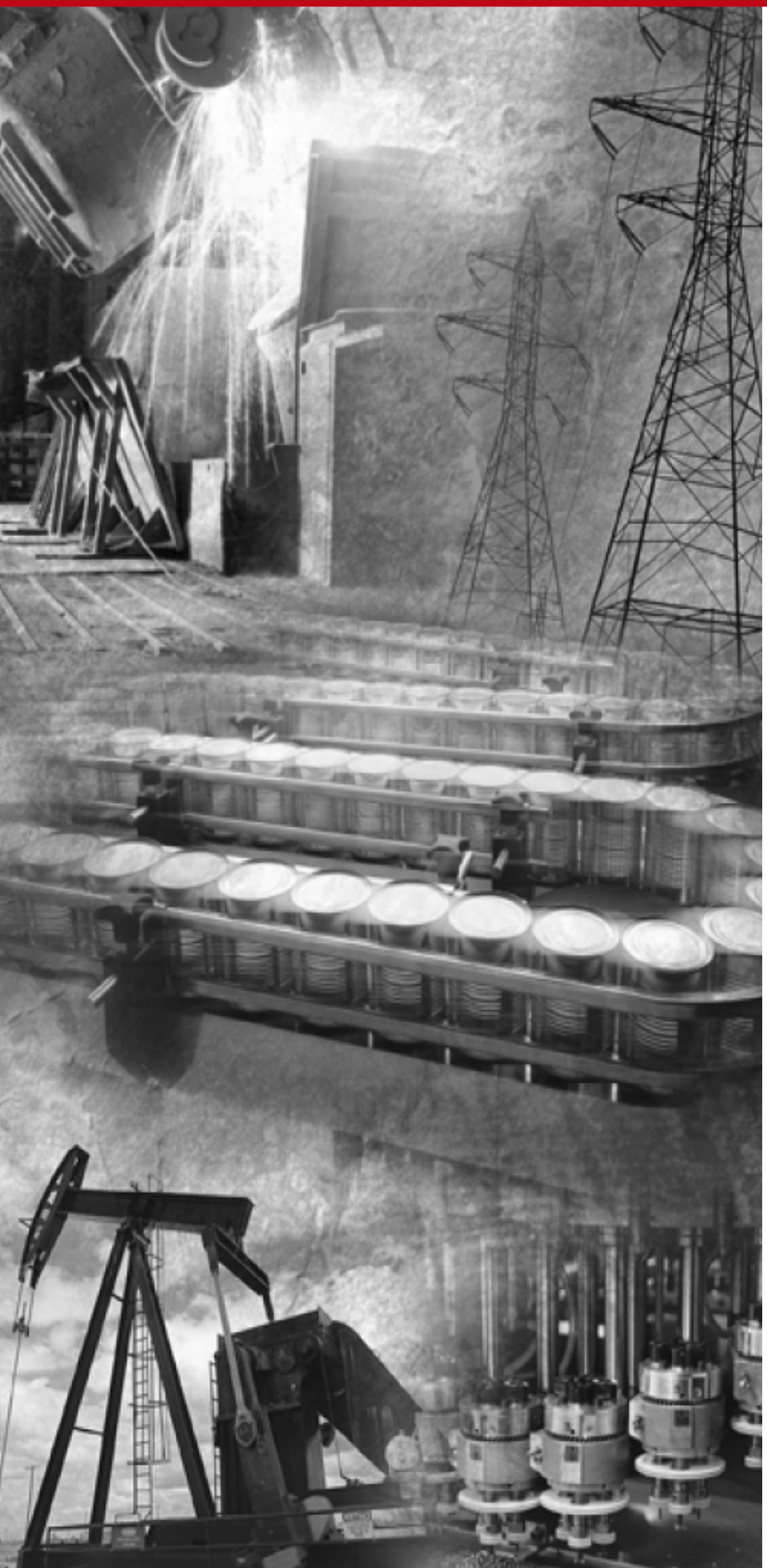


Allen-Bradley

Logix5000™ 控制器通用程序

1756-Lx,1769-Lx,1789-Lx, 1794-Lx
PowerFlex700

编 程 手 册

Allen-Bradley
RELIANCE ELECTRIC
DODGE
ROCKWELL SOFTWARE
Rockwell Automation

重要用户信息

鉴于本文所描述的产品应用范围十分广泛，负责本控制设备应用的人员必须保证做到以下几点：执行所有必要的步骤，以确保应用及使用过程达到所有性能及安全性要求，包括任何使用的法律、规章、法规及标准。

本手册中的插图、图表、示例程序和布线实例主要用于演示目的。由于实际的安装配置中有许多特殊要求和不确定因素。因此对依据本手册中的例子和图表所进行的实际应用，Allen-Bradley公司不承担任何责任或义务(包括知识产权责任)。

Allen-Bradley出版物SGI-1.1，"固态控制器应用、安装及维护的安全准则"(可从用户所在地的Allen-Bradley公司办事处获得)中，描述了固态设备与机电设备之间的重要区别。这些问题在应用本手册中描述的产品时应予以考虑。

未经Allen-Bradley公司的书面许可，不得对本手册中的部分或全部内容进行复制。

在本手册中，我们使用下面的符号提醒用户注意安全。使用这些符号以及相应的说明帮助用户识别并避免潜在的危险，以及认识危险产生的严重后果。

警　告

表明可能会导致人身伤害或死亡，财产损失，或经济损失的实践或事项信息。

注　意

表明可能产生潜在的爆炸危险的实践或事项信息。

重 要 提 示

指明对于产品的正确理解和应用特别重要的说明。

本手册系根据英文原文版翻译而成。手册中若有与英文不符之处，请以英文版本为准。

介绍

本手册中包括新增信息及修改信息。为了帮助用户找到这些新增信息及修改信息，要查找这些信息参见下面的目录。

最新信息

主要内容如下：

最新信息：	起始页码：
组态一个通讯驱动程序	9-1
下载一个工程到控制器(升级固件)	9-3
选择控制器模式	9-5
校正主要错误	9-6
发送一个信息到多处理器	10-1，10-13
与一个ASCII设备进行通讯	12-1
查阅缓存器代码	13-1，13-4
开发一个错误程序	15-1
清除非易失性内存	19-10
给一个工程加密	20-1
高速缓存器	术语表-6
路径	术语表-24
预扫描	术语表-26
源密码	术语表-33

备注：

手册的目的

这本手册将指导用Logix5000控制器开发一个工程。它提供了如何逐步执行以下任务，这些适合所有的Logix5000控制器。

- 管理工程文件
- 管理用户逻辑
- 管理标签
- 编程
- 测试工程
- 故障处理

术语*Logix5000控制器*指的是任何一个基于Logix操作系统的控制器，例如：

- CompactLogix™controllers(CompactLogix™控制器)
- ControlLogix™ controllers(ControlLogix™控制器)
- FlexLogix™ controllers(FlexLogix™控制器)
- SoftLogix™ controllers(SoftLogix™控制器)

这本手册可以跟一些特殊类型控制器的用户手册一起使用。用户手册包含的内容如下：

- 安装和组态I/O
- 各种网络设备之间的通讯
- 电池保养

本手册的使用对象

这本手册是专门为那些使用Logix5000控制器用户提供的，例如：

- 软件工程师
- 控制工程师
- 应用工程师
- 检测技术人员

使用本手册

当进行以下操作时需要用到本手册：

- 为应用程序开发机器代码
- 修改现有的应用程序
- 对用户程序进行单独测试

当用户在系统中对各种I/O设备，控制器，网络类型进行综合应用时：

- 特殊类型的控制器需要查阅用户手册
- 如有需要可将本手册作为参考

如何使用本手册

本手册按照Logix5000控制器的编程界面分成许多基本的任务进行讲解：

- 每一章包含一个任务
- 这些任务都按照典型的执行顺序进行管理

在使用本手册时用户将会发现有一些术语格式跟下文的不同:

正文中是：	标识：	举例：	含义：
斜体	在屏幕上或例子中实际条目名	鼠标右键点击User-Defined (自定义)	在条目名User-Defined (自定义)上右键点击
粗体	术语表中的一个链接	型如name(名字)…	如果用户想知道的name(名字)附加说明，请查阅术语表 如果用户想浏览name(名字)的PDF文档，点击name(名字)就可以进入术语表
专业术语	用户必须提供基于应用程序的信息 (一个变量)	右键点击 name_of_program... (程序名)	用户必须在应用中标识特殊的程序。代表性的如，用户所定义程序名或变量。

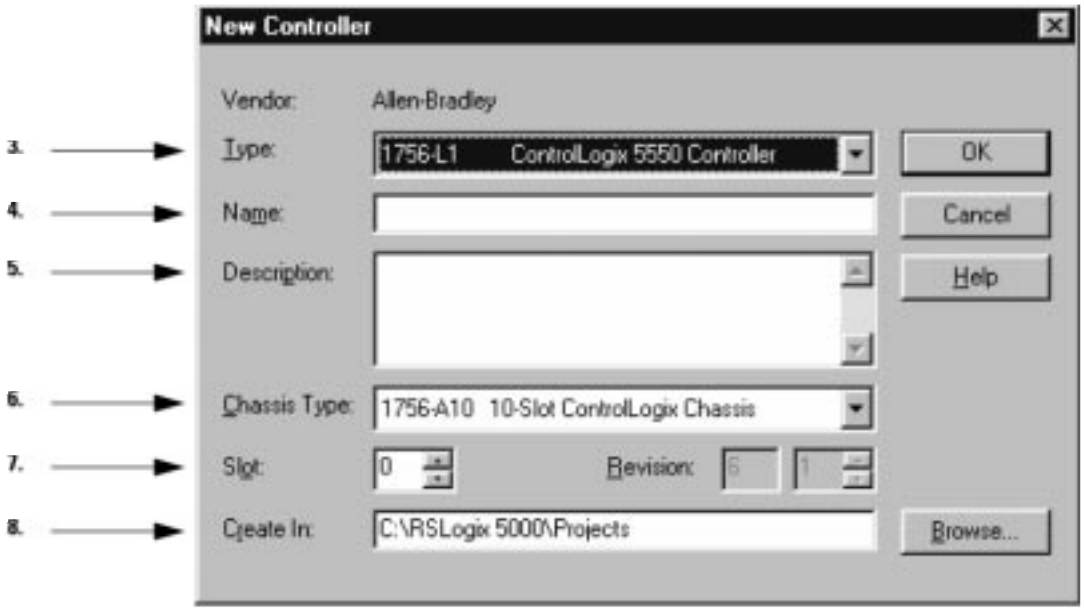
目录

管理工程文件

创建一个工程文件

在用户对一个Logix5000控制器编程之前，必须首先创建一个工程文件：

1. 打开RSLogix5000™软件。
2. 从File(文件)菜单选择New(新建)。



3. 选择控制器类型。
4. 给控制器输入一个**名字**。
5. 输入控制器所执行操作的说明(可选)。
6. 选择包含有控制器(不适用于某些控制器)的框架类型(槽数)。
7. 选择或填写控制器所在的槽号(不适用于某些控制器)。
8. 将文件保存在不同的文件夹(除了缺省的路径)，点击Browse(浏览)并选择文件夹。

9. 点击OK(完成)。

当用户创建一个工程时，工程文件名和控制器名是相同的。

保存用户修改

当用户创建梯形图逻辑和修改组态信息后，需保存工程。

如果想要：	操作如下：
保存修改	File(文件)菜单选择Save(保存)。
保存一个副本并且使用现有的控制器名	A. 从File(文件)菜单选择Save As(另存) B. 输入一个工程文件名。空格用下划线[_]代替。 C. 点击Save(保存)
保存一个副本并且重新指定控制器名	A. 从File(文件)菜单选择Save As(另存) B. 输入一个工程文件名。空格用下划线[_]代替。 C. 点击Save(保存) D. 在控制器项目管理器中右键点击 *Controller name_of_controller (控制器名)选择属性* E. 为控制器输入一个新的名字 F. 点击OK (完成)

注意：

- 指定所要**下载**的控制器，文本(说明、梯级注释)不能下载到控制器中。
- 如果想要修改控制器的名字，框架大小或槽号：

a. 在控制器项目管理器中，右键点击
Controller name_of_controller(控制器名)文件夹并选择属性

b. 修改需要的信息

c. 点击OK.(完成)。

注意：

管理任务

使用本章

在用户创建完工程文件之后，将工程组织到任务中。

如何使用本章

按下列步骤将用户工程组织到任务中：

- 确定现有的编程语言
- 组织用户逻辑
- 校验控制器

确定现有的编程语言

用户可以通过查看下表来确定控制器所能使用的编程语言：

控制器平台	可用的语言：	
	梯形图	功能块
CompactLogix	✓	✓
ControlLogix	✓	✓
FlexLogix	✓	✓
SoftLogix	✓	

注意：

- 因为控制器可用多种语言进行编程，所以在工程中可以混合使用编程语言。
- 要使用功能块语言编程，用户必须安装目录号为-9324-RLD700的RSLogix5000软件。
- 要查看RSLogix5000软件安装的内容：

1. 打开RSLogix5000软件。
2. 从*Help*菜单选择*About RSLogix5000*(关于RSLogix5000)。

组织用户逻辑

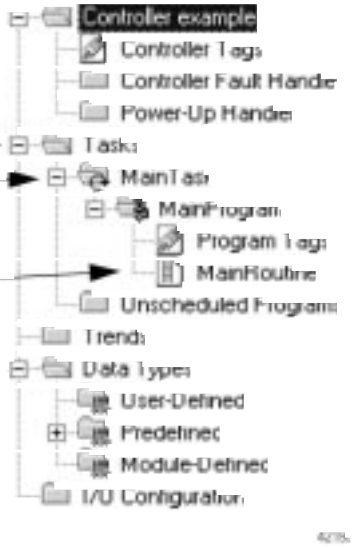
要执行用户逻辑就需用一个(或多个)**任务**。有两种类型的任务：

任务类型：	特点：
连续性任务	连续执行用户逻辑(用户只能有一个连续任务。)
周期性任务	• 中断连续性任务 • 每次执行一遍逻辑 • 返回到连续性任务 (用户可以有多个周期性任务)

在控制器项目管理器中显示了一个控制器的所有任务。

任务文件夹包含控制器的所有任务(也就是用户逻辑)

在缺省状态下主任务为连续性任务，并且始终保持运行状态，反复执行(*Main Program*)主程序。
只要执行(*Main Program*)主程序，它里面的逻辑就将执行。用户可以在(*Main Program*)主程序里调用其它程序(子程序)。



为用户逻辑选择任务：

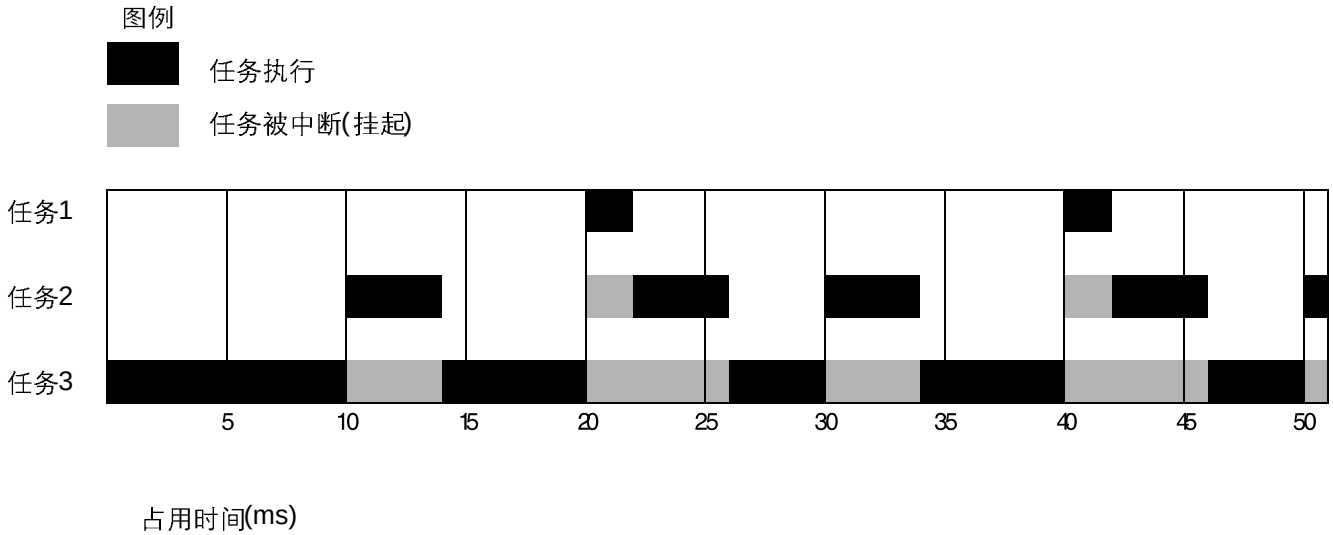
如果用户执行：	那么就：	详细步骤：
以恒定的扫描时间执行一个函数(例如：每100毫秒执行一个PID回路)	1.为函数创建一个周期性任务任务)。	A.在控制器项目管理器中右键点击Tasks(任务)并选择New task(新建 B.输入： • *name_of_task*(任务名) • *description*(可选) C.从Type(类型)列表中选择Periodic(周期性)项。 D.在Periodic Attributes(周期性属性)中输入： • *Rate*(频率) • *Priority*(优先级) E.单击OK(完成).
快速执行一个函数	2.给任务建立一个程序	A.右键点击*name_of_task*(任务名)并选择 *New Program*(新建程序) B.输入： • *name_of_program*(程序名) • *desctiption*(可选) C.单击OK(完成).
	3.创建并指定**主例程**(在程序中首先执行的例程)。	A.点击*name_of_task*(任务名)前的＋ 标识符 B. 右键点击*name_of_program*(程序名)并选择*New Routine*(新建例程). C.输入： • *name_of_main_routine*(主例程名) • *desctiption*(可选) D.从*Type*(类型)下拉列表中选择编程语言。 E.点击OK(完成). F.右键点击*name_of_program*(程序名)并选择*Properties*(属性)。 G.点击*Configuration*(组态)选项。 H.从*Main*的下拉列表中选择 • *name_of_main_routine* I.点击OK. J.重复以上的步骤B到E可添加新的例程(子程序)。
	4.将其余的函数放到Main Task(主任务)，MainProgram(主程序)。(见下一步骤)	
执行多函数时用逻辑来控制执行顺序	1.给每一个函数创建一个例程(子程序)：	A.在控制器项目管理器中右键点击*MainProgram*(主程序)选择*NewRoutine*(主例程) B.给例程输入下列属性： • *name_of_routine*(例程名) • *description*(可选) C.从*Type*(类型)下拉列表中选择编程语言。 D.点击OK(完成)..
	2.用一个JSR指令调用一个子程序。	
同时执行所有的函数	1.在*MainRoution*(主例程)中输入用户梯形逻辑。 2.使用梯级注释来标明不同的函数。	

以下是执行一个多任务工程的例子：

例 子：

在包含两个周期性任务和一个连续性任务的工程中，其任务执行的顺序是：

任务：	任务类型：	优先级别：	执行时间：
1	20ms周期性	5	2ms
2	10ms周期性	10	4ms
3	连续性	无(最低)	24ms



注意：

- 所有的周期性任务可中断连续性任务。
- 优先级最高的任务可中断所有优先级比它低的任务。
- 优先级高的任务可以多次中断比它优先级低的任务。
- 控制器完成对连续性任务的一次扫描后将继续重新开始扫描。
- 相同优先级的任务执行时间间隔为1ms。
- 要想修改一个任务，一个程序或一个例程的属性(名称、类型、属性等等)，右键点击这个任务、程序或例程并选择*Properties(属性)*。

校验控制器

用户在给一个工程编程时，要定时地进行校验：

1. 在RSLogix5000编程窗口最顶端的工具栏中，点击 
2. 如果发现错误，将表列在窗口的底部：
   a. 找到第一个错误或报警，按*F4*键。
   b. 根据结果窗口中所描述的错误进行校正。
   c. 返回第一步。
3. 关闭结果窗口，按*Alt +1*键。

注意:

管理标签

规划用户标签

Logix5000控制器将数据保存在**标签**中(对照用数字编址的固定数据文件)。用户可用标签进行:

- 通过管理用户数据来监测机器的状态。
- 建立的文档(通过标签名称)用于开发应用程序。

在创建一个标签时,需要指定以下属性:

表3.1 标签属性

属性:	**描述:**
作用域	定义哪些例程可以访问这个数据
名字	识别数据(在不同作用域的标签可以用同样的名字)。
数据类型	定义数据结构,如:位文件,整数文件或浮点文件。

下表概述了最常用的数据类型以及何时采用这些数据类型。

表3.2 数据类型

应用:	**选择:**
工作在浮点模式下的模拟量设备	REAL(实型)
工作在整型模式下的模拟量设备(高频)	INT(整型)
ASCII 字符	string(字符串)
位	BOOL(布尔量)
计数器	COUNTER(计数器)
数字量I/O点	BOOL(布尔量)
浮点数	REAL(实型)
整数(纯数据)	DINT(双整型)
顺序器	CONTROL(控制器)
计时器	TIMER(计时器)

根据下表管理用户数据：

对一个：	使用：	参考：
被多台机器使用的通用属性组	**自定义数据类型**	参见3-8页的 ″创建一个自定义的数据类型″。
具有相同数据类型的数组	**数组**	参见3-10页的 ″创建一个标签″。
单个数值	只含一个元素的标签	
I/O 设备		

下面的一些例子告诉用户如何组织不同类型的数据：

- 单个元素标签，见3-3页
- 一维数组，见3-4页
- 二维数组，见3-5页
- 保存配方的自定义数据类型，见3-6页
- 用来保存自定义的数据类型，该数据要求运行一台机器，见3-7页

例 子 单个元素标签

模拟量I/O设备 →
整数值 →
存储位(二进制数) →
计数器 →
定时器 →
数字量I/O设备 →

例 子：

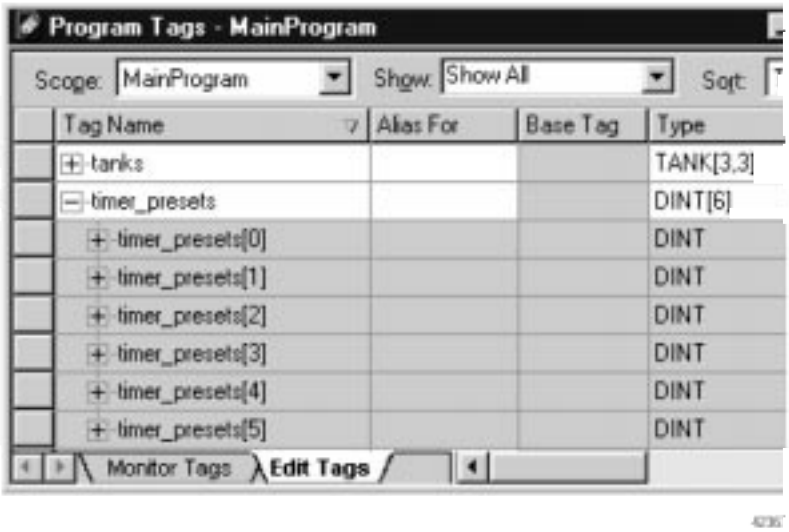
一维数组

在这个例子中，单个定时器指令可以按几档不同的时间宽度来定时。每一档都要求有不同的预置值。因为一个数组中的所有数值都具有相同的数据类型(双整型)。

为了展开一个数组并显示其元素，用鼠标单击＋标识符

为了折叠一个数组并隐藏其元素，用鼠标单击－标识符

timer_presets(计时器预置值)的元素



这个数组包含了六个数据类型为DINT的元素。

六个DINT型元素

例 子：

二维数组

一个打孔机可以在一本书上打一到五个洞。机器根据每个洞到书边缘的距离值来确定孔的位置。要组织这些数据，使用一个二维数组。第一个下标表示定位洞所需要的数据，第二个下标表示需要打洞的数目(一到五个)。

	第二维的下标							说明
		0	1	2	3	4	5	
第	0							
一	1		1.5	2.5	1.25	1.25	1.25	第一个洞相对于书边缘的位置
维	2			8.0	5.5	3.5	3.5	第二个洞相对于书边缘的位置
的	3				9.75	7.5	5.5	第三个洞相对于书边缘的位置
下	4					9.75	7.5	第四个洞相对于书边缘的位置
标	5						9.75	第五个洞相对于书边缘的位置

在这个标签窗口中，元素按照下面的顺序进行描述。

例 子：

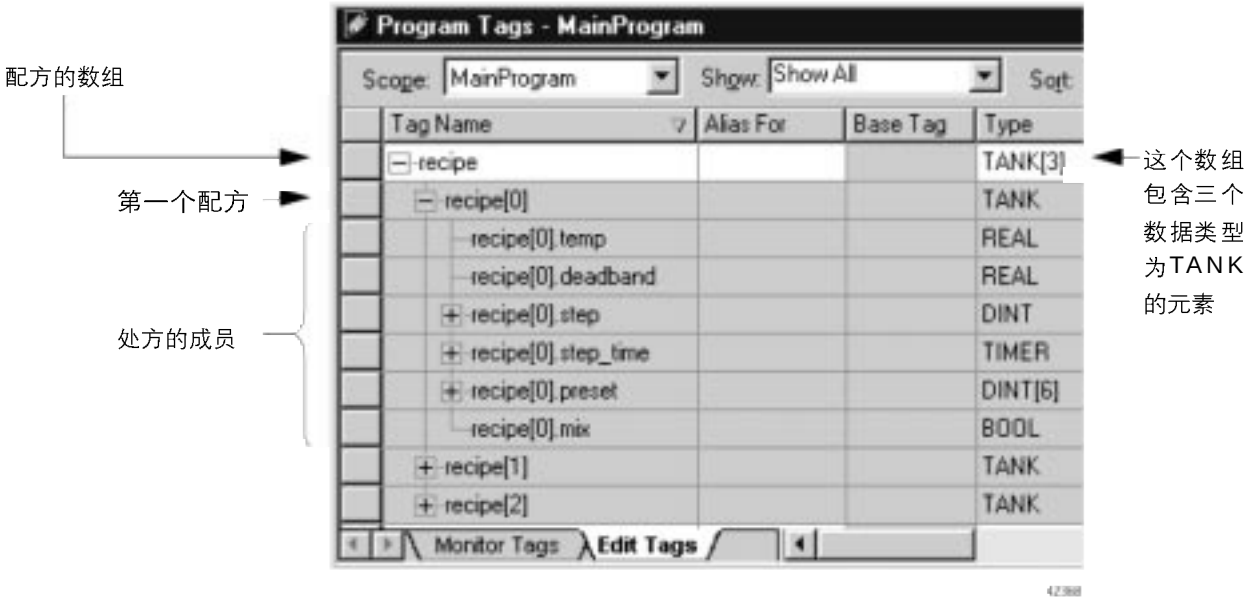
保存配方的自定义数据类型

在一个系统中有几个配料罐，每个罐都可以管理一类配方。因为这个配方需要混合的数据类型(REAL,DINT,BOOL,等)，所以需要使用自定义数据类型。

名称(数据类型)：TANK

成员名称	数据类型
temp	REAL
deadband	REAL
step	DINT
step_time	TIMER
preset	DINT[6]
mix	BOOL

基于此数据类型构建的数组如下图所示：

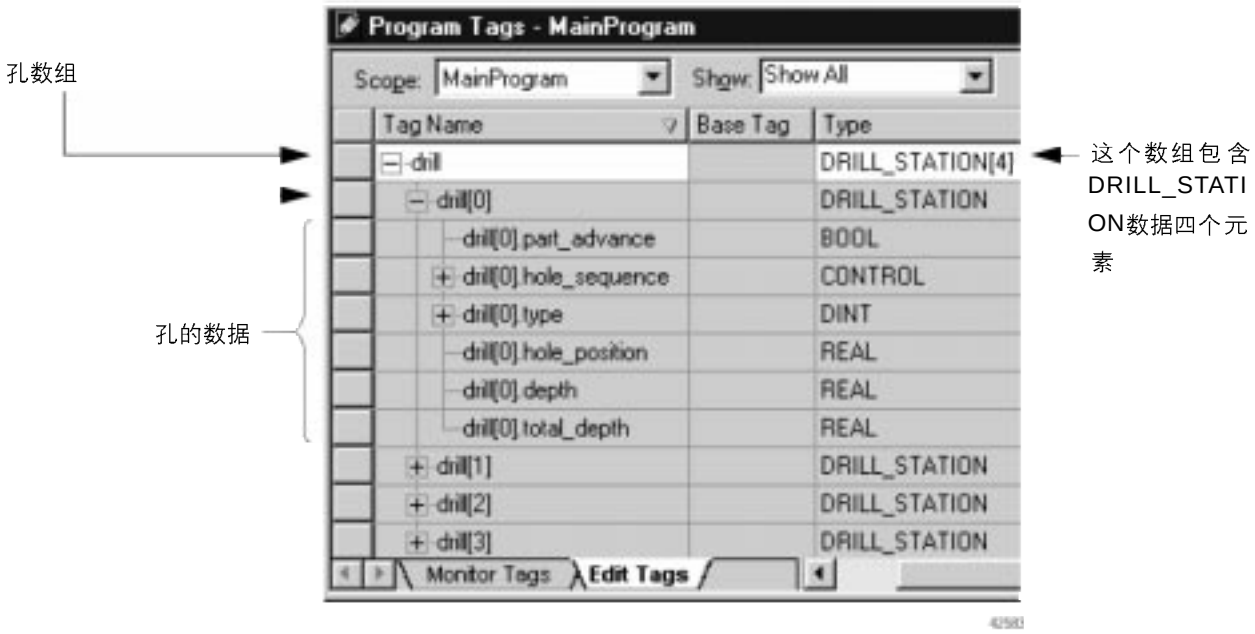


例 子：

例子 用来保存自定义的数据类型，该数据要求运行一台机器
因为许多孔的位置需要下列混和数据，这样就要创建一个自定义数据类型。

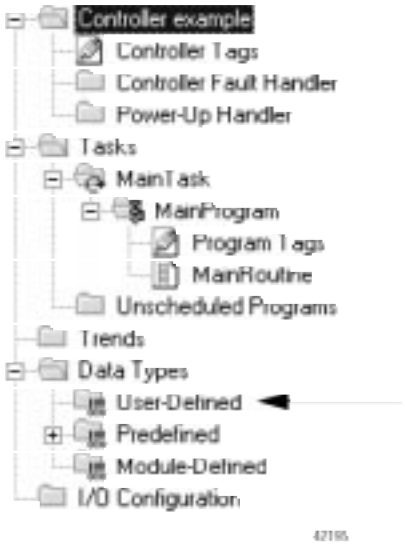
名称(数据类型)：DRIL_STATION	
成员名称	**数据类型**
part_advance	BOOL
hole_sequence	CONTROL
type	DINT
hole_position	REAL
depth	REAL
total_depth	REAL

基于此数据类型构建的数组如下图所示：

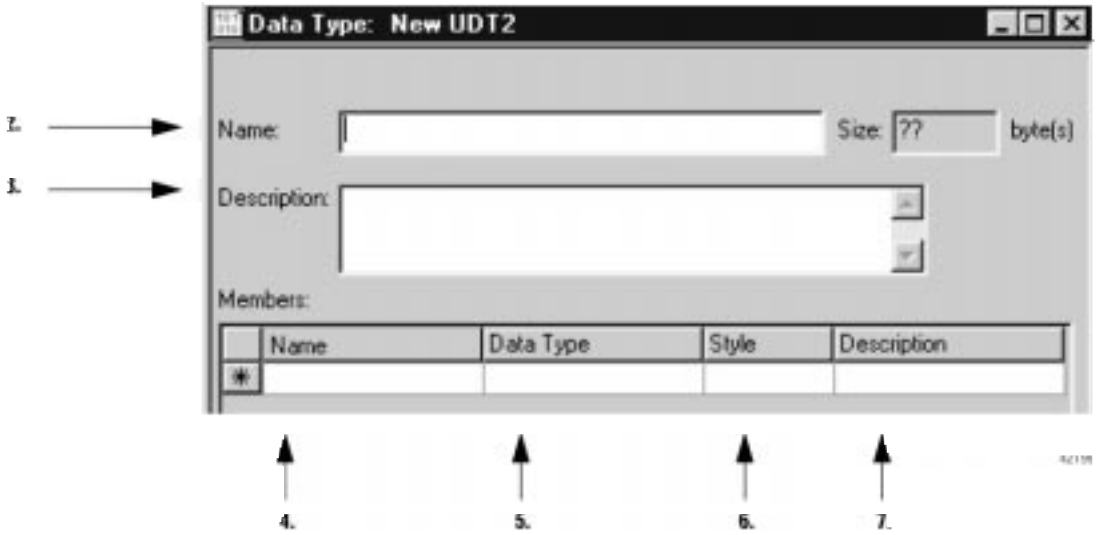


创建自定义数据类型

创建一个自定义的数据类型：



1、 右键单击*User-Defined*(自定义)并选择*New Data Type*(新建数据类型).。



2、 填写一个数据类型的*name*(名字)。

3、 填写一个**description**(描述)(可选)。

4、 填写第一个**member**(成员)的名字。

5、 确定成员的数据类型。参见3-1页的表3.2。

对于一个数组，使用如下的格式：

data_type [x]

其中：

x在这个数组中元素的个数。

例 子： 如果这个成员是一个含6个DINT型数据的数组，输入DINT[6]。

6、 用不同形式(基数)显示成员的数值，选择stytle(形式)。

7、 为成员输入描述(可选)。

8、 点击*Apply*(应用).

9、 是否想建更多成员？

如果：	那么就：
是	重复步骤4到8
否	点击*OK*(完成).

注意：

- 如果成员代表的是I/O设备，那么用户必须使用梯形图逻辑将数据复制到结构体的成员中和I/O设备相对应的标签中。参见8-1页 ″缓存I/O″。
- 在使用BOOL、SINT或INT数据类型时，将相同数据类型的数据按下列顺序放在成员中：

效率高
BOOL
BOOL
BOOL
DINT
DINT

效率低
BOOL
DINT
BOOL
DINT
BOOL

创建一个标签

创建一个标签(包括一个数组)：

1、 从Logic(*逻辑*)菜单中选择*Edit Tags(编辑标签)*。

2、 为标签选择一个作用域：

如果想用一个标签：	**那么选择：**
在同一个工程的多个程序中	*name_of_controller(controller)*
作为一个生产型或消费型	
在信息中	
仅在工程中一个程序中使用	用这个标签编程

3、 为标签输入一个名字。

4、 输入数据类型：

如果标签是：	**那么输入：**
不是一个数组(文件)	*data_type*
一维数组	*data_type[x]*
二维数组	*data_type[x,y]*
三维数组	*data_type[x,y,z]*

其中：

*data_type*是标签或数组存储数据的类型。
参见3-1 页表3.2

x 是**元素**的第一维
y 是**元素**的第二维
z 是**元素**的第三维

5、 输入一个**描述**(可选)。

用Microsoft® Excel® 创建标签

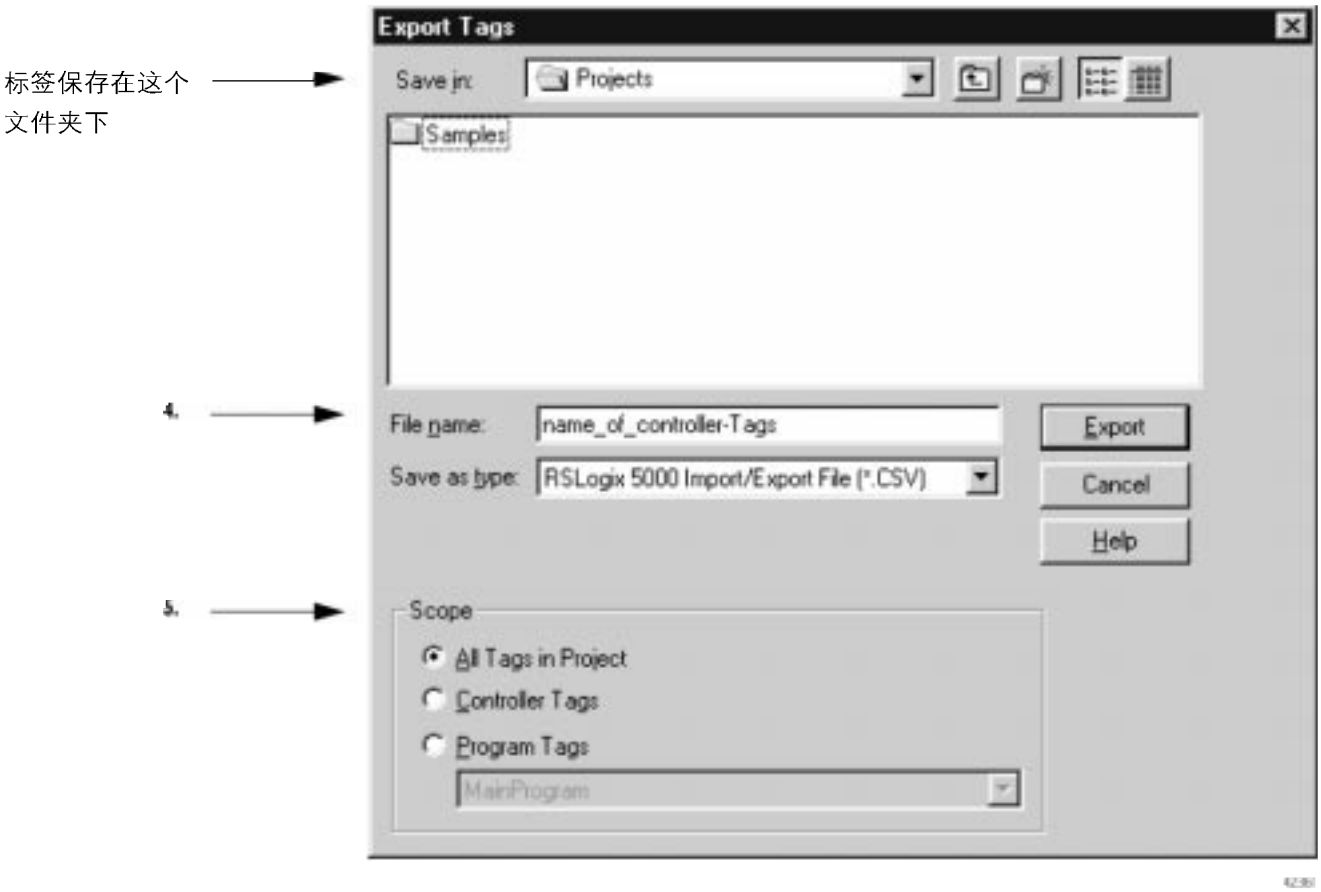
用户可以用电子制表软件(例如：Microsoft Excel)来创建和编辑标签。这样用户可以利用电子制表软件编辑的特点。

用Excel创建标签：

1、 打开RSLogix5000的工程。

2、 创建一些标签。(这样有助于使用Excel电子表格的形式)

3、 从*Tools*(工具)菜单中选择*Export Tags*(导出标签)。



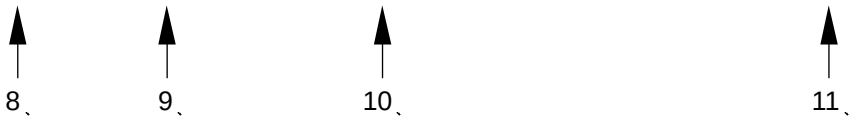
4、 记住导出的文件名(*project_name-tags*).

5、 选择导出标签的作用域。如果用户选择*Program Tags*(程序标签),那么会将程序区标签导出。

6、 点击*Export*(导出)。

7、 Mircosoft Excel 软件中打开导出的文件。

TYPE	SCOPE	NAME	DESCRIPTION	DATATYPE
TAG		in_cycle		DINT
TYPE	SCOPE	NAME	DESCRIPTION	DATATYPE
TAG	MainProgram	conveyor_alam		BOOL
TAG	MainProgram	conveyor_on		BOOL
TAG	MainProgram	drill_1		DRILL_STATION
TAG	MainProgram	hole_position		REAL[6,6]
TAG	MainProgram	machine_on		BOOL



8、 输入TAG

9、 确定标签的作用域：

如果作用域是：	那么：
控制器	这个单元空着
程序	输入程序名

10、输入标签名。

11、输入标签的数据类型。

12、每添加一个标签需重复步骤8到10。

13、保存并关闭文件。(保存为.CSV格式)。

14、在RSLogix5000软件中从Tools(工具)菜单栏选择*Import Tags*(导入标签)。

15、选择标签文件，点击*Import*(导入)。

标签导入到工程中。在RSLogix5000软件窗口的底部可看到显示的结果。

注意：

- 用户可以组态标签与其它控制器直接进行通讯：

如果要：	使用一个：
将数据按一定的时间间隔发送到背板和ControlNet网络上	**生产型标签**
按一定的时间间隔从背板或ControlNet网络上接收数据	**消费型标签**

用户如果想使用产生或消费型标签，在组织标签时就必须遵守以下规定。参见10-1页的“与其它控制器通讯”。

- 以下的整数数据类型也可使用：
  - -SINT(8位整数)
  - -INT(16位整数)

一般情况下，在程序执行期间，指令会将SINT或INT值转换成最佳的数据类型(通常为一个DINT或REAL值)。因为这些数据类型将占用额外的扫描时间和内存，尽量少用SINT和INT型数据。

注意：

编辑例程

使用本章

在组织完工程最初的例程和标签之后，使用本章来开发每个例程将要执行的逻辑。

如何使用本章

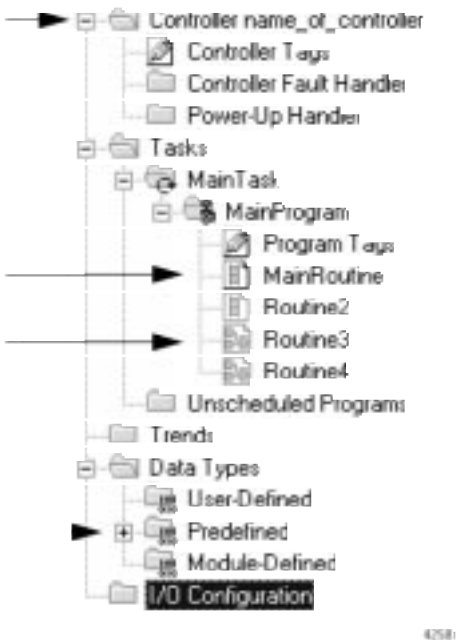
编程例程时按以下步骤执行：

- 打开例程
- 输入梯形图指令
- 输入功能块指令
- 指定操作对象
- 校验例程

打开例程

要关闭一个文件夹和隐藏它的内容(折层)，任选下面的一个操作：

- 双击文件夹
- 选择文件夹然后按←键
- 点击－标志



要打开一个例程，双击例程。

如果例程标志是灰色的，则打不开。

要打开一个文件夹和显示它的内容(折层)，任选下面的一个操作：

- 双击文件夹
- 选择文件夹然后按→键
- 点击"+"符号

查看例程的图标是否为灰色?

如果图标：	那么：
不是灰色	双击例程
灰色	用户不能打开，编程或编辑例程。查出原因： 1. 双击例程 2. 在 RSLogix5000编程窗口的底部，显示了一些状态信息：

如果：	那么：
打开例程失败，程序编辑器没有安装	功能块编辑器没有安装。要安装功能块编辑器，购买下列目录号的RSLogix5000软件: • 9324-RLD700
源程序不可用	源程序是不可用的。 用户可以： • 运行例程 • 显示例程的属性 • 可相互识别例程中的逻辑 用户不能： • 打开(显示)例程 • 编辑例程 • 改变例程属性 • 查找例程 • 在例程内互相参考 • 打印例程 • 导出例程

重要提示：

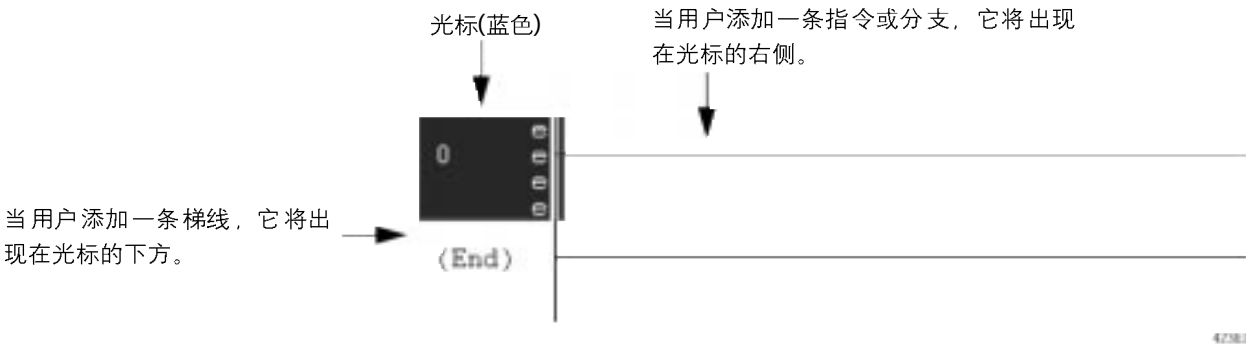
如果一个例程的源代码不可用，不要导出该工程。

- 一个导出的文件(.L5K)只包含具有可用源代码的例程。
- 如果用户导出一个工程，其中源代码对所有例程都不可用，用户就无法恢复整个工程。

输入梯形图指令

在打开的梯形图例程中：

1、 如果例程已包含逻辑指令，用户想添加逻辑指令只需要点击要添加的地方就可以进行添加。(一个新的例程包含已经准备好梯级。)



2、添加梯级中的元素：

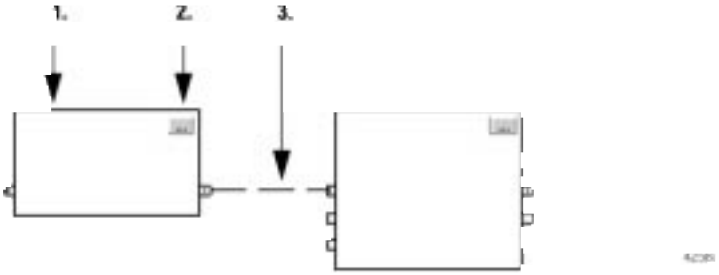
要添加一个：	操作如下：
梯线	*按Ctrl＋R键*
指令	A.按*Insert*(插入)键 B.输入指令的助记符 C.按*Enter*(回车)键
分支	A.按*Inser*(插入)t键 B.输入BST。 C.按*Enter*(回车)键。 D.输入新的指令后，拽着分支右边的引线放到梯线需要的地方。

输入功能块指令

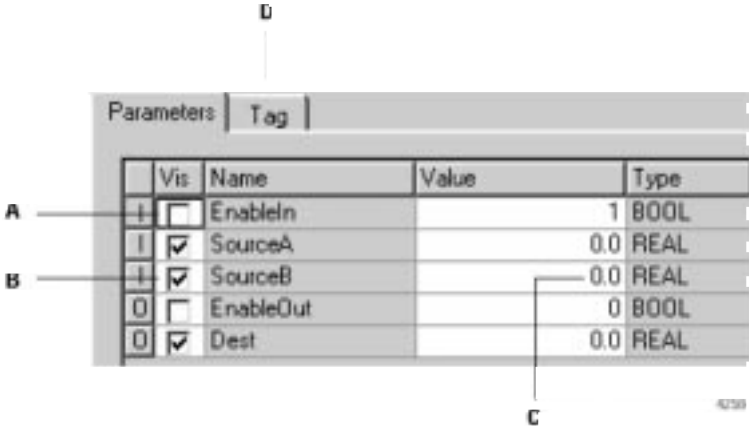
一个功能块例程包含下列元素：

元素：	目的：
功能块	对输入值执行一次操作并产生一个输出值 • 功能块左边的引脚为输入引脚 • 功能块右边的引脚为输出引脚
输入参考值(IREF)	这个值由输入设备、其它例程或控制器的标签提供
输出参考值(OREF)	这个值由输出设备、其它例程或控制器的标签提供
输出连接器(OCON)	连接相隔很远或在不同图表上的功能块。
输入连接器(ICON)	• 每一个OCON需要指定唯一的名字。 • 每一个OCON至少有一个ICON与之对应(例如：一个ICON与一个OCON具有相同的名字) • 多个ICON可指向一个OCON。这样就允许用户将例程中的数据分配到多个点。

要输入功能块指令，先打开一个功能块例程然后完成以下步骤：



步骤：	具体操作：
1、 输入例程所要执行函数的功能块。	A.按*Insert*（插入)键 B.输入所需功能块的助记符 C.选择*OK*(完成).。 D.将功能块拽到图表易于读取的地方。功能块的位置并不影响其执行顺序。
2、 组态每个功能块的属性	A.在功能块中，点击 ...

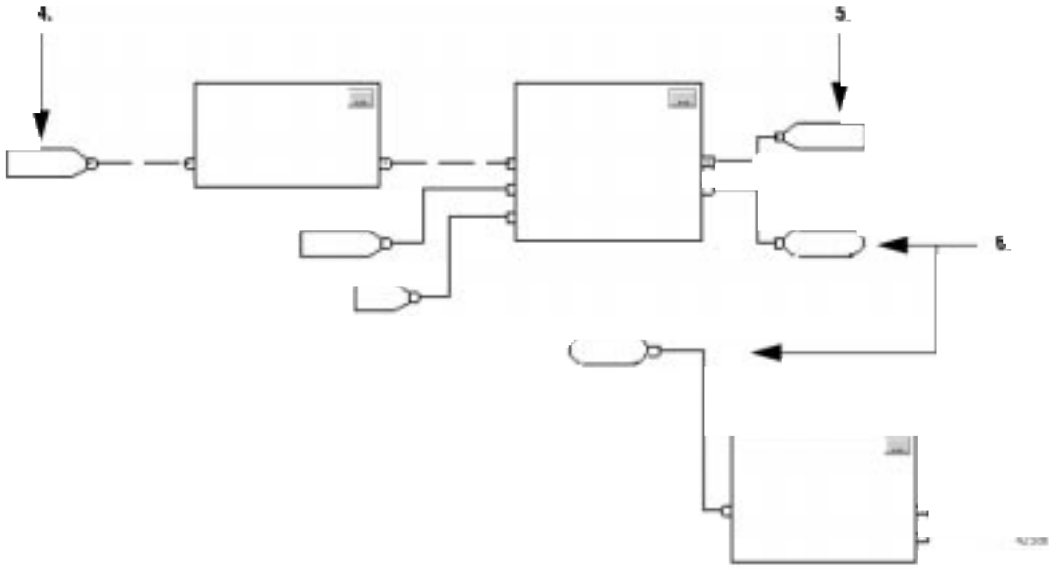


B.编辑功能块的属性：

如果要：	操作如下：
显示一个引脚的操作对象	选择操作对象后的选择框(A)
输入一个立即数值	A.清除(不选)操作对象后的选择框(B)。 B.在数值栏，给操作对象输入数值(C)。
修改指令的标签名	A.点击标签表(D)。 B.输入新的名字。

C.选择OK(完成)。

3、将输出引脚连到输入引脚	先点击一个输出引脚然后再点击要连接的输入引脚。绿色的点显示引脚有效。



步骤：	具体操作：
4、要从一个输入设备或标签获取一个数值，键入一个输入参考值(IREF)	A. 按Insert(插入)键 B. 选择*OK*(完成).(IREF 为缺省值)。 C. 将IREF 拽到需要的地方，一般将其放在功能块的左边便于输入数值。 D. 先点击IREF的引脚，然后再点击使用数值的输入引脚。
5、要给一个输出设备或标签提供一个数值，键入一个输出参考值(OREF)	A. 按*Insert*(插入)键 B. 输入*OREF*. C. 选择*OK*(完成)。 D. 将*OREF*拽到需要的地方，一般将其放在功能块的右边便于产生输出数值。 E. 先点击提供数值的输出引脚，然后再点击OREF的引脚。
6、要连接相隔很远或在不同图表上的功能块，输入一个输出连接器(OCON) 和一个输入连接器(ICON).	A. 按*Insert*(插入)键 B. 输入*OCON*. C. 选择*OK*(完成)。 D. 将OCON 点击到需要的地方，一般将其放在功能块的右边便于产生输出数值。 E. 先点击提供数值的输出引脚，然后再点击OCON 的引脚。 F. 显示图表中用户相连接的功能块 G. 按*Insert*(插入)键 H. 输入*ICON*. I. 选择*OK*(完成)。 J. 将ICON 拽到需要的地方，一般将其放在功能块的左边便于输入数值。 K. 先点击ICON 的引脚，然后再点击使用数值的输入引脚。

指定操作对象

每个指令需要以下选项中的一项或多项：

- 标签名
- 数值
- 例程名，标注，连接器等等

下表简述了标签名的格式：

类型：	规定：
标签	*tag_name*
位文件数据类型	*tag_name.bit_number*
数据结构的成分	*tag_name.member_name*
一位数组元素	*tag_name[x]*
二维数组元素	*tag_name[x,y]*
三维数组元素	*tag_name[x,y,z]*
结构体中的数组元素	*tag_name.member_name[x]*
数组元素成员	*tag_name[x,y,z].member_name*

其中：

x 是元素的第一维
y 是元素的第二维
z 是元素的第三维

在结构体中嵌套的结构体，添加另一个.member_name.

