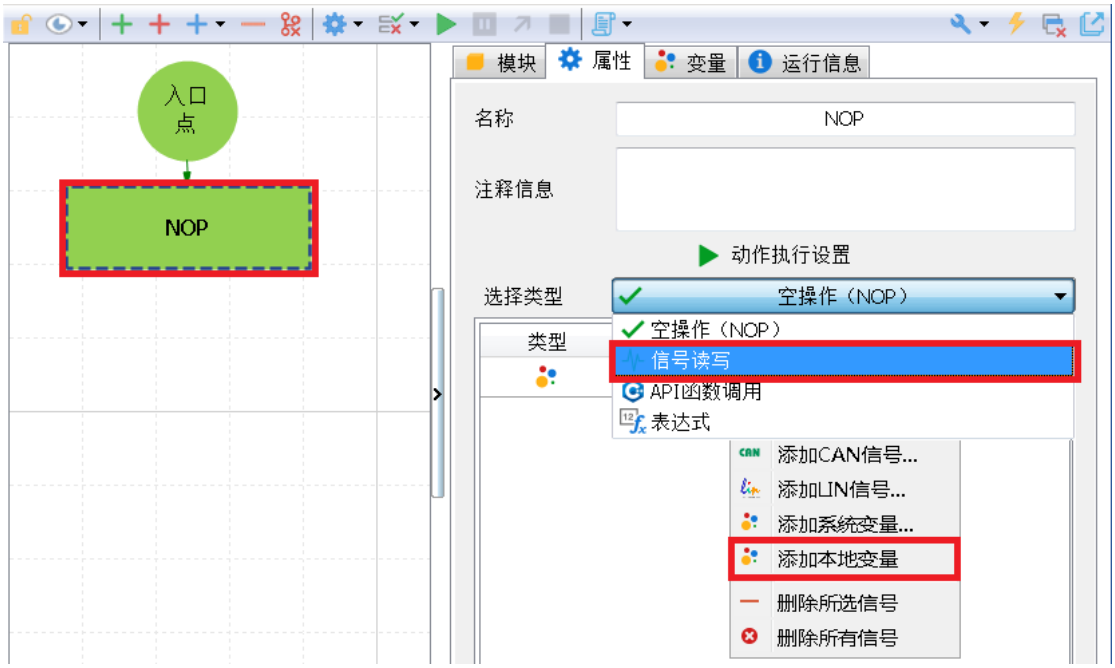
下面介绍如何添加一个本地变量：

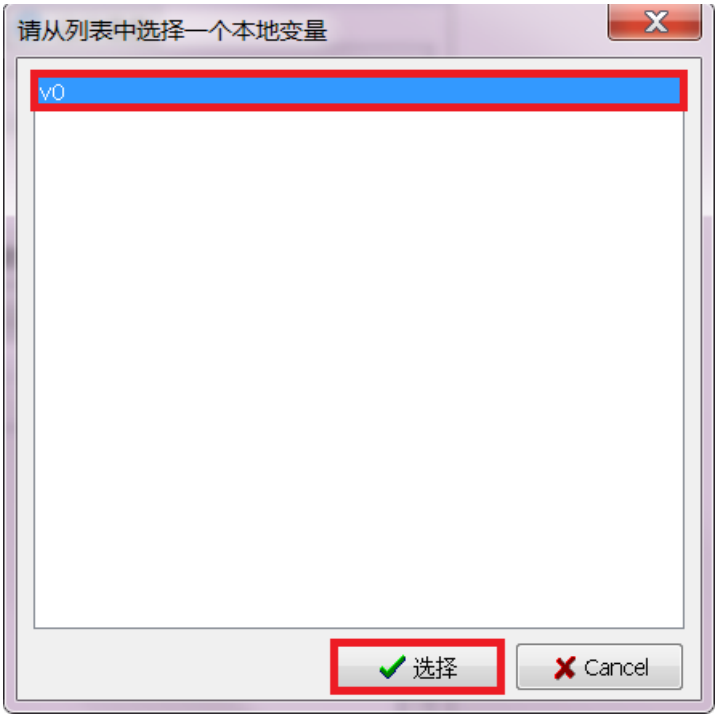
先点击属性页面“变量”，在页面空白处点击鼠标右键，在弹出的菜单上选择“添加本地变量”就添加好了一个本地变量，本地变量可被本模块的任意逻辑读写，与此同时本地变量也通过自动映射的方法注册到了系统变量表中，并带上了模块名作为前缀。变量名按照先后创建的顺序默认为 v0，v1 等等，可根据需要更改，变量的类型可点击“类型”标签的下拉箭头在列出的各种变量类型中选择所需要的，默认类型为整型，变量的初始值默认为 0，可在“初始值”标签下更改。



2.2.2 变量赋值

变量赋值可以在一个“动作”内完成。先按照上面介绍的方法新建一个本地变量，再选中一个动作（比如 NOP），在属性页面窗口“选择类型”点击下拉列表箭头选择“信号读写”，并在下方空白处单击鼠标右键，在弹出的菜单中选择“添加本地变量”，在弹出的窗口中选中一个变量（比如 v0），变量名就列出来了。





点击选中的变量名（v0）右侧空白处，填入要赋给变量的值（比如 5）即可。变量还可以由其它的如本地变量、CAN 信号（必须先建立好 CAN 数据库中的 DBC 文件）、LIN 信号和系统变量赋值。

- 选择本地变量。
- 选择 CAN 总线信号变量。
- 选择 LIN 总线信号变量。
- 选择系统变量。

2.3 动作的操作

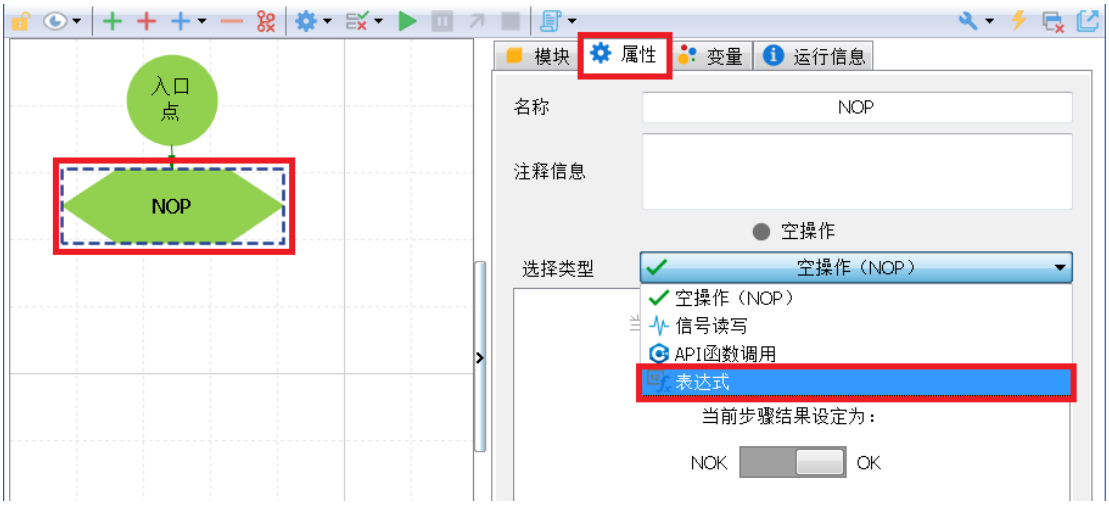
2.3.1 空操作 NOP

空操作执行的结果只有两个值，OK（绿色表示）和 NOK（红色表示）。

下面设置一个空操作的执行结果值为 NOK，按顺序单击“NOP”动作->“属性”->“NOK”->“运行”后，可以看到 NOP 变为红色。红色被认为是有故障，用户可根据动作运行后体现的颜色来查找问题。

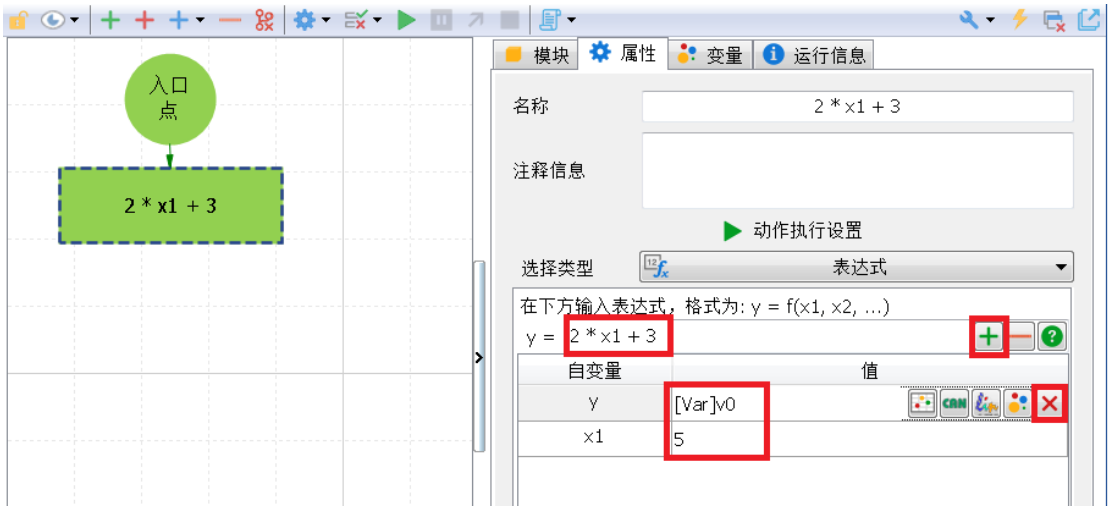
2.3.2 表达式

表达式按照 C 语言书写格式，完成加减乘除、三角函数、对数及指数等数学运算的功能。一个动作可以完成一个表达式的操作，表达式的输入（即自变量）可以是立即数、本地变量、CAN 和 LIN 信号变量及系统和用户变量，表达式的计算结果可以是本地变量、CAN 和 LIN 信号变量及系统和用户变量。按下述操作可以完成一个表达式的编写：按顺序单击一个动作“NOP”->“属性”->“表达式”如图：



在建立好的表达式属性界面点击图标“+”添加一个自变量 x1，自变量栏下的 x1 可以选立即数或其它变量，这里填入立即数（比如 5），在“Y=”一栏输入计算

公式，这里输入“2*x1+3”，最后在“自变量”一栏“Y”的旁边选择一个变量即将计算好的 Y 值赋值给这个变量，这里选择一个已经建立好的本地变量“如 v0”，这样表达式就建立好了。

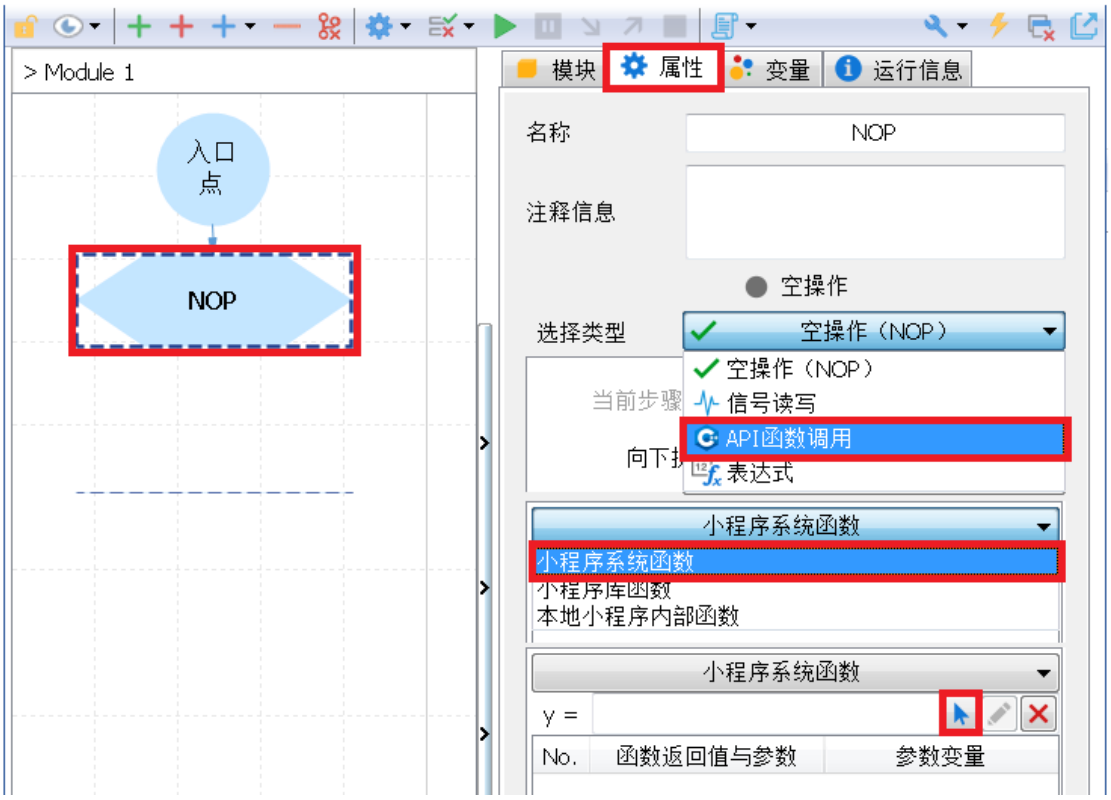


现在点击“运行”，结束后选择表达式动作“2*x1+3”，再点击“运行信息”就可以看到表达式计算的结果值。

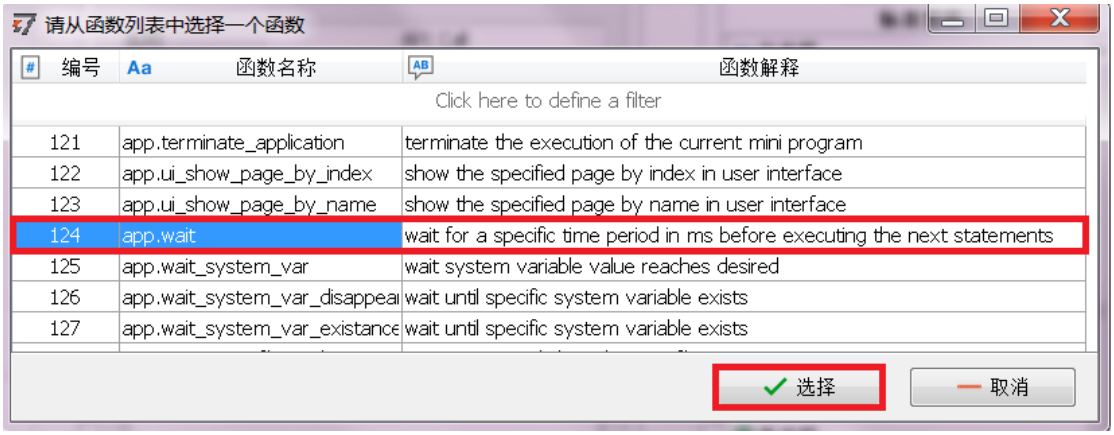
自变量“x1”的删除可以点击“-”号，变量“v0”的删除可以点击“X”按钮完成。

2.3.3 API 函数调用

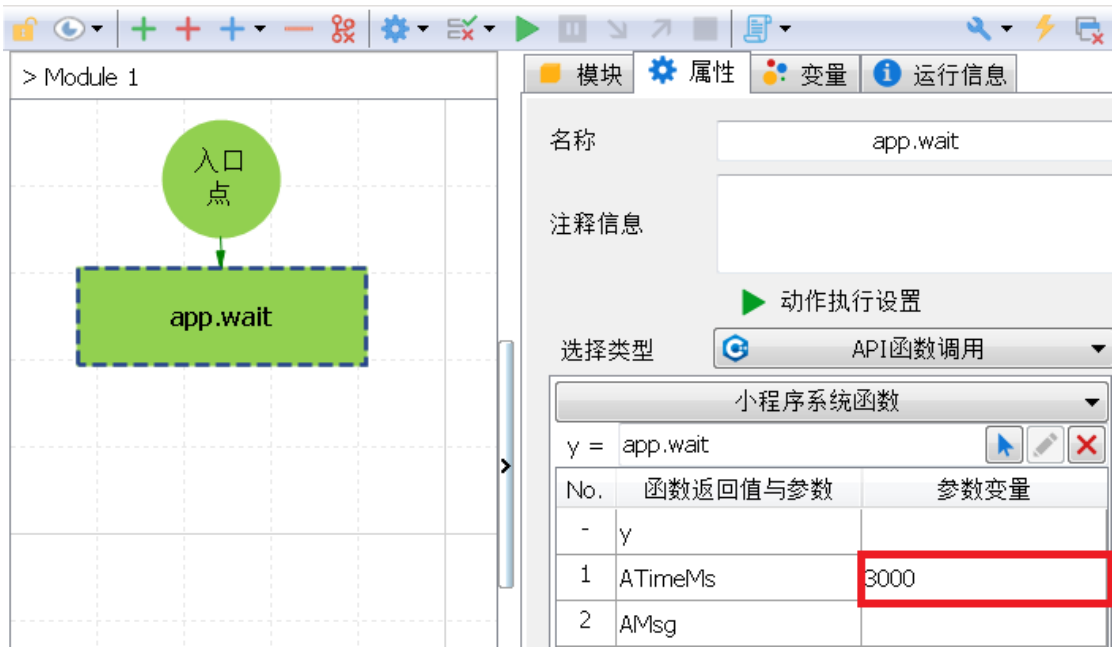
API 函数调用包括小程序系统函数、小程序库函数和本地小程序内部函数这三种函数。这里举一个调用例子小程序系统函数“app.wait”（延时功能）的例子，调用方法如下：点击动作“NOP”-> “属性” -> “API 函数调用”。



再点击“小程序系统函数”及添加函数的“箭头”，并在跳出的函数列表菜单中选中“app.wait”函数，再点击“选择”。



然后在“app.wait”小程序的“函数返回值与参数”一栏将“ATimeMs”的值填入要延时的时间，单位是毫秒，这里填入 3000 即 3 秒。完成后点击运行按钮可以看到程序运行后在“app.wait”动作处停留 3 秒的时间。

