1507, TFDCN-010-007, 001

NACHI

FD/CFD 控制装置
操作说明书
软件 PLC 篇

第 7 版

·在使用机器人之前，请详读本操作说明书，并请遵从所有关于安全事项与正文的指示。
·关于本机器人的安装、操作、维修，请仅由接受过本公司机器人讲习的人员进行。
·在使用本机器人的时候，必须遵守各个国家有关工业机器人的法律以及安全相关的法律条例。
·务必将本操作说明书交付给实际操作的人员。
·有关本操作说明书的不明之处以及有关本机器人的售后服务，请向记载在封底中的敝公司的各服务中心查询。

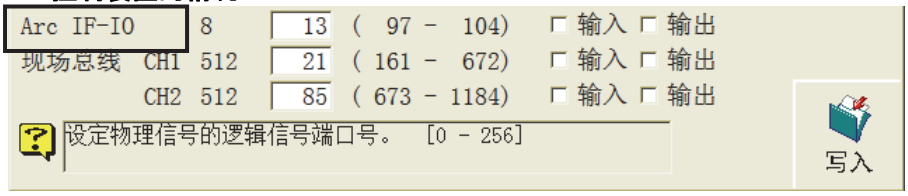
株式会社 不二越

面向 CFD 控制装置的操作者

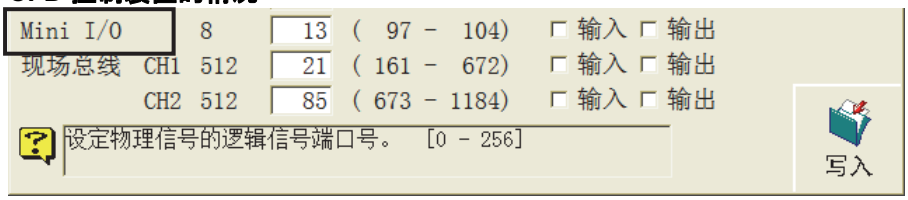
(1) 「**Arc IF-IO（电弧 I/F 基板）**」的名称成为「**Mini I/O(小型 I/O 基板)**」。（地址为通用）

（参照）下面显示＜常数设定＞［6 输出输入信号］［15 硬件设定］的画面。

FD 控制装置的情况



CFD 控制装置的情况



(2) 根据硬件构造不存在「**I/O 基板 3**」（数字 I/O 基板 3）。

(3) 下面显示工厂出货状态的输入输出信号的概要。

输入信号

逻辑输入信号	物理输入信号	备注
I1～I32	X0000～X0031	数字 I/O 基板 1（CNIN）
I33～I63	X0064～X0095	数字 I/O 基板 2（CNIN）
I64～I96	X0096～X0127	未使用
I97～I104	X0128～X0135	小型 I/O 基板
I105～I160		未连接于物理输入信号
I161～I672	X1000～X1511	现场总线输入 CH1（512 点）
I673～I1184	X1512～X2023	现场总线输入 CH2（512 点）
I1185～I1696	X2024～X2535	现场总线输入 CH3（512 点）
I1697～I2048	X2536～X3047	现场总线输入 CH4（352 点）

（补充）下面地址被使用于「系统输入信号（固定输入信号）」。

X0032～X0063

输出信号

逻辑输出信号	物理输出信号	备注
O1～O32	Y0000～Y0031	数字 I/O 基板 1（CNOUT）
O33～O63	Y0064～Y0095	数字 I/O 基板 2（CNOUT）
O64～O96	Y0096～Y0127	未使用
O97～O104	Y0128～Y0135	小型 I/O 基板
O105～O160		未连接于物理输出信号
O161～O672	Y1000～Y1511	现场总线输出 CH1（512 点）
O673～O1184	Y1512～Y2023	现场总线输出 CH2（512 点）
O1185～O1696	Y2024～Y2535	现场总线输出 CH3（512 点）
O1697～O2048	Y2536～Y3047	现场总线输出 CH4（352 点）

（补充）下面地址被使用于「系统输出信号（固定输出信号）」。

Y0032～Y0063

目　　录

5 章　监视器

6 章　输入输出继电器一览

7 章　指令一览

1章　概要

本章说明软件 PLC 的概要。

1.1 软件 PLC 的概要

1.1.1 软件 PLC 的概要

PLC（可编程逻辑控制器）是指载入输入信号，根据预先编制的程序使输出回路的接点 ON/OFF，以控制各种设备的装置。

软件 PLC 将 PLC 所具备的功能融入机器人控制装置的软件当中，其编程也可以通过悬式示教作业操纵按钮台加以实施。这样，无需在外部设置 PLC，可以降低设备成本。

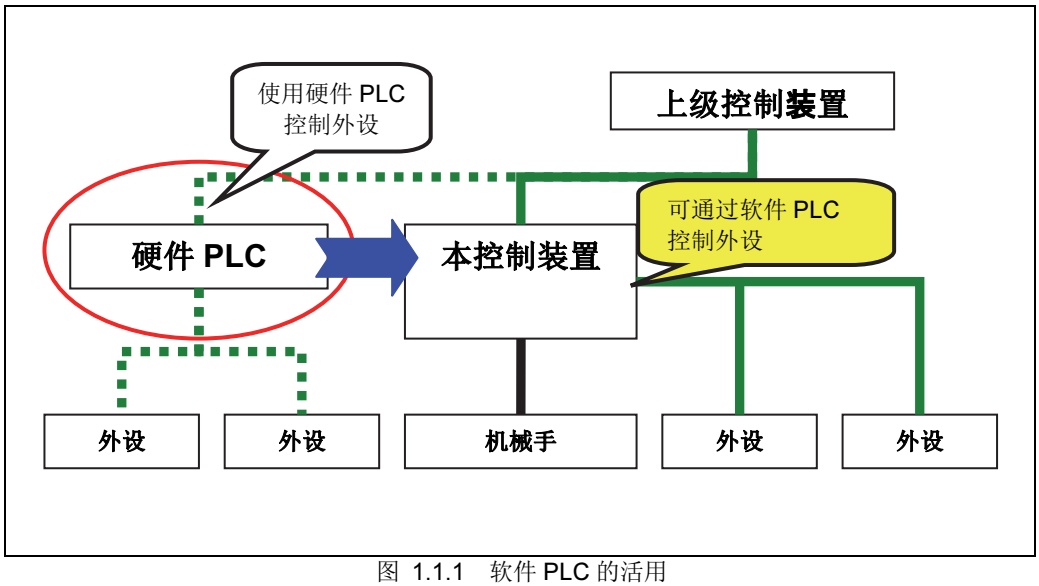


图 1.1.1 软件 PLC 的活用

如图 1.1.2 所示，软件 PLC 存在于机器人控制装置的内部和外界之间。将通过并行 I/O 或现场总线等连接的与外界之间的物理信号，通过 PLC 程序连接到逻辑信号。

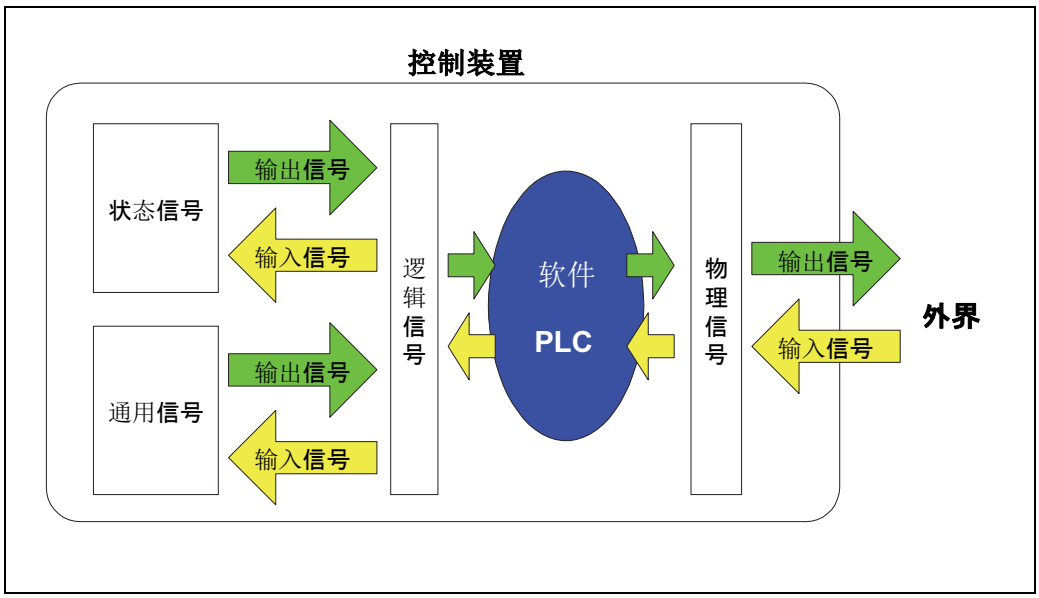


图 1.1.2 通过软件 PLC 的输入输出信号流程

重点

软件 PLC 的出厂设定

软件 PLC 的工厂出厂时的设定根据操作模式不同而异。
当前的操作模式可以在通过快捷方式代码“286”显示的系统环境画面中进行确认。详情请参见使用操作说明书“基本操作篇　1 章　前言”。

和操作模式无关，任何时候逻辑信号和物理信号均为直接相连。

操作模式 A：	**使用**软件 PLC，在**＜起动＞**状态下出厂。 通过工厂出厂时内藏的 PLC 程序“Default.stf”，将逻辑信号和物理信号直接相连。
操作模式 S：	在**不使用**软件 PLC**（分离）**的状态下出厂。 分离状态下，逻辑信号和物理信号直接相连。

PLC 程序和作业程序无关，将始终进行扫描。软件 PLC 可以通过＜常数设定＞－［1 控制环境］－［6 内藏 PLC］来切换使用 / 不使用。

1.1.2 软件 PLC 的能力

作为软件 PLC，本控制装置使用 ISaGRAF-PRO RUNTIME＜注＞。此 PLC 除了适用 LD（梯形图）语言以外，还适用国际标准规格 IEC61131-3 所规定的全部 5 种编程语言。以下是软件 PLC 的能力一览表。

＜注＞
ISaGRAF 及 ISaGRAF-PRO 是 ICS　Triplex　ISaGRAF 公司的注册商标。

表 1.1.1　软件 PLC 的能力

控制方式	循环扫描方式	
编程语言	支持 IEC61131-3 的 5 种语言 ・LD（梯形图，Ladder Diagram） ・FBD（功能块图，Function BLock Diagram） ・SFC（顺序功能图，Sequential Function Chart） ・ST（结构化语句，Structured Text） ・IL（指令表，Instruction List） ＜注＞使用（梯形图）以外的语言时，需要 AF 工作台。能够通过悬式示教作业操纵按钮台显示和编辑的语言仅为 LD（梯形图）。	
命令的种类	基本命令 12 种 应用命令 92 种	
程序容量	32K 字 （＝3.2k 字 / 文件×10 文件）	
处理时间	5～30msec	
输入输出继电器（输入输出变数）	逻辑输入	2048 点
	逻辑输出	2048 点
	物理输入	2176 点
	物理输出	2176 点
内部继电器（内部变数）	BOOL 变数	2000 点　（其中保持变数 500 点）
	实数变数	200 点　（全部为保持变数）
	整数变数	500 点　（全部为保持变数）
	短整数变数	100 点　（全部为保持变数）
	定时器变数	500 点　（全部为保持变数）
	文字变数	10 点　（全部为保持变数）
操作、编辑	运转	可以进行分离 / 停止 / 起动的指定
	监视器	可以进行通道监视器、梯形监视器、轮幅监视器等
	强制输出	可
编程工具	悬式示教作业操纵按钮台 ISaGRAF 工作台（需要使用适用 ISaGRAF-PRO Ver4.20 的产品。）	

关于各输入输出继电器和内部继电器的功能，参见 “1.2 输入输出继电器”。

＜注＞关于程序容量
上表的“字”是将（LOAD+OUT）的单纯 1 个回路按 2 字换算后的值。编辑用显示图像文件（STF 文件）单个的最大容量为 3.2K (相当于 1600 回路)，最多可以结合 10 个文件，因此计算后得到 3.2K x 10 = 32K (相当于 16000 回路)。
另外，编译后的中间文件（TIC 文件）的尺寸 64K 字节 / 文件是实际限制的上限值。（参见图 1.1.4）以上内容是为了更加浅显易懂，而换算成了显示图像文件来说明。根据制作回路内容的不同，最大回路数量也会发生变动。

＜注＞处理时间
处理时间可以在上表的范围内设定，工厂出厂时设为 30msec，一般直接使用此设定值。实际的扫描时间超过此设定时间时，会检测出扫描时间超时。

（参考）
以下简单介绍 LD（梯形图）以外的编程语言。

表 1.1.2　IEC61131-3 规定的 5 种编程语言

编程语言	概要
LD ***(Ladder Diagram)***	（梯形图） 一直广泛使用的阶梯状记述方法。最适合于 PLC 顺控程序的表述。
FBD ***(Function Block Diagram)***	（功能块图） 是结构化编程的新手法。 采用了基板的回路图似的形状。可以自由记述功能块（一项功能）并多次重复利用。
SFC ***(Sequential Function Chart)***	（顺序功能图） 是结构化编程的新手法。 通过梯形图等制作出一个一个的程序块，将之复数个相连后，由上而下类似于流程图的视觉感觉进行程序书写。
ST ***(Structured Text)***	（结构化语句） 采用了和一般软件相同的形式。 是最适合于数值算法的编程的语言。
IL ***(Instruction List)***	（指令表） 是 LD・AND・OR 等助记符形式的语言。

