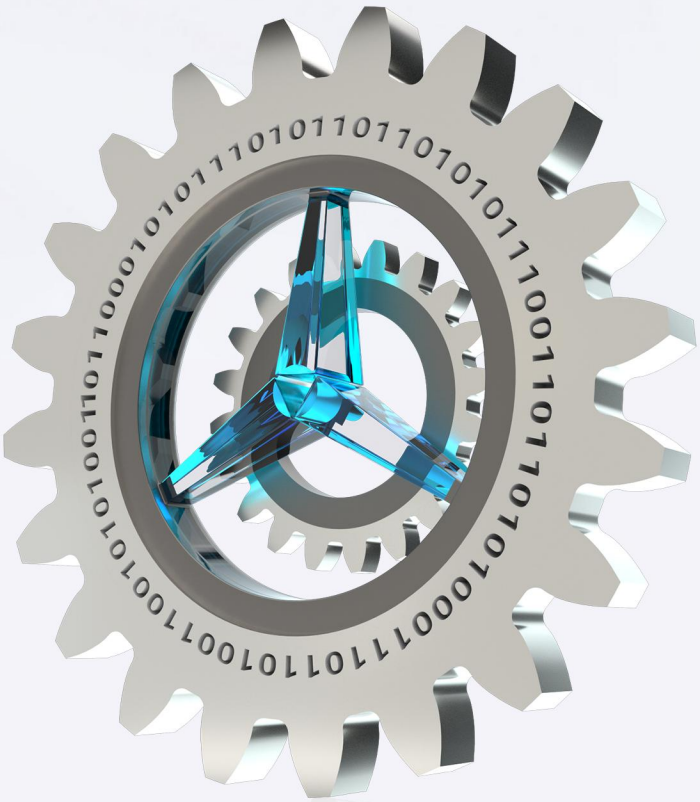
专业·创新·责任·服务



ICS Studio编程软件

用户手册

www.i-con.cn

英孚康（浙江）工业技术有限公司
Info Convergence Technology Co,Ltd

1.1 ICS Studio 软件

ICS Studio 是英孚康 ICC 系列控制器的通用编程平台，在一个编程环境中就可以对 ICC-T、ICC-P、ICC-B 系列的控制器进行编程。提供离线、在线编写程序和程序的上下载功能，支持结构化文本（ST）和梯形图（LD）等两种编程语言。

ICS Studio 软件具备以下这些主要特点：

- 易于配置，软件提供了图形化的控制管理器、I/O 配置对话框、运动配置等工具。
- 使用数组和用户定义的结构进行精密的数据处理，这使得应用程序更加活，而不是迫使应用程序适应由控制器数据表内存定义的特定内存结构。
- 简单易用的 I/O 寻址方法。
- 功能丰富的结构文本语言指令集。
- 诊断监控功能，使用趋势图组件来提供图形的实时数据直方图来诊断和监控数据。此外，语言编辑器、标签编辑器和数据监控器全都包含一个快速交叉引用工具，该工具可以快速定位到特定的标签、说明文本或指令。
- 高集成的运动控制支持。

1.1.1 ICS Studio 软件的组件及功能

首先要了解其界面组成和功能。

当使用 ICS Studio 新建一个工程项目时，会看到如图 1-1 所示的界面。

界面中各组成部分的功能如下：

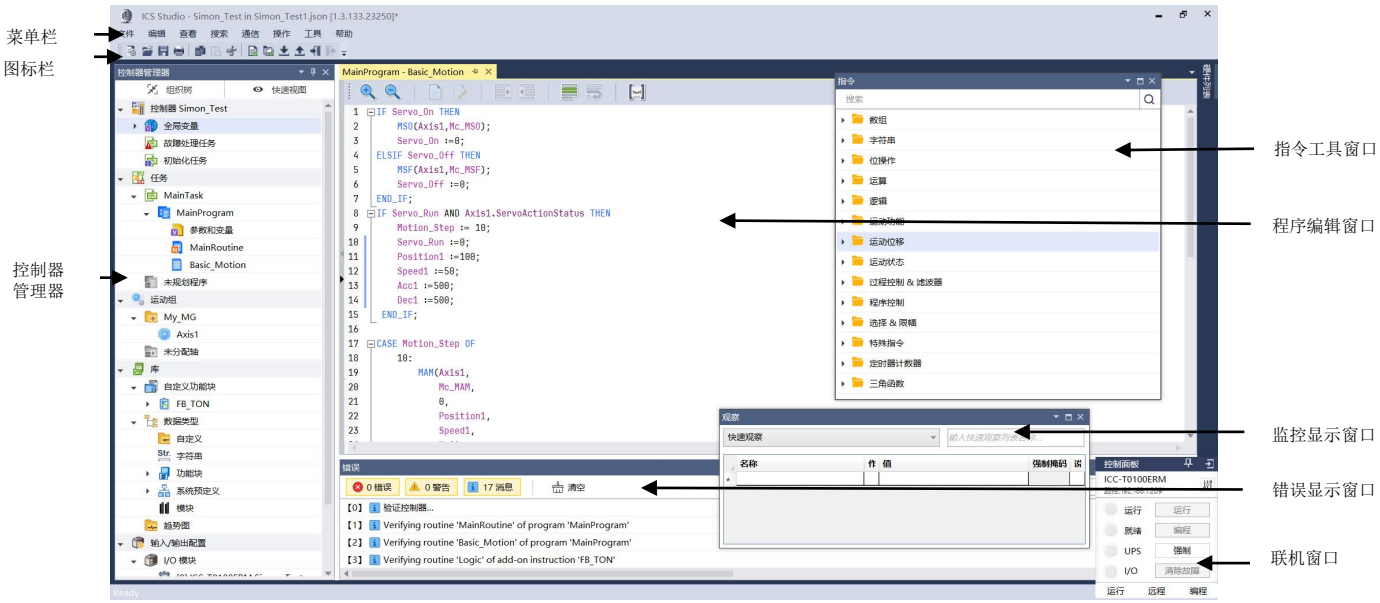


图 1-1

菜单栏：通过点击菜单，选择所显示的功能。

图标栏：包含许多用户在开发、调试程序时需反复使用的功能。用户将光标移动到图标上，会显示该图标的作用。

控制器管理器：是项目内容的图形表示。它由文件夹及文件夹的分层树组成，这

些文件和文件夹包含了关于当前项目中的程序和数据的所有信息。
指令工具窗口：显示按照文件夹分类的指令助记符。
程序编辑窗口：用户进行结构化文本（ST）编程的区域。
监控显示窗口：在线监控部分或当前编程页面所有标签的显示区域。
错误显示窗口：显示校验程序结果的区域，可将其隐藏或分离出来放置在屏幕任何地方。
联机窗口：提供控制器的状态，可在此窗口进行控制器模式切换、上下载程序操作。

1.1.2 ICS Studio 项目结构

一个 ICS Studio 的项目主要是由任务（task）、程序（Program）和例程（Routine）组成的。有的项目中还包括上电处理程序和控制器故障程序，其各部分的关系如图 1-2 所示。

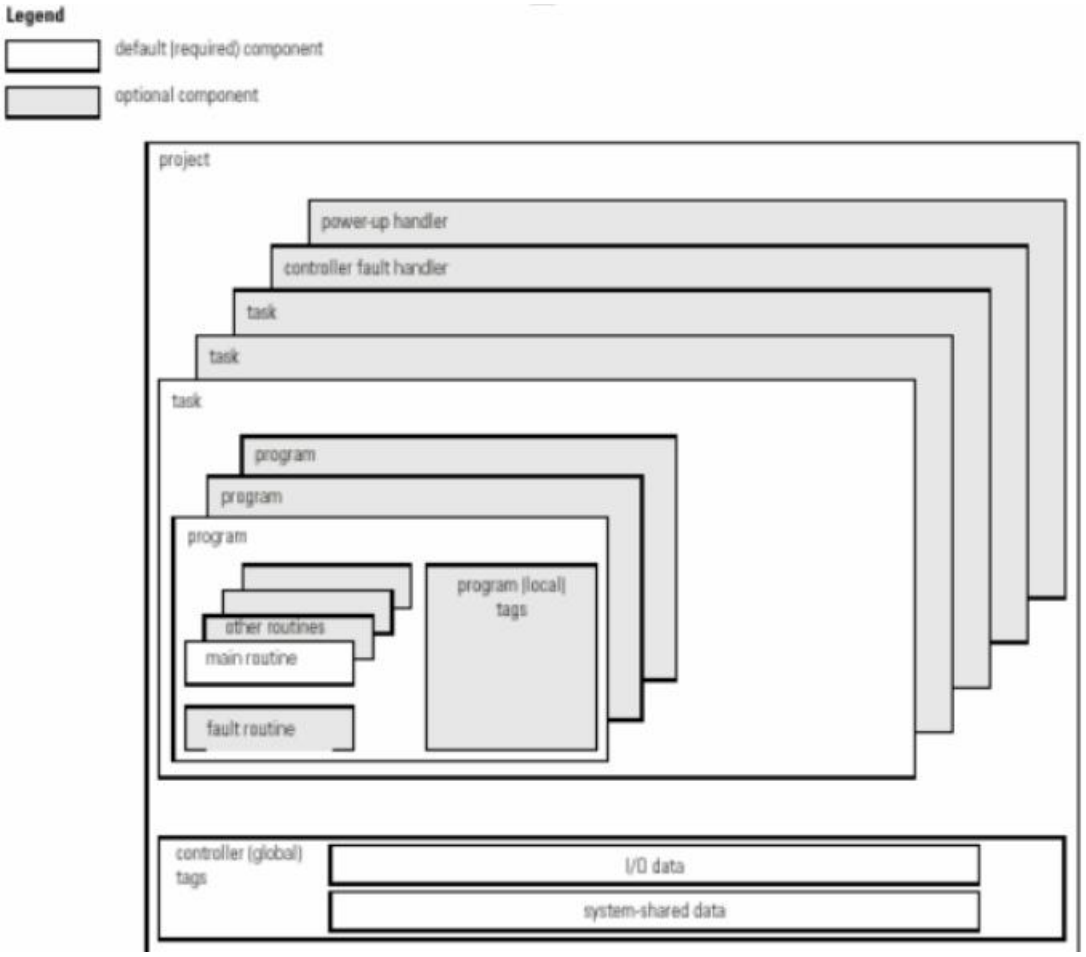


图 1-2

这些组成部分将按照表 1-1 所示的方式配合工作。

表 1-1　ICS Studio 项目组成及其定义

项目组成	定义
Task	提供由一个或多个 program 组成的 program 集的规划和优先级信息。当创建新项目时，软件将自动创建一个连续 task。当这个 task 完成完全扫描时，它将立即重新启动。
Program	每个 task 都至少需要一个 program，每个 program 有其自己的 program tag、主 routine 和其它 routine，一个 program 只能在一个 task 中规划，不能在多个 task 之间共享一个 program。
Routine	为控制器中的项目提供可执行的代码，每个 routine 都使用特定的编程语言（例如，结构文本或梯形图）
主 Routine	当一个 program 执行时，它的主 routine 首先执行。使用主 routine 来通过“跳转至子 Routine”(JSR)指令调用其他 routine（子 routine）
子 Routine	除主 routine 以外的任何 routine。要执行一个子 routine，请在另一个 routine（如主 routine）中使用“跳转至子 Routine”(JSR)指令

1.2 ICS Studio 的使用步骤

在这一节中，将以压缩机装配的整个工艺流程为例子来说明 ICS Studio 软件使用的常规步骤。

在该例子中，传送带上的压缩机经过三个装配站：冲压、卷边和焊接。然后，压缩机被传送到第二个传送带并接受质量检查。通过检查的压缩机码垛后装船运走。如图 1-3 所示。

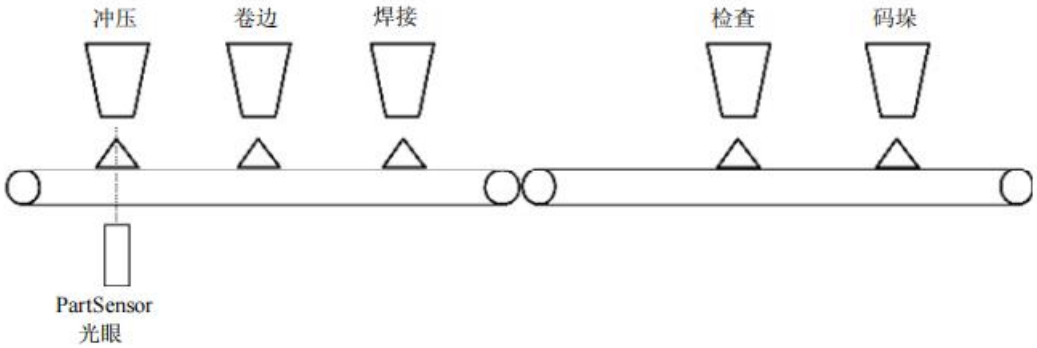


图 1-3

冲压、卷边和焊接三个装配站和传送带 1 由控制器 P1 控制，质量检查和码垛站以及传送带 2 由控制器 P2 控制。光眼检测到有部件放置到传送带上（Part Sensor 由 0 变为 1）后，站 1、2 和 3 顺序执行，然后传送带动作。当光眼再次检测到有部件送至传送带上，上述操作再次执行，以此循环。下面我们以时序图方式描述控制器 P1 的操作流程，如图 1-4 所示。

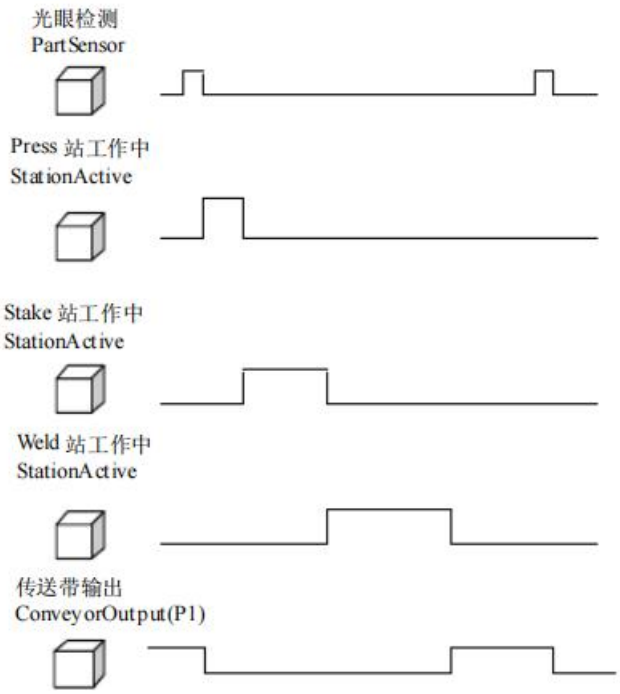


图 1-4 时序图

1.2.1 创建工程

1. 打开 ICS Studio 软件，单击*文件->新建* 或  创建新项目。这时出现 *新建项目* 页面，在此选择控制器的类型和填入项目名称及项目文件存储路径。如图 1-5 所示。

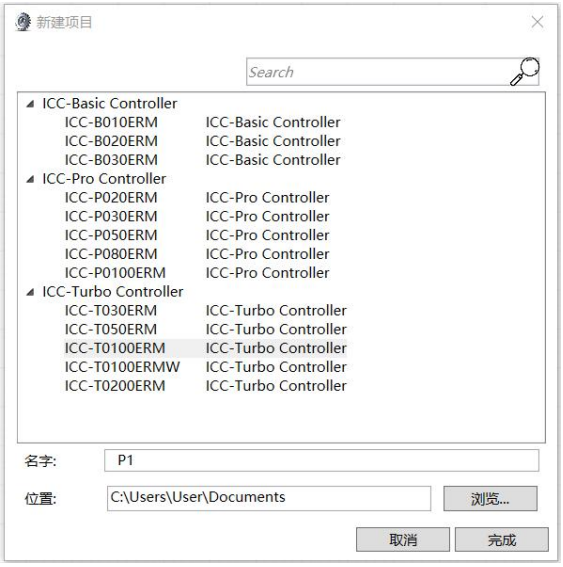


图 1-5 新建控制器对话框

单击 *完成* ，弹出工程画面，如图 1-6 所示。

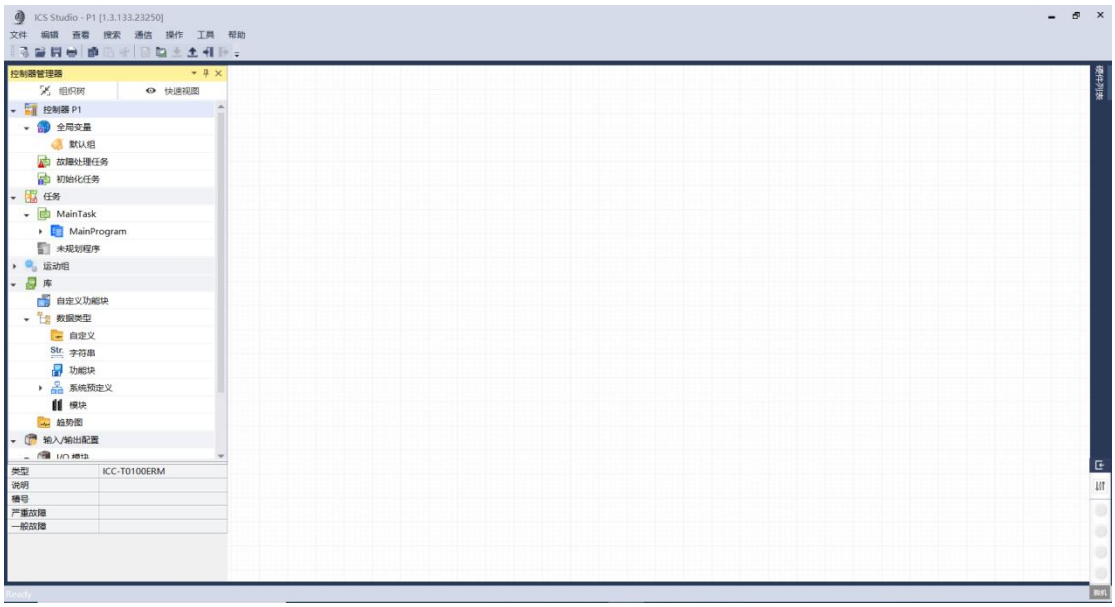


图 1-6 新建项目资源管理器

现在已经创建了一个 ICS Studio 项目。此时还没有添加任何与项目相关的 IO 模块，项目中也没有可执行的代码，正处于离线状态。所做的任何改变都仅限于软件中，并存储在计算机的硬盘中。在进行连线操作之前，这些变化并不能反映到 ICC 控制器中。

1.2.2 创建任务、程序、例程

2. 分析应用项目要求：接下来，根据应用实例要求来组织控制器 P1 项目中任务、程序和例程及其操作要求。控制器 P1 项目组织结构，如表 1-2 所示。

表 1-2 控制器 P1 项目组织

任务	包含程序	包含例程	执行动作
Assembly	Program_1_Press	Routine_Dispatch	使能子例程
		Station_1_Press	控制冲压站例程
	Program_2_Stake	Routine_Dispatch	使能子例程
		Station_2_Stake	控制卷边站例程
	Program_3_Weld	Routine_Dispatch	使能子例程
		Station_3_Weld	控制焊接站例程
Conveyor	Conveyor	Conveyor	控制传送带操作
Periodic_Dispatcher	Station_Dispatcher	Station_Dispatcher	初始化（使能）站操作例程