例 子 标签名

要访问：	标签名如下：
machine_on标签	machine_on
位文件为1的one_shots标签	one_shots.1
*running_seconds*计时器的*DN*位	running_seconds.DN
*north_tank*标签中的*mix*位	north_tank.mix
*recipe*数组中的元素2和tanks数组中的元素1.1	COP Copy File Source recipe[2] Dest tanks[1,1] Length 1
*north_tank*标签里，*preset*数组中的元素2	CLR Clear Dest north_tank.preset[2] 0
*drill*数组中元素1的*part_advance*位	drill[1].part_advance

42357

给指令指定一个立即数、标签、连接器、标注或类似的条目。

1、 填写或选择数值或条目名(例如：标签名，连接器名，标注名)：

如果用户想：	选择：	操作如下：
• 指定一个立即数 • 输入标签名，连接器名，标注或类似条目。	梯形图指令	A.点击？标记 B.输入数值，标签名，标注或类似条目 C.按Enter(回车)键
	功能块指令	A.点击？标记 B.再次点击？标记 C.输入数值，标签名或连接器名 D.按Enter(回车)键
从列表中选择一个标签，连接器，标注或类似条目。		A.打开条目表

如果是：	操作如下：
梯形图指令	双击？标记
功能块指令	A.点击？标记
	B.再次点击？标记

B.点击 ▼

C.选择一个名：

要选择一个：	操作如下：
连接器，标注或类似条目名	选择相应的名字
标签	双击标签名
位文件	A.点击标签名。
	B.在标签名的右边点击▼
	C.点击需要的位

D.按Enter(回车)键或点击框图上其它地方。

从标签窗口拽一个标签	梯形图指令		A.在标签窗口找到标签的位置 B.点住标签2到3秒直到变亮 C.将标签拽到指令上
		功能块指令	不能用

2、 确认用户是否已经定义(创建)了标签？

如果：	那么就：
是	进行下一项操作
否	创建标签：

A. 右键点击标签并选择New *"tag_name"*(新建 *"tag_name"*)(在老版本的软件中，菜单选项是*Create "tag_name"*)

B. 在*Description*(描述)栏中，给标签输入描述(可选)

C. 在*Data Type*(数据类型)栏中，给标签输入数据类型：

如果标签：	类型：
不是一个数组(文件)	*data_type*
一位数组	*data_type[x]*
二维数组	*data_type[x,y]*
三维数组	*data_type[x,y,z]*

其中：

data_type是标签或数组保存的数据类型。参见3-1页表3.2

x 是元素的第一维

y 是元素的第二维

z 是元素的第三维

D. 从*Scope*(作用域)列表中，选择标签的作用域：

如果用户想使用标签：	那么选择：
在工程的多个例程中	*name_of_controller(controller)*
作为一个生产型或消费型标签	
在一个信息中	
只在工程的一个程序中	编程要使用的标签

E. 选择*OK*(完成).

校验例程

在用户编程时，要定时对例程进行校验：

1. 在RSLogix5000窗口的最顶部的工具栏中，点击 

2. 如果发现错误，将罗列在窗口底部：

a. 找到第一个错误或警报处，按*F4*键。

b. 根据结果窗口描述的错误进行改正。

c. 然后重复步骤1。

3. 要关闭结果窗口，按*Alt＋1*键。

访问系统值

监视状态标记

控制器支持在用户逻辑中使用状态关键字对特殊事件进行监测：

- 状态关键字不区分大小写。
- 因为状态标记变化的很快，所以在RSLogix5000软件中没有显示状态标记。(也就是，即使设置了状态标记，程序指令对此标志也不显示。)
- 用户不能对状态关键字定义别名标签。

用户可用下列关键字：

确定：	使用:
用户保存的数值与目标值不匹配，因为： • 大于目标值的最大值 • 小于目标值的最小值 重要提示：每次S:V都清除它的设置，产生一个次要故障(类型4，代码4)	S:V
指令的目标值为0	S:Z
指令的目标值为负数	S:N
在算术运算中产生进位或借位时超出数据类型的范围 例如： • 加法3 +9进位为1 • 减法25-18借位为10	S:C
在当前程序中第一次正常扫描例程时的状态字	S:FS
至少产生一个次要故障： • 在程序执行过程中产生次要故障时控制器将此位置1。 • 与程序执行无关的次要故障，例如电池电压低，控制器不将此位置1。	S:MINOR

获取和设置系统数据

控制器将系统数据保存在对象中。与PLC-5的控制器一样没有状态文件。通过GSV/SSV指令来获取或设置保存在对象中的系统数据：

- GSV指令可获取指定的信息并将其放在目标文件中。
- SSV指令是用数据源的数据给系统设定指定属性。

注　意

在使用SSV指令时一定要小心。系统数据的改变可能引起控制器意外的操作或人身伤害。

要获取或设置系统值：

1. 打开RSLogix5000的工程
2. 从*Help*(帮助)菜单中选择*Contents*（内容)。
3. 点击*Index*（检索)表。
4. 输入*gsv/ssv*并点击*Display*（显示)。

5．点击需要查找的对象：

要获取或设置：	点击：
伺服轴	AXIS
系统过载时间	CONTROLLER
控制器硬件	CONTROLLERDEVICE
在一个框架上设备的协调系统时间	CST
串口的DF1通讯驱动程序	DF1
控制器的故障记录	FAULTLOG
信息指令的属性	MESSAGE
模块的状态、故障和模式	MODULE
运动轴组	MOTIONGROUP
程序的故障信息和扫描时间	PROGRAM
程序实例	ROUTINE
串口组态	SERIALPORT
任务的共用时间或属性	TASK
控制器的时钟	WALLCLOCKTIME

6. 在对象的属性列表中，确认用户想要访问的属性。

7．为属性值创建一个标签：

如果属性的数据类型是：	那么：
一个元素(例如：DINT)	为属性创建一个标签
多个元素(例如：DINT[7])	A. 创建一个自定义数据类型与属性所用到数据匹配。 B. 用步骤A的数据类型来创建一个标签。

8．在用户梯形图逻辑中，输入适当的指令：

如果要：	输入的指令：
获取属性的值	GSV
设置属性的值	SSV

9. 给指令指定一个必要的操作对象：

对下列操作对象：	选择：
类别名	对象名
注解名	一些特殊对象名(例如：一些必需的I/O模块，任务，信息) • 并非所有的对象都需要这个条目。 • 要定义当前的任务，程序，选择THIS.
属性名	属性的名字
目标(GSV)	存储获取数据的标签： • 如果标签是一个自定义的数据类型或一个数组，选择第一个成员或元素。
源(SSV)	存储设置数据的标签： • 如果标签是一个自定义的数据类型或一个数组，选择第一个成员或元素。

下面是一个获取系统当前日期和时间的例子。

例　子　获取一个系统值

在程序第一次扫描时，从WALLCLOKTIME对象中获取*DateTime*并将其存储在*wall_clock*标签中，这个标签必须是自定义数据类型。

要获取更多的信息，查阅*Logix5000 Controllers General Instruction Set Reference Manual*《Logix5000控制器指令集参考手册》，出版号1756-RM003。

指定别名

别名标签

别名标签允许用户定义一个标签去代表另一个标签。

- 两个标签共用相同的值。
- 当其中一个标签的值改变时，另外一个标签也随之改变。

下列情况下使用别名标签：

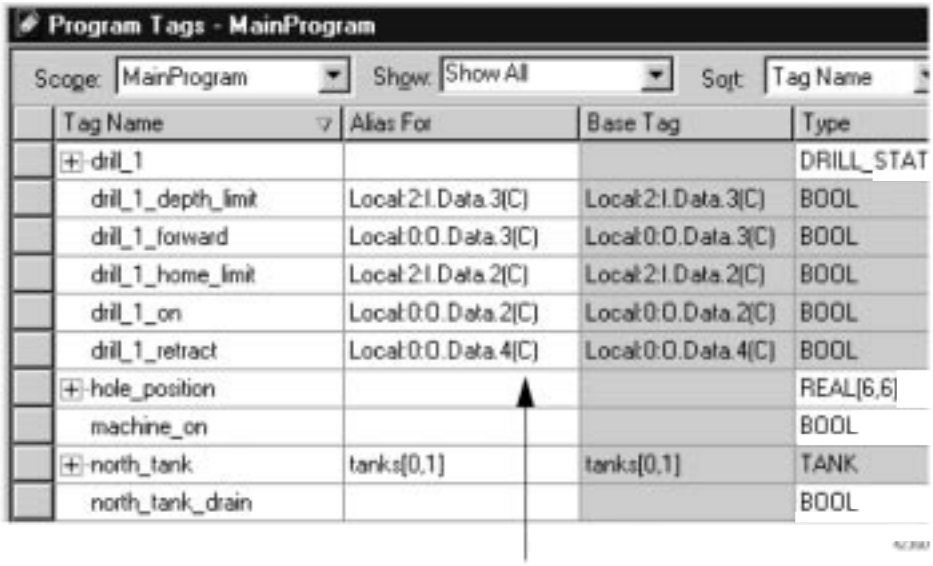
- 程序逻辑超出编程界面
- 给一个I/O设备指定一个描述名
- 给复杂的标签提供一个简单的名字
- 为数组的一个元素提供一个描述名

标签窗口显示别名标签的信息：

*drill_1_depth_limit*是*Local:2:1.Data.3*的别名(一个数字量输入点)。当输入导通时，别名标签也导通。

*drill_1_on*是*Local:0:0 Data.2*的别名标签(一个数字量输出点)。当别名标签导通时，输出也导通。

*north_tank*是*tanks[0,1]*的别名标签



(C)表示标签属于控制器作用域

通常在有接线图之前用别名标签进行编程：

1. 对每一个I/O设备，在创建标签时都要给设备起一个描述性的名字，例如给传动电机起名为conveyor。
2. 在编程时用标签的描述名。(用户甚至不用连接I/O设备就可以对梯形逻辑进行测试。)
3. 接下来，接线图建好后，将I/O模块添加到控制器的I/O组态中。
4. 最后，将标签的描述名转换为代表实际I/O点或通道的别名标签。

下面就是用标签的描述名编写的最初梯形图逻辑，例如*stop*和*conveyor_on*。接下来就是将其转换成与I/O设备相对应的别名标签。

*stop*是*Local:2:I.Data.1*的别名标签(操作面板上的停止 按钮)

*conveyor_on*是*Local:0:O.Data.0*的别名标签(传动电机的启动开关)

显示别名信息

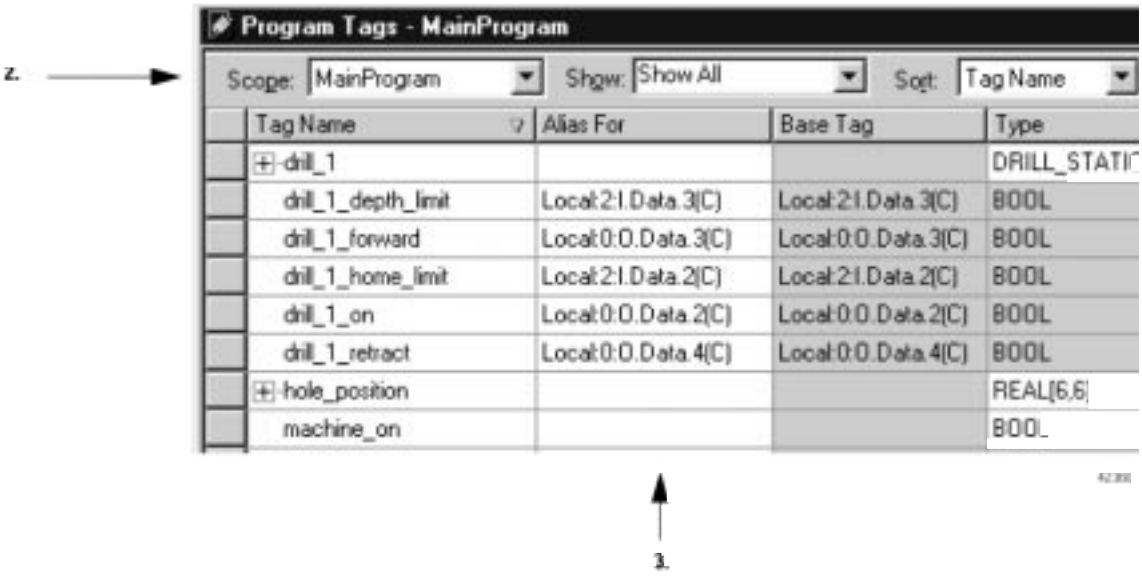
要显示(用户逻辑中)标签对应的别名点：

1. 从*Tools*(工具)菜单中选择*Options*(选项)。
2. 点击*Ladder Display*(梯级显示)表格
3. 选择*Show Tag Alias Information*(显示标签别名信息)复选框
4. 点击*OK*(完成)。

指定一个别名标签

要指定一个标签给其它标签作**别名标签**：

1. 从*Logic*(逻辑)菜单中选择*Edit Tags*(编辑标签)。



2. 选择标签作用域。

3. 在标签名的右边点击别名单元。这个单元将显示一个▼。

4. 点击这个▼。

5. 选择这个别名标签将要代表的:

是:	操作如下:
选择一个标签	双击标签名
选择一个位文件	A. 点击标签名 B. 在标签名的右边，点击▼ C. 点击所需要的位。

6. 按*Enter*(回车)键或点击其它单元。

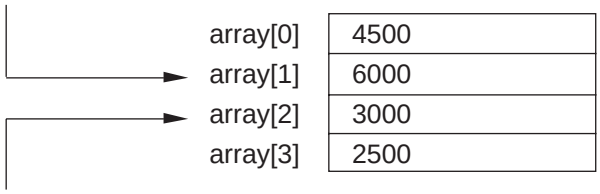
注 意:

指定一个间接地址

何时需要指定一个间接地址

如果用户想用一条指令去访问一个数组的不同元素，那么就用数组的下标中的一个标签(一个间接地址)。当改变标签值时就会改变梯形图逻辑所指的数组元素。

当i*ndex*等于1时，*array[index]*就指向这里



当*index*等于2时，*array[index]*就指向这里

下表简述了间接地址的一般用法：

如果要：	用下标中的一个标签并：
从配方数组里选择一个配方	在标签中输入配方号
从一个可能设好的数组中做一个特定机器的设置	在标签中输入要求的设置
从数组中读取参数或状态，每次读一个元素	A.执行完第一个元素对应的操作。
记录错误代码	B.用一条ADD指令将标签值加1，这样就指向数组的下一个元素。
对一个数组元素执行完几个动作操作再指向下一个元素	

下面是一个将一系列预置值装入计时器的例子，每次装入一个值(数组元素)。

例　子

通过一个数组实现的步骤：

*timer_presets*数组下一梯级的计时器存储了一系列预置值。*north_tank.step*标签指向所用数组元素。例如：当north_tank.step等于0时，指令就读取timer_presets[0]的值(60000ms)送给计时器。

当*north_tank.step_time*计时完成，梯级就使*north_tank.step*加1并指向下一个数，*timer_presets*数组元素就装入计时器中。

当*north_tank.step*执行完数组的值，梯级将复位标签，从数组的第一个元素重新计时。(数组包含的元素为0到3)。

表达式

用户也可以用一个表达式来定义一个数组的下标。

- 表达式可采用一些操作，例如＋或—去计算一个值。
- 控制器计算出表达式的结果并用这个值作为数组的下标。

用户可用以下操作来定义数组的下标：

操作：	描述：
＋	加
-	减/求反
•	乘
/	除
ABS	绝对值
AND	与
FRD	BCD码转换成整数

操作：	描述：
MOD	留余数
NOT	非
OR	或
SQR	平方根
TOD	整数转换成BCD码
TRN	舍位
XOR	异或

下面是一些表达式的格式：

如果操作需要：	使用格式：	例子：
一个值(标签或表达式)	*operator(value)*	ABS*(tag_a)*
两个值(标签，常数或表达式)	*value_a operator value_b*	*tag_b+5* *tag_c AND tag_d* *(tag_e**2) MOD (tag_f/tag_g)*

注 意：

缓存I/O

何时使用缓存I/O

缓存I/O是一项梯形图逻辑不直接对实际I/O设备标签进行取数或操作的技术。相反，梯形图逻辑使用的是复制的I/O数据。下列情况下采用缓存I/O：

- 防止在程序执行过程中输入或输出值发生变化。(I/O数据刷新与程序执行不同步)。
- 将输入或输出标签复制到数据结构体的一个成员或数组的一个元素中。

缓存I/O

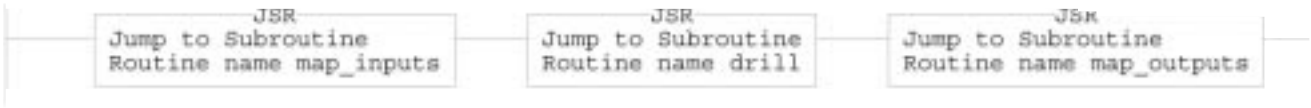
要想缓存I/O数据，执行以下操作：

1. 在执行函数所在的梯级之前，将所需的数据从输入标签复制或转移到与之相对应的缓存标签中。
2. 在函数执行时访问缓存标签。
3. 在执行完函数所在的梯级之后，将数据从缓存标签中复制到与之相对应的输出标签中，

下面是一个将钻孔机的输入和输出数据复制到一个数据结构体的标签中。

例　子　缓存I/O

程序的主例程按下列顺序调用子程序。



*map_inputs*例程是将输入设备的值复制到与之对应的标签中，这些标签将在*drill*例程中用到。

*drill*例程执行的是钻孔机的运行逻辑。

*map_outputs*例程是将*drill*例程中输出标签的数据复制到与之相对应的输出设备中。

下面的例子是一个用CPS指令去复制一组DeviceNet网络上输入设备的数据。

例 子 缓存I/O

*Local:0:I.Data*保存的是DeviceNet网络上的输入数据，这些数据是控制器通过0号槽上的1756-DNB模块获得的。要同时应用这些输入数据，CPS指令将其复制到*input_buffer*中。

- 在CPS指令复制数据时，I/O数据不能刷新。
- 在应用过程中，输入数据可从*input_buffer*中获得。

注 意：

测试一个工程

要测试一个工程，必须完成以下操作：

- 组态一个通讯驱动程序
- 下载一个工程到控制器中
- 给控制器选择一个模式
- 校验主要故障
- 保存用户的在线修改

另外，用户可以执行下列操作：

- 使用程序控制指令停止执行一些特殊例程或梯级。查阅*Logix5000 Controllers General Instruction Set Reference Manual*《Logix5000控制器指令集参考手册》，出版号1756-RM003。
- 强置输入或输出。参见14-1页的 ″强置值″。

组态一个通讯驱动程序

RSLogix5000软件必须通过通讯驱动程序才能与控制器进行通讯。用户使用RSLinx软件来组态通讯驱动程序：

1. 启动RSLinx™软件。
2. 从*Communication*(通讯)菜单中选择*Configure Drivers*(组态驱动程序)。
3. 从已有的驱动程序类型下拉菜单中，选择一个驱动程序：

网络类型：	驱动程序选择：	
	台式机	笔记本电脑
串行	RS-232DF1驱动程序	RS-232DF1驱动程序
DH +™	1784-KT/KTX(D)/PKTX(D)	784-PCMK
ControlNet™	1784-KTC(X)	1784-PCC
Ethernet™	Ethernet驱动程序	Ethernet驱动程序
DeviceNet	DeviceNet驱动程序 (1784-PCD/PCIDS, 1770-KFD SDNPT驱动程序)	DeviceNet驱动程序 (1784-PCD/PCIDS, 1770-KFD SDNPT驱动程序)

4．点击*Add New(新建)*。

5．如果用户想指定驱动程序的描述名，修改其缺省名即可。

6．选择*OK*(完成).

7．组态驱动程序：

驱动程序类型：	操作如下：
串行	A.从Comm Port(通讯端口)的下拉列表中，选择驱动程序将要使用的串口。 B.从Device(设备)下拉列表中，选择*Logix5550-Serial Port.* C.点击*Auto-Configure* (自动组态).
ControlNet	A.在*Station Name*(站点名)复选框中，输入一个名以便在RSWho窗口中识别计算机。 .选择中断值，内存地址，I/O背板地址。 C.在*Net Address* (网络地址)复选框中，输入用户分配给计算机的ControlNet节点。
DH＋	A.从*Value* (参数值)下拉列表中，选择驱动程序使用的接口卡的类型。 B.在*Property* (属性)列表中选择下一个条目。 C.在*Value* (参数值)框中，输入或选择适当的值。 D.重复步骤B和C，设置剩下的属性值。
Ethernet	用户想跟Ethernet上的每一个设备通讯(例如：1756-ENET模块或PLC-5E控制器)，必须添加映射入口： A.在*Host Name*(主机名)栏中，输入Ethernet设备的IP地址或主机名。 B.用户是否想跟网上其它设备通讯？

如果：	那么：
是	1.选择*Add New* (新建)
否	2.返回步骤A 进入下一步

3．点击*OK*(完成)。

4．点击*Close*(关闭)。

下载一个工程到控制器

使用这一步将工程下载到一个控制器中，这样用户就可以执行程序逻辑。

- 如果当前控制器中有工程和数据，在用户下载一个新的工程到控制器时，这些数据将丢失。
- 如果控制器的版本与工程应用程序的版本不匹配，用户必须更新控制器的固件。RSLogix5000软件允许用户作为下载顺序的一部分更新控制器的固件。

注 意

在用户下载一个工程或更新固件时，所有的伺服轴必须停止运行。在下载一个工程或更新固件之前，用户一定要确认伺服轴不会产生任何运动。

重 要 提 示

要更新控制器的固件，首先必须安装固件的升级软件包。

- 升级软件包在RSLogix5000软件附带的光盘里。
- 要下载一个升级软件包，用户可以访问www.ab.com。选择*Product Support*(产品支持), 选择*Firmware Updates*(固件更新)。

1. 打开想要下载的RSLogix5000的工程。
2. 从通讯菜单中，选择*Who Active*。
3. 在网络上找到控制器。

要访问网络的某一层，操作如下：

- 双击网络名
- 选择网络名并按 键
- 点击＋符号

4. 选择控制器
5. 选择*Download* (下载)。

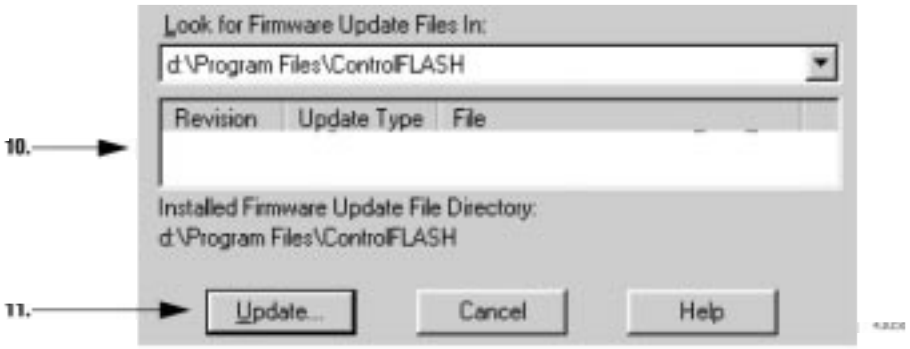
6. 软件的响应：

如果软件显示：	那么：
下载到控制器	进行步骤7
下载失败。离线状态下工程程序的版本与控制器的固件不匹配	进行步骤9

7. 选择*Download*(下载)。
   工程下载到控制器并将RSLogix5000软件上线。

8. 省略以下步骤。

9. 选择*Updata Firmware* (更新固件)。



10. 为控制器选择所需的版本。

11. 选择*Updata* (更新)。
    有一个对话框将提示是否确认更新。

12. 要更新控制器的固件，选择*Yes* (是)。

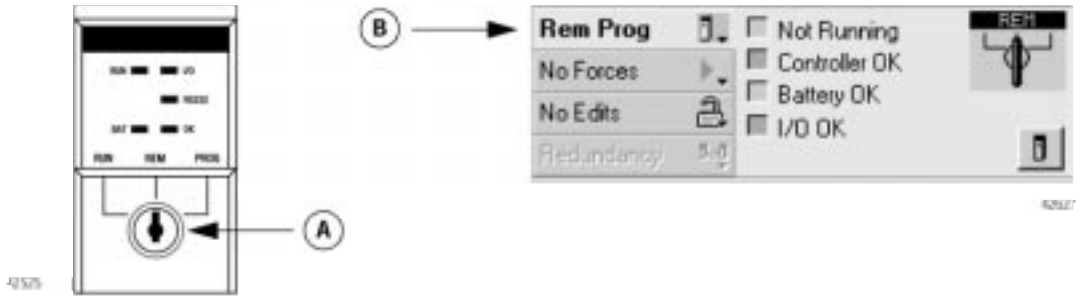
将出现以下情况：

- 控制器的固件被更新。
- 工程下载到控制器中。
- RSLogix5000软件上线。

选择控制器模式

工作模式

要测试一个工程，就得给控制器选择一个模式：

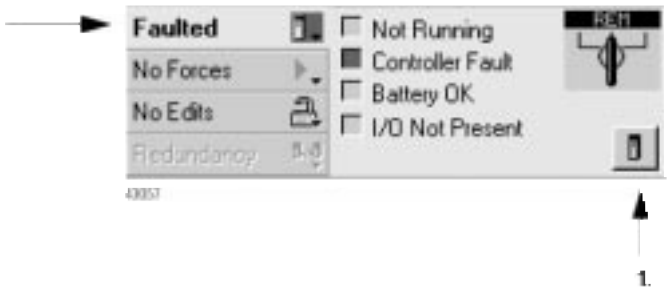


如果用户想：	那么选择这些模式：					
	运行	编程	远程			← 钥匙开关选择 (A)
			运行	编程	测试	← RSLogix5000选择 (B)
通过工程的逻辑输出	✓		✓			
数据送给状态命令值						
在编程模式下把输出数		✓		✓	✓	
据给组态状态值						
执行(扫描)任务	✓		✓		✓	
通过编程软件改变控制器的模式			✓	✓	✓	
下载一个工程		✓	✓	✓	✓	
规划ControlNet网络		✓		✓		
在线编辑		✓	✓	✓	✓	
发送信息	✓		✓		✓	
发送和接收数据来响应从其它控制器发送的信息	✓	✓	✓	✓	✓	
生产型和消费型标签	✓	✓	✓	✓	✓	

校验主要故障

如果控制器输入**错误的模式**，将会出现一个**主要故障**并且停止执行梯形图逻辑。

控制器处于出错状态，出现一个主要故障，控制器将不再执行梯形图逻辑。



校验一个故障：

1. 点击 按钮。

2. 使用*Recent faults list*（最近故障列表）中的提示信息去校验主要故障。参见A-1页的 ″主要故障代码″。

3. 点击*Clear Majors*（清除主要故障）按钮。

提 示

用户也可以用控制器上的钥匙开关来清除主要故障。将钥匙开关打到*Prog*（编程）状态，再打到*Run*（运行）状态，然后再打到*Prog*（编程）状态。

保存用户的在线修改

如果用户在线修改工程，必须保存，这样才能保证离线状态和在线状态下工程文件一致：

如果用户想：	操作如下：
保存在线修改和数据值	从*File*（文件）菜单中选择*Save*（保存）。
保存在线修改但不保存在线的数据值	A. 从*Communications*（通讯）菜单中选择*Go Offine*（离线）。 B. 从*Fiile*（文件）菜单中选择*Save*（保存）。

与其它控制器通讯

使用本章

使用本章用于在控制器之间传输数据(发送或接收数据)。
用户可用下列的任一种方式来传输数据：

- 生产和接收一个标签
- 发送一个信息

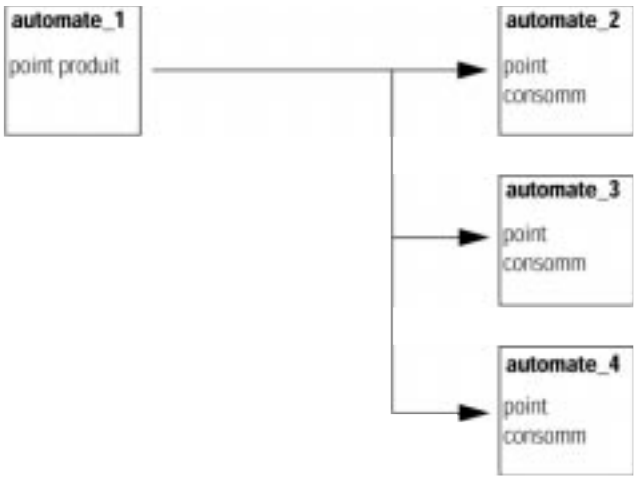
如何使用本章

选择一种在控制器之间传输数据的方式：

如果数据：	那么：	页码：
需要按一定的频率传输(也就是确定性)	生产和接收一个标签	10-1
应用中出现特殊情况时发送	发送一个信息	10-11
从多个控制器获得(没有消费型标签选项，或不想使用消费型标签)	发送一个信息到多个控制器	10-13

生产和接收一个标签

不需要用梯形图逻辑，**生产型标签**就可以将数据发送到一个或多个**消费型标签**(消费者)中。



在下列控制器和网络中用户可以使用生产型和消费型标签。

控制器：可在下列网络上生产和接收标签：

	背板	ControlNet	Ethernet
SLC500		✓	
PLC-5		✓	
ControlLogix	✓	✓	✓
FlexLogix		✓	
SoftLogix		✓	

生产型和消费型标签的应用如下：

- 在控制器之间用一个**连接**来传输数据：
  - 多个控制器可以消费(接收)数据。
  - 数据更新时间由消费型标签的**请求数据包时间间隔**(RPI)决定。
- 每一个生产型或消费型标签用到下列连接数：

每一个：	**用到连接数：**
生产型标签	*number_of_consumers+1*
消费型标签	1

例 子

生产型或消费型标签使用的连接：

- 为5个控制器(消费者)生产一个标签要用6个连接(5个消费者＋1＝6)
- 为1个控制器生产4个标签要用8个连接：
  - 每一个标签用2个连接
    (1个消费者＋1＝2)
  - 每个标签2个连接×4个标签＝8个连接
- 从一个控制器接收4个标签要用4个连接(每个标签1个连接×4个标签＝4个连接)。

准备工作

要同其它控制器共享数据，执行下列操作：

- 为生产型或消费型数据组织标签
- 生产一个标签
- 接收一个生产型标签

根据不同的用户系统，用户必须执行以下操作：

- 为PLC-5C控制器生产整数型标签
- 为PLC-5C控制器生产实数型标签
- 从PLC-5C控制器接收一个整数型标签
- 带宽限制调节

为生产型或消费型数据组织标签

用户创建生产或接收数据(共享数据)的标签，要遵循以下规则：

1. 在**控制器作用**域创建标签。用户只能共享控制器作用域的标签。
2. 用其中的一个数据类型：
   - DINT
   - REAL
   - DINT数组或REAL数组
   - 用户自定义
3. 要共享一个除了规则2中列出的数据类型的数据，就得创建一个自定义数据类型。
4. 生产型和消费型标签的数据类型必须一致。
5. 要与PLC-5C控制器共享标签，就得使用一个自定义数据类型。参见下列内容：
   - 为PLC-5C控制器生产一个整型标签，10-6
   - 为PLC-5C控制器生产一个实型标签，10-7
   - 从PLC-5C控制器接收一个整型标签，10-9

6．标签应小于或等于500字节。如果用户必须传输大于500个字节的数据，可以将数据打包传输，参见11-1页“生产一个大的数组”。

7．如果用户想在ControlNet上生产一个标签，那么标签必须小于500字节。参见10-10页“带宽限制调节”。

8．如果用户需要在同一个控制器上生产多个标签：
- 将数据组合到一个或多个自定义的数据类型中。(这样可以比生产每个标签减少连接)
- 根据相同的刷新频率来组合数据。(要保持网络带宽，就得使用一个大于临界数值的RPI)

例如，用户可以为临界数据创建一个标签，而其它标签不用临界值。

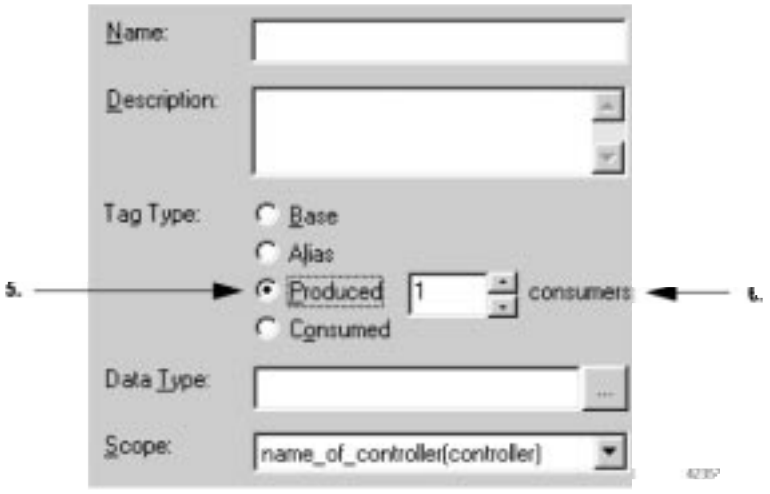
生产一个标签

1. 打开RSLogix5000工程，其中包含用户想要生产的标签。

2. 从*Logic*(逻辑)菜单中选择*Edit Tags*(编辑标签)。

3. 从*Scope*(作用域)中选择*name_of_controller(controller)*.(只有控制器作用域的标签才能生产数据)

4. 选择要生产数据的标签，按*ALT+Enter*键。



5．选择*Produced*(生产型)选项按钮。

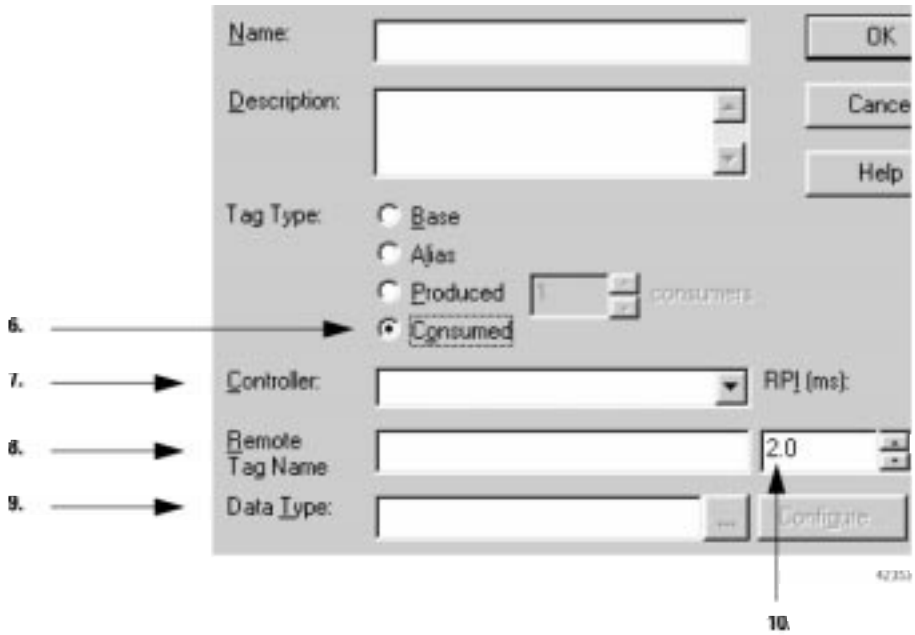
6．输入或选择将要消费(接收)标签的控制器的数量。

7．点击*OK*(完成).

8．在其它控制器组态一个标签来接收这个生产型标签。参见10-5页“接收一个生产型标签”。

接收一个生产型标签

1. 打开一个要接收**生产型标签**的RSLogix5000的工程。
2. 在控制器项目管理器【的*I/O Configuration*(I/O组态)】中添加含生产型标签的控制器。
3. 从*Logic*(逻辑)菜单中选择*Edit Tags*(编辑标签)。
4. 从*Scope*(作用域)中选择*name_of_controller*(controller).(只有控制器作用域的标签才能接收数据)
5. 在这个控制器中选择要接收的生产型标签，按*ALT+Enter*键。



6. 选择*Consumed*(消费型)选项按钮。

7. 选择含有生产型标签的控制器。
8. 输入生产型标签名。
9. 选择与生产型标签相同的数据类型。
10. 输入或选择标签的刷新时间：
    - 在用户程序中允许使用最大值。
    - 如果控制器要在ControlNet上接收一个标签，选择**网络刷新时间(NUT)**的2 n倍时间。

    例如，如果NUT是5ms，则输入5，10，20或40ms等
11. 点击*OK*(完成)。
12. 用户如果在ControlNet上共享标签，需要使用RSNetWork for ControlNet软件对网络进行规划。

重要提示 如果一个消费型标签的连接失败，所有其它正从远程控制器上接收数据的标签将停止接收新的数据。

为PLC-5C控制器生产一个整数型标签

1. 打开RSLogix5000工程
2. 创建一个含多个整型元素数组的自定义数据类型。例如，INT[2]。(必须生产两个或两个以上的整数型标签)。
3. 创建一个生产型标签，选择步骤2中定义的数据类型。
4. 打开RSNetWorx™ for ControlNet软件。
5. 在ControlNet中组态PLC-5C目标控制器：
   a. 插入一个Receive Scheduled Message (接收确定性信息)。
   b. 在Message(信息)大小栏中，输入生产型标签中整型文件的数量。
6. 在RSNetWork for ControlNet软件中对网络进行规划。

为PLC-5C控制器生产实数型标签

1. 打开RSLogix5000工程。

2. 用户需要生产多少个值？

如果要生产：	那么：
一个实数值	创建一个生产型标签并选择REAL数据类型
多个实数值	A. 创建一个含实数型数组的自定义数据类型。
	B. 创建一个生产型标签并选择步骤A中定义的数据类型。

3．打开RSNetWorx for ControlNet软件。

4．在ControlNet中组态PLC-5C目标控制器：

a． 插入一个Receive Scheduled Message (接收确定性信息)。

b． 在Message(信息)大小栏中，输入生产型标签中实型文件数量的二倍的数字。例如，如果生产型标签包含10个实型数据，则输入信息的大小为20。

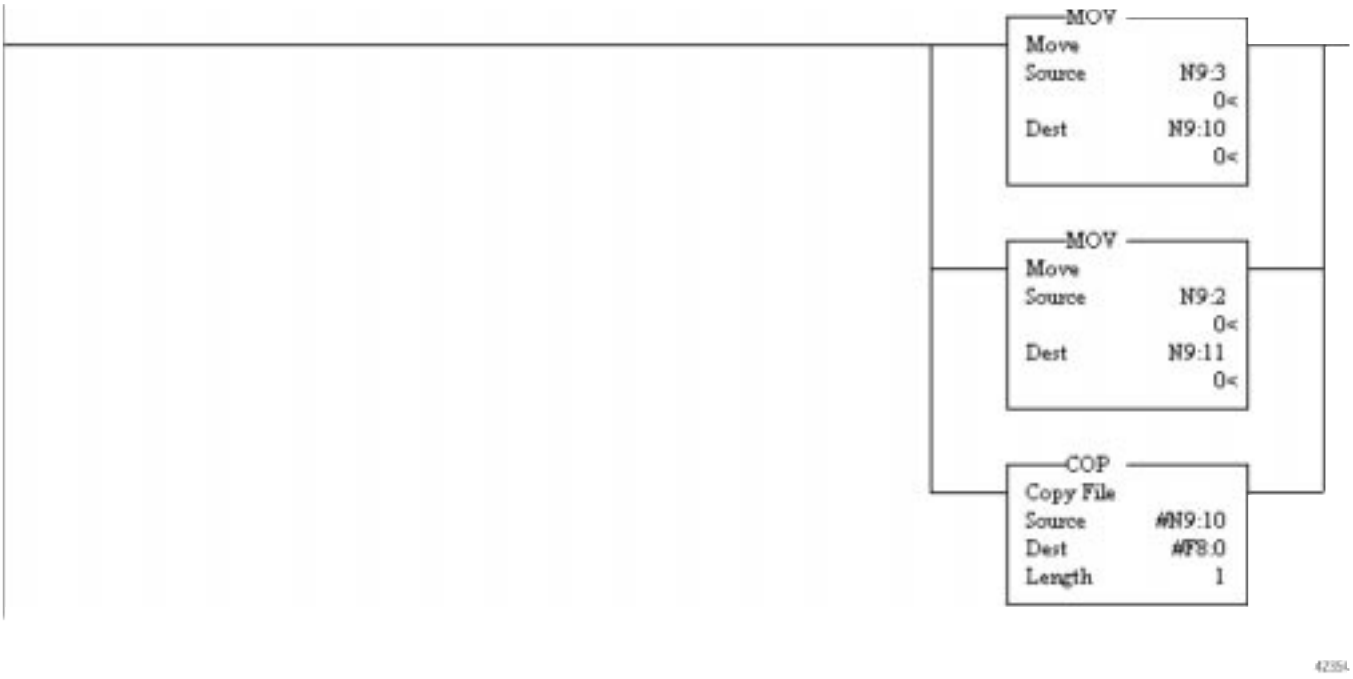
当PLC-5C控制器在接收Logix5000控制器的生产的标签时，将整型数据保存在连续的16位整数文件中。PLC-5C要保存浮点数据，控制器必须留出32位的空间，分配如下：

- 第一个整数文件包含数据的高(最左边)16位。
- 第二个整数文件包含数据的低(最右边)16位。
- 每个浮点数据都以这种形式保存。

5．在PLC-5C控制器中，下面例子描述了如何重新创建一个浮点数据：

例 子： 重新创建一个浮点数值

用两条MOV指令颠倒整数文件的顺序并将其保存在新的整数文件中。因为COP指令的目的地址是浮点数据类型，要占用两个连续的整数文件，总共32位，并且将数据转换成单浮点型的数值。



6．在RSNetWork for ControlNet软件中对网络进行优化。

从PLC-5C控制器接收一个整型数据

1. 打开RSNetWorx for ControlNet软件。

2. 在ControlNet中组态PLC-5C控制器，插入一个Send Scheduled Message (发送确定性信息)。

3. 打开RSLogix5000工程。

4. 在控制器项目管理器的I/O组态中添加PLC-5C控制器。

5. 创建一个自定义数据类型，其中包含下列成员：

数据类型：	**描述：**
DINT	状态
INT[x]，其中"x"是从PLC-5C控制器输出数据的大小。(如果用户只接收一个INT数据，就不需要维数。)	PLC-5C控制器产生的数据

6．用下列属性创建一个消费型标签：

标签属性：	填写或选择：
标签类型	消费型
控制器	生产数据的PLC-5C
远程请求	从PLC-5C控制器的ControlNet组态中选择信息的数量
RPI	选择NUT的2n倍时间。例如，如果NUT是5ms，则输入5，10，20或40ms等
数据类型	在步骤5中创建的自定义数据类型

7．在RSNetWork for ControlNet软件中对网络进行优化。

带宽限制调整

当用户在ControlNet上共享一个标签时，标签必须满足网络带宽。

- 因为ControlNet中连接数量的增加，而一些连接，包括生产型和消费型标签，这些都要占用网络刷新时间(NUT)。
- 因为ControlNet在一个NUT内只能传输500字节，所以每一个连接的数据不能超过500字节才能满足网络的NUT。

根据用户系统大小的不同，用户可能没有足够的网络带宽来传输500字节的标签。如果标签太大，可以进行下列一项或多项调整：

- 减小网络刷新时间(NUT)。更快的NUT可减少连接所占用的刷新时间。
- 增加用户连接的请求数据包时间间隔(RPI)。在大的RPIs下，数据在连接中轮流地发送。
- 在远程框架的ControlNet网桥(CNB)模块中，为框架选择最有效的通讯方式。

框架中大部分模块是否是不带诊断的数字量模块	给远程CNB模块选择下列通讯方式
是	机架最优化
否	无

机架最优化方式给框架的每槽增加了8个字节。模拟量模块发送/接收诊断、安全、时间标记或确定性数据的模块或要求直接连接，而不能采用机架最优化方式。选择“无”，这样就可以将每一槽的8个字节留给它用，例如，生产型和消费型标签。

- 将标签拆分成两个或多个小的标签
  - 根据相同的刷新频率将数据分组。例如，用户可以为临界数据创建一个标签，而其它标签不用临界值。
  - 给每个标签指定不同的RPI。
- 创建梯形图逻辑来传输小的数据单元(包)。参见11-1页“生产一个大的数组”。

发送一条信息

要给一条信息组织用户数据，遵循如下规则：

1. 对于每条信息，创建一个标签来控制：
•在控制器作用域创建标签
•使用MESSAGE数据类型

下面是一个使用MESSAGE数据类型的例子：

例 子： 给另一个Logix5000控制器发送一条信息

当*count_send* 闭合，*count_msg* 发送数据。

2. 为信息将要使用的标签(源或目的标签)创建控制器作用域标签。

3. 在Logix5000控制器中，尽可能为整数定义**DINT**数据类型：
－在Logix5000控制器运行时使用32位整数(DINTs)可以提高运行效率，减小内存。

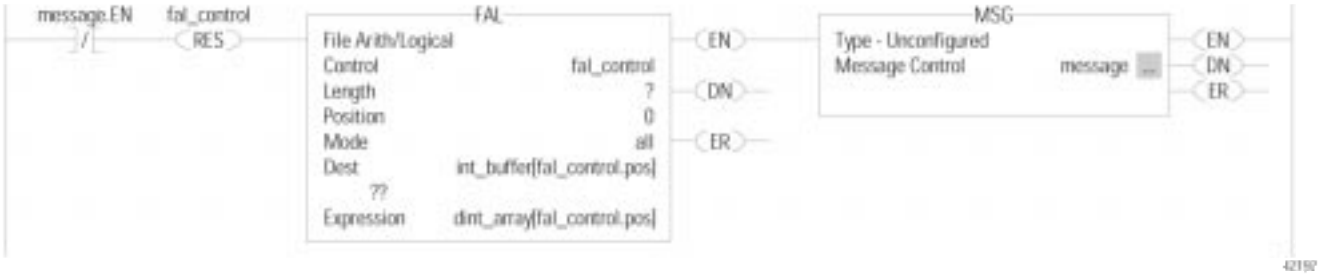
4．如果用户信息是发送给PLC-5®或SLC-500™控制器，或从这两类控制器接收信息，并且传输整型数据(不是实型)，使用INTs缓存器：

a．使用*INT[x]*数据类型为数据创建一个缓存器(控制器作用域)。

其中：
*x*是信息中整数的数量。(只有一个整数，省略*[x]*)
b．在信息中使用缓存器。
c． 使用一条FAL指令从缓存器中调用数据。

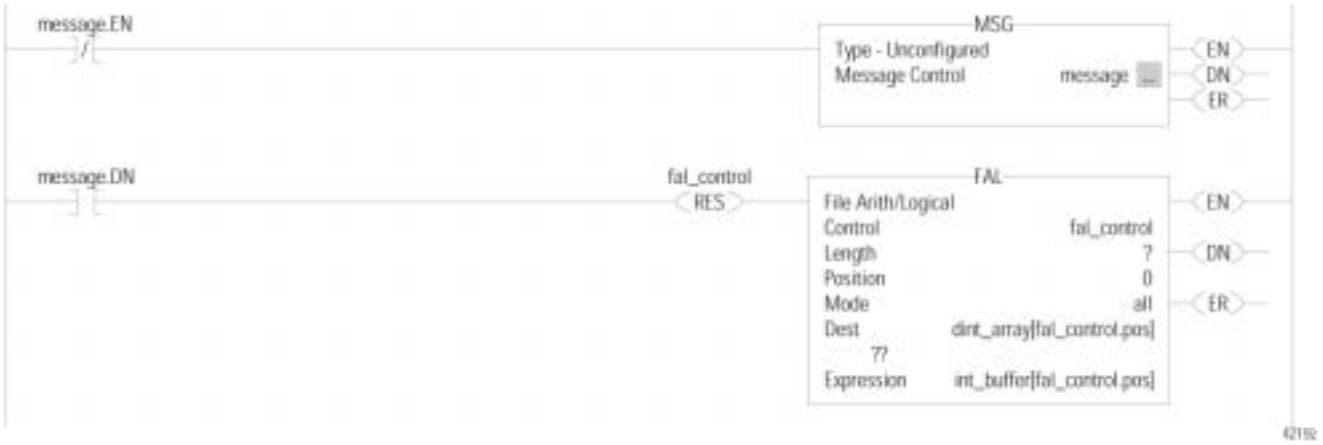
例 子： 给PLC-5控制器写入整数值

连续将数据从*dint_array*送入*int_buffer*。将数据转换成16位整数值(INTs)。然后用信息指令将*int_buffer*中的数据发送给PLC-5控制器。



从PLC-5控制器读取整数值

从PLC-5控制器中连续读取16位整数值(INTs)并将其保存在*int_buffer*中，然后使用FAL指令将数值送给*dint_array*。将这些数值转换成32位整数(DINTs)以提供给工程中其它指令使用。



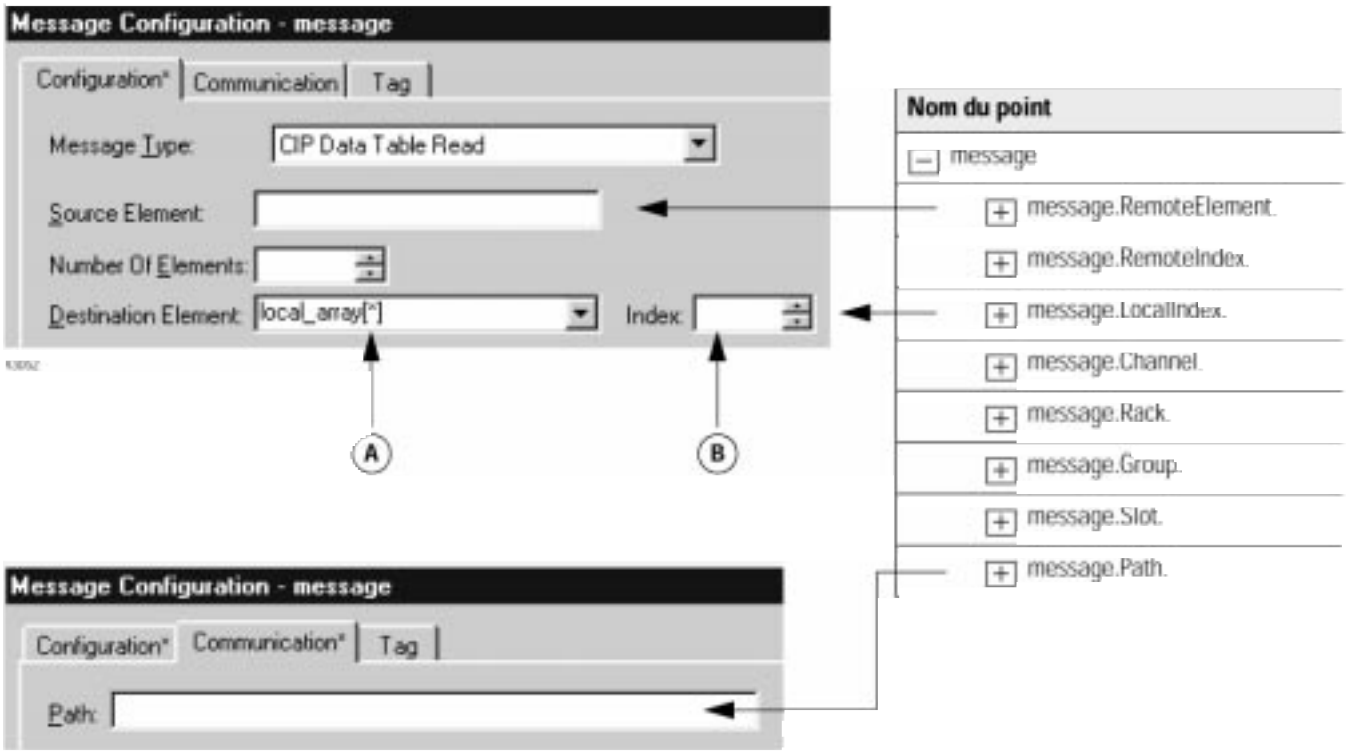
发送一条信息

下面列出了用一条信息指令与多个控制器通讯的步骤。在运行到多个控制器 时要重新组态一条MSG指令，只要给MESSAGE数据类型中的成员写入一个新的值就可以了。

重 要 提 示

在MESSAGE数据类型中，RemoteElement成员保存的是控制器中接收信息的标签名或数据地址。

如果信息：	那么RemoteElement是：
读数据	源元素
写数据	目的元素



A．如果用户给数组元素数量用一星号[*]标记，那么它的值就在 (B) 中显示。

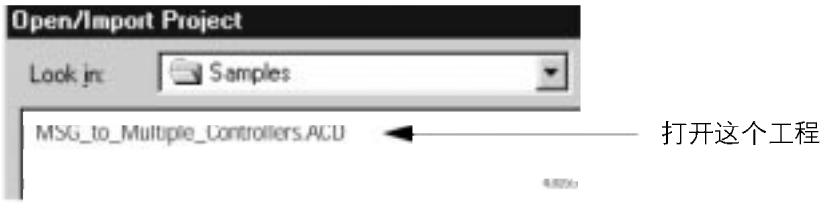
B．只有用户给源元素或目的元素以星号[*]标记时，*Index*(索引)框中的值才有效。指令中以*Index*(索引)框中的值代替星号[*]。