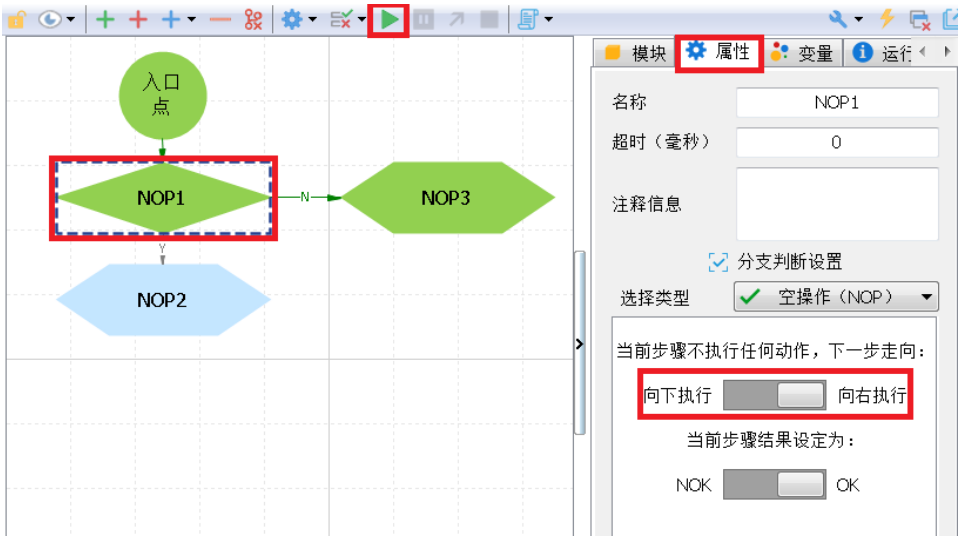


2.4 分支

当一个动作的下方和右方都存在另一个动作时，该动作就是一个具有分支功能的动作。

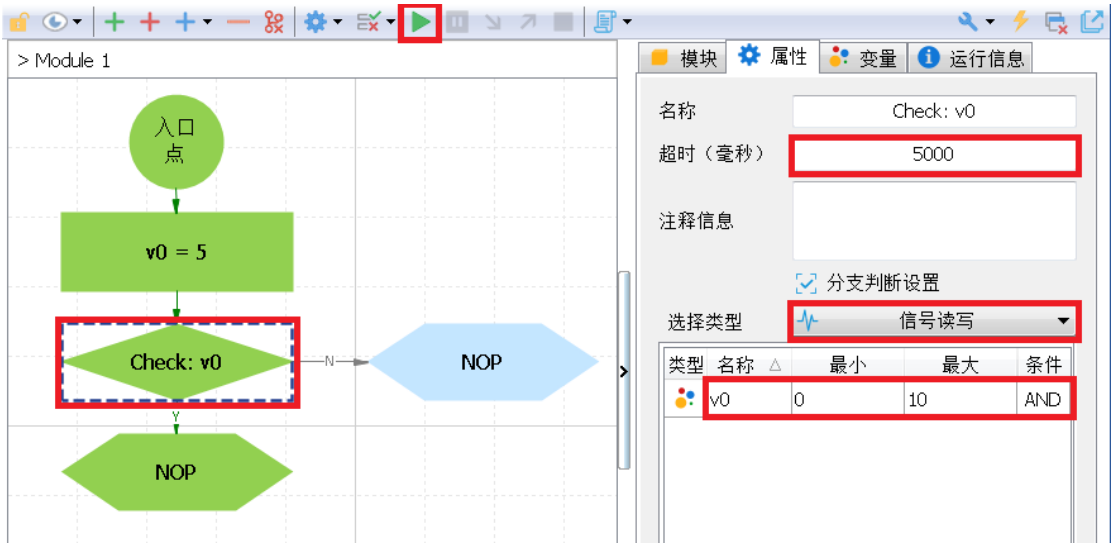
2.4.1 NOP 分支

NOP 分支可以构成一个具有固定分支功能，可以指定该动作的执行方向是向下或向右。下面设置一个向右的分支动作，按顺序单击“NOP1”->“属性”->“向右执行”->“运行”后，就看到程序运行经过的动作都变为绿色（动作执行结果都设定为 OK 时），这样可以清晰地看出程序运行所经过的路线。NOP 分支动作通常用来调试程序，控制程序的实际走向。



2.4.2 信号读写分支

信号读写可以作为一个可变的分支动作，满足信号变量条件时程序向下执行，不满足条件时程序向右执行。具体设置方法如下：选中一个具有分支的动作（如 NOP），选中“信号读写”，按照前面介绍的添加变量（比如 v0），变量的值可以在创建该变量时指定，也可以在一个信号读写的动作中赋值（比如 v0 等于 5），然后在该变量的“最小”和“最大”输入框内填入具体的数值（比如最小等于 0，最大等于 10），可以根据逻辑的要求在右侧的“条件”的下拉列表框内选择 AND 或 OR 操作（比如选择 AND），然后点击运行按钮可以看到程序是向下运行的，因为 v0 的值是 5 且大于 0 小于 10，条件满足向下执行。



“属性”页面有一栏是“超时（毫秒）”是在条件不满足时程序要在延时一定的时间后才向右执行，延时时间可以填入 0 或其它的数，这里填入 5000 毫秒，

即当条件不满足时延时 5 秒才向右运行。

2.4.3 表达式分支

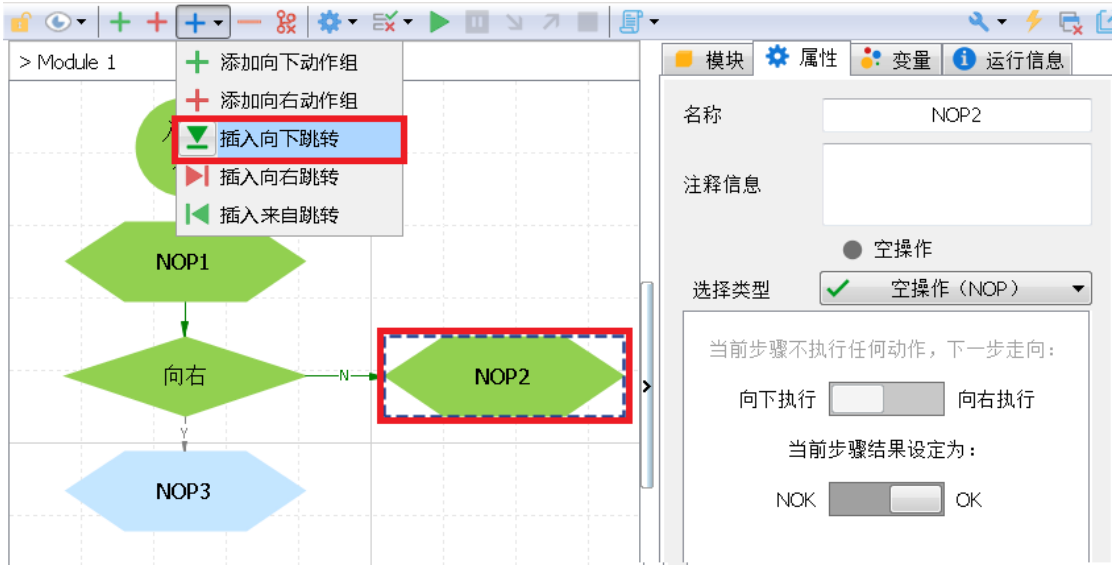
当一个分支动作是一个表达式时，程序将根据表达式计算的结果即 Y 值来决定程序的走向，当 Y 值等于 0 时程序向下执行，当 Y 值为非 0 时向右执行。

2.4.4 API 函数调用分支

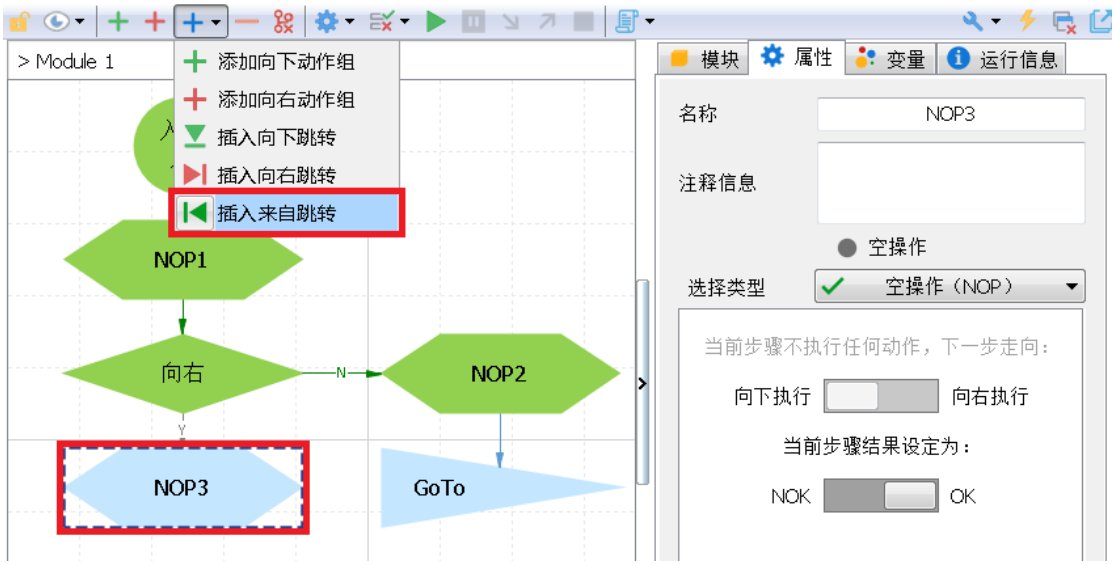
当一个分支动作是一个 API 函数调用时，程序将根据 API 函数返回的值来决定程序的走向，当返回值等于 0 时程序向下执行，当返回值为非 0 时向右执行。

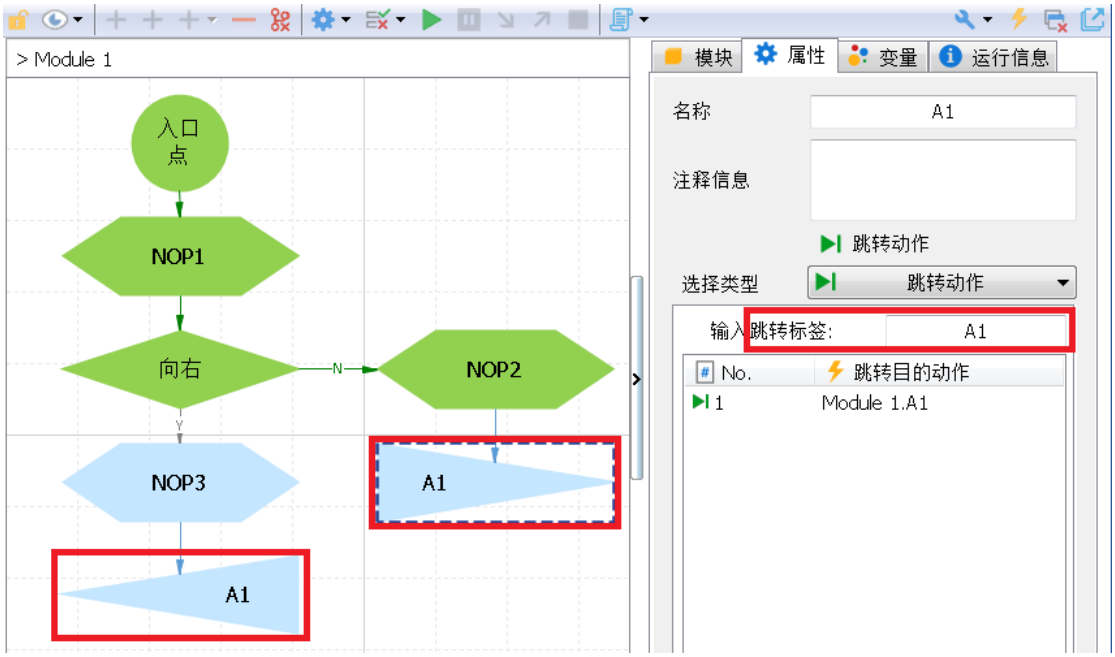
2.5 跳转

跳转动作可以改变程序运行的位置以实现一定的逻辑需求或实现循环等功能。如下图若要在“NOP2”的下方插入一个跳转动作，跳转至“NOP3”的下方，可以点击“NOP2”，再点击“插入向下跳转”，就增加了一个跳转的动作。



然后点击“NOP3”，再点击“插入来自跳转”，就创建了一个接受跳转的动作，就与程序语言设置一个跳转的地址标号相似。最后把“Go To”和“From”的标签改成一致的符号即可，这里填入“A1”，这样就完成了一个跳转的操作。“插入来自跳转”动作也可以放在程序的上方以实现循环的操作。