操作要求分析如下：

控制器 P1 中的任务必须符合如下要求：

- 装配线任务（站 1,2,3）
  - 执行时间不超过 500ms
  - 根据调度连续运行
- 传送带任务
  - 执行时间不超过 500ms

- 与调度任务分时执行（两任务的优先级相同）
- 每 50ms 执行一次

- 调度任务
  - 执行时间不超过 400ms
  - 与传送带任务分时执行（两任务的优先级相同）
  - 每 50ms 执行一次

3. ICC 控制器不仅支持连续型（Continuous）任务，还支持周期型（Periodic）任务。根据上述 P1 的操作要求，确定控制器 P1 中各任务的属性，如表 1-3 所示。

表 1-3 控制器 P1 中各任务的属性

Task（任务）	Type（类型）	WatchDog（看门狗）	优先级	执行周期
Assembly（装配线）	连续型任务	500ms	——	——
Conveyor（传送带）	周期型任务	500ms	5	50ms
Periodic_Dispatcher（定期调度）	周期型任务	400ms	1	50ms

4. ICC 控制器仅支持一个连续型任务，并且 ICS Studio 已经自动创建了一个连续型任务 MainTask。在控制器管理器的 MainTask 文件夹上单击右键，在弹出窗口上选择 *属性* ，将 MainTask 任务的名称改为 Assembly，并输入相应属性值。如图 1-7 所示。

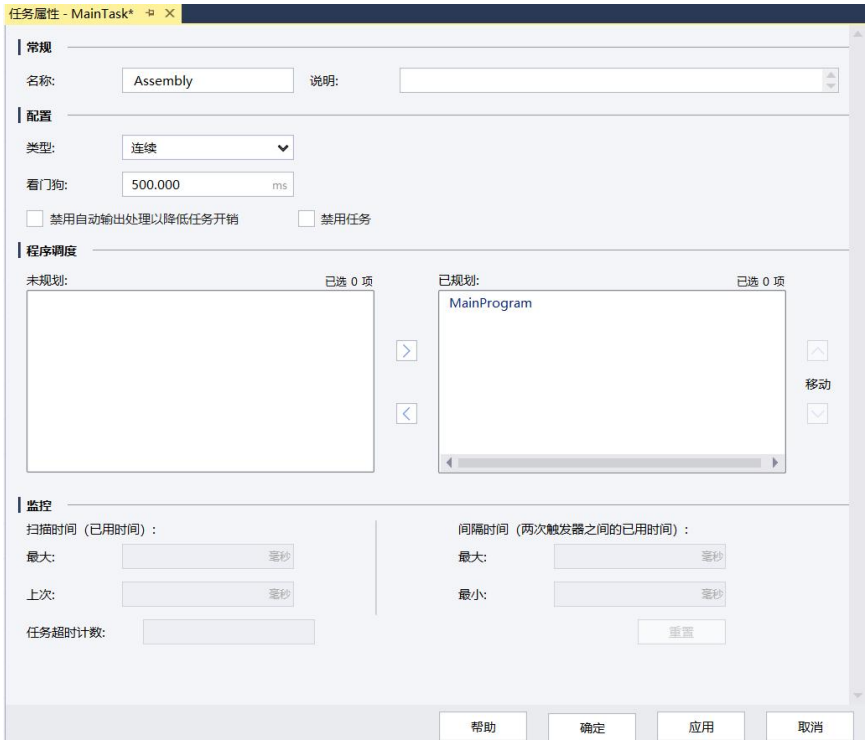


图 1-7 连续型任务属性

5. 在控制器管理器 *任务（Task）*文件夹上单击右键，在弹出菜单中选择 *新建任务*，建新任务 Conveyor，并设置相应属性，如图 1-8 所示。因为传送带任务要求 50ms 执行一次，所以选择的任务类型为周期型（Periodic）。同理，创建新任务 Periodic_Dispatcher，并设置

相应的属性，保存该项目。

图 1-8 创建新任务 Conveyor

6. 创建 Assembly（装配线）任务的程序。在控制器管理器的 Assembly 文件夹上单击右键，在弹出的菜单里选择 *添加->新建程序*，输入程序名称 Program_1_Press，并设置相应属性，如图 1-9 所示。同理创建 Program_2_Stake，以及 Program_3_Weld，并设置相应属性。

图 1-9 创建新程序

7. 规划 Assembly（装配线）任务的程序。右键单击 Assembly 任务，从弹出的对话框中选择 *属性*，将 ICS Studio 自动创建的 MainProgram 程序左移到 *未规划* 列，规划后的程序如图 1-10 所示。

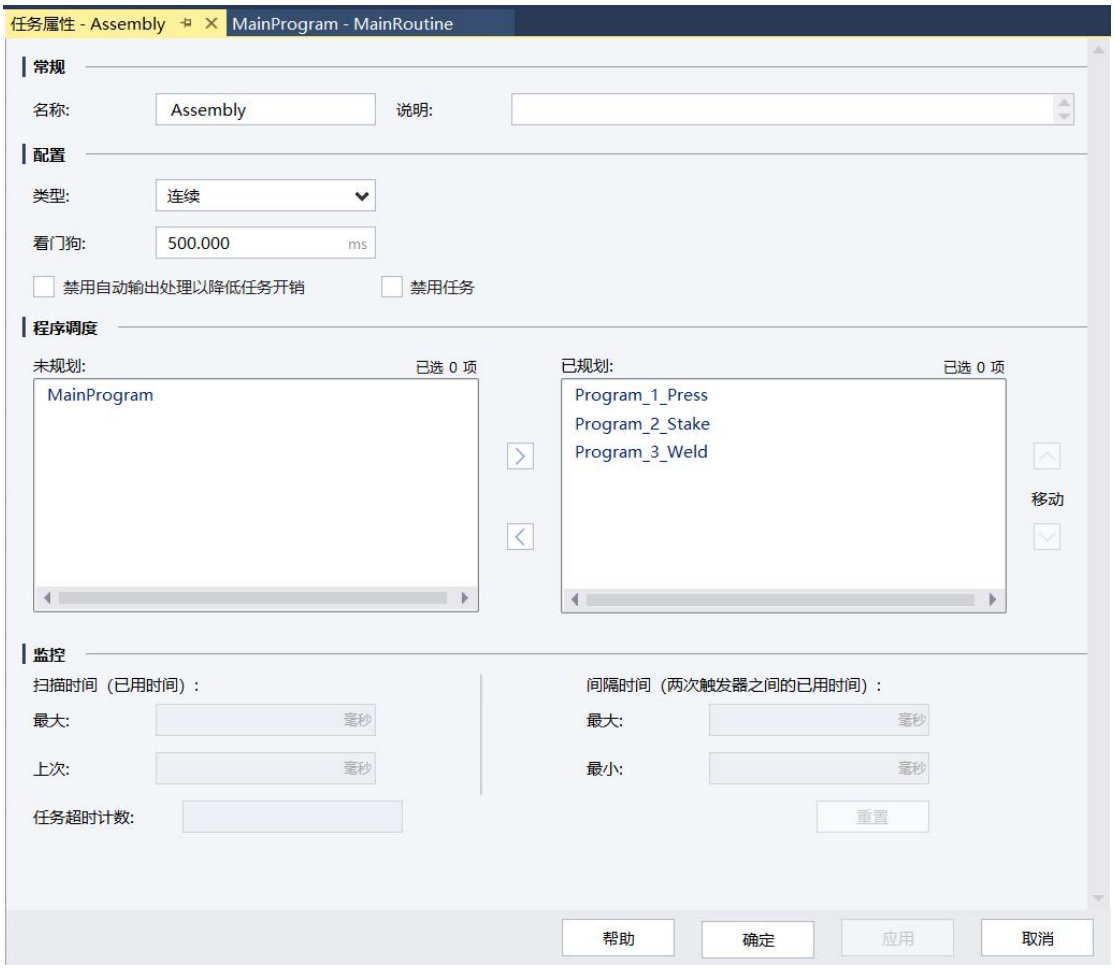


图 1-10 规划程序

8. 为 Assembly（装配线）任务的 Program_1_Press 程序创建例程。右键单击 Program_1_Press 程序，在弹出菜单中选择 *添加->新建例程*，在弹出的对话框内输入名称 Routine_Dispatch（调度例程），类型为 *结构化文本*， 隶属于程序 Program_1_Press，如图 1-11 所示。该例程用于调度程序中其他子例程，该例程自动被认作 Program_1_Press 程序的主例程（主例程为该程序第一个创建的例程）。

图 1-11 创建例程

同理，创建 Station_1_Press（冲压）例程，类型为 *结构化文本*，隶属于 Program_1_Press 程序。该例程用于控制冲压工序的时间。

9. 按照相同的步骤，用户可自行为 Program_2_Stake, Program_3_Weld 程序创建相应的例程，注意创建的顺序，以确保正确的主例程。

10. 对于 Conveyor（传送带）和 Periodic_Dispatcher（定期调度）任务，如图 1-12 所示，执行如下操作：

- ✓ 创建所需程序
- ✓ 创建所需例程

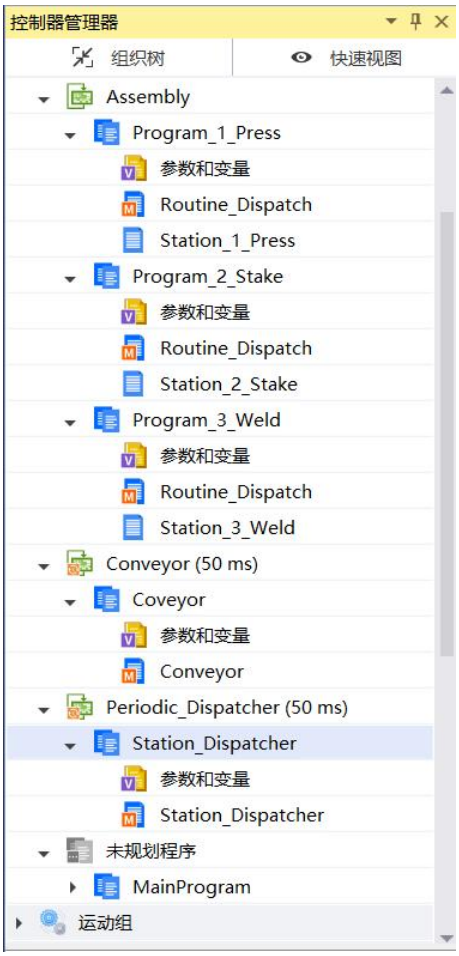


图 1-12 新建任务、程序和例程

11. 单击 *文件->保存* 或  图标保存该项目。该项目所有的任务、程序、例程都创建完毕。

1.2.3 创建标签、结构体和数组

在上一节中，创建完了任务、程序和例程，接下来要创建相应的标签、结构体和数组。ICC 控制器的特点是无需手动进行 I/O 映射，根据控制属性，自动创建/命名标签，并且支持结构体和数组。另外，控制器域（全局）和程序域（局部）标签的分类提高了代码的重用性。

12. 右键单击 *全局变量->默认组*，在弹出菜单选择 *新建变量* ，标签名称类似于其他编程语言中的变量，它们均用于存储数值。在对话框中输入名称 Call_Program_Value，数据类型 INT ，范围为 P1（Controller），外部访问为 None，如图 1-13 所示。

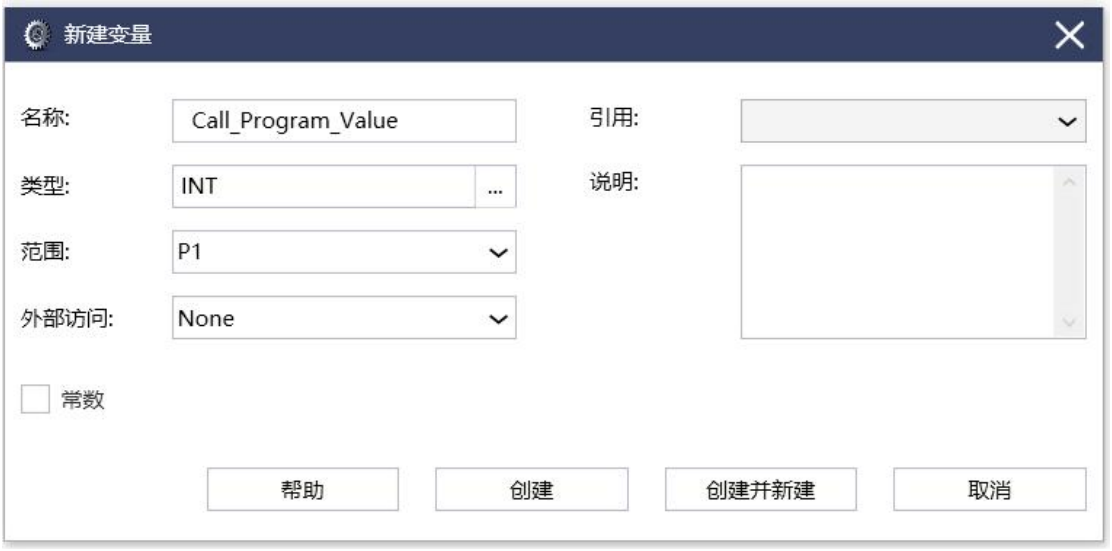


图 1-13 新建标签

13. 按照上述步骤逐个创建一下控制器域的标签，如图 1-14 所示，这些标签将在后面的实验中用到。

名称	数据类型	值	强制掩码	格式	说明
Call_Program_Val...	DINT	0		Decimal	
Complete	BOOL	0		Decimal	
ConveyorOutput	BOOL	0		Decimal	
InCyle	DINT	0		Decimal	
PartSensor	BOOL	0		Decimal	
StationComplete	BOOL	0		Decimal	

图 1-14 控制器域标签

创建下面 Conveyor 程序域内的标签，如图 1-15 所示。

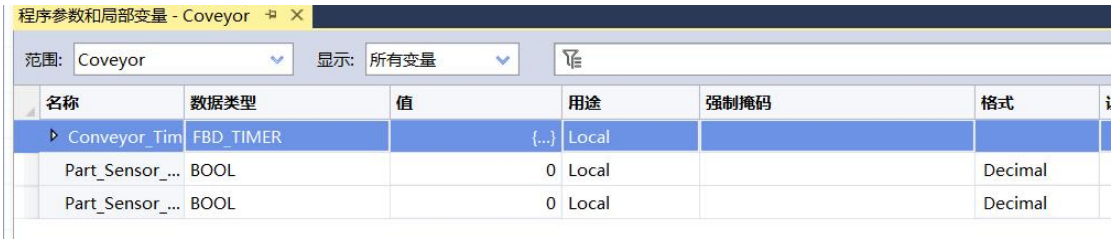


图 1-15 Conveyor 程序域内标签

创建下面 Station_Dispatcher（站调度）程序域内的标签，如图 1-16 所示。

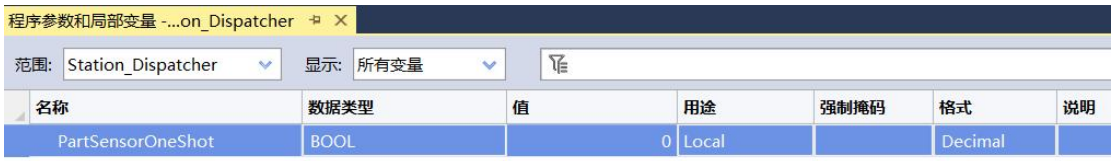


图 1-16 Station_Dispatcher 程序域内标签

创建下面 Program_1_Press（冲压站）程序域内标签，如图 1-17 所示。

图 1-17 Program_1_Press 程序域内标签

14. 将 Program_1_Press 程序域的标签复制并粘贴到 Program_2_Stake 和 Program_3_Weld 程序域内，无需重建标签，提高代码的重用性。在此可以注意到，ICC 控制器中，不同程序域的标签名称是可以相同的。

15. 创建用户自定义数据类型。在控制器 P1 中为每个压缩机生成一个产品编号（Product ID），每个产品编号由零件编号（Part_ID）、序列号（Serial_No）和目录号（Catalog_No）三部分构成。使用用户自定义数据结构可以更方便管理这种数据类型的标签。如图 1-18 所示，右键单击 *库->数据类型->自定义*，在弹出菜单中选择 *新建数据类型*。

图 1-18 新建用户自定义数据类型

16. 在打开的画面中输入自定义数据类型的*名称*和*成员*，如图 1-19 所示。此时，创建

了一个自定义的数据类型，如果需要在例程中使用它，必须创建相应的标签。

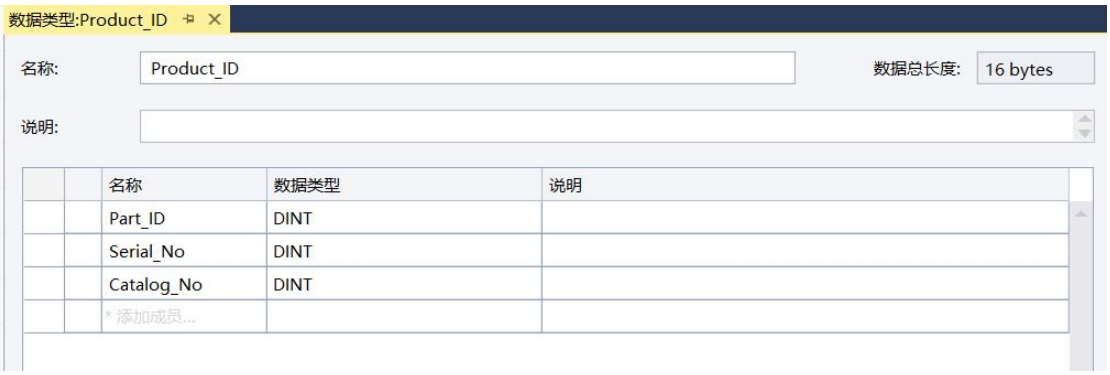


图 1-19　自定义数据类型中的名称和成员

17. 在控制器域内创建创建数据类型为 Product_ID 的标签 Station_Data，如图 1-20 所示。

图 1-20　创建数据类型为 Product_ID 的标签

至此，完成了标签、结构体和数组的创建。

1.2.4 编写结构化文本程序

创建了任务、程序、例程和所需标签之后，我们需要编写工作站（冲压、卷边、焊接）、传送带和站调度的逻辑程序。ICS Studio 支持结构化文本和梯形图等编程语言，用户可以根据自己的需求灵活选择编程语言。对于本例，我们选择结构化文本（ST）编程语言。

18. 输入结构化文本程序。右键单击 *任务->Assembly->Program_1_Press->Routine_Dispatch*，在弹出菜单中选择 *打开*，如图 1-21 所示。