要发送一条信息到多个控制器

- 建立I/O组态
- 定义用户的源和目的元素
- 创建MESSAGE_CONFIGURATION数据类型
- 创建组态数组
- 确认本地数组大小
- 给控制器写入信息属性
- 组态信息
- 对下一个控制器组态
- 重复以上步骤

提 示

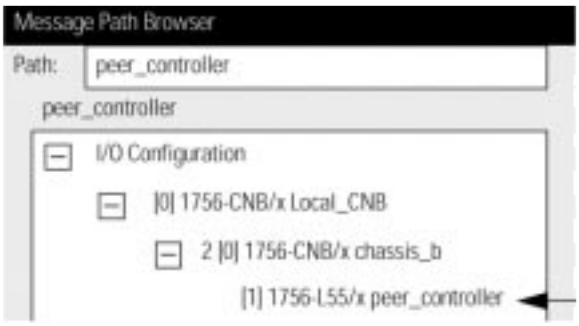
要从工程例子中复制以上内容，打开 *...\RSLogix5000\Projects\Samples*文件夹



建立I/O组态

虽然不要求，但还是建议用户在控制器的I/O组态中添加通讯模块和远程控制器。这样可以很容易地确定每一个远程控制器的路径。

例如，一旦用户添加了本地通讯模块、远程通讯模块和目的控制器，就可以从*Browse* (浏览)按钮中选择目的控制器。



定义用户源和目的元素

在这一步中，读取或写入每个远程控制器的数据，保存在一个数组中。数组的每个元素都对应于不同的远程控制器。

1. 下表来组织本地和远程控制器的标签：

远程控制器名	远程控制器中的标签或数据地址	本地控制器中的标签
		local_array[0]
		local_array[1]
		local_array[2]
		local_array[3]

2. 创建*local_array*标签，用其保存控制器的数据。

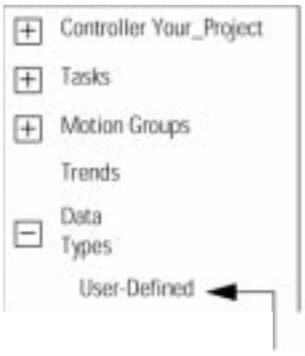
标签名	类型
local_array	*data_type[length]* 其中： *data_type* 是信息发送或接收数据的数据类型，例如DINT，REAL或STRING。 *Length*是本地数组元素的数量。

创建MESSAGE_CONFIGURATION数据类型

在这一步中，用户创建一个自定义数据类型，来保存每一个控制器中信息的组态信息。

- 一些必须的数据类型的成员使用字符串数据类型。
- 缺省的STRING数据类型保存82个字符。
- 如果用户路径或远程标签或地址少于82个字符，用户可选择创建一个新的能保存较少字符的字符串数据类型。这样可以节省内存。
- 要创建一个新的字符串类型，选择*File(文件)New Component(新建组件)Sting Type(字符串类型)...*
- 要创建一个新的字符串类型，用它来代替这一步中的STRING数据类型

创建新的数据类型



点击右键选择*New Data type*

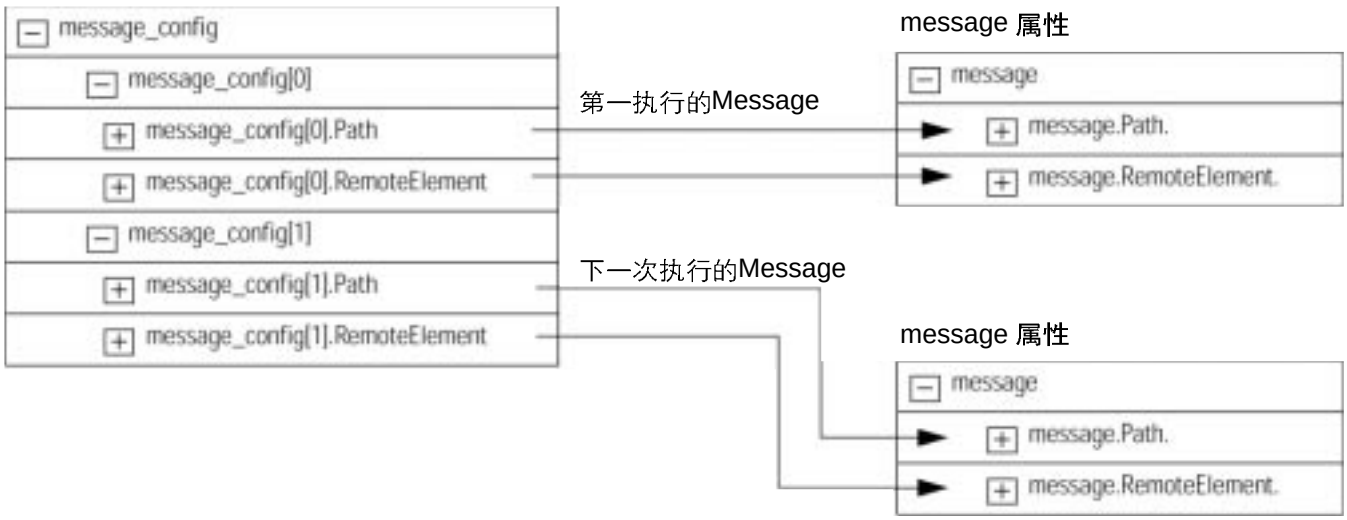
要保存每个控制器的组态信息，请创建以下自定义数据类型。

Type de donn es :MESSAGE_CONFIGURATION			
nom	MESSAGE_CONFIGURATION		
description	Configuration properties for a message to another controller		
Membres			
nom	**type de donn es**	**style**	**description**
Path	STRING		
RemoteElement	STRING		

创建组态数组

在这一步中，用户将每个控制器的组态属性保存在一个数组中。在每条MSG指令执行之前，用户程序指令先读取新的属性，这样将信息发送给不同的控制器。

图10.1将新的组态属性加载到一条MSG指令中



步骤：

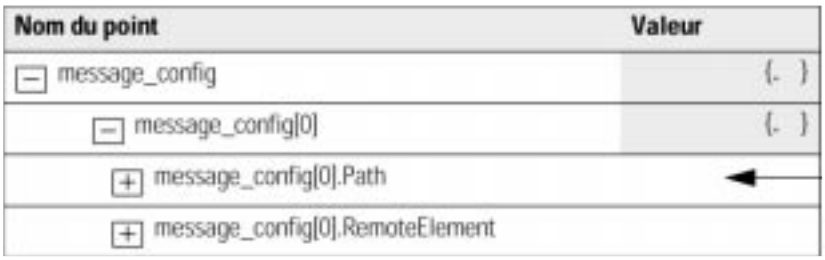
1. 要保存信息的组态属性，创建下列数组：

标签名	类型	作用域
Message_config	MESSAGE_CONGFIGATONG[number]	任意

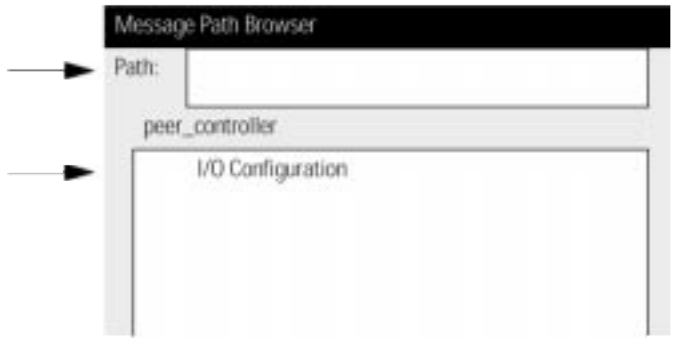
其中：

*number*是信息将要发送的控制器的数量。

2．在*message_config*数组中，输入接收信息的第一个控制器的路径。

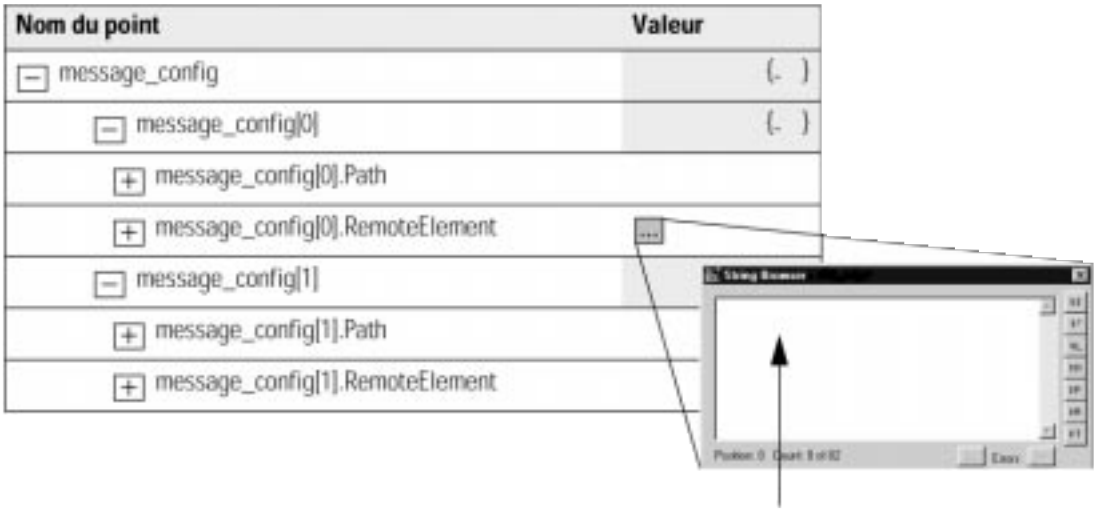


右键点击并选择
Go to Message Path Editor



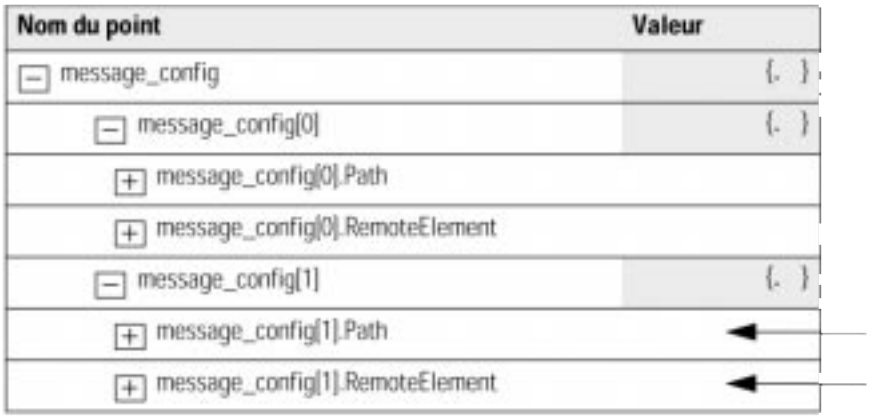
输入远程控制器的路径
或
浏览远程控制器

3．在*message_config*数组中，在接收信息的第一个控制器中输入标签名或数据地址。

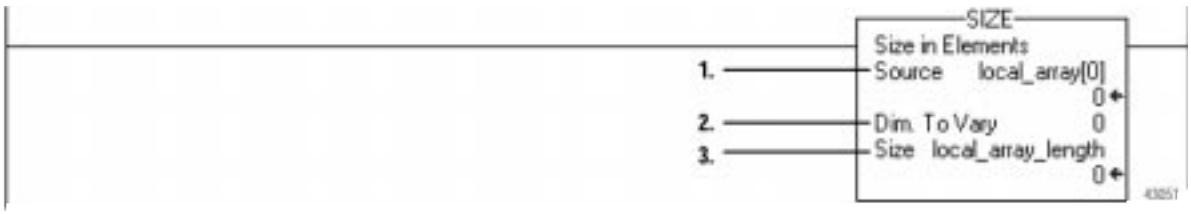


在另一个控制器中输入标签名或数据地址

4．为其它每一个控制器输入路径或远程元素。



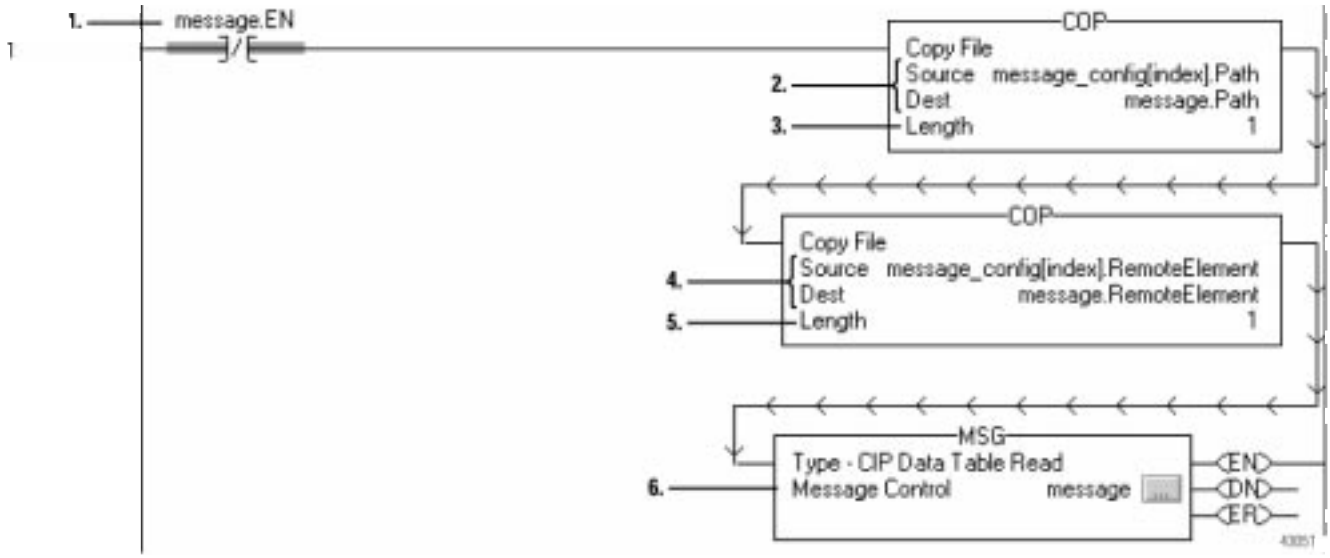
获取本地数组的大小



1. SIZE指令可以计算出*local_array*元素的数量。
2. SIZE指令可以计算出0维数组中元素的数量。如果这样的话，数组只有一维。
3. *Local_array_length*保存*local_array*的大小(元素的数量)。当信息发送到所有的控制器并从第一个控制器重新开始时，这个值将传给下一个梯级。

标签名	类型
local_array_length	DINT

为控制器写入信息属性



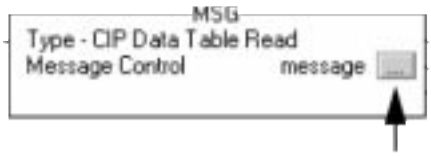
1. 这条XIO指令保持梯级连续发送信息。

标签名	类型	作用域
message	MESSAGE	控制器

2. COP指令将读取信息的路径。Index(索引)框中的值将决定指令读取*message_config*中的哪一个元素。参见10-17页图10.1

标签名	类型	作用域
index	DINT	任意

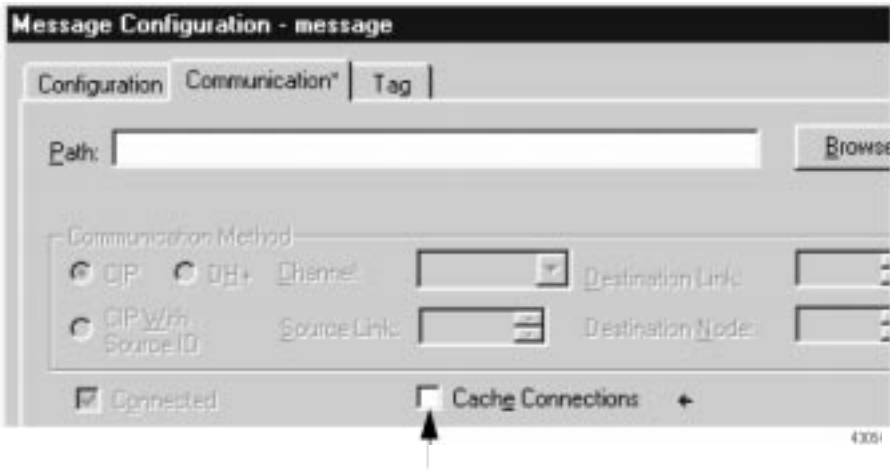
3. 指令从*message_config*中读取一个元素。
4. 在接收信息的控制器中，COP指令将读取标签名或数据地址。*Index*(索引)框中的值将决定指令读取*message_config*中的哪一个元素。参见10-17页图10.1。
5. 指令从*message_config*中读取一个元素。
6. MSG指令。



组态信息

即使用户梯形逻辑控制的是远程信息的元素和路径，用户也必须首先组态信息属性对话框。

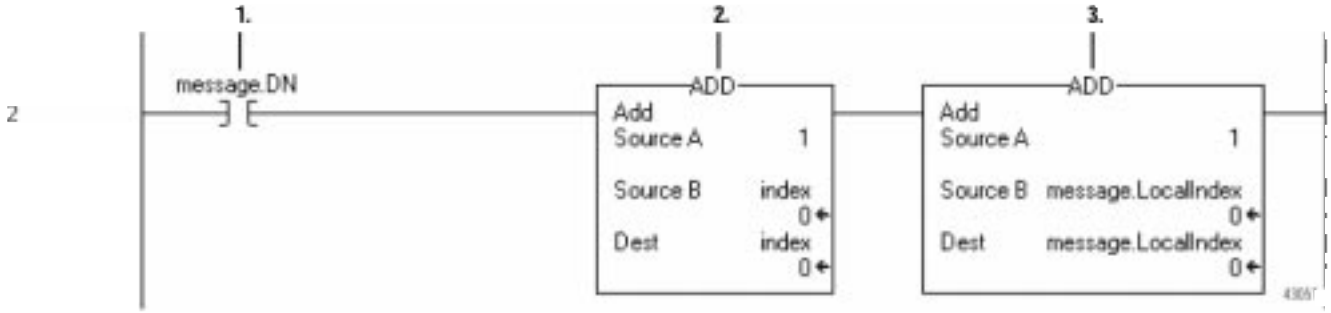
重要提示



清除*Cacbe Connection*选择框

在这项中：	如果用户想：	下列条目：	输入或选择：
组态	从其它控制器读(接收)数据	信息类型	read_type与其它控制器一致
		源元素	在第一个控制器中数据的标签或地址
		元素的数量	1
		目的标签	local_array[*]
		下标	0
	向其它控制器写(发送)数据	信息类型	write_type与其它控制器一致
		源标签	local_array[*]
		下标	0
		元素的数量	1
		目的元素	在第一个控制器中数据的标签或地址
通讯	→	路径	到第一个控制器的路径
		缓冲器连接	清除Cacbe Connection选择框。因为

对下一个控制器组态

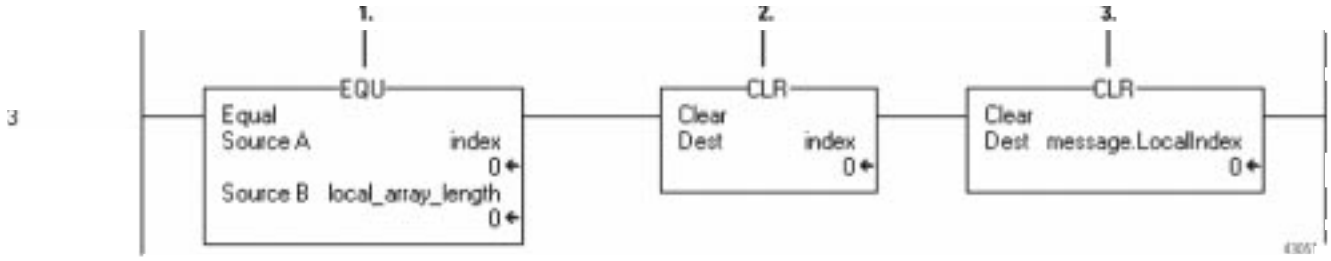


1. 在MSG指令发送信息后……

2. ADD指令将*index*加1。这样就使梯形逻辑将下一个控制器的信息属性读入到MSG指令中。

3. ADD指令将MSG指令中的*LocalIndex*成员加1。这样就使梯形逻辑将下一个控制器的值读入到*local_array*的下一个元素中。

重复以上步骤



1. 当*index*值等于*local_array_length*时，控制器已将信息发送到其它所有控制器。

2. CLR指令将*index*清0。这样就使梯形逻辑将第一个控制器的信息属性读入到MSG指令中并重复以上顺序。

3. CLR指令将MSG指令中的*LocalIndex*成员清0。这样就使梯形逻辑将第一个控制器的值读入到*local_array*的第一个元素中。

生产一个大的数组

使用本章

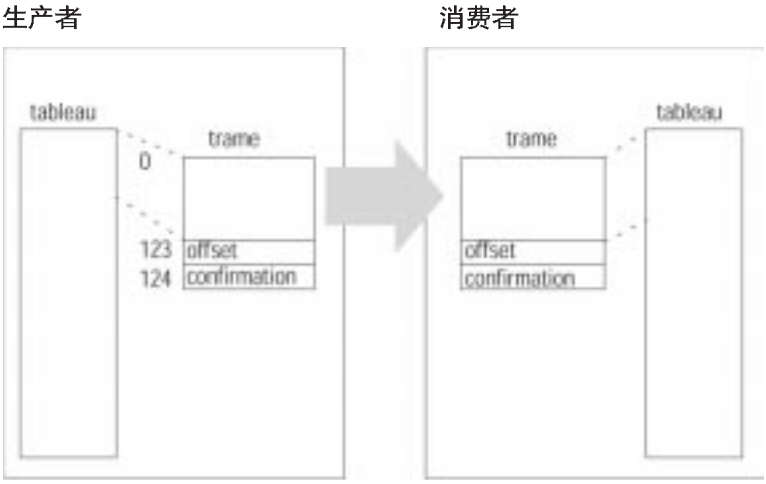
Logix5000控制器可以在单个确定性的连接上发送500个字节的数据。这相当于一个数组的125个DINT或REAL元素。为了传送多于125个DINT或REAL元素构成的数组，用一个含125元素的生产型/消费型标签生成一个数据信息包。这样用户就可以使用这个信息包向其它的控制器逐段地发送数组。

当用户用一个较小的信息包传送一个大的数组时，由于以下原因，用户必须保证信息包在数据移送到目的数组之前已经传送完毕。

- 在ControlLogix背板上的生产型数据是以50个字节为单位来发送的。
- 数据的传输与程序的扫描是异步的。

在数据移送到目的数组之前，数据段用确认字来确认每个信息包内是否含有新的数据，这个逻辑同样也使用一个偏移量来指出数组中信息包的起始元素。

由于使用偏移量和应答元素，每个信息包从数组中携带123个数据元素，如下所述：



另外，这个数组必须含有额外的122个元素。换句话说，它必须比用户想要传送的最大元素数目多122个元素：

- 这些元素起到缓冲器的作用。
- 因为每个信息包含有相同数目的元素，缓冲器用来避免控制器的复制超出数组的范围。
- 如果没有缓冲器，这将会出现最后一个信息包比含有123个元素的真实数据少的情况。

生产一个大数组

1. 打开一个想要生产大数组的RSLogix 5000 工程。

2. 在控制器标签文件夹中，创建以下标签：

P	标签名	类型
	array_ ack	*DINT [2]*
✓	*array_ packet*	*DINT [125]*

其中：

*array*是用户要发送数据的名字。

3. 将*array_ ack* 转换成一个消费型标签：

对于：	规定：
控制器	接收信息包的控制器的名字
远程标签名	*array_ ack* 两个控制器使用同样的名字来共享数据。

参见10-5页的 ″接收一个生产型标签″。

4、 无论在控制器标签文件夹还是在程序标签文件夹中都将包含传送数据的逻辑，创建以下标签：

标签名	类型
array	DINT[x]其中，x等于要传送的元素数加122
*array*_offset	DINT(双整型)
*array*_size	DINT(双整型)
*array*_transfer_time	DINT(双整型)
*array*_transfer_time_max	DINT(双整型)
*array*_transfer_timer	TIMER(计时器)

其中：
*array*是用户发送的数据的名称。

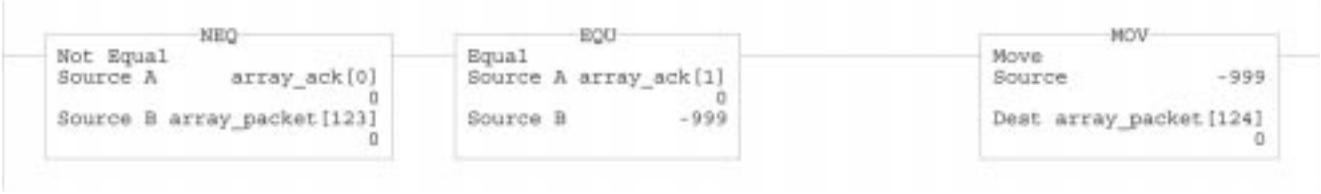
5、 在*array_size*标签中，输入实际数据元素的数目。(这个值由步骤4中的x减去122个缓冲器元素得到。)

6、为将要创建数据信息包的梯形图逻辑创建或打开一个例程。

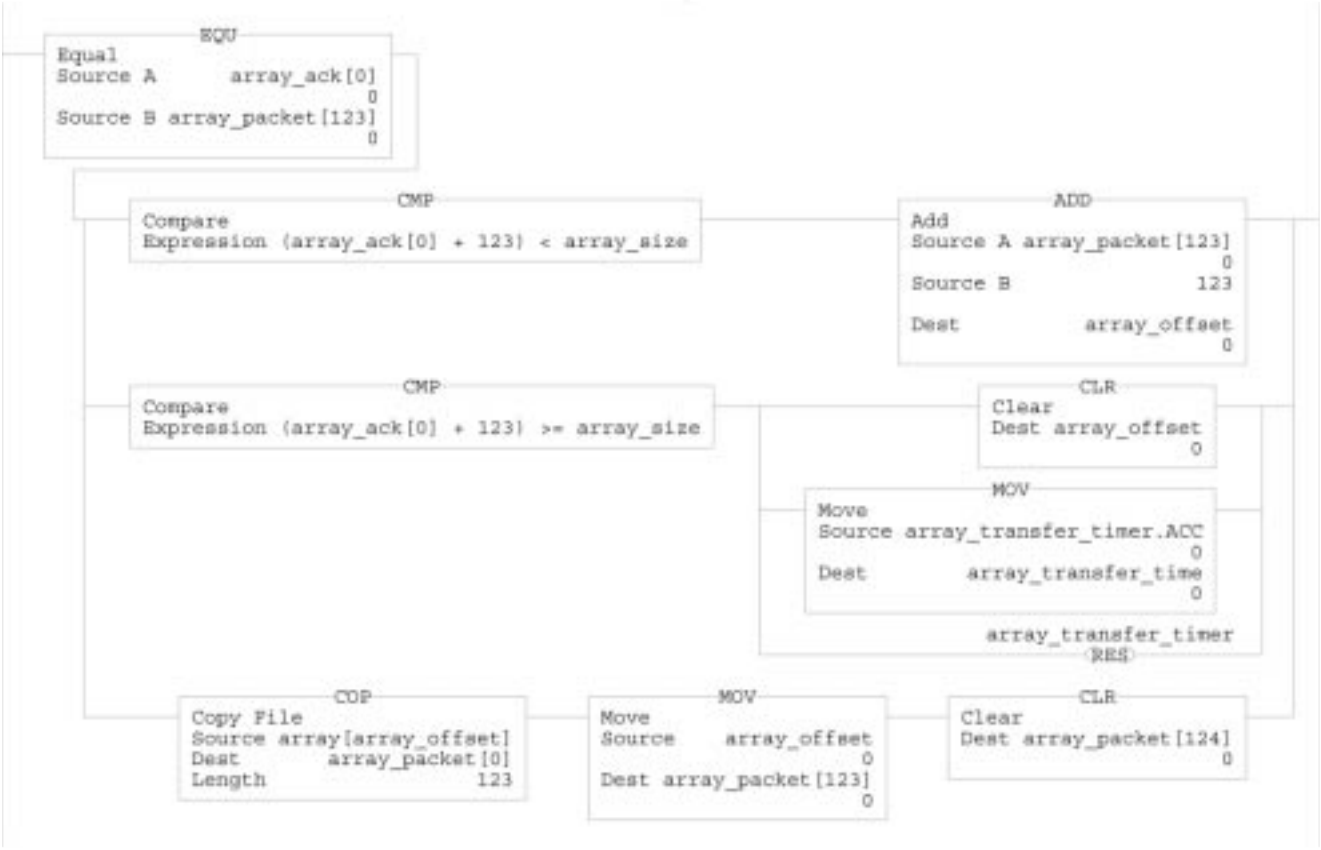
7、输入以下的逻辑：

发送整个数组需要的时间

当*array_ ack [0]*中偏移量的值不等于当前偏移量的值，而*array_ ack [1]*等于-999时，消费者已经开始接收一个新的信息包，这样梯级就将-999送到数据包最后一个元素。这个消费者直到它收到-999的值之后才开始将信息包复制到数组中。这就保证了消费者有新的数据。



当*array_ ack [0]*中保存的偏移量的值等于当前偏移量的值时，消费者已经将信息包复制到数组中；这时梯级检查是否有更多的数据需要传送。如果偏移量的值加上123还小于数组的大小，说明有更多的数据需要传送，这时梯级将偏移量的值增加123。否则，不再有数据传送；这时梯级将重置这个偏移量的值，记录传送时间，并复位计数器。无论哪种情况，梯级都会用新的偏移量值来创建一个新的数据信息包，将新的偏移量值附加给这个信息包，并且清除信息包中的应答元素(*packet[124]*)。



如果当前的传输时间大于传输时间的最大值，更新传输时间的最大值。保存传输数据所需最长时间的记录。

8. 打开将要接收数组的RSLogix 5000工程。

9. 在控制器标签文件夹中，创建下面的标签：

P	标签名	类型
✓	*array_ ack*	DINT[2]
	array _packet	DINT[125]

其中：
*array*是用户发送数据的名称。与生产控制器中的名字相同{步骤2}。

10. 将*array _packet*转换成消费型标签：

对于：	规定：
控制器	发送信息包的控制器的名字
远程标签名	*array _packet* 两个控制器使用同样的名字来共享数据。

参见10-5页的 ″接收一个生产型标签″。

11. 无论在控制器标签文件夹还是在程序标签文件夹中都将包含传送数据的逻辑，创建以下标签：

标签名	类型
array	DINT[x]其中，x等于要传送的元素数加122
*array*_offset	DINT(双整型)

其中：
*array*是用户发送的数据的名称。

12. 为将要把信息包的数据传送到目的数组创建或打开一个例程。

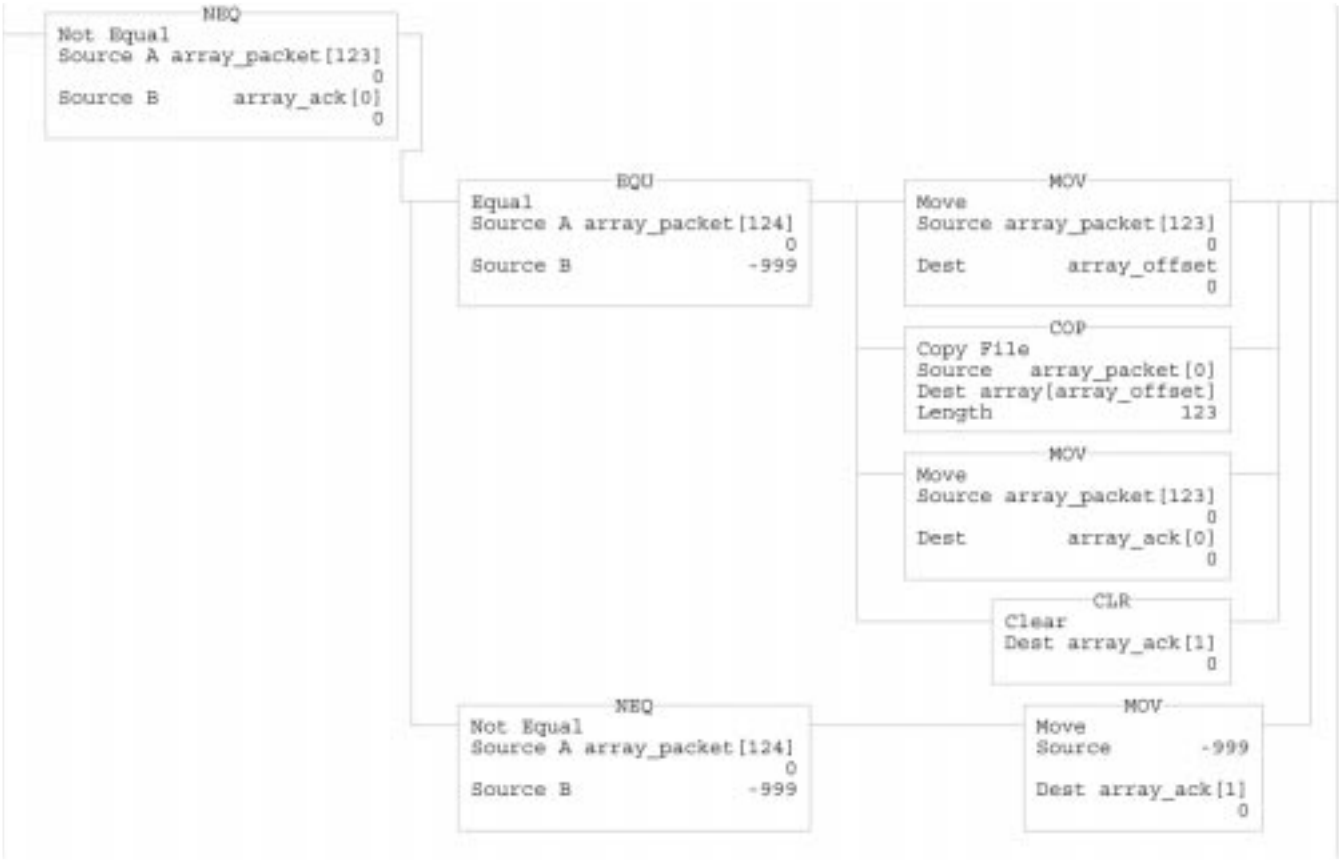
13. 输入下面的梯形图逻辑：

当*array_ ack [123]*中保存的偏移量值与*array_ ack [0]*中保存的值不同时，控制器已经开始接收一个新的数据信息包；这时梯级将检测信息包的最后一个元素值是否为-999。

如果这个信息包的最后一个元素等于-999，说明控制器已经收到一个完整的新的数据信息包并且开始复制运行：

- 偏移量的值从信息包送给array_offset。
- COP指令将数据从信息包复制到目的数组中，从偏移量的起始值开始。
- 偏移量的值移到*array_ ack [0]*时，表明复制成功。
- *array_ ack [1]*置零并等待一个新信息包到达的信号。

如果信息包的最后一个元素不等于-999，表明传送到控制器的信息包可能没有传完；这时将-999送给*array_ ack [1]*。生产者将-999送到信息包的最后一个元素，产生一个信息包传输校验信号。



采用小的信息包传送一个大的数组在提高系统性能方面优于其它传输数据的方法：

- 这种做法比用户将数据拆分成多个数组，并用生产型标签发送所使用的连接少。例如，在使用单个数组传送的情况下，一个含有5000个元素的数组将会占用40个连接(5000/125=40)。
- 比用户使用信息指令发送整个数组，要用更快的传输速度。
  - 只有在Logix5550执行“系统开销”部分期间，信息发送是不确定的，同时信息也在被执行。因此，信息将占用很长的时间来完成数据的传送。
  - 用户可以通过增加系统开销时间来减少传输时间，但是这将降低连续性任务执行的效率。

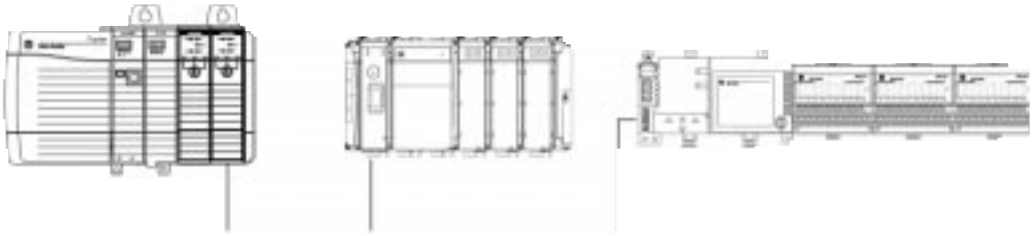
注 意;

与一个ASC II 设备通讯

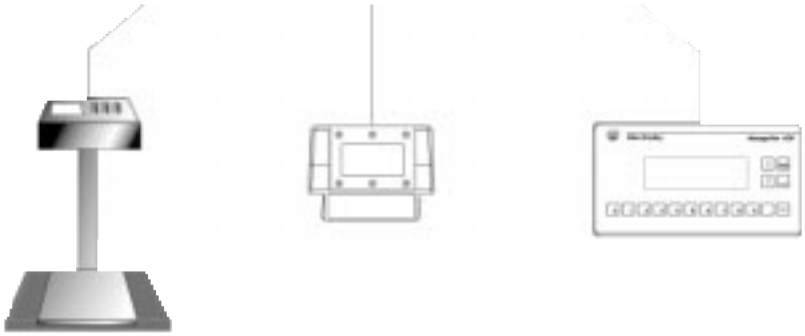
使用本章

通过控制器的串行端口与设备交换ASC II 数据。例如，用户可以使用串行端口：

- 从一个称重模块或条形码阅读机中读取ASC II 字符。
- 从一个ASC II 触发设备发送和接收信息，例如一个MessageView终端。



控制器的串口与ASCII设备的连接



如何使用本章

在用户使用本章之前：

- 为用户的应用组态ASC II 设备。

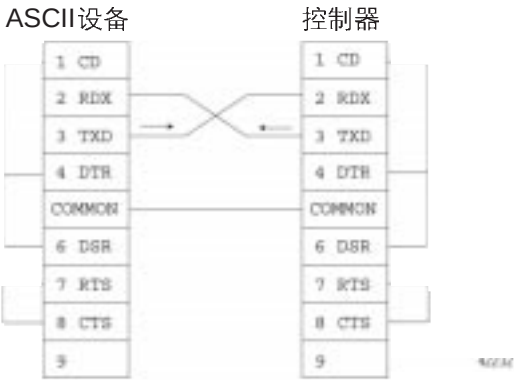
要完成这一过程，需要执行以下任务：

- 连接ASC II 设备
- 组态串行端口
- 组态用户协议
- 创建字符串数据类型
- 从设备中读取字符
- 向设备发送字符

连接ASC II设备

1. 对于ASC II 设备中的串行端口，要弄清哪些引脚 发送数据，哪些引脚接收数据。

2. 将接收引脚与相对应的发送引脚相连并连接跳线：

如果通讯：	连接器的连接方式如下：
握手	
无握手	

3. 将两个连接器的屏蔽电缆相连。

4. 用电缆连接控制器和ASC II 设备。

组态串行端口

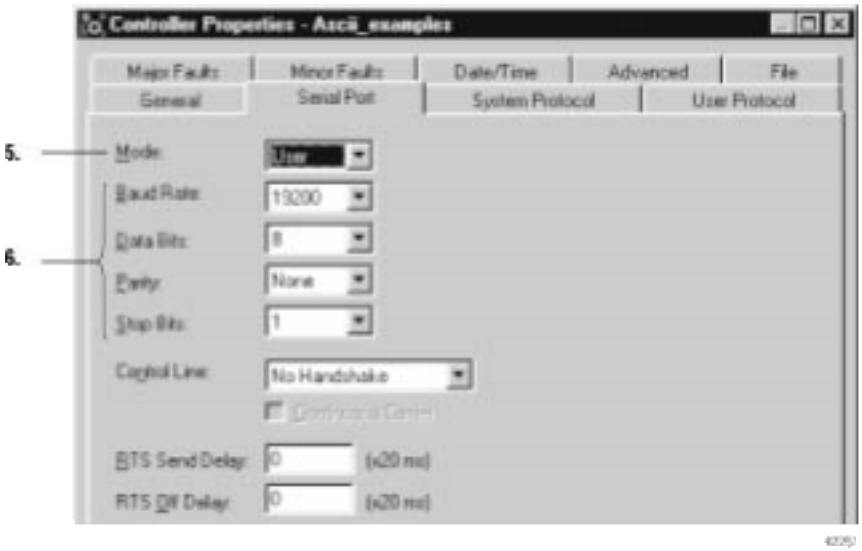
1、对ASC II 设备进行如下的通讯设置：

a. 波特率
b. 数据位
c. 奇偶校验
d. 停止位

2、打开RSLogix5000™工程

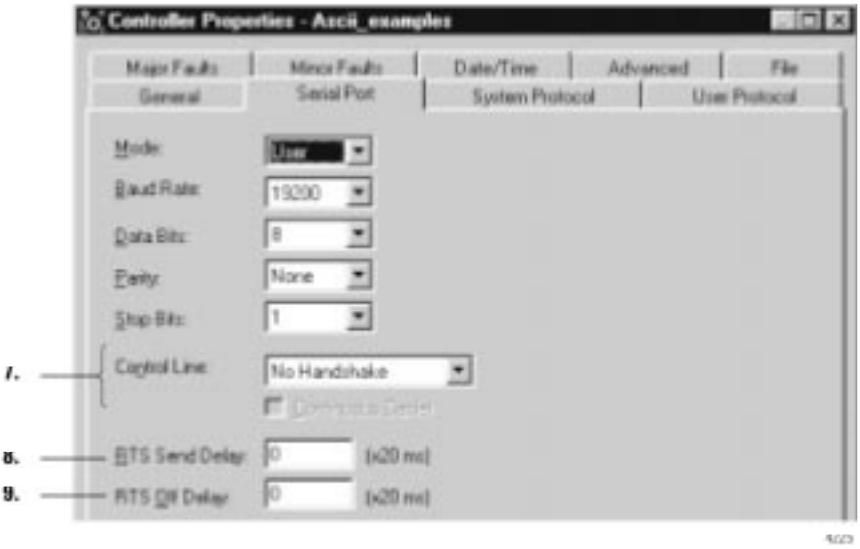
3、从在线工具栏中点击控制器按钮。

4、点击 *Serial Port*(串行端口)选项。



5、选择*User*(用户)。

6、为ASC II 设备进行第1步中的设置。



7. 选择控制线选项。

如果：	和：	这是：	选择：	然后：
用户不使用	→		无握手 *No Handshaking*	进行第10步
调制解调器用户使用调制解调器	点对点连接的两个调制解调器是全双工的。	→	全双工 Full Duplex	
	主调制解调器是全双工的，从调制解调器是半双工的。	主控制器 从控制器	全双工 Full Duplex 半双工 Half Duplex	选择Continuous Carrier选择框。
	系统中所有的调制解调器都是半双工的。	→	半双工 Half Duplex	清除Continuous Carrier选择框(缺省)。

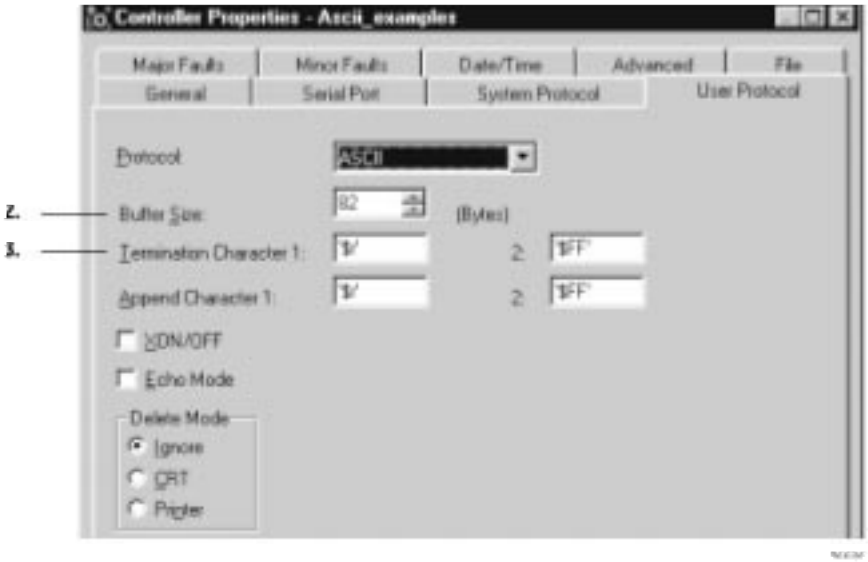
8. 填写介于RTS信号有效(高电平)时间和发送数据时间之间的延迟时间数(以20ms为单位)。例如：如果值为4，则产生80ms的延时。

9. 填写介于发送最后一个字符的时间和RTS信号无效(低电平)的时间之间的延迟时间数(以20ms为单位)。

10. 点击*Apply*(应用)。

组态用户协议

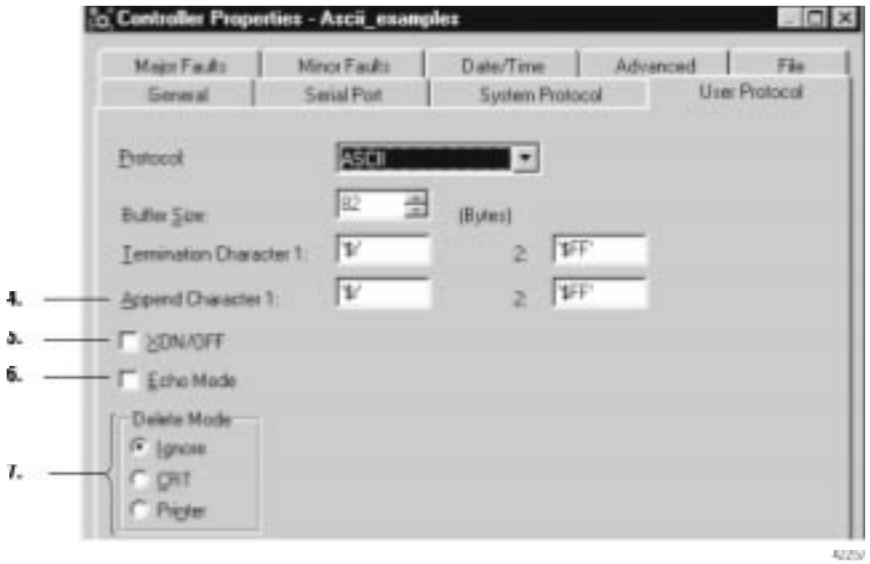
1. 点击*User Protocol*(用户协议)选项。



2. 选择或填写一个数值，这个值应大于或等于传送字符的最大值。(建议使用字符数值的二倍。)

3. 如果用户使用ABL或ARL指令，填写标志数据结尾的字符。要查找一个字符的ASC II 码，参见本手册的封底。

如果设备发送：	那么：	注释：
一个终止字符	A.在终止字符1的文本框内输入第一个字符的十六进制ASC II 码。 B.在终止字符2的文本框内输入$FF。	对于可印刷字符，如1或A，直接填写这个字符。
两个终止字符	在终止字符1和2的文本框中输入各自字符的十六进制ASC II 码。	



4. 如果用户使用AWA指令，填写附加给数据的字符。要查找一个字符的ASC II 码，参见本手册的封底。

要附加：	那么：	注释：
一个字符	A. 在附加字符1的文本框内输入第一个字符的十六进制ASC II 码。 B. 在附加字符2的文本框内输入$FF。	对于可印刷字符，如1或A，直接输入这个字符。
两个字符	在附加字符1和2的文本框中输入各自字符的十六进制ASC II 码。	

5. 如果ASC II 设备组态为*XON/XOFF*流程控制，则选择*XON/XOFF*选择框。

6. 如果ASC II 设备是一个CRT或者预组态为半双工传输方式的设备，选择*Ecbo Mode*选择框。

7. 选择删除模式：

如果ASC II 设备是：	选择：	注释：
CRT	*CRT*	• DEL 字符($7F) 和在DEL的字符之前的字符不被送到目的文件。 • 如果选择echo mode(回应模式)且ASC II 指令读取DEL字符，回应返回三个字符：BACKSPACE SPACE SPACEBACE($08 20 08)。
打印机	*Printer*	• DEL 字符($7F) 和在DEL的字符之前的字符不被送到目的文件。 • 如果选择echo mode(回应模式)且ASC II 指令读取DEL字符，回应返回两个字符：/($2F) 在被删除的字符之后。
非以上所述	*Ignore*	DEL 字符($7F) 和其它字符一样对待。

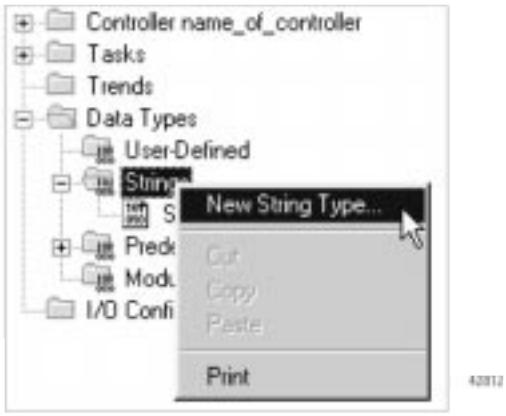
8. 点击*OK*(完成)。

创建字符串数据类型

用户使用字符串数据类型在标签内存储ASC II 字符。

用户使用缺省STRIG数据类型。可保存82个字符

或



用户可以创建一个新的字符串数据类型，并定义其保存字符的数量.