1.1.3 软件 PLC 的结构

说明软件 PLC 的结构根据实际操作的流程。

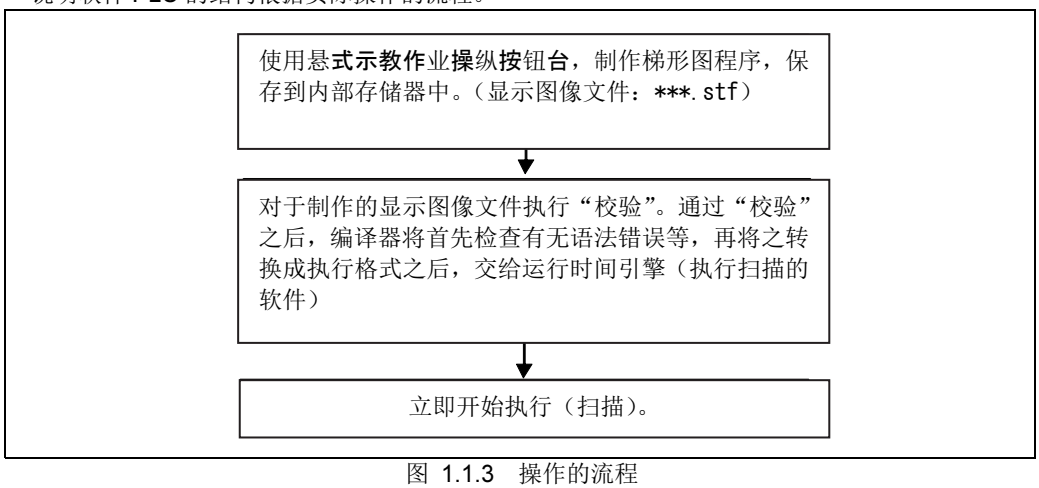


图 1.1.3 操作的流程

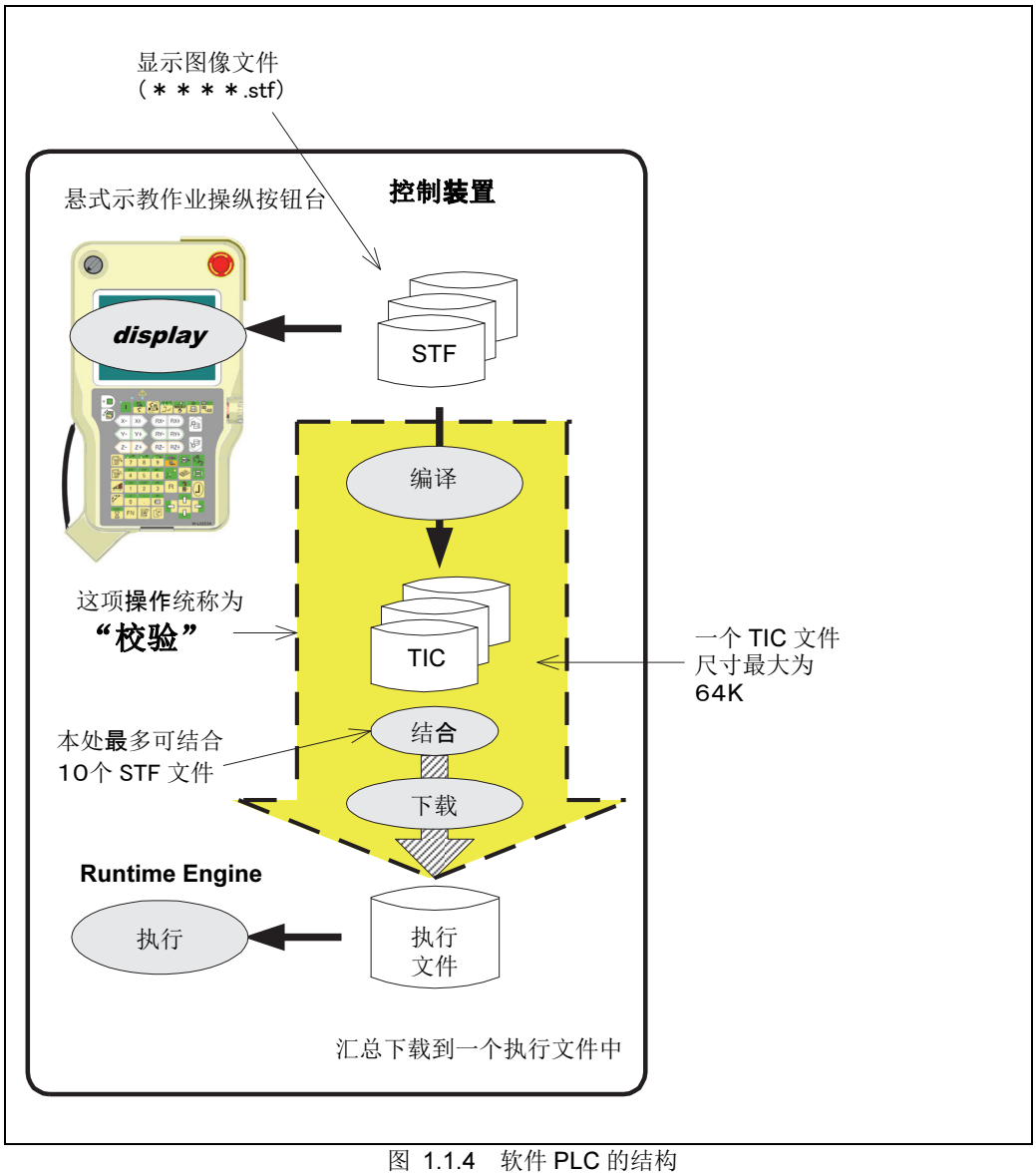


图 1.1.4 软件 PLC 的结构

重要

显示图像文件(****.stf)名称的长度将受到以下限制。限制根据系统软件版本有所差异。
V3.21 以前：16 个文字（超过 16 个文字也可以执行程序，但无法执行监视器等功能）
V3.22 以后：124 个文字

表 1.1.3　软件 PLC 操作的流程

1．使用悬式示教作业操纵按钮台，制作梯形图程序，保存到内部存储器。 （显示图像文件：＊＊＊.stf）	
(1)	内部存储器中可以保存复数个梯形图程序（显示图像文件）。 可以对应点焊 / 弧焊等不同用途，保存好不同的程序，从其中选择必要的程序加以执行。
(2)	长的梯形图程序可以分割保存到复数个程序块。编辑和维护都很简单。最多可分别为 10 个。 之后再结合后形成执行文件。 表 1.1.1
(3)	梯形图程序可以自由命名。
2．对于制作的显示图像文件执行“校验”。通过“校验”后，编译器首先检查有没有语法错误等，再将之转换成执行格式后，交给运行时间引擎（执行扫描的软件）。	
(1)	从复数个梯形图程序（显示图像文件）当中，选择必要的梯形图程序，执行“校验”。→ 图 1.1.5 使用复数个梯形图程序（显示图像文件）时，此时需要指定执行的顺序。
(2)	首先通过编译器检查语法错误等，转换成 TIC 文件（临时的执行图像文件）。存在错误时不生成 TIC 文件。
(3)	所有的梯形图程序（显示图像文件）的编译成功后，TIC 文件（临时的执行图像文件）完成结合，形成一个执行文件。
3．立即开始执行（扫描）。	
(1)	交给运行时间引擎（执行扫描的软件）的执行文件立即开始扫描。
(2)	通过服务监视器，可以监控和显示出梯形图扫描的状态或各通道的状态。