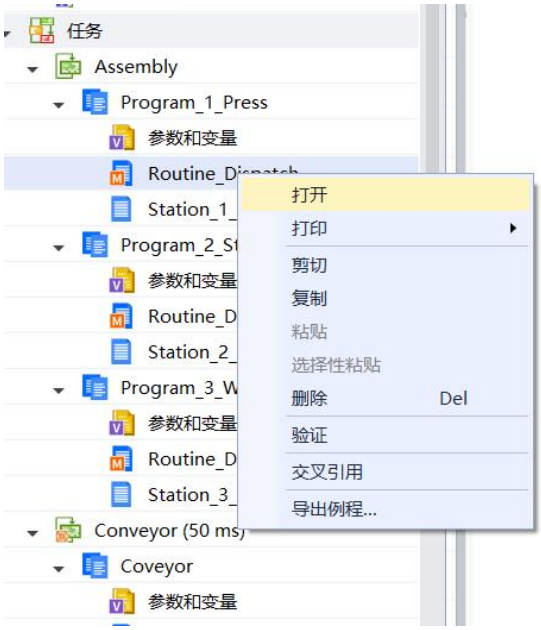


图 1-21 打开 Routine_Dispatch 例程

19. 在打开的编程窗口中编写调度程序，如图 1-22 所示。

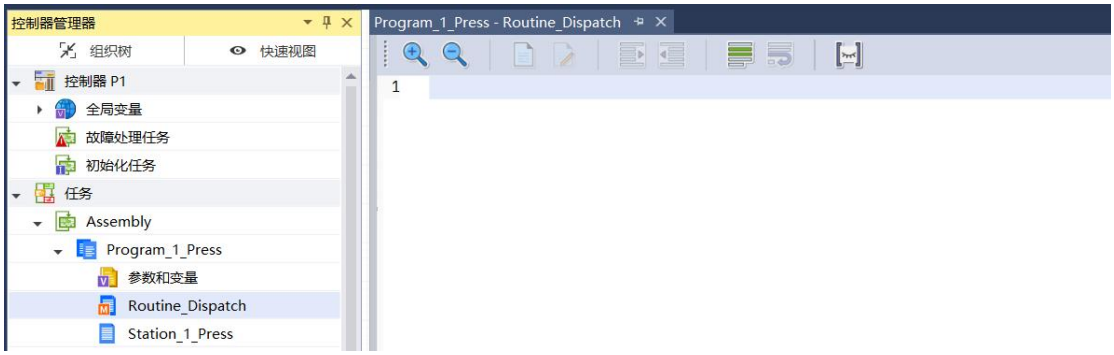


图 1-22 Routine_Dispatch 例程编程窗口

20. Routine_Dispatch 主例程的作用是初始化子程序和调度子程序。初始化子程序将 Station_1_Press 例程中的计时器 StationTimer 复位清零。如果标签 Call_Program_Value（调用程序号）由 Station_Dispatcher 例程设定为 1，则跳转到子程序 Station_1_Press 中。

由以上描述可知，这段程序是个判断条件，我们采用 IF THEN 语句，在程序编写窗口敲入 IF，然后按 TAB 键，程序自动会生成一个 IF THEN 的结构，如图 1-23 所示。

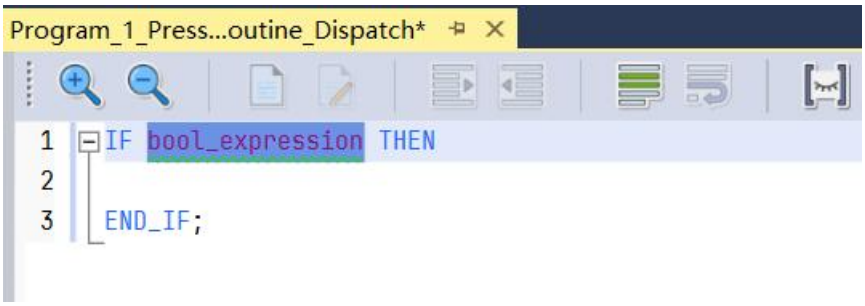


图 1-23 IF THEN 语句结构

21. 现在需要在 IF 后面填入判断条件。判断条件是 Call_Program_Value 是否为 1，因为标签已经预先定义过了，在 IF 后面敲 C，会有下拉菜单自动显示相关标签，双击选择 Call_Program_Value。如图 1-24 所示。

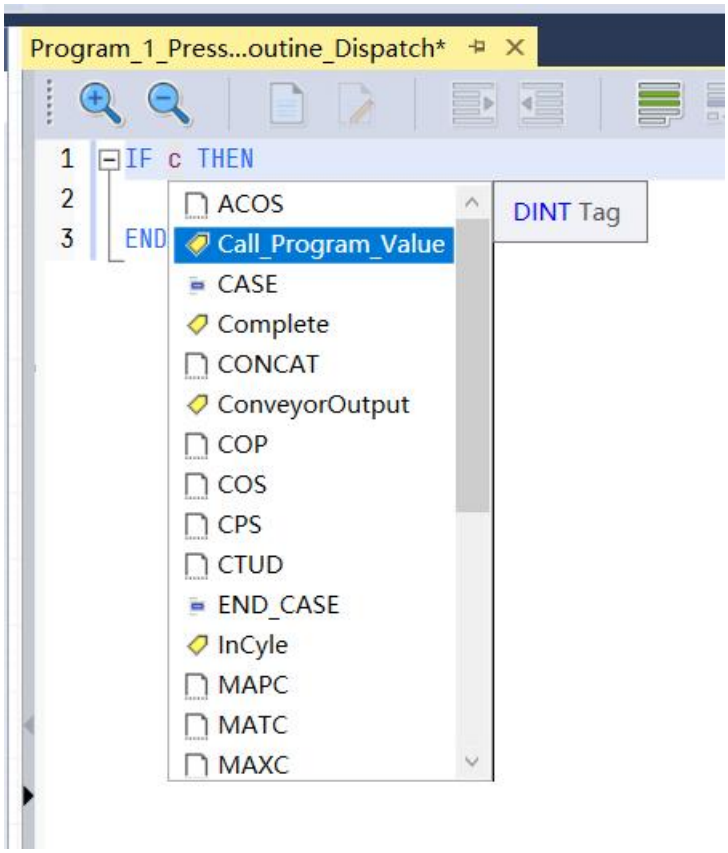


图 1-24 选择已有标签

判断条件如图 1-25 所示。

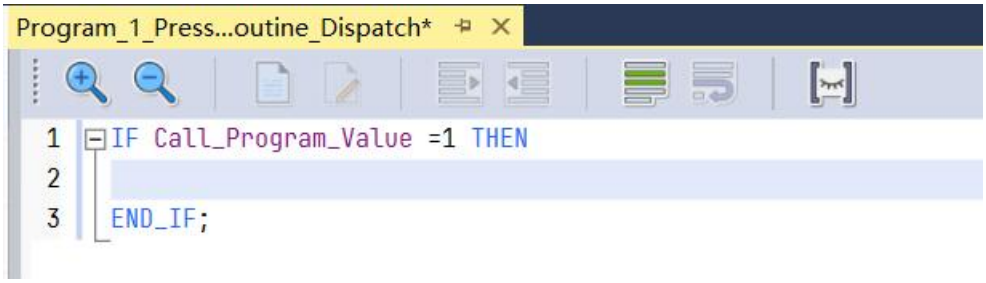


图 1-25 判断条件输入

22. 判断条件如果不使用立即数 1 的话，也可以使用变量，如图 1-26 所示，右键新建变量，并设其初始值为 1。

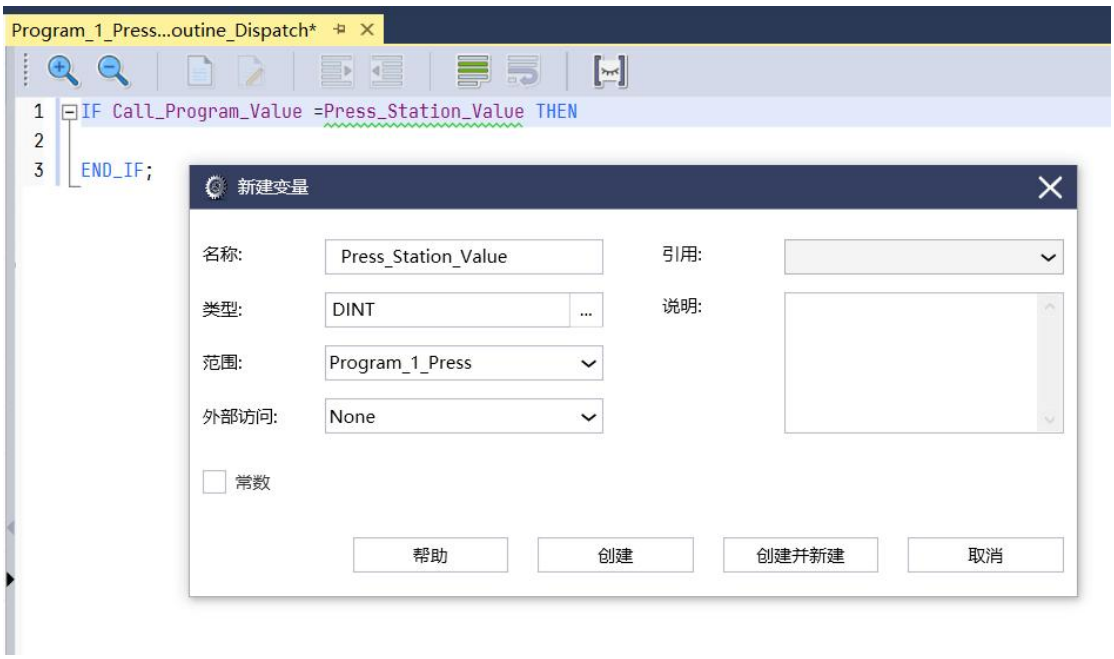


图 1-26 判断操作数为变量

23. 判断条件成立的话，首先要对计时器复位，由于我们不希望复位信号一直被置位，所以需要边缘触发指令 OSRI，清除定时器累加值的程序如下图 1-27 所示。指令的具体用法请参照指令帮助。

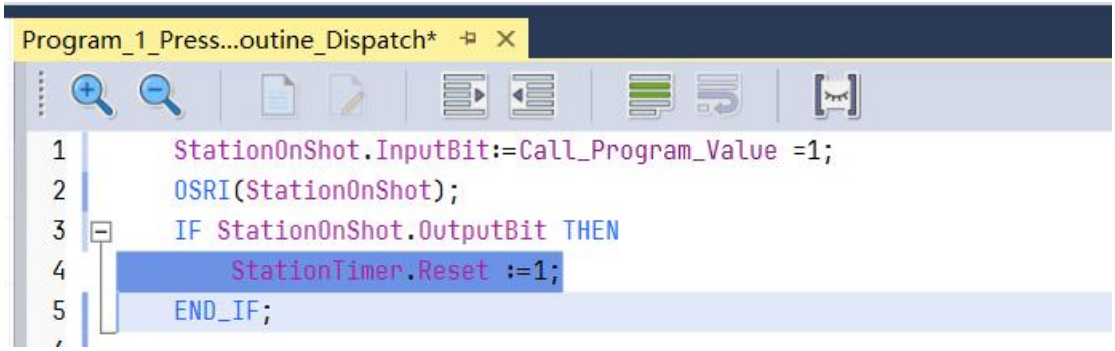


图 1-27 定时器累加值清除程序

24. 判断条件如果成立的话，则跳转执行 Station_1_Press 子程序，开始执行压缩机部件的冲压工序。在 THEN 之后添加 JSR 指令，操作字为被调用子程序的名称。如图 1-28 所示。注意程序的顺序，跳转指令在清除定时器指令之后执行。

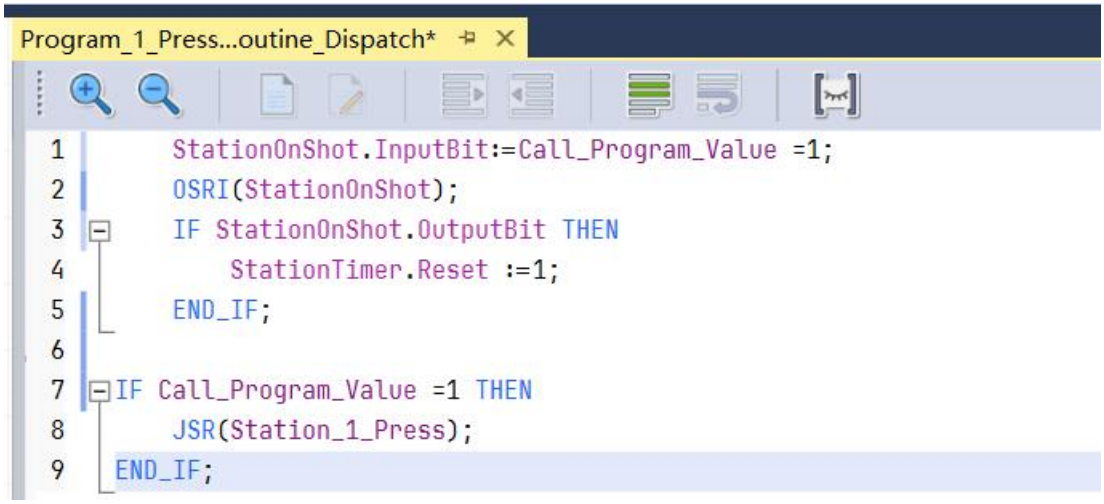


图 1-28 调用子程序逻辑

25. 将 Program_1_Press->Routine_Dispatch 中的程序复制到 Program_2_Stake->Routine_Dispatch 和 Program_3_Weld->Routine_Dispatch 中，需要修改以下参数：

- ✓ 判断条件的第二操作数分别改为立即数 2 和 3，或相应的标签
- ✓ JSR 指令的操作字分别对应各自的被调用子程序

26. 校验，单击 图标校验每个例程，出现错误提示后，纠正错误。然后，单击 图标校验整个项目并纠正出现的错误。

27. 在 Program_1_Press->Station_1_Press 例程中写入如下程序，如图 1-29 所示。

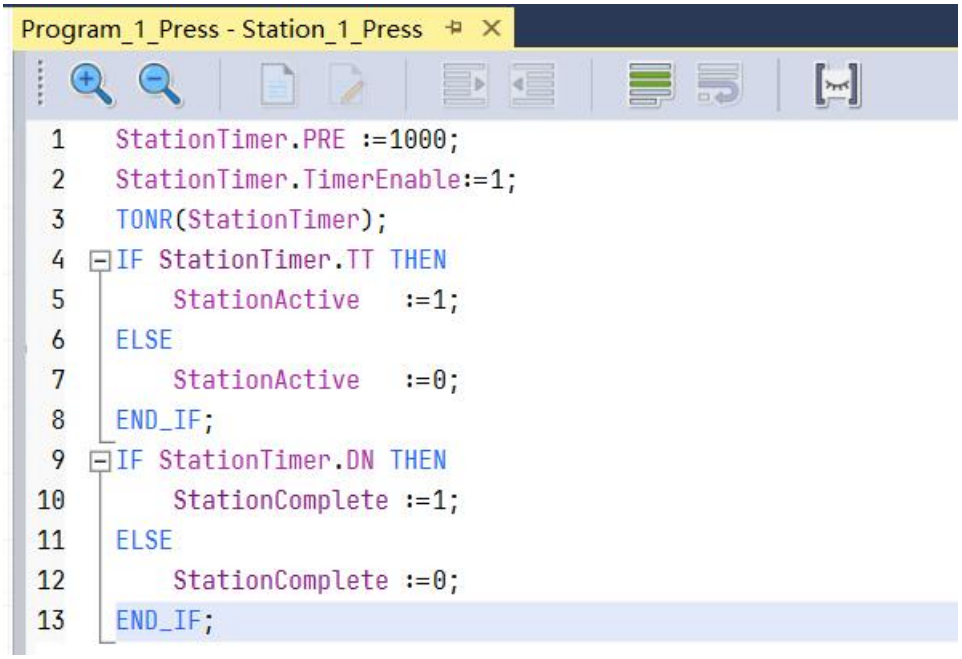


图 1-29 Station_1_Press 程序

28. 用户可以将 Station_1_Press 的程序直接复制到 Station_2_Stake 和 Station_3_Weld 例程中，需要做如下修改：

✓ StationTimer.PRE 的值分别改为 2000 和 3000

✓ Station_3_Weld 例程定时结束之后，增加 Complete 输出，表示三道工序都已完成，用于控制 Conveyor 输出。修改后的结果如图 1-30 所示。

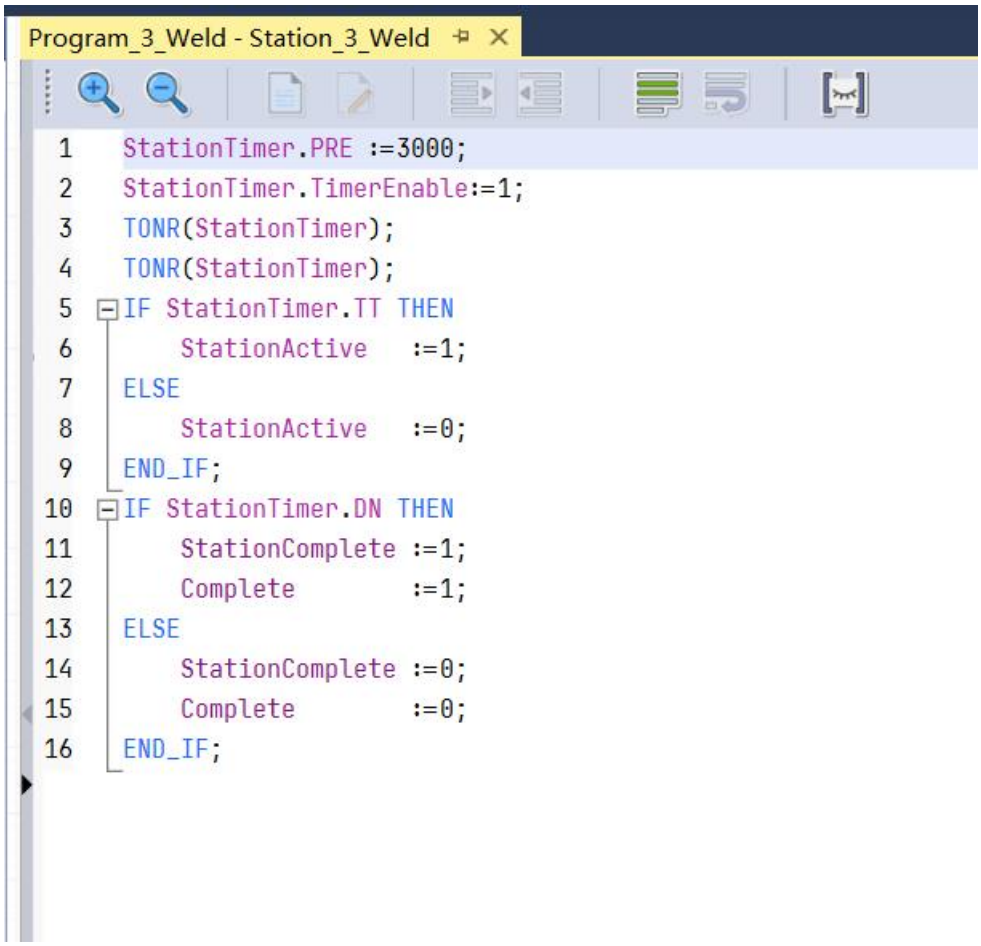


图 1-30 Station_3_Weld 程序

29. 单击图标校验每个例程，出现错误提示后，纠正错误。然后，单击图标校验整个项目并纠正出现的错误。

至此，三个工作站的程序已经完成。我们发现在创建过程中，实际上仅仅是 Program_1_Press 是从头创建的，其他两个程序都是对第一个程序的复制粘贴以及一些简单的修改。