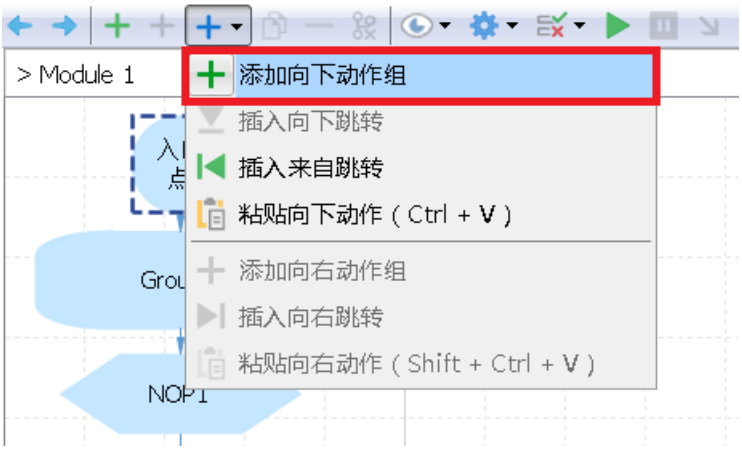
2.6 子模块

当一个由若干个动作组成具有一定功能的模块作为上一级模块的一个动作，该模块就是子模块，子模块可以无限嵌套。

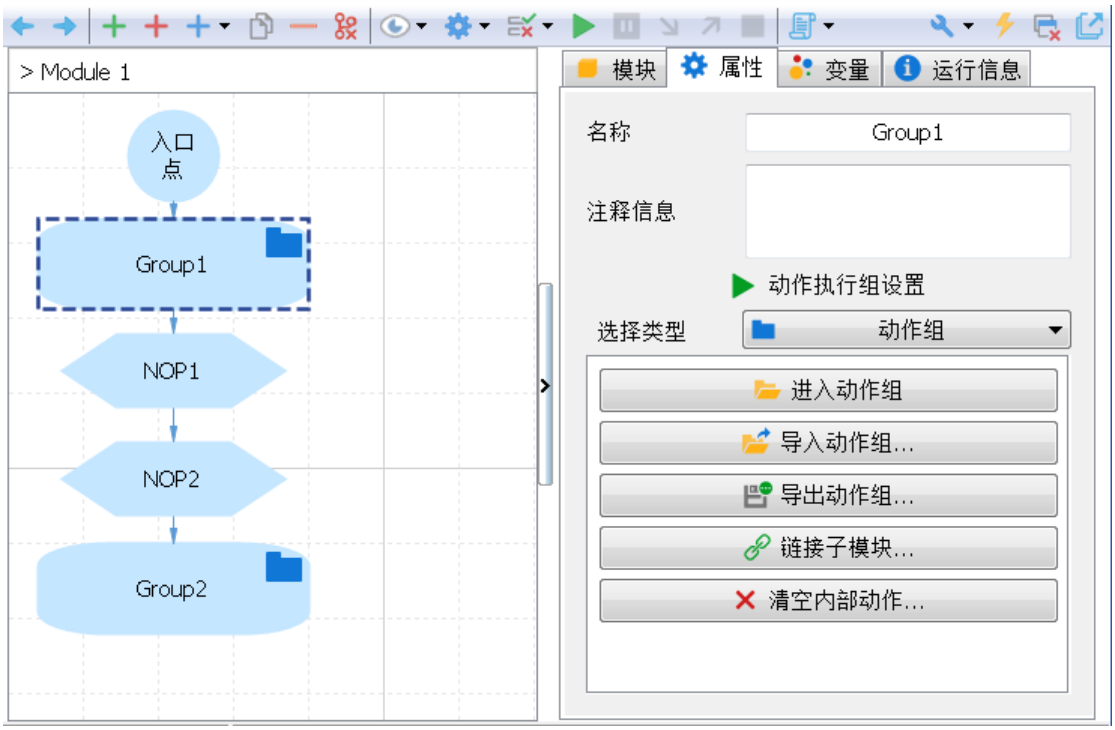
2.6.1 添加子模块

选中一个动作，点击“添加向下（或向右）动作组”即可在其下方或右方添加

一个子模块。

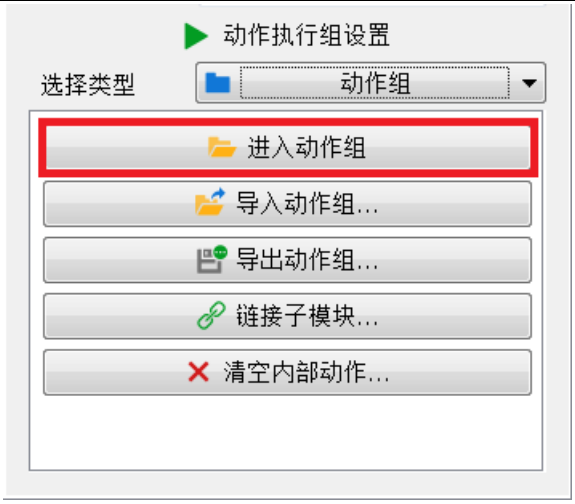


在本例子中添加了 2 个子模块并命名为“Group1”和“Group2”，也称动作组，如下图：



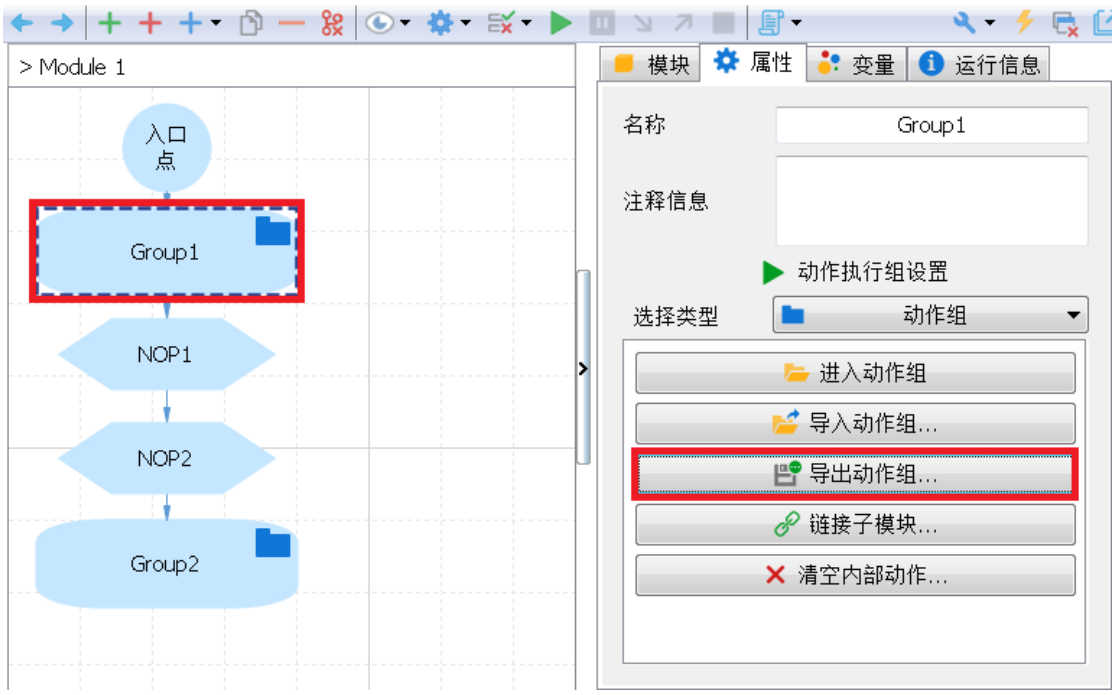
2.6.2 编辑子模块

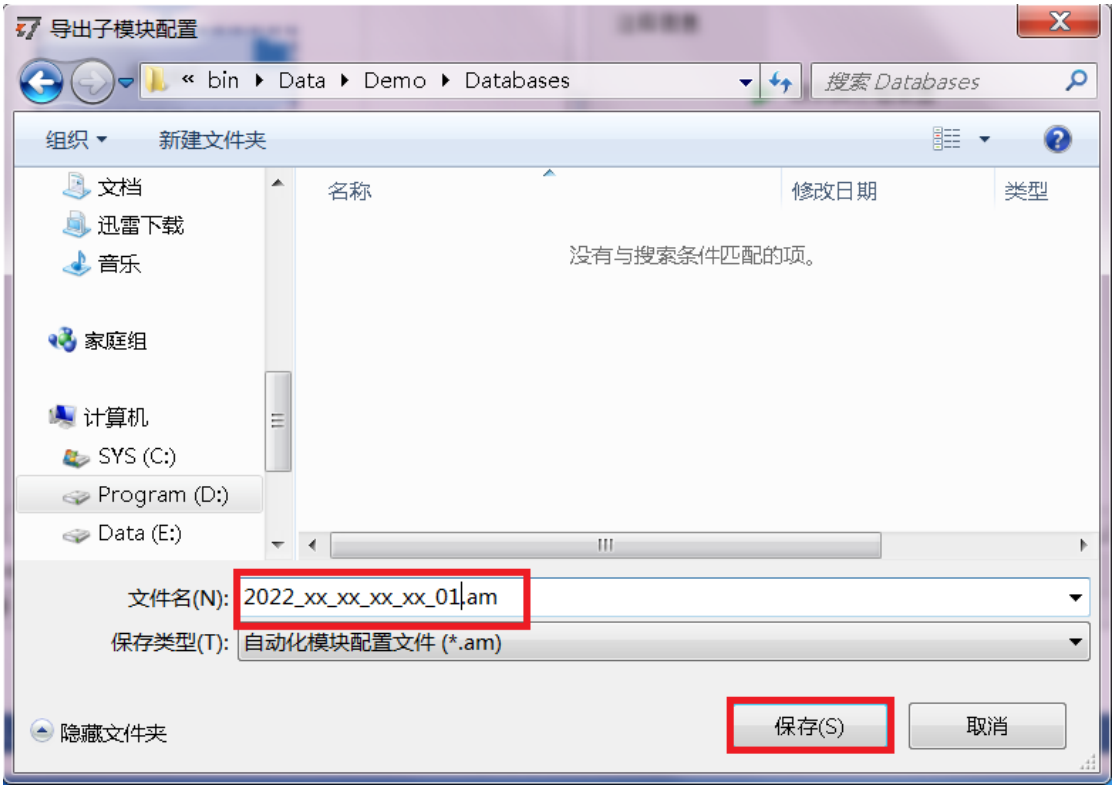
进入子模块可以用鼠标双击动作组如“Group1”，或先点击动作组再点击“进入动作组”按钮，就进入到子模块的窗体。子模块可以当做普通的模块进行设计，这里在子模块中添加 2 个“NOP”动作，其中第二个“NOP”动作设定为“NOK”，也就是程序执行到这个动作时是以红色显示其状态的。



2.6.3 导出动作组

子模块编辑好后可以点击左上角的箭头“返回上一级模块”，在上一级模块中保存子模块的方法是点击“导出动作组”按钮，并设置后缀是“am”的文件名保存即可。

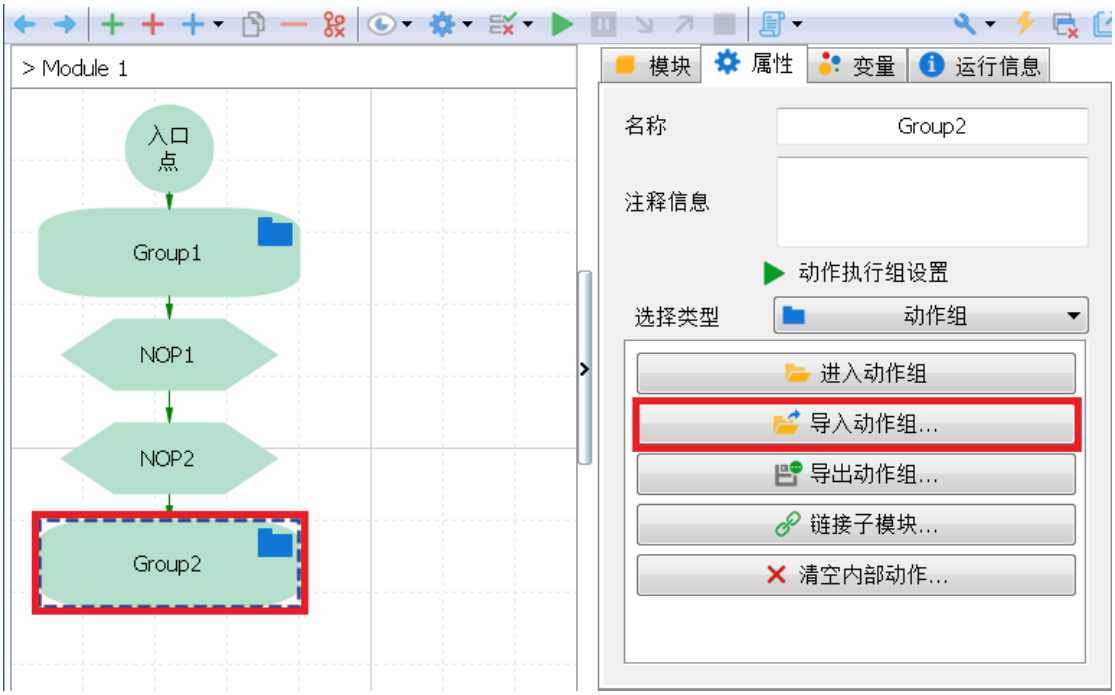


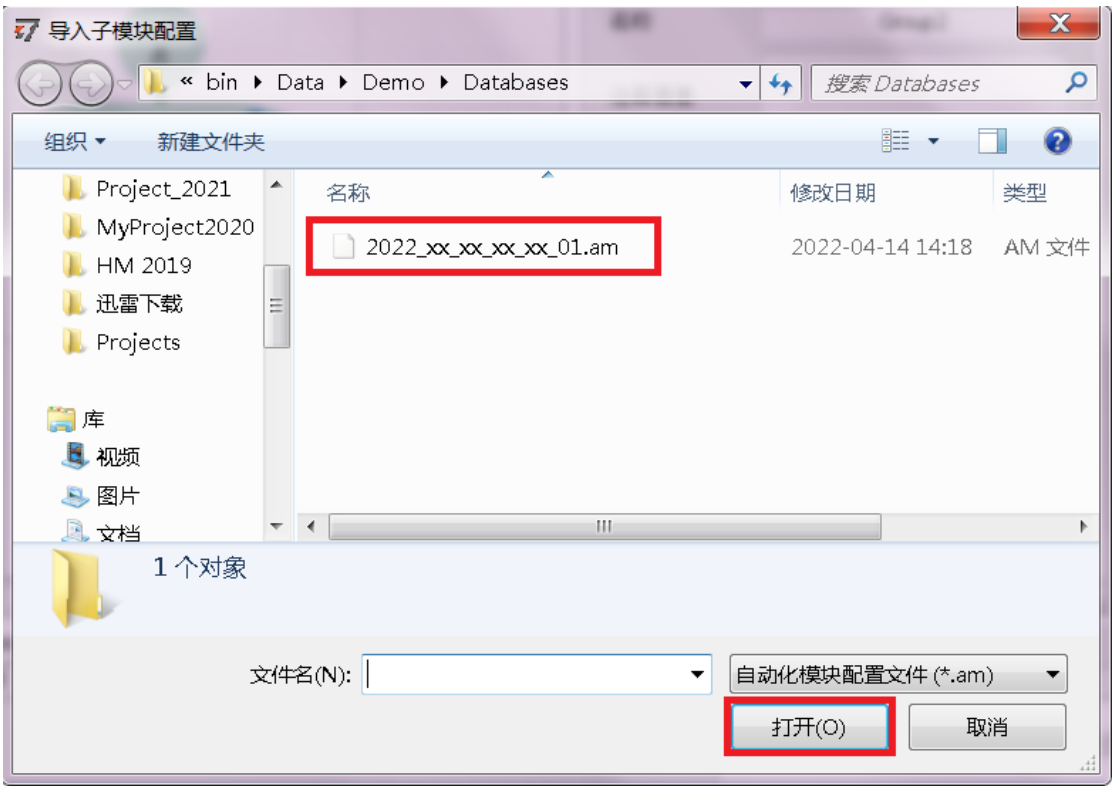


2.6.4 导入动作组

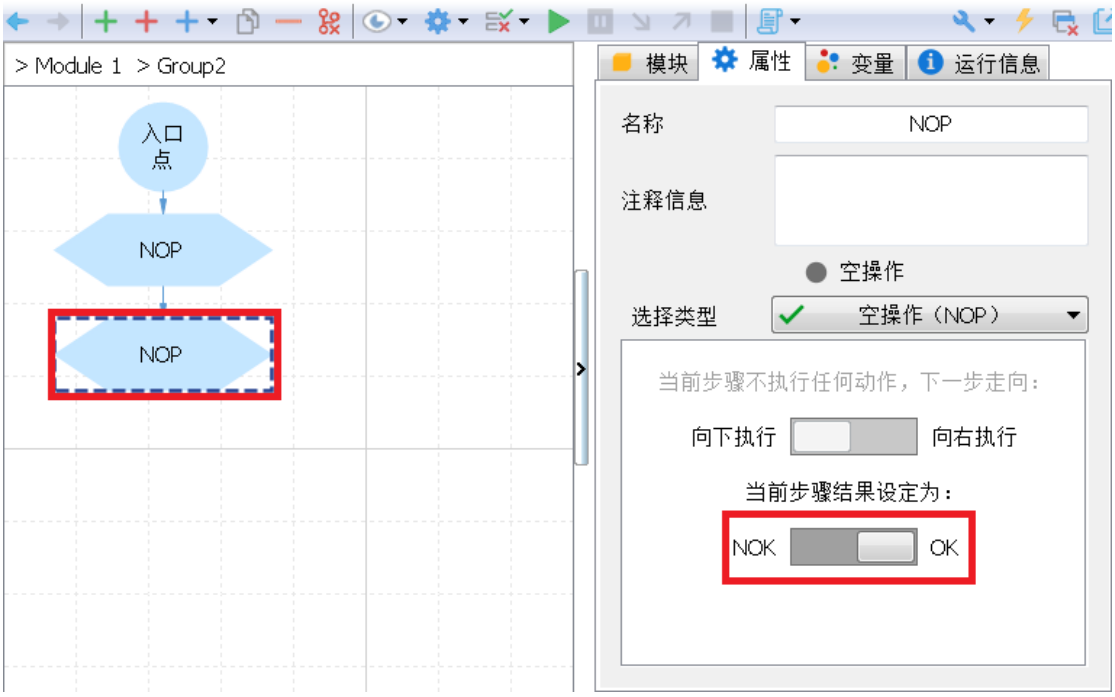
保存好的动作组 am 文件可以被赋值给其它的动作组，先点击一个动作组再点击“导入动作组”，选择一个 am 文件点击导入即可。这里选择“Group2”，将刚才导出的动作组文件导入，这样“Group2”和“Group1”就具有相同功能的子模

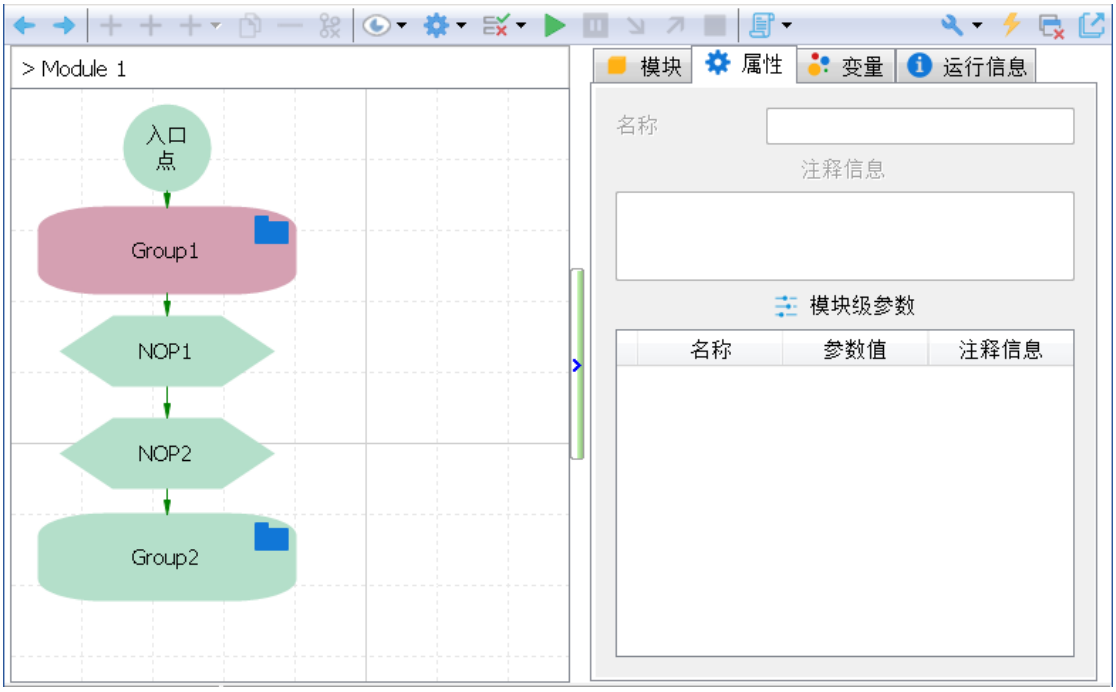
块了。





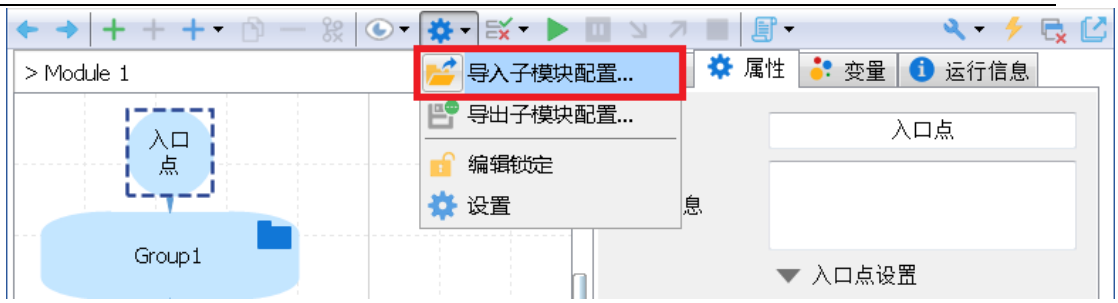
这时双击动作组“Group2”可以看到子模块的内容与“Group1”的相同，但是两个模块没有任何联系，即改变“Group2”的内容不会影响到“Group1”的内容。这里我们进入“group2”，修改第二个“NOP”的运行结果为“OK”看看运行结果，点击运行按钮可以看到“Group1”显示红色，“Group2”显示绿色。



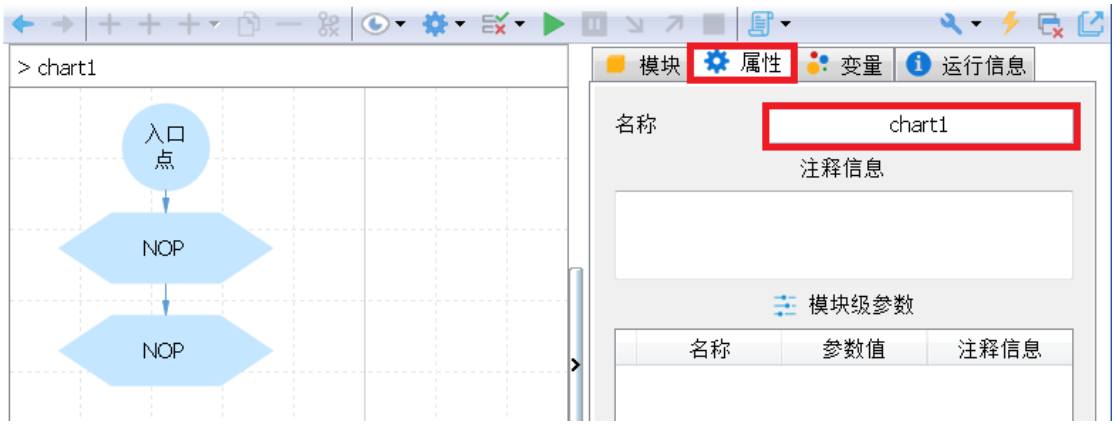


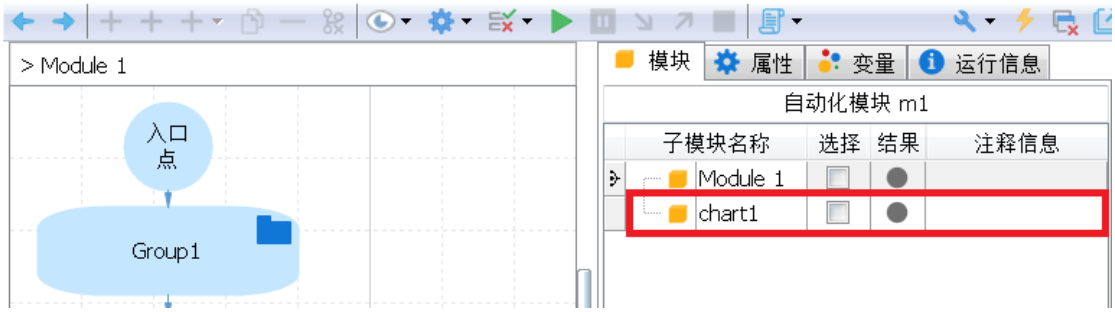
2.6.5 链接子模块

当我们需要在多个地方运行一个具有相同功能的子模块，且修改其中一个子模块的功能可以应用到所有调用这个子模块的地方，这时就需要用到“链接子模块”的功能。可以按照以下操作：在工具栏上点击“导入子模块配置”按钮，在打开的对话框里选择一个 am 文件点击打开即可。