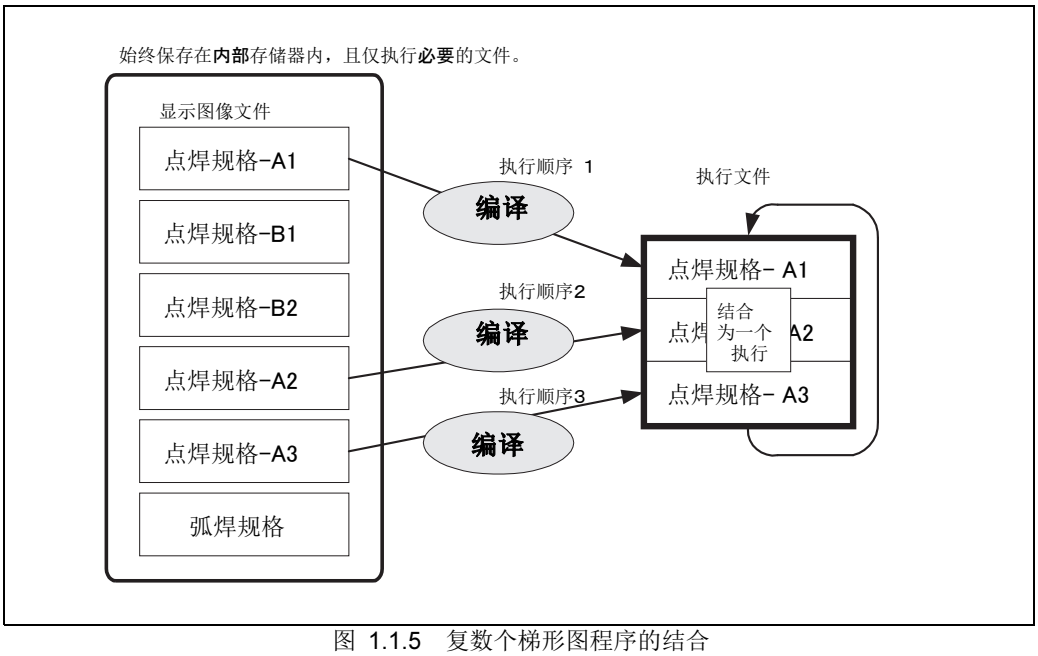


图 1.1.5　复数个梯形图程序的结合

1.2 输入输出继电器

关于输入输出继电器及内部继电器进行描述。软件 PLC 中将继电器称之为“变数”。执行 ON/OFF 的线圈接点与具备整数数据的均作同等处理。

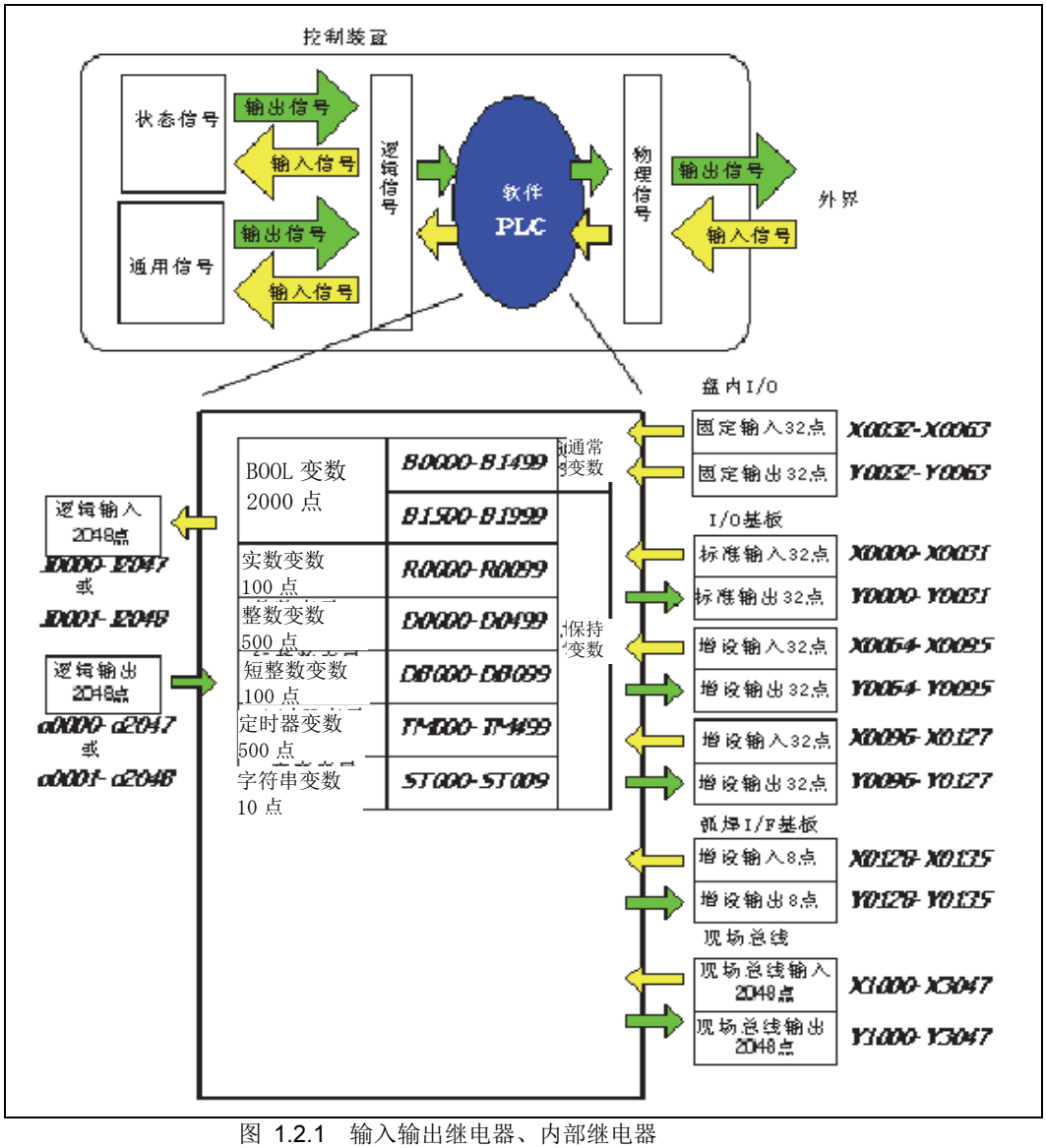


图 1.2.1 输入输出继电器、内部继电器

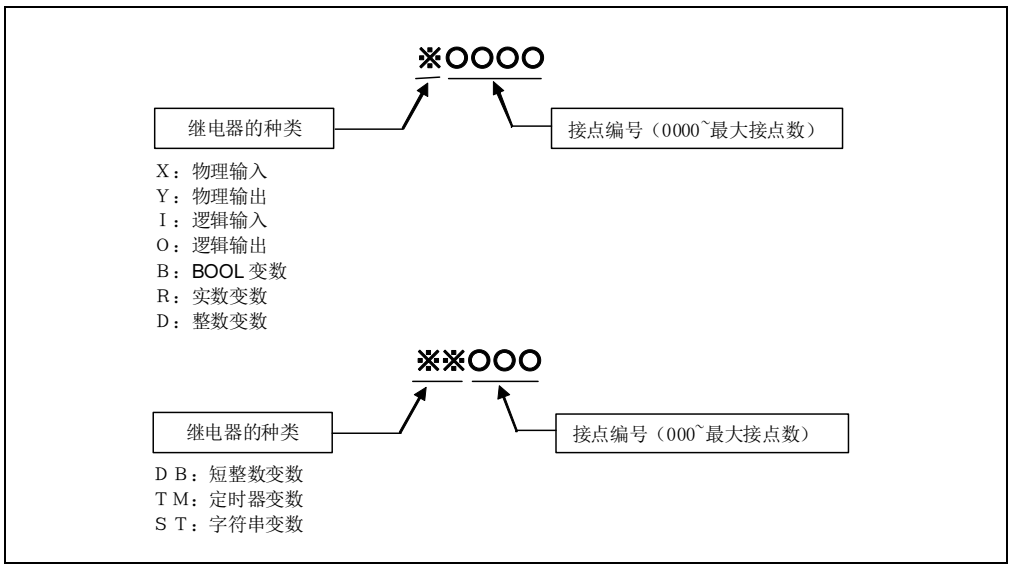


图 1.2.2　继电器编号的记述

1.2.1　输入输出继电器

表 1.2.1　输入输出继电器的种类

输入输出继电器名	点数	说明
逻辑输入	2048 点	从软件 PLC 来看，是控制装置侧的输入输出信号。 所有均为用户 I/O。
逻辑输出	2048 点	
固定输入	32 点	是物理输入输出信号。 是伺服 ON/OFF 等的控制装置内部控制所使用的输入输出信号。软件 PLC 仅可参照。
固定输出	32 点	
I/O 基板 1 输入	32 点	是物理输入输出信号。是作为选购件装备的 I/O 基板（第 1 片）的连接器 CNIN（输入）及 CNOUT（输出）的输入输出信号。
I/O 基板 1 输出	32 点	
I/O 基板 2 输入	32 点	是物理输入输出信号。是作为选购件装备的 I/O 基板（第 2 片）的连接器 CNIN（输入）及 CNOUT（输出）的输入输出信号。
I/O 基板 2 输出	32 点	
I/O 基板 3 输入	32 点	是物理输入输出信号。是作为选购件装备的 I/O 基板（第 3 片）的连接器 CNIN（输入）及 CNOUT（输出）的输入输出信号。
I/O 基板 3 输出	32 点	
电弧 I/F 板输入	8 点	是物理输入输出信号。是作为选购件装备的电弧 I/F 板的输入输出信号。
电弧 I/F 板输出	8 点	
现场总线输入	2048 点	是物理输入输出信号。 是作为选购件装备的 DeviceNet、Profibus、RIO 等的现场总线用输入输出信号。最多可安装 4 个通道的现场总线。
现场总线输出	2048 点	

重点

I/O 基板 1、2、3 为选购件。

重点

固定输入输出信号当中的实际可由软件 PLC 参照的信号仅为如下信号。（还有部分未使用的信号）请参见“6.2.1 固定输入输出”。

固定输入　X0032 ～ X0063 的 32 点

固定输出　Y0032 ～ Y0047 的 16 点

1.2.2 内部继电器

表 1.2.2 内部继电器的种类

内部继电器名	点数	说明
BOOL 变数	2000 点	是仅具备 ON 或 OFF（TRUE 或 FALSE）的状态的变数。相当于以往的内部辅助继电器。
实数变数	100 点	是具备实数(浮动小数点)型的连续值的变数。 （单精度 float）
整数变数	500 点	是具备整数型的连续值的变数。 （-2147483648～2147483647）
短整数变数	100 点	是具备短整数型的连续值的变数。 （-128～+127）
定时器变数	500 点	是具备定时器型的连续值的变数。 （0～23h59m59s999ms）
字符串变数	10 点	是字符串（ASCII 代码）。 最大为半角 2 5 5 个文字。
通常变数		是切断电源后会被复位的变数。 BOOL 变数当中的一部分（B0000～B1499）为通常变数。
保持变数		是即使切断电源也可以保持内容的变数。 上述的 B0000～B1499 以外全部为保持变数。

1.2.3 现场总线输入输出

关于现场总线输入输出请参见“6 章 输入输出继电器一览”及选配件操作说明书“DeviceNet 功能”。

1.3 使用工作台时

使用 ISaGRAF 工作台的话，可以离线进行编程（不使用机器人控制装置）。将工作台和机器人控制装置通过以太网进行连接，通过工作台进行编程，然后将程序下载到机器人控制装置加以执行。修正作业或监控显示也通过工作台进行操作。

通过工作台编制的程序不能直接通过机器人悬式示教作业操纵按钮台进行显示及编辑。这是因为工作台和机器人控制装置对于 I/O 继电器、内部继电器的称呼是不相同的。要在显示图像文件（STF 文件）层面实现完全互换，需要在工作台侧预先登录和机器人控制装置相同的 I/O 继电器、内部继电器的定义数据库（下图的 DB）。

另外，在工作台使用 LD（梯形图）以外的语言时，即使采取上述措施，程序也不能通过机器人悬式示教作业操纵按钮台进行显示。

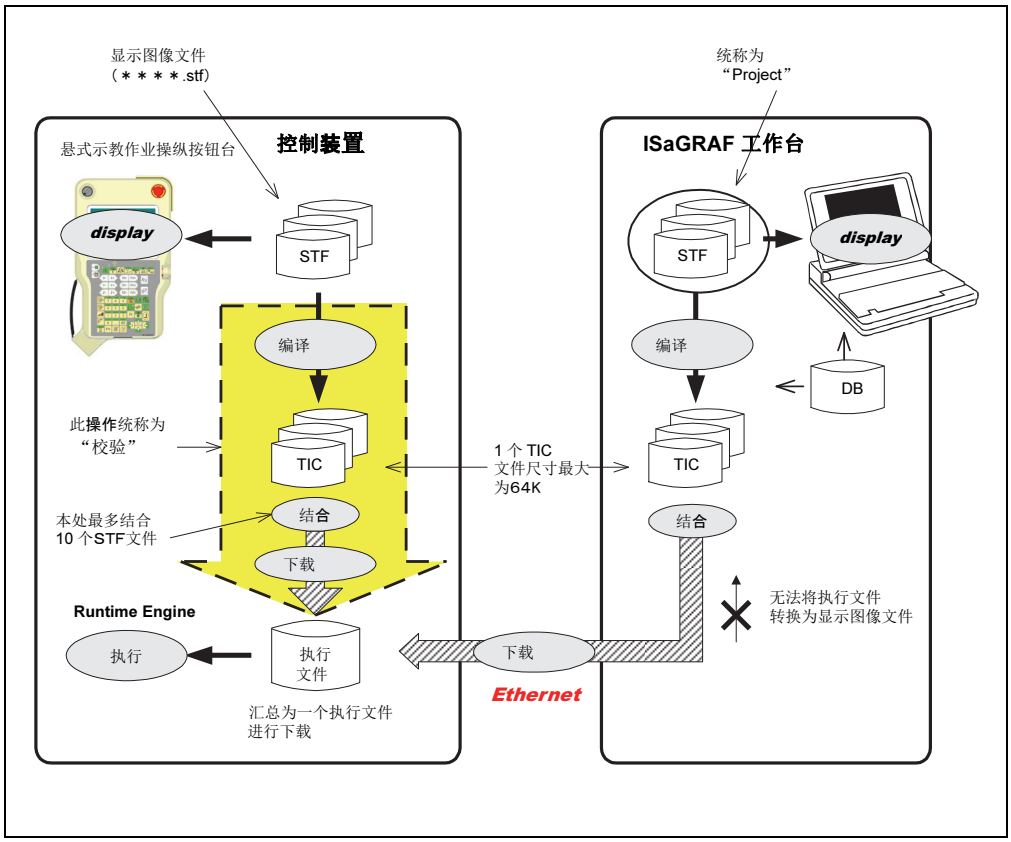


图 1.3.1　工作台

需要另行从 ICS　Triplex　ISaGRAF 公司购买 ISaGRAF 工作台。

2章 编程

本章说明使用梯形图编辑器编制 PLC 程序的方法。

2.1 编程

2.1.1 起动梯形图编辑器

新编制 PLC 程序或编辑已经示教的程序时，使用梯形图编辑器。可以在悬式示教作业操纵按钮台上显示梯形图（LD 语言）并直接编辑。示教 / 再生模式两者均可。

PLC 程序能够以显示图像文件的格式在存储器上记录任意数量。

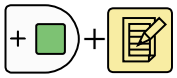
1 **选择＜维修＞－［14 PLC 程序编辑］，从显示的菜单当中选择［1 PLC 程序编辑］。**

≫ 显示出如下所示的梯形图程序（显示图像文件）的一览。

根据工厂出厂时的操作模式，可能会嵌入 Default.stf。详情参见“1 章 概要”。

即使已经处于梯形图程序的扫描执行过程中，也可以执行新的梯形图程序的编辑。

这里一览显示的程序是存储器中记录的显示图像文件，并非运行时间引擎实际扫描的文件。



2 **新编制程序时，首先要决定梯形图程序的文件名称。**

将光标对齐文件名称栏，按下［动作可能］＋［编辑］。

≫显示软键盘画面，请登录文件名称。

文件名使用英文数字。

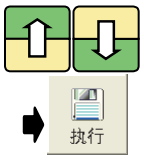
名称输入完成后，按下 f12＜写入＞。 →至 4

重要

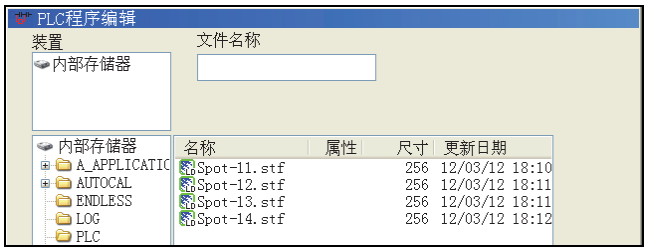
梯形程序的文件名称长度将受到以下限制。限制根据系统软件版本有所差异。

V3.21 以前：16 个文字（超过 16 个文字也可以编制或执行程序，但无法执行监视器功能）

V3.22 以后：无法设定 124 个文字以上的文件名称。



3 **已经记录有梯形图程序时，对于编辑的文件通过［上下］进行选择，再按下 f12＜执行＞。**



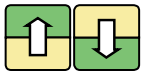
梯形图程序（显示图像文件）的扩展名为“**stf**”。显示已经保存的所有 **stf** 文件一览。

4 **确定编辑文件后，起动梯形图编辑器，显示如下所示梯形图编辑画面。**

选择已经保存的梯形图程序（显示图像文件）时，显示在梯形图程序的前头。

梯形图编辑始终为“插入”状态。不能覆写。

通过



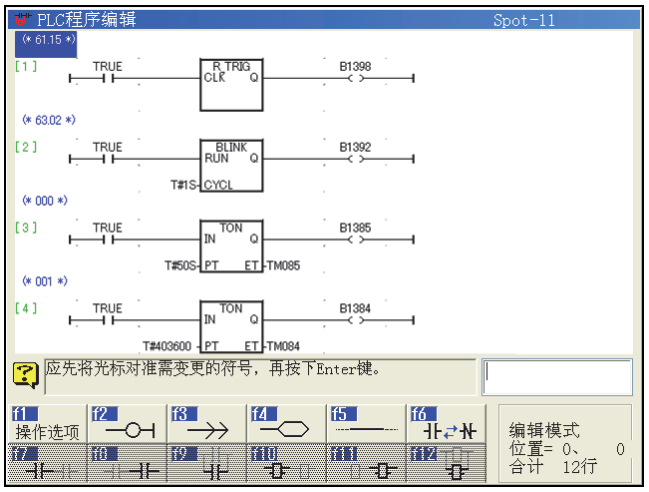
［上下］

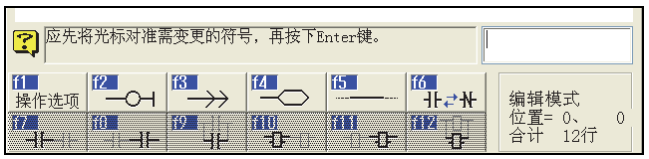
或

［缓动旋钮］

可以上下滚动。

f 键有 2 个画面，可通过 f1＜操作选项＞进行切换，选择相应的菜单编辑梯形图。





梯形图编辑区域

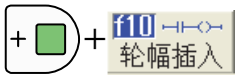
功能键菜单

引导信息
数据输入区域

图 2.1.1　梯形图程序编辑画面

2.1.2 输入新的轮幅

轮幅是指闭合回路。按如下所示输入新的轮幅。



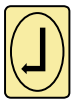
1 通过 f1＜操作选项＞切换 f 键，在空白区域按下［动作可能］＋f10＜轮幅插入＞。

≫插入新的闭合回路（1 轮幅），显示如下。

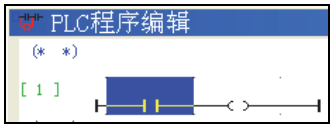
按下［动作可能］＋［Enter］也可同样插入闭合回路（1 轮幅）。

蓝色背景的部分为光标位置（编辑对象的区域）。

（＊ ＊）中显示轮幅的注释（以闭合回路为单位记述的注释），[ ] 内显示轮幅编号（从 1 开始的闭合回路序号）。



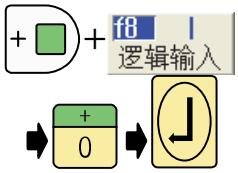
2 为了输入接点编号，将光标对齐接点，按下［Enter］。



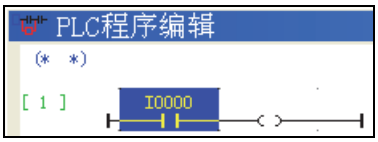
≫f 键切换为如下所示参数选择菜单。

第 1 页	
功能名称	输入示例、说明
F1＜变数转换＞	切换参数候补。
F2＜TRUE＞	常数 TRUE(常时 ON)的插入
F3＜FALSE＞	常数 FALSE(常时 OFF)的插入
F4＜短整数变数＞	例：DB123 通过继电器编号＋“．”＋位编号 0～7 指定位值。
F5＜整数变数＞	例：D1234 通过继电器编号＋“．”＋位编号 0～31 指定位值。
F6	
F7	
F8＜逻辑输入＞	例：I1234
F9＜逻辑输出＞	例：O1234
F10＜物理输入＞	例：X1234
F11＜物理输出＞	例：Y1234
F12＜BOOL 变数＞	例：R1234