30. 接下来编写 Conveyor 例程的逻辑程序，如图 1-31 所示。

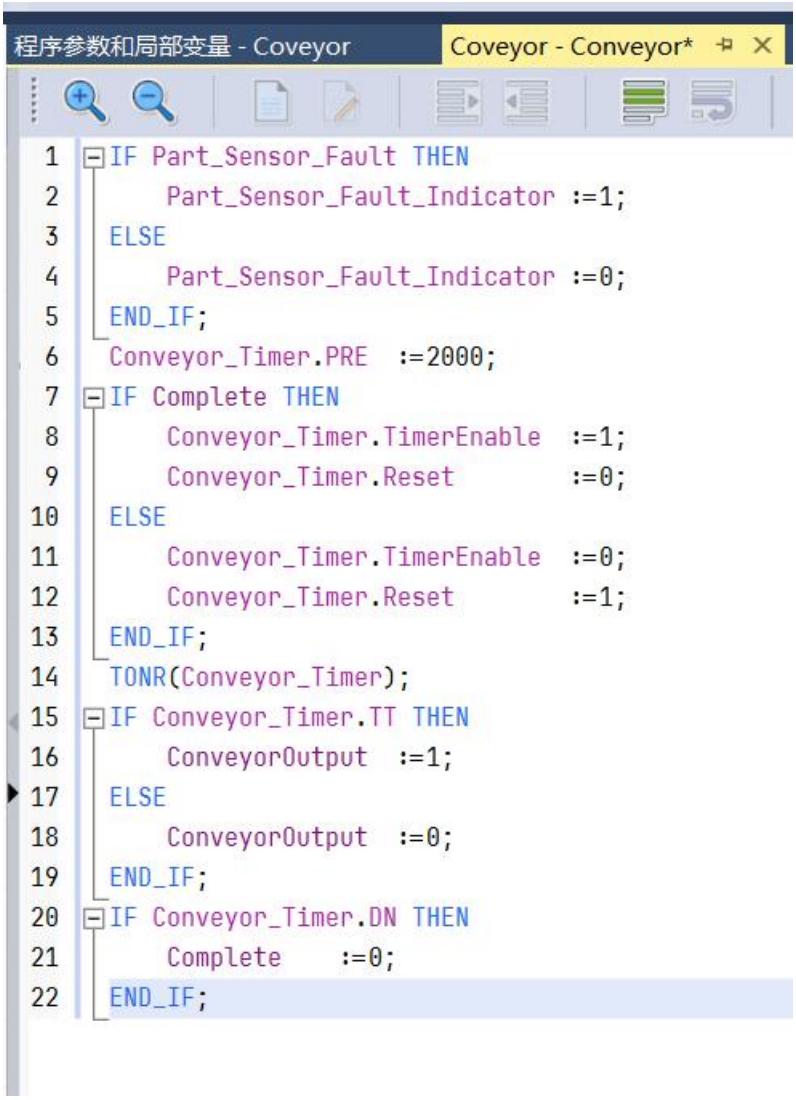


图 1-31 Conveyor 程序

其中第一段用于对光眼故障的报警，后两段用于控制输送带输出。

31. 继续编写工作站调度例程，如图 1-32 所示。

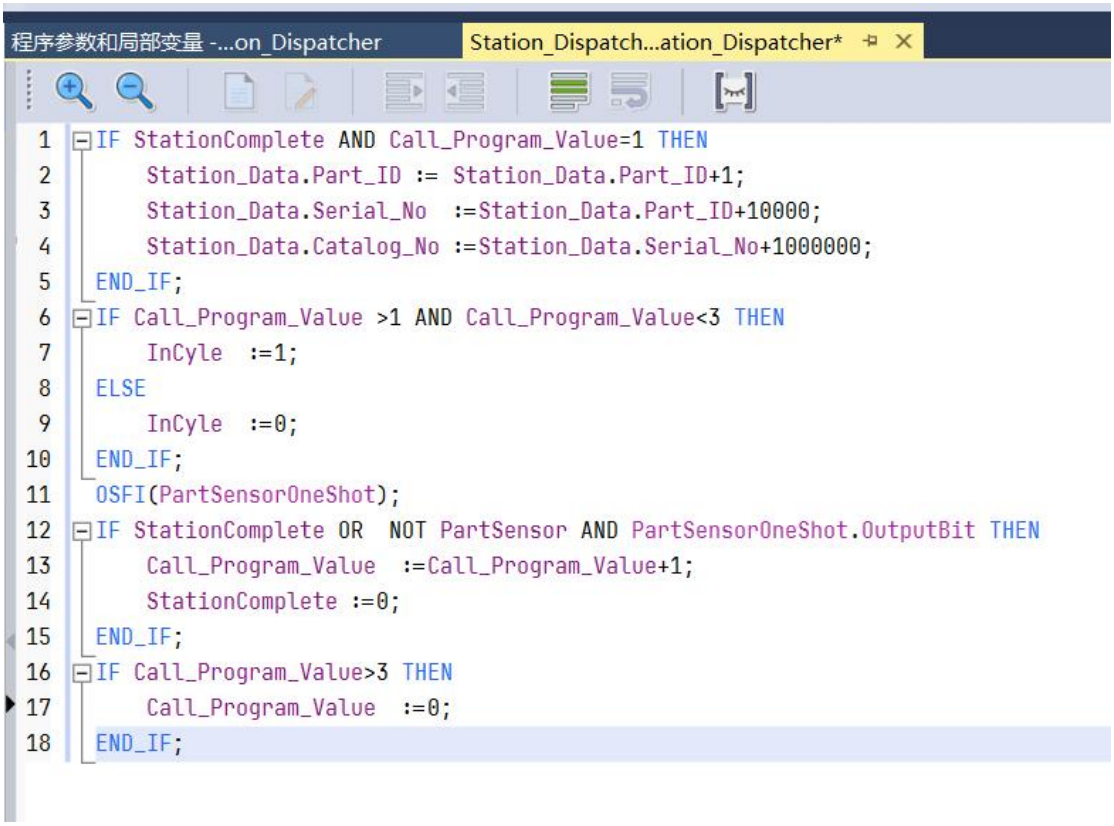


图 1-32　Station_Dispatch 程序

其中第一段用于生成压缩机产品编号，第二段判断三道工序是否正在工作，后两段用于调度工作站。

32. 单击  图标校验每个例程，出现错误提示后，纠正错误。然后，单击  图标校验整个项目并纠正出现的错误。

我们使用例程和项目校验工具时，只能查出程序中出现的语法错误，并不能查出程序中的逻辑错误。但现场条件往往并不允许直接连接 I/O 模块进行调试，通过趋势图，我们可以观察时序，进而分析程序逻辑关系是否正确。

1.2.5　趋势图

33. 右键单击控制器管理器中的*趋势图*文件夹，在弹出菜单内选择 *新建趋势图*，如图 1-33 所示。

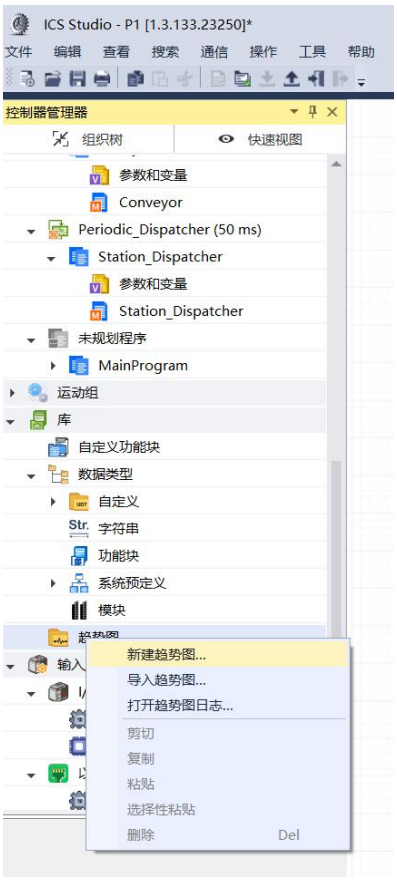


图 1-33 新建趋势图

34. 从弹出的对话框中命名新趋势图 Compressor，采样周期 10nm，单击下一步，如图 1-34 所示。

图 1-34 趋势图命名

35. 在弹出的 *添加/配置变量* 对话框中，范围选择 P1（控制器）或其他程序，从 *选择变量* 下拉菜单里选择需要添加的变量，单击 *添加* 按钮，如图 1-35 所示。

图 1-35 添加趋势图变量

最后按 *确定* 按钮。

36. 弹出趋势图显示画面如图 1-36 所示。

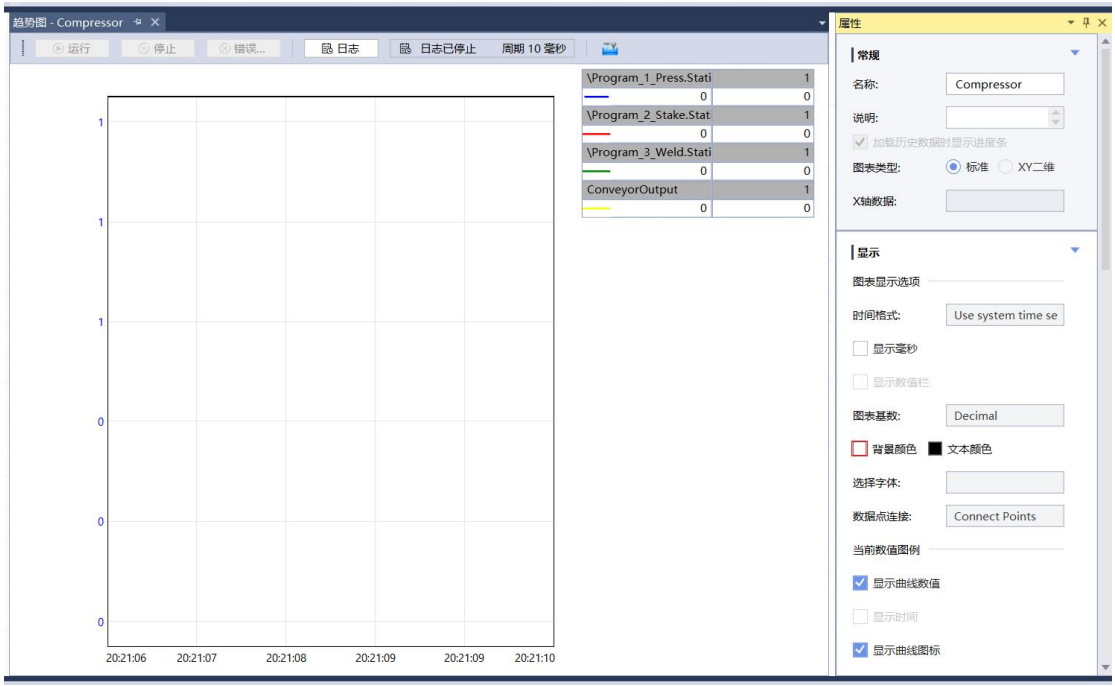


图 1-36 趋势图显示画面

画面的右侧是属性窗口，可对趋势图的各项属性进行配置，当配置完成之后，可将此窗口关闭，是趋势图监视画面更大。如需再次打开属性窗口，可以在菜单栏点击 *查看->属性* 打开。

在属性->显示部分，可以选择趋势图的背景颜色，默认是白色。

在趋势图属性->采样部分，可以设定 X 轴（时间轴）的相关参数，如图 1-37 所示。

图 1-37　趋势图采样属性

在趋势图属性->曲线数据源管理部分，可以添加或删除曲线，如图 1-38 所示。

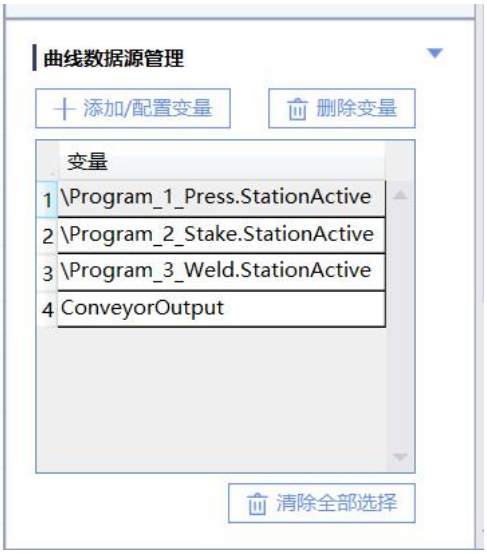


图 1-38　曲线数据源管理

在趋势图显示画面双击曲线监控栏的右半部分，可以弹出该曲线的竖轴属性设定，如图 1-39 所示。

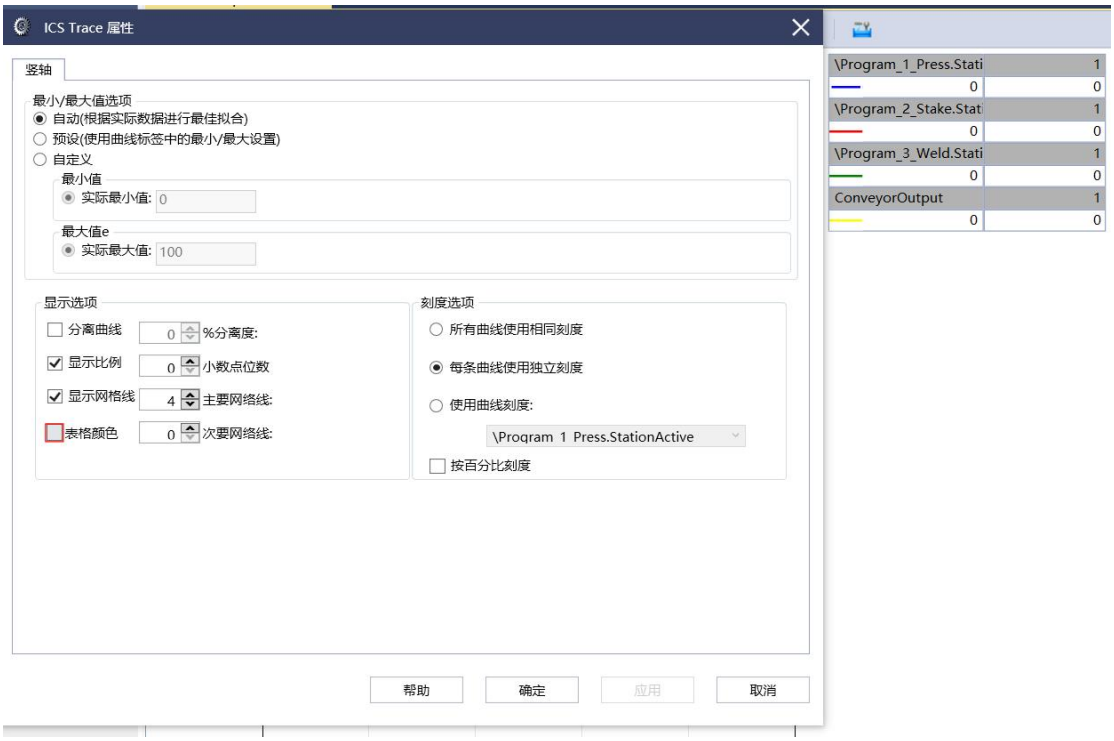


图 1-39　竖轴属性设定

在趋势图显示画面双击曲线监控栏左半部分，可设置曲线属性，如图 1-40 所示。

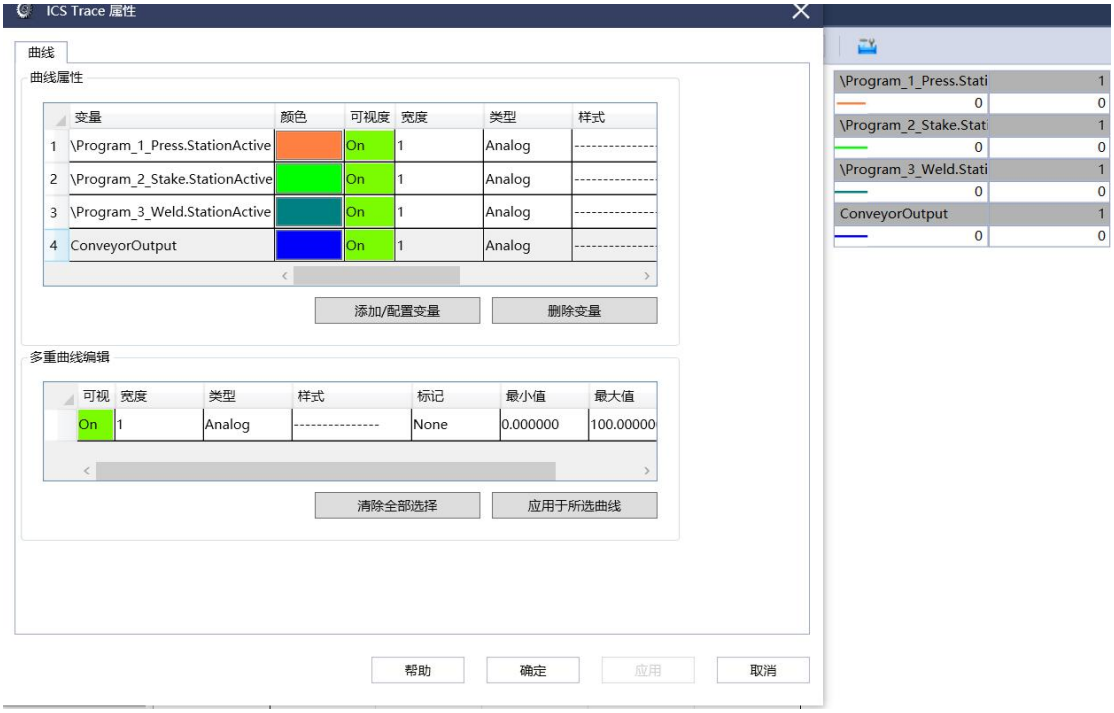


图 1-40　设置曲线属性

37. 接下来，要将该程序下载到控制器中运行，通过趋势图观察其运行结果是否正确。下载前单击菜单栏 *通信->网络路径* ，在弹出窗口确认电脑连接控制器的通讯路径，如图 1-41 所示。

图 1-41 通讯路径选择

勾选 *自动浏览* ，在电脑的网络连接下面找到 ICC 控制器（电脑网口 IP 地址需设置成跟控制器同网段）并选中，点击 *登入* 按钮或在图标栏按  图标，连接电脑和控制器。然后点击 *下载* 按钮或  图标，下载程序到控制器。