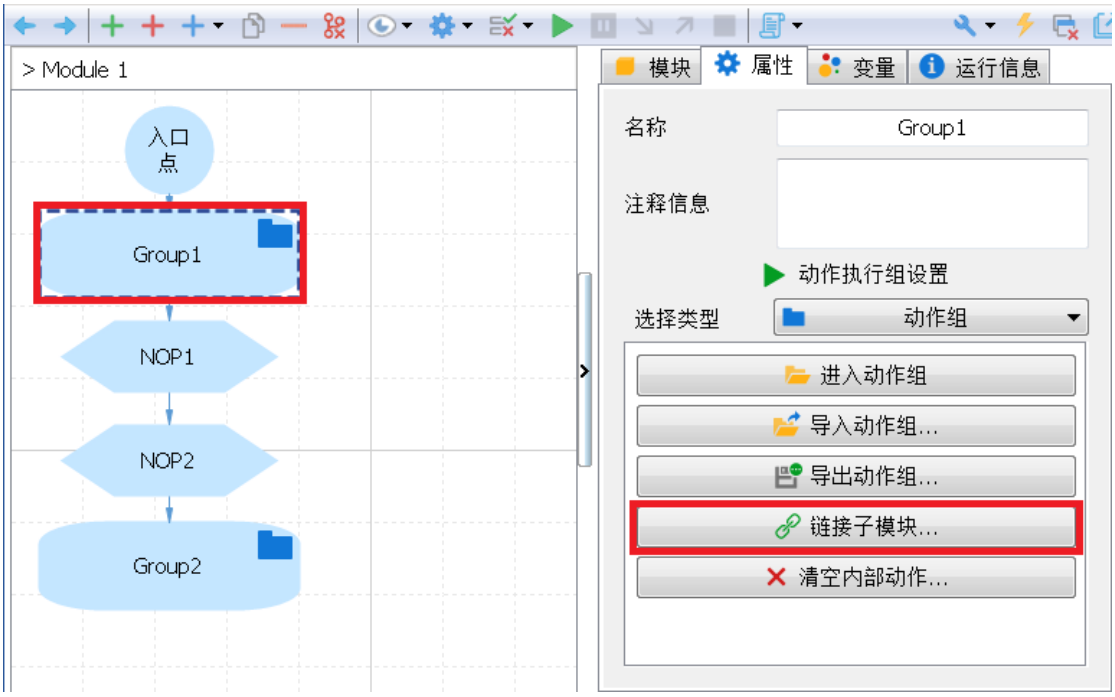


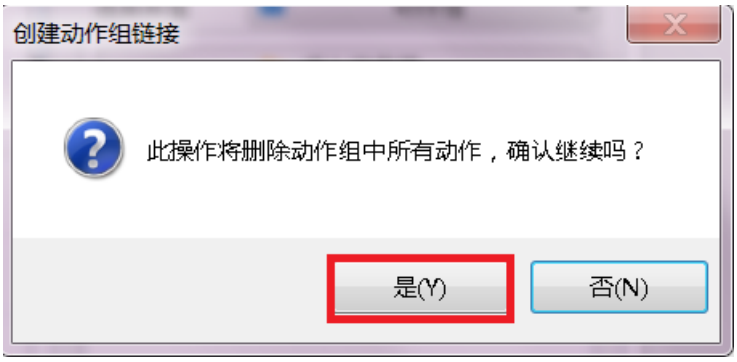
导入后将子模块的名称更改为“chart1”，这时回到上一级模块，点击属性页的“模块”按钮就可以看到“Chart1”模块也在列表中了。

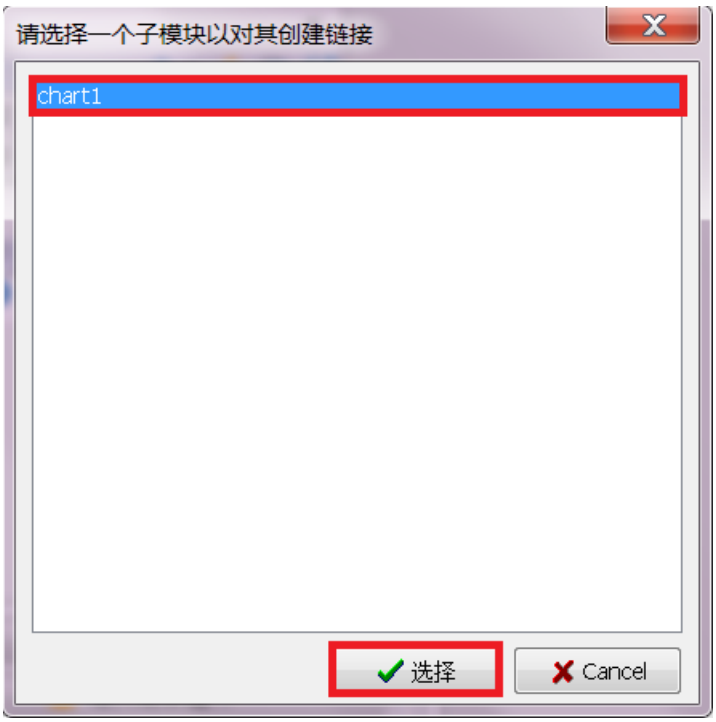




按如下操作将子模块“Chart1”链接进动作组“Group1”和“Group2”中。先点击一个动作组，再点击“链接子模块”按钮，在弹出的对话框中选择“是”，并选择模块“Chart1”再点击“选择”即可。本例子中将“Chart1”都链接进动作组“Group1”和“Group2”中。

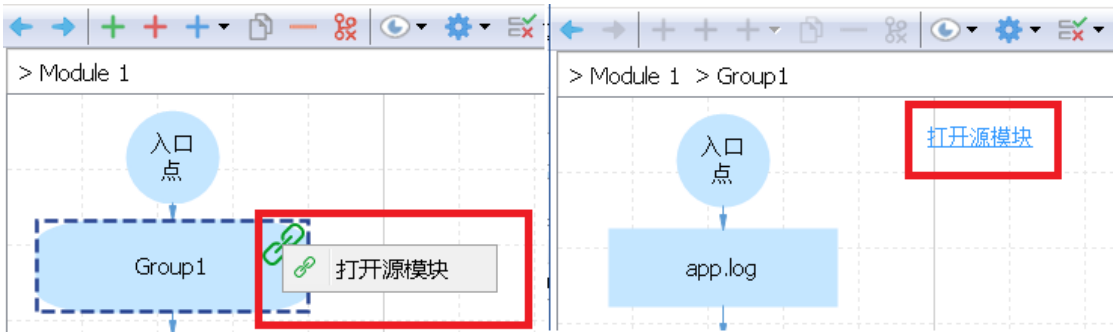

> Module 1
入口点
Group1
NOP1
NOP2
Group2
模块
属性
变量
运行信息
名称
Group1
注释信息
动作执行组设置
选择类型
动作组
进入动作组
导入动作组...
导出动作组...
链接子模块...
清空内部动作...


创建动作组链接
此操作将删除动作组中所有动作，确认继续吗？
是(Y)
否(N)

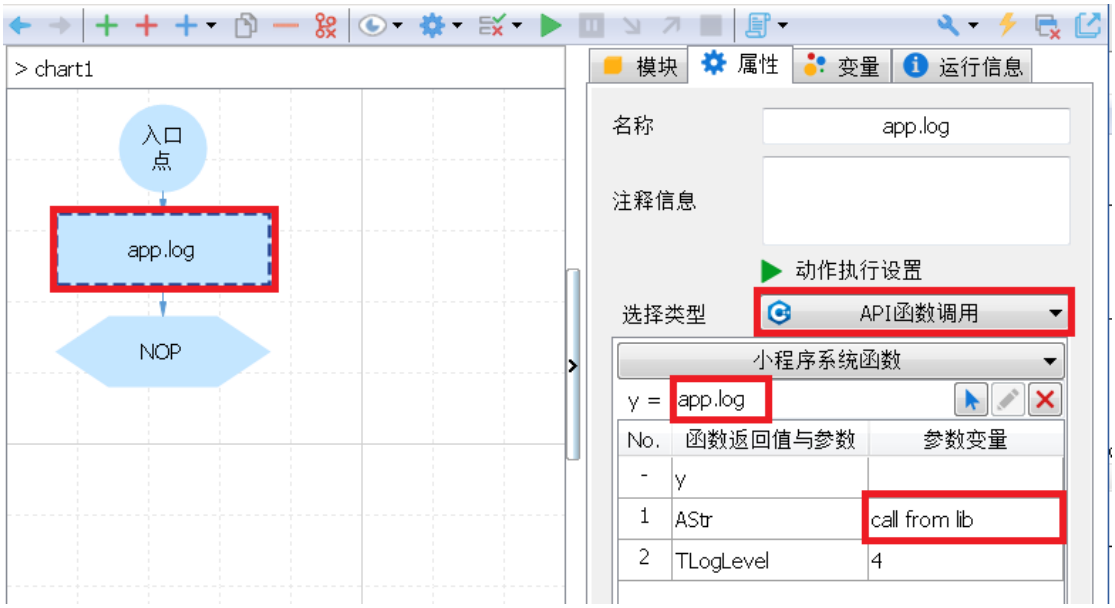

请选择一个子模块以对其创建链接
chart1
选择
Cancel

链接完成后动作组的图标右上方有一个锁链模样的图标，点击锁链图标即可进入子模块的编辑窗口，这时修改子模块的功能将影响所有调用该子模块的动作组功能。

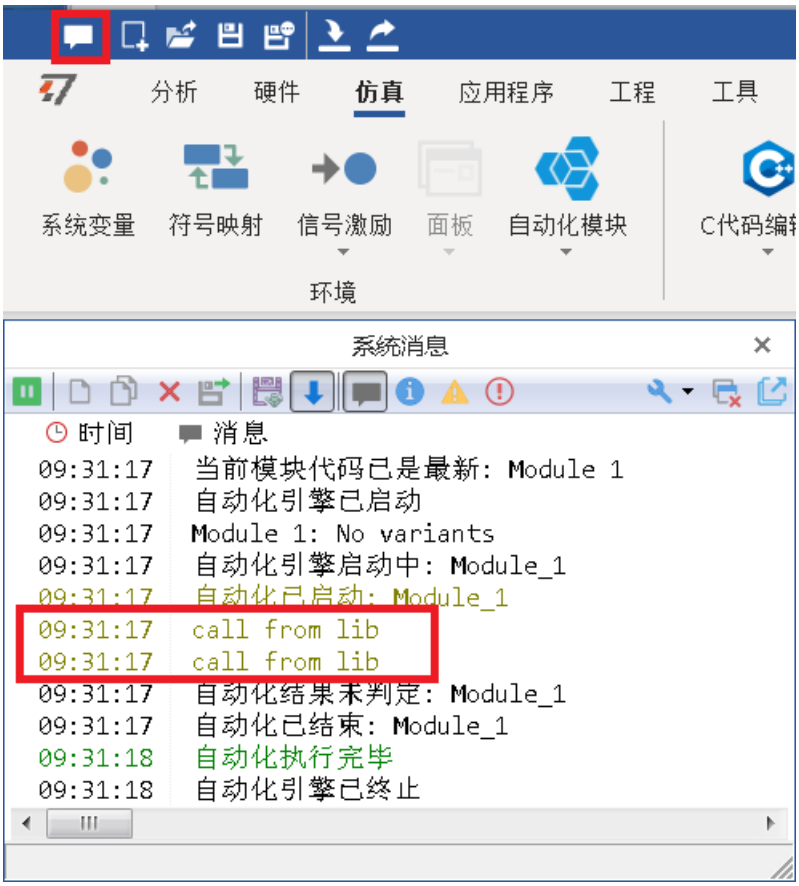
进入编辑子模块窗口的另一个方法是双击动作组在弹出的子模块窗口中点击“打开源模块”即可。



在这里我们将将修改一个子模块的功能看是否关联到另一个动作组，先点击动作组“Group1”的锁链图标进入子模块编辑窗口，将第一个“NOP”动作改为输出信息的函数“app.log”，在 AStr 一栏输入“call from lib”如下图：



完成后回到主模块，点击运行按钮待程序运行结束后再点击软件主窗体左上角的信息窗口图标，在打开的信息窗口可以看到信息“call from lib”输出了两次，说明两个动作组都执行了一样的操作。



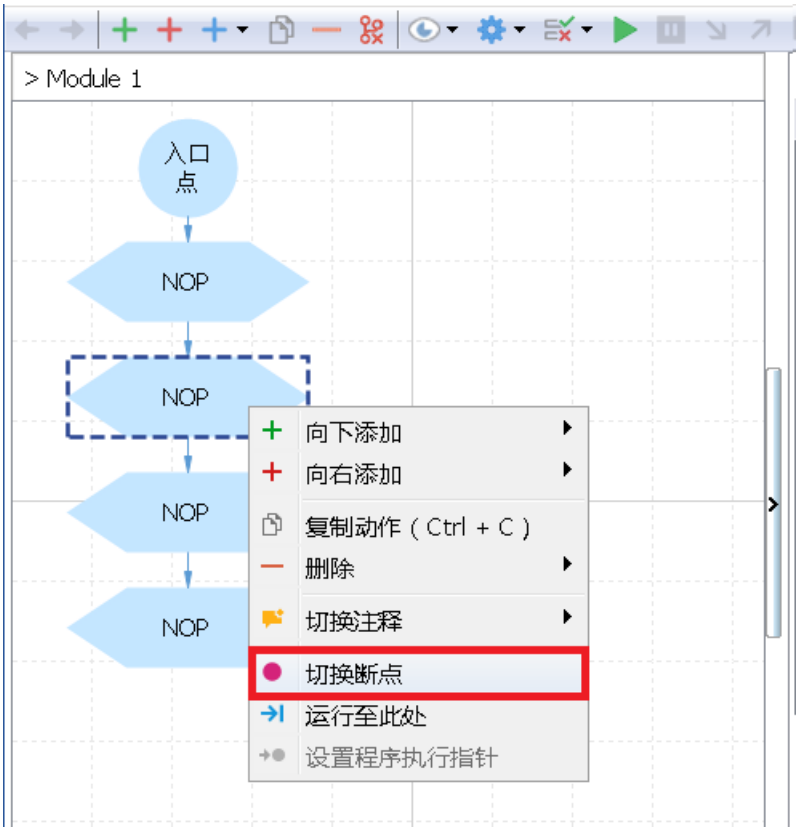
2.7 调试

充分利用程序的调试功能可以有效地控制程序的运行和观察状态的变化。

2.7.1 断点

当某一个动作设置断点后，程序运行到该动作就会暂停下来。设置方法是点击要设置断点的动作，再点击鼠标右键，在弹出的菜单中选择“切换断点”即可，动作的外框变成红色，当点击运行按钮后程序运行后就会停在该动作。当一个动作设置好断点后，若要取消断点，仍然按照如上操作即可去除断点，也可以点击工具栏上的“清除所有断点”按钮，此时将清除在所有动作已设置好的断点。

让程序在某个动作停下来的另一个方法就是点击该动作，再点击鼠标右键，在弹出的菜单中选择“运行至此处”，此时程序就开始运行并在该动作处停止。



2.7.2 暂停和单步运行

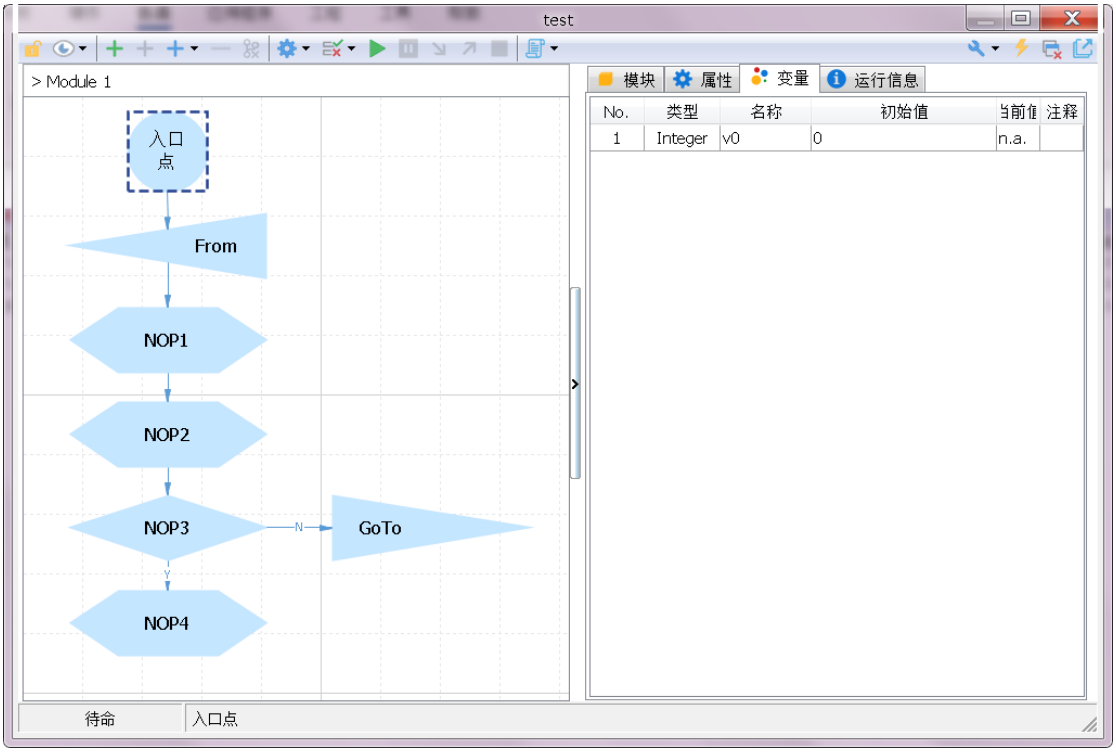
当程序运行在需要较长时间等待的动作时，若想暂停程序的运行可以点击“暂停”按钮，当程序遇到断点或按了暂停按钮而停下来后，可以按向下的箭头（进入子模块）或向上的箭头按钮（步过子模块即运行完一个子模块）让程序单步运行。