第 2 页	
功能名称	输入示例、说明
F1＜变数转换＞	切换参数候补。
F8＜文字变数＞	例：ST123
F9＜定时器变数＞	例：TM123
F10＜实数变数＞	例：R1234

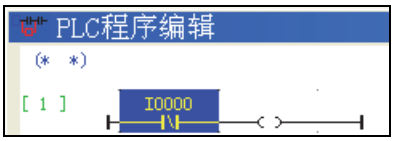


3 **输入逻辑输入 I0000 时，按下［动作可能］＋f8＜逻辑输入＞，按下［0］[Enter]。**
≫如下图所示，在接点记号的上方显示为 I0000。

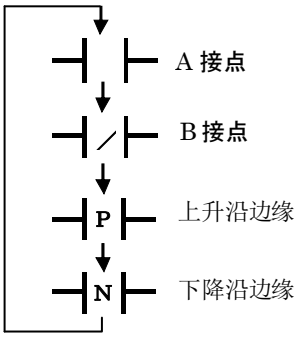


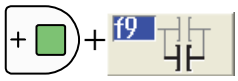
按下 f 键返回原先的菜单配置。

4 **尝试变更接点的种类。按一次 f6＜种类切换＞。**
≫如下图所示，变成 B 接点。

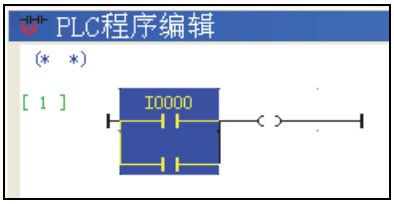


每次按下 f6＜种类切换＞，接点种类的变化分别如下所示。



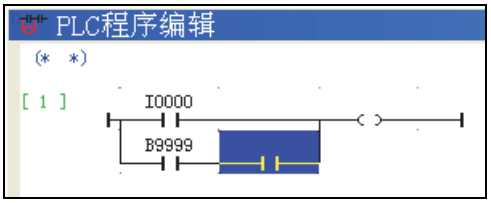


5 **尝试插入 OR 回路。按下［动作可能］＋f9＜OR 的插入＞。**
≫插入的符号如下图所示。

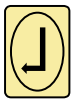


按照和 2～3 相同的步骤输入插入的接点编号。

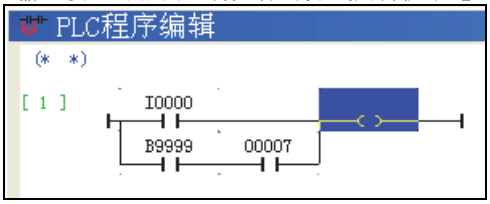
6 **尝试在其右侧插入继电器接点。**
按下［动作可能］＋f8＜在光标右侧插入继电器＞。
≫插入的符号如下图所示。



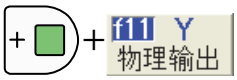
按照和 2～3 相同的步骤输入插入的接点编号。



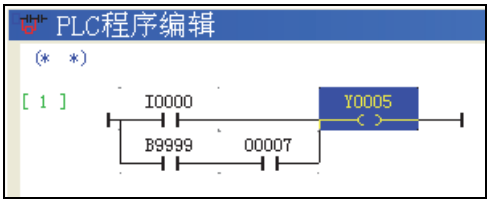
7 **接着，输入线圈编号。**
和输入接点时相同，将光标对齐线圈再按下［Enter］。

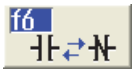


≫f 键切换为如下所示的参数选择菜单。

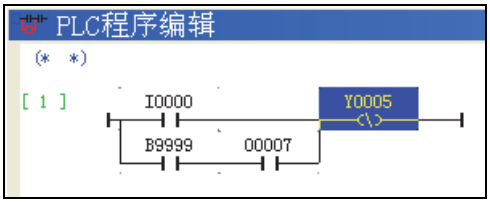


8 **按照和接点时相同的要领输入线圈编号。（物理输出 Y0005 的示例）**

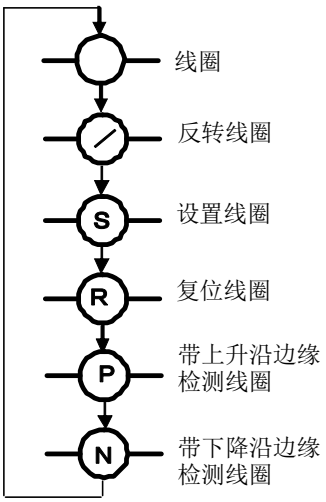




9 **尝试变更线圈的种类。按下一次 f6＜种类切换＞。**
≫如下图所示，变成反转线圈。

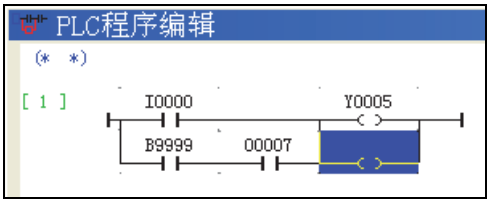


每次按下 f6＜种类切换＞，线圈的种类作如下变化。



10 **输出复数个线圈时，将光标对齐线圈按下 f2＜线圈＞。**

≫以后同样输入线圈编号。

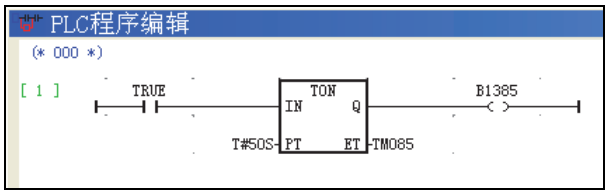


2.1.3 输入 LD 块

LD 块是指类似于比较命令或定时器命令等应用命令的总称。通过编号或名称输入 LD 块。

本控制装置的软件 PLC 中，LD 块全部使用“功能块”。“功能块”以长方形（块）的形式表述，是具备输入和输出的函数。和以往 PLC 不同的是没有作为单纯线圈处理。

以“定时器命令”为例进行说明。例如“定时器命令”可按 LD 块 No.24＝TON（上升沿延迟时间）编程如下。

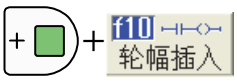


中央方框写有“TON”的部分是具备定时器功能的 LD 块（功能块）。通过连接到“IN”的 BOOL 型变数的上升沿检测，开始定时器值的增量计时，到时限后，BOOL 型的输出“Q”从 FALSE 转为 TRUE。将其状态在其他闭合回路进行参照时，使用连接到输出“Q”的 B1385 的接点。到时限的设定是通过连接到“PT”的定时器型常数进行指定。该情况下为 50 秒。另外，连接到“ET”输出的定时器变数可以观察当前的经过定时器值。

如上所述，功能块的特点是可以存在于闭合回路中的任何位置，并存在输出。

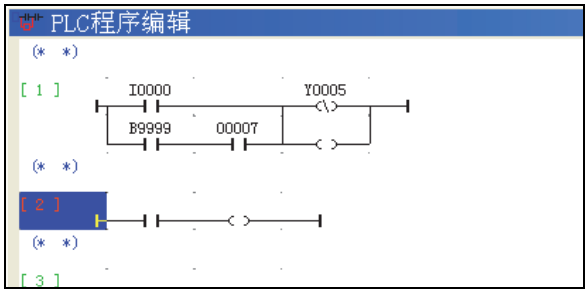
LD 块详情参见“7 章 指令一览”。

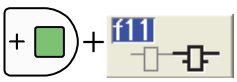
本节说明 LD 块的输入方法。



1 通过［动作可能］＋f10＜轮幅插入＞，追加新的轮幅。

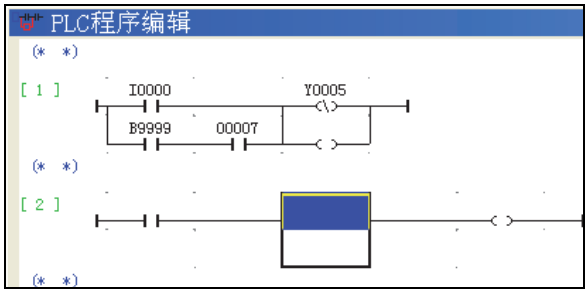
按下［动作可能］＋［Enter］可插入同样的闭合回路（1 轮幅）。

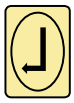




2 将光标对齐接点，按下［动作可能］＋f11＜在右方插入 LD 块＞。

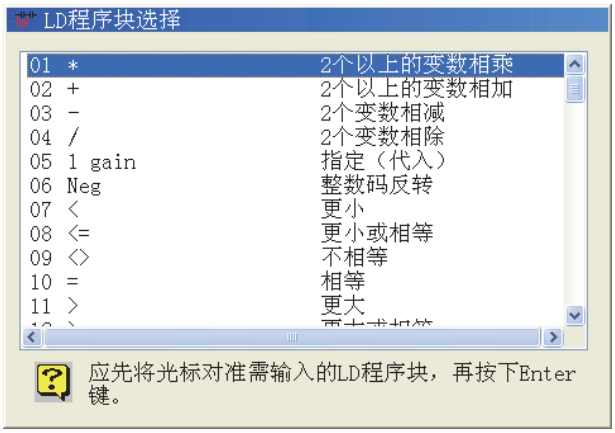
≫如下图所示，在接点和线圈之间插入一个 LD 块。



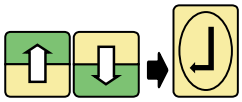


3 **将光标对齐 LD 块，按下［Enter］。**

≫显示下图所示的 LD 块一览。

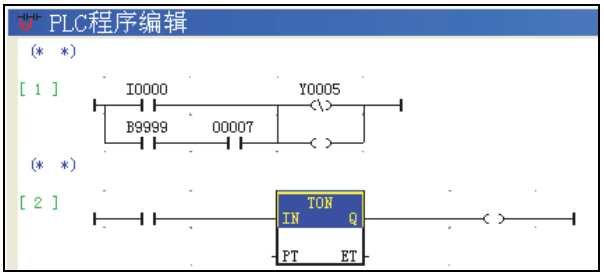


LD 块按照编号由小到大的顺序显示。

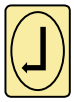


4 **将光标对齐输入的 LD 块，按下［Enter］。**
或直接输入左端显示的 LD 块编号。
例如，选择 No.24＝TON（上升沿延迟时间）。

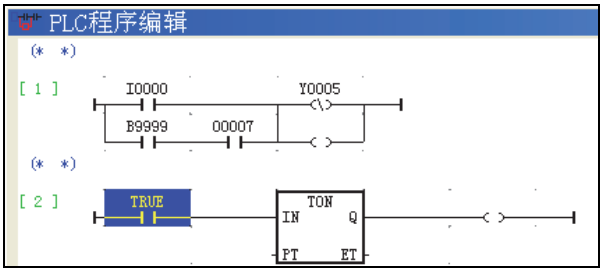
≫LD 块的显示为 TON。



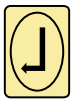
可以看出 LD 块“TON”需要输入 2 点（IN 和 PT）、输出 2 点（Q 和 ET）。这些均需要正确输入。



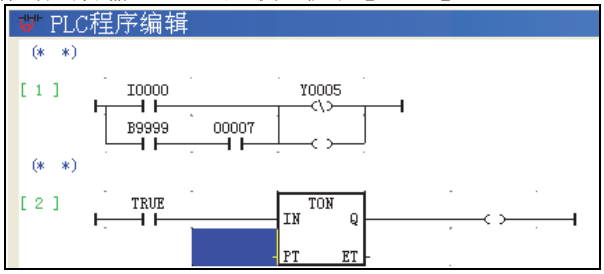
5 **将光标对齐连接到输入“IN”的接点，按下［Enter］后，输入所要的接点。**

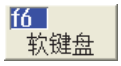


此例中输入的是 TRUE（表示常时 ON 的接点）。LD 块“TON”将通过输入 IN 的上升沿检测，开始定时器值的增量计时，因此属于从电源接通时累加计时的定时器。

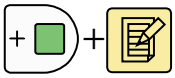


6 **接着，输入到时限值 500msec。**
将光标对齐输入“PT”的左侧，按下［Enter］。

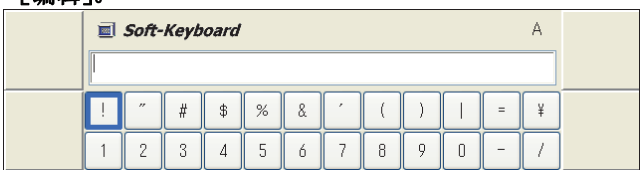




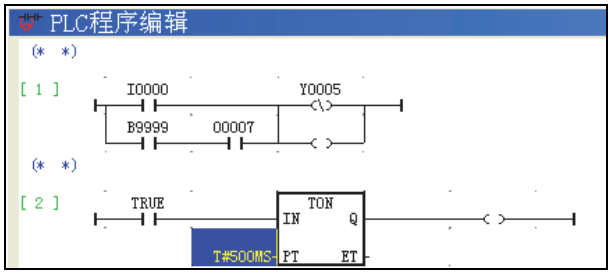
或



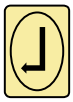
7 **转为参数输入画面，按下 f6＜软键盘＞，显示文字输入画面。也可以按下［动作可能］＋［编辑］。**



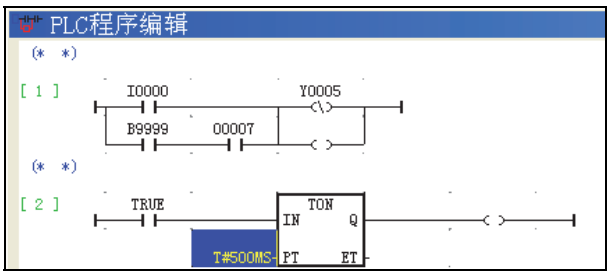
8 **时限值的指定使用以“T#”开头、以“S”或“MS”结尾的定时器型常数。要设定为 500msec 时，在文字输入画面输入 T#500MS，**
按下 f12＜确定＞。
≫如下图所示显示时限值。