38. 鼠标移到编程软件界面右下侧，在弹出的 *控制面板* 窗口上单击 *运行* 按钮，使控制器处于运行状态，如图 1-42 所示。

图 1-42 控制面板

39. 在控制器管理器中双击打开趋势图 Compressor，在弹出的趋势图画面上单击 *运行*，开始实时绘制曲线。

40. 通过手动触发 PartSensor 标签，使模拟的生产线运行起来。双击 Station_Dispatcher（站调度）例程，打开程序窗口，单击窗口上方  图标，使其变成  ，程序内的变量下方将出现实时数值，在 PartSensor 下方双击将 0 改为 1，如图 1-43 所示。

41. 双击打开趋势图 Compressor，观察到时序图，如图 1-44 所示。

1.3 变量

1.3.1 变量概述

变量是控制器的一块内存区域，用来存储表示设备及其计算、故障等信息的数据。在 ICC 控制器中，数据的读取与存储是通过变量来实现的。与传统的可编程控制器不同，在控制器的内部直接采用基于变量的寻址方式。这样就不需要额外的标记名称与实际 I/O 物理地址对映的交叉参考列表。

在控制器中，处理器能够直接运用实名变量，例如使用“tank level”，“flow rate”等。这样，就不必使用交叉参考列表完成变量名称与物理地址之间的转换。唯一的地址就是变量名称，这样就使程序具有更高的可读性，即使没有说明性的文档也能够看懂。

综上所述，使用变量来存储和读取数据，同传统的解决方案相比，带来了如下诸多的优点：

- 变量实名功能，不仅缩短了初期的开发时间，还可以节省后期维护成本
- 避免了导入、导出和复制复杂数据库，对于英孚康的控制类产品，例如 ICD 系列的 I/O 模块、ICM 系列伺服驱动器等，可以自动创建变量。
- 避免了由于采用单一数据库出现故障后对整个系统造成重大损失
- 程序更容易阅读

变量可分为 Controller Variables（全局变量）和 Program Variables（局部变量）区别如下：

全局变量：例如创建 I/O 变量，工程中所有的任务和程序都可以使用；

局部变量：变量只有在与之相联的程序内才可以使用。

1.3.2 变量的操作

1. 创建变量

创建变量即为数据创建存储区。在 ICC 控制器中，数据分为 I/O 数据和中间变量数据。I/O 数据在组态 I/O 模块完成后自动生成，本节主要介绍如何创建中间变量数据。

创建变量有两种方式：在参数和变量表中创建；在程序中创建。

(1) 在参数和变量表中创建变量

新建工程之后，双击左侧 *全局变量->默认组* ，在打开的窗口中，底部选项卡选择 *编辑变量*，如图 1-45 所示。

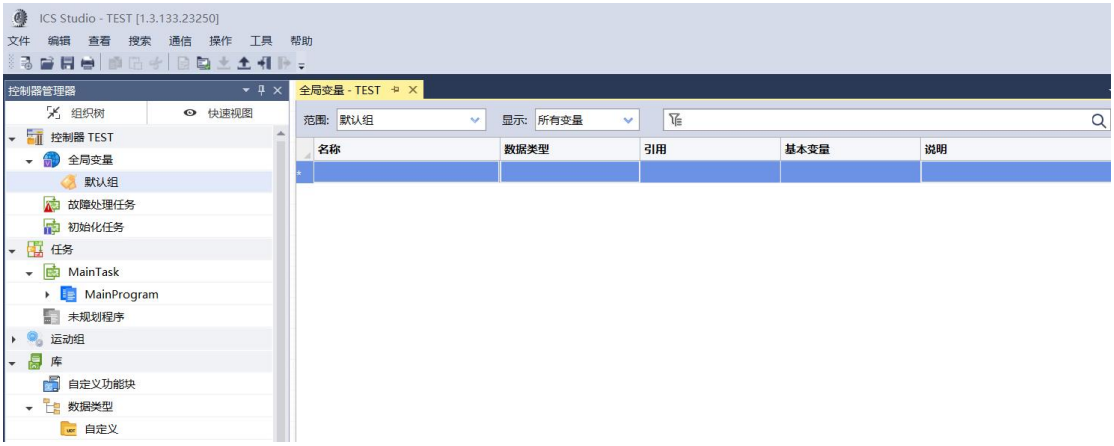


图 1-45　参数和变量表中编辑变量

在变量编辑区域，有 *名称，数据类型，引用，基本变量，说明，外部访问，常量，格式* 等列。在 *名称* 处输入变量名之后，自动出现默认的数据类型，点击数据类型后的按钮，可从弹出的菜单中选择所需要的数据类型，如图 1-46 所示。

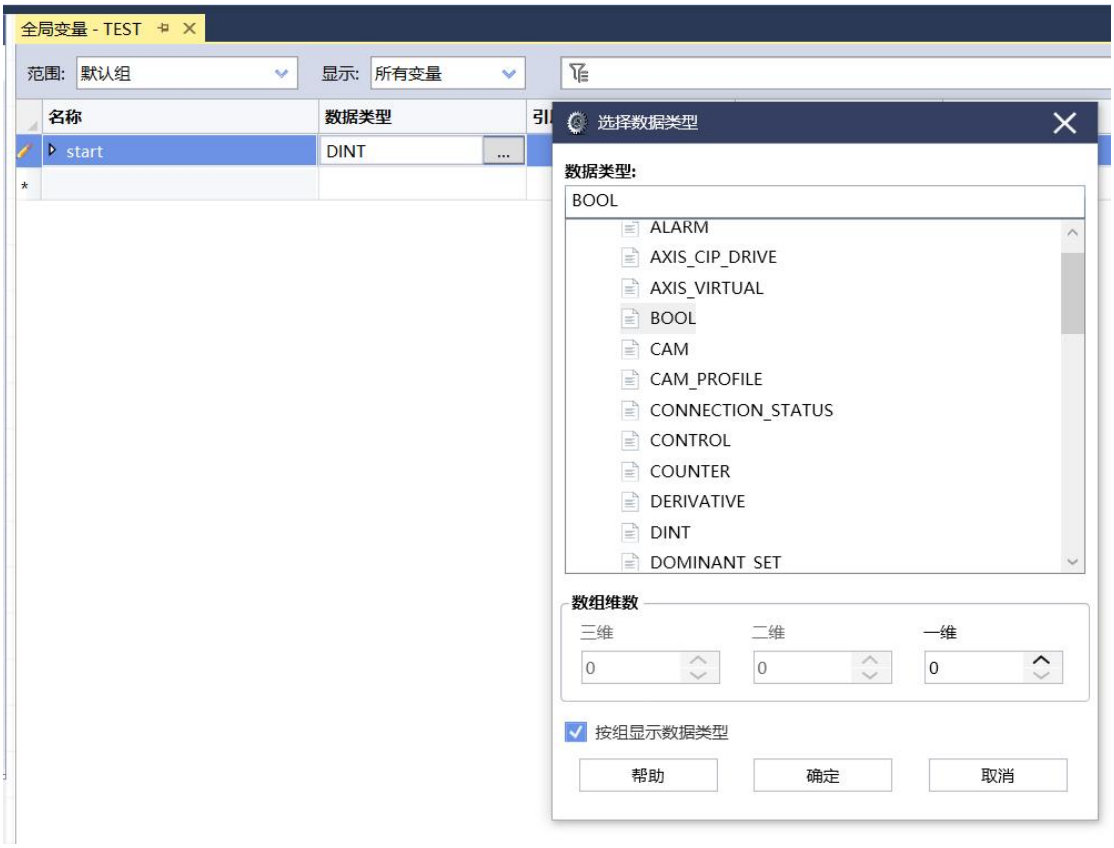


图 1-46 编辑变量数据类型

现在，将这个变量创建为 BOOL 类型，在输入框输入或在菜单选择 BOOL 即可，如果要建立数组，可在 *数组维数* 填入数组的个数即可。

点击 *说明* 下方的空白处即可输入注释信息，如图 1-47 所示。

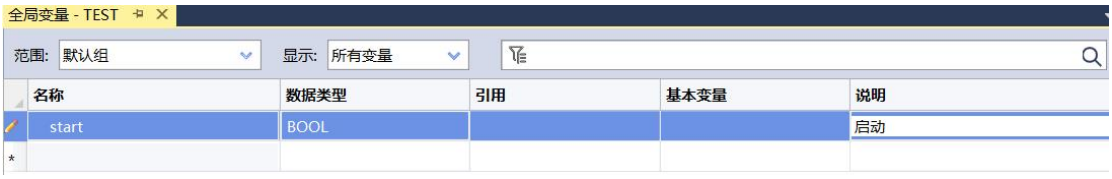


图 1-47　添加注释信息

这样就创建完毕一个布尔型的变量，在程序中直接使用即可。如图 1-48 所示。

图 1-48　变量添加完毕后的信息

(2)　在编程时创建变量

在程序中输入变量名称，然后右键单击变量名称，在弹出菜单中选择 *新建变量*，如图 1-49 所示。

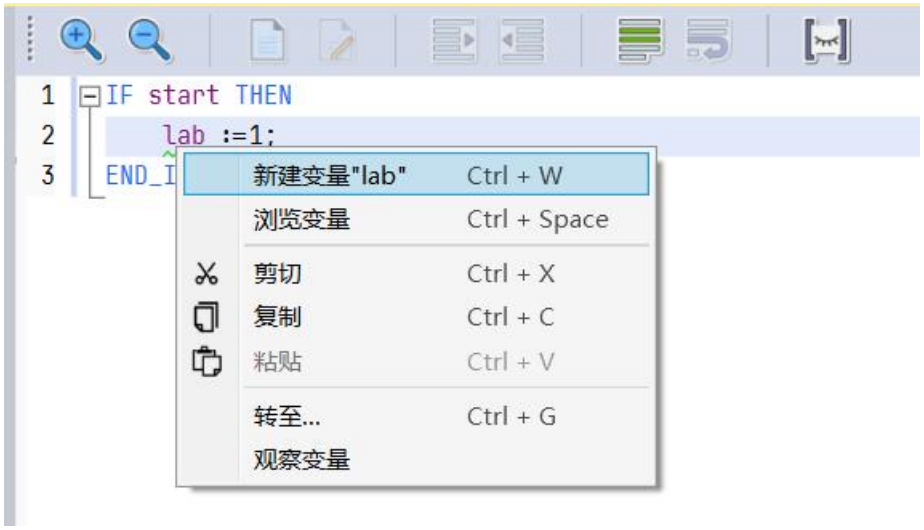


图 1-49　新建变量

弹出窗口如图 1-50，其中有 *名称，类型，范围，外部访问，引用，说明*和是否 *常数* 的勾选等。

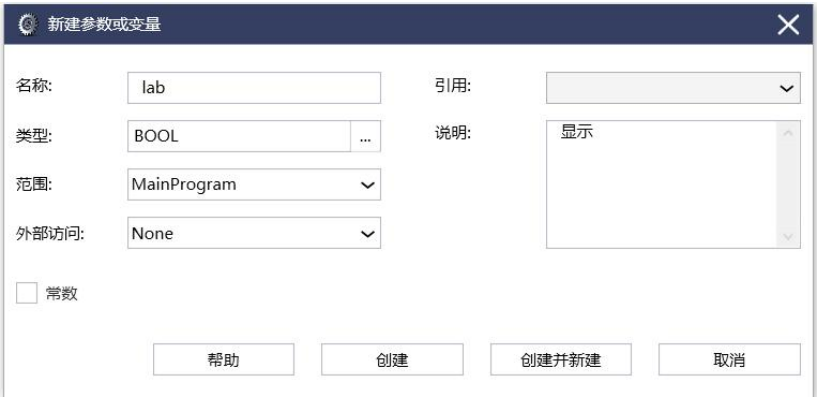


图 1-50　编辑变量

在 *说明* 处写入“显示”，其他如图 1-50 设置，点击 *创建* 之后，程序中的标签如图 1-51 所示。

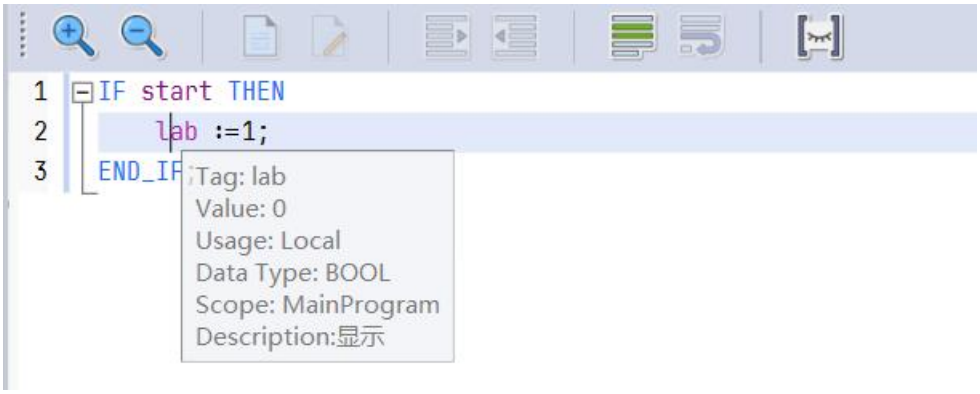


图 1-51　编辑完毕后的变量

2.　变量的查找及交叉引用

在进行工程的开发和调试时，经常会查找已经使用过的相同名称的变量。可以通过搜索的方法，打开某个工程，在程序中选中待查找的变量，同一例程中出现该变量的地方都会变成黄色底色，如图 1-52 所示。