单步运行也可以通过按键来控制，按“F7”进入子模块按“F8”步过子模块就能控制程序的单步操作。

3 使用自动化模块实现自动测量

本章节介绍一个使用自动化模块实现自动测量的例子。例子内容为循环读取仿真的 CAN 信号并进行计算处理，当满足一定的条件时终止程序的运行。

按照前面介绍的方法创建一个自动化模块如下图：



3.1.1 创建整形变量

创建一个整型变量“v0”，初始值为 0。

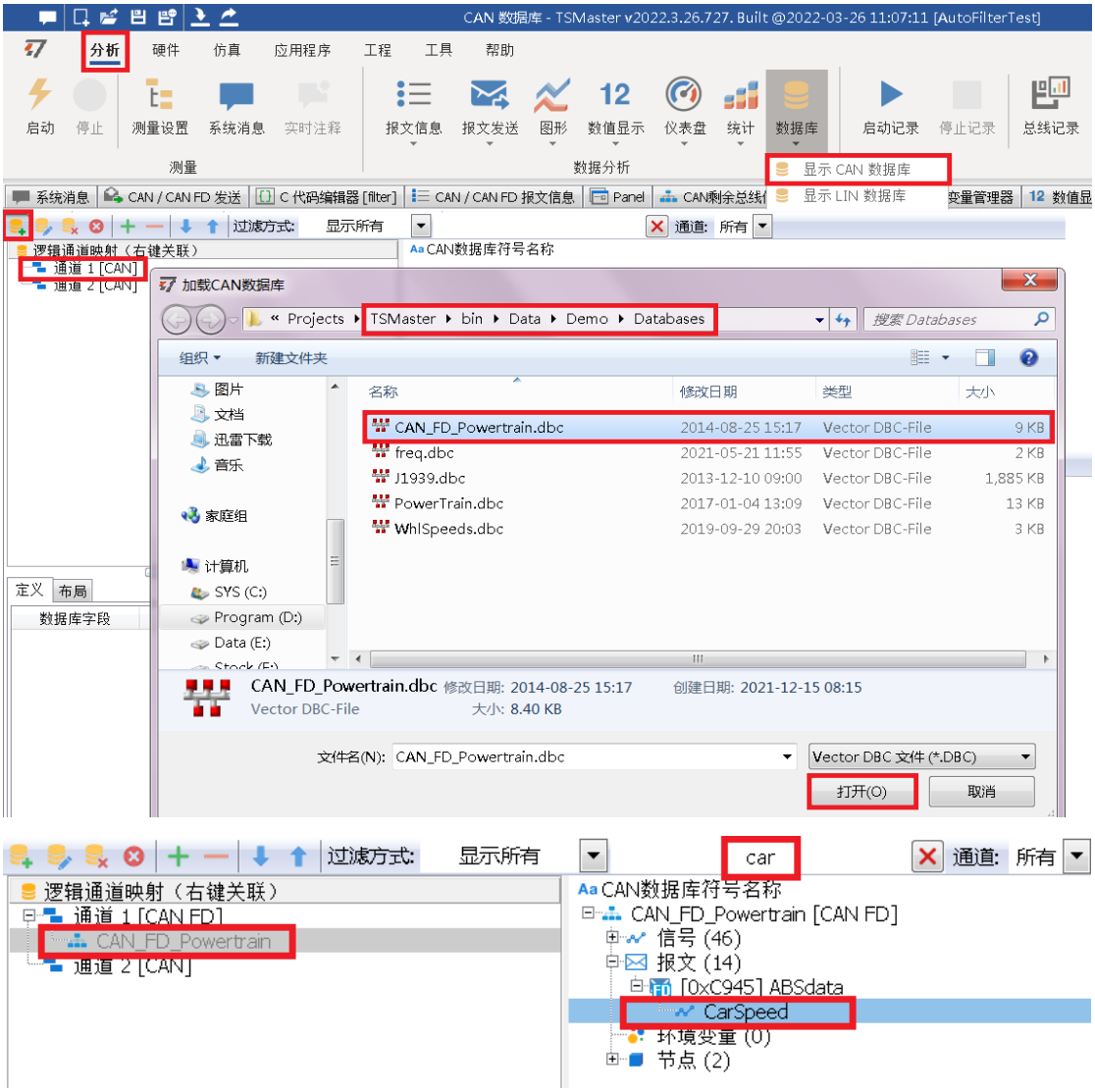
3.1.2 更改跳转标签

将“Form”和“GoTo”标签都改成“Loop“，实现循环操作。

3.1.3 添加 CAN Dbc 文件

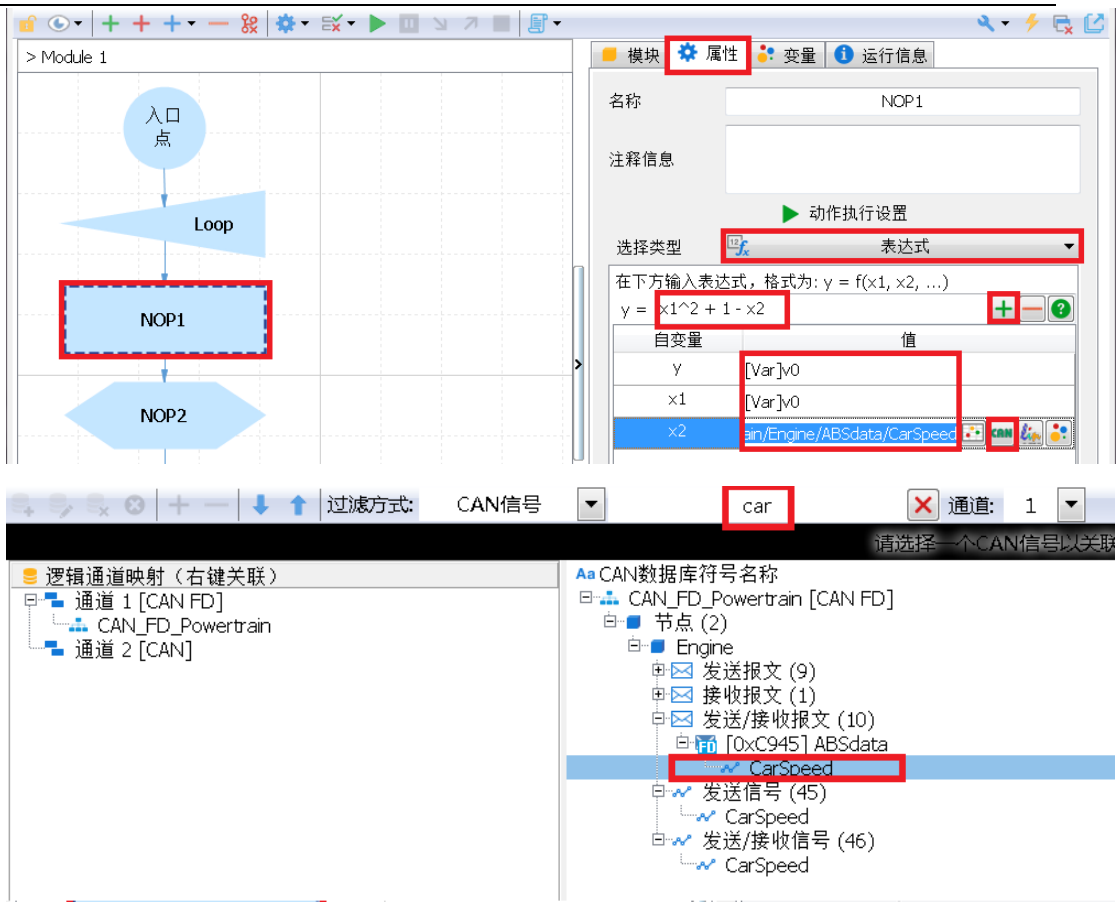
添加一个 TSMaster 自带的 DBC 文件，方法如下：点击“分析”->“显示 CAN 数据库”-> “通道 1[CAN]” -> “+”，在弹出的对话框中找到 TSMaster 自带的 DBC 文件“CAN_FD_Powertrain.dbc”点击 “打开”，就添加好了一个 DBC 文件。

在搜索框中填入“car”并回车，找到 CAN 信号“CanSpeed”，这就是本章例子所要用到的测试数据。



3.1.4 设置表达式

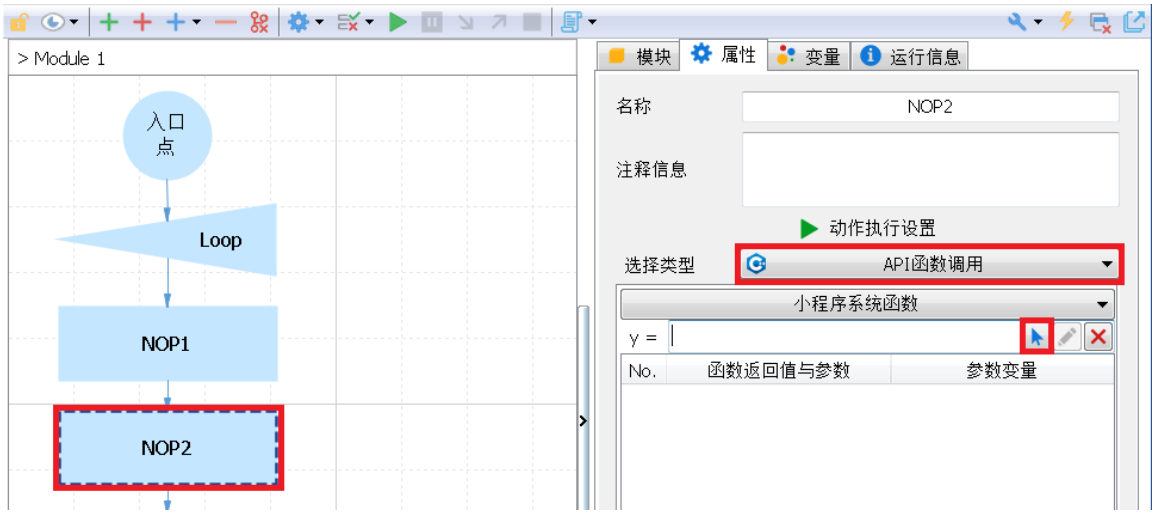
回到自动化模块，将“NOP1”改成表达式计算式。方法如下：点击“NOP1”->“属性”->“表达式”，再点击“+”号添加两个自变量“x1”和“x2”，自变量“Y”和“x1”均关联本地变量“v0”，“x2”关联上一节添加的 CAN Dbc 信号“CarSpeed”，并在“Y=”旁输入计算式，此处的例子是“x1^2 + 1 - x2”，x1 是本地变量 v0，x2 是接收到的 CAN 信号 CarSpeed，计算结果再送回本地变量 v0。这样通过表达式计算影响本地变量 v0 的值，计算公式可以根据需要自行书写。



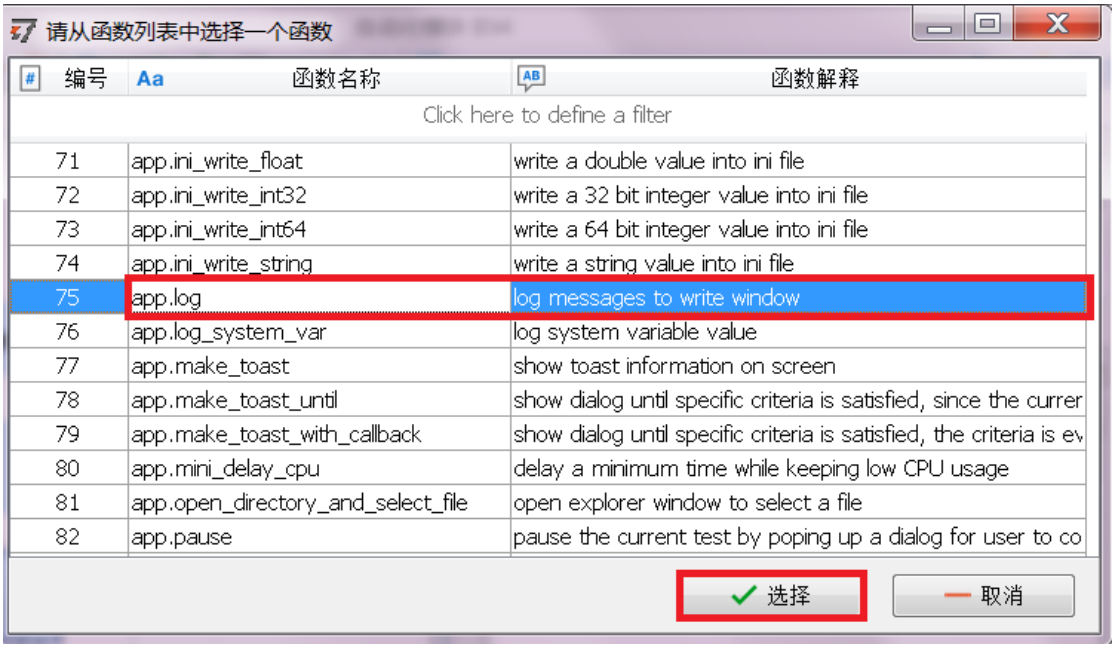
3.1.5 添加观察动作

添加一个可以监视程序运行时变量 v0 的变化实时值可以了解程序运行的每一步的状态，以便于调试。

下面就介绍添加一个具有监视功能的动作，点击“NOP2”->“API 函数调用”->“箭头”，打开函数选择窗口。



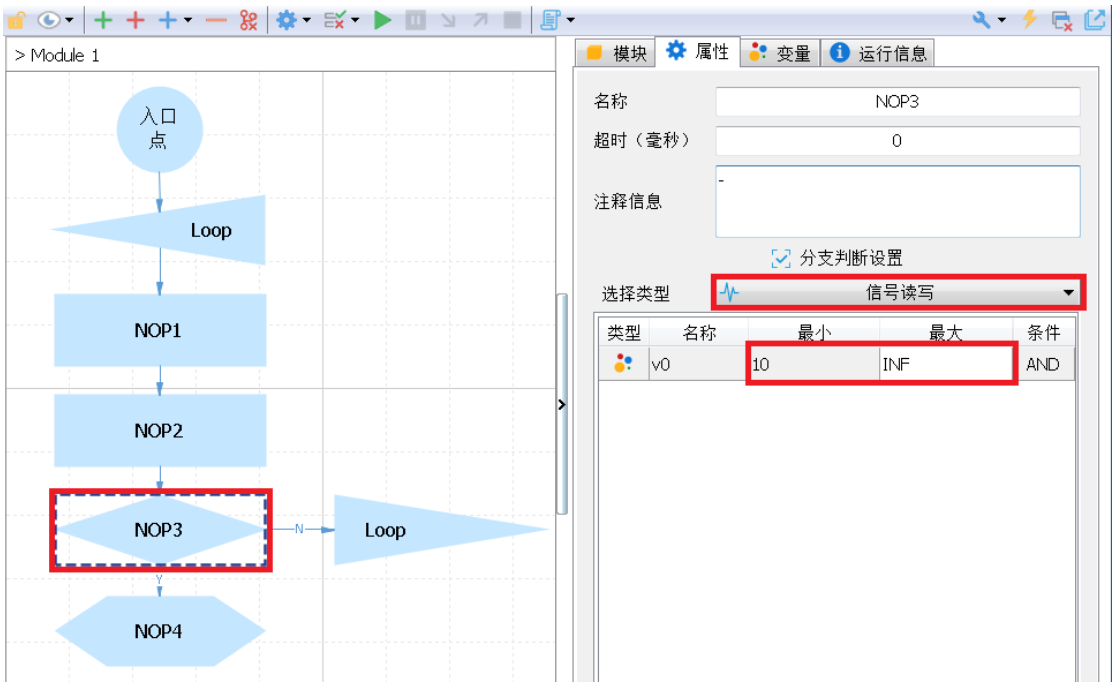
在弹出的窗口中选择“app.log”，并点击“选择”按钮，就完成了 app.log 函数的选择。



接着在函数返回值与参数“AStr”一栏选择本地变量 v0。

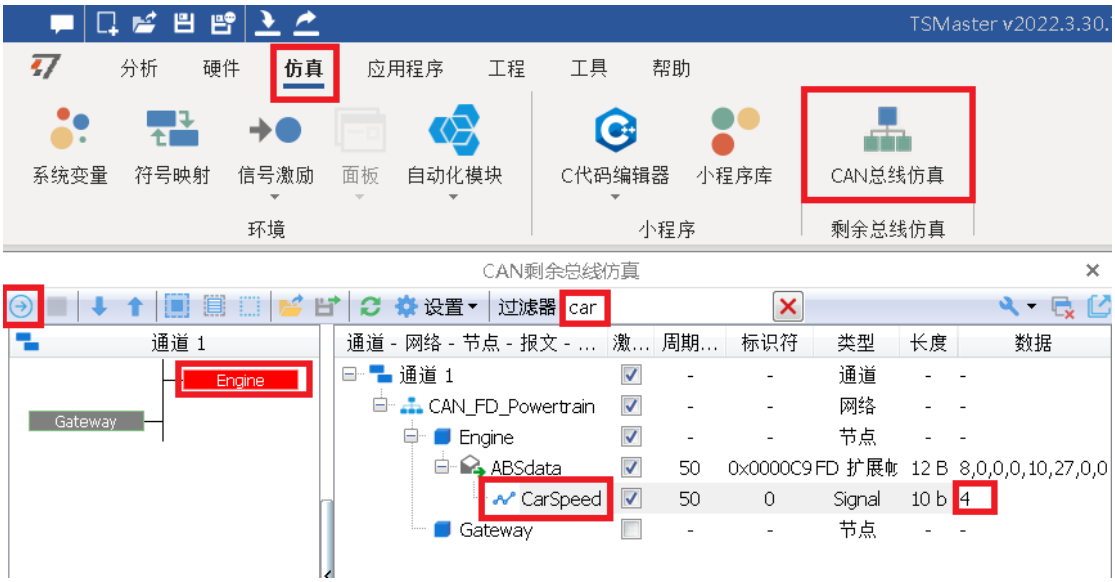
3.1.6 设置分支动作

分支动作的作用是当条件满足时才向下执行，条件不满足时就跳转以实现循环的功能。这里设置变量 v0 的值大于 10 才向下执行，方法是点击“NOP3”->“信号读写”，再点击添加本地变量将 v0 加入，并将 v0 的最小值设为 10，最大值设为无穷大即填入 INF。