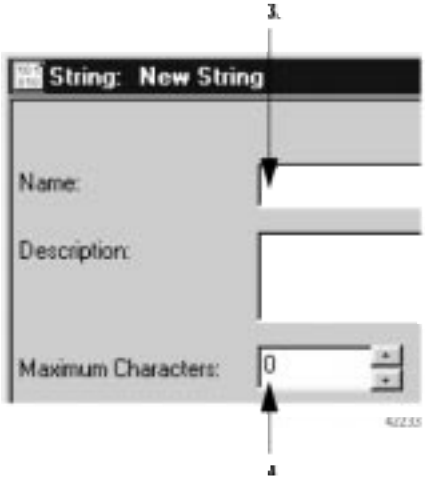
重要提示

当用户创建一个新的字符串数据类型时，一定要谨慎。如果用户决定以后改变字符串数据类型的大小，可能会使当前使用此数据类型的标签中的数据丢失。

如果用户：	那么：
使一个字符串数据类型变小	• 数据被截取 • LEN没有变化
使一个字符串数据类型变大	数据和LEN复位为零

1. 用户是否想要创建一个新的字符串数据类型？

如果：	那么：
否	按照12-9页所述从设备中读取字符。
是	进行第2步。



2. 在控制器项目管理器中，右键点击*Strings*(字符串)并选择New *String*(新建字符串)…

3. 为数据类型输入一个名字。

4. 填写字符串数据类型将要保存字符数量的最大值。

5. 选择*OK*(完成)。

从设备中读取字符

一般来讲，在用户使用ACB或ABL指令读取缓冲器之前，要核实缓冲器中包含所需的字符。

- ARD或ARL指令连续地读缓冲器直到指令读取到所需的字符。
- 当ARD或ARL指令读缓冲器时，除了ACL指令，其 它的ASC II 串行端口指令都不能执行。
- 核实缓冲器包含所需字符，这样可阻止当输入设备发送数据时，ARD或ARL禁止其它ASC II 串行端口指令的执行。

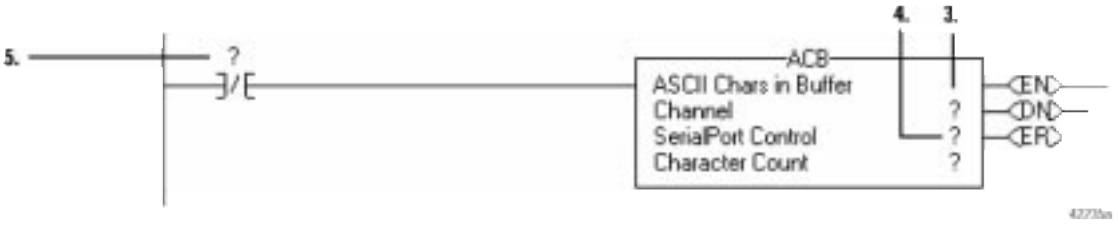
要了解关于ASC II 设备串行端口指令的详细信息，参见*Logix5000 Controllers General Instruction Set Reference Manual*《Logix5000控制器指令集参考手册》，出版号1756-RM003。

重要提示 如果用户对在RSLogix5000工程中如何输入梯形图逻辑不熟悉，请先回顾4-1页的 ″规划例程″。

1．用户在读取的设备属于哪种类型？

如果设备是：	那么：
条形码阅读机	进行步骤2。
发送字符数量固定的称重计	
信息或显示终端	进行步骤14。
发送字符数量变化的称重计	

2．输入下面的梯级：

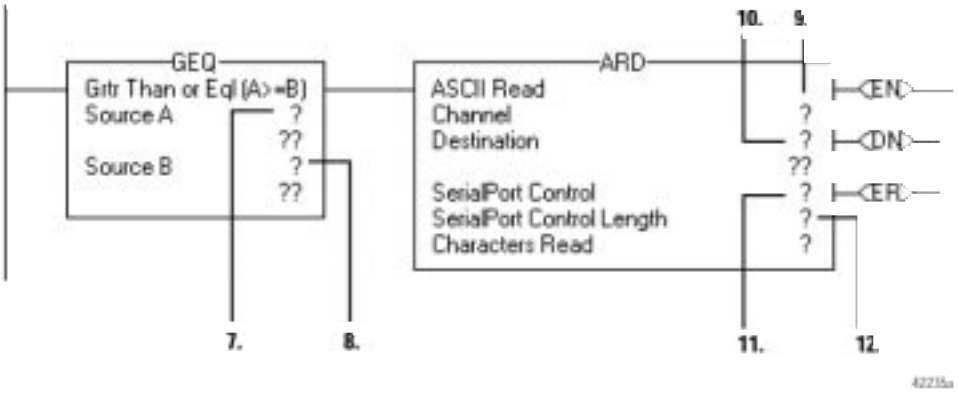


3．输入0。(0通道是串行端口。)

4．为ACB指令输入一个标签名并定义此标签的数据类型为SERIAL_PORT_CONTROL。

5．输入ACB标签的EN位。(步骤4中的标签。)

6. 输入以下梯级：



7. 输入ACB标签的POS成员。(步骤4中的标签。)

8. 输入数据中的字符数量。

9. 输入0。

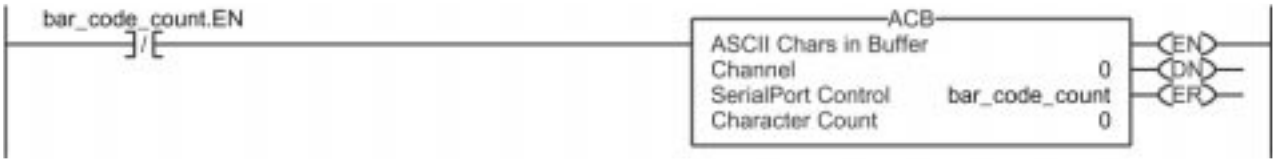
10. 输入一个用于保存ASC II 字符的标签名。定义数据类型为**string**(字符串)。

11. 为ARD指令输入一个标签名并定义此标签的数据类型为SERIAL_PORT_CONTROL。

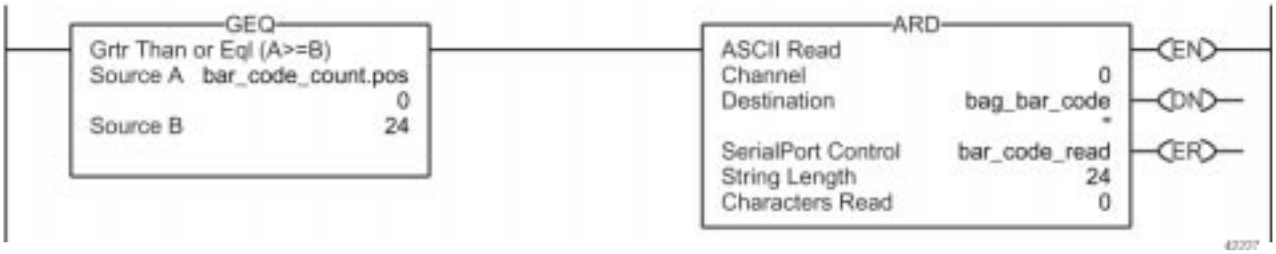
12. 输入数据中的字符数量。

例子

一个条形码阅读机向控制器的串行端口(0通道)发送条形码。每个条形码包含24个字符。为了确定控制器什么时候接收条形码，ACB指令不断地计算缓冲器中的字符。



当缓冲器中包含至少24个字符时，控制器已接收到一个条形码。ARD指令将条形码转移到*bag_bar_code*标签中。



13. 用户是否想给设备发送数据？

如果：	那么：
是	按照12-14页所述向设备中发送字符
否	停止。用户已完成这一步，要使用这个数据，参见13-1页的“处理ASC II 字符”。

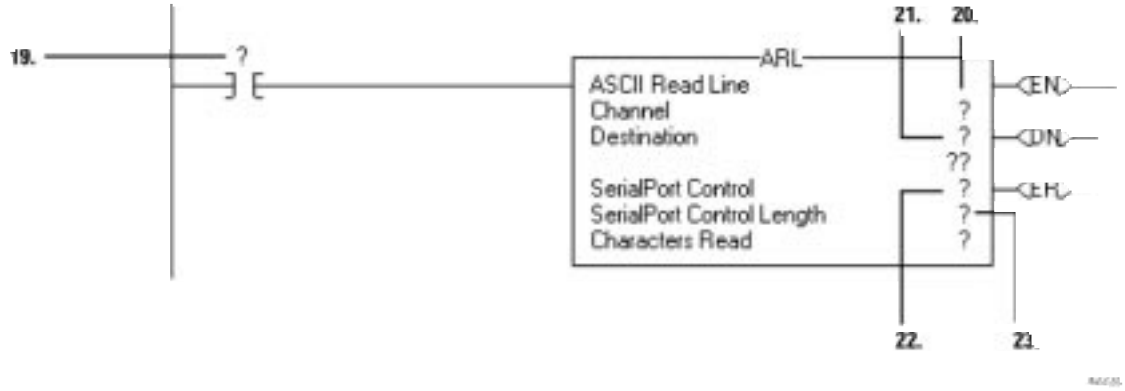
14. 输入下面的梯级：

15. 输入0。

16. 为ABL指令输入一个标签名并定义此标签的数据类型为SERIAL_PORT_CONTROL。

17. 输入ABL标签的EN位。(步骤16中的标签。)

18. 输入下面的梯级：



19. 输入ABL标签的FD位。(步骤16中的标签。)

20. 输入0。

21. 输入一个用于保存ASC II 字符的标签名。定义数据类型为**string**(字符串)。

22. 为ABL指令输入一个标签名并定义此标签的数据类型为SERIAL_PORT_CONTROL。

23. 输入0。

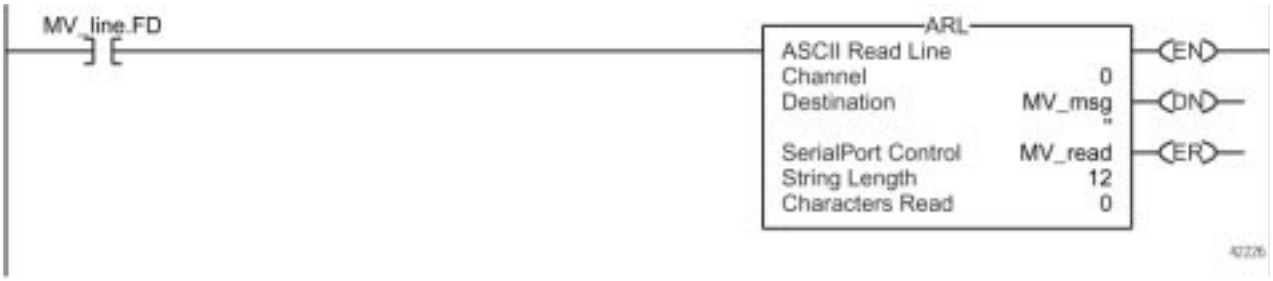
这样使得指令设置的串行端口的控制长度与目的文件的大小相等。

例子

不断地检测缓冲器中来自MessageView的信息。

- 由于每个信息以回车键($0D)结束，回车键作为终止字符在控制器属性对话框的用户协议选项中被组态。
- 当ABL发现一个回车键时， 将FD置位。

当ABL指令发现Carriage返回值(MV-line.FD设置)，控制器将从缓冲区取出字符串，直到包括Carriage返回值，然后在MV-msg标签内替代它们



24. 用户是否想给设备发送数据？

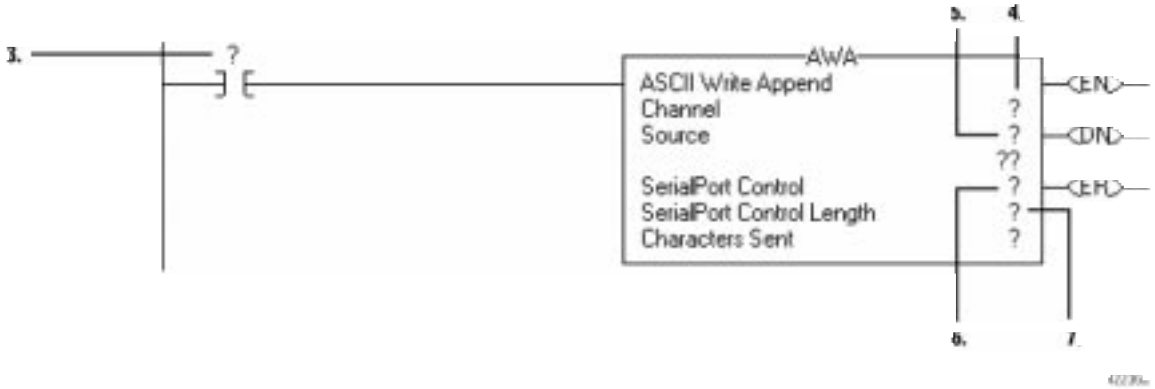
如果：	那么：
是	按照12-14页所述向设备中发送字符
否	停止。用户已完成这一步，要使用这个数据，参见13-1页的 ″处理ASC II 字符″。

向设备发送字符

1. 确定起始位置：

如果用户：	并且：	那么：
数量的字符	总是发送同样 想自动地在数据末尾附加一个或两个字符	跳到第2步。
	不想附加字符	跳到第9步。
发送不同数量的字符	想自动地在数据末尾附加一个或两个字符	跳到第16步。
	不想附加字符	跳到第24步。

2. 输入下面的梯级：



3. 输入输入条件，确定何时发送字符：
   - 用户可使用任意类型的输入指令。
   - 每次发送字符时，指令必须由假变真。

4. 输入0。

5. 输入一个用于保存ASC II 字符的标签名。定义数据类型为**string**(字符串)。

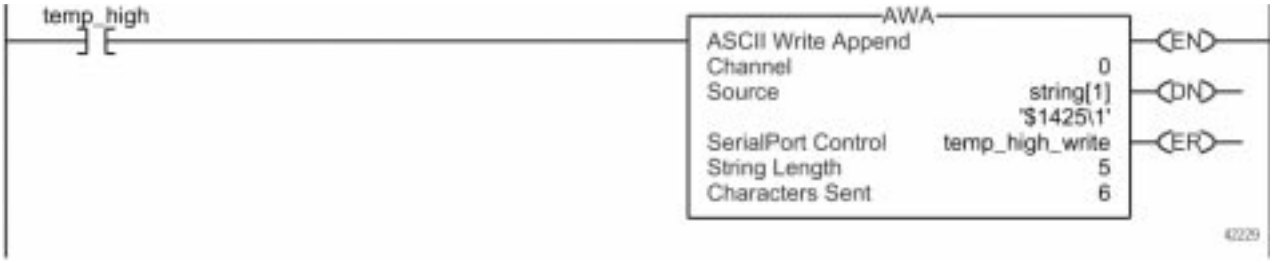
6. 为AWA指令输入一个标签名并定义此标签的数据类型为SERIAL_PORT_CONTROL。

7. 输入发送字符的数量。省略通过指令附加的字符。

例子

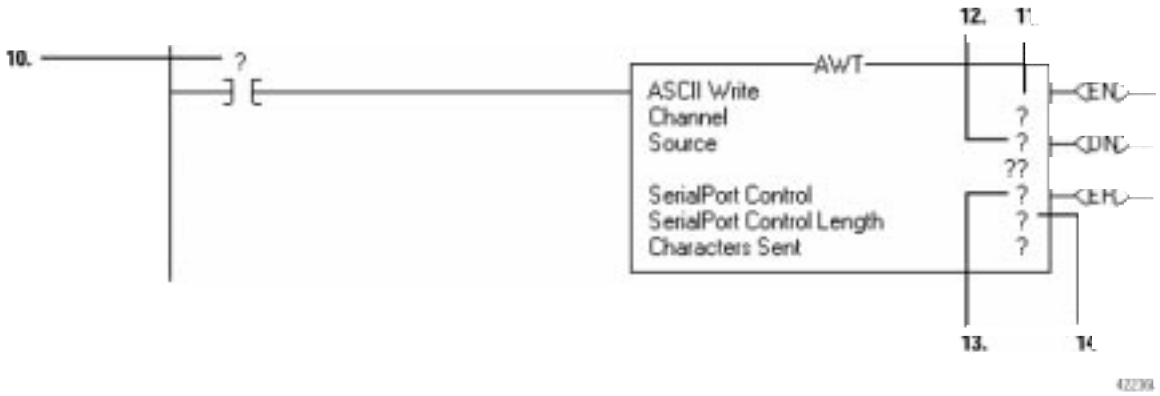
当温度超出上限时(*temp_high*闭合)，AWA指令从*string[1]*标签向一个MessageView终端发送5个字符。

- $14被算作一个字符。它是Ctrl-T字符的十六进制代码。
- 指令也发送(附加)在用户协议中定义的字符。在这个例子中，AWA指令发送一个回车键($0D)，这个回车键标记在信息的末尾。



8. 参见12-21页的输入ASC II 字符。

9. 输入下面的梯级：



10. 输入输入条件，确定何时发送字符：
    - 用户可使用任意类型的输入指令。
    - 每次发送字符时，指令必须由假变真。

11. 输入0。

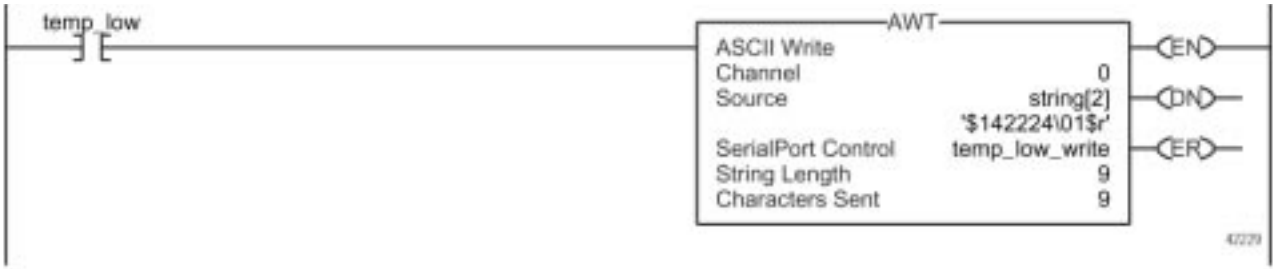
12. 输入一个用于保存ASC II 字符的标签名。定义数据类型为**string**(字符串)。

13. 为AWT指令输入一个标签名并定义此标签的数据类型为SERIAL_PORT_CONTROL。

14. 输入发送字符的数量。

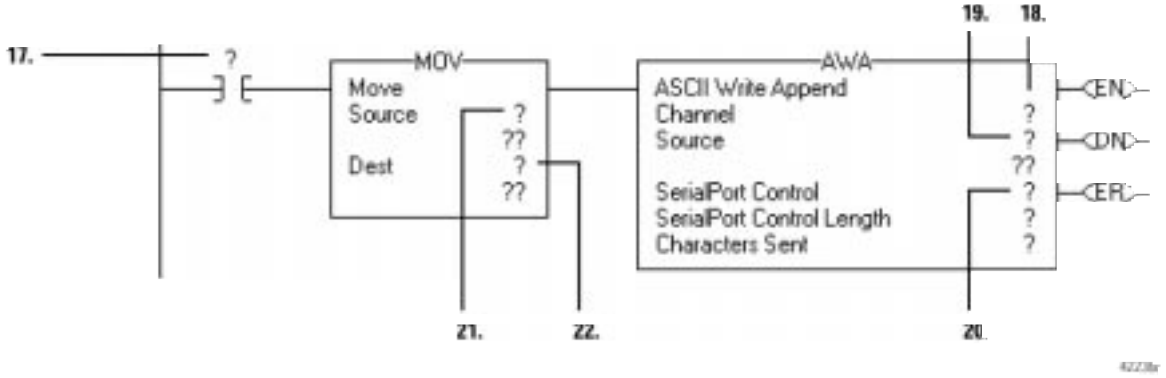
例子

当温度达到下限时(*temp_low*闭合)，AWT指令从*string[2]*标签向一个MessageView终端发送9个字符。(*$14*被算作一个字符。它是Ctrl-T字符的十六进制代码。)



15. 参见12-21页的输入ASC II 字符。

16. 输入下面的梯级：



17. 输入输入条件，确定何时发送字符：
    - 用户可使用任意类型的输入指令。
    - 每次发送字符时，指令必须由假变真。

18. 输入0。

19. 输入一个用于保存ASC II 字符的标签名。定义数据类型为**string**(字符串)。

20. 为AWA指令输入一个标签名并定义此标签的数据类型为SERIAL_PORT_CONTROL。

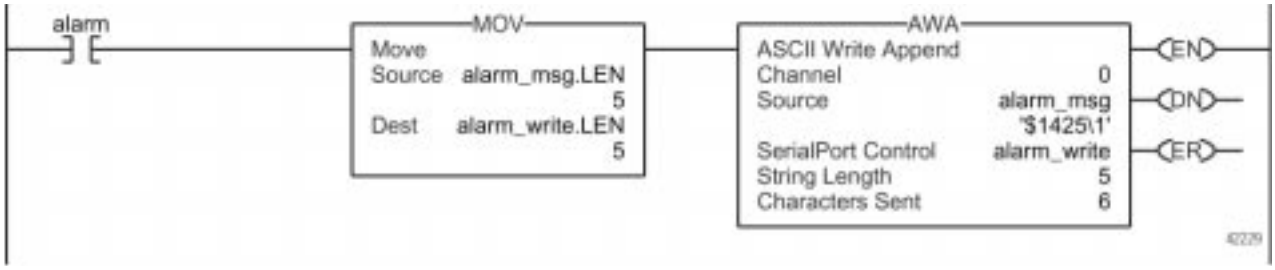
21. 输入源标签LEN成员。(步骤19中的标签。)

22. 输入AWA指令的LEN成员。(步骤20中的标签。)

例子

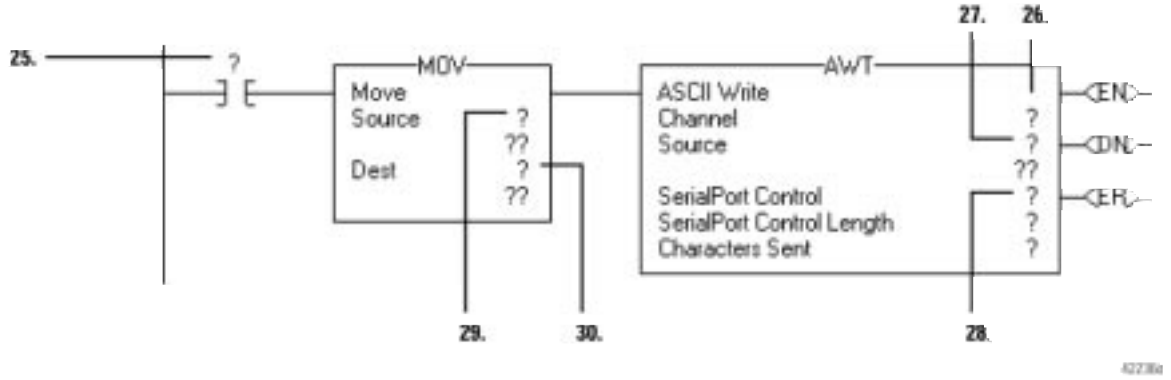
当*alarm*为真时，AWA指令发送*alarm_msg*中的字符并且附加一个终止字符。

- 由于*alarm_msg*中的字符数量是变化的，梯级首先向AWA指令的长度(*alarm_write.LEN*)中送入*alarm_msg*的长度(*alarm_msg.LEN*)。
- 在*alarm_msg*中，*$14*被算作一个字符。它是Ctrl-T字符的十六进制代码。



23. 参见12-21页的输入ASC II 字符。

24. 输入下面的梯级：



25. 输入输入条件，确定何时发送字符：
- 用户可使用任意类型的输入指令。
- 每次发送字符时，指令必须由假变真。

26. 输入0。

27. 输入一个用于保存ASC II 字符的标签名。定义数据类型为**string**(字符串)。

28. 为AWA指令输入一个标签名并定义此标签的数据类型为SERIAL_PORT_CONTROL。

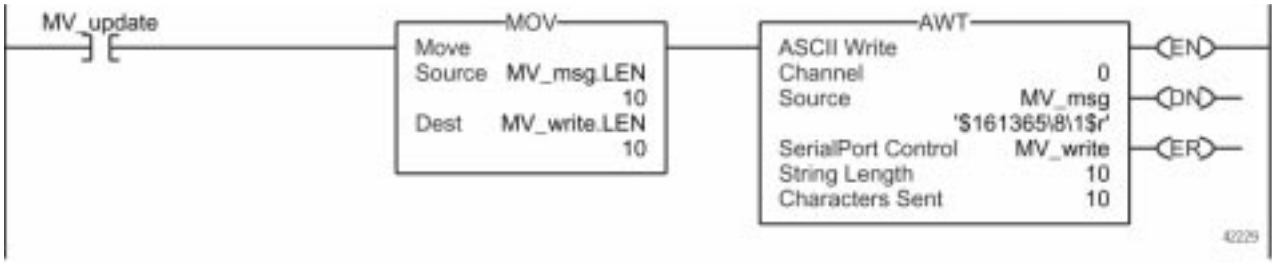
29. 输入源标签LEN成员。(步骤27中的标签。)

30. 输入AWT指令的LEN成员。(步骤28中的标签。)

例子

当*MV_update*为真时，AWT指令发送*MV_msg*中的字符。

- 由于*MV_msg*中的字符数量是变化的，梯级首先向AWT指令的长度(*MV_write.LEN*)中送入*MV_msg*的长度(*MV_msg.LEN*)。
- 在*MV_msg*中，*$16*被算作一个字符。它是Ctrl-V字符的十六进制代码。



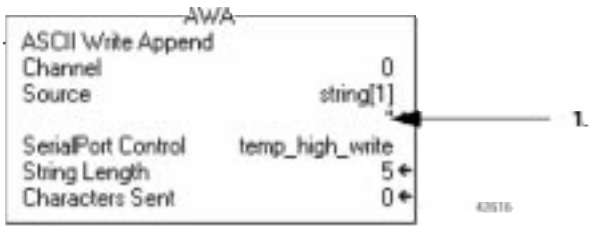
31. 参见12-21页的输入ASC II 字符。

输入ASC II 字符

确定用户是否必须完成这一步：

如果：	那么：
用户想用梯形图逻辑创造字符串。	参见12-1页的 ″处理ASC II 字符″。
用户想输入字符。	跳到第1步。

重要提示 字符串浏览器窗口显示字符等于字符串标签中的LEN成员的值。字符串标签可能包含字符串浏览器窗口没有显示的附加数据。



1. 双击源的值区域。

用户在窗口中所看见的字符的数量。这跟字符串标签中的LEN成员的值相同。

字符串标签中所能保存字符数量的最大值。

2. 输入字符串的字符。

3. 点击OK(完成)。

注 意：

处理 ASCII 字符

使用本章

通过本章可以:

- 解释条形码，并进行基于条形码的操作。
- 当重量以 ASCII 字符形式发送时，使用称重计显示的重量。
- 对 ASCII 触发设备的信息译码，如操作员终端。
- 在用户应用中使用变量为 ASCII 触发设备建立一个字符串。

如何使用本章

要 点: 如果用户对在 RSLogix5000 工程中如何输入梯形图逻辑不熟悉，请先回顾 4-1 页的 ″编辑例程″。

根据实际应用,用户不需要完成本章中所有的任务。使用下面的表格来确定开始的位置

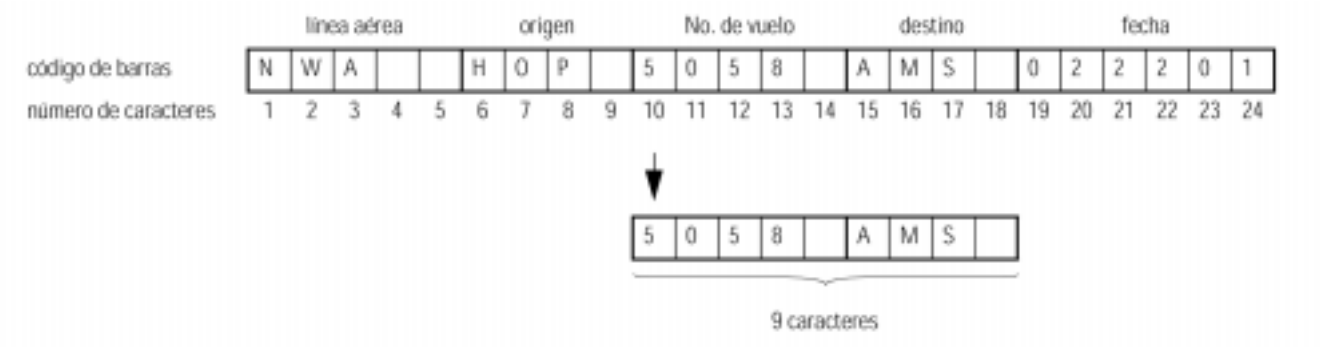
如果用户想:	参见:	页码:
从条形码中分离出指定的信息	提取条形码的一部分	13-2
对一个指定字符的字符串搜索一个数组	查看一个条形码	13-4
比较两个字符串的字符	检查条形码字符	13-10
使用称量计称量	转换成数值	13-12
从操作员终端进行信息译码	译解 ASCII 码	13-14
创建一个字符串并发送到操作员终端	建立一个字符串	13-18

要了解与 ASC II 相关指令的详细信息，参见 Logix5000 Controllers General Instruction Set Reference Manual 《Logix5000 控制器指令集参考手册》，出版号 1756-RM003。

提取条形码的一部分

按照下面的步骤来提取部分条形码，这样用户就可以对其值进行操作。

例如，一个含有机场传送带上包裹信息的条形码。要检查包裹的班机号和目的地，用户可提取 10-18 中的字符。



步骤:

1. 输入下面的梯级:

2. 输入读取条形码的 ARD 指令的 EM 位。

3. 输入包含条形码的字符串标签。

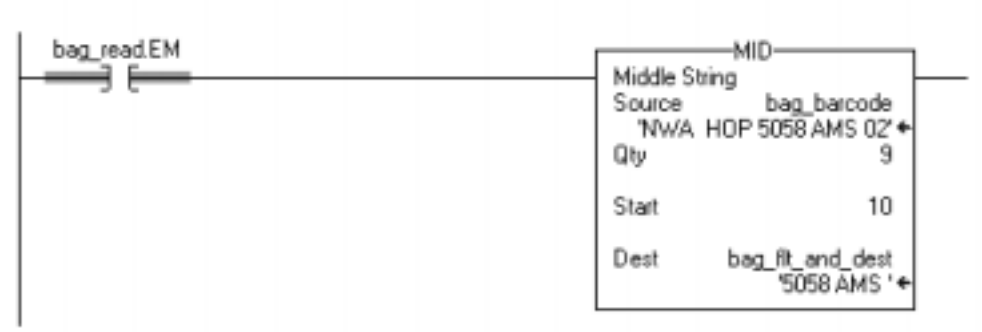
4. 输入用户想检验的部分条形码的字符数。

5. 输入用户想检验的部分条形码的首字符的位置。

6. 输入一个标签名，用来保存想要检验的部分条形码。将其数据类型定义为字符串。

例 如

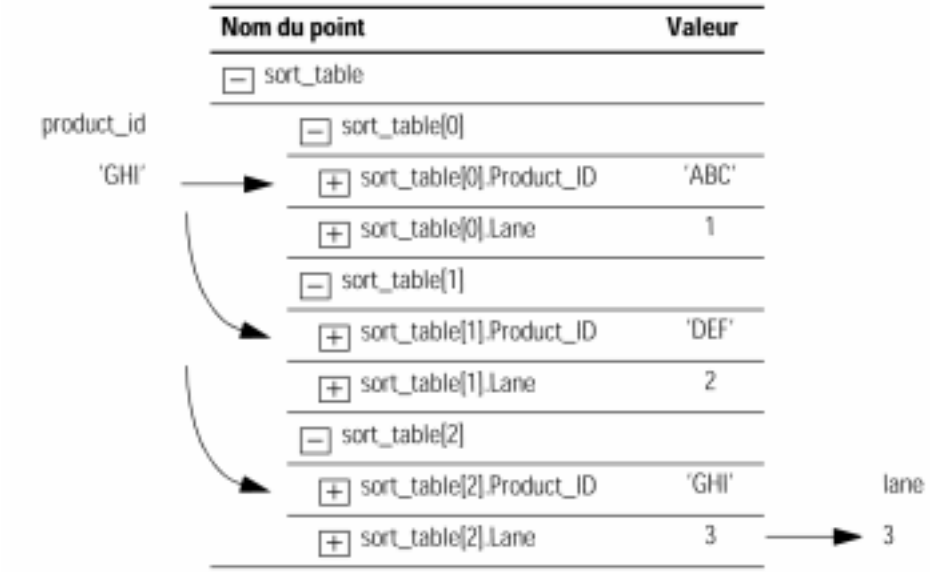
在机场的行李传送管理中，每个包裹都有一个条形码。条形码的10-18位是包裹的班机号和目的机场。在读完条形码后（*bag_read*.EM 闭合），MID 指令将班机号和目的机场复制到 *bag_flt_and_dest* 标签中。



查看一个条形码

按照下列步骤返回基于条形码数据项的具体信息。

例如，在分类操作中，一个自定义数据类型的数组创建显示每种类型货物航线编号的表格。要决定用哪条航线来运输货物，控制器可以按照货物的ID号来查表（根据条形码的字符来确定货物）。

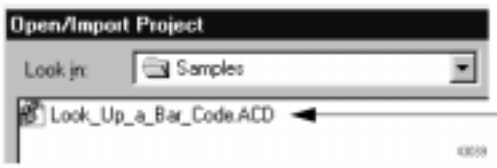


要查看一个条形码:

- 创建 PRODUCT_INFO 数据类型
- 搜索字符
- 识别航线编号
- 去掉错误字符
- 输入货物的 ID 号和航线编号

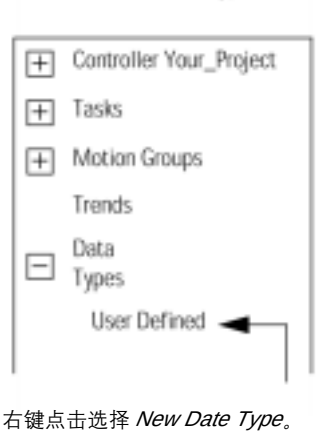
提　示:

要从工程例子中复制以上内容，打开 *...\RSLogix 5000\Projects*\Samples 文件夹。



创建 PRODUCT_INFO 数据类型

创建新的数据类型:



右键点击选择 *New Date Type*。

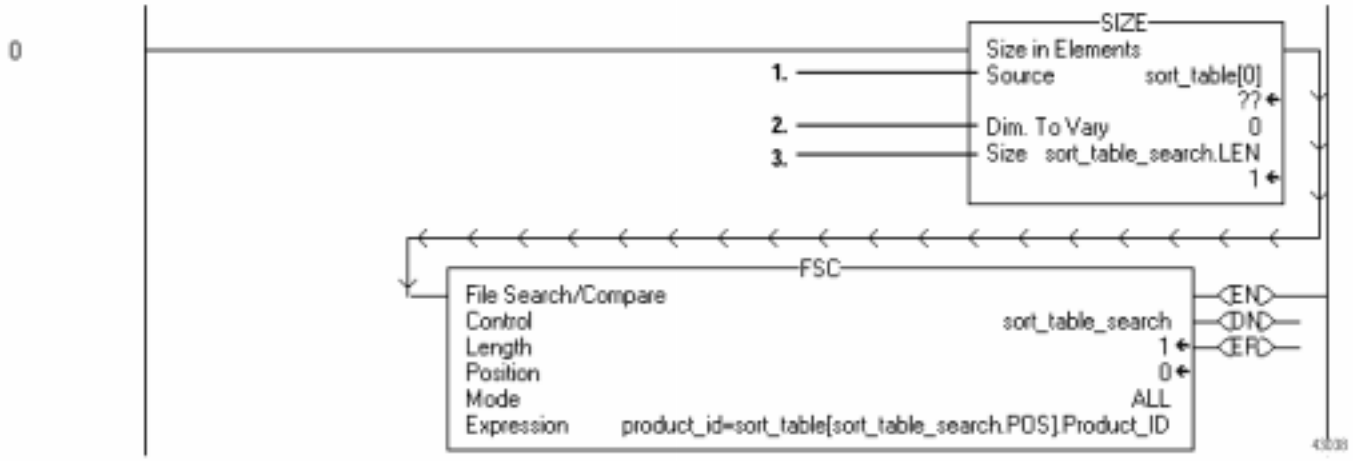
创建以下自定义数据类型。

Type de données :PRODUCT_INFO

Nom	PRODUCT_INFO
Description	Identifie la destination d'un élément sur la base d'une chaine ASCII qui identifie l'élément
Members	

	Nom	User Defined	Style	Description
	⊞ Product_ID	STRING		Caractères ASCII qui identifient l'élément
	Lane	DINT	Decimal	Destination de l'élément, sur la base de son ID

搜索字符



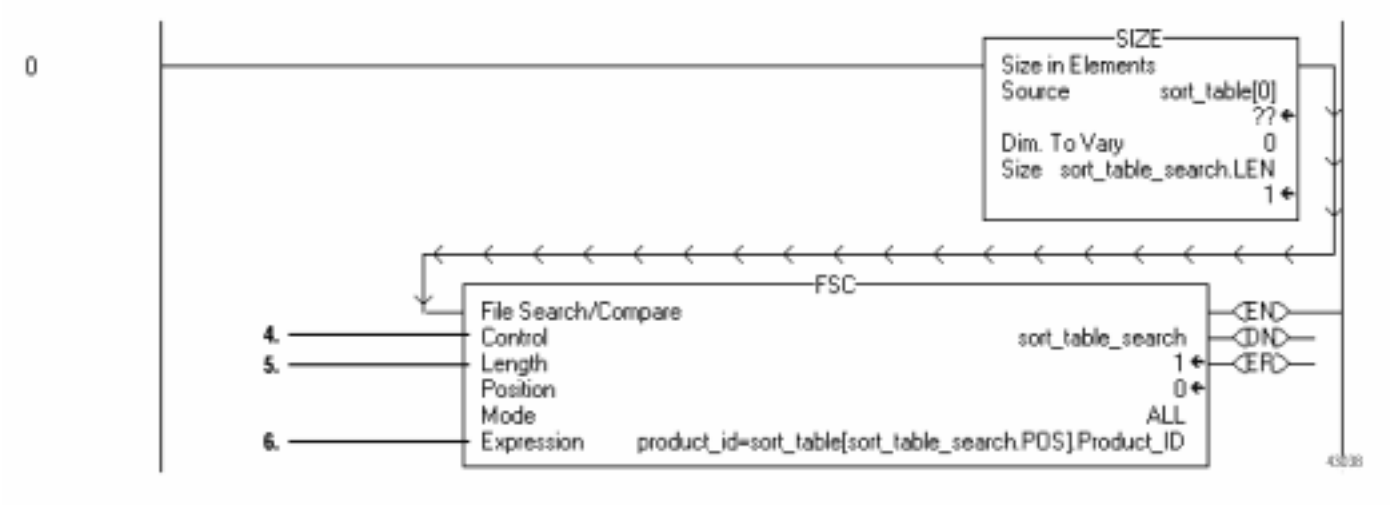
1. SIZE 指令计算 *sort_table* 数组中元素的数量。这个数组包含每个数据项的货物的 ID 号和相应的航线编号。

标签名	类型
sort_table	PRODUCT_INFO[*number_of_items*] 其中， *number_of_items* 是用户必须分类的数据项的随机编码的数量。

2. SIZE 指令计算零维数组中元素的数量。换句话说，数组只有一维。

3. SIZE 指令将其后面 FSC 指令的长度设置成 *sort_table* 数组的长度。这样可以保证 FSC 指令搜索数组的准确长度。

标签名	类型
sort_table_search	CONTROL



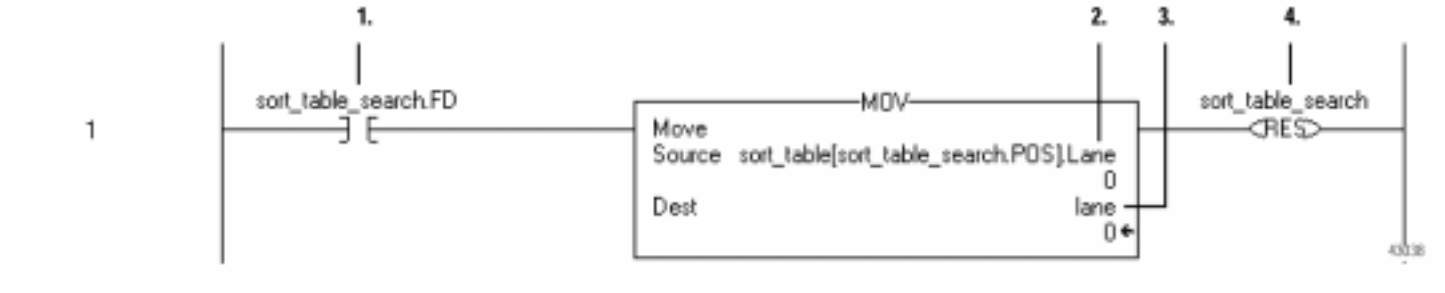
4. *sort_table_search* 标签控制 FSC 指令，在 *sort_table* 数组中查找条形码字符。

5. 尽管前一条指令设置了这条指令的长度，但在校验工程时需要给软件赋初始值。

6. *product_id* 标签含有确认数据项条形码的字符。FSC 指令在 *sort_table* 数组中查找 Product_ID 的每一个成员，直到指令找到与 *product_id* 标签匹配的数据为止。

标签名	类型
product_id	STRING

识别航线编号



1. 当 FSC 指令在 *sort_table* 数组中找到货物的 ID 号后，指令将 FD 置位。

2. 当 FSC 找到一个匹配的数据时，POS 成员显示 *sort_table* 数组中的与之匹配的元素数量。相应的 LANE 成员显示匹配的航线编号。

3. 根据 POS 的值 MOV 指令将相应的航线编号送给 lane 标签。控制器使用这个标签的值来发送数据项。

标签名	类型
lane	DINT

4. 在 MOV 指令设置完 *lane* 标签的值后，RES 指令将 FSC 指令复位，以便搜索下一个货物的 ID 号.

去掉错误字符

1. 如果 FSC 指令在 *sort_table* 数组中没有找到货物的 ID 号，指令将 DN 置位。

2. 当找不到匹配的数据时，MOV 指令将 999 送给 lane 标签。这告诉控制器去掉或者重新发送数据项。

3. 在 MOV 指令设置完 lane 标签的值后，RES 指令将 FSC 指令复位，以便搜索下一个货物的 ID 号.

输入货物的 ID 号和航线编号

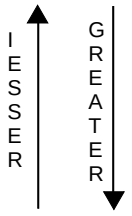
在 *sort_table* 数组中，输入确认每个数据项 ASCII 字符来和相应的航线编号。

Tag Name	Value
⊟ sort_table	{...}
⊟ sort_table[0]	{...}
⊞ sort_table[0].Product_ID	ASCII characters that identify the first item
⊞ sort_table[0].Lane	lane number for the item
⊟ sort_table[1]	{...}
⊞ sort_table[1].Product_ID	ASCII characters that identify the next item
⊞ sort_table[1].Lane	lane number for the item

检查条形码字符

在这个任务中，用户使用一条比较指令（EQU,GEQ,GRT,LEQ,LES,NEQ）来检查具体的字符。

- 用字符的十六进制值来确定一个字符串是小于还是大于其它字符串。
- 当两个字符串像电话簿里的电话号码一样排序时，字符串的排列顺序将确定它们的大小:



ASCII 字符	十六进制代码
1ab	$31$61$62
1b	$31$62
A	$41
AB	$41$42
B	$42
a	$61
ab	$61$62

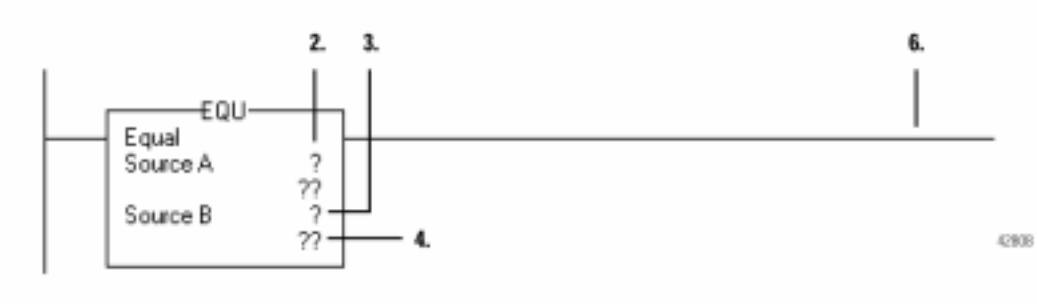
AB<B

a>B

步骤:

1. 输入一个梯级和一个比较指令。

要看字符串是否:	输入指令:
等于指定的字符	EQU
不等于指定的字符	NEQ
大于指定的字符	GTR
大于或等于指定的字符	GEQ
小于指定的字符	LES
小于或等于指定的字符	LEQ



2. 输入存有用户想要检验的部分条形码的标签。(提取部分条形码的目的文件，步骤6.)。

3. 输入一个标签名，来存储用户想要逆向检验的字符。将其数据类型定义为字符串。

4. 双击数据源 B 区的值。

5. 输入要逆向检验的 ASCII 字符，并选择 *OK*（完成）。

6. 输入所需的输出。

例 如　当 *bag_glt_and_dest* 等于 *gate[1]*时，*xfer[1]*启动。把包裹发送到指定的门。

7. 用户想检验条形码的其它部分吗？

如果	那么
是	参见 13-2 页提取条形码的一部分。
否	停止。用户已经完成这一步。

转换成数值

使用下列步骤可以将表示 ASCII 的值转换成在用户应用程序中可用的 DINT 或 REAL 形式的值。

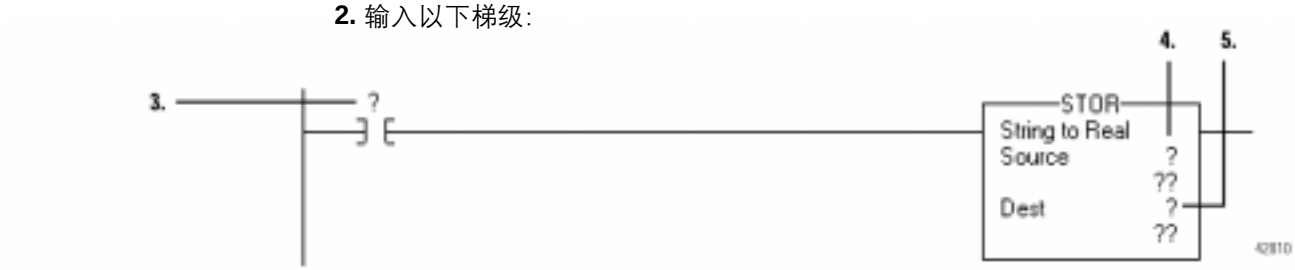
- STOD 和 STOR 指令可以跳过任何初始的控制或非数字字符（除了负数）。
- 如果字符串中包括多组被分隔符（如: /）分隔的数，STOD 和 STOR 指令只能转换第一组数。

步骤:

1. 哪种类型的数值?

如果是:	**那么**:
浮点型	转到第 2 步。
整型	传到第 7 步。

2. 输入以下梯级:



3. 输入读数指令 ARD 或 ARL 的 EM 位.

4. 输入包含数值的字符串标签。

5. 输入一个存有用户应用程序中所用数值的标签名。定义数据类型为 REAL 实数。

例　子

在读完称重计的数据之后（*weight_read*.EM 闭合）STOR 指令将保存在 *weight_ascii* 中的数字字符转换为 REAL 数值并将结果保存在 *weight* 中。

6. 跳到第 11 步。

7. 输入以下梯级:

8. 输入读数指令的 ARD 或 ARL 的 EM 位.

9. 输入包含数值的字符串标签。

10. 输入一个存有用户应用程序中所用数值的标签名。定义数据类型为 DINT 双整型。

例　子

当 *MV_read.EM* 闭合时，STOD 指令将 *MV_msg* 中的第一批数字字符转换成整型数值。指令跳过初始控制字符 ($06) 并在分隔符 (\ ) 处停止。

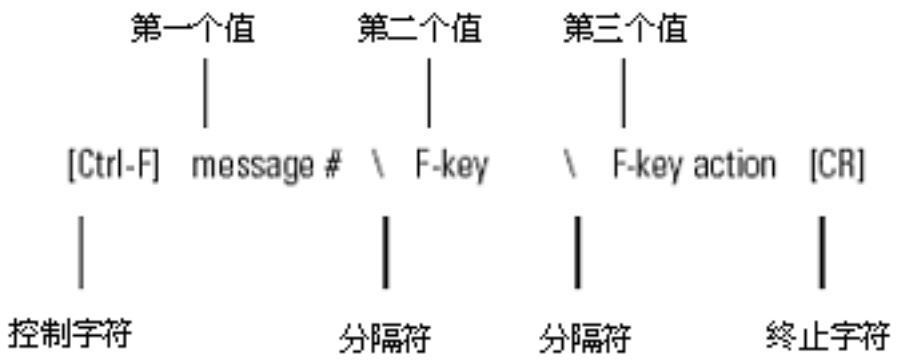
11. 这个字符串是否具有用户想使用的其它数值?

如果:	**那么**:
是	参见 13-14 页译解 ASCII 码信息。
否	停止。用户已经完成这一步。

译解 ASCII 码信息

使用如下步骤将含有大批数值的 ASCII 码信息提取并转换。

例如，一个信息可能像下面这样:



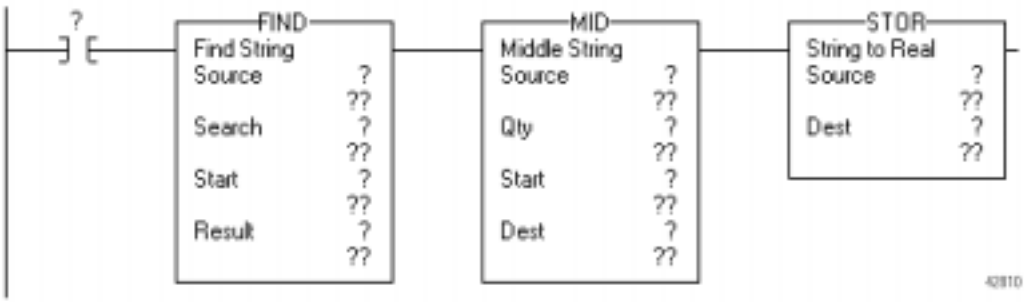
1. 决定起始位置:

如果:	并且:	那么:
字符串含有多个值	这是第一个数	参见 13-12 的转换成数值
	这不是一个数	跳到第 2 步
字符串只含一个值	——→	参见 13-12 的转换成数值

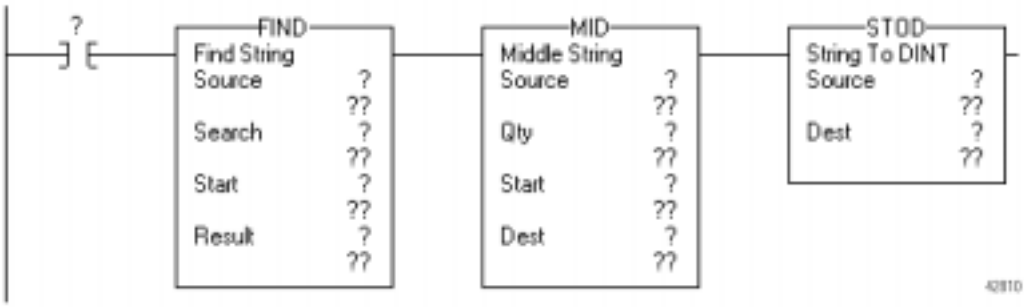
2. 哪种类型的数值?