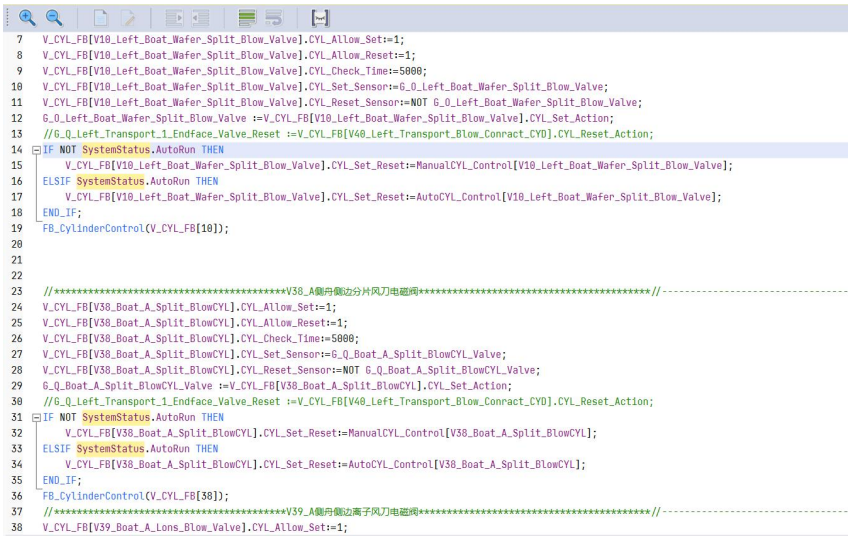


图 1-52 同一例程中相同名称变量搜索

另一种方法，在变量上单击右键，选择 *转至标签名称的交叉引用*， 如图 1-53 所示。

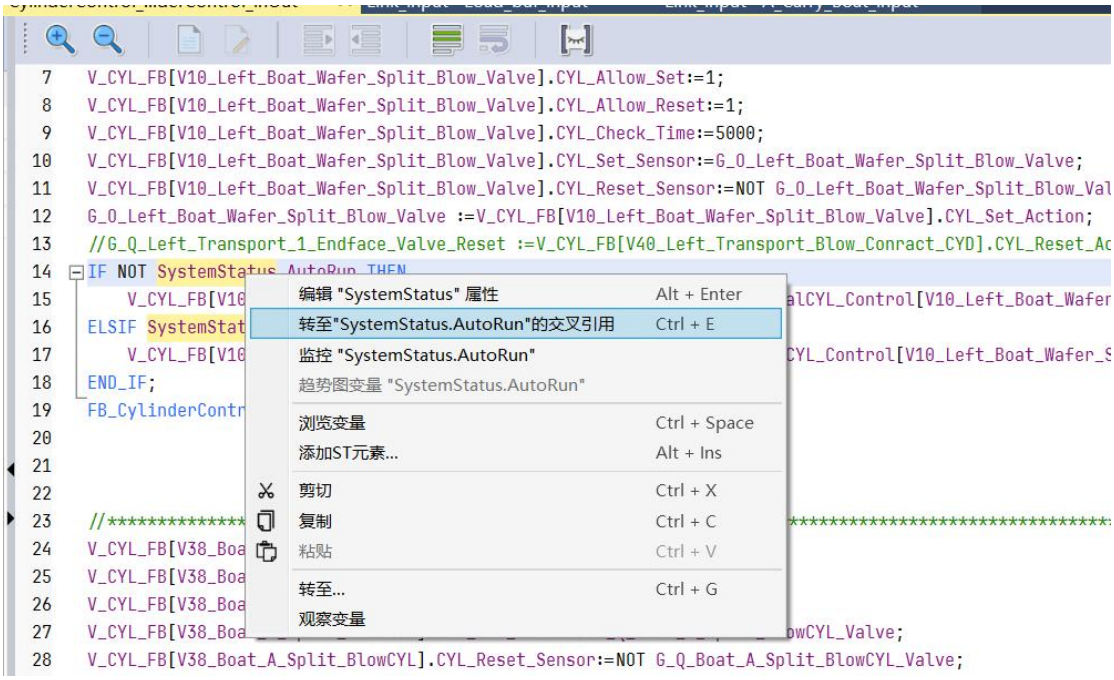


图 1-53 选择交叉引用功能

点击即可启动此功能，弹出窗口如图 1-54 所示。

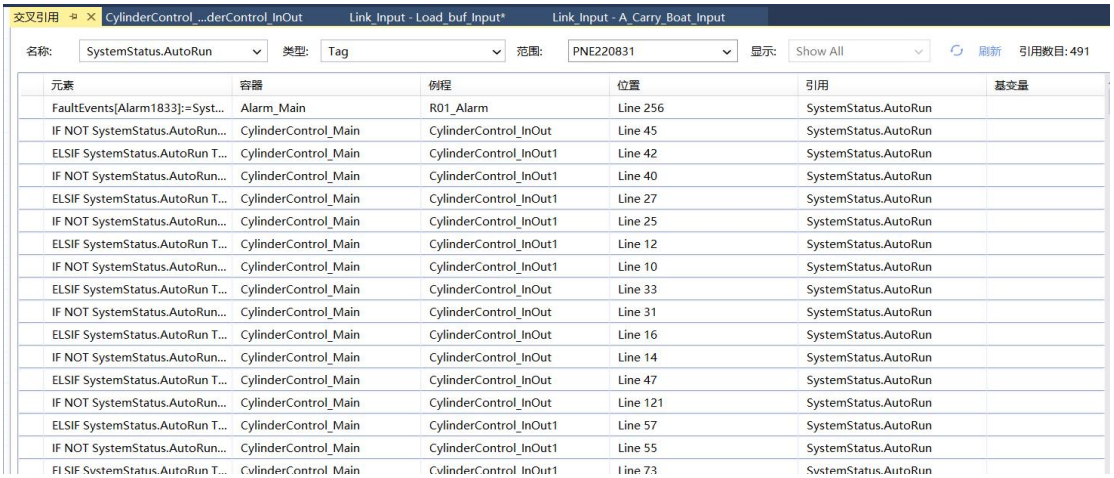


图 1-54 交叉引用列表

选择其中一行双击即可跳到对应的例程语句。

3. 变量监视

在线状态下，可进行变量的监视。具体操作如下，在待监视的变量上单击右键，选择 *监控变量名*，如图 1-55 所示。

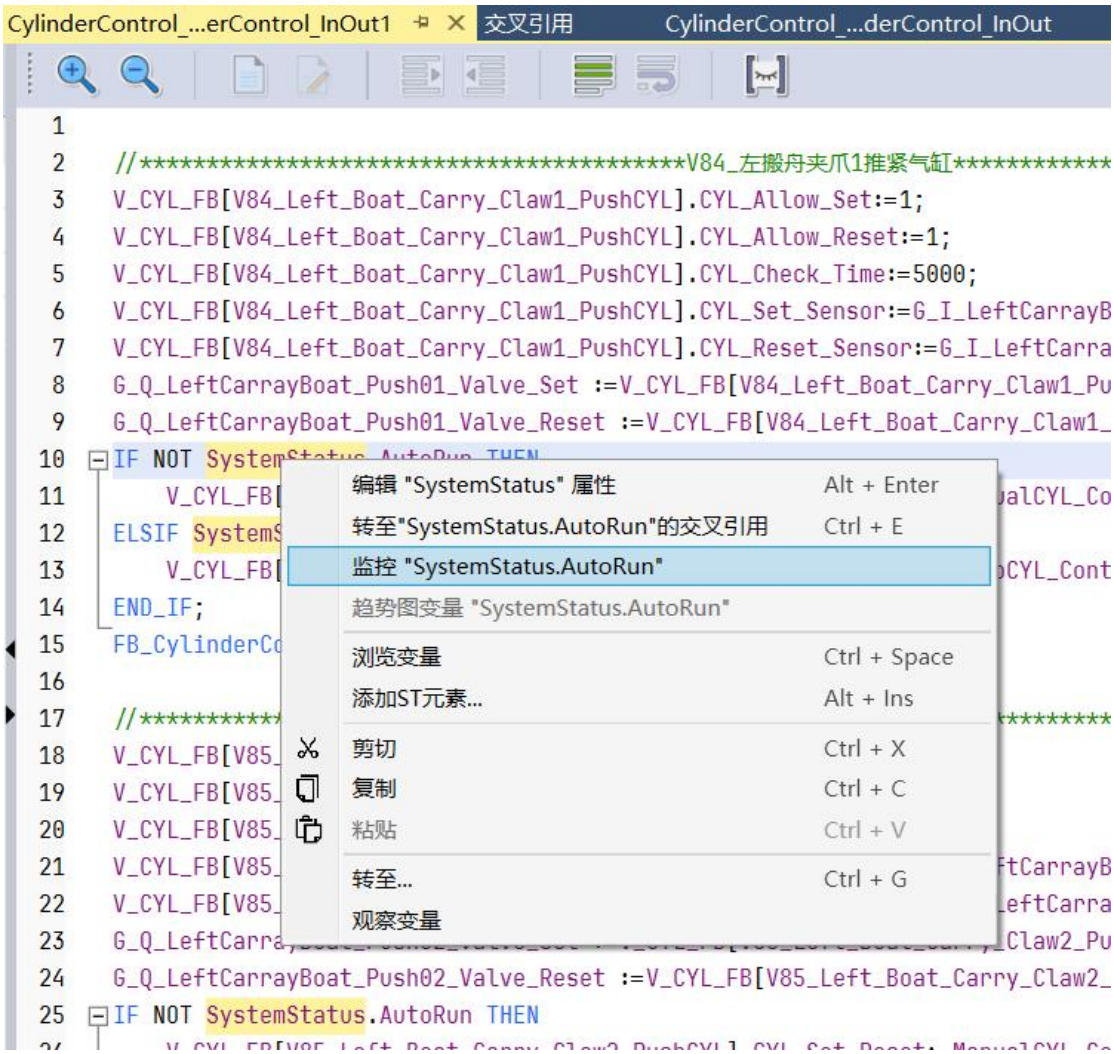
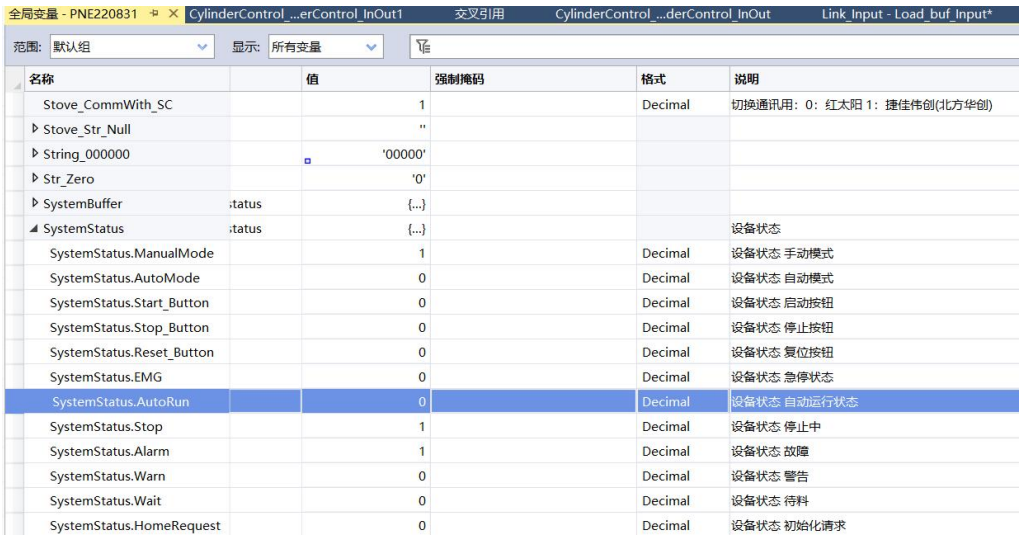


图 1-55　选择监控变量功能

点击即可启动此功能，窗口跳转到此变量所在的全局或局部变量列表，并显示实时值，如图 1-56 所示。



名称	值	强制掩码	格式	说明
Stove_CommWith_SC	1		Decimal	切换通讯用：0：红太阳 1：捷佳伟创(北方华创)
Stove_Str_Null	''			
String_000000	'00000'			
Str_Zero	'0'			
SystemBuffer	{...}			
SystemStatus	{...}			设备状态
SystemStatus.ManualMode	1		Decimal	设备状态 手动模式
SystemStatus.AutoMode	0		Decimal	设备状态 自动模式
SystemStatus.Start_Button	0		Decimal	设备状态 启动按钮
SystemStatus.Stop_Button	0		Decimal	设备状态 停止按钮
SystemStatus.Reset_Button	0		Decimal	设备状态 复位按钮
SystemStatus.EMG	0		Decimal	设备状态 急停状态
SystemStatus.AutoRun	0		Decimal	设备状态 自动运行状态
SystemStatus.Stop	1		Decimal	设备状态 停止中
SystemStatus.Alarm	1		Decimal	设备状态 故障
SystemStatus.Warn	0		Decimal	设备状态 警告
SystemStatus.Wait	0		Decimal	设备状态 待料
SystemStatus.HomeRequest	0		Decimal	设备状态 初始化请求

图 1-56　变量监视区域

也可以直接在变量所在的全局或局部变量列表中，选择下方 *监控变量* 选项卡监视变量实时值，如图 1-57 所示。

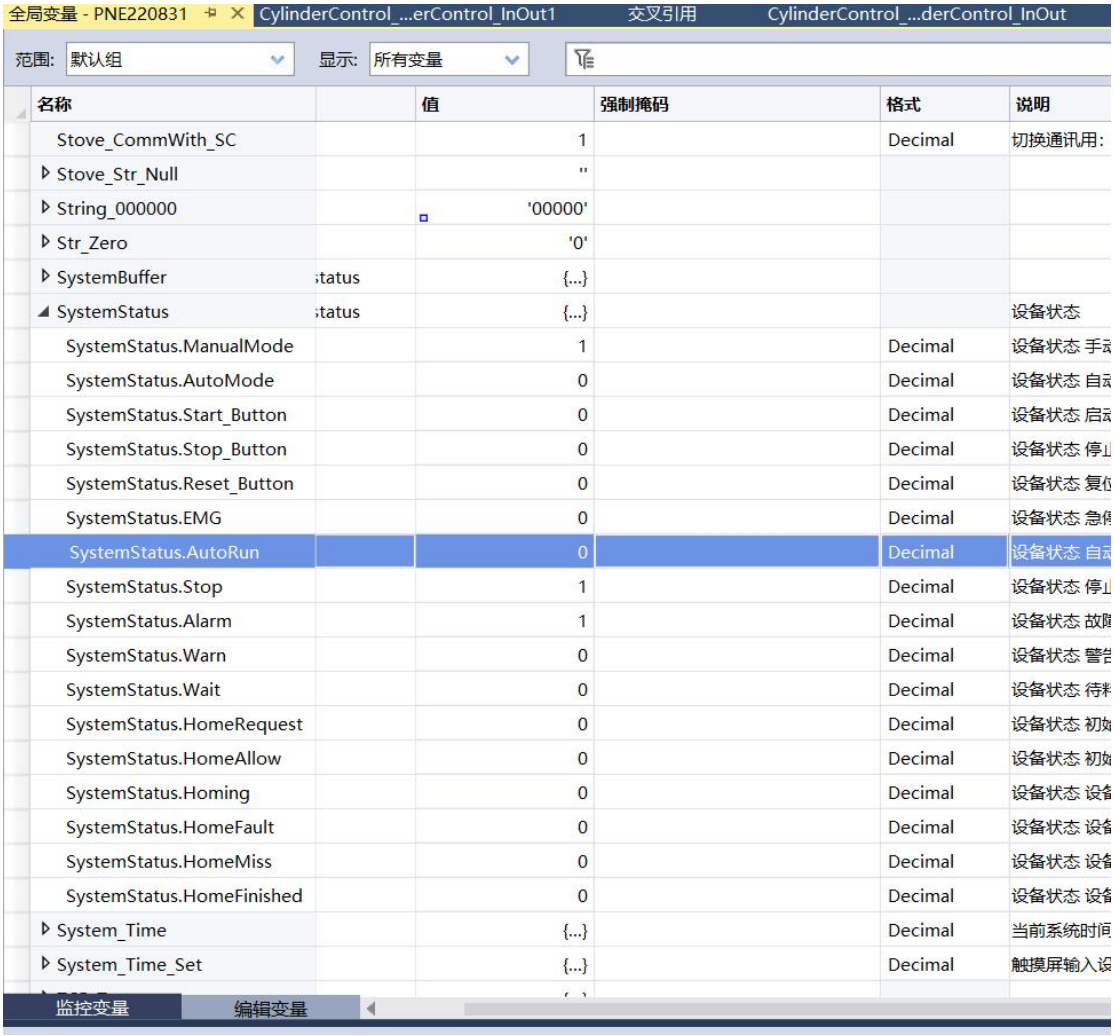


图 1-57　变量实时监控

1.3.3　数据结构

数据类型是用来定义变量使用的数据位、字节或字的个数。数据类型的选择是根据数据源而定的，常见的数据类型由下表所示。

表 1-4　常见的数据类型

数据类型	定义
BOOL（布尔型）	为单个数据位。这里 1=接通；0=断开（可以用来表示离散量装置的状态。例如按钮和传感器的状态）。
SINT（单整型）	短整型（8 位），范围是-128 到+127。
INT（整型）	一个整型数或者字（16 位）范围是-32,768 至+32,767。
DINT（双整型）	双整型（32 位 ），用来存储基本的整型数据，范围是-2,147,483,648 至+2,147,483,647
REAL（实数型）	32 位浮点型（例如用来表示模拟量数据，如电位计的数据）。
STRING（字符串型）	用来保存字符型数据的数据类型（例如存储“car”和“this is text”）。

数据类型之所以重要，是因为它涉及到数据在控制器中的内存分配为题。

任何数据的最小内存分配的数据类型为 DINT 型（双整型或者 32 位）。DINT 型为 ICS Studio 的主要数据类型。当用户分配了数据后，控制器自动为任何数据类型分配下一个可用的 DINT 内存空间。

当给变量分配数据类型（如 BOOL，SINT, INT 时），控制器仍占用一个 DINT 型的空间，但实际只占用部分空间，如图 1-58 所示。

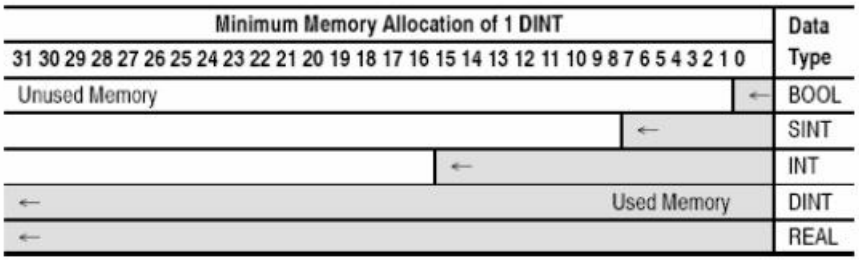


图 1-58　最小内存分配示意图

由于上述原因，推荐用户在创建变量时尽可能创建 DINT 类型的变量，并且最好创建数组（这主要考虑到和上位机的通讯）。

1.3.4　数组与结构体

1. 数组

ICC 控制器允许用户使用数组数据。

数组是包含一组多个数据的变量，它有以下的特征：

- 在数组中，每一个数据称为一个元素
- 每个元素使用相同的数据类型
- 数组变量占据控制器中的一个连续内在块，每个元素顺序排列
- 用户可以使用高级指令（文件指令等）操作数组中的元素
- 数组有一维、二维、三维三个种类

数组中的每个元素都有下标标识，下标从 0 开始，至元素数目减 1 的位置结束。如图 1-59 是通常的数组标签：

图 1-59　数组变量示意图

创建数组的过程比较简单，在创建变量时，在选择数据类型时点击旁边的按钮，会弹出如图 1-60 所示对话框。

图 1-60　创建数组维数

另外，需要特别说明的是使用数组数据类型不但可以节省内存，加速通讯速度，还有专门的用于处理数组的指令，可大大方便编程，缩短工程开发时间。

2.　用户自定义结构体

用户自定义结构体可以使用户根据控制对象创建适合其应用的结构体数据类型，可大大方便编程和进行设备维护。

下面以创建电机控制的自定义结构体为例：

在控制器管理器的 *库->数据类型->自定义* 右键单击，选择 *新建数据类型* ，如图 1-61 所示。

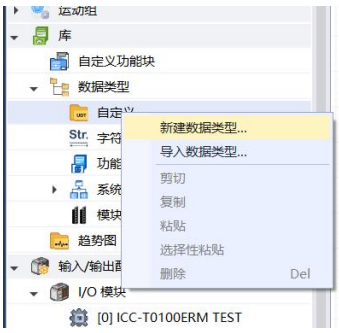


图 1-61　创建新的数据类型

在打开的窗口中输入自定义结构体的名称，说明信息以及各个成员的名称，如图 1-62 所示。

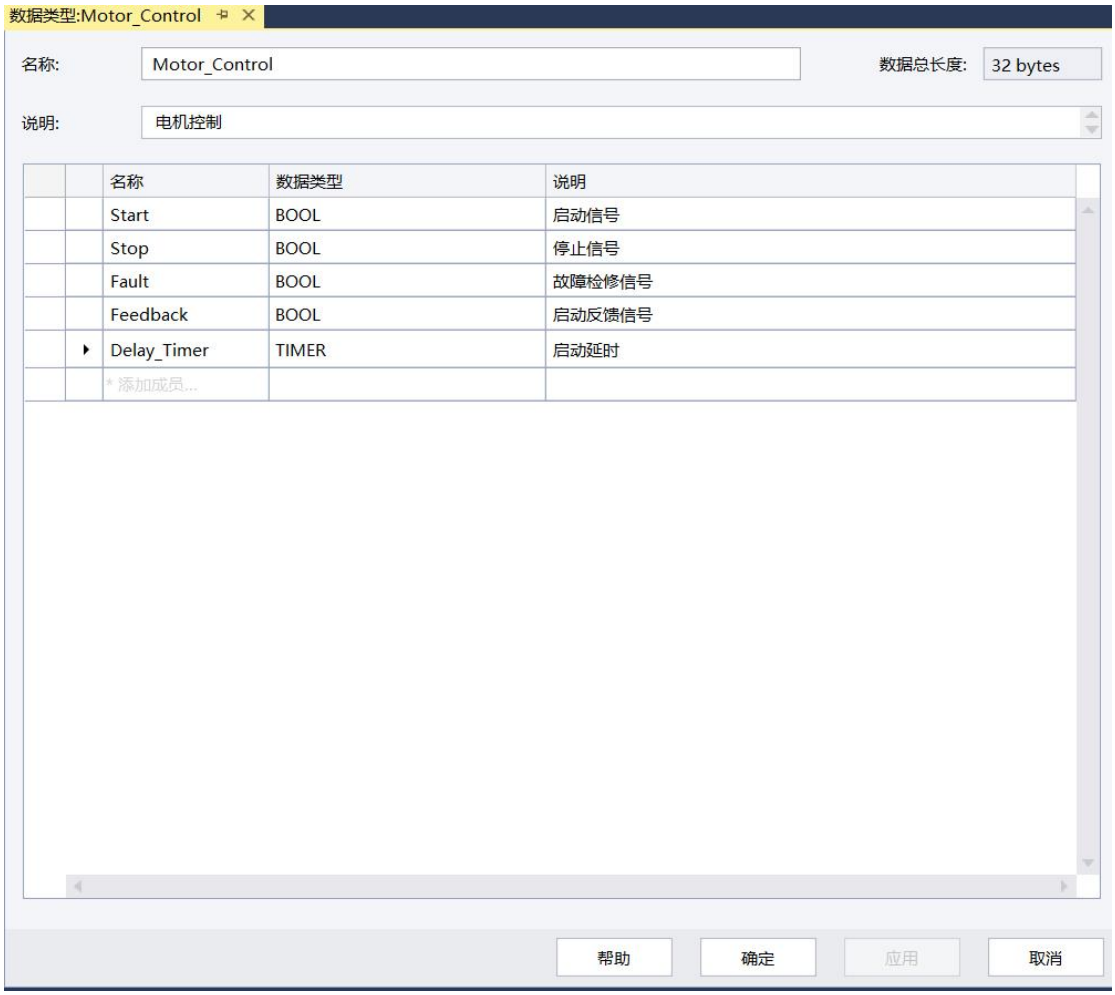


图 1-62　输入用户自定义结构体名称

安排自定义结构体各成员时，将相同数据类型的成员放在一起，可以解决内存。当添加完所有成员后，点击 *应用* 按钮，在窗口右上角显示该自定义结构体所占内存的长度。

用好数组和结构体可以使编程更加巧妙和简单。

1.4 任务、程序与例程

ICC 控制系统中的程序是通过任务（Task）来执行的，每个工程最多支持 16 个任务（Task），每个任务（Task）中又包含程序（Program），程序（Program）包含例程（Routine），在例程（Routine）中写入程序。ICS Studio 的编程符合 IEC-61131 标准，支持结构化文本、梯形图等编程语言。