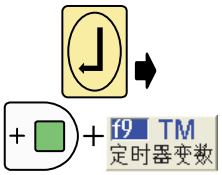
定时器型常数可以按照秒和毫秒这 2 种单位进行指定。
500msec 时・・T#500MS
2sec 时・・・・・T#2S



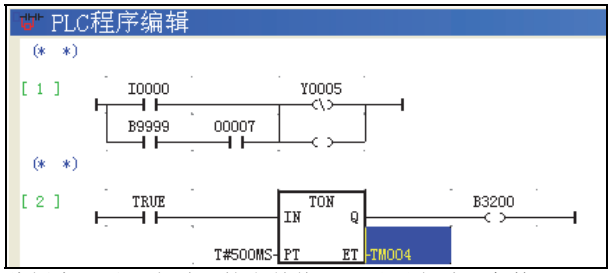
9 **接着指定输出。**
将光标对齐连接到输出“Q”的线圈，按下［Enter］，输入所要的变数（TON 时为 BOOL 变数）。



此例中，到达时限后，**BOOL** 变数 B3200 从 FALSE 变成 TRUE。



10 **最后，指定定时器值监视器。**
将光标对齐输出“ET”的右侧，按下［Enter］，从参数候补中选择［动作可能］＋f9＜定时器变数＞，输入定时器变数的编号。

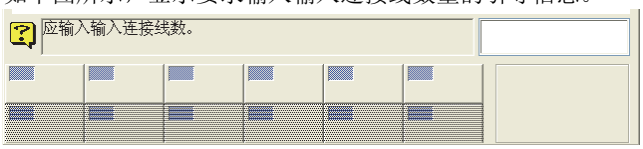


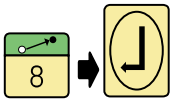
此例中，对于定时器的当前值，可以用定时器变数 TM004 观察。
即使不使用时，也必须指定。

11 **至此完成 LD 块“TON”的输入。**

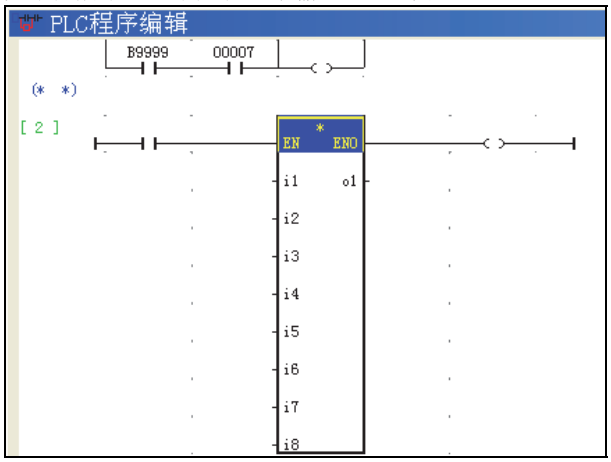
使用输入输出数量可变的 LD 块时，如下所述。

12 **选择 LD 块 01“2 个以上的变数相乘”。**
≫如下图所示，显示要求输入输入连接线数量的引导信息。





13 **输入 2～20 之间的输入连接线数量，按下［Enter］。**
这里指定为 8 个。
≫如下图所示，显示具备 8 个输入的 LD 块 01。

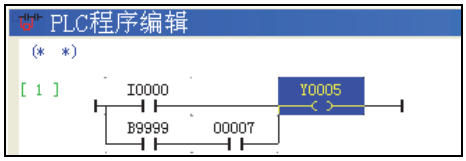


14 **之后按同样方法指定其他输入输出。**

2.1.4 输入返回命令

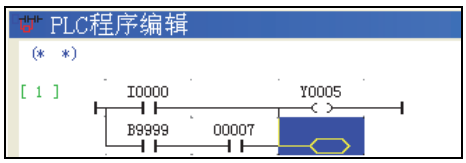
返回命令是指不执行之后的回路、返回程序前头的命令。执行返回命令（其逻辑为 TRUE）后，不执行之后的回路，而返回程序的前头，从前头开始执行。

1 **将光标对齐线圈的符号。**



2 **按下 f4＜返回＞。**

≫如下图所示，追加了返回命令。



此例中，Y0005 为 TRUE 时，执行返回操作，不执行之后的轮幅。Y0005 成为 FALSE 时，转为执行之后的轮幅。

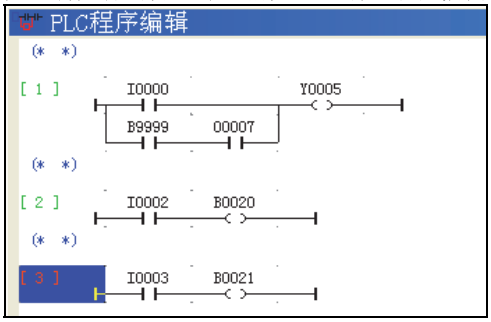
2.1.5 输入转移命令和标签

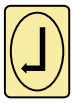
通过转移命令和标签的组合，可以将程序转移到执行所需轮幅。执行转移命令（其逻辑为 TRUE）后，不执行之后的回路，转移到该处记述的标签位置，从标签处开始执行。

整个程序中最多可以输入 1000 个标签。（不能输入中文。可输入英文或数字 40 字符）

1 **首先输入标签。**

首先，将光标对齐回路的前头（左端显示轮幅编号的部位）。

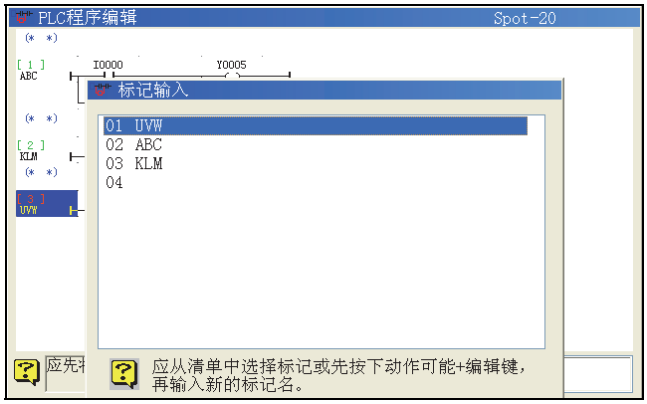


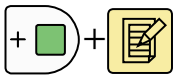


2 **然后按下［Enter］。**

≫如下图所示，显示已经登录的标签名称一览。

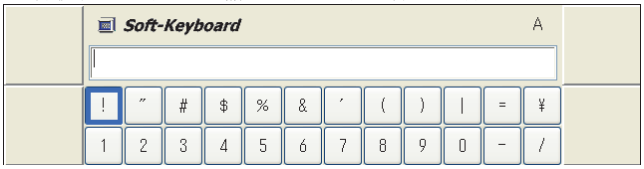
（未登录任何标签时为空栏。）



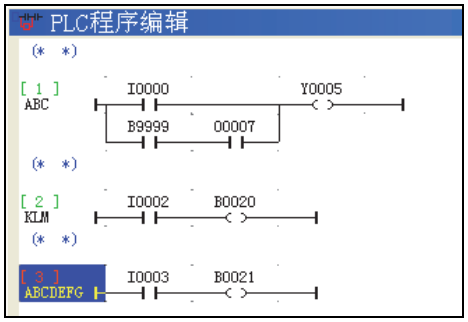


3 **登录新标签时，按下［动作可能］＋［编辑］。**

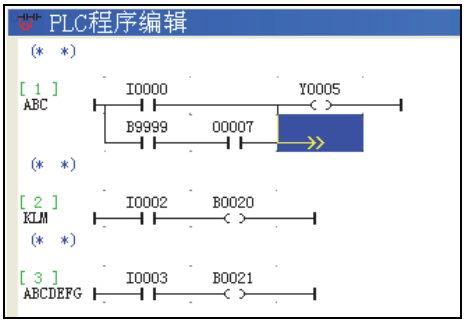
≫启动软键盘，在此处输入想要登录的标签名称。

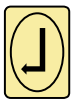


4 **标签名称输入完毕后，关闭软键盘，如下图所示，在左端显示登录的标签名称。（例；“ABCDEFG”）**

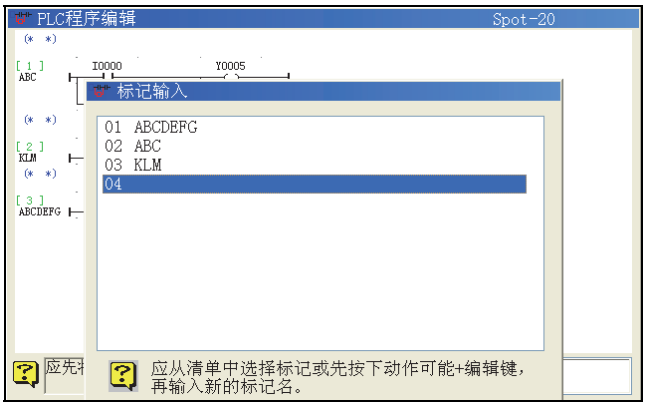


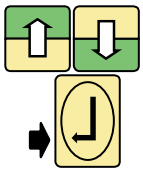
5 **接着，输入转移命令。**
将光标对齐线圈的符号，按下 f3＜转移＞。
≫如下图所示，在 OR 回路输入转移符号。



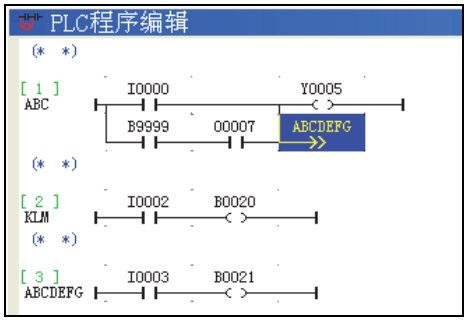


6 **将光标对齐转移命令，按下［Enter］。**
≫显示下图所示已登录的标签一览表。





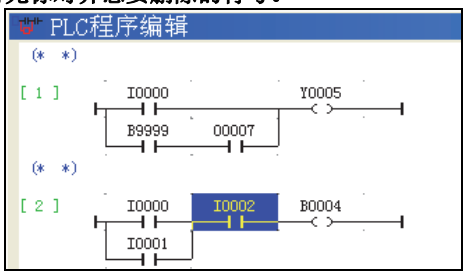
7 **通过 ［上下］键选择转移对象的标签，按下［Enter］。**
≫如下图所示，输入转移对象的标签名称。（例：“ABCDEFG”）



此例中，Y0005 为 TRUE 时执行转移，不执行轮幅编号［2］。从标签 ABCDEFG 即轮幅编号［3］开始再次执行。

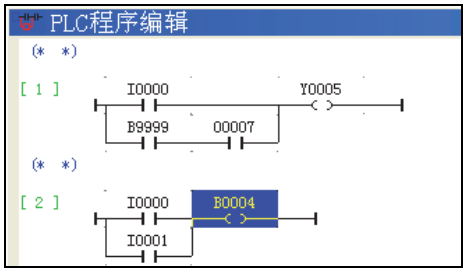
2.1.6 删除符号

1 **将光标对齐想要删除的符号。**

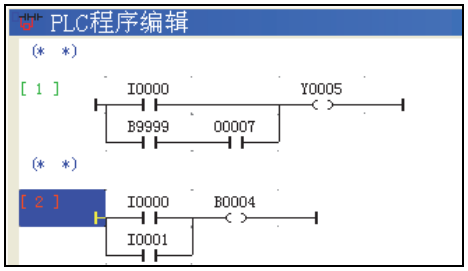


2 **按下［删除］。**

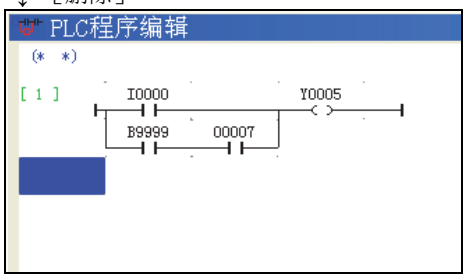
≫光标位置处的符号被删除，其右侧符号向左顺移显示。



3 **将光标对齐左端的轮幅编号栏（绿色方括号包围的数字），按下［删除］后，可以删除各个轮幅。**



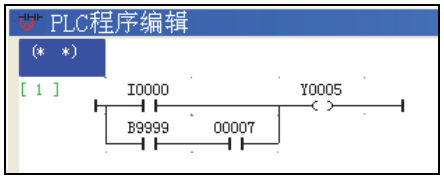
↓［删除］

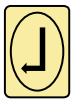


2.1.7 输入轮幅注释

对于所有的轮幅（闭合回路）可以按各个轮幅分别输入注释。
轮幅注释最多可以输入半角英文数字 200 个字符（相当于全角 100 个字符）。

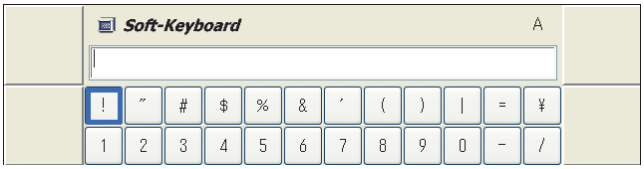
1 **将光标对齐左端的轮幅注释栏（蓝色的（＊　＊）部分）。**



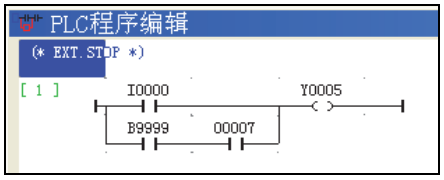


2 **然后按下［Enter］。**

≫启动软键盘，在这里输入注释。



3 **注释输入完成后，关闭软键盘，显示如下所示的轮幅注释。（例：输入为“EXT。STOP”）**

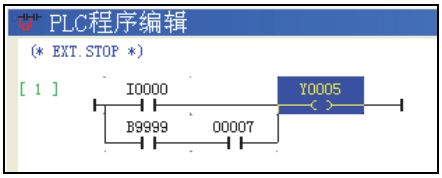


2.1.8 输入和显示别名

别名是指给各个符号赋予的注释。
预先设定接点的用途和功能名称，可便于查看梯形图程序。
别名可输入半角英文数字最多达 8 字符（相当于全角 4 个字符）。
通过 f 键的单触操作，可以切换通常显示（变数显示）和别名显示。