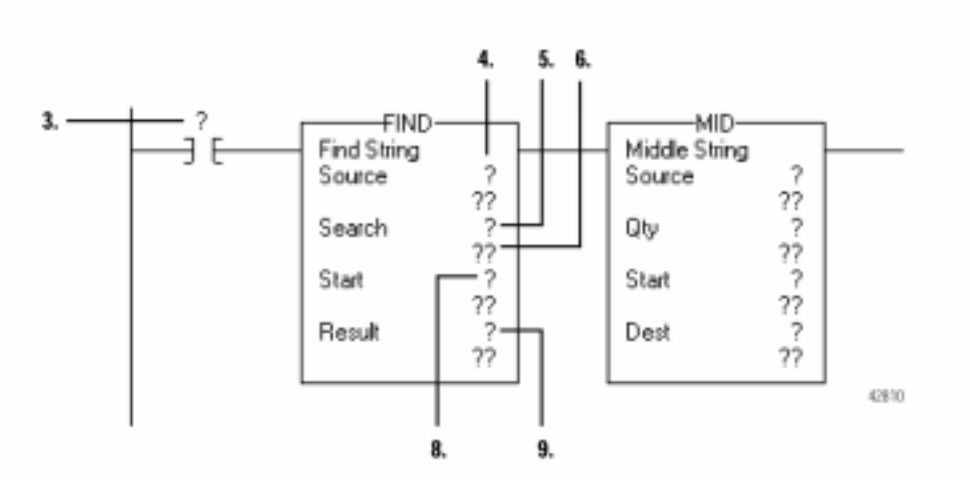
如果:	那么:
浮点型	输入梯级 A: 查找并转换浮点型数值
整型	输入梯级 B: 查找并转换整型数值。

梯级 A: 查找并转换成浮点型数值



梯级 B: 查找并转换成整型数值





3. 输入可以读数值的 ARL 指令的 EM 位。

4. 输入包含数值的字符串标签。

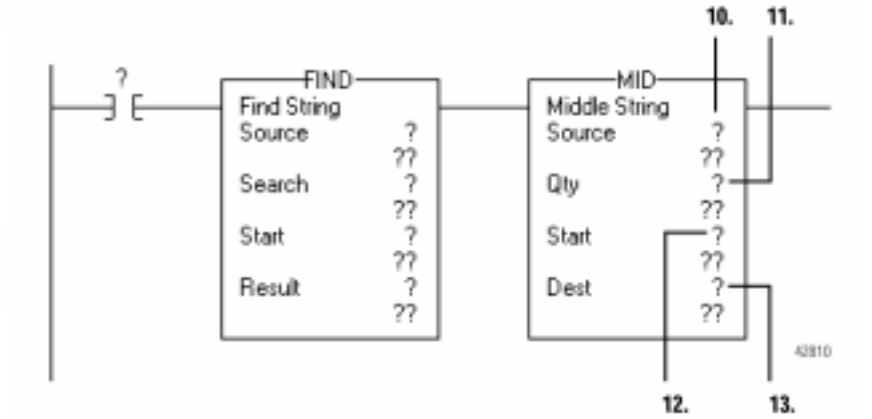
5. 输入一个标签名，它存有以分隔符标识数值起始的标志。定义数据类型为字符串。

6. 双击 Search 的值域。

7. 输入分隔符并选择 *OK* (完成)。

8. 输入在字符串中开始搜索的指针位置。
- 最初，用户可以用 0 找到第一个分隔符。
- 为了解译附加信息，增加该值来搜索下一个分隔符。

9. 输入标签名来存储分隔符的位置。定义数据类型为 DINT 双整型。



10. 输入包含数值的字符串标签。

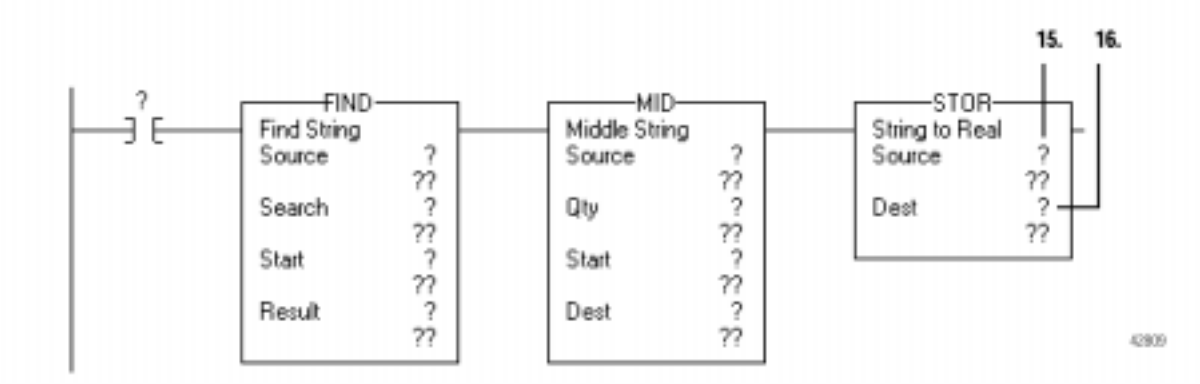
11. 输入能包含的最大字符数的数值。

12. 输入存储分隔符位置的标签。(步骤 9 中的标签。)

13. 输入一个标签名来存储数值。定义数据类型为字符串。

14. 用户使用哪种转换指令?

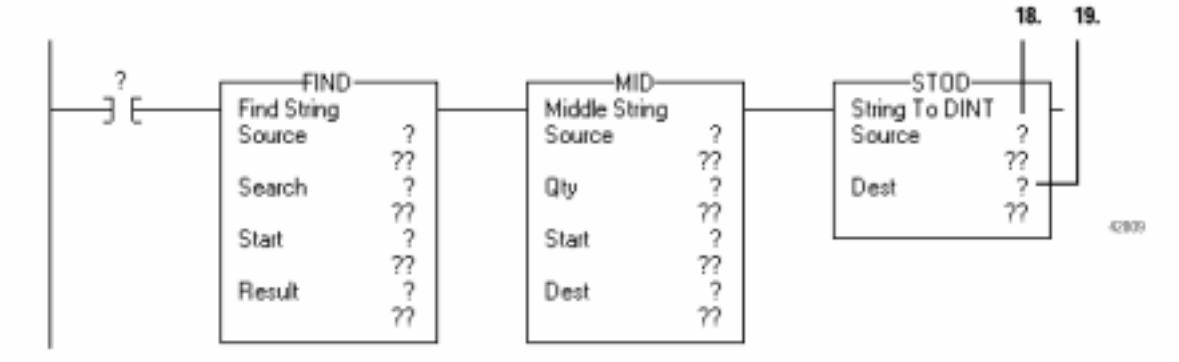
如果:	**那么**:
STOR	跳到第 15 步。
STOD	跳到第 18 步。



15. 输入保存数值的标签。(步骤 13 中的标签。)

16. 输入一个标签名，用来保存用于用户应用程序中的数值，定义数据类型为REAL实数。

17. 跳到第 20 步。



18. 输入保存数值的标签。(步骤 13 中的标签。)

19. 输入一个标签名，用来保存用于用户应用程序中的数值，定义数据类型为DINT双整型。

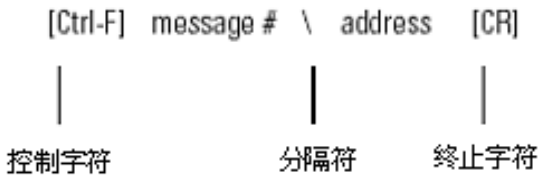
20. 这个字符串是否 有用户想使用的其它数值?

如果:	**那么**:
是	A.将 Find 指令的 Result 加 1。（步骤 9 中的标签） B.重复 2-19 步。
否	停止。用户已经完成这一步。

建立一个字符串

按照下面的步骤在用户应用程序中由变量建立一个字符串。这样用户就可以发送字符串到 ASCII 触发设备，如 MessageView 终端。

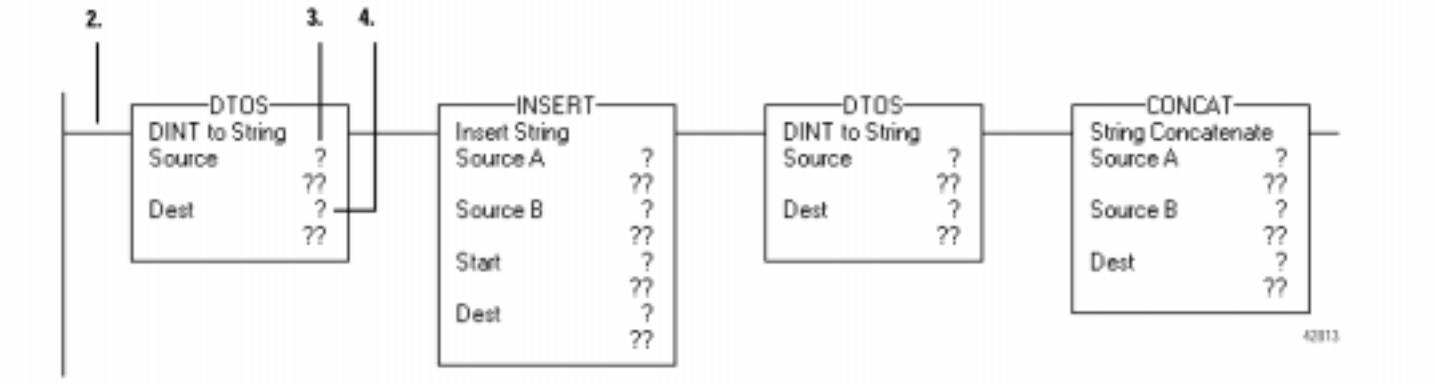
- 在这一步中，用户建立一个含有两个变量的字符串。例如，一个操作员终端可能需要一个象这样的字符串：



- 如果用户需要包括多个变量，使用另外的 INSERT 或 CONCAT 指令。
- 如果用户需要发送一个浮点值，使用 RTOS 指令代替 DTOS 指令。
- 最后的字符串将不包括终止字符。当用户发送字符串时，使用 AWA 指令来自动添加终止字符。

步骤：

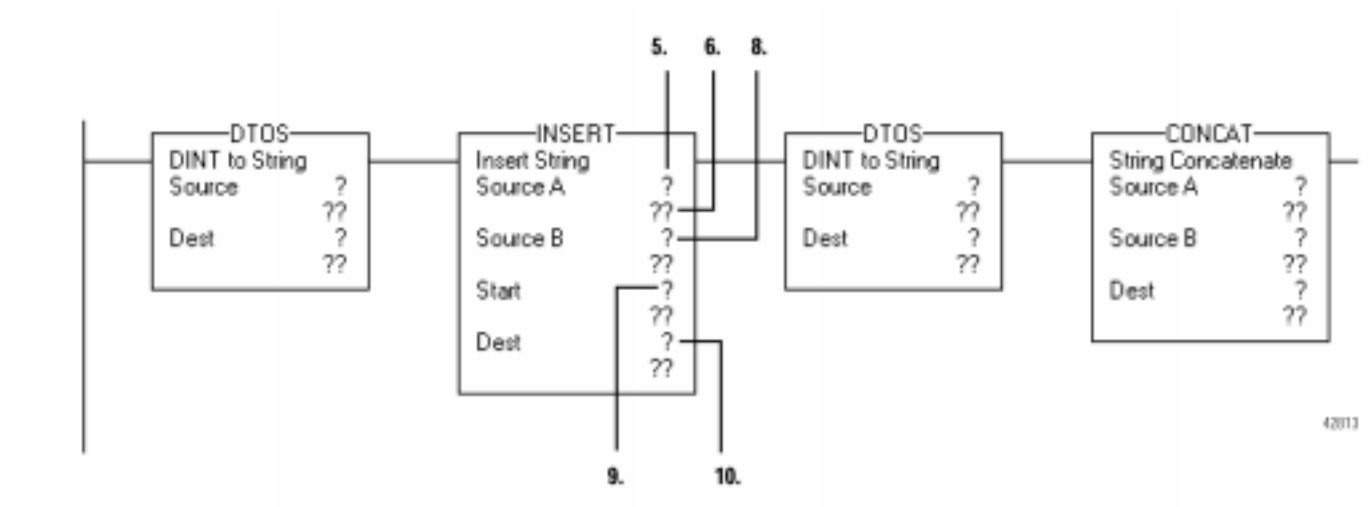
1. 输入下面的梯级：



2. 输入决定何时建立字符串的输入条件。

3. 输入含有字符串第一个值的 DINT 标签。

4. 输入一个保存表示 ASCII 值的标签名。定义数据类型为字符串。



5. 输入一个保存字符串的控制符和分隔符的标签名。定义数据类型为字符串。

6. 双击 Source A 的值域。

7. 输入控制符和分隔符，并选择 OK(完成)。

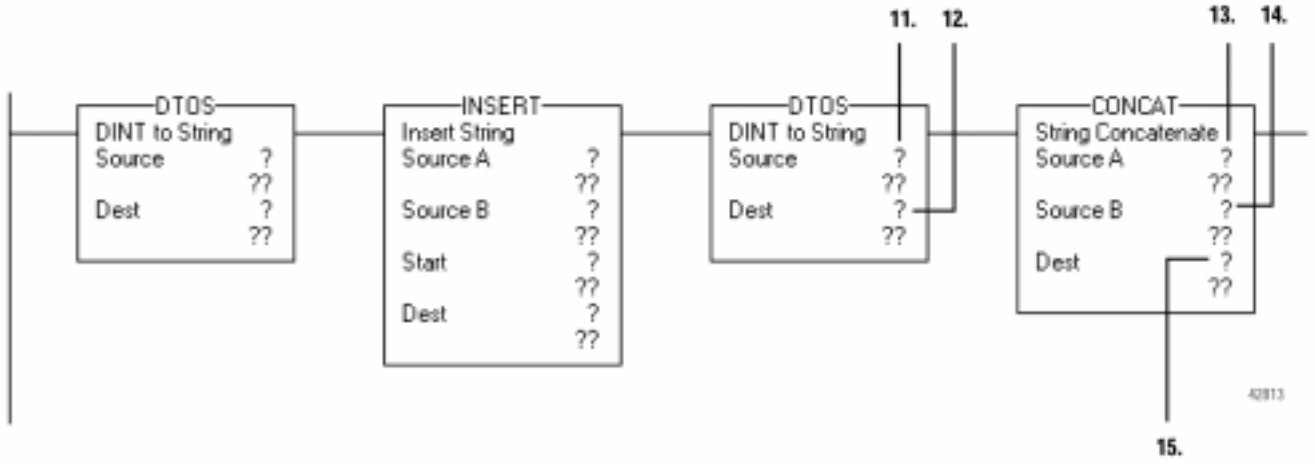
为一个控制符输入十六进制代码。要查看十六进制代码列表，参见本手册的封底。

8. 输入保存表示 ASCII 第一个值的标签。(步骤 4 中的标签)。

9. 输入 2.

在 Source A 中，将值放在首字符(控制符)的后面。

10. 输入一个用来存储部分完整的字符串的标签名。定义数据类型为字符串。



11. 输入含有字符串第二个值的 DINT 标签。

12. 输入一个保存表示 ASCII 值的标签名。定义数据类型为字符串。

13. 输入存储部分完整字符串的标签。(步骤 10 中的标签)。

14. 输入存储表示 ASCII 第二个值的标签。(步骤 12 中的标签)。

15. 输入一个用来存储完整字符串的标签名。定义数据类型为字符串。

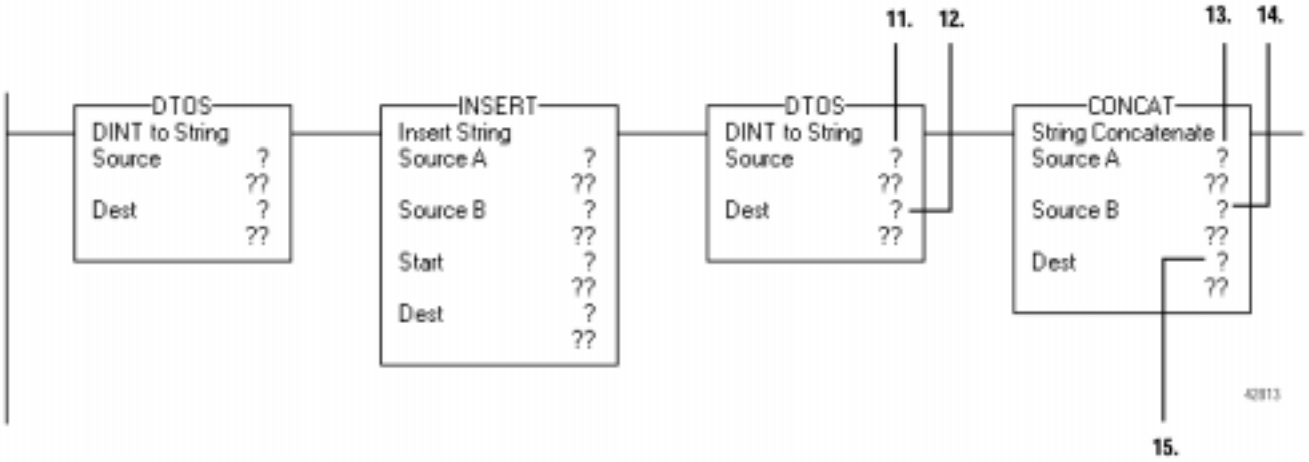
例　子

要给 MessageView 终端触发信息，控制器用如下格式向终端发送: [Ctrl-T] message # \ address [CR]

当 *send_msg* 闭合时，梯级进行如下操作:

- 第一个 DTOS 指令将信息的数量转换成 ASCII 码。INSERT 指令在控制字符[Ctrl-T]后面插入信息数量 (ASCII 形式)。(Ctrl-T 的十六进制代码是 $14)
- 第二个 DTOS 指令将终端节点数转换成 ASCII 码。
- CONCAT 指令将节点数(ASCII 形式)放在反斜线[\ ]后并将最终的字符串保存在 *msg* 中。

要发送信息，AWA 指令发送 *msg* 的标签并附加回车键[CR]。



Notes:

强置值

何时强置一个值

用一个强置值来替换一个输入或输出值:

如果用户想替换:	那么强置:	注意:
其它控制器生产型标签的值	消费型标签	强置一个输入或消费型标签:
输入设备的值	输入数据位或值	· 不管是物理设备的值还是生产型标签的值都可替换。 · 不影响其它监测输入或生产型标签的控制器接收到的值
用户逻辑和指定生产型标签的值	生产型标签	·强置一个输出或生产型标签来替换物理设备或其它控制器的逻辑。
用户逻辑和指定输出设备的状态	输出数据位或值	· 其它以只听能力监测输出模块的控制器也可看见强置值。

当用户强置一个值时:

- 用户可以强置除组态信息数据以外所有的 I/O 数据。
- 如果标签是一个数组或结构体，例如一个 I/O 标签，强置一个 BOOL,SINT,INT, DINT,或者 REAL 元素或成员。
- 如果数据值是一个 SINT,INT,或者 DINT，用户可以强置整个值或数值中的个别位。个别位可以有如下强置状态:
  - 无强置
  - 强置开
  - 强置关
- 用户也可以给一个 I/O 结构体成员，生产型标签或消费型标签的别名强置一个值。
  - 一个别名标签与它的基本标签共享相同的数据，因此强置一个别名标签也是强置一个与其相关的基本标签。
  - 删除别名标签的强置值就是删除与其相关的基本标签的强置值。

注　意

如果强置值被启用并且所有数据都被强置,操作人员必须离开机械所在的区域。强置功能能够产生不希望的机械运动，可能造成人员伤害。

重要提示 强置功能会增加梯形图逻辑的执行时间。用户强置的值越多，梯形图逻辑执行的时间就越长。

重要提示 强置功能遵守控制器规范，而不是编程工作站规范。即使编程工作站被切断，强置功能仍然维持。

输入一个强置值

从标签窗口的监视标签项或梯形图窗口输入一个强置值。

从标签口输入强置值

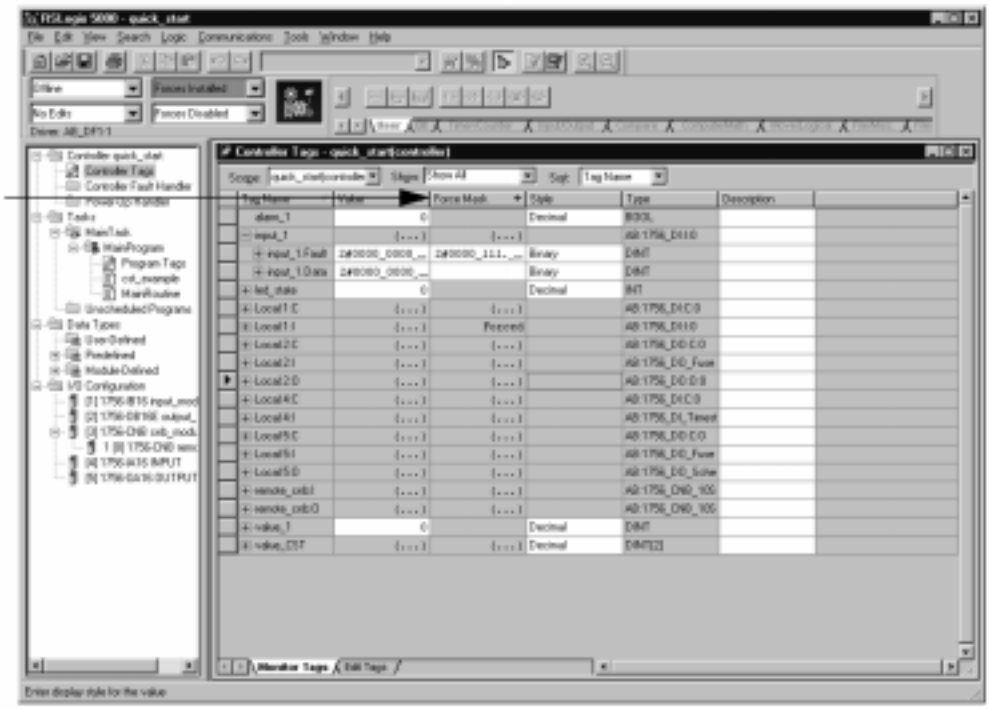
从标签窗口的监视标签项，用户可以用两种方法强置一个值，用户可以:

- 强置整个数据值。

  对于 SINT,INT,DINT 和 REAL 值，用户可以用强置一个整体（整个值）来强置所有的位。

- 强置 SINT,INT,或 DINT 值中的个别位。

从此栏输入强置值

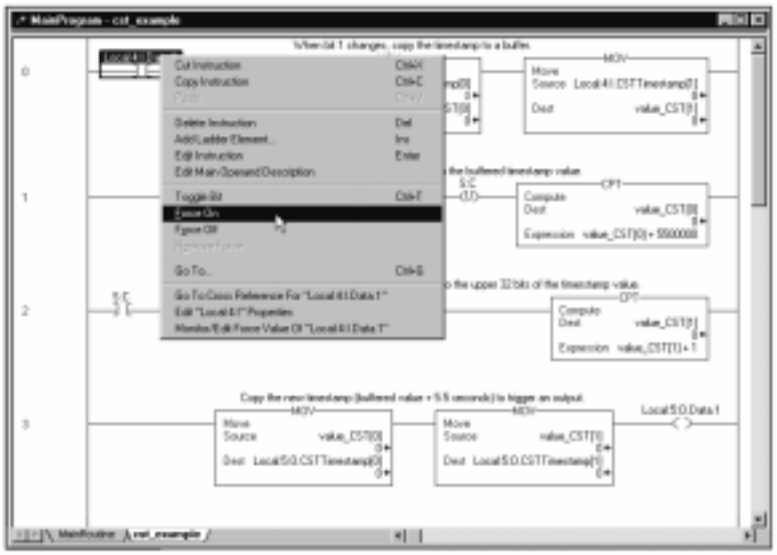


如果用户想:	操作如下:
强置整个 SINT,INT,DINT,或 REAL 值	若要强置整个值，在 Force Mask 栏内以十进制、八进制、十六进制、或浮点 / 指数形式输入强置值。 要删除整个强置值，输入一个空格。
强置数值中的位	强置SINT、INT或DINT值中个别位，展开该值并编辑Force Mask 栏。强置值以二进制显示，其中: • “0” 表示强置关 • “1” 表示强置开 • “.” 表示无强置 用户也可使用位标来选择一个位来强置。
强置一个 BOOL 值	要强置一个 BOOL 值，输入强置值，其中: • “0” 表示强置关 • “1” 表示强置开 要删除一个强置值，输入一个空格。

从梯形图逻辑输入强置值

在梯形图逻辑中，用户只能对位指令中的 BOOL 标签或整数文件的位进行强置。

在 BOOL 标签或位值上点击右键。选择 **Force On, Force Off,**或者 **Remove Force**



对于更复杂指令中的强置值，用户只能去掉强置。要对这些数强置值，用户对这些值必须使用数据监测来设置它们。

在强置值上点击右键。选择 **Remove Force.**

使能强置

用户通过使能强置，使强置有效。用户只能在控制器中使能或禁止强置。用户不能对一个特殊模块、标签集合或标签元素使能或禁止强置。

注　意

使能强置会导致输入、输出、生产型或消费型标签的值发生变化。操作人员必须离开机械所在的区域。强置功能能够产生不希望的机械运动，可能造成人员伤害。

用户从 Online Bar 中使能强置。

Forces Installed 表示已经输入强置值。

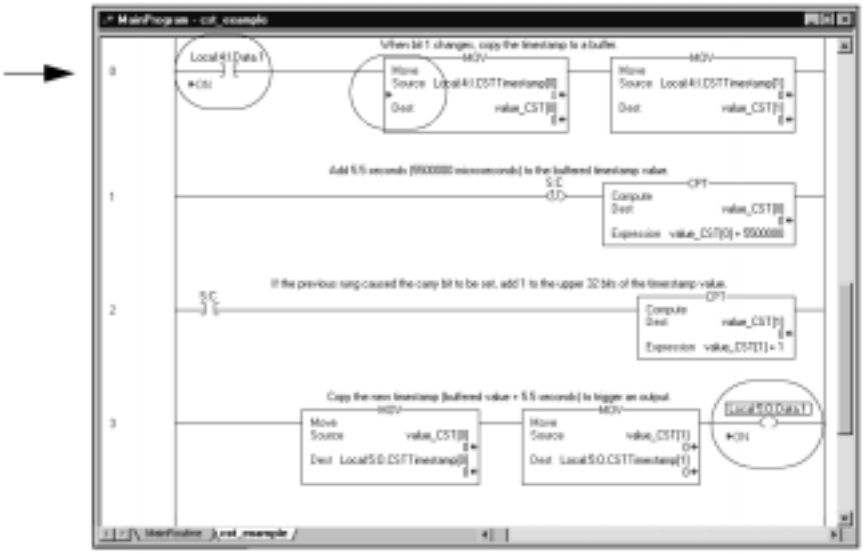
选择 Enable all forces

重要提示

如果用户要下载一个已使能强置的工程，在下载完毕后程序会提示用户使能或禁止强置。

当强置被使能时，在梯形图编辑框中一个>将出现强置值前面。

当强置使能时，梯形图编辑框中显示哪些强置被使能。



禁用强置值

用户可以不通过去掉个别值或控制器的强置值来禁用强置。强置禁用后，工程仍可以按已编制的程序执行。虽然已经输入强制值，但它们并不执行。

选择 Disable all forces.

去掉强置值

用户可从个别值或整个控制器上去掉强置值。

用户可以从数据监测窗口去掉个别的强置。

如果用户要去掉强置:	操作如下:
整个 SINT,INT,DINT,或 REAL 值	在数据监测窗口的值上单击右键并选择 **Remove Force**
数据位	打开值并编辑 Force Mask 栏。将数据位的值改为 ″.″ 以表示无强置。
BOOL 值	输入空格。

如果强置是加在一个BOOL标签或者数据位上，用户也可以从梯形图编辑器上去掉强置值，在数据位上点击右键选择 **Remove Force**.

如果用户逐个的去掉每个强置值，强置值仍可被使能。

注 意

如果用户已去掉强置，但强置仍被使能，用户设置强置值则会立即生效。操作人员必须离开机械所在的区域。强置功能能够产生不希望的机械运动，可能造成人员伤害。

去掉别名标签的强制值也就去掉了基本标签的强制值。

用户可从控制器级别上去掉所有的强置。去掉所有的强置值可使禁用强置并清除所有的强置掩码值。

选择 **Remove all forces.**

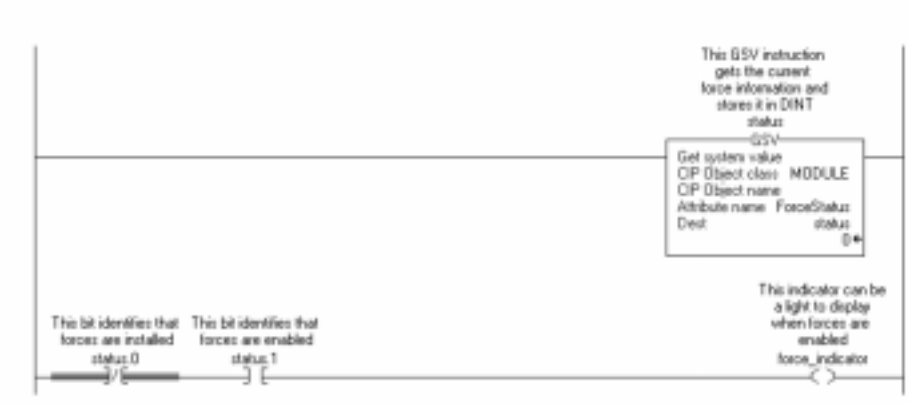
监视强置值

用户可以通过下列方式来监测强置状态:

- RSLogix5000 软件
- 应用程序的逻辑
- FORCE LED.(Logix5550 控制器没有 LED 显示强置状态。)

如果 **FORCE LED** 是:	那么:
灭	• 没有含强置值的标签 • 强置无效 (禁用)
闪烁	• 至少一个标签含有强置值。 • 强置值无效。
亮	• 强置有效 (使能) • 强置值可能存在也可能不存在

下面的例子显示如何检查强置是否存在，并被使能以及如何设置用户自己的LED指示器。



Note:

开发一个故障例程

使用本章

如果出现一个严重的故障使控制器关闭，则控制器产生一个主要故障信息并停止执行梯形图逻辑。

- 根据用户的应用，并不想让所有的主要故障都关闭整个系统。
- 在这种情况下，用户可以用一个故障例程来清除具体的故障，至少使用户系统的某些部分继续运行。

例　子

使用一个故障例程

在系统中使用配方号作为间接地址，一个错误类型的数字可能产生一个主要故障，例如，类型 4，代码 20。

- 要保证整个系统不被关闭，故障例程必须清除任何类型 4，代码 20 的主要故障。

如何使用本章

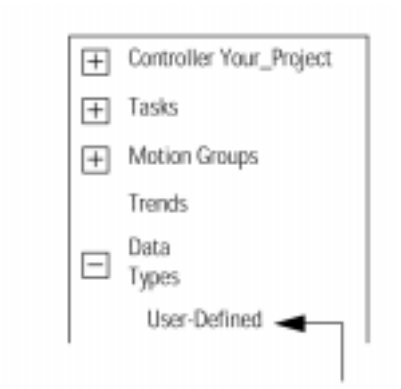
开发一个故障例程:

- 创建 FAULTRECORD 数据类型
- 创建一个故障例程
- 清除主要故障
- 在预扫描期间清除主要故障
- 测试故障例程

创建 FAULTRECORD 数据类型

创建如下自定义数据类型，用它保存故障的相关信息。

创建一个新的数据类型:



右键点击并选择 *New Data Type*（新建数据类型）

Type de données :FAULTRECORD				
Nom	FAULTRECORD			
Description	Stocke les attributs MajorFaultRecord ou MinorFaultRecord de l'objet PROGRAM.			
Membres				
	Nom	**Type de données**	**Style**	**Description**
	Time_Low	DINT	Décimal	32 bits inférieurs de la valeur d'horodatage du défaut
	Time_High	DINT	Décimal	32 bits supérieurs de la valeur d'horodatage du défaut
	Type	INT	Décimal	type de défaut (programme, E/S, etc)
	Code	INT	Décimal	code du défaut
	Info	DINT[8]	Hex	informations spécifiques au défaut

创建一个故障例程

故障例程允许用户使用梯形图逻辑清除指定的故障并使控制器继续执行。例程的位置取决于用户想清除故障的类型:

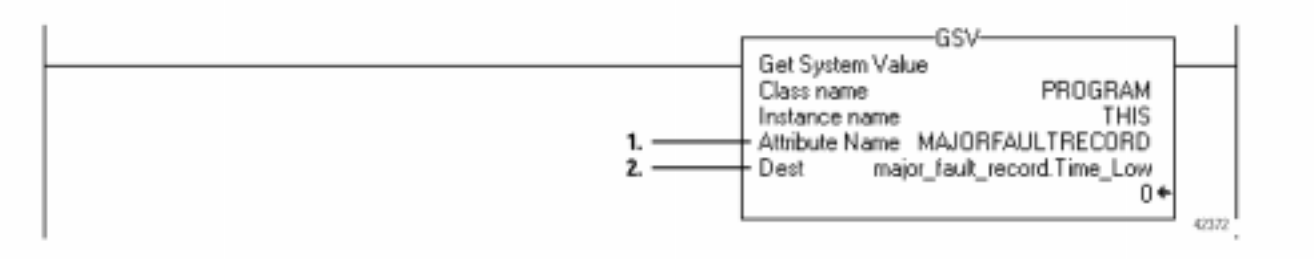
故障是关于:	操作如下:
指令的执行	为程序创建一个故障例程: A. 在控制器项目管理器中，右键点击 *name_of_program* 并选择 *New Routine（新建例程）.* B. 在名称框中，为故障例程输入一个名字 *（name_of_routine）.* C. 从 *Type*（类型）下拉列表中，选择 *Ladder*（梯形图）. D. 点击 *OK（完成）.* E. 右键点击 *name_of_program* 并选择 *Properties（属性）.* F. 点击 *Configuration（组态）*选项。 G. 从 *Fault （故障）*下拉列表，选择 *name_of_fault_routine* H. 选择 *OK（完成）.*
掉电 I/O 任务看门狗 模式转换 运动轴.	为 *Controller Fault Handler*（控制器故障处理器）创建一个程序和主例程: A.在控制器项目管理器中，右键点击 *Controller Fault Handler 并 New Program（新建程序）* B.输入: • *name_of_program* • *description（描述）(任选)* C.点击 *OK（完成）.* D.点击 *Controller Fault Handler（控制器故障处理器）*前的＋标志. E.右键点击 *name_of_program* 并选择 *New Routine（新建例程）.* F.输入: • *name_of_program* • *description（描述）(任选)* G.从 Type（类型）下拉列表中为例程选择编程语言。 H.点击 OK（完成）。 I. 右键点击 *name_of_program* 并选择 *Properties（属性）*。 J. 点击 *Configuration（组态）*选项。 K. 从 *Main （主）*下拉列表中选择 *name_of_routine.* L. 点击 *OK（完成）.*

清除主要故障

要清除在用户工程执行过程中出现的主要故障，在对应的故障例程中输入以下梯形图逻辑。（参见 15-3 页的创建故障例程。）

- 获得故障类型和代码
- 检查具体的故障
- 清除故障

获得故障类型和代码

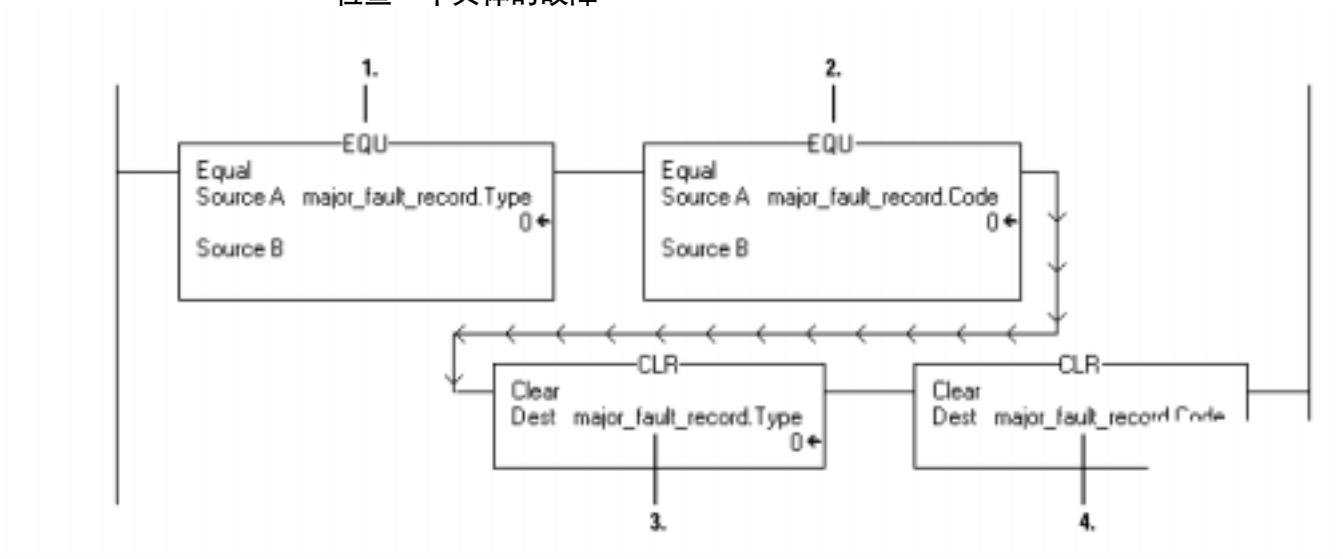


1. GSV指令访问本程序的MAJORFAULTRECORD属性。这个属性保存的是故障的相关信息。

2. GSV 指令将故障信息保存在 *major_fault_record* 标签中。当用户输入一个结构体标签时，应输入标签的第一个成员。

标签名	类型
major_fault_record	FAULTRECORD

检查一个具体的故障



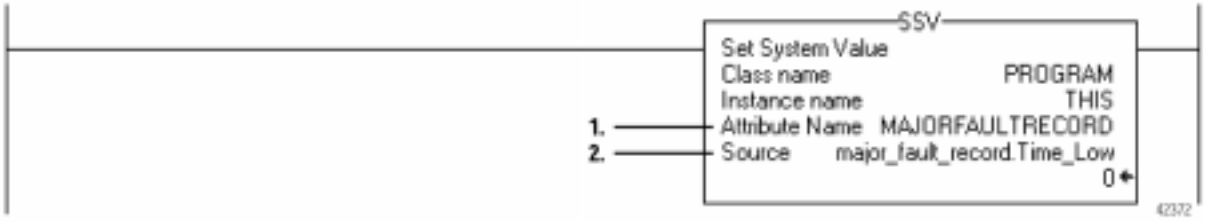
1. EQU 指令用于检查具体的故障类型，如程序，I/O。在 Source B 中输入用户想要清除的故障类型值。

2. EQU 指令用于检查指定故障代码。在 Source B 中，输入用户想要清除的代码值。

3. CLR 指令将 *major_fault_record* 标签中的故障类型值置零。

4. CLR 指令将 *major_fault_record* 标签中的故障代码值置零。

清除故障



1. SSV 指令给程序的 MAJORFAULTRECORD 属性写入新的值。

2. SSV 指令将值写入到 *major_fault_record* 标签中。因为 *Type（类型）*和 *Code（代码）*成员设置为零，所以故障被清除，控制器继续执行。

预扫描期间清除故障

如果在用户将钥匙开关打到Run模式之后控制器立即发生故障,那么检查故障预扫描操作。例如，预扫描检查间接地址（一个作为数组元素指针的标签）。

- 如果一个间接地址在运行时就初始化，对数组在预扫描期间来说可能太大。
- 如果控制器在预扫描期间发现一个超出范围的间接地址,就会产生一个主要故障。
- 为了使控制器完成预扫描，采用程序的故障例程来中断并清除故障。

要清除预扫描期间发生的主要故障:

- 确认控制器处在预扫描的状态。
- 获得故障类型和代码。
- 检查指定故障
- 清除故障。

确认控制器处在预扫描状态

在用户程序主例程中，输入以下梯级:

1. 在程序主例程中输入此梯级并将其作为第一梯级。

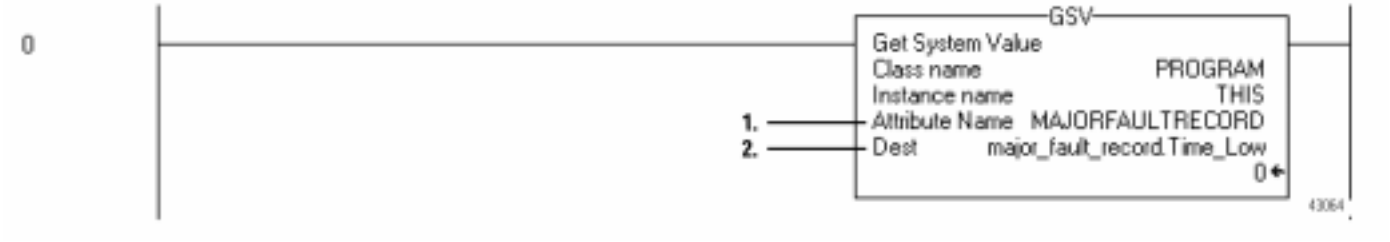
2. 程序主故障例程通过此位的状态确定故障是发生在预扫描阶段还是在梯形图逻辑的正常扫描阶段:

- 在预扫描阶段，此位是断开的。（在预扫描阶段，控制器将 OTE 指令涉及的所有位复位。）
- 一旦控制器开始执行梯形图逻辑，此位将一直保持闭合状态。

标签名	类型
CPU_scanning	BOOL

获得故障类型和代码

在程序故障例程中输入以下梯级:



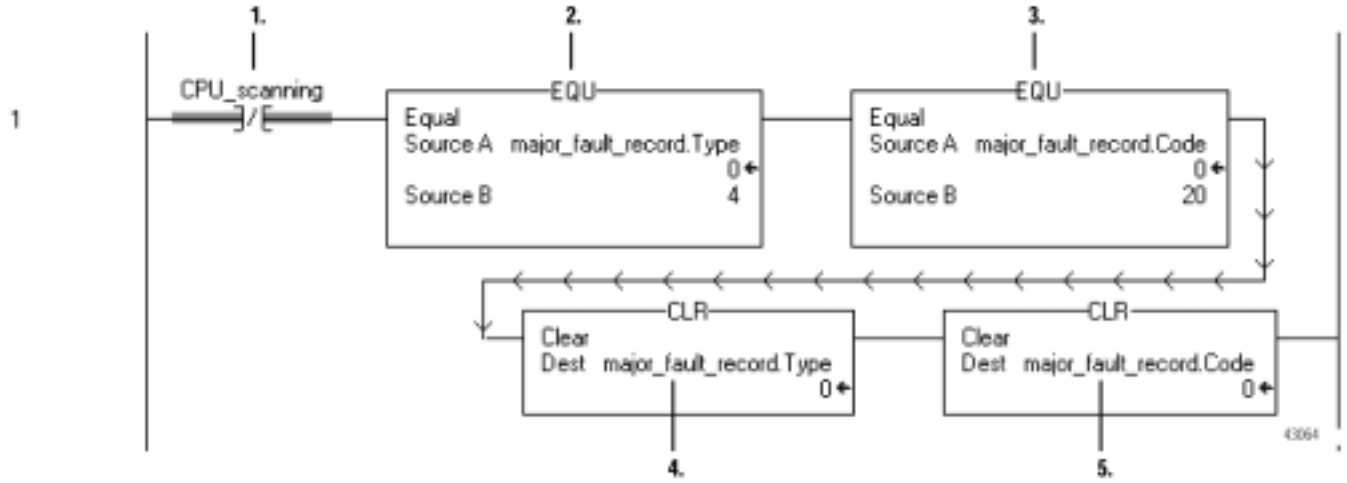
1. GSV 指令访问本程序的 MAJORFAULTRECORD 属性。这个属性保存的是故障的相关信息。

2. GSV 指令将故障信息保存在 *major_fault_record* 标签中。当 用户输入一个结构体标签时，应输入标签的第一个成员。

标签名	类型
major_fault_record	FAULTRECORD

检查一个指定故障

在程序的故障例程中输入一个梯级：



1. 在预扫描期间所有 OTE 指令的位都是断开的，而这条指令为真。一旦控制器开始执行梯形图逻辑，这条指令就一直为假。

2. EQU 指令检查类型为 4 的故障，它意味着程序中有一条指令产生了故障。

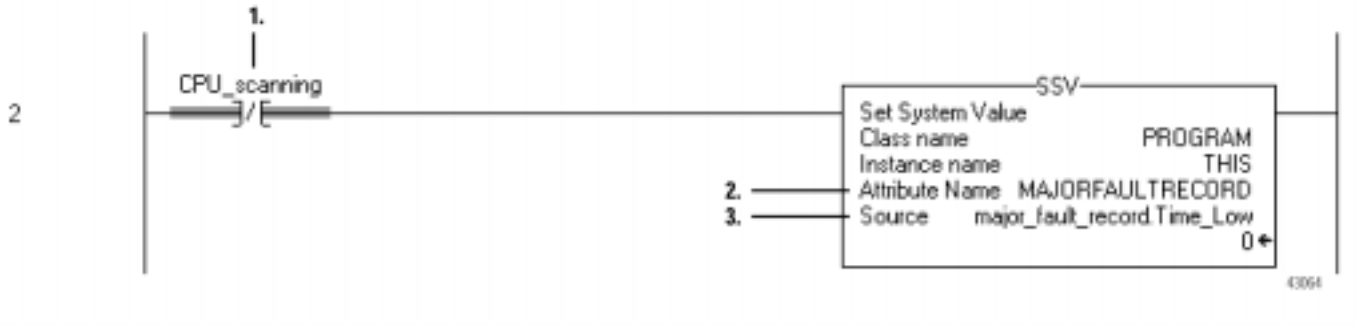
3. EQU 指令检查代码为 20 的故障，它意味着不是数组下标太大，就是 CONTROL 结构体的 POS 或 LEN 值无效。

4. CLR 指令将 *major_fault_record* 标签中保存的故障类型值置零。

5. CLR 指令将 *major_fault_record* 标签中保存的故障代码值置零。

清除故障

在程序故障例程中输入以下梯级:



1. 在预扫描期间所有OTE指令的位都是断开的，而这条指令为真。一旦控制器开始执行梯形图逻辑，这条指令就一直为假。

2. SSV 指令给程序的 MAJORFAULTRECORD 属性写入新的值。

3. SSV 指令将值写入到 *major_fault_record* 标签中。因为 *Type（类型）*和 *Code (代码)*成员设置为零，所以故障被清除并且控制器继续执行。

测试故障例程

用户可以利用JSR 指令测试程序的故障例程而不产生错误 (例如，模拟一个故障)：

1. 创建一个 BOOL 标签，用户将用其启动故障。

2. 在程序的主例程或子例程中，输入以下梯级：

其中：	是：
aaa	用户用于启动故障的标签 （步骤 1）
bbb	程序的故障例程

3. 设置输入条件来模拟故障，。

例　子　测试一个故障例程

当 *test_fault_routine* 闭合时，主要故障发生且控制器执行 *Fault_Routine*。

创建自定义主要故障

使用本章

如果用户想在应用程序中基于条件来中止（关断）程序的运行，用户可以创建一个自定义的主要故障。使用自定义主要故障:

- 用户定义一个故障代码值
- 控制器处理自定义故障时，和处理其它主要故障一样:
  - 控制器变成 faulted mode (故障模式) 并且停止执行梯形图逻辑。
  - 输出被设成组态状态或故障模式下的值。

例 子 自定义主要故障

当 *input_value* 大于 80，产生一个主要故障并生成一个值为 999 的故障代码。

创建自定义主要故障

1. 程序中是否存在一个故障例程?

如果:	**那么**:
是	进行第 2 步
否	给程序创建一个故障例程: A. 在控制器项目管理器中，右键点击 *name_of_program*，选择 *New Routine (新建例程)*。 B. 在名称栏中，输入故障例程的名字 *(name_offault_routine)* C. 在 *type (类型)* 的下拉菜单中，选择 *ladder (梯形图)* D. 单击 *OK (完成)* E. 右键点击 *name_of_program*，选择 *properties（属性）.* F. 选择 *configuration（组态）*选项 G. 从 fault (故障) 下拉菜单中，选择 name_of_fault_routine H. 单击 *OK (完成)* I. 双击 *name_of_fault_routine* J. 输入一个 NOP 指令 (此时程序校验没有错误)

2. 在程序主例程中输入以下梯级:

其中:	**是**:
name_of_fault_routine	步骤 1 中的例程
x	故障代码值

例　子　创建自定义主要故障

当 *input_value* 大于或等于 80 时，程序执行跳转到 *name_of_fault_routine*。一个主要故障产生并且控制器进入故障模式。输出进入故障状态。在控制器属性对话框的主要故障选项中，显示故障代码为 999。

监测次要故障

使用本章

当一个故障的严重程度不足以关闭控制器时，那么控制器产生一个 **minor fault（次要故障）**。

- 控制器继续运行。
- 无须清除次要故障。
- 为了优化运行时间和确保程序的精确性，用户应该监测并校正次要故障。

监测次要故障

使用梯形图逻辑来获取次要故障的信息:

要检查一个:	**操作如下**:
周期性任务的重叠	1. 输入 GSV 指令，通过此指令获取 *FAULTLOG* 对象的 *MimorFaultBits 属性*。 2.监测第 6 位
从非易失性内存中读取	1. 输入 GSV 指令，通过此指令获取 *FAULTLOG* 对象的 *MimorFaultBits 属性*。 2.监测第 7 位
串行端口的问题	1. 输入 GSV 指令，通过此指令获取 *FAULTLOG* 对象的 *MimorFaultBits 属性*。 2.监测第 9 位
电池低电压	1. 输入 GSV 指令，通过此指令获取 *FAULTLOG* 对象的 *MimorFaultBits 属性*。 2.监测第 10 位
指令的问题	1.创建自定义的数据类型，用来保存故障信息。数据类型的命名为 *FaultRecord*，并指定以下成员:

名称:	**数据类型**:	**格式**:
TimeLow	DINT（双整型）	十进制
TimeHigh	DINT（双整型）	十进制
Type	INT（整型）	十进制
Code	INT（整型）	十进制
Info	DINT[8]	十六进制

2.创建一个标签用来保存 *MinorFaultRecord* 属性的值。从步骤 1 中选择数据类型。
3.监测 S:*MINOR*
4.如果 *S:MINOR* 位为真，使用 GSV 指令获取 MinorFaultRecord 属性的值。
5.如果用户想检测由另一个指令引起的次要故障，将 *S:MINOR* 复位。（S:MINOR 将保持设置直到扫描结束。）

下面是一个检查电池低电压报警的例子。

例　子　检查次要故障

*Minor_fault_check*计时一分钟 (6000ms)，然后自动重新计时。

每隔一分钟 *Minor_fault_check.DN*闭合一次。当 *Minor_fault_check.DN*闭合时，GSV 指令获取 *FAULTLOG* 对象的 *MinorFaultBits* 属性的值，并且将其保存在 *minor_fault_bits* 标签中。因为 GSV 指令一分钟只执行一次，所以最大扫描的扫描时间减少了。

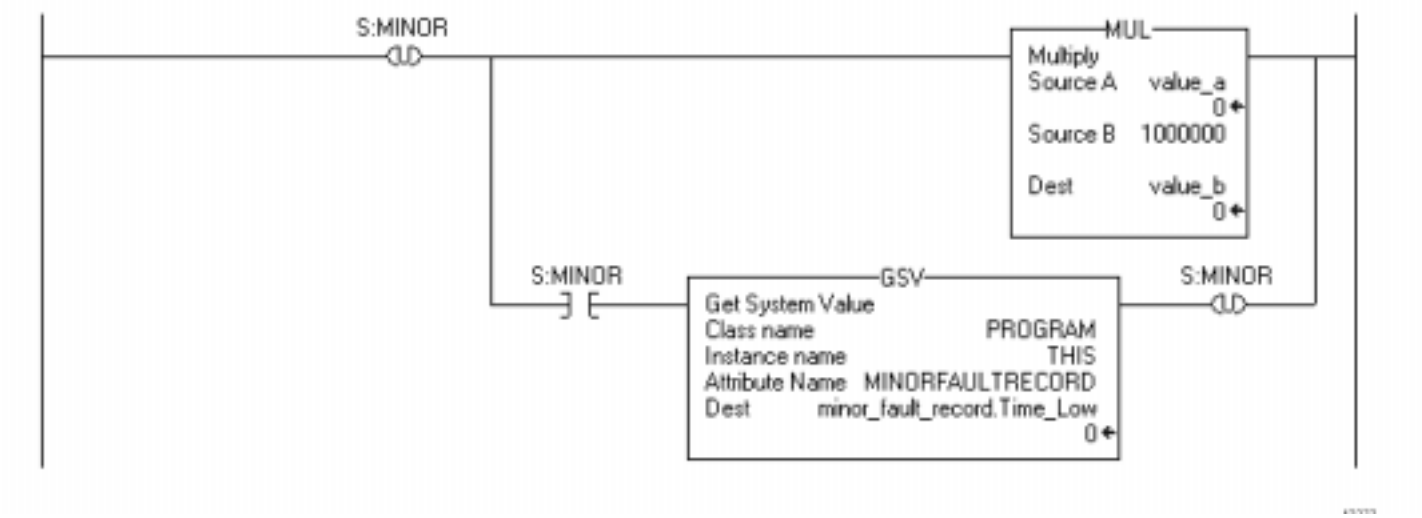
如果 *minor_fault_bits.10* 闭合，则电池低电压。

下面的例子是检查由具体指令产生的次要故障。

例　子　检查由指令引起的次要故障

value_a 和 1000000 相乘并检查次要故障，例如计算溢出：

- 确定前面的指令没有产生这一故障，梯形图首先对 S：MINOR 复位。
- 然后执行乘法指令。
- 如果指令产生这一次要故障，则控制器会对 S：MINOR 置位。
- 如果 S：MINOR 被置位，那么 GSV 指令就会获取关于故障的信息，然后将 S：MINOR 复位。

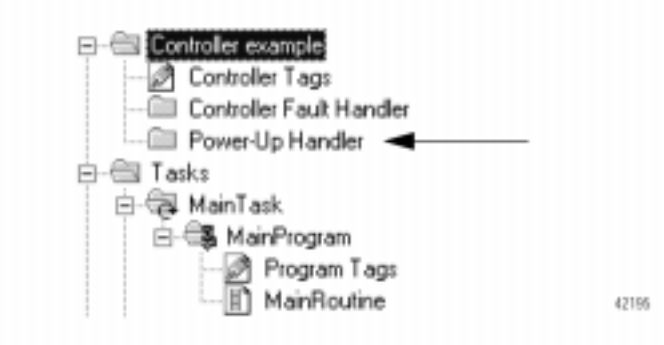


Note:

开发一个上电例程

使用本章

上电处理程序是一种可选任务，当控制器在 Run 方式下上电时，则执行该程序。



用户想在掉电后，完成接下来的程序并重新保存数据，这时使用上电处理程序。

- 防止控制器回到运行模式。
  - 上电处理将产生一个类型为 1，代码为 1 的主要故障，并且控制器会进入故障模式。
- 采取具体的操作并将梯形图逻辑恢复到正常运行状态。

开发一个上电例程

开发一个上电例程步骤和开发一个故障例程的步骤类似：创建一个自定义数据类型，用来保存故障信息。数据类型的命名为 *FaultRecord* 并且指定下列成员：

名称：	数据类型：	格式：
TimeLow	DINT（双整型）	十进制
TimeHigh	DINT（双整型）	十进制
Type	INT（整型）	十进制
Code	INT（整型）	十进制
Info	DINT[8]	十六进制

2. 创建一个标签用来保存故障信息。选择 *FaultRecord* 数据类型。

3. 创建一个上电处理程序:

操作:	**具体步骤**:
1. 创建一个程序。	A.在控制器项目管理器中，右键点击 *Power-Up Handler*并选择 *New Program(新建程序)*。 B. 输入: •*name_of_program* •*description(描述)*(可选) C. 选择 *OK (完成)*。
2. 创建并指定一个主例程 (这个例程在程序中第一个执行)	A.点击 Power-handler 前的＋标志。 B.右击 *name_of_program* 并选择 *New Routine (新建例程)*。 C.输入: •*name_of_program* •*description(描述)*(可选) D.在 *Type (类型)*的下拉菜单中，给例程选择编程语言。 E.选择 *OK(完成)*。 F.右键点击 *name_of_program* 并选择 Properties（属性）。 G.选择 *Configuration (组态)*选项。 H. 从 *Main (主)* 下拉菜单中，选择 *name_of_main_routine*。 I. 选择 *OK (完成)*。 J. 如果要给程序添加另外的例程 (子程序)，重复步骤 B 到 E。

3. 用户如何处理掉电?