1.4.1　系统任务

ICC 控制器支持两种类型的任务，它们分别是连续型（Continuous）任务和周期型（Periodic）任务。它们的说明如下表 1-5 所示：

表 1-5　ICC 支持的任务

如果用户需要以下列方式执行程序	使用任务类型	说明
在全部的时间内都执行	连续型任务	连续型任务在后台运行。任何不分配给其它操作（其它的操作指：运动、通讯以及周期型任务或事件型任务）的 CPU 时间，用于执行连续型任务中的程序。 连续型任务始终运行。当连续型任务完成一次全扫描之后，它会立刻重新开始进行扫描。 一个工程有且必须只有一个连续型任务。
以一个固定的周期（例如：每 100ms）执行。 在扫描其它逻辑程序时多次运行某一程序	周期型任务	周期型任务按照指定的周期来执行。 只要到达周期型任务指定的时刻，该种类型的任务就会自动中断所有低优先级的任务。执行一次，然后将控制权交回先前正在执行的任务。 周期型任务的执行周期缺省值为 10ms，用户可以选择的范围是 0.1ms 至 2000ms。

1. 连续型任务

控制器一直执行的任务是连续型任务。每个连续型（周期型任务也一样）的任务中可以建立多个程序，每个程序下也可以建有多个例程。这样就极大地方便了用户在编程时可以按照工艺或者功能的不同划分不同的任务、程序和例程。下面讲述如何在连续型任务中建立程序和例程。（周期型任务同理）。

创建工程完毕后，在 *任务* 处会有一个 MainTask 的任务和 MainProgram 的程序以及 MainRoutine 的例程，这是系统自动生成的，如图 1-63 所示。

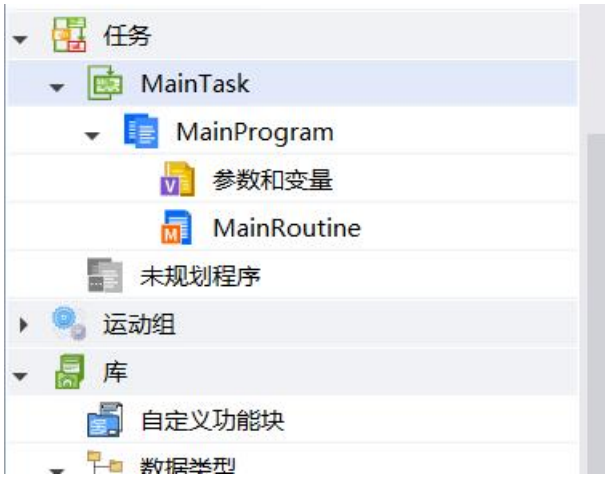


图 1-63　主任务和主程序以及主例程

并且 MainTask 是连续型任务，如要修改任务名称，在 MainTask 上单击右键，选择 *属性*，如图 1-64 所示。

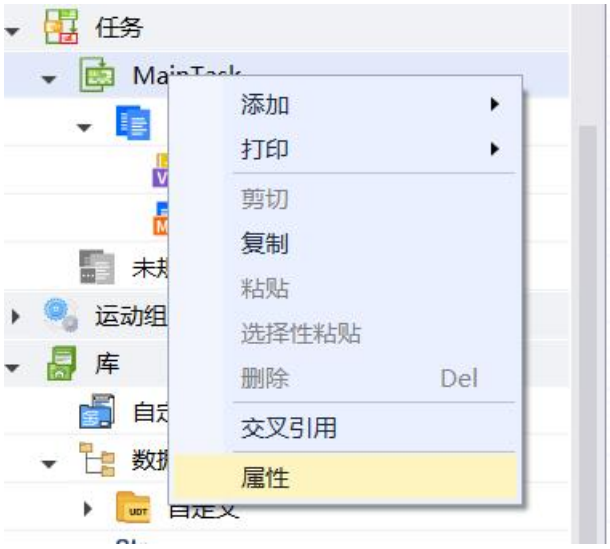


图 1-64 选择任务属性

点击 *属性* 选项即出现任务属性窗口，如图 1-65 所示。

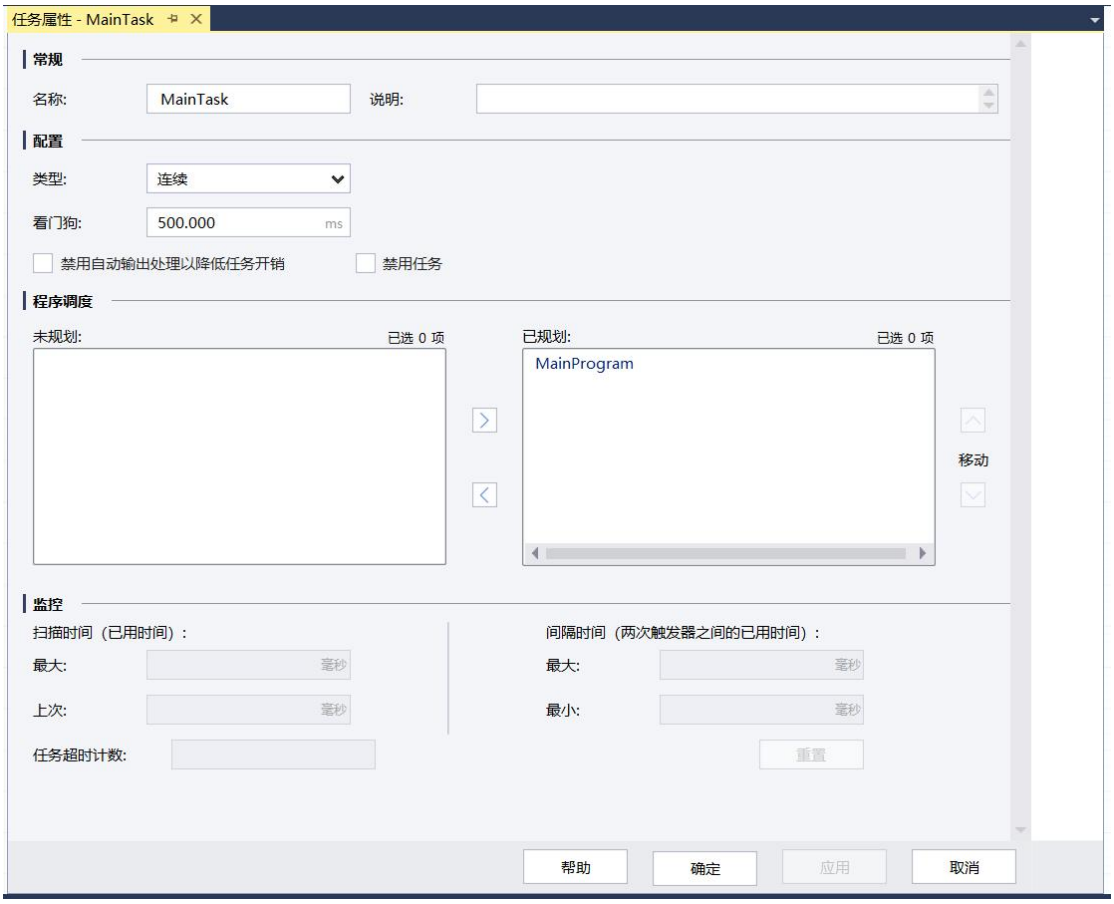


图 1-65 任务属性窗口

在 *常规* 栏可修改任务名称以及添加说明。

在 *配置* 栏可选择任务类型为连续或周期；设定看门狗时间，以及可以进行禁止输出和禁用任务的操作。

2. 周期型任务

周期型任务有以下特征：

- 它是由用户指定时间间隔来执行的任务
- 它会一直中断连续型任务
- 可以中断其他优先级低的周期型任务
- 在执行周期型任务时，控制器会从头到尾扫描所有程序
- 在一次扫描完毕后，更新输出，控制器继续从先前的断点处开始继续执行

下面将示例创建一个周期型任务：

在 *任务* 文件夹处右键单击，选择 *新建任务* ，如图 1-66 所示。

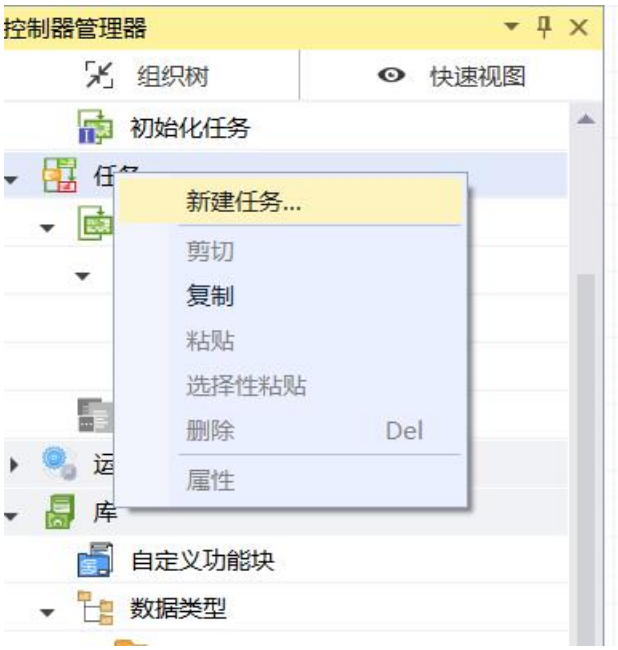


图 1-66　新建任务

在弹出的对话框内输入任务名称，设定任务周期以及优先级，如图 1-67 所示。

图 1-67　设置任务

设置完毕后，程序会添加一个周期型任务，然后新建程序。同样，在周期型任务处右键单击，*添加->新建程序*，如图 1-68 所示。

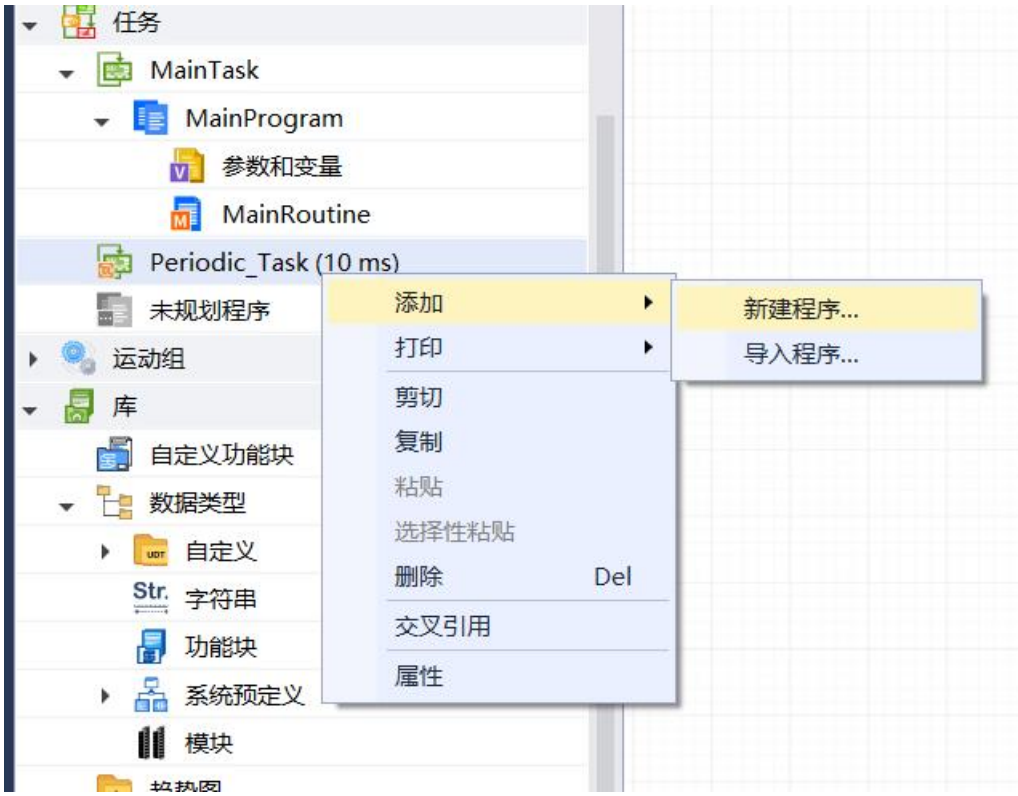


图 1-68　新建程序

在弹出对话框内配置程序，如图 1-69 所示。

图 1-69　配置程序

配置完毕后，程序的下方会自动生成 *参数和变量* 文件夹，如图 1-70 所示。

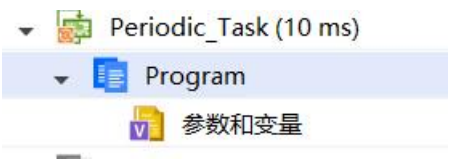


图 1-70　自动生成的程序变量

下面要建立例程，也即是编制程序的地方。在程序名处右键单击，选择 *添加->新建例程*， 如图 1-71 所示。

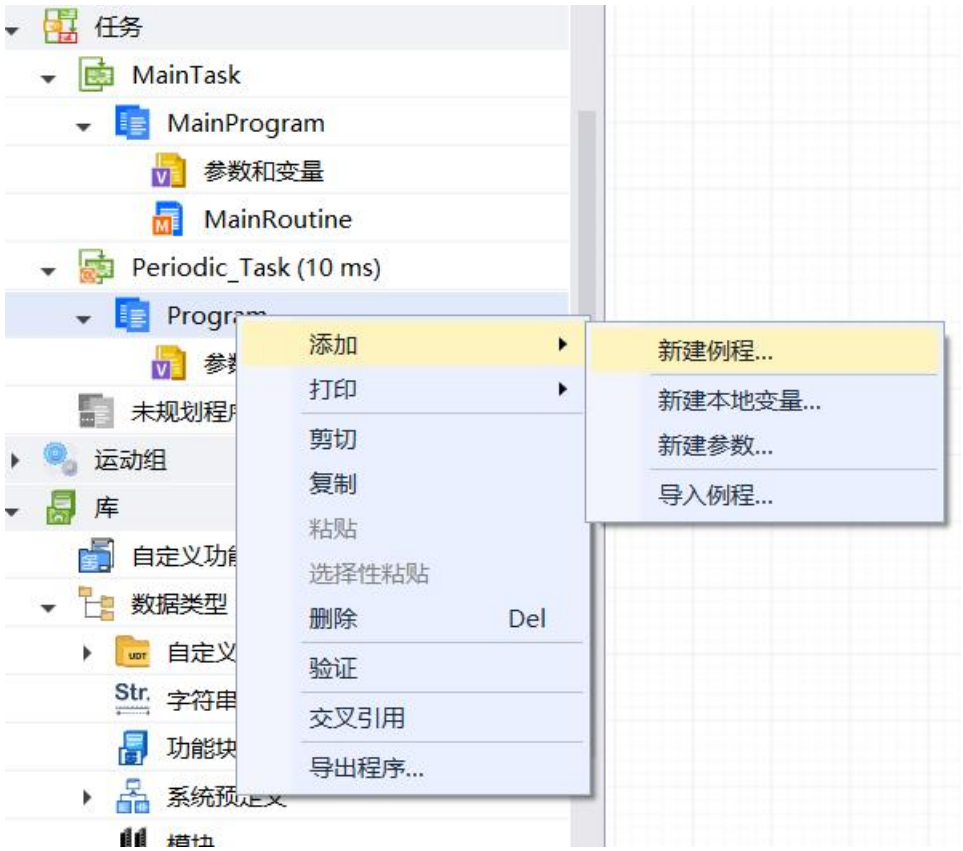


图 1-71　新建例程

在弹出的对话框内配置例程，输入例程名称及说明信息，如图 1-72 所示。

图 1-72　配置例程

配置例程时可以选择例程使用的编程语言是结构化文本或梯形图，在 *类型* 处选择。

需要指出的是，一个程序有多个例程，这就需要指定其中某个例程为主例程，其他例程都通过主例程调用。设置主例程的过程为：选择程序名，在画面右侧打开属性窗口（菜单栏 *查看->属性*），在属性窗口配置栏 *主要* 处选择主例程，如图 1-73 所示。

图 1-73　主例程设定

1.4.2　任务的注意事项

1.　关于优先级

优先级的设置是在任务的属性设置中，周期型任务的优先级如图 1-74 所示。（注意连续型任务是不需要设置优先级的）。

图 1-74　周期型任务优先级设置

优先级中设置的数字越小，该任务的优先级越高。控制器执行各任务的优先级如图 1-75 所示。

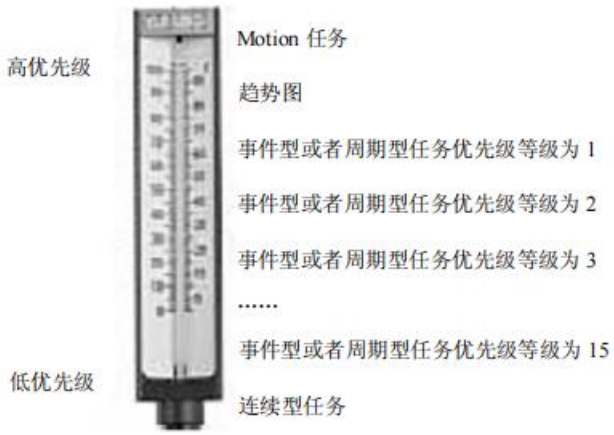


图 1-75　任务的优先级示意图

2.　关于扫描时间

在程序执行期间，ICS Studio 软件显示执行任务所用的最大扫描时间和上次的扫描时间，这一功能在任务属性画面的 *监控* 栏以毫秒级别显示。注意，上次扫描时间是实时变化的，如图 1-76 所示。

图 1-76　程序扫描时间

周期型任务是在指定的时间间隔内触发的。例如，某周期型任务每隔 20ms 触发一次，执行该任务需要。它的执行顺序如图 1-77 所示。

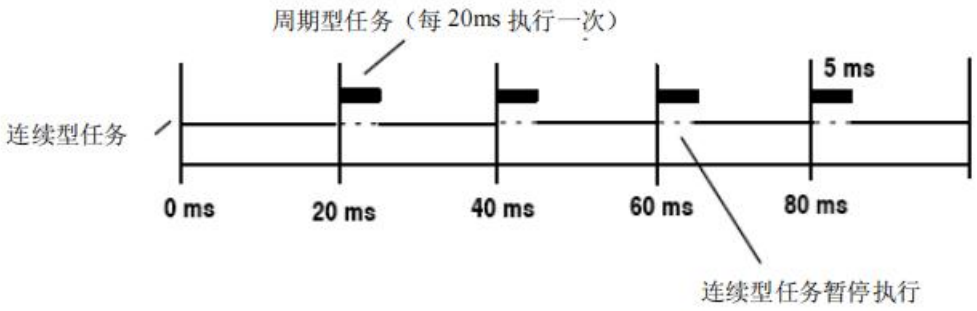


图 1-77　周期型任务的执行示意图

3.　关于周期型任务的优先级

如果有两个周期型任务 A 和任务 B，如果任务 A 每 20ms 触发一次，并且其优先级为 3；任务 B 每 22ms 触发一次，其优先级为 1，它们的执行情况如图 1-78 所示。