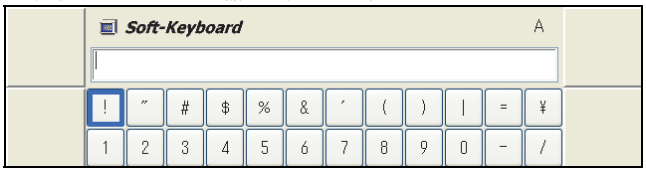
别名以梯形图程序（显示图像文件）单位进行管理。在复数个梯形图程序中使用相同的别名时，制作一个登录别名的梯形图程序，然后按各文件执行复制，并开始新的编辑。

1 **将光标对齐想要设定别名的符号。**



f6 别名编辑

2 **然后按下 f6＜别名编辑＞。不显示 f 键时，通过 f1＜操作选项＞切换 f 键群后可以显示。**
≫启动软键盘，在这里输入别名（注释）。



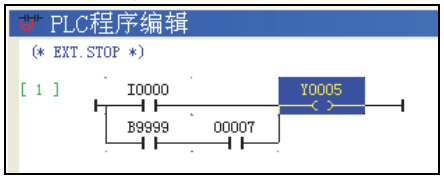
如果已经设定了别名时，会显示所设定的别名。在这里可以进行修正。

3 **完成注释输入后，关闭软键盘，返回梯形图显示画面。**

f5 别名显示

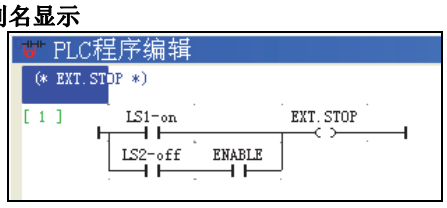
4 **按下 f6＜别名显示＞后，会交互切换别名显示和通常显示。未设定别名的符号会显示和通常显示相同的变数名称。**

通常显示（变数名称显示）



↑↓ f5＜别名显示＞

别名显示

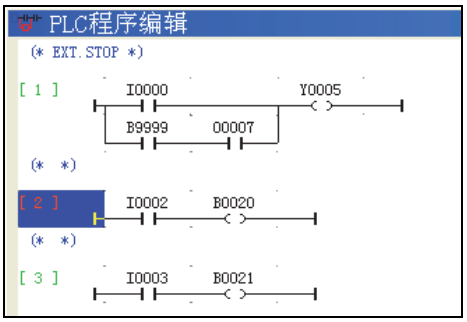


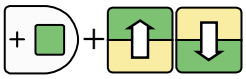
由于画面空间的因素，别名只能显示从前头起的最多 7 个字符内容。

2.1.9 剪切＆粘贴

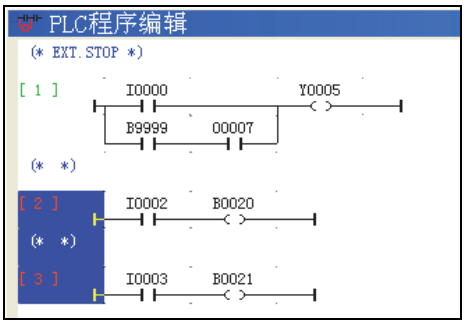
可以按复数轮幅（闭合回路）单位执行删除（剪切）、复制（存储到粘贴缓存中进行复制）。是方便编辑的功能。

1 **将光标对齐轮幅（闭合回路）的前头（左端）。**

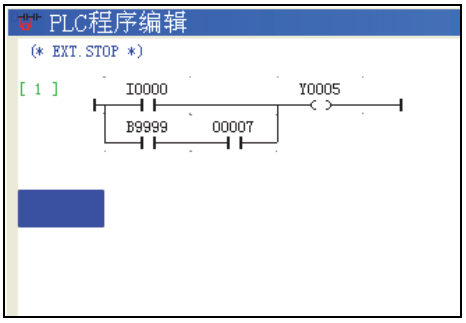




2 **按下［动作可能］＋［上下］后，选择复数个轮幅。**

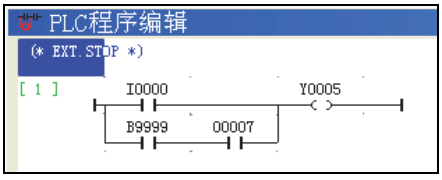


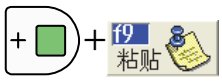
3 **按下［动作可能］＋f7＜剪切＞的话，选中的复数个轮幅被删除。**



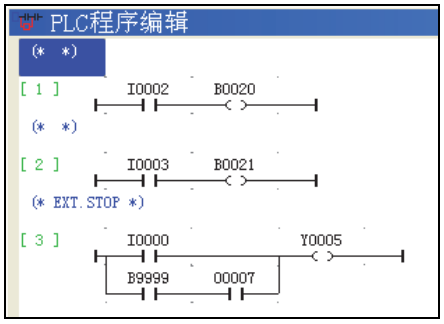
此时删除的复数个轮幅会保存在粘贴缓存中。

4 **将光标移动到想要粘贴的部位。**
光标对齐轮幅（闭合回路）的前头（左端）。





5 **按下［动作可能］＋f9＜粘贴＞的话，存储在粘贴缓存中的复数轮幅将会插入在光标位置的上方。**

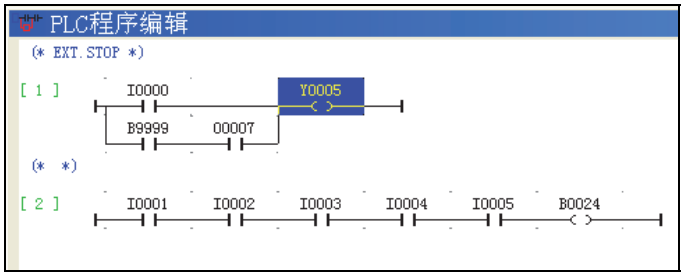


6 **前述步骤中不按下［动作可能］＋f7＜剪切＞，而是按下［动作可能］＋f8＜拷贝＞，不进行删除（剪切）便可将内容存储到粘贴缓存。**
然后同样可以按［动作可能］＋f9＜粘贴＞进行插入。

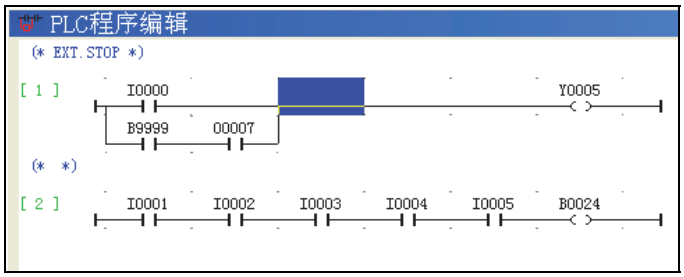
2.1.10 使梯形图易于查看

本控制装置的软件 PLC 中的线圈（右端）显示位置，根据接点数量的不同自由变化。为了对齐线圈（右端）的显示位置，可以追加“横棒”。
“横棒”和程序的执行时间无关。

1 **将光标对齐想要插入“横棒”的位置。**



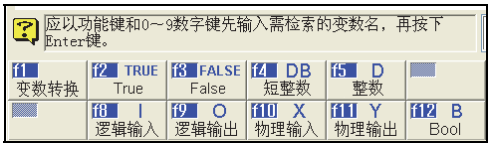
2 **按下 f5＜横棒＞。**
≫按下 1 次，则追加一条“横棒”。此例中，按下 3 次，线圈的位置在纵方向整齐排列，变得易于查看。

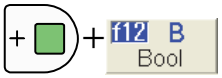


2.1.11 检索

1 **在任意的光标位置处，按下 f2＜检索＞。**

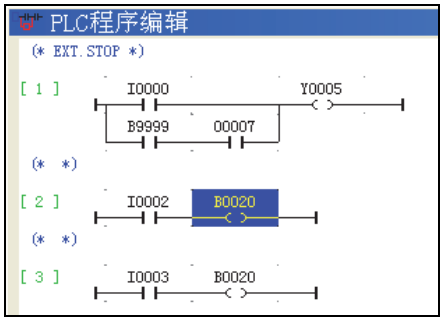
≫f 键显示如下所示。



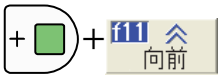


2 **如果要检索 BOOL 变数 B0020，按下［动作可能］＋f12＜BOOL＞，再按下［0］［0］［2］［0］［Enter］。**

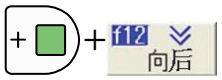
≫从光标位置处向后开始检索，光标移动到发现 B0020 的部位。



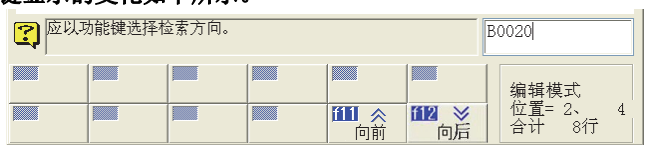
按下［动作可能］＋f12＜BOOL＞，指定［2］［0］时无法执行检索。检索输入时不能省略［0］。



或



3 **f 键显示的变化如下所示。**



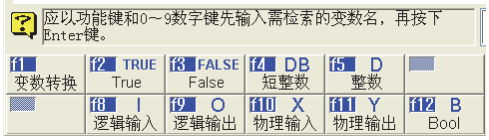
按下［动作可能］＋f11＜向前＞之后，
继续向前检索，光标移动到下一个 B0020。
按下［动作可能］＋f12＜向后＞之后，
继续向后检索，光标移动到下一个 B00200。

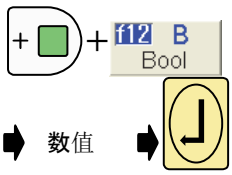
4 **结束检索时，按下［复位 / R］。**

2.1.12 置换

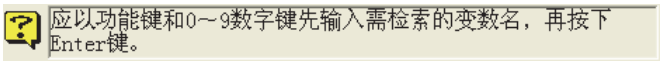
1 **在任意的光标位置处按下 f3＜置换＞。**

≫f 键的显示如下所示。

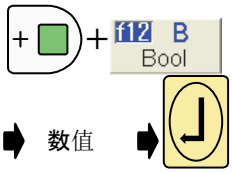




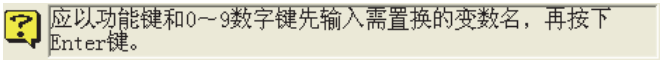
2 **指定置换的源变数。**



转换 BOOL 变数 B0020 时，按下［动作可能］＋f12＜BOOL＞，再按下［0］［0］［2］［0］［Enter］。

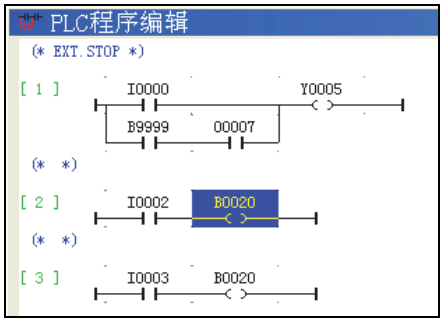


3 **接着指定置换后的变数。**

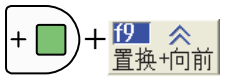


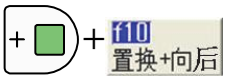
要转换成 BOOL 变数 B0021 时，按下［动作可能］＋f12＜BOOL＞，再按下［0］［0］［2］［1］［Enter］。

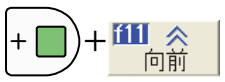
≫从光标位置处向后进行检索，光标移动到发现置换源 B0020 的部位。

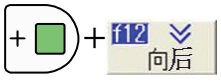


按下［动作可能］＋f12＜BOOL＞并指定［2］［1］时无法检索。检索输入时不能省略［0］。

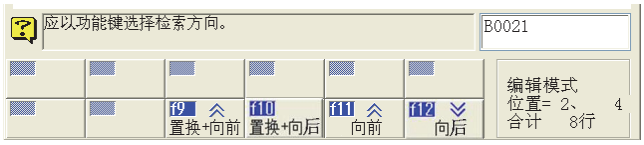








4 **f 键显示的变化如下所示。**



按下［动作可能］＋f9＜置换+向前＞之后，
上述符号从 B0020 置换成 B0021，接着向前检索，光标移动到下一个 B0020。
按下［动作可能］＋f10＜置换+向后＞之后，
上述符号从 B0020 置换成 B0021，接着向后检索，光标移动到下一个 B0020。
按下［动作可能］＋f11＜向前＞之后，
不进行置换，继续向前检索，光标移动到下一个 B0020。
按下［动作可能］＋f12＜向后＞之后，
不进行置换，继续向后检索，光标移动到下一个。