3.1.7 设置仿真

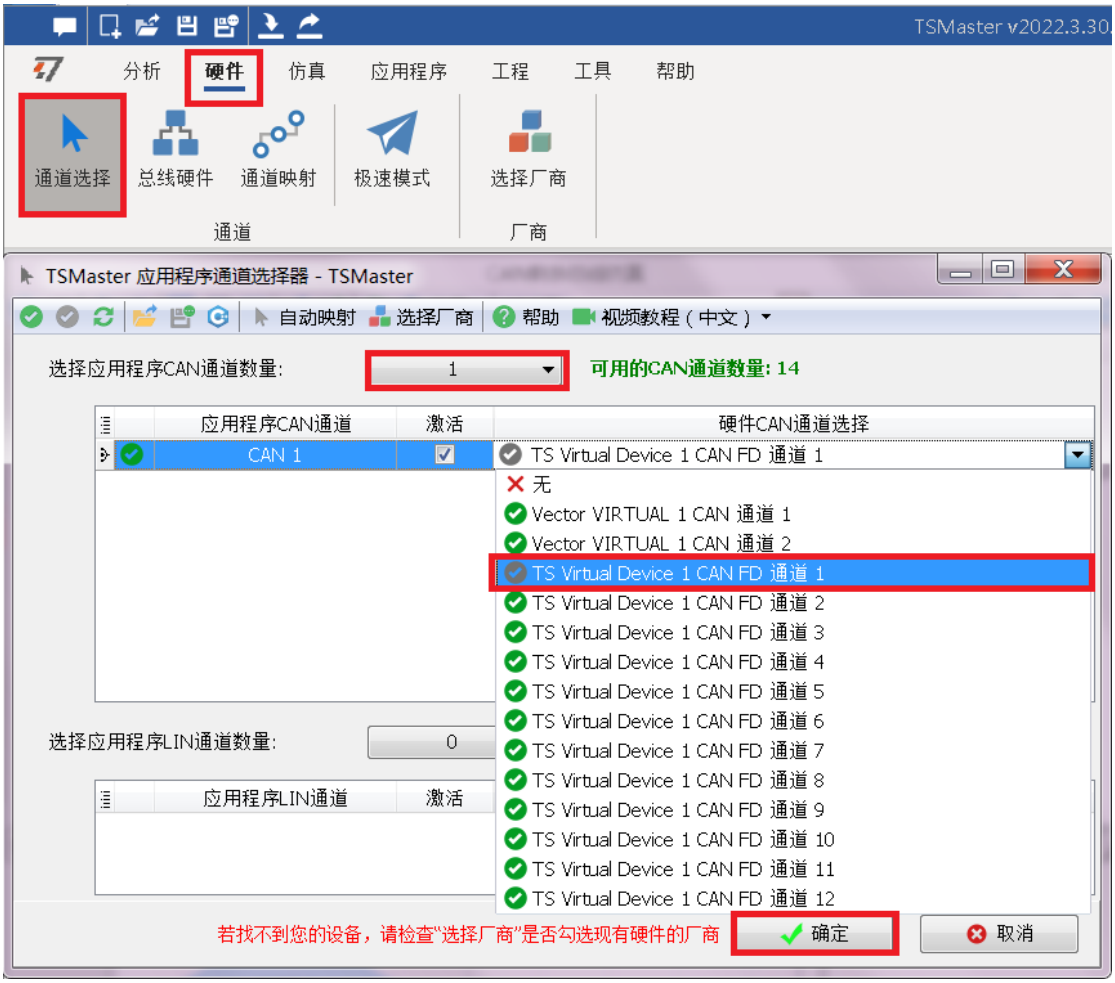
仿真就是用一个模拟的信号代替真实的信号来观察程序运行的效果。设置如下：点击“仿真”->“CAN 总线仿真”->“Engine”，在“过滤器框”输入“car”找到“CarSpeed”并点击，再点击左上角的箭头“运行剩余总线仿真”，最后在“数据”栏输入该信号的初始值，这里输入“4”，然后关闭该窗口。



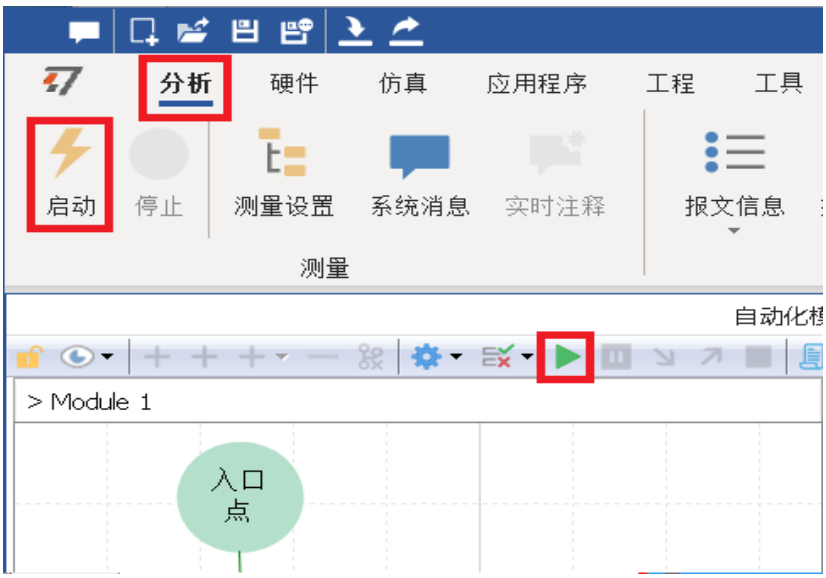
3.1.8 启动仿真

在启动仿真之前须先设置好虚拟通道，方法如下：点击“硬件”->“通道选择”，

在“选择应用程序 CAN 通道数量”为 1，在“硬件 CAN 通道选择”一栏选择任意一个可用通道，在此选择了“Vector VIRTUAL 1 CAN 通道 1”。

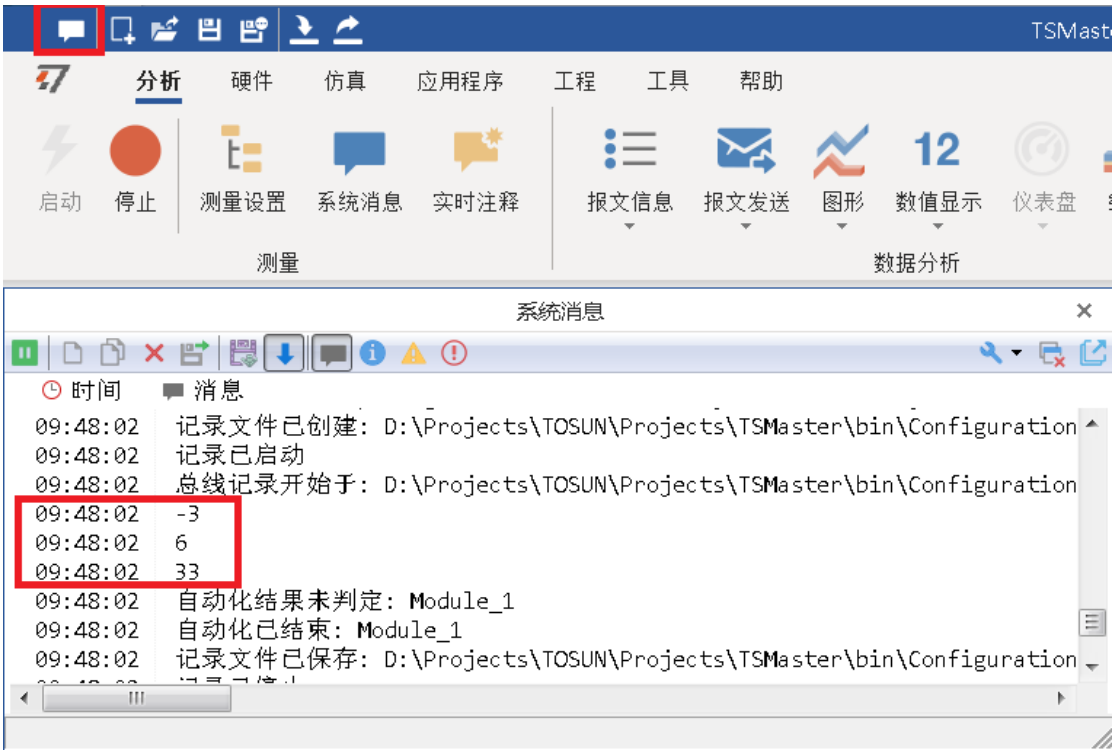


虚拟通道设置好后按下述方法即可启动仿真测试了。点击“分析”->“启动”->“运行自动化模块”就启动完成了。



3.1.9 观察运行结果

本例子中是以 v0 的值来判断程序运行的方向，可以打开信息窗口观察 v0 的变化状况。在程序运行结束后点击“系统消息”即可观察到 v0 变化情况。



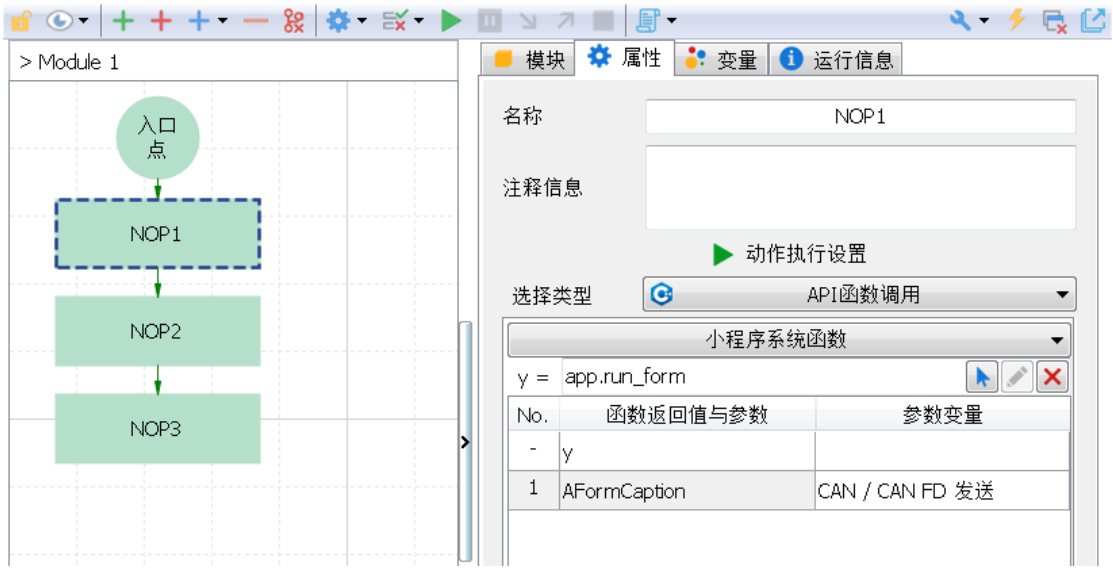
4 使用自动化模块发送 CAN 报文

4.1 使用自动化模块发送窗口 CAN 报文

本章节介绍一个使用自动化模块实现自动控制软件内子窗口的自动化的例子，在此使用“CAN / CAN FD 发送”窗口。例子内容为启动发送窗口内所有非周期和周期报文后，延时一段时间后停止所有的报文发送。

4.1.1 新建自动化模块

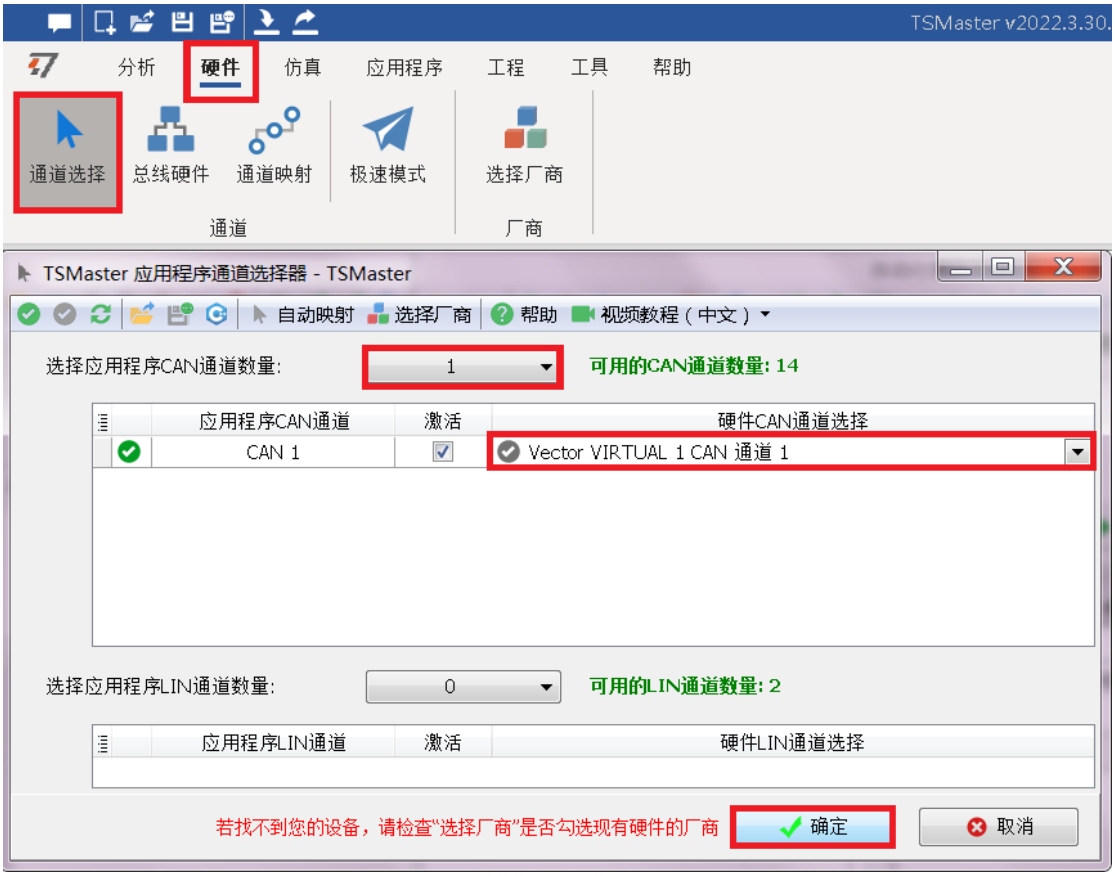
按照前面介绍的方法创建一个自动化模块如下图：



4.1.2 连接一个虚拟 CAN 设备

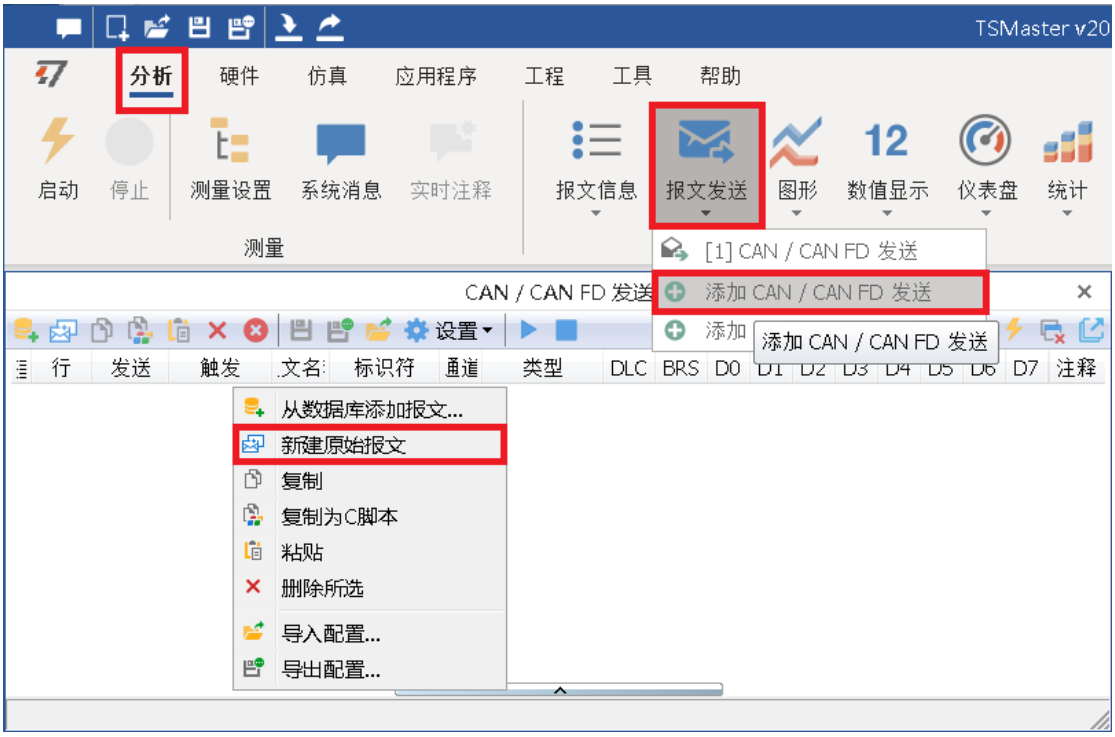
连接一个虚拟的 CAN 设备供自动化模块控制发送 CAN 报文。如果有真实设备，也可连接真实设备通道。