要:	**操作如下**:
防止控制器回到运行模式完成。	当电源恢复，一个类型为 1，代码为 1 的主要故障就会产生，并且控制器会进入故障模式。
当电源恢复时，采取具体的操作并且恢复正常运行	A.打开（双击） *name_of_routine* B. 给操作输入梯形图逻辑

4. 输入以下梯形图逻辑来清除故障:

获取故障信息并将其保存在 *major_fault* 标签中 (自定义结构)

将 *major_fault* 标签中故障类型和代码设置成 0，并且将 *MAJORFAULTRECORD* 赋予新值，这样故障被清除。

其中:

major_fault 是步骤 2 中的标签。

Note:

用非易失性内存保存和加载工程

使用本章

目前，仅有 1756-L55M23 和 1756-L55M24 具有保存工程的非易失性内存。在未来将会有更多的 Logix5000 系列的控制器具有这一功能。

重要提示 在用户保存工程时，非易失性内存也保存用户内存的内容。

- 在保存工程之后所进行的修改不会影响非易失性内存。
- 如果用户要保存如在线编辑、标签值或者 ControlNet 网络规划，应在用户修改后保存工程。

这一过程是采用控制器的**非易失性内存**来**保存**或**加载**一个工程。

- 如果控制器掉电并且电池电量不足，则会丢失在用户内存中的工程。
- 非易失性内存使用户在控制器上保存了一份工程的备份。控制器不需要用电源来保持这份备份。
- 用户可以从非易失性内存中将备份复制到控制器的用户内存中：
  - 在每次上电时
  - 控制器中没有程序并且上电时
  - 每次使用 RSLogx 5000 软件时

保存或加载下列参数:

表 19.1 保存或加载的参数:

参数:	**保存**:	**加载**:
保存或加载需要花费多少时间?	几分钟	几秒钟
在哪种控制器模式下可以存储或加载工程?	程序模式	
控制器保存或加载时是否可以上线?	不可以	
在保存或加载时 I/O 处于何种状态?	I/O 保持编程模式中的组态状态	

如何使用本章

如果用户想要:	**那么**:
在控制器的非易失性内存中保存一个工程	参见 19-3 页的 ″保存一个工程″
用保存在控制器非易失性内存中的程序覆盖控制器中目前的程序	参见 19-6 页的 ″加载一个程序″
因为没有电池，所以掉电清除内存后加载工程	
用梯形图逻辑来标记加载用户工程的非易失性内存	参见 19-9 页的 ″检查加载内容″
从控制器的非易失性内存中删除一个工程	参见 19-10 页的 ″清除非易失性内存″

保存一个程序

在这个任务中，用户可以在控制器的非易失性内存中保存一个程序。这会将非易失性内存中当前的工程覆盖。

注 意

在保存过程中，所有运行的伺服轴都必须停止。在用户保存工程以前，确保伺服轴不会产生意外的运动。

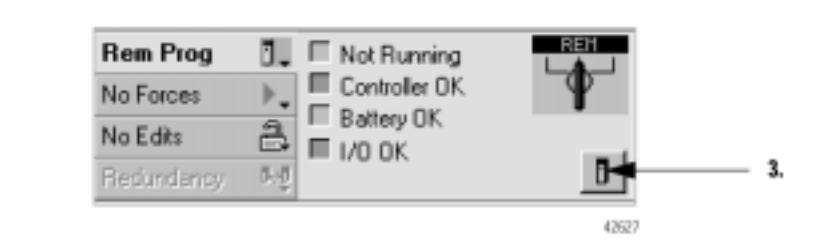
在用户保存工程之前:

- 将所有必要的编辑写到梯形图逻辑中
- 将工程下载到控制器

步骤:

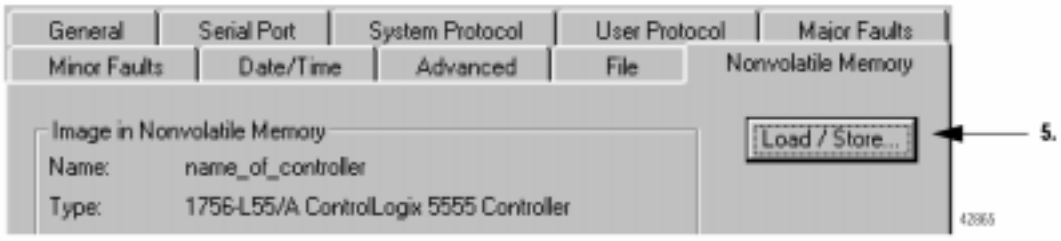
1. 将控制器上线

2. 将控制器置于编程模式(远程或编程)

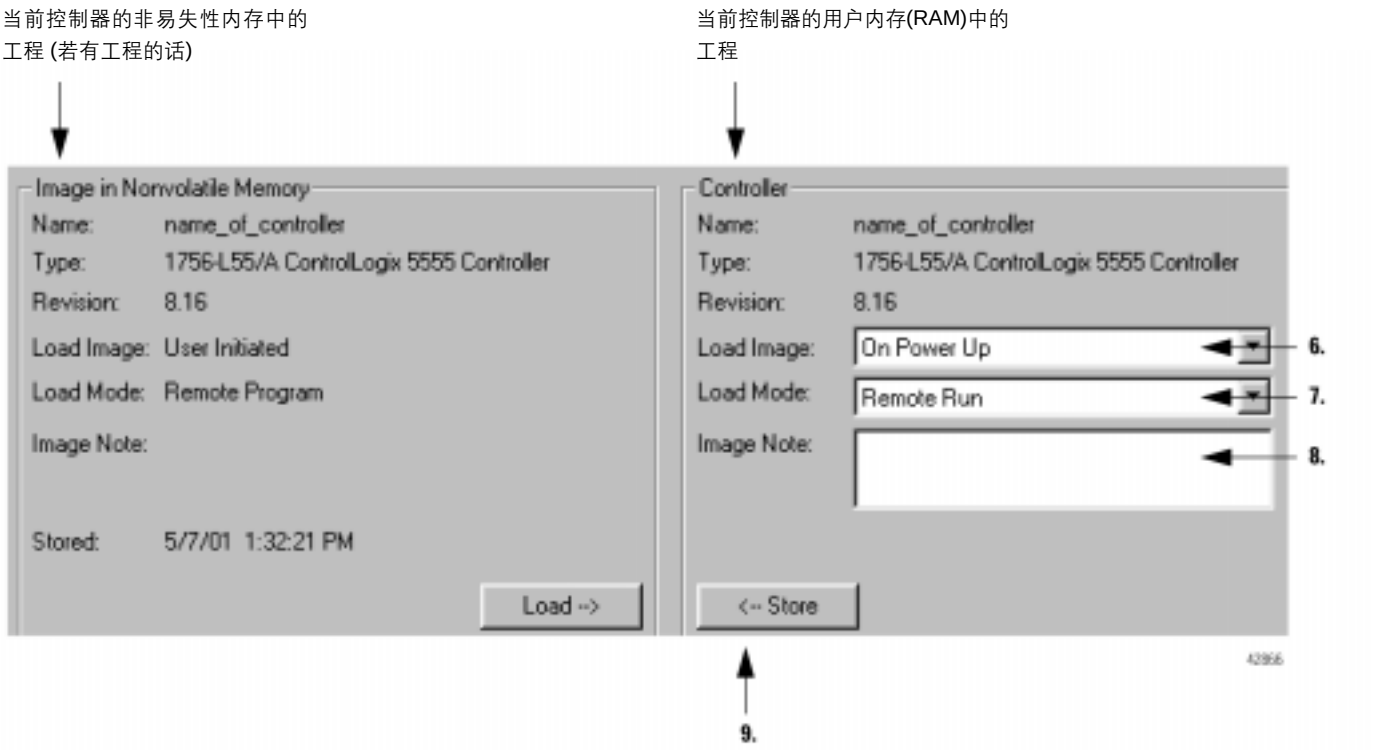


3. 从在线工具栏中，点击控制器属性按钮

4. 点击 *Nonvolatile Memory (非易失性内存)* 选项



5. 选择 *Load/Store (加载 / 保存)*。



6. 用户想要在何时 （在何种情况下） 将工程加载到控制器用户内存中?

如果用户想要加载它:	则选择:	备注:
用户上电或周期背板供电	*On power up*	• 周期性供电时，用户会丢失所有周期供电的尚未存于非易失性内存中的在线修改、标签值和网络规划。 • 在此选项下，当用户更新控制器的固件时工程仍可被加载。加载结束之后，从 Load Mode 模式下拉菜单中，控制器自动进入步骤 7 中。 • 用户可以经常使用 RSLogix5000 加载工程。
用户上电或周期背板供电	*On current memory*	•在此选项下，当用户修改控制器的固件时工程仍可被加载。加载结束之后，控制器会自动进入从 Load Mode 下拉菜单中 (步骤 7 中) 选择的模式。 • 用户可以经常使用 RS Logix5000 加载程序。
仅使用 RSLogix5000 软件	*User initiated*	

7.在步骤 6 中，用户选择了哪种加载映射选项?

如果	那么:
On power up	选择在加载后控制器的模式:
On current memory	• 远程编程 • 远程运行状态 要确定加载后的模式，将钥匙开关打到 REM 位置。
User Initiated	跳到步骤 8。

8.必要的话，在保存工程时输入一个描述。

9. 选择<-*Store*
对话框将问用户是否确认保存。

10.要保存工程，选择 *Yes*。

在保存过程中，会出现下列情况:

- 控制器前面的 OK 指示灯会以下列顺序显示:
  闪烁: 绿 ⇒ 红 ⇒ 绿
- RSlogix 5000 软件离线。

11.选择 *OK*。

在保存结束之后，依然处于离线状态。如果用户想上线，需要手动上线。

加载一个工程

在这个任务中，用户使用 RSLogix5000 软件从非易失性内存中加载工程。

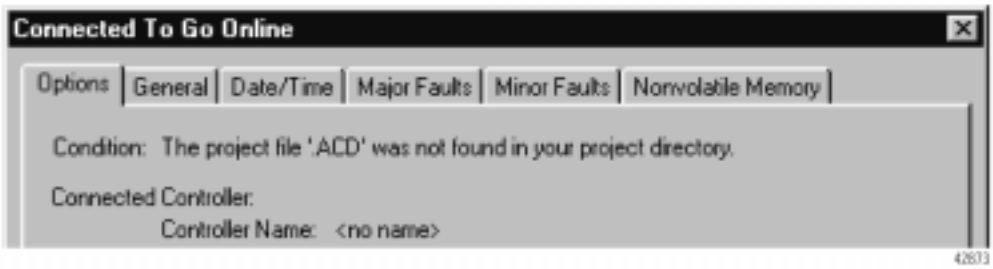
注 意:

在加载过程中，所有运行的伺服轴都必须停止。在用户保存工程以前，确保伺服轴不会产生意外的运动。。

步骤:

1.控制器上线。

2. 是否打开以下对话框?

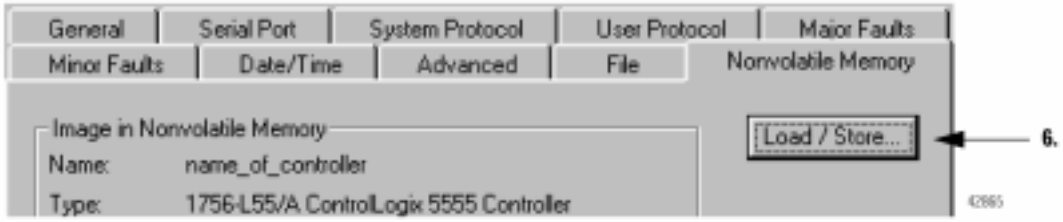


如果:	那么:
否	跳到步骤 3
是	跳到步骤 5

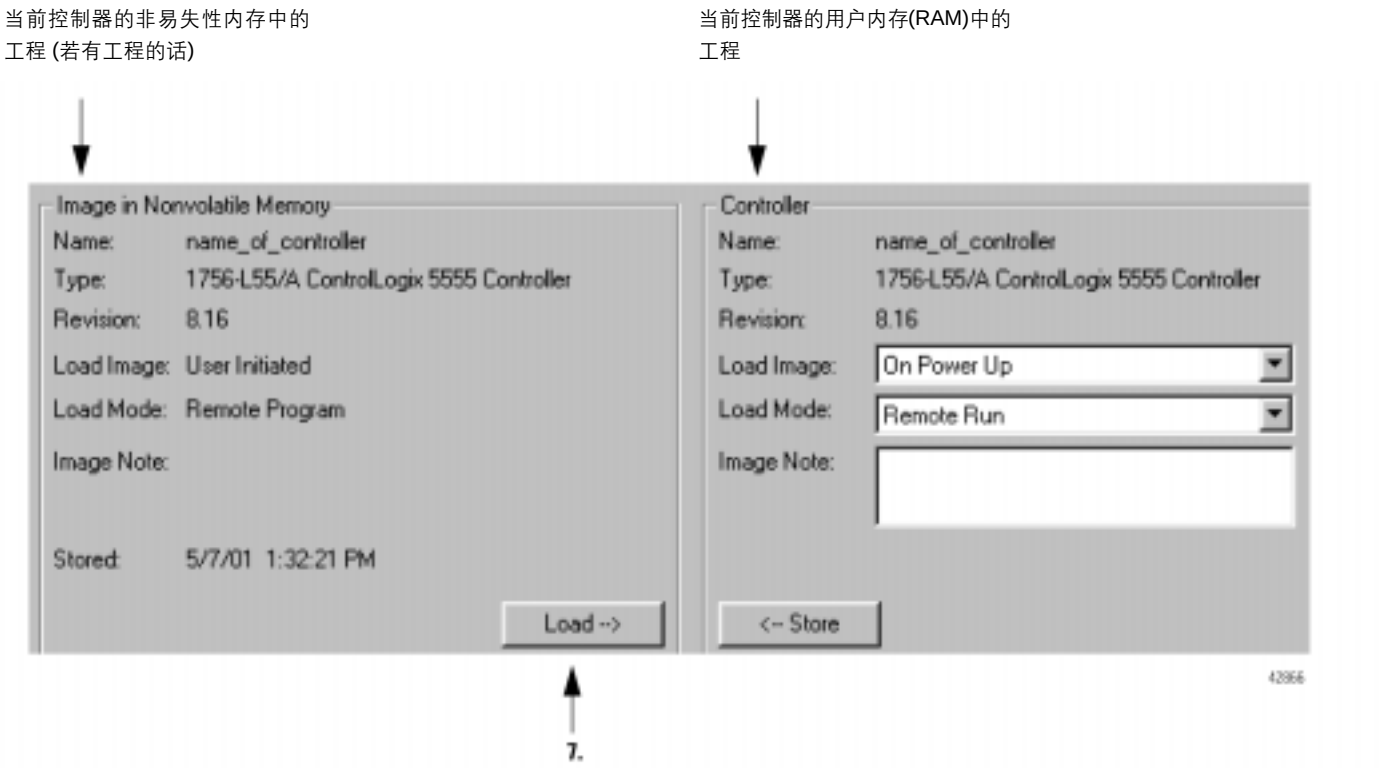
3. 将控制器打到编程模式 (远程或编程)。

4. 在上线工具栏中，点击控制器属性按钮。

5. 选择 *Nonvolatile Memory (非易失性内存)* 选项。



6. 选择 *Load/Store* (加载 / 保存)。



7. 选择 *Load->*。

对话框将提示用户是否确认加载。

8. 要从非易失性内存中加载工程，选择 *Yes*。

在加载过程中，会出现下列情况：

- 控制器前的 OK 指示灯会按照下列顺序显示

  红 ⇒ 绿

- RSLogix5000 软件离线。

当加载完成后，依然处于离线状态，如果用户想上线，需要手动上线。

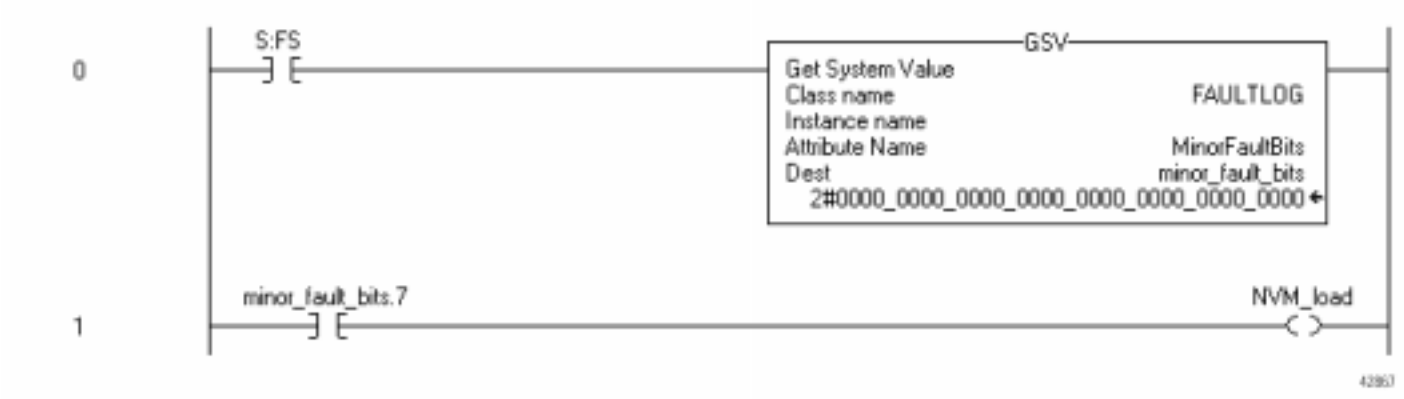
检查加载内容

当控制器从非易失性内存中加载一个工程时，会提供以下信息:

- 记录一个次要故障 (类型 7，代码 49)
- 将 FAULTLOG 对象的 MinorFaultBits 属性中的第 7 位置位。

如果想标记从控制器非易失性内存中加载的工程，使用以下梯形图逻辑:

在工程的第一次扫描时（*S: FS* 闭合），GSV 指令获取 FAULTLOG 对象的 MinorFaultBits 属性，并且将其值保存在 *minor_fault_bits* 中。如果第 7 位为 1，那么控制器从非易失性内存中加载工程。



其中:	解释:
minor_fault_bits	标签用来保存 FAULTLOG 对象的 MinorFaultBits 属性。 数据类型为 DINT (双整型)。
NVM_load	用来显示控制器从非易失性内存中加载工程的标签。

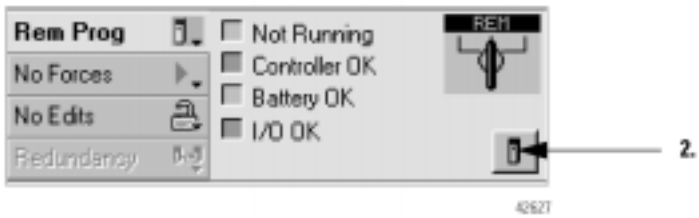
清除非易失性内存

通常，用户没有必要清除控制器的非易失性内存。

- 当用户保存一个工程时，会覆盖非易失性内存的全部内容
- 在更新控制器固件时，非易失性内存的内容会被擦除 (10.x 或更高版本)

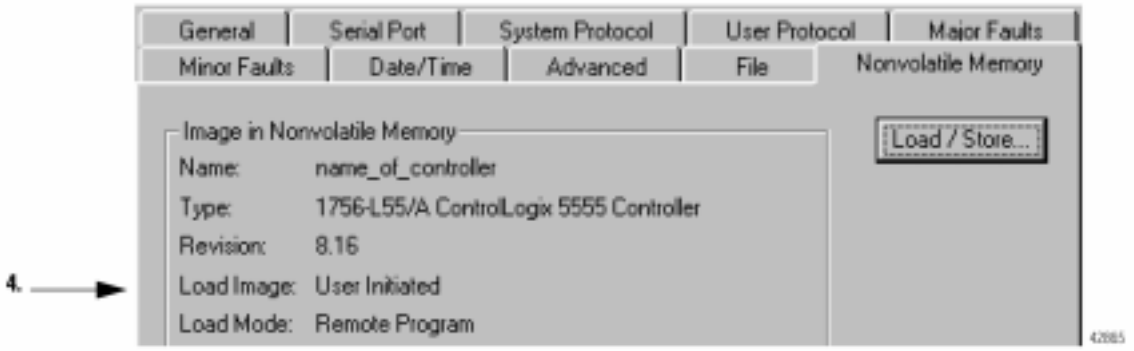
如果用户不想用更新控制器固件的方法清除非易失性内存中所有的内容，完成以下步骤：

1. 让控制器上线



2. 从在线工具栏中，选择控制器属性按钮

3. 点击 *Nonvolatile Memory* 选项。



4. *Load Image* 选项是否为 *User Initiated?*

如果:	那么:
否	跳到步骤 5
是	跳到步骤 11

5. 选择 *Load/Store (加载 / 保存)*

6. 在 *Load Image* 下拉菜单中，选择 *User Initiated*。

7. 选择<-*Store*

对话框将提示用户是否确认保存。

8. 要保存一个工程，选择 *Yes*

对话框将告诉用户工程正在保存过程中。

9. 选择 *OK*。

10. 直到控制器前面的 OK 指示灯一直是绿色，显示保存完毕。

11. 断开控制器电池。

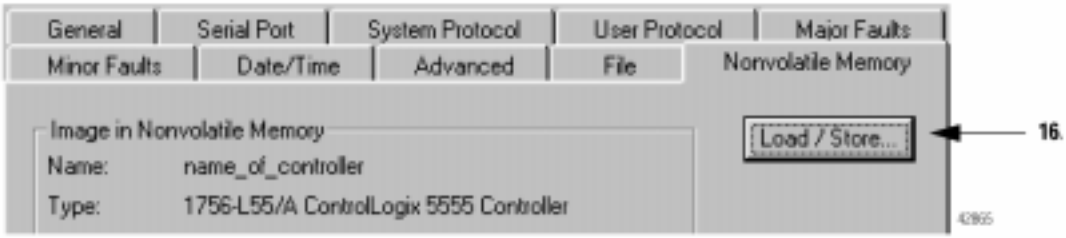
12. 给框架周期性供电。

13. 重新接上控制器电池。

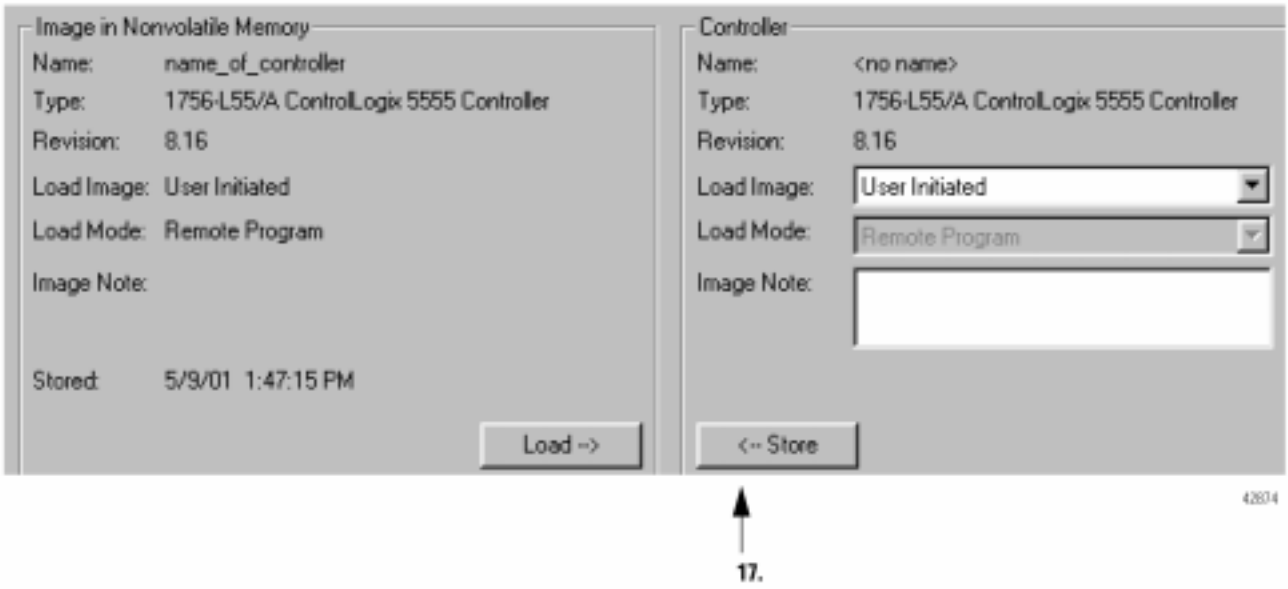
14. 让控制器上线。

上线连接对话框弹出。

15. 点击 *Nonvolatile Memov* 选项。



16. 选择 *Load/Store*。



17. 选择<-*Store*。

对话框将提供用户是否确认保存。

18. 要保存一个工程，选择 *Yes*。

在保存过程中，会出现下列情况:

- 控制器前的 OK 指示灯会以下列顺序:
  闪烁: 绿 ⇒ 红 ⇒ 绿
- RSlogix 5000 软件离线。

19. 选择 *OK*。

在保存完成之后，依然处于离线状态。如果用户想要上线，需要手动上线。

加密一个工程

使用本章

使用本章可控制谁有权限访问用户工程。要加密一个工程，现有以下可用的选项:

如果用户想:	那么:	页码:
防止他人查看工程中一个或多个例程的梯形图逻辑	使用例程源保护	20-1
指定访问工程的多级权限。 例如: • 工程师全部访问 • 维护人员只做有限的修改 • 操作员只能查看梯形图逻辑和数据	使用 RSI 安全服务器建立一个工程	20-9

用户可以同时使用这两个选项。

使用例程源保护

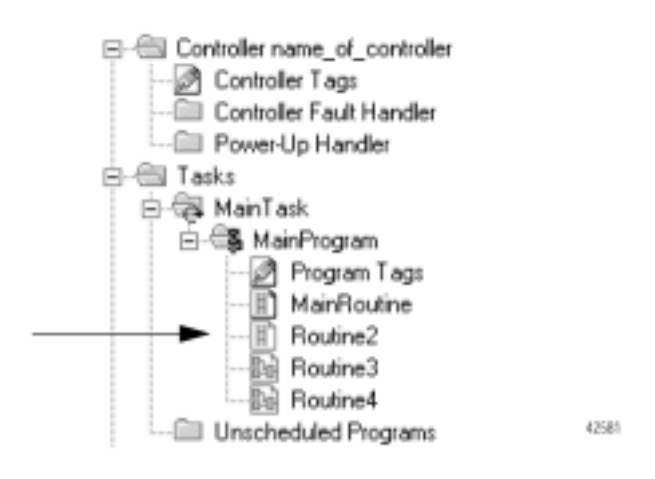
为了限制访问例程的权限，使用RSLogix 5000源保护工具给例程指定一个源密码。一旦指定了源密码，计算机就需要源密码来执行下面的操作:

- 打开 (显示) 例程
- 编辑例程
- 修改例程的属性
- 搜索例程
- 例程内的相互参考
- 打印例程
- 导出例程

不管源密码是否有用，用户都可以下载工程并执行所有的例程。

如果一个例程是灰色的，是指例程受源密码保护。
如果用户双击这个例程，状态行显示 ″源不可用″

要打开此例程，用户计算机需要例程的源密码。



重要提示

如果例程源不可用，不要导出工程。

- 一个导出文件 (.L5K) 只含有那些源代码可用的例程。
- 如果用户导出一个所有例程的源代码都不可用的工程，那么用户将不能重新保存整个工程。

要指定并管理源密码:

- 安装 RSLogix5000 源保护软件
- 为源密码创建一个文件
- 用一个源密码保护例程
- 去掉访问写保护例程
- 获得访问写保护例程

安装 RSLogix5000 源保护软件

1. 获取 RSLogix5000 软件 CD.

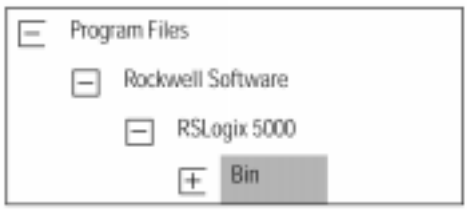
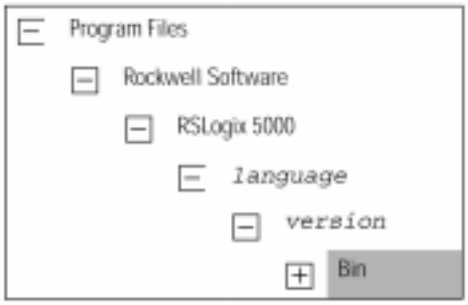
2. 在 CD 上，找到下面的文件:

language \Tools\Source Protection Tool\SP.exe.

其中:

language 是用户软件的语言。例如，要用英文版软件，打开 ENU 文件夹。

3. 用户使用的是 RSLogix5000 软件的哪一种版本?

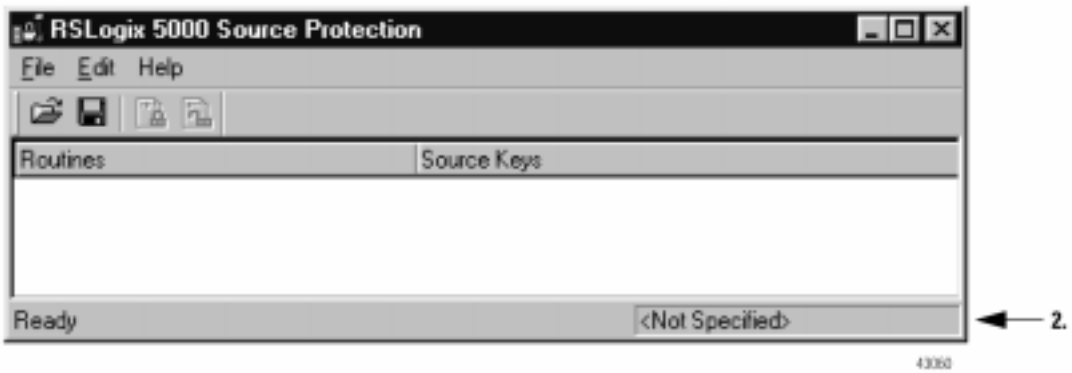
如果	**那么**
9.00 或更早	复制 *SP.exe* 文件到这个文件夹 
10.00 或更新	复制 *SP.exe* 文件到这个文件夹 

其中:	**是**:
languag	用户软件的语言。例如，英文版软件，打开 ENU 文件夹。
version	用户软件的版本，例如 v10

为源密码创建一个文件

用户计算机采用源密码文件格式 (sk.dat) 来保存源密码。
RSLogix 源保护软件允许用户选择保存文件的文件夹。

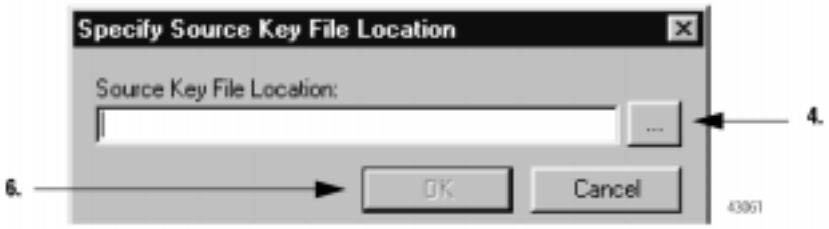
1. 打开 RSLogix5000 源保护软件 (SP.exe).



2. 状态栏是否显示 sk.dat 文件的位置?

如果	那么
是	计算机已经存在源密码文件，参见 20-5 页用一个源密码保护例程。
否	跳到步骤 3。

3. 从 *Edit* (编辑) 菜单中，选择 *Specify Source Key File Location (指定源密码文件位置)*。



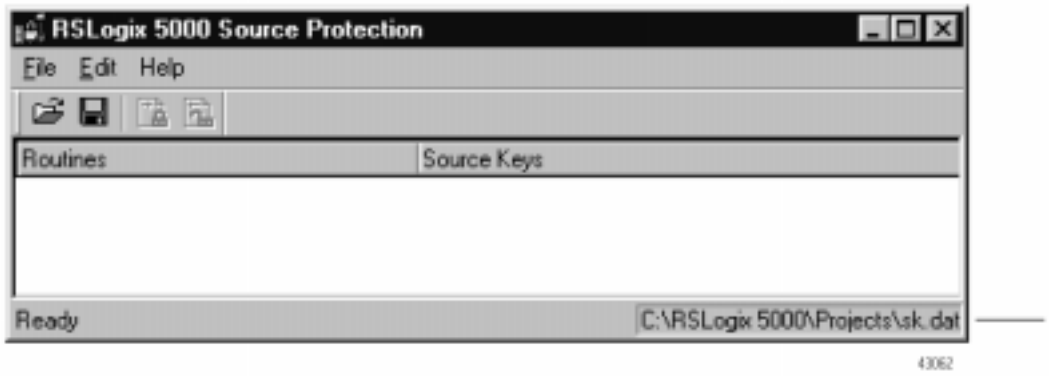
4. 点击 

5. 选择一个文件夹来存储文件并选择 *OK*。

6. 选择 *OK*。

出现一个对话框，问是否想创建一个源密码文件（sk.dat）。

7. 选择 *Yes*。



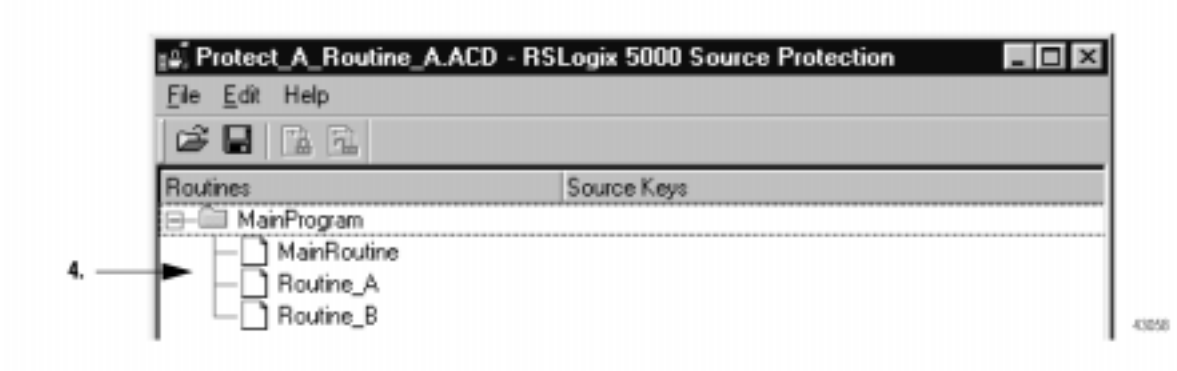
状态栏显示源密码文件 (sk.dat) 的位置。

用一个源密码保护例程

1. 打开 RSLogix5000 源保护软件(SP.exe)。

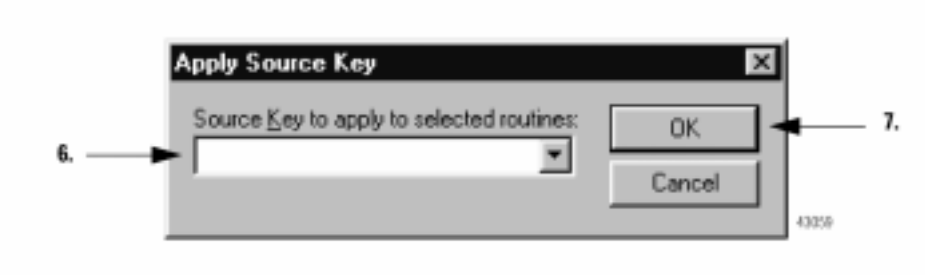
2. 在 *File (文件)* 菜单中，选择 *Open(打开)*。

3. 打开含有用户想要保护例程的工程。



4. 选择用户想要保护的例程。

5. 从 *Edit* (编辑) 菜单中，选择 *Protect Routines With Source Key (用源密码保护例程)*。



6. 填写用户想采用的源密码的名字。源密码命名规则同其它 RSLogix5000 的组成如例程、标签、模块相同。

7. 选择 *OK*。

8. 在 *File(文件)*菜单中，选择 *Save(保存)*。

9. 当用户给所需加密的工程指定完源密码后，关闭 RSLogix5000 源保护软件。

去掉访问写保护例程

只要计算机含有源密码文件，任何人都能使用该计算机去访问被保护的例程。要防止计算机访问被保护的例程，从计算机中去掉源密码文件。

重要提示 在从计算机中去掉源密码文件（sk.dat）之前，记下源密码，或将文件复制一份并保存在安全的地方。

1. 打开 RSLogix5000 源保护软件 (SP.exe)。

2. 如果 RSLogix5000 源保护软件已经打开，那么从工具栏中关闭打开的工程。

3. 在 *Edit (编辑)* 菜单中，选择 *Remove Source Key File Location (删除源密码文件)*。

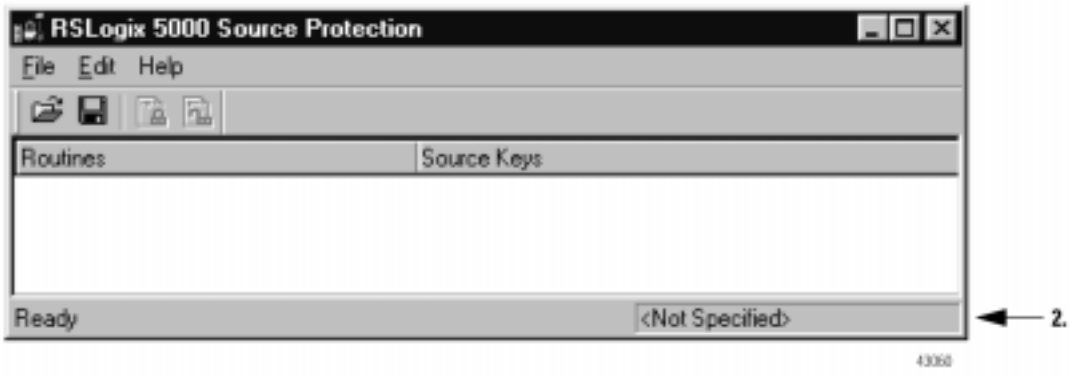
出现一个对话框，问是否删除源密码文件(sk.dat)。

4. 选择 *Yes*。

获得访问写保护例程

为了获得访问被保护的例程，用户计算机需要例程的源密码。源密码放在保护例程的位置。为了把它加到计算机中，用户首先必须从给例程指定源密码的人那里得到例程的源密码名。

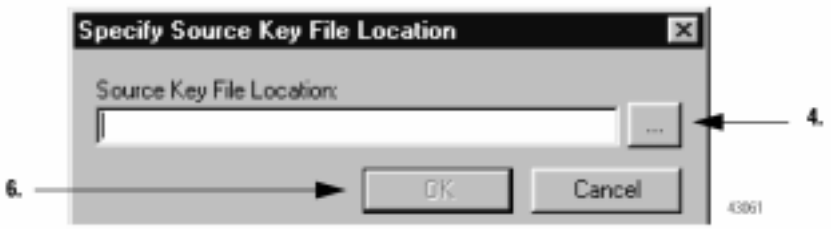
1. 打开 RSLogix5000 源保护软件 (SP.exe).



2. 状态栏是否显示 sk.dat 文件的位置?

如果	那么
是	跳到步骤 8
否	跳到步骤 3

3. 在 *Edit (编辑)* 菜单中，选择 S*pecify Source Key File Location (指定的源密码文件位置)*。



4. 点击 

5. 选择存储文件的文件夹，并选择 *OK* 。

6. 选择 *OK* 。

出现一个对话框，问是否要创建源密码文件(sk.dat)。

7. 选择 *Yes* 。

8. 从 *Edit* (编辑) 菜单中选择 *View Source Key File (查看源密码文件)*。
- 如果用户根据提示选择了一个可以打开文件的程序,选择一个字给打开的例程,例如 Notepad。
- 打开 sk.dat 文件。

9. 输入源密码的名字。输入多个源密码，在每一行单独输入密码的名字。

10. 保存并关闭 ″sk.dat″ 文件。

使用 RSI 加密服务器

RSI 加密服务器软件允许用户限制访问单个 RSLogix 5000 保护例程工程。使用这个软件，用户可以通过以下几个方法来把例程的访问方法客户化:

- 用户目前被记录在工作站里
- 用户正在访问的 RSLogix 5000 工程
- 用户正在访问的 RSLogix 5000 工程所在的工作站

在用户对 RSLogix 5000 工程使用加密服务器之前，安装如下软件:

- 安装 RSI 加密服务器软件
- 建立 DCOM
- 为 RSLogix 5000 软件使能加密服务器
- 导入 RSLogix 5000Security.bak 文件
- 为用户定义全局操作
- 为用户定义工程操作
- 添加用户
- 添加用户组
- 给 RSLogix 5000 软件指定全局访问权
- 为新的 RSLogix 5000 工程指定工程操作

只要给RSLogix 5000安装了加密服务器软件，再完成以下操作就可以保护一个工程:

- 加密一个 RSLogix 5000 工程
- 为 RSLogix 5000 工程指定路径
- 如果需要，更新 RSLogix 5000 软件

安装 RSI 加密服务器软件

重要提示 在用户安装加密服务器软件时,如果发现用户的计算机上已经有了 RSLogix 5000 软件，那么当用户被提示时，给 RSLogix 5000 软件使能加密。

参见 *Getting Results with Rockwell Software's Security Server(Standalone Edition)*, *《罗克韦尔软件加密服务器快速入门》（独立版）*附带在 RSI 加密服务器软件中。

建立 DCOM

参见*Getting Results with Rockwell Software's Security Server (Standalone Edition)*，*《罗克韦尔软件加密服务器快速入门》（独立版）*附带在 RSI 加密服务器软件中。

为 RSLogix 5000 软件启用加密服务器

用户是否在安装 RSLogix 5000 软件*之前*安装加密服务器?

如果:	那么:
是	跳到步骤 1。
否	参见 20-11 页的导入 RSLogix5000Security.bak File

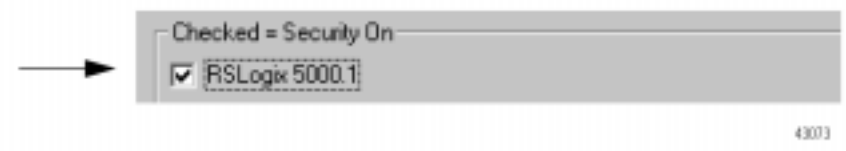
1. 运行这个文件

其中:	解释
language	用户软件的语言。例如，英文版软件，打开 ENU 文件夹。
version	用户软件的版本，例如 v10.