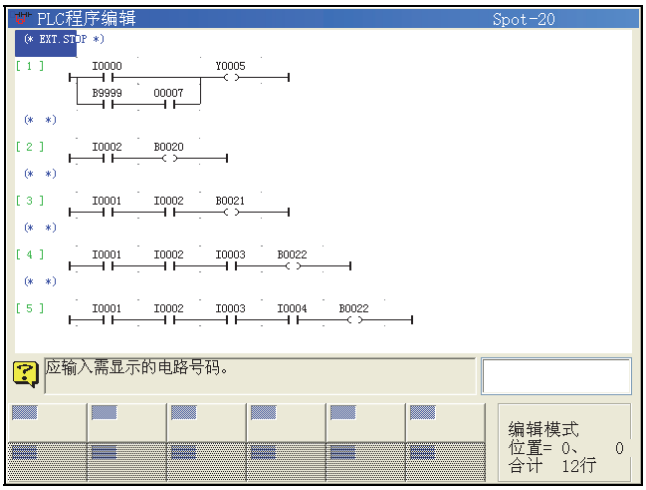
5 **结束检索、置换时，按下［复位 / R］。**

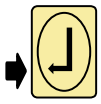
2.1.13 按轮幅编号转移

可以指定想要显示的轮幅编号（闭合回路的编号），完成一次性转移。
编辑较长的梯形图程序时非常方便。

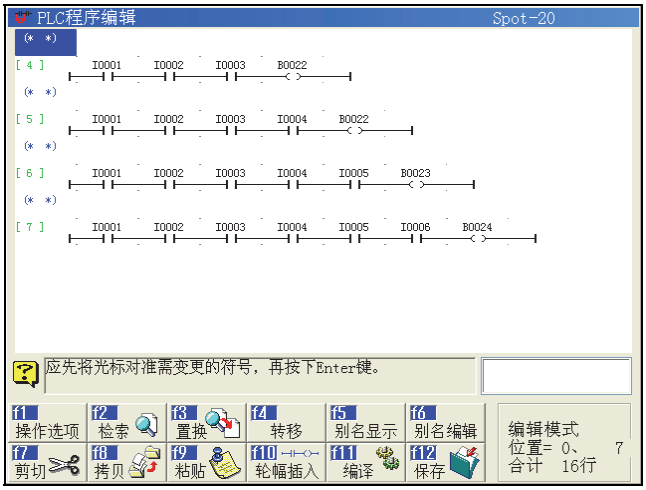
f4 转移

1 在任意的光标位置处按下 f4＜转移＞。



数值 

2 输入想要显示的轮幅编号（闭合回路的编号），按下［Enter］。
轮幅编号（闭合回路的编号）是指左端显示的绿色的［ ］包围的数字。
≫ 指定的轮幅（闭合回路）显示在画面的前头。

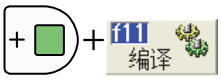


2.2 校验有无语法错误

编制的 PLC 程序记述显示图像，不能直接执行。执行时需要将显示图像文件转换成可以执行的形式。称之为“编译”。编译器（执行编译的软件）将会检验显示图像文件是否有语法错误并可执行，通过校验后，才转换成可执行的形式。

本章说明这在打开梯形图编辑器的状态下，校验能否编译及确定错误部位的方法。

也可以从梯形图编辑器，按照和“3.2 从编译到下载”相同的操作，执行所编制的 PLC 程序的编译及下载操作。



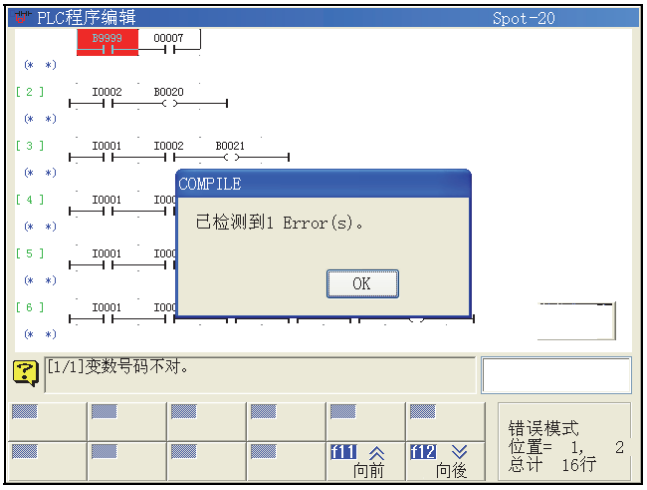
1 **按下［动作可能］＋f11＜编译＞。**

≫对当前编辑中的梯形图程序，开始编译，稍后显示如下结果。

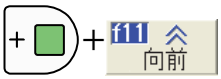
≫没有编译错误时

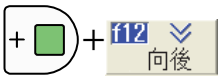
显示出后述的 5 的画面，前进至 6 的步骤。

≫存在编译错误时

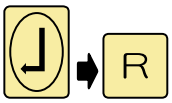


存在编译错误时，用信息显示出错误的总数，第一个错误部位呈红色反转显示。



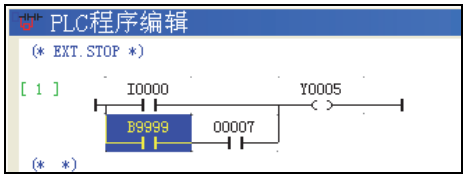


2 **按下［动作可能］＋f11＜向前＞或［动作可能］＋f12＜向后＞之后，光标继续分别移动到下一个错误部位。**
引导信息区域分别显示相应的错误内容。
“［ ］”内的数值是现光标的位置 / 错误总数。

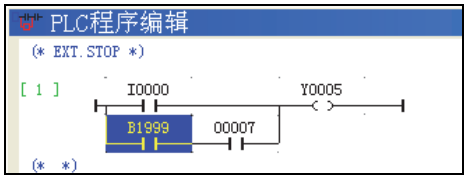


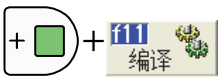
3 **要修正编译错误部位时，按下［Enter］消除消息后，按下［复位 / R］返回通常的编辑画面。**

≫表示错误部位的红色反转显示，恢复成表示编辑状态的蓝色反转显示。

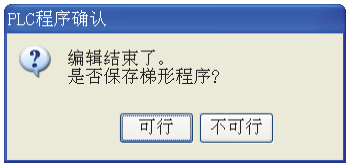


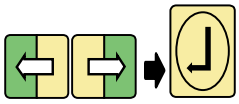
4 **此例中，BOOL 变数的编号偏离了使用范围（2000 为止）。**
将 B9999 修正为 B1999。





5 **再次按下［动作可能］＋f11＜编译＞。**
≫显示不存在任何编译错误的下一信息。



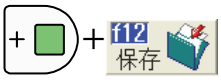


6 **不作保存并继续编辑作业时，通过［左右］选择“不可行”，按下［Enter］。**
选择“可行”时，可以保存编辑中的程序。详情参见下一项“2.3 保存编辑中的程序”。

2.3 保存编辑中的程序

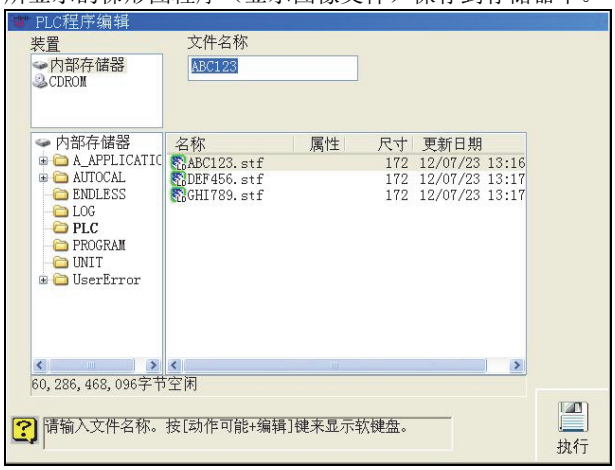
将完成的 PLC 程序以显示图像的形式保存到存储器。

也可以从梯形图编辑器执行所编制 PLC 程序的编译及下载操作。操作和“3.2 从编译到下载”相同。



1 **在任意的光标位置处按下［动作可能］＋f12＜保存＞。**

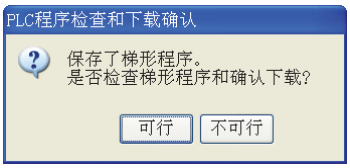
≫所显示的梯形图程序（显示图像文件）保存到存储器中。

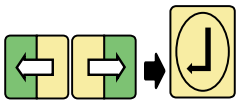


开始编辑之前，请确定指定名称的显示图像文件显示在一览表中。

2 **按下 f12＜写入＞。**

≫显示以下信息。



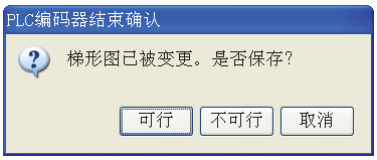


3 **不执行校验和传送并继续编辑作业时，通过［左右］选择“不可行”，按下［Enter］。选择“可行”时，可以执行程序校验和传送。关于程序校验和传送，参见第 3 章“程序校验”。**

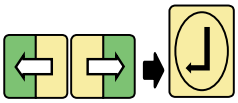
2.4 结束编辑器

结束梯形图编辑器。

1 **在任意的光标位置处按下［复位 / R］。**
≫梯形图程序发生了变更时，会显示如下信息。



梯形图程序未发生变更时，不显示该信息而直接结束。



2 **想要不保存梯形图程序并结束编辑器时，通过［左右］选择“不可行”，按下［Enter］。**
选择“可行”时，保存编辑中的梯形图程序并结束。
选择“取消”时，不结束编辑器。

3章 程序校验

本章说明编制梯形图程序（显示图像文件）正式开始执行前的一连串操作。

3.1 程序校验的概要

要执行所编制的程序文件（显示图像文件），首先要转换成可以执行的形式。称之为编译。
接着将其交给运行时间引擎（执行扫描的软件）。称之为下载。
本控制装置将此一连串的操作统称为“校验”。

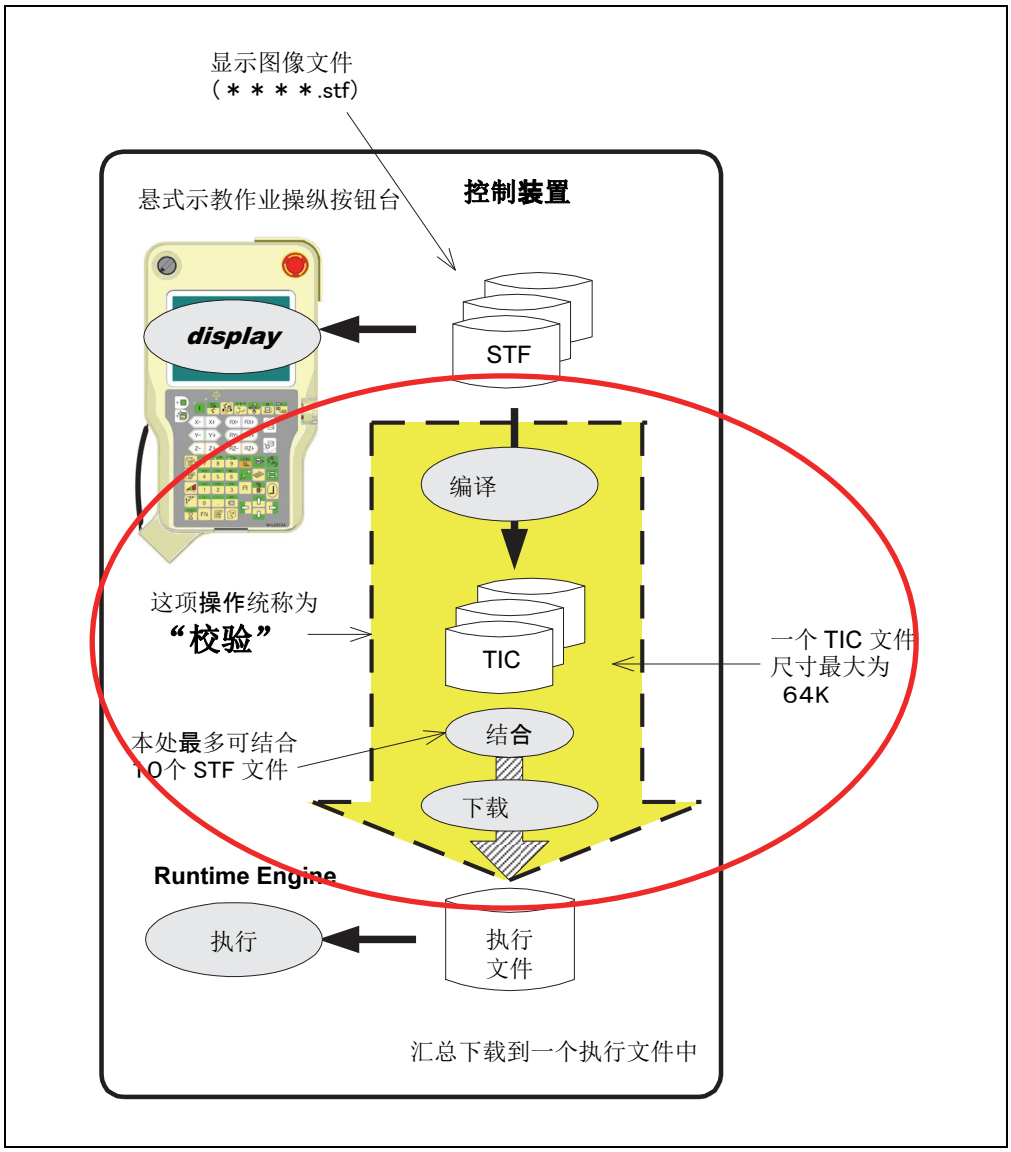


图 3.1.1　软件 PLC 的结构

内部存储器可以存储复数个梯形图程序（显示图像文件）。如图 3.1.2 所示，对应不同用途的梯形图程序全部存储在内部存储器中，从这些程序中选择要执行的程序，执行编译及下载。

另外，本控制装置的软件 PLC **最多可以结合 10 个**梯形图程序执行。较长的程序通过分割可以更为容易编辑。图 3.1.2 的例中，结合了 3 个梯形图程序。