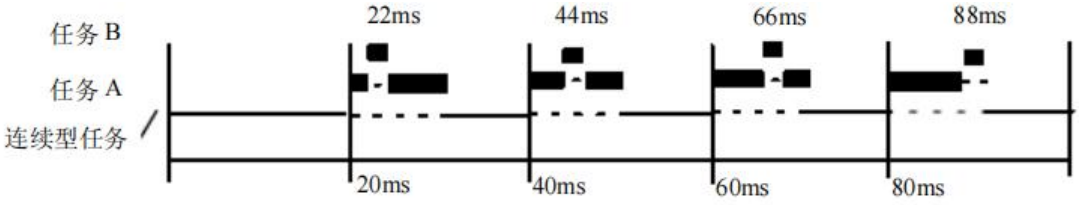


图 1-78　多个周期型任务的执行示意图

1.5 I/O 组态

在本节中，将以组态 ICC-BOB16 模块为例介绍如何通过 ICS Studio 软件组态本地 I/O 模块，16 点数字量输出模块 ICC-BOB16 处于第一槽位。

(1)　在 *输入/输出配置->I/O 模块* 的第 0 槽控制器处右键单击，选择 *属性* ，如图 1-79 所示。

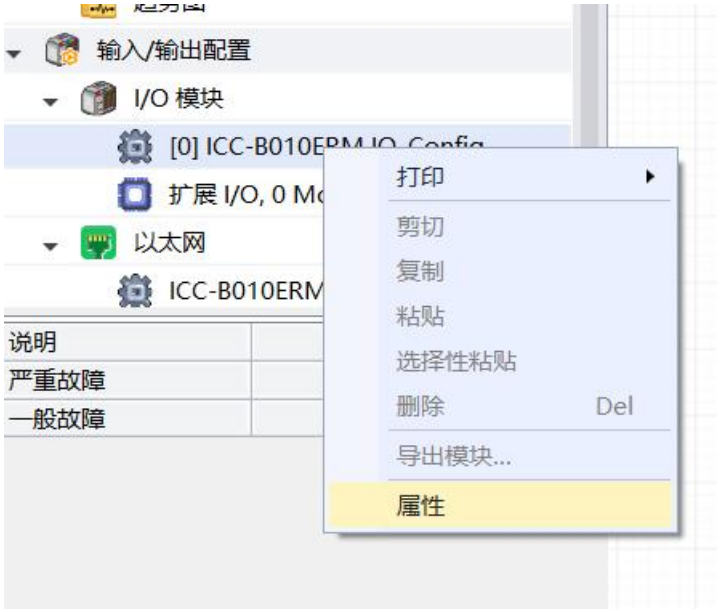


图 1-79　控制器属性

在弹出的属性窗口选择 *常规* 选项卡，在 扩展 I/O 处可下拉选择需扩展的 I/O 模块数

量，在此我们选择一个模块，如图 1-80 所示。

图 1-80　确定扩展 I/O 模块数量

(2) 按 *应用*，并*确定*，关闭控制器属性窗口。在 输入/输出配置->I/O 模块->扩展 I/O 右键单击，选择 新建模块， 如图 1-81 所示。

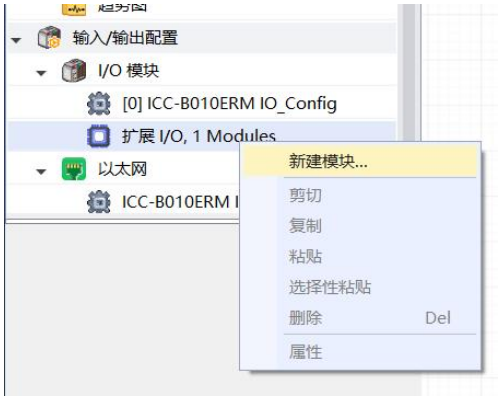


图 1-81　新建模块

在弹出的 *选择模块类型* 窗口中，通过勾选*模块供应商过滤器*和*模块类型过滤器*，能够缩小模块查找的范围。在模块列表中找到需要添加的模块 ICC-BOB16，并点击 *创建*，如图 1-82 所示。

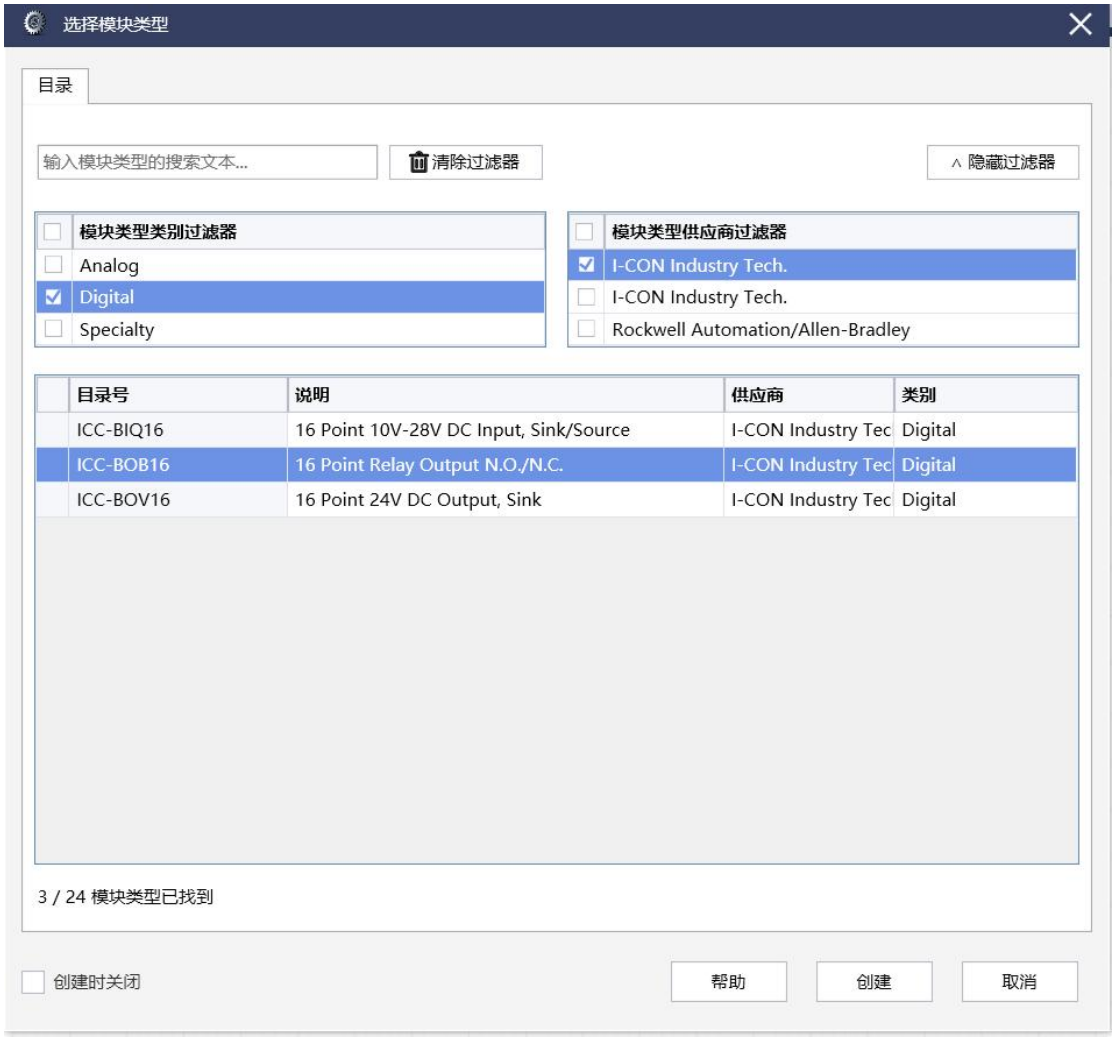


图 1-82　选择模块类型

(3) 在弹出的 新建模块 窗口 常规 页面，输入模块的名称和选择所在槽位，模块定义部分定义了该模块的系列、固件以及电子锁匹配模式，如图 1-83 所示。

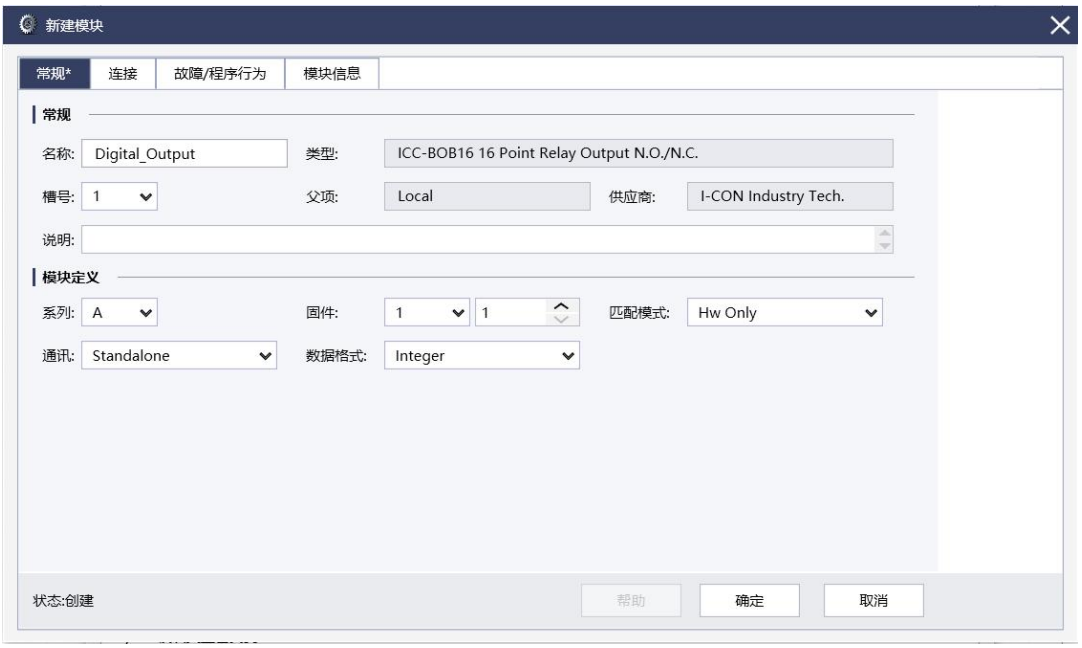


图 1-83　新建模块常规组态对话框

电子锁匹配模式有三种，分别是：HW and FW / HW only / Disable，对应的是精确匹配（固件版本号必须精确匹配）、兼容匹配（类型和主要版本号匹配）和禁用电子锁（只要类型正确，不检查版本号）。

(4) 选择 *连接* 选项卡，在这里设置报文周期（请求数据表间隔时间）以及是否禁用此模块。如图 1-84 所示。

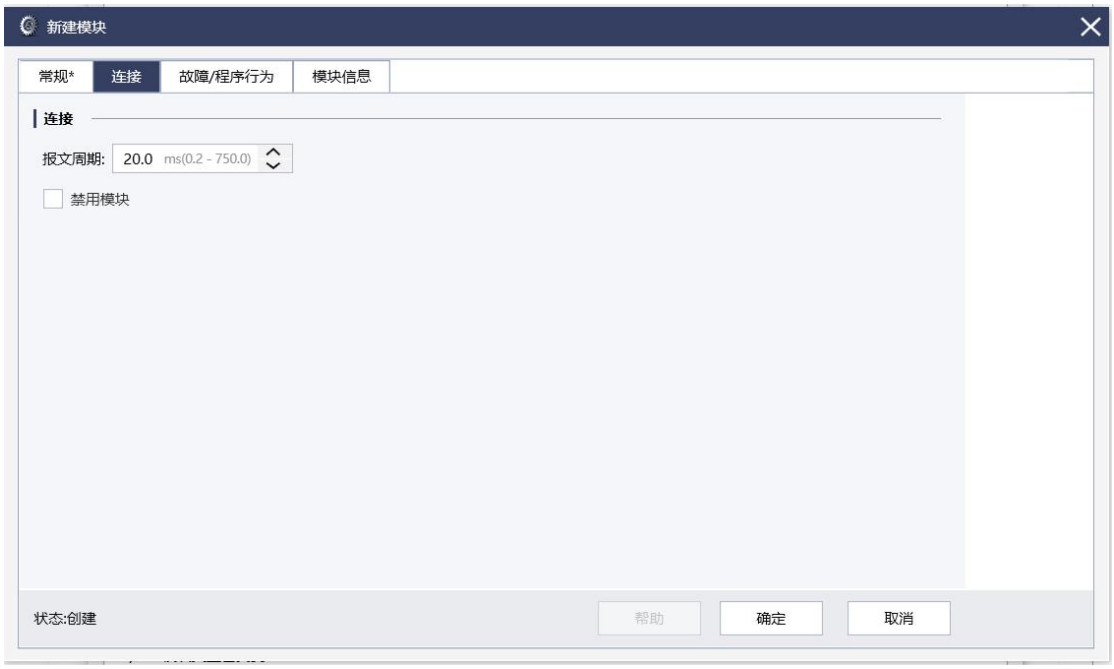


图 1-84　新建模块连接组态对话框

(5) 选择 *故障/程序行为* 选项卡，设置模块在编程模式下或故障模式下，各点的输出状态，有 OFF/ON/HOLD 三种状态可选择，如图 1-85 所示。

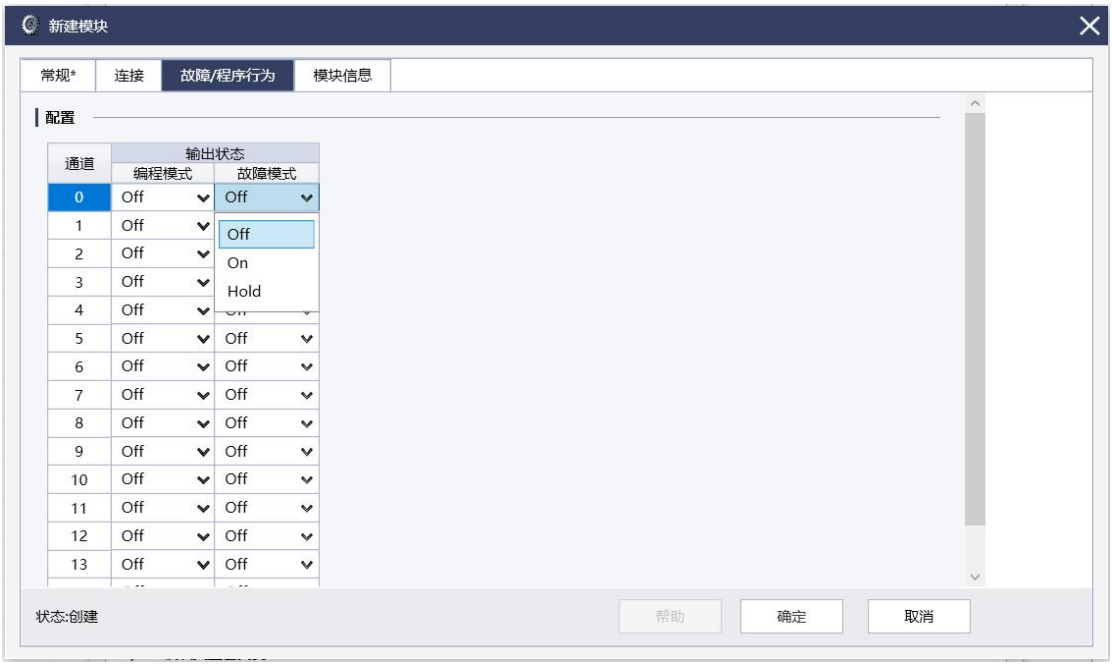


图 1-85　模块故障/程序行为组态窗口

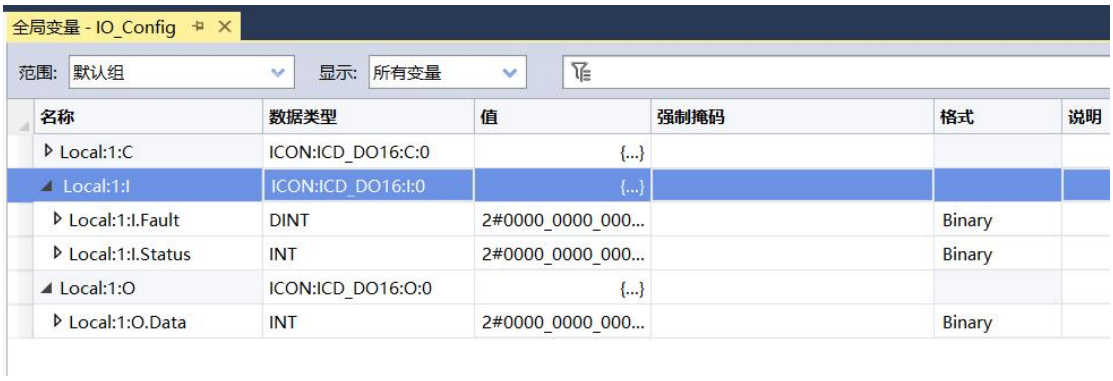
(6)　模块信息 窗口可以看到该模块的一般信息以及在线时的故障信息等，如图 1-86 所示。

图 1-86　模块信息窗口

(7)　点击 确定，模块便被组态到了扩展 I/O 下面了，如图 1-87 所示。

图 1-87　完成组态的 ICC-BOB16 模块

双击 控制器->全局变量->默认组 ，可以看到自动生成的 ICC-BOB16 模块标签结构体，如图 1-88 所示。



名称	数据类型	值	强制掩码	格式	说明
Local:1:C	ICON:ICD_DO16:C:0	{...}			
Local:1:I	ICON:ICD_DO16:I:0	{...}			
Local:1:I.Fault	DINT	2#0000_0000_000...		Binary	
Local:1:I.Status	INT	2#0000_0000_000...		Binary	
Local:1:O	ICON:ICD_DO16:O:0	{...}			
Local:1:O.Data	INT	2#0000_0000_000...		Binary	

图 1-88　ICC-BOB16 标签结构体

标签的组成如图 1-89 所示。

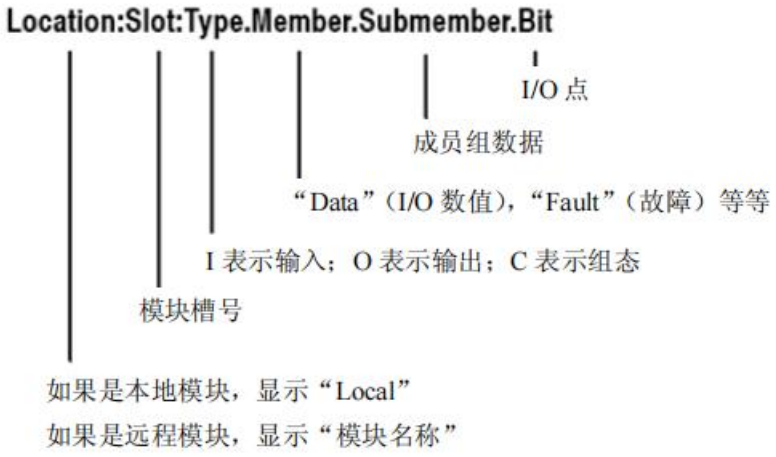


图 1-89　模块标签命名规则