打开“硬件”窗口连接一个虚拟的 CAN 通道，如图：

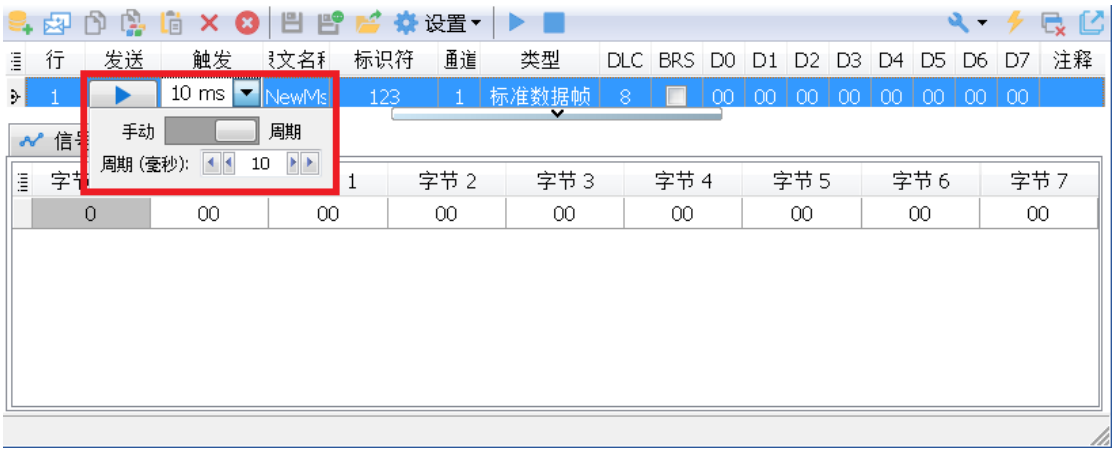


4.1.3 添加一个发送报文窗口

打开“分析”窗口，点击“报文发送”并点击添加一个“CAN / CAN FD 发送”窗口，在窗口的空白处点击鼠标右键“新建原始报文”，如图：

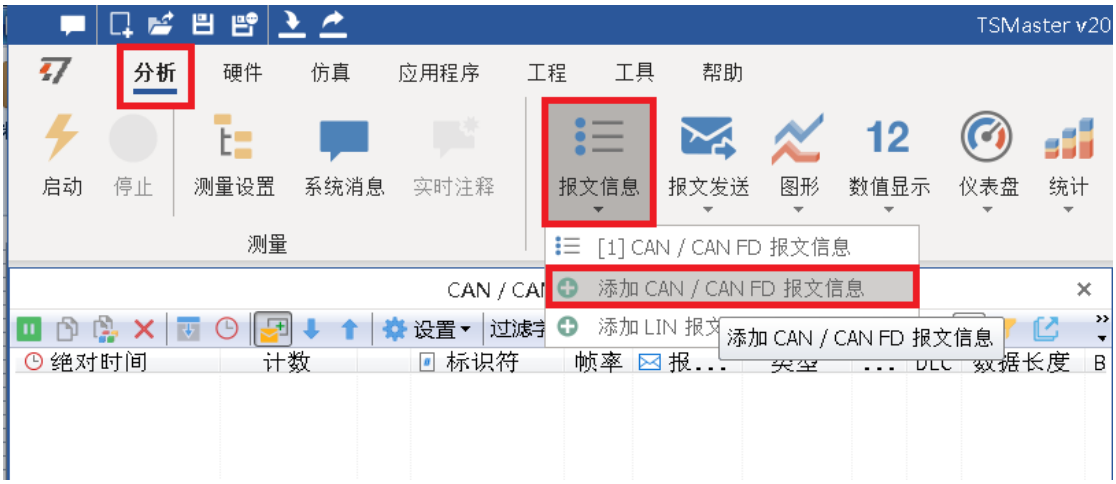


在新建的报文中“触发”一栏选择周期发送，发送间隔选 10ms 即 100 周每秒。



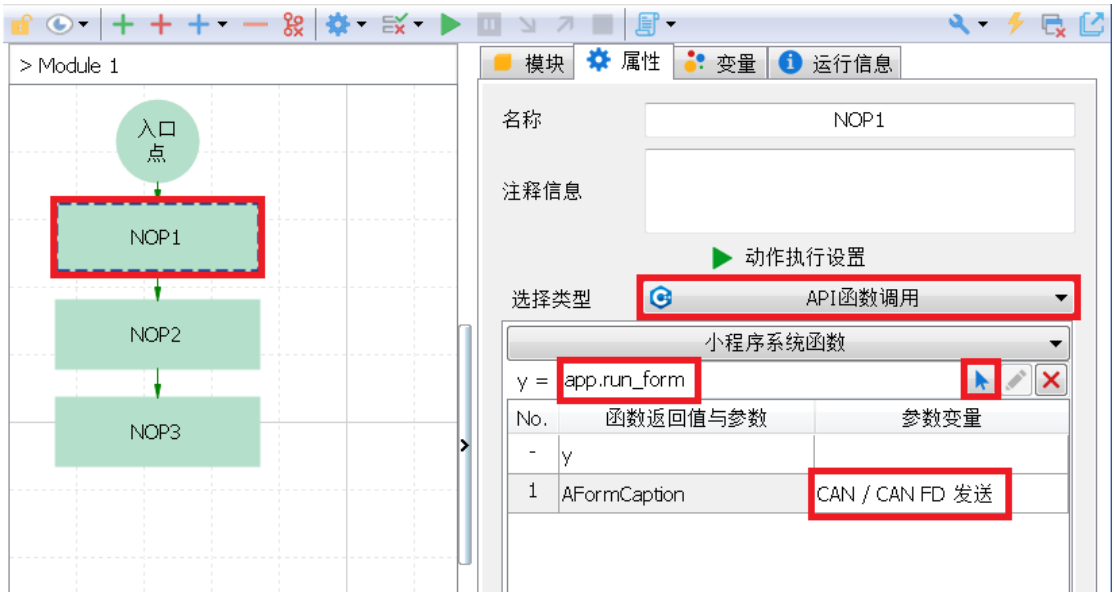
4.1.4 添加一个报文接收信息窗口

打开“分析”窗口，点击“报文信息”并点击添加一个“CAN / CAN FD 报文信息”窗口，用来观察 CAN 报文发送的各个状态，如图：



4.1.5 在自动化模块添加发送 CAN 报文函数

按照前面章节介绍的添加函数的方法，将“NOP1”动作设置成函数“app.run_form”，意图为启动窗体的在线测量功能，并将“AFormCaption”参数设置成 CAN 报文发送窗口的标题“CAN / CAN FD 发送”，注意加入空格。

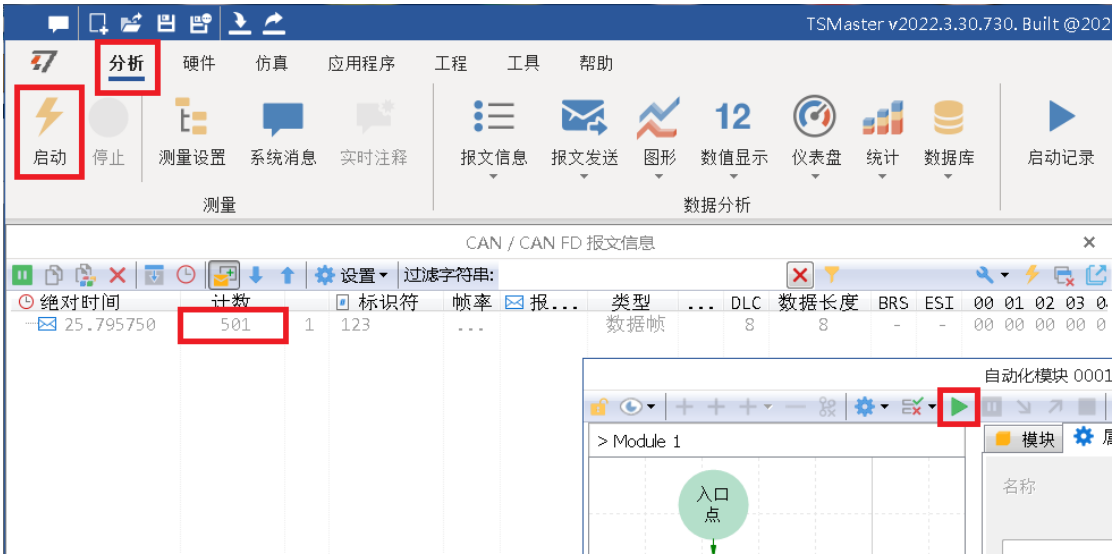


将“NOP2”动作设置成延时函数“app.wait”并将“ATimeMs”参数设置成 5000 即延时 5 秒，以利于观察动作执行情况。

最后再将将“NOP3”动作设置成函数“app.stop_form”并将“AFormCaption”参数设置成 CAN 报文发送窗口的标题“CAN / CAN FD 发送”。

4.16 运行自动化模块并观察发送 CAN 数据状态

打开“分析”窗口，点击“启动”连接已创建好的虚拟 CAN 通道，再打开已创建好的报文接收信息窗口，最后点击自动化模块窗口的运行按键，CAN 报文就被自动按照 10ms 发送一帧的间隔共发送 5 秒钟的时间，看发送信息窗口的“计数”一栏显示共发送了 500 帧左右的 CAN 数据。

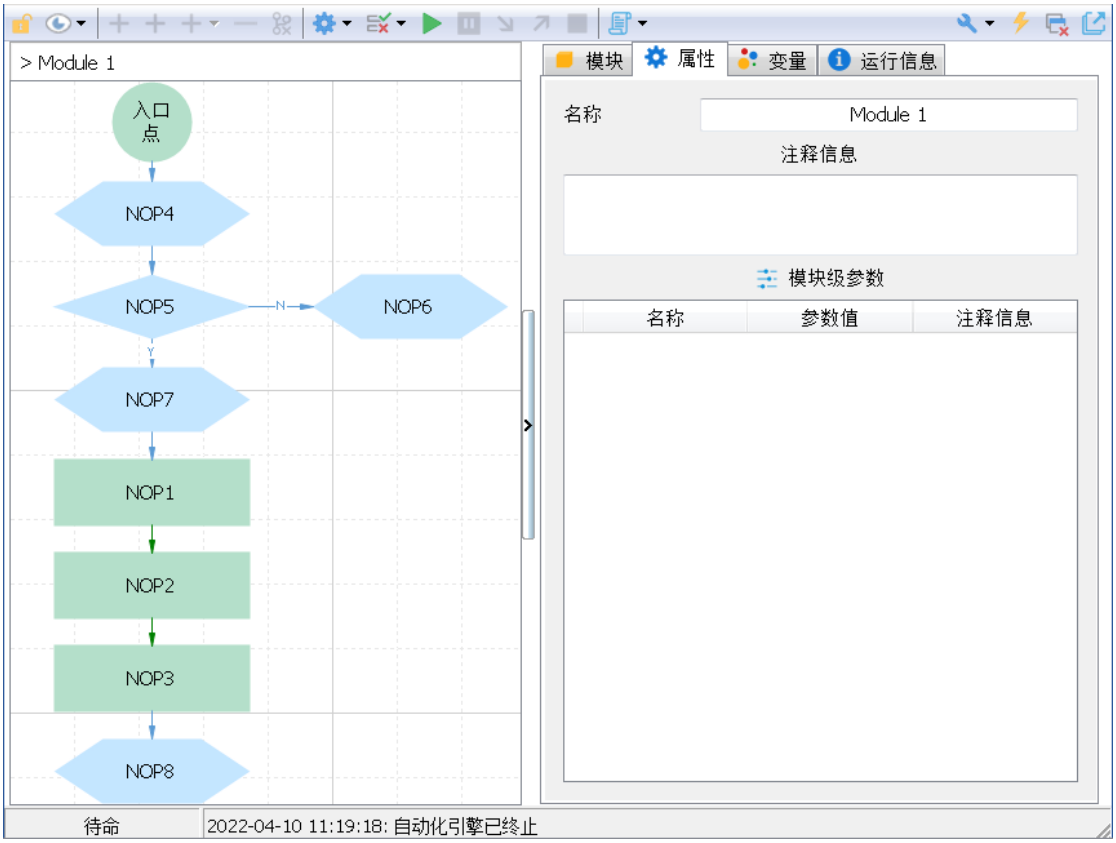


4.2 使用自动化模块测试仿真数据并发送 CAN 报文

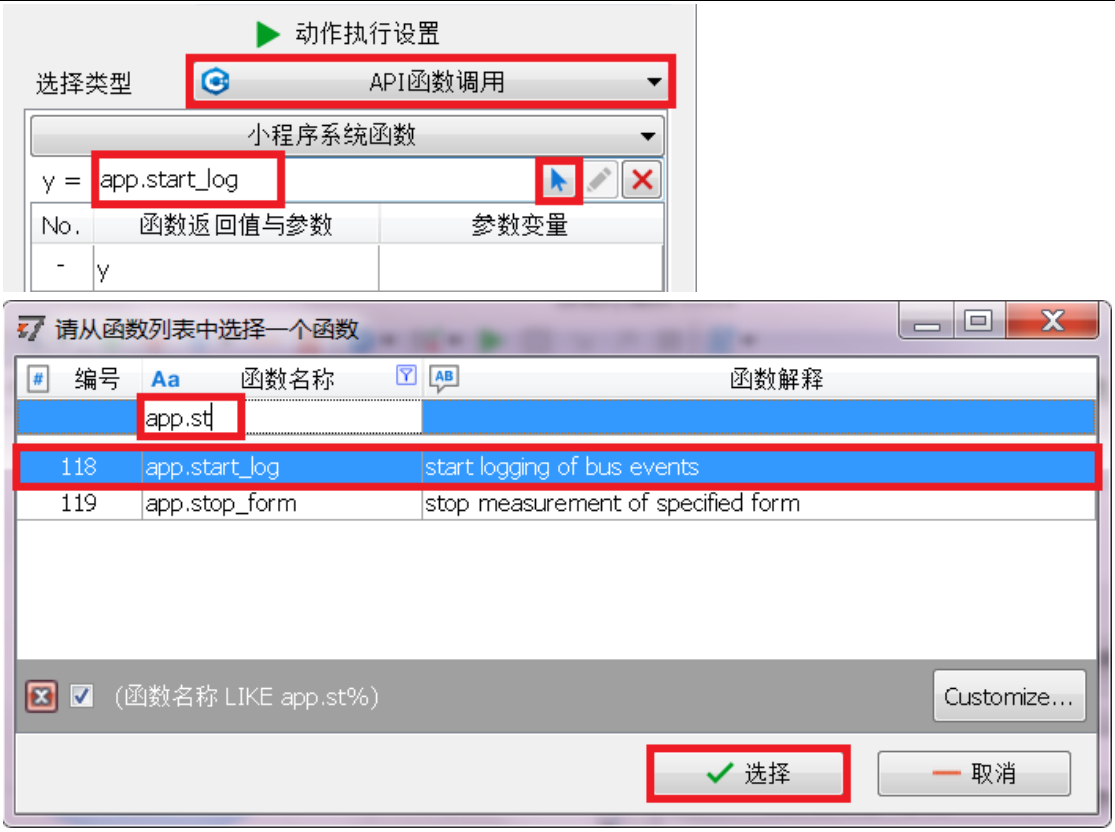
本节介绍一个例子，即在上一节的发送模块基础上增加一个测试 CAN 数据的动作，在该数据达到一定的值后再发送报文，并把整个过程以“blf”的格式记录下来。

4.2.1 添加动作节点

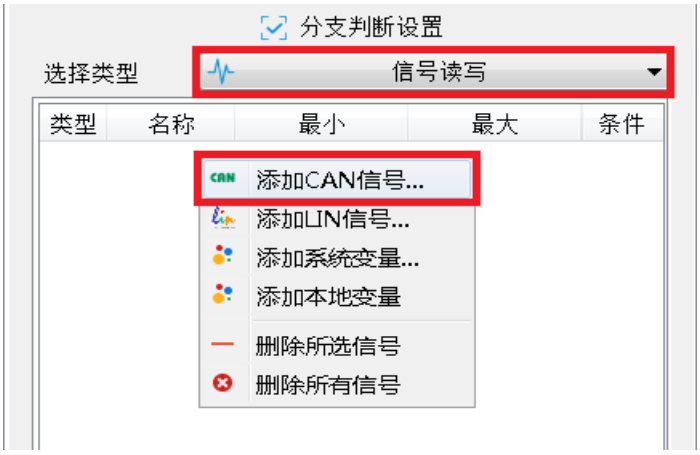
在上一节的基础上添加动作 NOP4-NOP8，每一个动作的名称都可以根据该动作的功能来定义，这里为叙述方便未作更改。添加动作后的流程图如下：



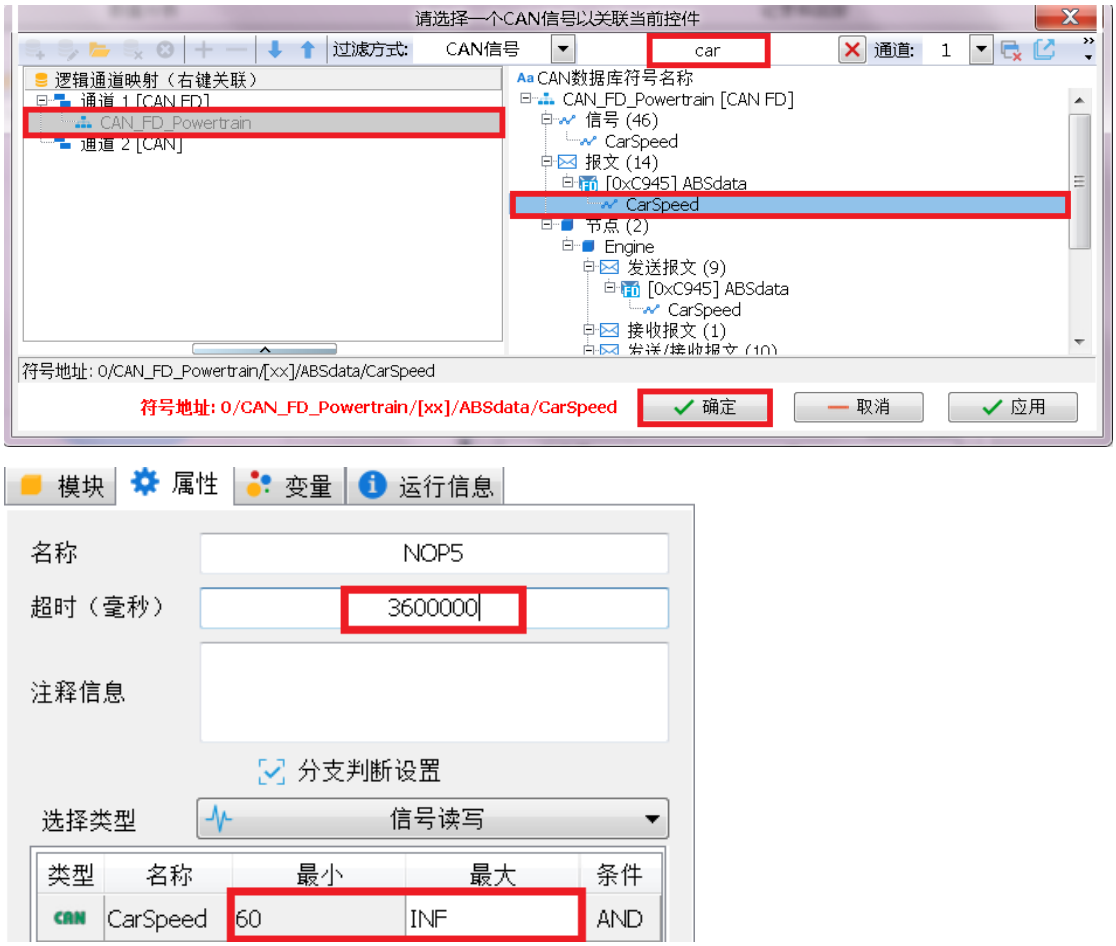
动作“NOP4”设置成具有启动报文记录的功能，按照前面介绍的添加系统 app 的方法添加函数“app.start_log”，可在函数添加列表窗口的搜索框输入“app.st”字样可快捷的找到所需的函数。该函数不需要设置参数，当该动作执行时就启动记录功能。



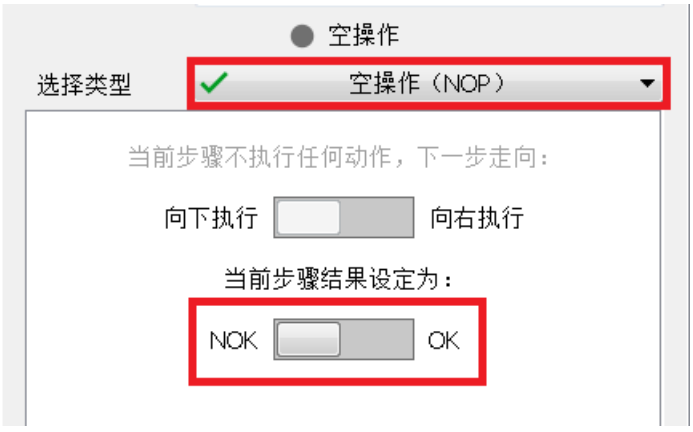
动作动作“NOP5”设置成“信号读写”且具有分支的功能，用来监听接收的仿真 CAN 报文数据，这里取 CAN DBC 文件中的“CarSpeed”数据。方法是在“信号读写”属性页点击鼠标右键，在弹出的菜单上选择“添加 CAN 信号”。