打开本地工程文件对话框。缺省时，Key.ini 文件已经被选中。

2. 选择 *open*

3. 选中 RSLogix5000 选择框并点击 OK。



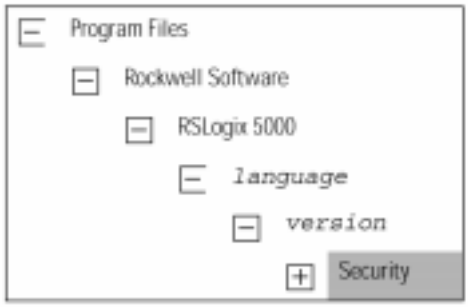
导入 RSLogix5000Security.bak 文件

RSLogix5000Security.bak文件提供的组态信息是加密服务器必须与RSLogix 5000软件同时运行。

1. 启动加密组态信息管理器。

2. 在 *File* (文件) 菜单中，选择 *Import Database* (导入数据库)。

3. 用户使用的加密服务器软件的版本:

如果:	**那么**:
2.00	在此文件夹中查找  **其中**: **是**: *language* 用户软件的语言。例如，英文版软件，打开 ENU 文件夹。 *version* 用户软件的版本，例如 v10
2.01	在此文件夹中查找 

4. 选择 *RSLogix5000Security.bak* 文件，并选择 *Open* (打开)。

为用户定义一个全局操作

全局操作是不依靠特殊工程的任务，例如，创建一个新的工程，或者更新控制器固件。下面的全局操作将应用于 RSLogix5000 软件。

表 20.1 全局操作

要让用户:	那么允许进行以下操作:
加密任意未加密的控制器	加密控制器
创建一个新的 RSLogix5000 工程	新建工程
在新创建的工程中打开 RSLogix 5000 软件中的.L5K 文件	
将 PLC 或 SLC 工程转换成.L5K 文件	
使用 RSLogix5000 软件启动 ControlFLASH 软件，并更新控制器固件	更新固件

使用下面的工作表来记录用户允许每组使用者执行的全局操作。

表 20.2 每组使用者的全局操作

本组使用者	需要访问		
	加密控制器	新建工程	更新固件

为用户定义一个工程操作

工程操作允许用户对一个指定的工程或者工程组执行指定的任务。

- 当用户对一个 RSLogix5000 工程使能加密，或者创建一个带加密运行的新工程，该工程将成为 *New RSLogix 5000 Resources* 组的成员。

  - 使用这组工程工作的人员需要适当的权限。
  - 我们建议用户允许有权创建工程的人能够完全访问。

- 为了使访问工程的权限客户化，将工程从 *New RSLogix 5000 Resources* 组中移出并对其指定特权。

以下操作将应用于已加密的 RSLogix5000 工程或工程组。

表 20.3 工程操作

要让用户:	并且:	并且:	允许以下操作:
• 离线打开一个例程	→	→	浏览工程
• 从工程中复制组件 • 导出工程的标签	上线并监测工程	→	上线
		• 保存工程 • 保存成不同的.ACD 文件 • 打开旧版本的工程 • 压缩工程 • 导出工程 • 下载或上载工程 • 改变控制器的模式 • 改变控制器的路径 • 打印报告 • 清除故障 • 改变时钟 • 创建，删除，编辑并且运行趋势图 • 改变 I/O 模块的组态 • 改变 MSG 指令的组态 • 输入，启用，禁止和去掉强制值 • 改变标签值 • 更新固件	维护工程
执行 RSLogix5000 软件中所有有效操作，除了解除一个控制器密码。	→	→	完全访问
解除控制器密码	→	→	完全访问和解除控制器密码
更新控制器固件	→	→	更新固件

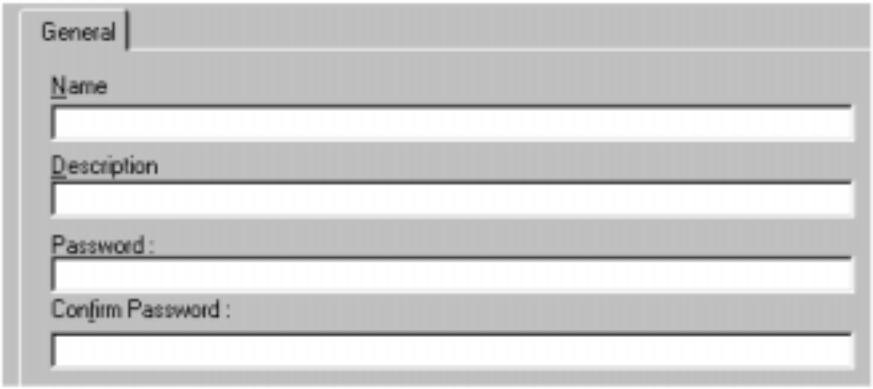
使用 20-15 页的工作表来记录允许用户或用户组执行的工程操作。

表 20.4 对新的 RSLogix5000 源组中的工程和单独工程的工程操作

对工程或 工程组:	用户或用户组: 浏览 工程	需要访问: 上线 工程	维护 访问	完全 密码	解除控制器	更新固件
新的 RSLogix5000 源						
新的 RSLogix5000 源						
新的 RSLogix5000 源						
新的 RSLogix5000 源						

添加用户

1. 右键点击并选择 *New (新建)*。

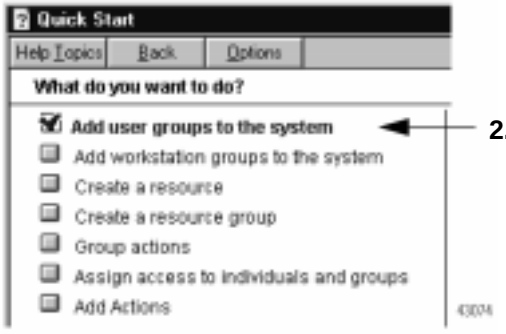


2. 输入用户信息，然后选择 *OK (完成)* 。

添加用户组

用户组允许用户管理具有类似权限的多个用户。

1. 在 *Help (帮助)* 栏中，选择 *Quick Start (快速启动)*。



2. 按此任务的步骤操作。

指定全局访问 RSLogix5000 软件

要允许用户执行全局操作:

1. 在组态信息管理器中，选择 RSLOGIX5000 组。

2. 在 *Help (帮助)* 栏中，选择 *Quick Start(快速启动)*。

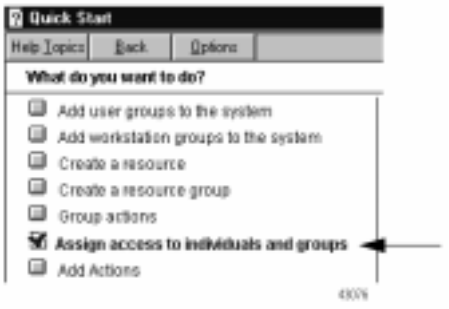
3. 按此任务的步骤操作。指定 20-12 页的表 20.2 记录的操作。

为新的 RSLogix5000 工程指定工程操作

要允许用户对 New RSLogix 5000 Resources 组中的工程执行操作:

1. 在组态信息管理器中，选择 *New RSLogix 5000 Resources* 组。

2. 在 Help (帮助) 栏中，选择 *Quick Start (快速启动)*。

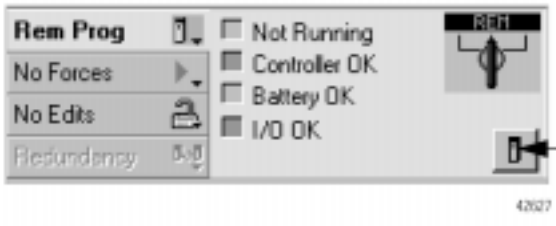


3. 按此任务的步骤操作，指定 20-15 页的表 20.4 记录的操作。

加密 RSLogix5000 工程

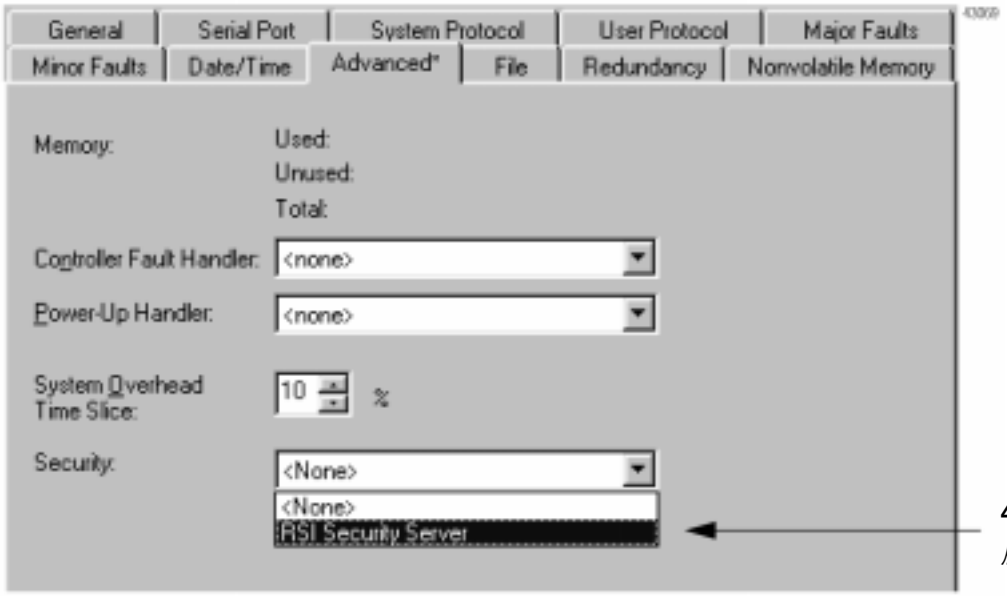
当用户创建一个新工程时，加密选项会有效的。要使用加密服务器软件保护现有的工程，对工程使能加密。

1. 打开 RSLogix5000 工程。



2. 点击控制器属性按钮。

3. 点击 *Advance (高级)* 选项。



4. 选择 *RSI Security Server (加密服务器)*。

5. 选择 *OK*，然后点击 *Yes*。

在加密服务器软件中，工程将作为 *New RSLogix 5000 Resources* 组的成员出现。如果加密服务器软件已经打开，那么在 *View (视图)* 菜单中，选择 *Refresh (更新)*。

为 RSLogix 5000 工程指定权限

对于 New RSLogix 5000 Resources 组中的工程，由权限控制列表来决定对其执行的操作。为了使访问工程的权限客户化，将工程从组中移出并对其指定特权。

1. 在组态信息管理器中选择 *New RSLogix 5000 Resources* 组。

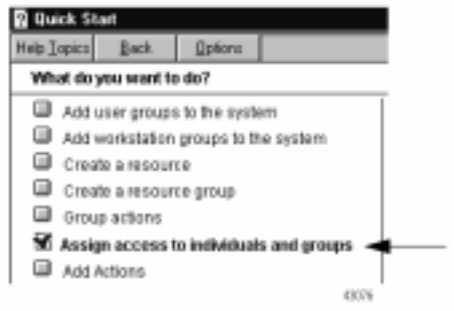
2. 点击 *Group members* 选项。

3. 在 *Member items* 列表中选择工程并点击<<按钮。

4. 选择 *apply (应用)*。

5. 在组态信息管理器中选择工程。

6. 在 *Help (帮助)* 栏中，选择 *Quick Start (快速启动)*。



7. 按照此任务中的步骤，指定 20-15 页的表 20.4 记录的操作。

必要时更新 RSLogix 5000 软件

如果RSLogix 5000工程的打开与在RSI 加密服务器软件中进行的修改，会影响该工程，那么更新 RSLogix 5000 软件。

在 *Tool(工具)*菜单中选择 *Security⇒Refresh Privileges*。

Note:

故障代码

何时使用本附录

使用此附录来解释:

- 主要故障代码
- 次要故障代码

主要故障代码

通过下表可以确定一个主要故障的原因及校正办法。故障类型和代码如下:

- 控制器属性对话框，主要故障选项
- PROGRAM 对象，MAJORFAULTRECORD 属性

表 A.1 主要故障类型和代码

类型:	代码:	原因:	校正方法:
1	1	控制器在 Run 模式下上电	执行掉电处理程序
3	16	必要 I/O 模块连接失败	检查框架中的 I/O 模块。检查电子锁的要求规格。 查看控制器属性中主要故障选项和模块属性中连接器选项来获取更多的故障信息。
3	20	控制总线背板可能存在问题	不可修复，更换背板
3	23	在运行之前未建立任何连接	等控制器 I/O 指示灯变绿后再将控制器打到运行状态。
4	16	未知的指令冲突	删除未知指令。程序转换处理引起。
4	20	数组下标过大，控制结构体.POS 或.LEN 位无效	将值调整到有效范围内。不要超过数组或维数范围。
4	21	控制结构体.POS 或.LEN 位<0	将其值调整到>0
4	31	JSR 指令参数与其相关的 SUB 数太多，超出的部分视为无错误被忽略。	传递适当数量的参数。如果传递的参很或 RET 指令参数不匹配
4	34	计时器指令中有一个负的预置值或累加值	确认程序中没有给计时器装入负的预置值或累加值
4	42	JMP 指令跳转的标记不存在或被删除	纠正 JMP 的跳转标记或添加丢失的标记
4	83	数据校验超出其规定的范围	修改数值使其在规定的范围内
4	84	堆栈溢出	减少子程序的嵌套或传递参数的数量

表 **A.1** 主要故障类型和代码 (续)

类型:	**代码**:	**原因**:	**校正方法**:
6	1	任务看门狗过期 在规定的时间内，用户的任务 没有执行完。程序由于故障 导致无限循环，或程序太复 杂而不能按指定的速度执行， 或为了保证优先级高的任务完成 而禁止此任务执行。	增加任务看门狗，缩短执行时间，提高 任务优先级使其高于原高级别任务，将 一些代码移到其它控制器。
7	40	保存到非易失性内存失败	1.试着将工程再保存一遍。 2.如果工程保存失败，更换内存板
7	42	从非易失性内存中读取数据失败， 因为非遗失性内存中工程的固件 版本与控制器的固件版本不匹配。	更新控制器的固件版本使其 非易失性内存固件版本相同。
8	1	在程序下载过程中用钥匙开关将 控制器打到运行模式	等下载完之后将故障清除
11	1	实际位置超过正的超程限制	将轴往反方向转动直到其位置回到超程限 制之内，并清除运动轴故障
11	2	实际位置超过负的超程限制	将轴往正方向转动直到其位置回到超程限 制之内，并执行运动轴故障复位
11	3	实际位置超出故障偏差	将其位置转到偏差范围内并执行运动轴故障复位
11	4	编码器通道 A、B 或 Z 连接被损坏	重新连接编码器并执行运动轴故障复位
11 位	5	编码器出现噪声或信号没有正交	确定编码器连接电缆的接法并执行运动轴故障复
11	6	驱动故障输出被激活	清除驱动故障然后执行运动轴故障复位
11	7	同步连接失败	首先执行清除运动轴故障。如果还不工作将 伺服模块拔出再插进去。如果还是失败责更 换伺服模块。
11	8	发现伺服模块出现严重的硬件故障	更换模块
11	9	异步连接失败	首先执行运动轴故障复位。如果还不工作将 伺服模块拔出再插进去。如果还是失败则更 换伺服模块。
11	32	运动控制任务经常重叠	数组的刷新频率太高而不能正确的操作。清除 数组的故障标签，再增加数组的刷新频率并清 除主要故障。

次要故障代码

通过下表可以确定一个次要故障的原因及校正办法。故障类型和代码如下所示:

- 控制器属性对话框，主要故障选项
- PROGRAM 对象，MINORFAULTRECORD 属性

表 **A.2** 次要故障类型和代码

类型:	**代码**:	**原因**:	**校正方法**:
4	4	某条指令出现运算溢出	通过检查算术运算操作（顺序）或调节数值来调整程序
4	7	由于 GSV/SSV 指令的目的标签太小而不能保存数据	调整目的标签使其有足够大的空间。
4	35	PID 增益时间<0	调整 PID 增益时间使其 >0
4	36	PID 设定点超出范围	调整设定点使其在设置范围内
4	51	字符串标签的 LEN 值大于其 DATA 的大小	1. 确认没有指令向字符串 LEN 成员写入数据 2. 在 LEN 值中，写入字符串所包含的字符的个数
4	52	输出字符串大于目的标签	给输出字符串创建一个新的足够大的字符串数据类型。用新的字符串类型作为目的标签的数据类型
4	53	输出数量超出目的标签数据类型的范围	任选其一: ï 减小 ASCII 码值的大小 ï 给目的标签使用一个大的数据类型
4	56	起始值或数量值无效	1.确认起始值在 1 和源 DATA 大小之间 2.确认起始值加数量值小于或等于源的 DATA 大小
4	57	AHL 指令执行失败，因为串行端口设置成无握手	任选其一: ï 修改串行端口控制线的设置 ï 删除 AHL 指令
6	2	周期性任务重叠。 在周期性任务没有执行完之前又开始执行此任务	简化程序，或延长周期，或提高其相对优先级，等等。
7	49	从非易失性内存中上载一个工程	
9	0	在维修串口时出现一个未知故障	联系 GTS 部门人员
9	1	当前组态中 CTS 连线故障	断开控制器的串口电缆再连接上。 确认电缆连接正确无误。

表 A.2 次要故障类型和代码 (续)

类型:	代码:	原因:	校正方法:
9	2	轮询列表故障。 在 DF1 主控制器轮询列表中发现一个问题，例如指定站点数多于文件规定的大小，超过 255 个站点，试图指向列表或轮询广播地址（STN#255）的末端。	在轮询列表中查询下列故障: • 站点总数超出轮询列表标签的空间 • 站点总数大于 255 • 当前站点指针大于轮询列表末端值 • 站点号大于 254 冲突
9	5	DF1 从控制器轮询超时 主控制器没有在规定的时间内轮询到该控制器。	计算出轮询延迟时间并改正
9	9	调制解调器的连接失败。 DCD 和 / 或 DSR 控制行没有按正常的顺序和 / 或状态被程序接收。	校正调制解调器与控制器的连接
10	10	没安装电池或电池需要更换。	安装新的电池

IEC61131-3 规范

使用本附录

介绍

国际电工技术委员会 (IEC) 已经制定了一系列有关可编程序控制器的规范。制定这些规范的目的就是为了促进控制工业中所使用设备和编程语言的国际统一化。这些标准也为 Logix5550 控制器和 RSLogix5000 编程语言提供了基础。

IEC 可编程序控制器规范可以分解为五个单独的部分，每个部分又集中在控制系统不同的方面:

- 第 1 部分：一般信息
- 第 2 部分：设备及要求的测试
- 第 3 部分：编程语言
- 第 4 部分：用户指南
- 第 5 部分：信息服务规范

总体上控制工业集中在第 3 部分(IEC 1131-3)，即编程语言，因为它提供了执行其它标准的基础，并且通过减少培训费用它还提供了最重要的最终用户利益。正因为如此，这里我们只介绍 IEC 1131-3。

IEC 1131-3 编程语言规范设计到可编程序控制器的许多方面，包括：操作系统执行，数据定义，编程语言及指令集。IEC 1131-3 规范的组成部分根据需要分类如下：规定，任选和扩展。通过这种方法，IEC 1131-3 规范提供了一种经过扩展即可满足最终用户应用需要的最小功能性组合。这种方法的最后阶段既是每个可编程控制系统厂家都可以执行规范中的不同部分，或提供不同的扩展。

操作系统

Logix5000 的抢先多任务操作系统(OS)是符合 IEC 1131-3 定义的。在 IEC 1131-3 中，可编程序控制器操作系统(OS)可以包含零个或多个任务，每个任务又可执行一个或多个程序，在每个程序中可以包含一个或多个函数或例程。根据 IEC 1131-3，这些组成部分中每一类的数量多时由设备决定的（implementation dependent）。Logix5550 提供了 32 个任务，每个任务包含 32 个程序和任意数量的函数和例程。

IEC 1131-3 提供了一种选项，可用于创建不同的任务执行类别。任务可以被组态成连续的、周期性的和基于事件的三种类型。连续任务不需要安排执行时间，每当其它任务暂停时连续任务就将使用全部剩下的时间。周期性任务则根据重新生产的时间周期定期运行。IEC 1131-3 规范没有为周期性任务组态指定时间片。而基于事件的任务则是根据检测到的组态输入上升沿来进行触发。Logix5000 支持连续任务和周期相任务。此外，对于周期性任务，可组态的启动最低可达 1 毫秒。

数据指南

IEC 1131-3 规范提供了通过创建命名变量来实现对内存的访问。IEC 1131-2 规定的变量名称最少需由 6 个字符组成(RSLogix5000 编程软件支持最少由一个字符组成的变量)，并且是以下划线〝_〞和字母字符(A-Z)开头，其后可以跟一个和多个由下划线〝_〞，字母字符(A-Z)和数字(0-9)组成的字符。此外，只要它们不区分大小写(A = a，B = b，C = c…)，也一样支持小写的字母(a-z)。Logix5550 完全遵循上述定义，支持小写选项和扩展名称，最多达 40 个字符的名称。

IEC 1131-3中的数据变量定义为: 所有资源内和控制器内的程序多可以访问这些数据变量，或者提供对单个程序中函数或例程的有限制的访问。为了在多个任务或控制器之间传递数据，可以组态访问路径，它定义了系统中数据的位置。Logix5550遵循上述规定，提供了程序作用域和控制器作用域的数据，并且允许利用生产型/消费型数据来进行访问路径组态。

在IEC 1131-3中变量的内存说明是通过基本数据类型和可选派生数据类型来定义的，派生数据类型可以从一个多数据类型组来创建。Logix5550支持下列数据类型的使用: BOOL(1位)，SINT(8位整数)，INT(16位整数)，DINT(32位整数)和REAL(IEEE浮点数)基本数据类型。此外，控制器通过创建用户定义结果提及数组即可支持可选派生数据类型。

编程语言

IEC 1131-3规范定义了五种不同的编程语言和一组公用元素。所有语言均定义为可选的，但是若想符合规范要求则必须至少支持其中的一种语言。IEC 1131-3编程语言的组成元素定义如下:

- 公用语言元素
- 公用图形元素
- 指令表(IL)语言元素
- 结构化文本(ST)语言元素
- 梯形图(LD)语言元素
- 顺序功能图(SFC)语言元素
- 功能块图(FBD)语言元素

Logix5550和RSLogix5000支持公用语言元素和梯形图语言。此外，工作平台采用了基于结构化文本语言的ASC II输入/输出格式。有关指令集和程序文件呼唤特性的详细内容将在下面的章节讨论。

指令集

IEC 1131-3 指定的指令集在整体上都是可选的。如果执行的指令必须符合规定的执行标准及直观显示，则规范还列出了一些有限制的指令。但不管怎样，对于那些列于规范中的指令集，IEC 1131-3并未对其进行限制。除了规范中列出的以外，每一家PLC厂商都可以自由提供指令表格中的附加功能性。类似的扩展指令不如那些需要执行诊断，PID 闭环控制，运动控制及数据文件处理的指令。由于 IEC 1131-3规范中没有定义扩展指令，因此不能保证不同的PLC厂家在执行扩展指令时能够彼此兼容。这次阿指令的使用降低了不同厂家之间逻辑程序的可移植性。

Logix5550 和 RSLogix5000 提供了一组可按照 IEC 1131-3 规范定义执行的指令。这些指令的实际形式与现有的系统指令有许多相似之处，这样就可以减少与工作平台运行有关的培训费用。除了符合 IEC 1131-3 的指令要求外，现有产品种的全部指令都已被抑制到了工作平台中，因而保留了原有的功能性。

IEC 1131-3编程可移植性

在一个符合 IEC 1131-3 标准的工作平台上，终端用户创建程序的目标程序移植之一是在不同厂家开发的控制器之间实现程序的移动和移植。而这也正是IEC 1131-3的一个缺点，因为该规范没有定义文件的互换格式。这就意味着如果在某一厂家的工作平台中创建的任意一个程序要求处理，需将它移植到另外一个厂家的系统中。

为了减少在进行跨厂家程序移植时的工作量。Logix5550控制器的RELogix5000编程软件包含了一个完整的 ASC Ⅱ输出和输入工具。此外，该工具使用的文件格式是基于 IEC 1131-3 结构化文本语言定义的混合基础上。控制器操作系统及数据定义遵循适当的IEC 1131-3格式。为了将梯形图逻辑程序转换成ASC Ⅱ文本需进行扩展，因为 IEC 1131-3 没有对此进行定义。

欲了解 RSLogix5000 编程软件的有关 ASC Ⅱ输入和输出的更多信息，请参看 Logix5000 控制器导入 / 导出参考手册，1756-RM084 出版。

IEC 规范表

Logix5550 和 RSLogix5000 遵守 IEC 1131-3 对下列语言特征的要求:

表格编号(1):	特征编号:	特征说明:	延伸和执行注释:
1	2	小写字母	无
1	3a	数字符（#）	用于立即数值数据类型标志
1	4a	美元符（$）	用于说明及字符串控制字符
1	6a	下标分割符（[ ]）	数组下标
2	1	使用大写字母和数字的标识符	任务，程序，例程，结构体及标签名称
2	2	使用大写字母，数字和内嵌下	任务，程序，例程，结构体及标签名称
		划线的标识符	
2	3	使用大写和小写字母，数字和内	任务，程序，例程，结构体及标签名称
		嵌下划线的标识符	
4	1	整数直接量	12，0，-12
4	2	实直接量	12.5，-12.5
4	3	带指数的实直接量	-1.34E-12，1.234E6
4	4	以 2 作基数的直接量	2#0101_0101
4	5	以 8 作基数的直接量	8#377
4	6	以 16 作基数的直接量	16#FFE0
4	7	Boolean 型的 0 和 1	0，1
5	1	空字符串	说明
5	2	一个包含字符 ´A´ 的长度字符串	说明
5	3	一个包含空格 ´ ´ 的长度字符串	说明
5	4	一个包含单引号字符 ´$´ 的长度字符串	说明
6	2	字符串美元符 ´$$´	说明
6	3	字符串单引号 ´$´	说明
6	4	字符串换行 ´$L´ 和 ´$l´	说明
6	5	字符串换行 ´$N´ 和 ´$n´	说明
6	6	来自 Feed（页）的字符串 ´$P´ 和 ´$p´	说明
6	7	字符串回车 ´$R´ 和 ´$r´	说明
6	8	字符串标签 ´$T´ 和 ´$t´	说明
10	1	BOOL 数据类型	标签变量定义
10	2	SINT 数据类型	标签变量定义
10	3	INT 数据类型	标签变量定义
10	4	DINT 数据类型	标签变量定义
10	10	REAL 数据类型	标签变量定义
10	12	Time（时间）	标签变量定义，TIMER 结构体

表格编号(1):	特征编号:	特征说明:	延伸和执行注释:
10	16	STRING 数据类型	无
11	1	数据类型体系	无
12	1	从基本类型直接派生	用户定义数据类型结构体
12	4	数组定义结构类型	标签变量定义
12	5	结构数据类型	用户定义数据类型结构体
13	1	BOOL,SINT,INT,DINT 初始值为 0	标签变量定义
13	4	REAL,LREAL 的初始值为 0.0	标签变量定义
13	5	时间初始值为 T # s	标签变量定义，复位指令（RES）
13	9	空字符串 ´´	说明
14	1	直接派生类型初始化	输入 / 输出
14	4	数组数据类型初始化	输入 / 输出
14	5	结构类型元素初始化	输入 / 输出
14	6	派生结构数据类型初始化	输入 / 输出
20	1	EN 和 ENO 的用途	在梯形图中以函数呈现，但未标记 在 FBD 中可用
20	2	无 EN 和 ENO 时的用法	在 FBD 中可用
20	3	有 EN 无 ENO 时的用法	在 FBD 中可用
20	4	无 EN 有 ENO 时的用法	在 FBD 中可用
21	1	过载函数 ADD（INT,DINT） 或 ADD（DINT,REAL）	所有支持的指令过载类型均可用一 条指令来证明
22	1	_To_convertion 函数	RAD,DEG 指令弧度转换为十进制, 或世纪之转换为弧度
22	2	截取转换功能	TRN 指令
22	3	BCD 到 INT 的转换	FRD 指令
22	4	INT 到 BCD 的转换	TOD 指令
23	1	绝对值	ABS 指令
23	2	平方根	SQR 指令
23	3	自然对数	LN 指令
23	4	以 10 为底的对数	LOG 指令
23	6	正弦（采用弧度制）	SIN 指令
23	7	余弦（采用弧度制）	COS 指令
23	8	正切（采用弧度制）	TAN 指令
23	9	反正弦	ASN 指令
23	10	反余弦	ACS 指令
23	11	反正切	ATN 指令
24	12	算术加法	ADD 指令

表格编号(1):	特征编号:	特征说明:	延伸和执行注释:
24	13	算术乘法	MUL 指令
24	14	算术减法	SUB 指令
24	15	算术除法	DIV 指令
24	16	求模	MOD 指令
24	17	求幂	XPY 指令
24	18	数值移动	MOV 指令
25	1	向左移位	功能包含在移位 1 梯级的 BSL 指令中
25	2	向右移位	功能包含在移位 1 梯级的 BSR 指令中
25	3	向左旋转	功能包含在移位 1 梯级的 BSL 指令中
25	4	向右旋转	功能包含在移位 1 梯级的 BSR 指令中
26	5	按位与	AND 指令
26	6	按位或	OR 指令
26	7	按位异或	XOR 指令
26	8	按位取反	NOT 指令
27	1	选择	FBD 中的 SEL 指令
27	2a	最大量选择	功能包含在 FBD 中的 ESEL 指令中
27	2b	最小量选择	功能包含在 FBD 中的 ESEL 指令中
27	3	高 / 低界限	FBD 中的 HLL 指令
27	4	多路器	FBD 中的 MUX 指令
28	5	大于比较	GRT 指令
28	6	大于或等于比较	GRE 指令
28	7	等于比较	EQU 指令
28	8	小于比较	LES 指令
28	9	小于或等于比较	LEQ 指令
28	10	不相等比较	NEQ 指令
29	1	字符串长度	包含 STRING 数据类型的参量
29	4	中间字符串	梯级中的 MID 指令
29	5	字符串串连	梯级中的 CONCAT 指令
29	6	字符串插入	梯级中的 INSERT 指令
29	7	字符串删除	梯级中的 DELETE 指令
29	9	发现字符串	梯级中的 FIND 指令
32	1	输入读取	FBD 编辑器

表格编号(1):	特征编号:	特征说明:	延伸和执行注释:
32	2	输入写入	FBD 编辑器
32	3	输出读取	FBD 编辑器
32	4	输出写入	FBD 编辑器
34	1	双稳态设定权	FBD 中的 SETD 指令
34	2	双稳态复位权	FBD 中的 RESD 指令
35	1	上升型边缘探测器	梯级中的 OSR 指令和 FBD 中的 OSRI 指令
35	2	下降型边缘探测器	梯级中的 OSR 指令和 FBD 中的 OSRI 指令
36	1b	上升型计数器	功能包含在 CTU,梯级中的 RES 指令和 FBD 的 CTUD 指令中。
37	2a	开启的延时计数器	功能包含在梯级中的 TON 指令和 FBD 中的 TONR 指令中。
37	3a	关断的延时计数器	功能包含在梯级中的 TOF 指令和 FBD 中的 TOFR 指令中。
38	2	开启的延时定时	功能包含在梯级中的 TON 指令和 FBD 中的 TONR 指令中。
38	3	关断的延时定时	功能包含在梯级中的 TOF 指令和 FBD 中的 TOFR 指令中。
57	1,2	梯级水平线	梯形图编辑器，流程图编辑器
57	3,4	垂直线	梯形图编辑器，流程图编辑器
57	5,6	水平 / 垂直连接	梯形图编辑器，流程图编辑器
57	9,10	连接和非连接转角	梯形图编辑器，流程图编辑器
57	11,12	带连接的程序块	梯形图编辑器，流程图编辑器
57	7,8	无连接的线路交叉	流程图编辑器
57	13,14	连接器	流程图编辑器
58	2	无条件跳转	梯级中的 JMP 指令
58	3	跳转目标	梯级中的 LBL 指令
58	4	条件跳转	梯级中的 JMP 指令
58	5	条件返回	梯级中的 RET 指令
58	8	无条件返回	梯级中的 RET 指令
59	1	左侧电源线	梯形图编辑器
59	2	右侧电源线	梯形图编辑器
60	1	水平链	梯形图编辑器
60	2	垂直链	梯形图编辑器
61	1,2	常开触点 --| |--	梯级中的 XIC 指令
61	3,4	常闭触点 --| / |--	梯级中的 XIO 指令
61	5,6	正跳变检测触点 -| P |-	梯级中的 ONS 指令

表格编号(1):	特征编号:	特征说明:	延伸和执行注释:
62	1	线圈 --( )--	梯级中的 OTE 指令
62	3	设置（锁存器）线圈	功能包含在梯级中的 OTL 指令中
62	4	复位（开锁）线圈	功能包含在梯级中的 OTU 指令中
62	8	正跳变检测线圈	梯级中的 OSR 指令
62	9	负跳变检测线圈	梯级中的 OSF 指令

(1)除了梯形图和功能块外，与语言有关的表格均被跳过。

Note:

A

别名标签 (alias tag)

该标签引用另一个标签。一个别名标签可以引用另一个别名标签或基本标签。别名标签还可以通过引用结构体成员、数组元素、标签中的一位或成员中的一位来引用另一个标签的组件。参见基本标签。

ASCII

一个7位的编码(带有一个可选的奇偶位)，通常用来表示字母数字符、标点符号和控制代码的字符。参看手册的封底的 ASCII 代码列表。

异步 (asynchronous)

彼此动作是独立的，而且没有规定的模式。在Logix5000控制器中，I/O数值的更新与梯形图逻辑的执行是异步的:

- 一个任务中的程序直接从控制器作用域内存读取输入和输出数据。
- 任何任务中的逻辑都能修改控制器作用域内的数据。
- 数据和 I/O 值是异步的，并且在任务执行过程中都可以改变。
- 在任务开始执行时引用一个输入值和在任务执行后再引用一个输入值,结果是不相同的。

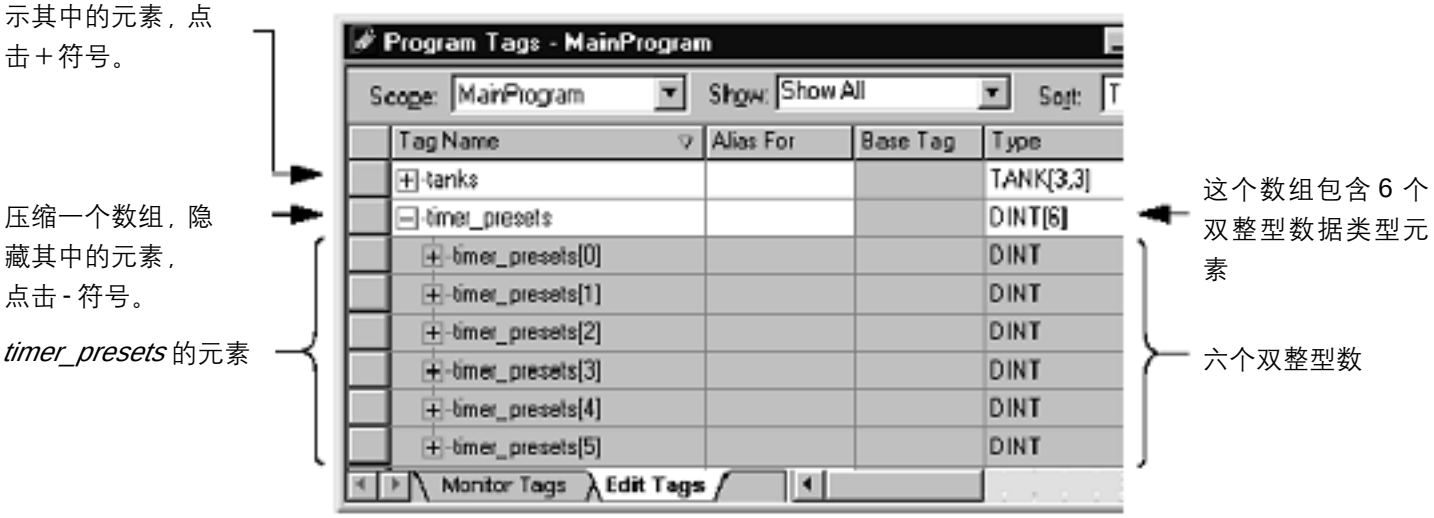
注 意

确保在任务执行过程中数据存储器包含适当的值。
在程序开始扫描时用户可以为其梯形图逻辑引用的值复制或缓存数据。

数组(array)

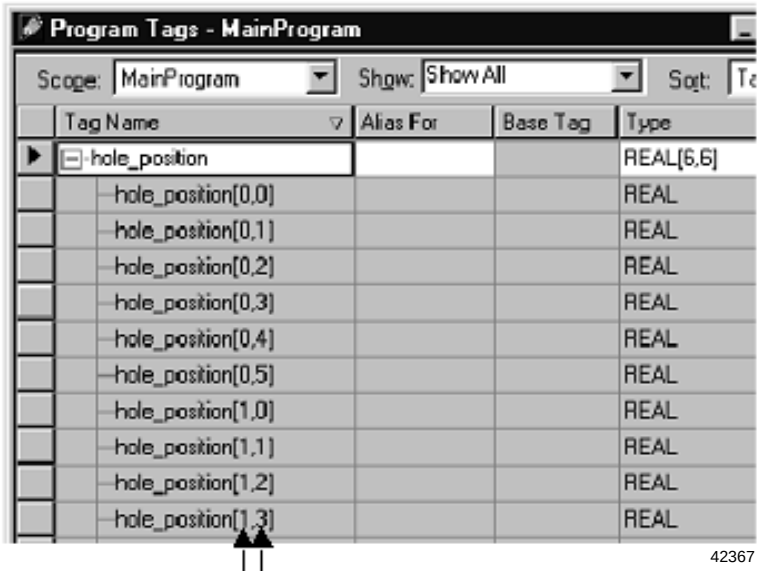
数组允许用户用共同的名称保存数据（具有相同类型）。

- 数组与文件类似。
- 用数组的下标来识别数组中的每个元素。
- 数组的下标从 0 开始直到元素总数减 1（下标以零为基准）。



- 一个数组标签占据了控制器中一个连续的内存区,其中的每个元素都是按顺序排列的。
- 用户可以使用数组和序列发生器指令来操作或引用数组中的元素。
- 一个数组最多可以为三维。这样可以使用户灵活地用一个，两个或三个下标(坐标)来定义一个元素。

• 数组中元素的总数为各维数的乘积，如下面例子中描述:

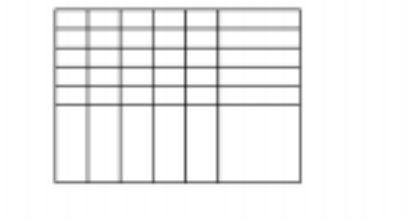
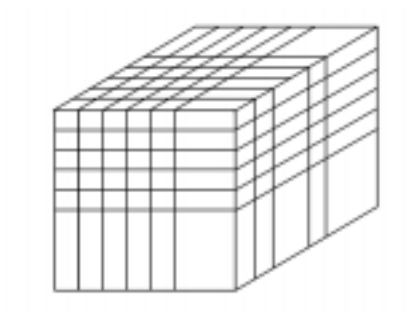


这个数组包括一个二维的栅格，6 个元素乘 6 个元素。

最右边的一维增大到最大值，增加完毕。

当最右边的一维增加完毕，这一维左边的那一维开始逐个的增加。

• 在一个二维或三维数组中，内存中最右边的一维首先增加。

数组:	想这样保存数据:	举例:				
一维		标签名:	类型	0 维	1 维	2 维
		one_d_array	DINT[7]	7		
		元素总数 = 7				
		有效下标范围 DINT[x] ,这里 x=0-6				
二维		标签名	类型	0 维	1 维	2 维
		two_d_array	DINT[4,5]	4	5	
		元素总数 = 4*5 =20				
		有效下标范围 DINT[x,y],这里 x=0-3;y=0-4				
三维		标签名	类型	0 维	1 维	2 维
		three_d_array	DINT[2,3,4]	2	3	4
		元素总数 = 2*3*4 = 24				
		有效下标范围 DINT[x,y,z],这里 x=0-1;y=0-2;z=0-3				

• 用户可以在程序离线状态下在修改数组的维数，并不丢失标签数据。但在程序上线状态下不能修改。

应用程序 (application)

是例程，程序，任务及用于定义单个控制器操作的 I/O 组态的组合。参见工程 。

B

基本标签 (base tag)

该标签实际定义了存储数据元素的内存区域。参见*别名标签*。

双向连接 (bidirectional connection)

这种连接允许数据双向流动: 从发送方到接收方到或者从接收方到发送方。参见*连接，单*向连接。

二进制 (binary)

以基数 2 显示和输入的整数值（每个数字表示一个单独的位）。前缀为 2 #。将填充至布尔型或整数的字长（1，2，16 或 32 位）。当显示时，每 4 个数字为一组，并且为了容易辨认在组与组之间用下划线隔开。参见*十进制，十六进制，八进制*。

位 (bit)

二进制位。最小的存储单元。采用数字 0（清除）和 1（置位）来表示。

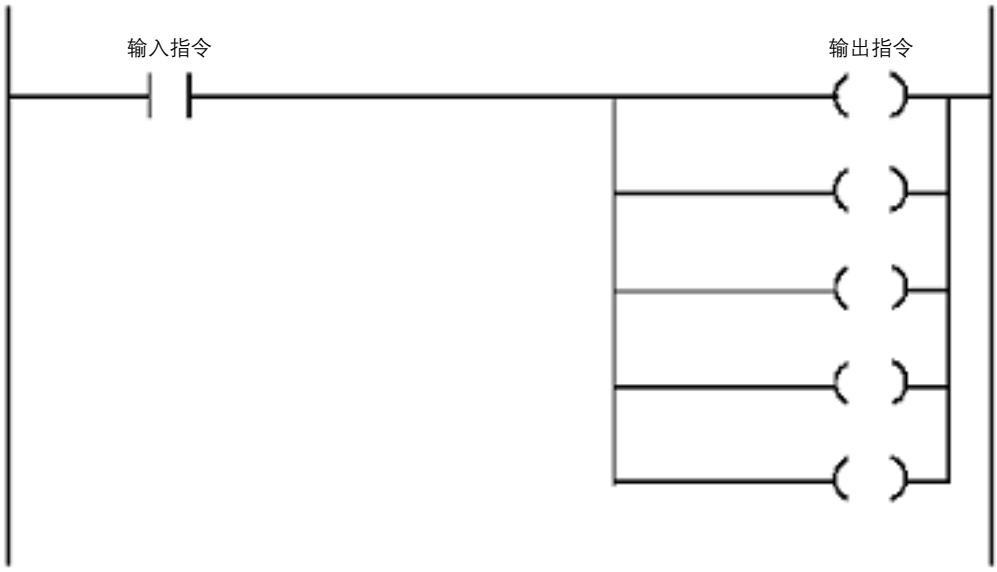
布尔变量 (BOOL)

一种用于保存单独一位状态的基本数据类型，这里:

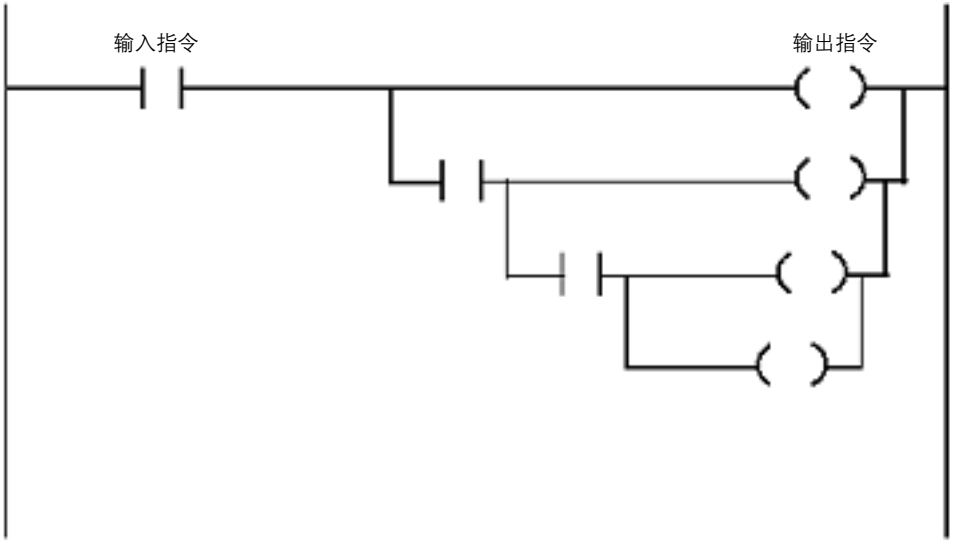
- *0*表示 *关*
- *1*表示 *开*

分支 (branch)

用户可以输入的并联分支层数没有限制。下图显示了一个包含 5 层的并联分支。主梯级是第一个分支层，后面有 4 个附加的分支。



用户最多能嵌入6层分支。下图显示了一个嵌套分支。底部的输出指令位于具有3层深度的嵌套分支中。



C

字节 (byte)

一个8位的存储单元。

高速缓冲存储器 (cache)

在MSG指令完成后，高速缓存连接仍可保持。这对用户程序重复执行MSG指令很有用，因为每次启动连接将会增加扫描时间。

下表显示了哪些信息需要使用连接和用户能否高速缓存这个连接。

信息类型:	通讯方式:	连接:	用户能够高速缓存:
CIP 数据表读或写	CIP	✓	✓
PLC2,PLC3,PLC5 或 SLC(所有类型)	CIP		
	带源 ID 的 CIP		
	DH+	✓	✓
通用 CIP	N/A	✓[1]	✓[2]
块传送读或写	N/A	✓	✓

[1] 只有少数目标模块需要连接。

[2] 只考虑需要高速缓存连接的目标模块。

按照下面的步骤，为用户可以高速缓存的信息选择一个高速缓冲存储器选项。

1. 用户的梯形图逻辑是否改变信息的路径？(例如，一条 MSG 指令是否与多个设备通讯？)

如果:	**那么**:
是	A.清除 Cache Connection（高速缓存连接） 选择框。 B.跳到第 2 步。
否	跳过第 2 步。

2. 这个控制器能把用户可以高速缓存的信息发送到多少个控制器？

如果:	**那么**:
16 个或少于 16 控制器	选择 (选中) *Cache Connection (高速缓存连接)* 选择框。
多于 16 个控制器	A.选择 16 个信息需求最频繁的控制器。 B.信息是否发送到其中的一个控制器？

如果:	**那么**:
是	选中 *Cache Connection (高速缓存连接)* 选择框。
否	清除 *Cache Connection (高速缓存连接)* 选择框

参见连接，非高速缓存连接。

状态改变 (COS)

I/O 模块上一个 I/O 点或一组 I/O 点的状态变化。

CIP

参见控制与信息协议。

通讯方式 (communication format)

定义 I/O 模块如何与控制器通讯。选择一种通讯方式来定义下列各项:

- 通过编程软件可以使用那些组态选项
- 标签结构体及组态方法

兼容模块 (compatible module)

一种电子锁保护方式，该方式要求厂家，目录号和实际模块的主要版本属性，以及为了与模块建立起连接所需要的在软件匹配中的组态模块。参见禁止锁,精确匹配。

连接 (connection)

是指在控制系统中从控制器到另一个模块的通讯机制。单个控制器所能建立的连接数量是有限制的。与I/O模块、消费型标签、生产型标签的通讯以及MSG指令都使用连接来传送数据。

消费型标签 (consumed tag)

该标签通过ControlNet网络或ControlLogix背板接收产生标签广播的数据。消费型标签必须满足:

- 控制器作用域
- 与远程标签 (生产型标签) 具有相同的数据类型 (包括任何数组维数)。

参见*生产型标签*。

连续性任务 (continous task)

连续运行的任务。

- 连续性任务在后台运行。因为所有没分配给其它操作 (例如运动控制，通讯，以及周期性任务) 的 CPU 时间都被用来执行连续性任务中的程序。
- 在该任务的最后一个程序完成后将重新启动连续性任务。
- 一个工程可以没有连续性任务。
- 如果使用，只能有一个连续性任务。
- 所有的周期性任务都能中断连续性任务。
- 当用户创建一个工程时，缺省的 *MainTask (主任务)* 是连续性任务。用户可以保留该任务，也可以更改它的属性 (名字，类型，等)`。

参见*周期任务*。

控制与信息协议 (Control and Information Protocol)

Allen-Bradley 的 Logix5000 系列控制设备使用信息传输协议。本地通讯协议用在 ControlNet 网络上。

控制器故障处理程序 (controller fault handler)

控制器故障处理程序是一个可选的任务，在下列情况下执行:

- 主要故障不是一个指令执行故障
- 编辑故障例程:
  - 不能清除主要故障
  - 存在故障
  - 不存在

用户只能为控制器故障处理创建一个程序。该程序创建后，用户必须组态一个例程作主例程。

- 控制器故障程序不执行故障例程。
- 如果用户为控制器故障程序指定了一个故障例程，控制器将永远不执行该例程。
- 用户可以创建另外的例程，并从主例程中调用它们。

控制器作用域 (controller scope)

在控制器中的任意位置均可存取数据。控制器中含有一个标签集，该集合中的标签可以被任意程序中的例程和别名标签引用，就像在控制器作用域内的其他别名一样。参见*程序作用域*。

协调系统时间 (CST)

一个 64 位数值，表示自 CST 主控制器起动计数以来的微秒数。

- CST 值是作为一个 DINT[2]元素的数组来存储的，其中:
  - 第一个元素存储在低 32 位
  - 第二个元素存储在高 32 位
- 用户可以使用 CST 时间标记来比较数据采样之间的相对时间。

计数器 (counter)

用于计数器指令的状态和控制信息的结构体数据类型。

D

数据类型 (data type)

对内存大小及内存格式的一种定义，当创建某一种数据类型的标签时将按此定义为其分配内存。

十进制 (decimal)

以 10 为基数输入或显示的整数值。没有前缀。不填充至整数的字长。参见*二进制，十六进制，八进制*。

描述 (description)

用户可以用可选文本来进一步描述该应用程序。

- 用户可以使用任何打印字符，包括回车符，制表符和空格。
- 描述不能下载到控制器中。它们保留在离线状态下的工程文件里。
- 描述的长度限制:
  - 标签的描述最多可以用 120 个字符。
  - 其它对象 (任务，程序，模块等) 的描述最多可以是 128 个字符。

维数 (dimension)

指定数组的大小。数组最多可以是三维。参见*数组* 。

DINT

一种用于存储 32 位 (4 字节) 有符号整数值的数据类型 (-2，147，483，648 至 2，147，483，647)。在 Logix5000 控制器中，整数使用 DINT 数据类型:

- 用 32- 位整数 (DINTs) 代替 16- 位整数 (INTs) 或 8- 位整数 (SINTs) 工作，Logix5000 控制器将更有效的执行，且使用的内存更少。
- 一般情况下，指令在执行期间使 SINT 或 INT 值转变成最佳数据类型 (通常 DINT 或 REAL 值)。这个过程需要额外的时间和内存，所以要将使用的 SINT 和 INT 数据类型减小到最少。

直接连接 (direct)

是指一种I/O连接，采用这种连接时控制器可以与I/O模块建立一种单机连接。参见*机架优化*。

禁止锁 (disable keying)

一种电子锁保护方式，这种方式不要求实际模块的属性，也不要求与软件中的组态模块匹配，但这种方式仍然可以建立与模块的连接。参见*兼容模块*，*精确匹配*。

下载 (download)

将工作站中的工程内容传送至控制器的过程。参见*上载* 。

E

消耗时间 (elapsed time)

单个任务内全部操作组态的执行所需要的时间。

- 如果控制器被组态成执行多个任务，则消耗时间还包括其它任务执行另外的操作所用 / 共享的时间。
- 在上线时，用户可以使用 Task Properties 对话框以 ms 形式来查看当前任务的最大扫描时间和最后扫描时间。这些值都是消耗时间，其中还包括用来等待较高优先级任务所花费的时间。

参见执行时间。

电子锁 (electronic keying)

1756I/O系列的一种功能部件，它可以要求模块执行一次电子检查，以确保实际模块和软件所组态的一致。通过软件可以使用户防止无意中使用故障模块或故障的模块版本。*参见兼容模块*，*禁止锁*，*精确匹配*。

元素 (element)

一个可寻址的数据单元，该数据单元是一个大型数据单元的子单元。数组的一个单元成为元素。

- 用户可以通过元素的下标来选定数组中的元素:

数组:	**指定**:
一维	*array_name[subscript_0]*
二维	*array_name[subscript_0,subscript_1]*
三维	*array_name[subscript_0,subscript_1,subscript_2]*

参见*数组*。

精确匹配 (exact match)

是指一种电子锁保护方式，为了建立与模块的连接，这种方式要求实际模块的全部属性（厂家，目录号，主要版本和次要版本）与软件组态的模块匹配。

执行时间 (execution time)

执行一个单独程序所需要的全部时间。

- 执行时间仅包括单个程序所使用的时间，以及在其它任务中执行另外的操作所共享 / 使用的时间。
- 当程序上线时，用户可以使用 *Programe Properties* 对话框来查看当前任务的最大扫描时间和最后扫描时间(以 us)。这些值都是程序的执行时间，其中不包括用来等待其它程序或较高优先级任务的时间。

参见*运行时间*。

指数 (exponential)

用科学计数法或指数格式显示或输入的实数。数字总是显示为: 总是在十进制小数点的左边有一位，然后是十进制部分，最后是一个指数。参见*格式*。

F

故障模式

控制器产生一个主要故障，不能清除该故障，并且已经停止工作

参见*主要故障*

浮点数

以浮点格式显示和输入的实数。十进制小数左边的位数不定，而由数的大小决定。参见*格式*。

H

十六进制数

以基数 16 显示和输入的整数值 (每个数字表示 4 位)。前缀为 16 #。将填充至布尔型或整数的字长 (1，8，16 或 32 位)。当显示时，每 4 个数字位一组，并且为了容易辨认出在组与组之间用下滑线隔开。参见*二进制，十进制，八进制*。

I

立即数

一个实际的 32- 位有符号实数值或整数值。但不是一个存储数值的标签。

索引

一种用于指定组内元素的索引。

指令

控制器基于指令前面的梯级条件 (梯级输入条件) 来运行梯形图指令。

在梯级中，只有输入指令可影响后面指令的梯级输入条件:

- 如果一条输入指令梯级输入条件为真，那么处理器执行这条指令，并且令梯级输出条件得到执行的结果。
  —如果指令执行为真，那么梯级输出条件为真。
  —如果指令执行为假，那么梯级输出条件为假。
- 一条输出指令不改变梯级输出条件。
  —如果一条输出指令的梯级输入条件为真，那么梯级输出条件被设置为真。
  —如果一条输出指令的梯级输入条件为假，那么梯级输出条件被设置为假。

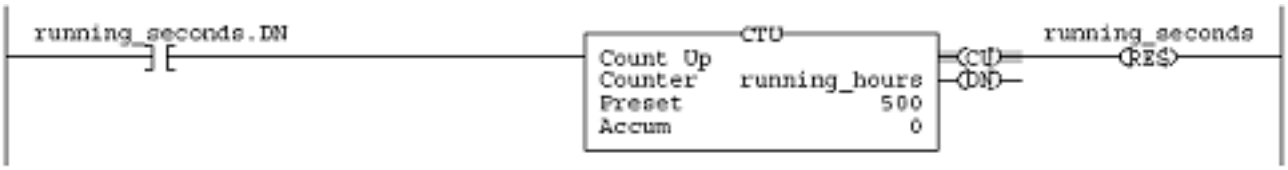
在 Logix5000 控制器中，用户可以给梯形图逻辑的每个梯级输入多个输出指令:

- 在梯级中顺序放置 (串行)
- 只要梯级的最后一条指令是输出指令，输出指令可放在输入指令之间。

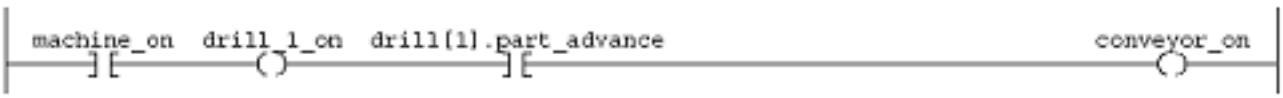
以下例子在一个梯级中使用了多个输出。

例 子 在一个梯级上放置多个输出

当 *running_seconds.DN* 闭合时，*running_bours* 开始加计数并将 *running_seconds* 复位。



当 *machine_on* 闭合，使 *drill_1_on* 为真，当 *machine_on* 和 *drill[1].part_advance* 都闭合时， *conveyor_on* 为真。



INT

一种用于存储16位整数的基本数据类型 (-32，768至+32，767)。少用这种数据类型:

- 一般情况下，指令在执行时将 SINT 或 INT 转换成为最佳数据类型 (optimal data type) (通常为 DINT 或 REAL)。
- 因为要求附加时间和内存，尽量少用 SINT 和 INT 数据类型

接口模块

一种预先布线的 I/O 现场接线设备。

L

只听连接

一种 I/O 连接，在这种连接中控制器拥有来自 I/O 模块的组态数据或为 I/O 模块提供组态数据。当主控制器在有效控制着I/O模块时，采取只听连接方式的控制器不能写入组态数据，而只能保持与 I/O 模块的连接。参看宿主处理器。

加载

将工程从非易失性内存中复制到控制器的用户内存中 (RAM)。这将会覆盖控制器中当前所有的工程。参见*非易失性内存，存储*。

M

主例程

当一个程序执行时，第一个例程执行。使用主例程来调用 (执行) 其它例程 (子例程)。

主要故障

一种非常严重的故障，如果不清除，那么控制器就会关闭。当主要故障发生时，控制器会:

1.主要故障位置位。

2.如果有用户提供的故障逻辑程序存在，则运行。

3.如果用户提供的故障逻辑程序不能清除故障，那么控制器进入故障模式。

4.在编程模式期间，根据它们的输出状态来设置输出。

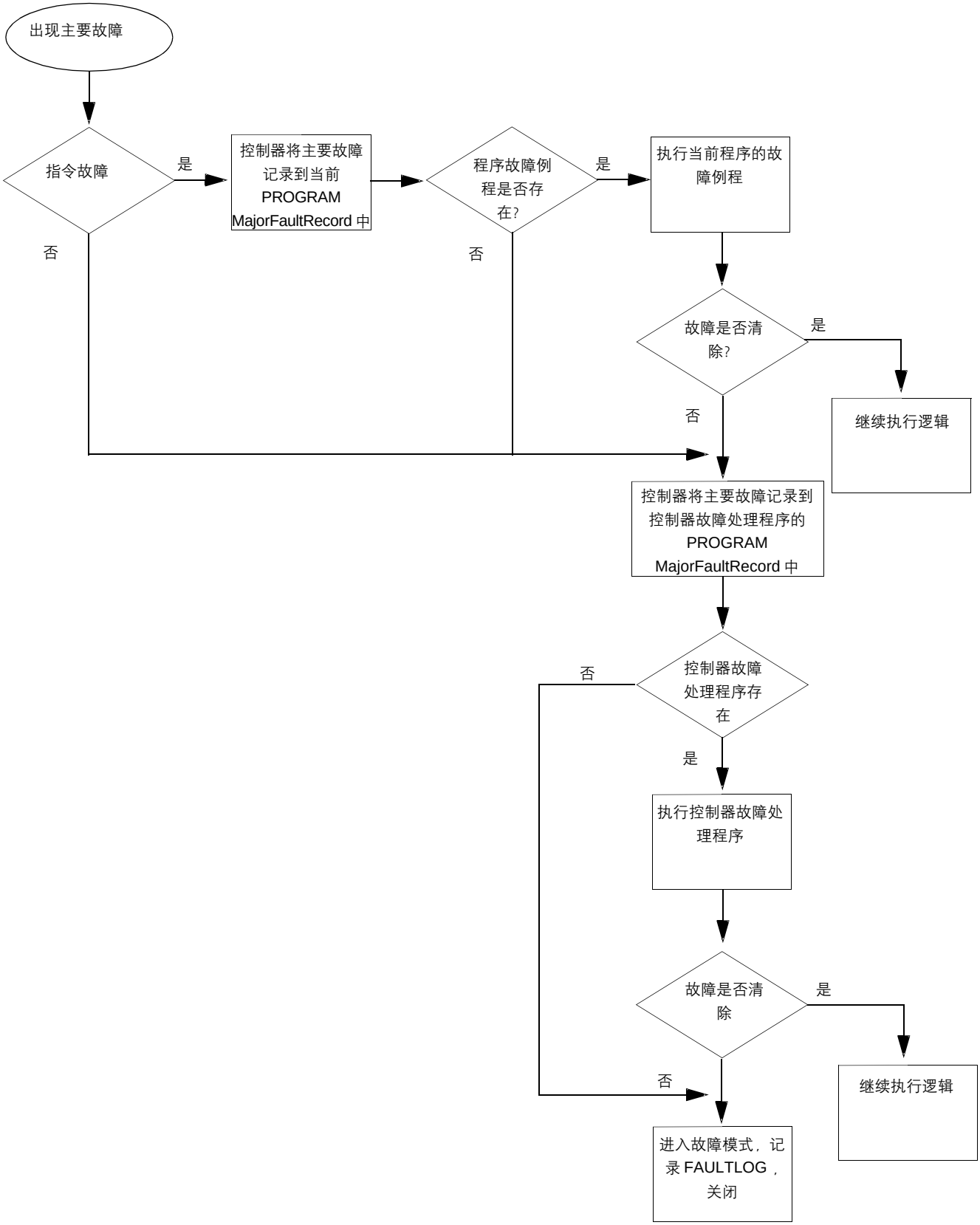
5.OK 指示灯闪烁红灯。

控制器提供了两级处理主要故障的方法:

- 程序故障例程
  - 每个程序有它自己的故障例程
  - 当一个指令出错时，控制器执行程序故障例程。
  -如果程序故障例程不能清除故障或程序故障例程不存在,那么控制器继续执行控制器故障处理程序(如果已定义)。
- 控制器故障处理程序
  -如果控制器故障处理程序不存在或不能清除主要故障,那么控制器进入故障模式并且关闭。这时，FAULTLOG(故障记录)被更新。(见下页。)
  - 所有的非指令故障 (I/O，任务看门狗，等。)直接执行控制器故障处理程序。(不调用程序故障例程。)

故障没有被清除,一直到两个附加故障也没被清除,就被记录在控制器故障记录中。

enter faulted mode record FAULTLOG shut down



参见故障状态，次要故障。

主要版本

1756模块系列具有主要及次要版本指示器。每当模块的功能有变化时，主要版本将随时更新。*参见电子锁，次要版本*。

主控制器 (CST)

在一个单独的框架内，有且只能有一个master，控制器将被设计作为协调系统时间 (CST) 主控制器 (master)。所有框架内的其它模块都将使它们的 CST 值与 CST master 同步。

成员

有自己的数据类型和名称的结构体中的元素，。

- 成员同样可以是结构体，即创建嵌套的结构体数据类型。
- 结构体中的每个成员的数据类型可以不同。
- 要使用结构体中的成员，格式如下:
  tag_name.member_name

例如:

地址:	**参考**:
timer_1.pre	*timer_1* 结构体的 PRE 值
input_load 作 load info 的数据类型 *input_load.height*	*height* 是 *input_load* 自定义结构体的成员

- 如果一个结构体嵌套在另一个结构体中，那么使用最高级别结构体的标签名，后面带有子结构体的标签名和成员名:
  tag_name.substructure_name.member_name

例如:

地址:	**参考**:
input_location 作 *location* 的数据类型 *input_location .load_info.height*	*input_location* 结构体中 *load_info* 结构体的 height 成员

- 如果结构体定义了一个数组，使用数组标签，跟在数组位置的后面，是任意子结构体及成员名。

  array_tag[position].member

  或

  array_tag[position].substructure_name.member_name

例如:

地址:	**参考**:
conveyor[10].source	*source* 是 *conveyor* 数组的第 11 个元素(数组元素是基于零的)。
conveyor[10].info.height	*height* 是 *conveyor* 矩阵 info 结构体的第 11 个元素(数组元素是基于零的)。

参见*结构体*

内存

安装在控制器内部的电子存储介质。用于保存程序和数据。

次要故障

一种严重程度不足以关闭控制器的故障:

如果发生:	**控制器**:
指令故障	1.将 S:MINOR 置位 2.记录故障信息到 PROGRAM 对象，MinorFaultRecord 属性。 3.将 FAULTLOG 对象，MinorFaultBits 属性的第 4 位置位。
周期任务重叠	将 FAULTLOG 对象，MinorFaultBits 属性第 6 位置位。
串行端口问题	将 FAULTLOG 对象，MinorFaultBits 属性第 9 位置位。
电池低电压	将 FAULTLOG 对象，MinorFaultBits 属性第 10 位置位。

要清除次要故障:

1. 在控制器项目管理器中，右键点击 *Controller name_of_controller* 文件夹并选择 *Properties*。

2. 点击 *Minor Faults* 选项。

3. 使用 *Recent Faults* 列表中的信息来校正故障。参见 A-3 页 ″次要故障代码″。

4. 点击 *Clear Minors* 按钮。

次要版本

1756模块系列具有主要及次要版本指示器。每当模块的功能有变化时，次要版本将随时更新。*参见电子锁，主要版本*。

多播

广播是指一种机制，一个模块采用这种机制在网络上发送数据时，数据可以同时被多个收听者收到。ControlLogix I/O 系列具有该特性，它支持多个控制器在同一时间从相同的 I/O 模块接收输入数据。

多主

一种组态设置，多个控制器具有完全相同的组态信息，可以同时拥有相同的输入模块。

N

名称

名称用于识别控制器，任务，程序，标签，模块等。名称约定遵循 IEC-1131-3 规范。一个名称:

- 必须以字母符号 (A-Z 或 a-z) 或下划线 (_) 开头
- 只能包括字母符号，数字符号和下划线
- 最多可以包含 40 个字符
- 不能连续使用下划线或用下划线作后缀
- 不区分大小写
- 下载到控制器

网络刷新时间 (NUT)

数据可以在 ControlNet 网络上发送的重复时间间隔。网络刷新时间范围从 2ms-100ms。

非易失性内存

在控制器没有电源或电池的情况下，控制器内存依然能保存其内容。*参见加载，保存*。

O

对象

一种用于存储状态信息的数据结构体。当用户输入一条GSV/SSV指令时，用户指定对象及想要访问的对象属性。在某些情况下，一个对象可能有多个相同类型的实例，因此用户可能还必须指定对象名称。例如，在用户的应用程序中可能有几个任务。每个任务都有自己的 TASK 对象，用户需通过任务名称来访问。用户可以访问这些对象。

八进制

以基数 8 显示和输入的整数值 (每个数字表示 3 位)。前缀为 8 #。将填充至布尔型或整数的字长 (1，8，16 或 32 位)。当显示时，每 3 个数字为一组，并且为了容易辨认在组与组之间用下划线隔开。参见*二进制，十进制，十六进制*。

离线

在工作站硬盘上浏览和编辑一个工程。参见*在线*。

在线

在控制器上浏览和编辑一个工程。参见*离线*。

最佳数据类型

Logix5000 指令实际使用的数据类型 (典型的有 DINT 和 REAL 数据类型)。

- 在指令集参考手册中，粗体显示的最佳数据类型。
  - 如果所有的指令操作使用下列数据类型，那么指令将执行的更快，占用的内存更少:
    - 相同的数据类型
    - 最佳数据类型
- 如果用户混合数据类型并且使用的标签也不使用最佳数据类型，那么控制器将根据下列原则来转换数据类型:
  - 所有的操作对象是否是实型数值?

如果:	**那么输入操作对象 (例如，源，表达式中的标签，上下限) 转换数据类型**:
是	REALs
否	DINTs

  - 指令执行完之后，如果需要，再将结果 (一个 DINT 或 REAL 值) 转换成目的数据类型。
- 因为数据转换需要占用额外的时间和内存，用户可以用以下方法提高程序执行的效率:
  - 整个指令中使用相同的数据类型
  - 少用 SINT 或 INT 数据类型

换句话说，在用户指令中所有的标签连同立即数值都使用DINT或REAL数据类型。

• 下表概述了控制器是怎样在不同数据类型之间进行数据转换的:

<table>
<tr><th>转换:</th><th>结果:</th></tr>
<tr><td>大的整数到小的整数</td><td>控制器将截取大整数的高位部分并产生一个溢出。

例如:
<table><tr><th></th><th>十进制</th><th>二进制</th></tr><tr><td>DINT</td><td>65,665</td><td>0000_0000_0000_0001_0000_0000_1000_0001</td></tr><tr><td>INT</td><td>129</td><td>0000_0000_1000_0001</td></tr><tr><td>SINT</td><td>-127</td><td>1000_0001</td></tr></table></td></tr>
<tr><td>SINT 或 INT 到 REAL</td><td>数据精度不改变</td></tr>
<tr><td>DINT 到 REAL</td><td>数据精度可能改变。这两种数据类型都占用 32 位空间，但是 REAL 类型要占用一些空间来存储小数值。如果精度改变，控制器将使用 DINT 数据的有效位。</td></tr>
<tr><td>REAL 到整数型</td><td>控制器将小数部分四舍五入并截取其高位部分。如果数据丢失，控制器将溢出状态标志位。

四舍五入方法如下:
• 除了 X.5 以外其它的数保留为离它最近的数。
• X.5 保留为离它最近的偶数。

例如:
<table><tr><th>REAL (源)</th><th>DINT (结果)</th></tr><tr><td>-2.5</td><td>-2</td></tr><tr><td>-1.6</td><td>-2</td></tr><tr><td>-1.5</td><td>-2</td></tr><tr><td>-1.4</td><td>-1</td></tr><tr><td>1.4</td><td>1</td></tr><tr><td>1.5</td><td>2</td></tr><tr><td>1.6</td><td>2</td></tr><tr><td>2.5</td><td>2</td></tr></table></td></tr>
</table>

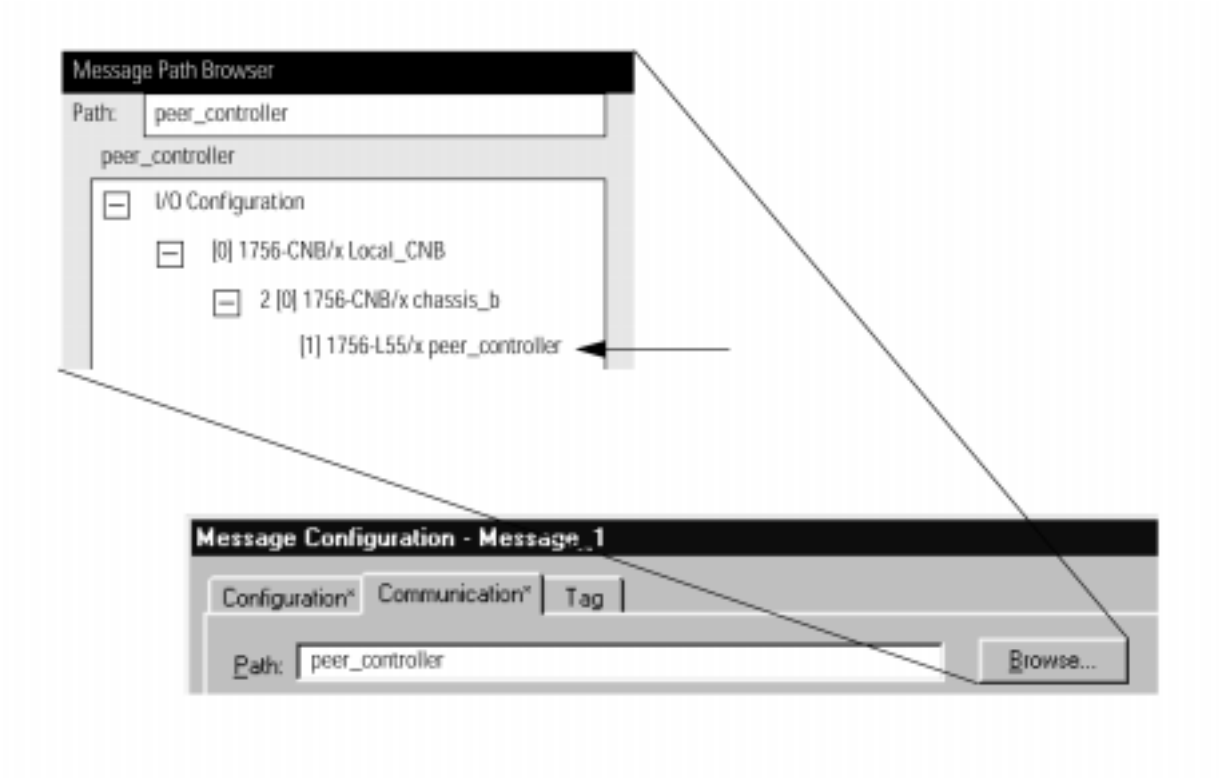
主控制器

创建原始组态及与模块通讯连接的控制器。主控制器可以写入组态信息数据并且可以和模块建立连接。参见*只听连接*.

P

路径

路径描述了一条信息到达目的地的路由。如果控制器的 I/O 组态中包含这个目的设备，使用 *Browse* (浏览) 按钮来选择这个设备。这样就会自动地定义路径。



如果这个 I/O 组态中不包含这个目的设备，那么请使用如下格式输入目的路径:

端口，地址，端口，地址

其中:	**因为这个**:	**程序列表**:
端口	任何 1756 控制器或模块的背板	1
	Logix5000 控制器的 DF1 端口	2
	1756-CNB 模块的 ControlNet 端口	
	1756-ENBx 或 -ENET 模块的 Ethernet 端口	
	1756-DHRIO 模块中在通道 A 中的 DH ＋端口	
	1756-DHRIO 模块中在通道 B 中的 DH ＋端口	3
地址	ControlLogix 背板	槽数
	DF1 网络	工作站地址 (0-254)
	ControlNet 网络	节点数 (十进制 1-99)
	DH ＋网络	8# 后面接节点数 (八进制 1-77) 例如，要定义 37 的八进制节点地址，输入 8#37。
	EtherNet 网络	用户可以使用以下任何形式在 EtherNet/IP 网络上定义一个模块。 IP 地址 (例如，130.130.130.5) IP 地址 端口(例如，130.130.130.5：24) DNS 名称 (例如，tanks) DNS 名称 端口(例如，tanks：24)

参看连接。

周期性任务

操作系统按照重复的时间周期来触发任务。

- 使用一个周期性任务，来执行要求精确的或确定性的函数。
- 只要计满时间周期，都将触发任务并执行程序。
- 在任务中程序将建立的数据和输出量的值一直保持,直到下一个任务的执行或被另一个任务所操作。

- 用户可以将时间周期设置为 1ms 到 2000s。缺省值为 10ms。

注 意

确保这个时间周期比指派给任务的所有程序执行时间的和要长。如果控制器发现一个已经运行的周期性任务被触发，则出现一个次要故障。

- 周期性任务总是中断连续性任务。
- 根据优先级别的不同，控制器中的一个周期性任务可能中断另一个周期性任务。

参看连续任务。

周期性任务重叠

当一个任务正在执行时，如果再次触发同一个任务将导致产生周期性任务重叠条件。任务的执行时间大于为任务组态的周期频率。参见*周期性任务*。

预先确定的结构体

一个结构体数据类型为一个特殊指令存储相关信息，例如 TIMER 结构体为计时器指令存储相关信息。无论系统硬件结构如何，预先确定的结构体通常是可用的。参看*产品定义结构体*。

预扫描

预扫描是在进行 Run 模式转换期间的一种媒介扫描。

- 当用户将控制器从 Program 模式切换到 Run 模式时，控制器将执行预扫描。
- 预扫描检测所有的程序和指令，并根据结果初始化数据。
- 一些指令在预扫描和普通扫描期间的执行不同。

优先级

如果两个任务同时被触发，确定哪个任务先执行。

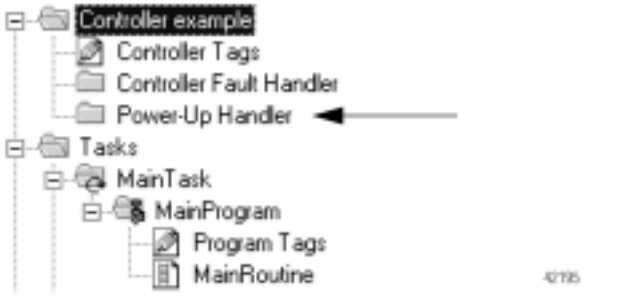
- 优先等级较高的任务最先执行。
- 优先等级的范围从 1-15，其中 1 为最高优先等级。
- 一个较高优先级的任务将中断其它优先级较低的任务。
- 如果具有相同优先级的两个任务同时被触发，控制器将每隔一毫秒在两个任务间进行切换。

后期扫描

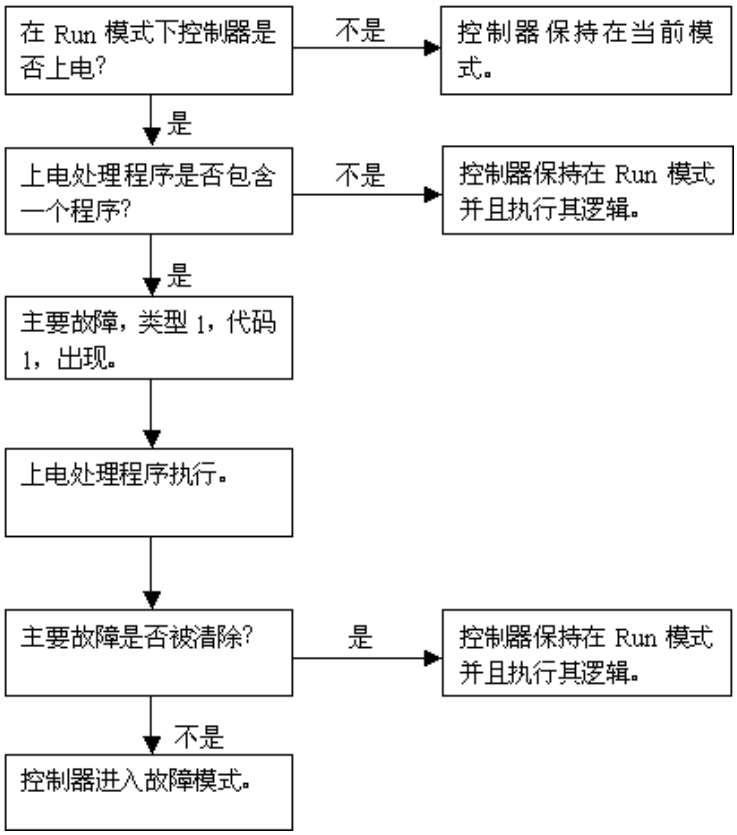
后期扫描是控制器的一种功能，采用这种方式时，在禁止该程序以前将按复位指令和数据的顺序检查程序内的逻辑。

上电处理程序

在 Run 模式下给控制器电源上电时，可执行一个可选任务。要使用上电处理程序，用户必须创建一个上电程序和与之相关的主例程。



上电程序将按如下步骤执行:



生产型标签

使用这种标签时，控制器将使其可被其它控制器使用。生产型标签总是在控制器作用域中的。参见消费型标签。

产品定义结构体

是一种由软件和控制器自动定义的结构体数据类型。通过组态一个 I/O 模块，用户即可把产品定义结构体添加到模块中。

程序

一组相关的例程和标签。

- 每个程序都包含程序标签，一个主执行例程，其它例程和一个可选故障例程。
- 要执行程序中的例程，用户将向任务中指定 (规划) 这个程序:
  - 当一个任务被触发后，任务中确定的程序将从第一个到最后一个完成执行。
  - 当任务执行一个程序时，将首先执行程序的主例程。
  - 主例程可以按照顺序，使用 JSR 指令来执行子程序。
- *未确定程序*文件夹包含没指定到一个任务的程序。
- 如果程序的逻辑出现一个主要故障，程序将执行跳转到已组态的故障例程。
- 一个程序的例程可以访问以下标签:
  - 编程的程序标签
  - 控制器标签
- 例程不可以访问其它程序的程序标签。

参看例程，任务。

程序作用域

只能在当前程序中存储数据。每一个程序中包含一个标签集，它们只能被该程序中的例程和别名标签引用。参见控制器作用域。

工程文件

这个文件在用户的工作站 (服务器) 上，用来存储梯形图逻辑，组态，数据和控制器的文件。

- 具有扩展名为.ACD 的工程文件。
- 当用户创建一个工程文件时，文件名即为控制器名。
- 控制器名不受程序文件名的约束。如果用户用其它的名字来存储当前程序文件，控制器的名字不改变。
- 如果控制器名与程序文件名不同,RSLogix5000软件的标题栏会将这两个名字都显示出来。

参见应用程序。

R

最佳机架连接

是指一种 I/O 连接，采用这种方式时 1756-CNB 模块将把所有的数字 I/O 字都集中到一个机架映像表中 (类似于 1771-ASB)。一个最佳机架连接将保存 ControlNet 连接及频带宽度，不过当使用这种连接方式时，仍然可以使用一些有限的状态和诊断信息。参见*直接连接*。

速率

对一个周期性任务，控制器执行这个任务的速率从 1ms 到 2,000,000ms (2000 秒)。缺省值为 10ms。

实数

一个用于存储 32- 位 IEEE 浮点数的数据类型，使用以下范围:

- -3.402823E^{38} 到 -1.1754944E^{-38} (负数)
- 0
- 1.1754944E^{-38} 到 3.402823E^{38} (正数)

REAL 数据类型也可以存储 ± 无穷大到 ± NAN，但是软件的显示格式不一致。

显示格式:	**等价的**:	
实数	+ 无穷大	1.$
- 无穷大	-1.$	
+NAN	1. #QUAN	
-NAN	-1. #QUAN	
指数	+ 无穷大	1.#INF000e+000
	- 无穷大	-1.#INF000e+000
	+NAN	1. #QUAN00e+000
	-NAN	-1. #QUAN00e+000

带电插拔 (RIUP)

是 ControlLogix 的一个特点，它允许用户在框架带电的情况下安装和拆除模块。

请求数据包时间间隔 (RPI)

当在网络上进行通讯时，RPI 是连续生成输入数据之间时间最大值。

- 一般情况下，该间隔以微秒来组态。
- 实际的数据生成时间受最大的网络刷新时间的倍数限制,网络刷新时间小于选择的 RPI。
- 使用 2n 倍的 ControlNet 刷新时间 (NUT)。

例如，NUT 为 5ms，输入的速率为 5，10，20，40ms 等。

参看网络更新时间 (NUT)。

例程

例程是指采用一种编程语言编写的一组逻辑指令，例如一个梯形图。

- 例程为控制器中的工程提供了可执行代码(类似于 PLC 或 SLC 控制器中的程序文件)。
- 每一个程序都有一个主例程:
  - 当控制器触发相关的任务并执行相关的程序时，主例程是第一个被执行的例程。
  - 为了访问程序中的其它例程，请在主例程中输入 JSR 指令。
- 用户也可以指定一个可选的程序故障例程。
  - 如果在任何一个相关程序中的例程产生一个主要故障，控制器执行程序故障例程。

参见程序，任务。

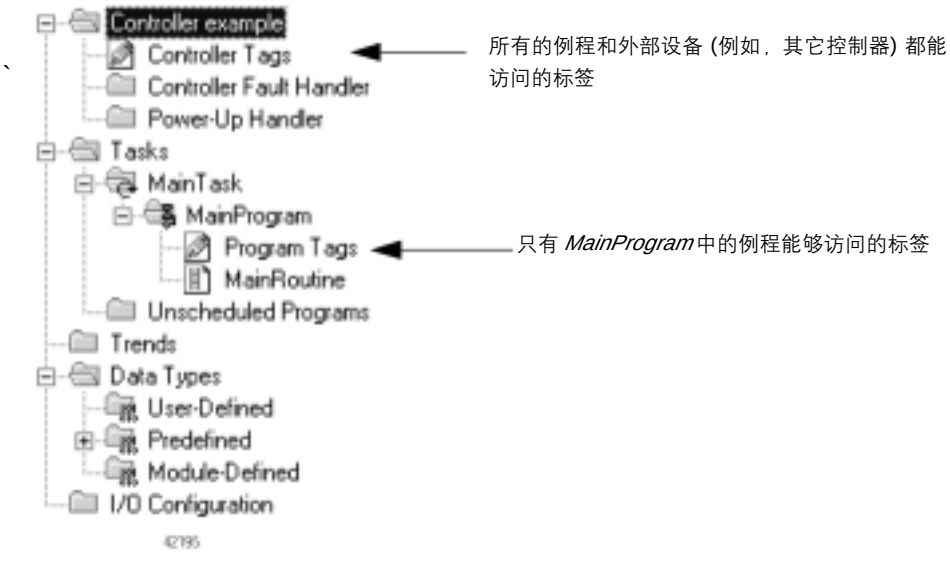
S

扫描时间

参见占用时间，执行时间。

作用域

定义用户可以访问一组特定标签的区域。当用户创建一个标签时，用户将其指定(范围)到一个具体的程序中作为控制器标签或程序标签，描述如下。



用户可以创建相同名称的多个标签:

- 每一个标签必须有不同的作用域。例如，一个控制器标签或其它标签可以在不同的程序中作为程序标签。或者，每一个标签在一个不同的程序中可以作为程序标签。

- 在一个程序中，如果一个相同名字的标签在那个程序中作为程序标签使用，用户不能使用这个名字作控制器标签。

参见控制器作用域，程序作用域。

单整数 (SINT)

一种用于存储8_位有符号整数的基本数据类型（-128-＋127）。少用这种数据类型。

- 一般情况下，指令在执行过程中将 SINT 或 INT 转换为最佳数据类型 (通常是 DINT 或 REAL)。由于这需要额外的时间和内存空间，将 SINT 和 INT 数据类型的使用降到最低。

源密码

一个可以限制查看例程权限的技巧。

- 用户可以为一个或多个例程指定源密码。
- 源密码的命名规则和其它 RSLogix5000 成分相同，例如：例程，标签和模块。
- 要给例程指定源密码 (保护例程)，使用 RSLogix5000 源保护软件。
- 一个源密码文件 (sk.dat) 存储源密码。源密码文件与 RSLogix5000 的工程文件 (.acd) 是分离的。
- 为了查看一个被源密码保护的例程，用户必须拥有这个源密码。
- 没有源密码，用户不能打开一个例程。RSLogix5000 软件的状态行会显示 ″源不可用。″
- 不管源密码是否可用，用户都能下载并执行所有例程。

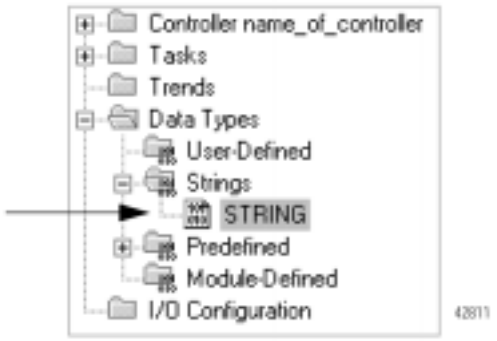
参看名称

保存

将一个工程复制到控制器的非易失性内存中。这将覆盖当前非易失性内存中所有的工程。*参见加载，非易失性内存。*

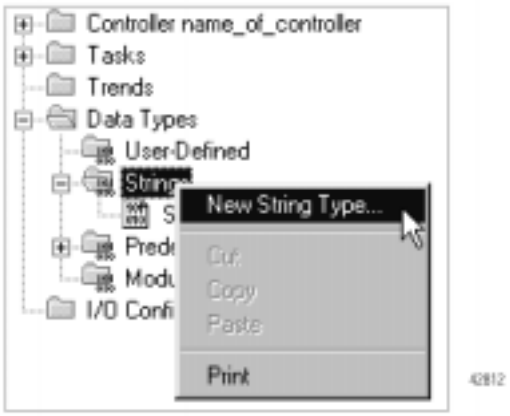
字符串

一组存储 ASC II 字符的数据类型。



用户使用缺省的 STRING 数据类型。能保存 82 个字符。

或



用户创建一个新的字符串数据类型用来保存用户定义的字符数。

每个字符串数据包含以下成员:

名称:	数据类型:	描述:	备注:
LEN	DINT	字符串中 字符数量	LEN 会自动地刷新新的字符数量: • 使用字符串浏览器对话框输入字符 • 使用指令读取、转换或处理字符串 LEN 显示当前字符串的长度。DATA 成员 可能包含 LEN 计算范围之外的额外的旧字符。
DATA	SINT 数组	字符串中的 ASCII 码	• 为了访问字符串中的字符，写出标签名的地址。 例如，要访问 *string_1* 标签中的字符，输入 *string_1* • DATA 数组的每个元素包含一个字符。 • 用户可以创建新的字符串数据类型用来存储数量不等的字符。

新的字符串数据类型在以下情况下是很有用的:

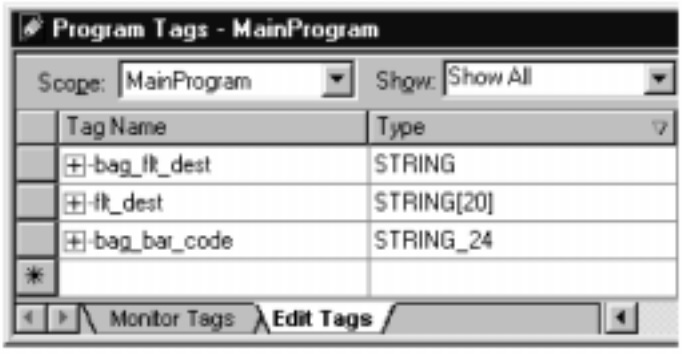
- 如果用户有大量的少于82个字符的固定大小的字符串，用户可以通过创建一个新的字符串数据类型来保存。
- 如果用户必须处理大于82个字符的字符串，用户可以创建一个新的字符串数据类型来满足这些字符所需的数量。

重 要

当用户创建一个新的字符串数据类型时一定要谨慎。如果用户以后想改变字符串数据类型的大小，用户可能会丢失当前使用这一数据类型的标签中的所有数据。

如果用户：	那么：
将字符串的数据类型变小	数据被截取 。 LEN 无变化。
将字符串的数据类型变大	数据和 LEN 被复位为零。

下面的例子说明 STRING 数据类型和一个新的字符串数据类型。



这个标签使用缺省的 STRING 数据类型。

这个标签是缺省的含 20 个数组元素的 STRING 数据类型

这个标签使用一个新的字符串数据类型。

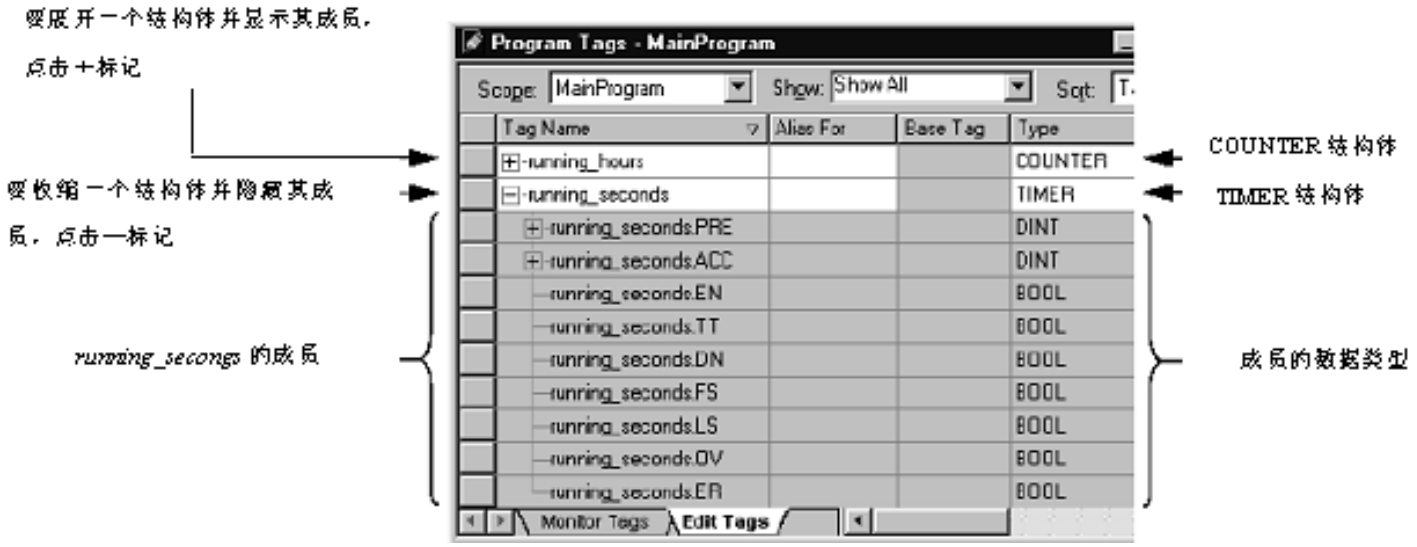
- 用户命名这个字符串数据类型为 *STRING_24*。
- 这个新的字符串数据类型只能存储 24 个字符。

结构体

一些数据类型是一个结构体。

- 结构体用来存储一组数据，结构体中的每个成员可以是不同的数据类型。
- 在一个结构体中，每一个单个的数据类型被称为一个成员。
- 象标签一样，成员拥有名字和数据类型。
- 用户创建自己的结构体，被称作自定义数据类型，使用单个标签和大部分其它的结构体的**任意组合**。
- 向一个结构体中拷贝数据，请使用COP指令。参见 *Logix5000 Controllers General Instruction Set Reference Manual*《*Logix5000 控制器指令集参考手册*》。出版号 1756-RM003。

COUNTER 和 TIMER 数据类型是通常在结构体使用的例子。



参见成员，自定义数据类型。

格式

是指数据显示的格式，参见 *ASC II 码，二进制，十进制，指数，浮点数，十六进制，八进制。*

系统内务处理时间片

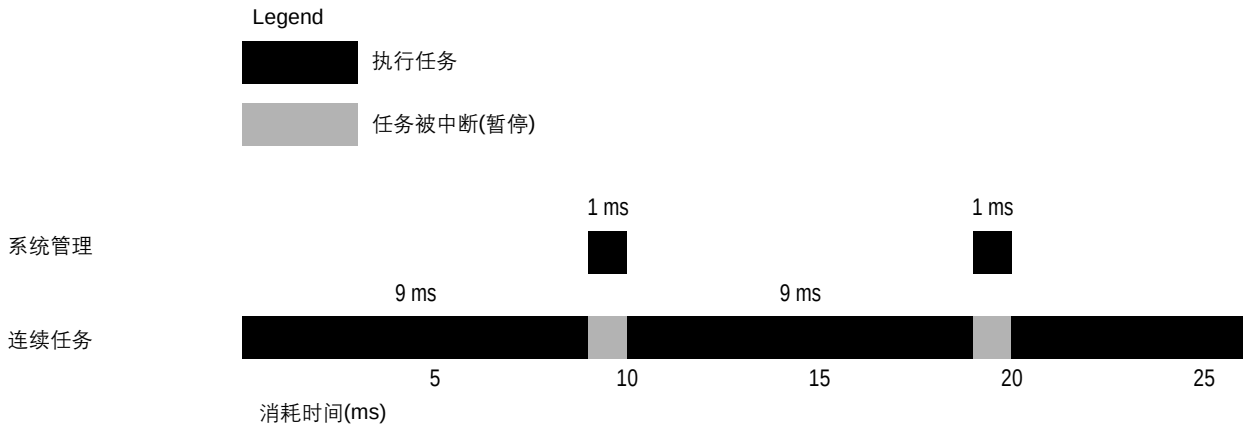
指定控制器时间的百分比 (包括完成定期任务的时间)，分配用来执行通讯和后台功能 (系统内务操作) 的时间:

- 每次控制器执行内务操作功能的时间等于 1 毫秒。
- 如果控制器完成内务操作的时间少于 1 毫秒，重新开始连续的任务。
- 通讯和后台功能包括以下内容:
  - 与程序和 HMI 设备进行通讯 (如 RSLogix5000 软件)
  - 信息响应
  - 发送信息，包括块传送
  - 重建并监测 I/O 连接 (如 RIUP 条件)。这不包括出现在程序执行期间的常规的 I/O 通讯
  - 控制器串行端口与其它的 ControlLogix 设备经由 ControlLogix 背板的网桥通信
- 如果通讯没有完成的足够快，将增加系统内务处理时间片

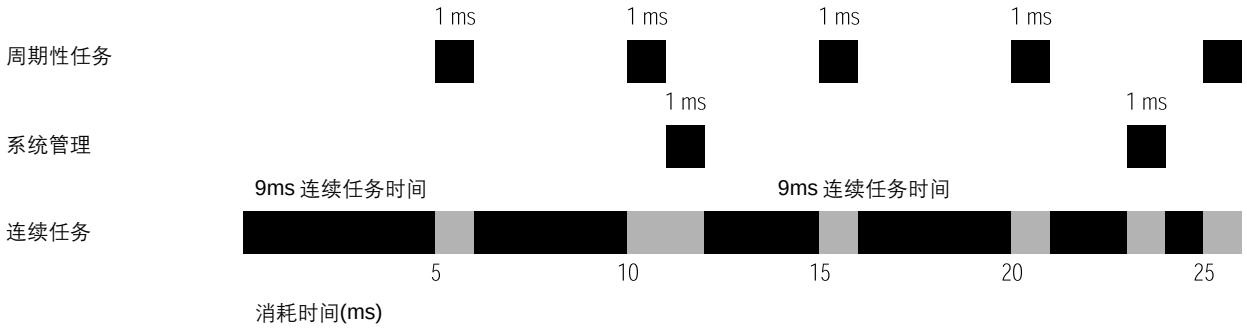
下面的表格指出连续任务和系统内务处理功能的比率:

时间片:	连续任务运行:	内务处理的出现等于:
10%	9ms	1ms
20%	4ms	1ms
33%	2ms	1ms
50%	1ms	1ms

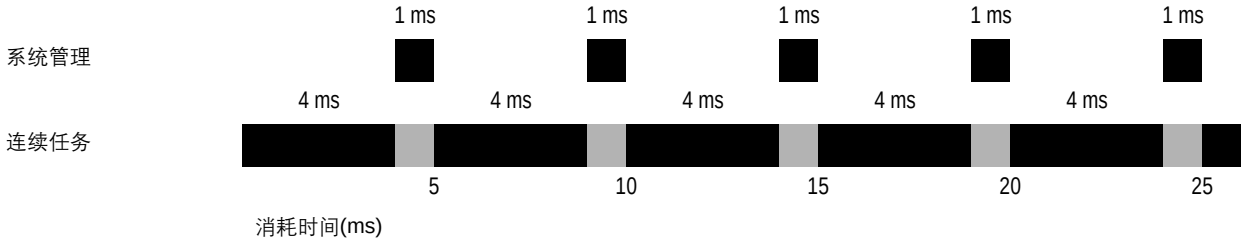
在缺省时间片的10%，系统内务处理每9ms（属于连续任务时间）中断连续任务一次。



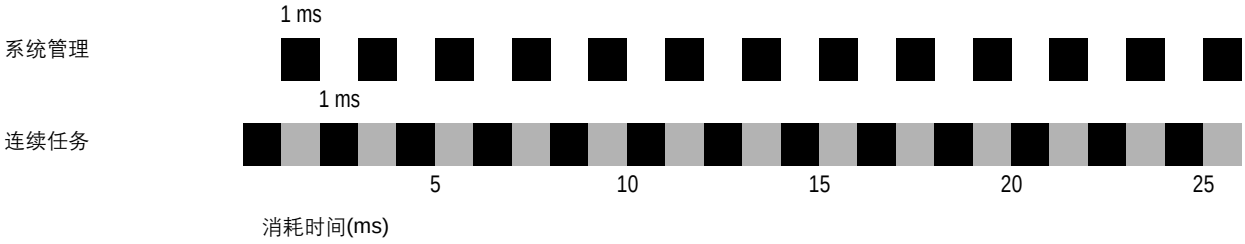
在系统管理执行过程中，周期性任务的中断增加了消耗时间（时钟时间）。



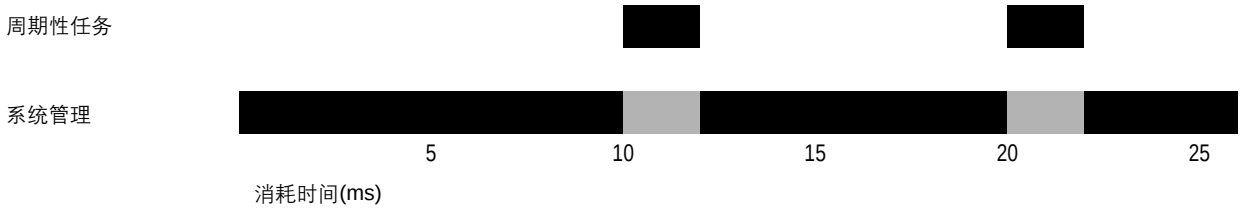
如果将时间片增加至20%，系统管理每4ms（属于连续任务时间）中断连续任务一次。



如果增加时间片到 50%，系统管理每 1ms（属于连续任务时间）中断连续任务一次。



如果控制器只包含一个周期性任务，系统管理时间片的值不受影响。只要周期性任务不运行时，系统管理就运行。



修改系统管理时间片：

1. 打开 RSLogix5000 工程。

2. 在控制器项目管理器中，右键单击 *Controller_name_of_controller* 文件夹并选择 *Properties*。

3. 点击 *Advanced* 选项。

4. 在 *System Overhead Time Slice* 文本框中，输入或选择内务处理时间的百分比 (10%-90%)。

5. 点击 *OK*。

T

标签

在控制器内存中保存数据的区域名字

- 标签是指定内存的基本结构，可以访问梯形逻辑的数据，也可以监测数据。
- 给一个标签分配的最小的内存单元是四个字节。
  - 当用户创建一个标签来保存一个BOOL,SINT,或INT (这些均小于四个字节) 数据时，控制器仍然分配四个字节的空间，但数据只占用它所需要的那部分空间。
  - 自定义数据类型和数组将数据保存在连续的内存中，并将小的数据类型的数据保存在 32 位的两个字中。

下面是给不同标签分配内存的例子:

- *start*，使用 BOOL 数据类型

内存	**位**	
分配	**31** **1**	**0**
内存分配	不使用	*start*

- *station_status*，使用 DINT 数据类型

内存	**位**	
分配	**31**	0
内存分配	*station_status*	

- *mixer*，使用自定义数据类型

内存	**位**			
分配	**31** **24**	**23** **16**	**15** **8**	**7** **0**
内存分配 1	*mixer.pressure*			
内存分配 2	*mixer.temp*			
内存分配 3	*mixer.agitate_time*			
内存分配 4	未用	未用	未用	0 位 *mixer.inlet* 1 位 *mixer.drain* 2 位 *mixer.agitate*

- *temp_buffer*，使用包含 4 个 INT 数据类型的数组（INT[4]）：

内存分配	位	
	31 **16**	**0**
内存分配 1	*temp_buffer[1]*	*temp_buffer[0]*
内存分配 2	*temp_buffer[3]*	*temp_buffer[2]*

参见*别名标签，基本标签，消费者标签*

任务

给正在执行的程序安排执行顺序。

- 缺省状态下，每一个新的工程文件包含一个预先组态好的连续性任务。
- 用户可以根据需要组态另外的周期性任务
- 一个任务为一组程序提供规划和优先信息，这些程序是基于指定标准执行的。
- 一旦一个任务被触发 (激活)，所有被指定 (规划) 到这个任务里的程序就会按一定的顺序执行，这些均显示在控制器项目管理器中。
- 用户可以每次只将一个程序指定到一个任务中。

参见 *continuous task ,periodic task.*

时间戳

当变化发生时，ControlLogix 控制器的程序就记录一个带相关时间值的输入数据的变化。

U

无高速缓冲连接

使用MSG 指令时，一个无高速缓冲连接显示控制器在完成MSG 指令后断开连接。清除这个连接以留给其它控制器使用。参见 *connection,cached connection*.

单向连接

数据只向一个方向传输的连接：从发送者到接收者。参见 *connection,bidirectional connection.*

上载

处理器将控制器中的内容传输到工作站的一个工程文件中。

如果用户没有这个控制器的工程文件,可以从控制器上载并创建一个新的工程文件。但是并不能把控制器中工程文件所有的项目都上载上来。如果用户从控制器中上传一个工程文件，在新建的工程文件中将不包括:

- 梯级的注释
- 标签，任务，程序，模块，或自定义结构体的描述
- 别名标签连接 (一个别名指向其它别名标签)

别名标签连接并没有在控制器中完全重建。可能存在一个数据条目对应多个别名标签名，这时固件和软件将选择一个最适合的别名标签来跟它对应，这也许不影响在原始工程文件中指定的别名标签。

参见 *download*。

自定义数据类型

用户也可以创建一个自己的结构体,称作自定义数据类型 (也可以叫自定义结构体)。一个自定义数据类型将不同的数据类型组合到一起称作一个实体。

- 在一个用户自定义数据类型中用户可以定义成员。
- 跟标签一样，成员也有名字和数据类型。
- 用户的定义中可以包括数组和结构体。
- 只要创建了一个自定义数据类型，用户就可以用这个数据类型来创建一个或多个标签。
- 尽量少用下列的数据类型，因为它们将增加控制器内存和程序的执行时间:
  - **INT**
  - **SINT**

例如，一些系统的值使用 SINT 或 INT 数据类型。如果用户创建一个自定义数据类型来保存这些数值，就必须使用与之相一致的 SINT 或 INT 数据类型。

- 如果用户使用的是代表I/O设备的的成员，就必须使用梯形逻辑将数据从结构体成员中拷贝到对应的 I/O 标签中。参见 8-1 页 ″缓存 I/O″。
- 当你看见 OOL,SINT,或 INT 数据类型时，将相同数据类型的数据按下列顺序放在成员中:

效率高
BOOL
BOOL
BOOL
DINT
DINT

效率低
BOOL
DINT
BOOL
DINT
BOOL

- 用户可以使用一维数组。
- 只有程序离线时用户才能创建，编辑和删除自定义数据类型。
- 如果用户想修改一个自定义数据类型和大小,那么现有数据类型中所有标签的值将会置零(0)。
- 要想将数据复制到一个结构体，使用 COP 指令。参见 *Logix5000 Controllers General Instruction Set Reference Manual《Logix5000 控制器指令集参考手册》*，版本号 1756-RM003

参见 *structure.*

W

看门狗

在触发主要故障之前指定任务的运行时间。

- 每一个任务都有一个看门狗时钟来监视任务的执行。
- 看门狗时钟范围为 1ms 到 2,000,000ms（2000s）。缺省值为 500ms。
- 当任务开始执行时看门狗时钟开始计时，当任务中所有的程序执行完之后停止计时。
- 如果任务执行超过看门狗时间，将出现一个主要故障: (这个时间包括被其它任务中断的时间。)
- 如果一个任务在执行过程中再次被触发 (连续性任务重叠) 将出现一个看门狗超时故障 (主要故障)。低优先级任务被高优先级中断，这样就延长了低优先级任务的执行时间，也会导致超时故障。
- 用户可以用控制故障处理程序来清除看门狗故障。不管控制故障处理程序是否清除看门狗故障，如果相同的逻辑扫描再次出现看门狗故障，这就说明控制器进入了故障的模式。

注 意

如果看门狗时钟达到预设置，将出现一个主要故障。根据控制器故障处理程序，控制器可能被关闭。

要修改一个任务的看门狗时间:

1. 打开 RSLogix5000 工程。
2. 在控制器项目管理器中，右键点击 *name_of_task* 并选择 *Properties (属性)*。
3. 点击 *Configuration (组态)* 选项。
4. 在 *Watchdog (看门狗)* 文本框中输入看门狗时间。
5. 点击 *OK (完成)*。

欢迎访问我们的网址：

www.rockwellautomation.com.cn
www.rockwellautomation.com
www.theautomationbookstore.com

Rockwell Automation Headquarters 1201 South Second Street, Milwaukee, WI 53204, USA, Tel: (1)414 382-2000, Fax: (1)414 382-4444

香港 – 香港铜锣湾威菲路道 18 号万国宝通中心 27 字楼 电话：(852)28874788 传真：(852)25109436

北京 – 北京市建国门内大街 18 号恒基中心办公楼 1 座 4 层 邮编：100005 电话：(8610)65182535 传真：(8610)65182536

上海 – 上海市仙霞路 319 号远东国际广场 A 幢 7 楼 邮编：200051 电话：(8621)62351098 传真：(8621)62351099

厦门 – 厦门市湖里工业区悦华路 38 号 邮编：361006 电话：(86592)6022084 传真：(86592)6021832

沈阳 – 沈阳市沈河区青年大街 219 号华新国际大厦 15-F 单元 邮编：110015 电话：(8624)23961518 传真：(8624)23963539

武汉 – 武汉市青山区和平大道 939 号 13 层 邮编：430081 电话：(8627)86543885 传真：(8627)86545529

广州 – 广州市环市东路 362 号好世界广场 2703-04 室 邮编：510060 电话：(8620)83849977 传真：(8620)83849989

重庆 – 重庆市渝中区邹容路 68 号大都会商厦 2506 室 邮编：400010 电话：(8623)63702668 传真：(8623)63702558

大连 – 大连市西岗区中山路 147 号森茂大厦 11 层 邮编：116011 电话：(86411)3687799 传真：(86411)3679970

西安 – 西安市南大街 30 号中大国际大厦 505 室 邮编：710002 电话：(8629)7203143 传真：(8629)7203123

深圳 – 深圳市深南东路 5047 号深圳发展银行大厦 15L 邮编：518001 电话：(86755)25847099 传真：(86755)25870900

Rockwell Automation

Publication 1756-pm001D-ZH April 2003

Copyright 2003 Rockwell International Corporation. Printed in CHINA.