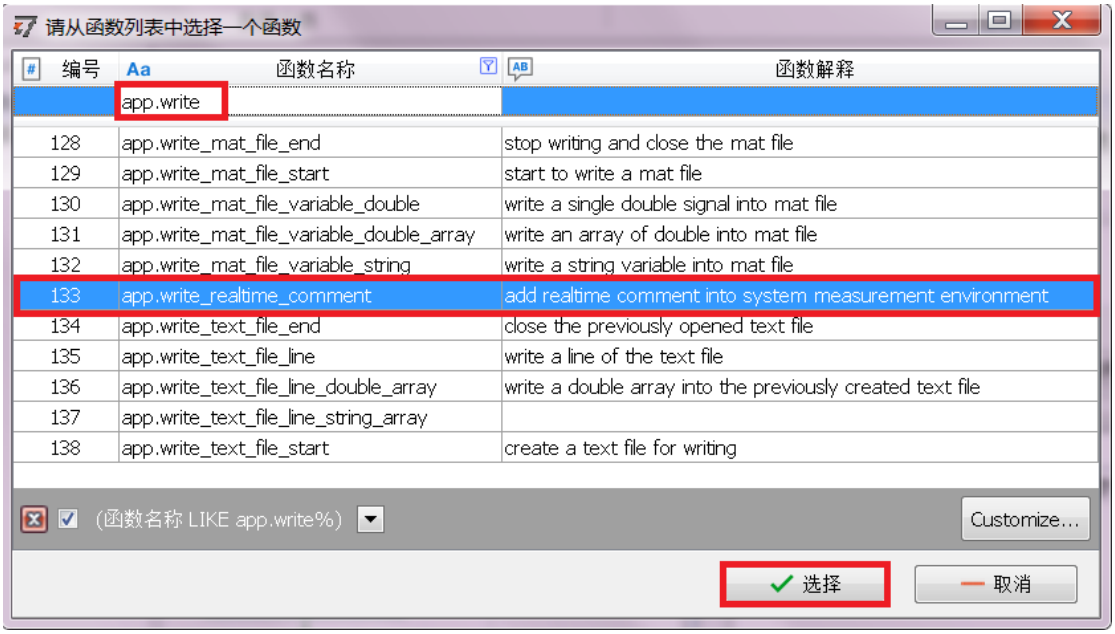
在“CarSpeed”一栏的最小、最大分别填入“60”和“INF”，并在“超时”一栏填入 3600000ms 即 1 小时的延时，这个动作表示当 CarSpeed 的值大于 60 公里每小时就向下执行动作，在 1 小时延时时间到后 CarSpeed 的值仍小于 60 公里每小时就执行向右的动作。



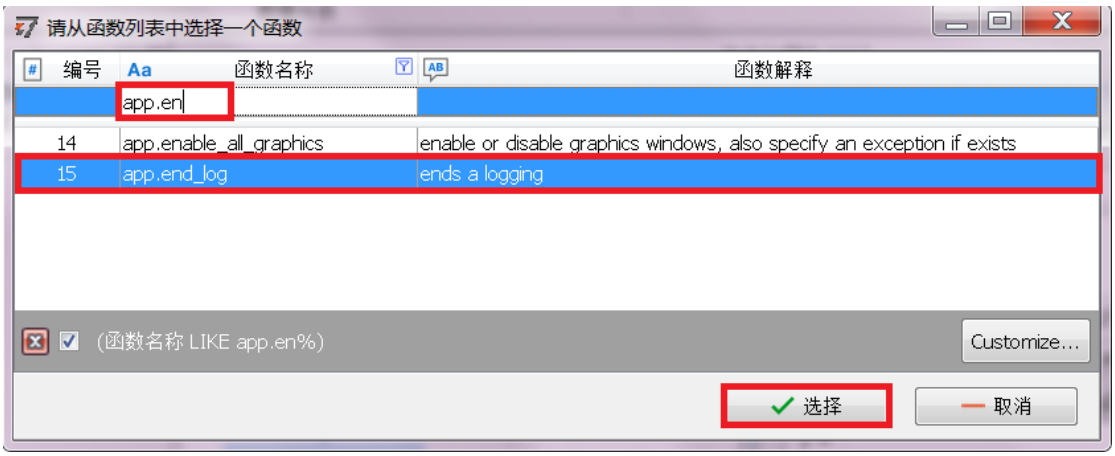
动作“NOP6”设置成空操作，当前步骤结果设定为“NOK”。



动作“NOP7”选择一个具有添加注释信息功能的函数“app.write_realtime_comment”用来记录一个注释信息在图表中。函数返回值与参数“AComment”一栏选择 CAN 信号“CarSpeed”，这样就可以把“CarSpeed”的实时值记录在图表中。

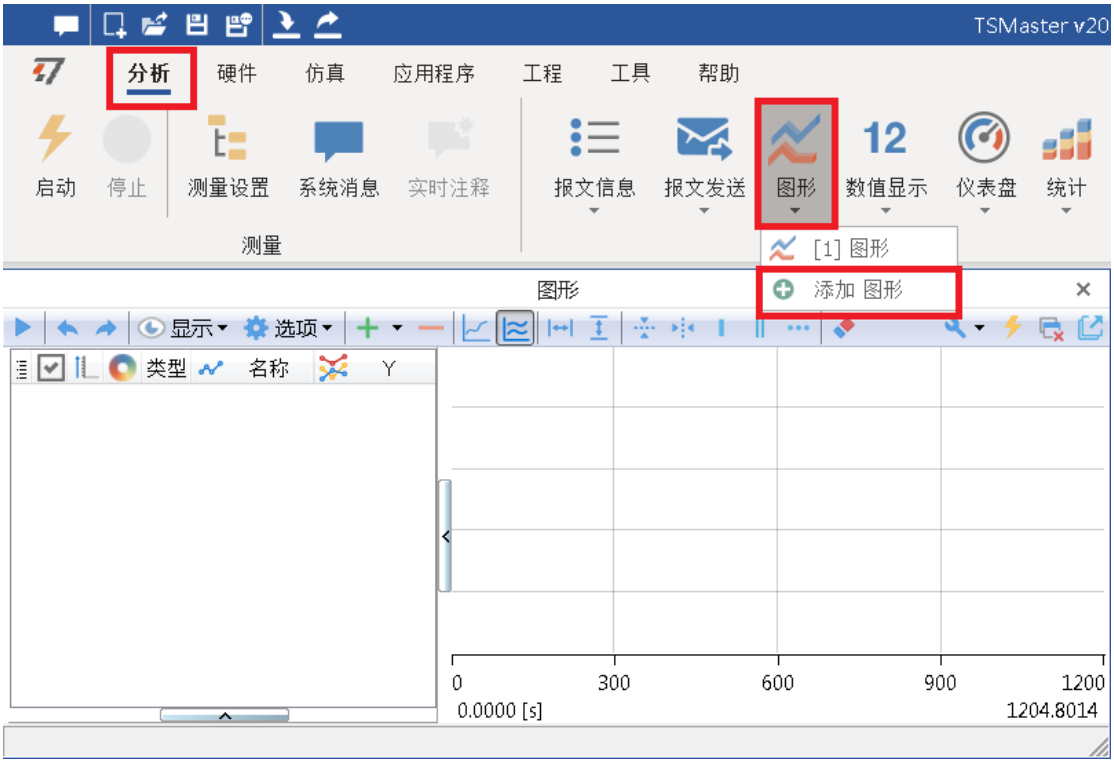


动作“NOP8”选择停止报文记录的函数“app.end_log”。

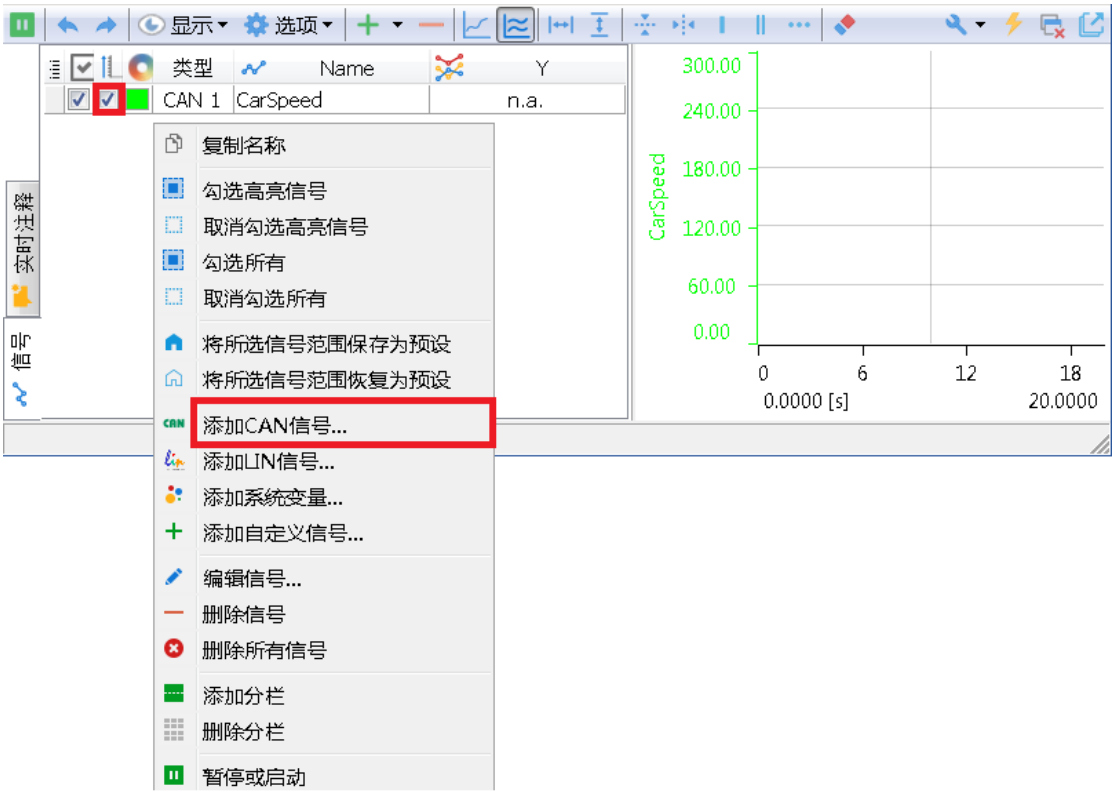


4.2.2 添加图形显示窗口

图形显示窗口可以将选中的 CAN 信号实时的动态的显示在图表中，可按下述方法进行添加。

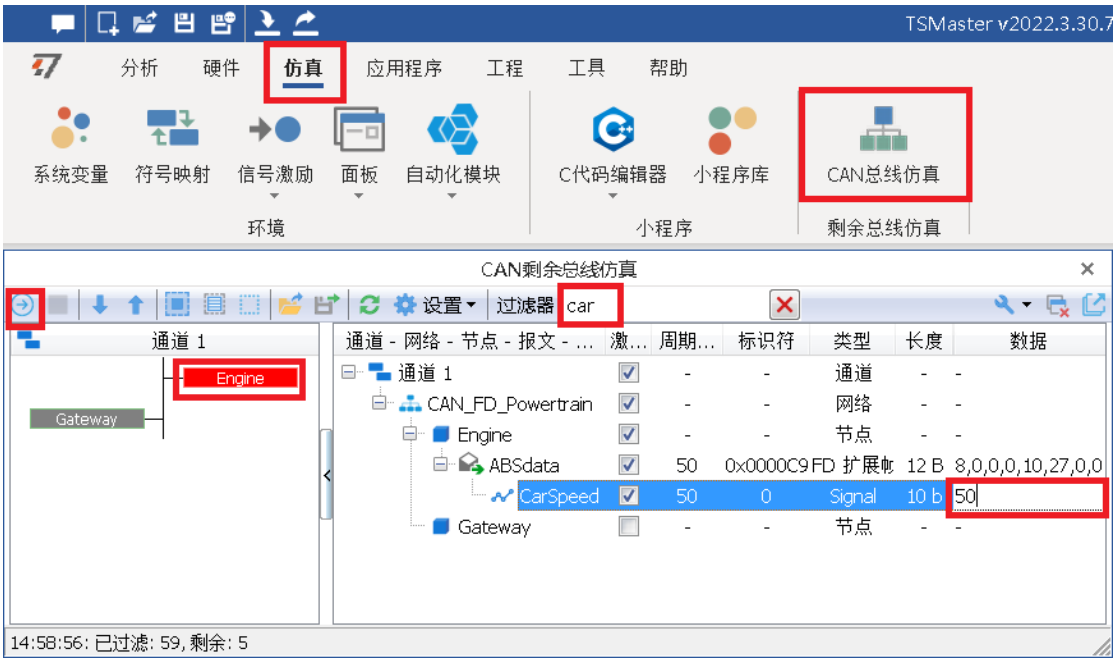


在窗口的左方空白处点击鼠标右键并选择“添加 CAN 信号”，接着在弹出的窗口中选择要显示图形的 CAN 信号“CarSpeed”，将坐标显示选择框打勾。

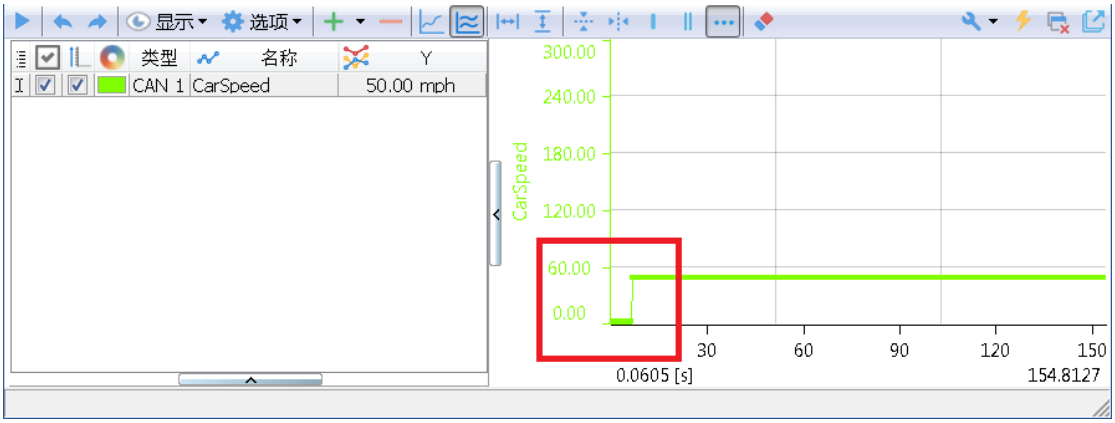


4.2.3 启动仿真信号

打开仿真窗口，启动车速信号“CarSpeed”的自动发送，并将车速信号设成 50 公里每小时，如下图：



这时打开图形显示窗口可以看到显示的车速波形。

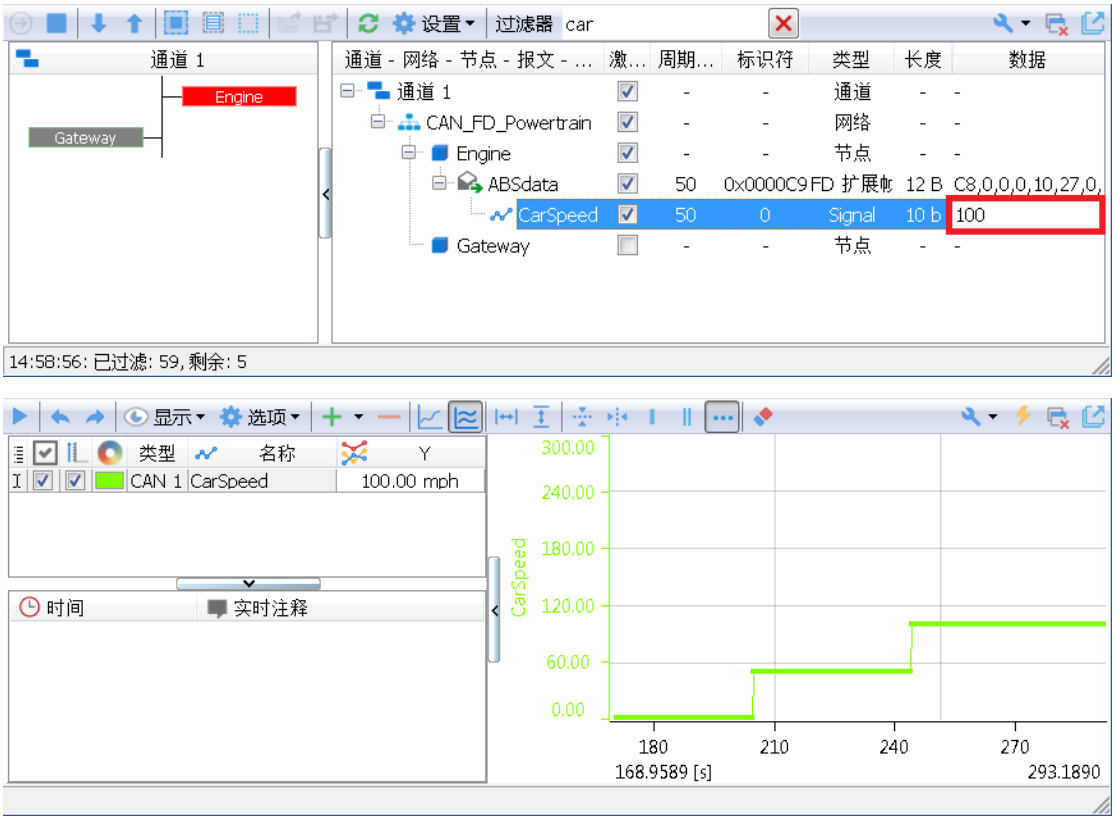


4.2.4 运行自动化模块

当仿真模块启动后再点击自动化模块的运行按钮，程序运行先开始记录 bjf 文件，后停在“NOP5”动作上，该动作是表达式读写带分支功能，因为读取到的仿真信号“CarSpeed”的值设成 50 公里每小时，而表达式的最小值是 60 公里每小时，所以程序在该动作等待，若超时（1 小时）则向右执行。

这时再打开仿真窗口，将车速“CarSpeed”的值改成 100 公里每小时并回车，可以看到自动化模块的程序就开始向下执行，先在图形窗口完成一个注释信息，

并发送 CAN 报文 5 秒钟，最后关闭发送，关闭 blf 文件的记录。



4.2.5 查看 blf 文件

点击“分析”->“记录文件夹”就可以打开记录 blf 文件的文件夹窗口，可以看到一个 blf 文件“TSMaster_2022_xx.blf”，记录了 5 秒内发送 CAN 报文的信息。