在结合复数个梯形图程序使用时，编译时需要指定这些程序的执行顺序。

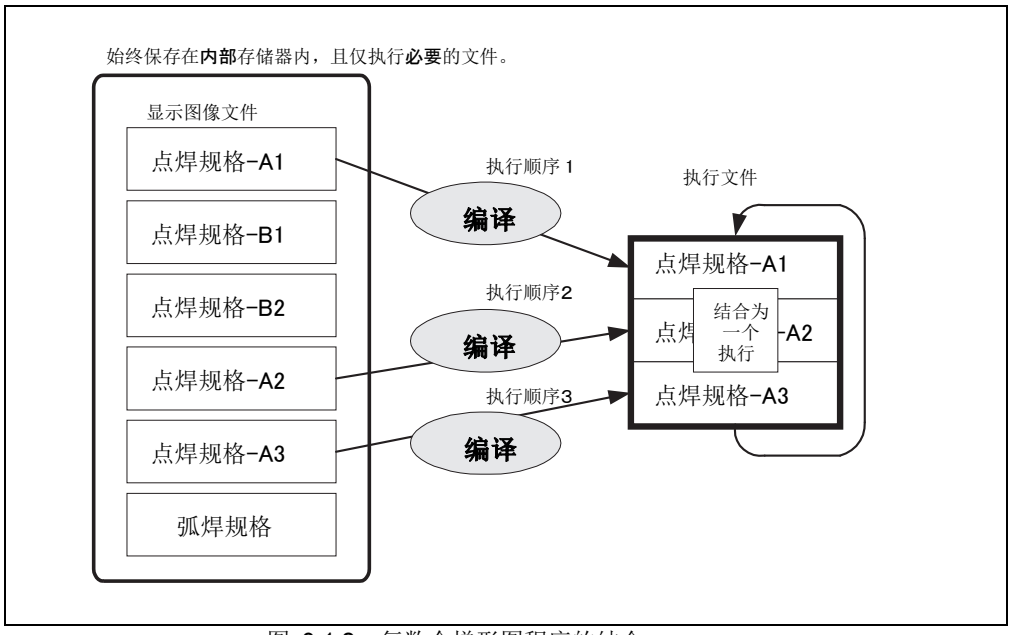


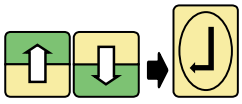
图 3.1.2　复数个梯形图程序的结合

3.2 从编译到下载

本节说明选择内部存储器中存储的梯形图程序（显示图像文件），一边指定执行顺序一边进行编译、下载的操作。

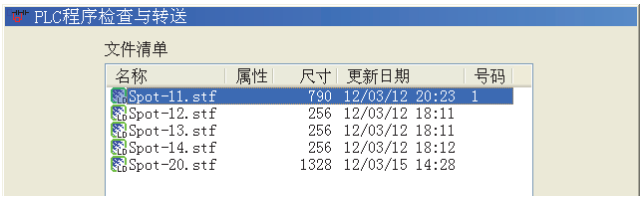
1 **选择＜维修＞－［14 PLC 程序编辑］，从显示的菜单当中选择［3 PLC 程序检查与转送］。**

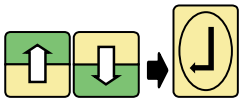
≫显示如下所示的梯形图程序（显示图像文件）的一览。



2 **通过［上下］选择想要编译的梯形图程序，按下［Enter］。**

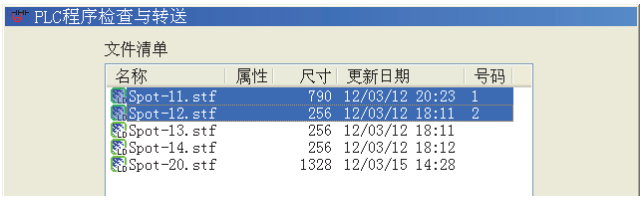
≫选中的梯形图程序呈蓝色反转显示，其右侧显示出“1”的数字。这是表明在结合复数个梯形图程序加以执行时，此文件是第一个执行的梯形图程序。





3 **通过［上下］选择要编译的梯形图程序，按下［Enter］。**

≫选中的梯形图程序呈蓝色反转显示，其右侧显示出“2”的数字。这表明在结合复数个梯形图程序加以执行时，此文件是第二个执行的梯形图程序。



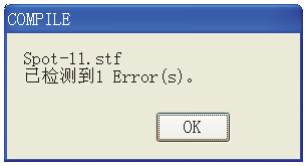
如上所述，按照执行顺序分别选择各个梯形图程序。
采用一个执行文件便可时，仅选中那一个。

4 **执行顺序的指定出错时，将光标对齐该梯形图程序，按下［BS］。**

≫该梯形图程序的执行顺序编号显示消失。

5 **所有的梯形图程序选择完毕后，按下 f12＜执行＞。**

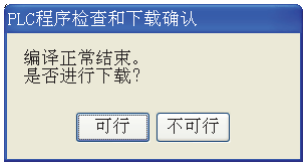
≫从执行顺序编号 1 开始依次执行编译。编译完成后，显示如下所示的结果。

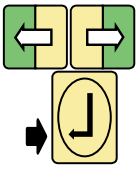


此例表明 ABC123.stf 文件为 OK，ABC456.stf 文件发生了一处编译错误。

6 **存在编译错误时，按下［复位 / R］退出程序校验的菜单。**
通过［1 PLC 程序编辑］起动梯形图编辑器，在这里再次进行编译，确定错误部位加以修正。

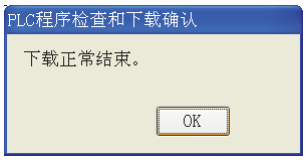
7 **如果没有编译错误，会显示出如下所示的信息。**





8 **通过［左右］选择“可行”，按下［Enter］。**

≫编译的复数个梯形图程序结束为一个文件，下载至运行时间引擎。下载正常结束后，显示如下所示的信息。



至此，已经将程序交给运行时间引擎。
PLC（运行时间引擎）处于启动状态时，暂时停止扫描，下载之后再次开始扫描。

通过［复位 / R］退出菜单。

≫PLC（运行时间引擎）未启动时，请通过“3.3 程序的启动 / 停止 / 分离”的步骤设成起动。

3.3 程序的启动 / 停止 / 分离

本章描述开始 / 停止下载到运行时间引擎的梯形图程序扫描的方法。此操作从常数菜单执行。
通常，将 PLC 状态设定设成“启动”后，无需再变更设定。将控制装置的电源置于 ON 后，开始扫描。
另外，根据外设的连接情况等，也可以暂时不使用软件 PLC（＝设成分离状态）。

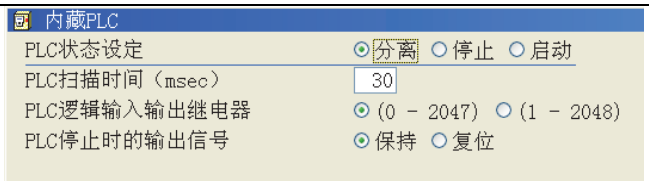
PLC 状态设定分成以下 3 种。

表 3.3.1 程序的启动 / 停止 / 分离

选项	说明
分离	不使用内藏 PLC。 也就是逻辑输入输出和物理输入输出为 1 对 1 连接状态。 PLC（运行时间引擎）不执行扫描。
停止	使用内藏 PLC。 不过，PLC（运行时间引擎）不扫描下载的程序，物理 I/O 的状态无变化。
启动	使用内藏 PLC。 PLC（运行时间引擎）扫描下载的程序。 即使在程序校验时实施下载处理，也不会自动转成＜启动＞。必须在此菜单中切换成＜启动＞。

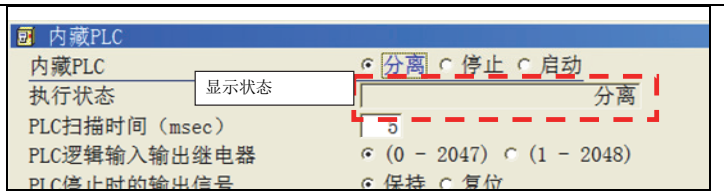
1 在示教模式下，选择 f5＜常数设定＞－［1 控制环境］，从显示的菜单当中选择［4 内藏 PLC］。
≫根据版本而状态显示方法有所不同，因此设定菜单会改变。
显示出如下所示的内藏 PLC 的相关设定菜单。

V3.33 以前

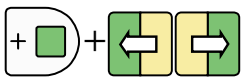


V3.33 以前的情况，当 PLC 操作状态改变，「PLC 状态设定」也会改变，将显示状态。

V3.34 以后



V03.34 以后的情况，将显示状态。如 IO 模拟设定为有效等当 PLC 操作状态改变，显示状态就改变。



2 同时按下［动作可能］＋［左右］，从“分离 / 停止 / 启动”当中选择其一。

3 按下 f12＜写入＞。
≫转成“分离 / 停止 / 启动”的控制状态。

将 PLC 状态设定改成“启动”时，输入输出信号的状态将根据 PLC 程序发生变化。工件把持过程中或者和外设存在干涉的过程中等，执行此操作非常危险。操作时应充分加以注意。

NOTE

4章 程序核对

本章说明软件 PLC 的程序核对。核对是指检查梯形图编辑器上显示以及编辑的程序和实际扫描的程序是否相同。

4.1 程序核对的概要

如图 4.1.1 所示，梯形图编辑器所显示的显示图像文件（STF 文件）和运行时间引擎实际扫描的执行文件不相同。因此，即使在梯形图编辑器更改梯形图程序，并保存文件，如果不进行下载，则两者保持不相同状态，I/O 或梯形监视器无法正确运行。

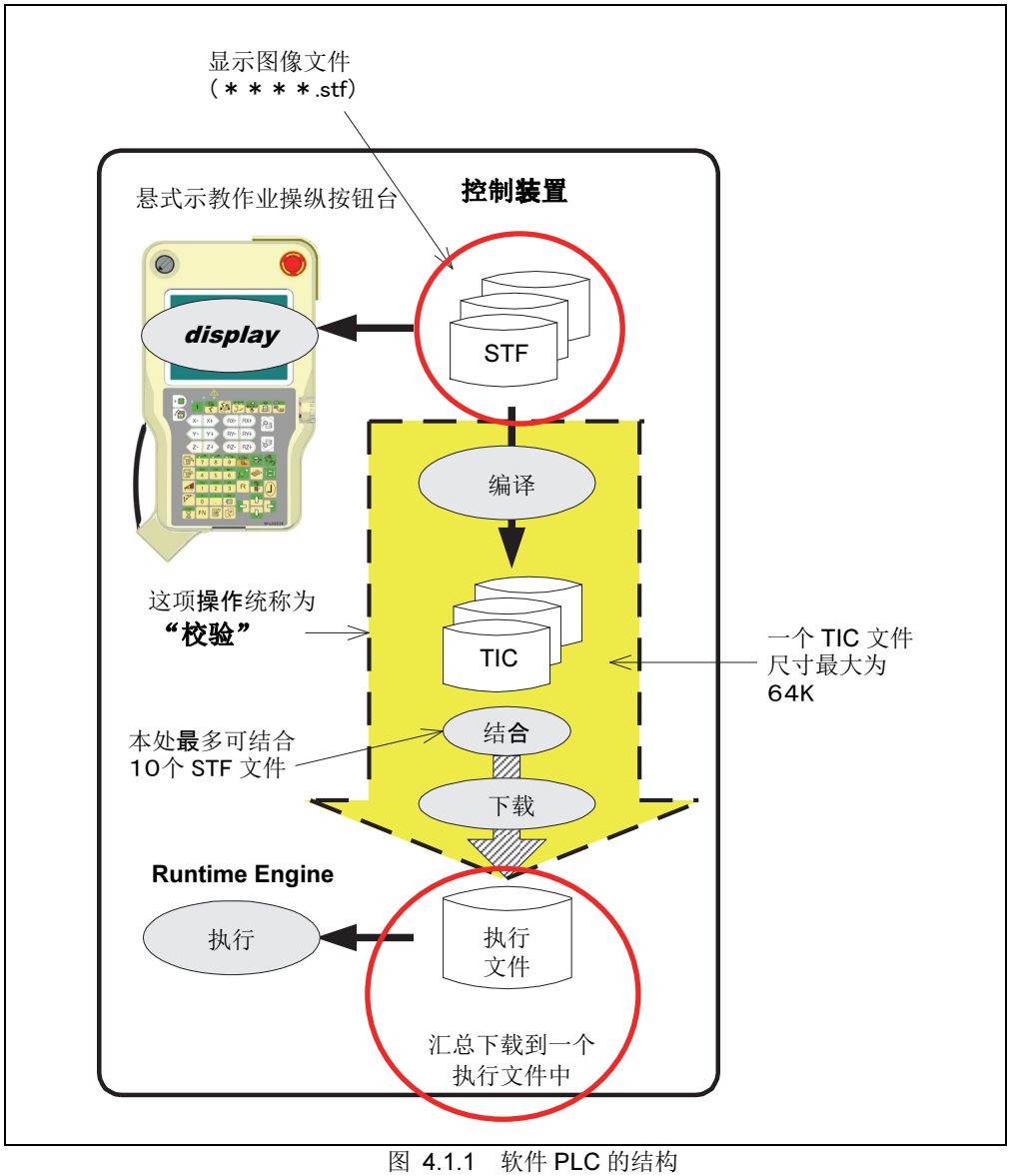


图 4.1.1　软件 PLC 的结构

因此，需要检查梯形图编辑器所显示、所编辑的梯形图程序（显示图像文件），和运行时间引擎实际扫描的程序是否相同。称之为“程序核对”。正处于 PLC 扫描过程中也可执行程序核对。

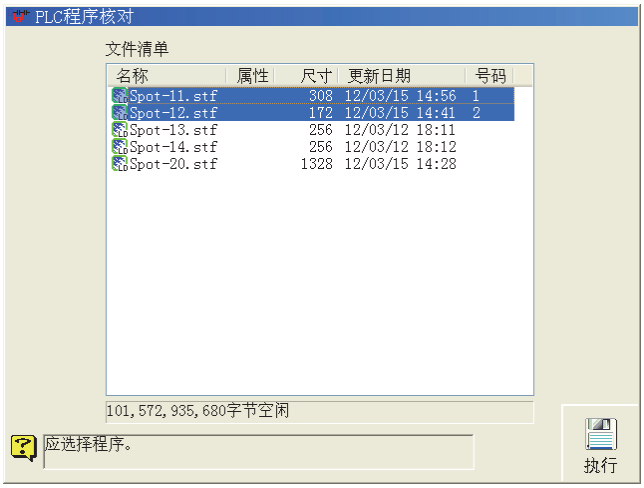
程序核对时，将核对对象的（复数个）STF 文件临时编译、结合后的结果和运行时间引擎具备的执行文件比较。

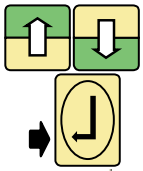
程序核对时如果检测到不一致，也无法确定不一致部位具体处在何处。

4.2 程序核对

以下描述选择内部存储器中存储的梯形图程序（显示图像文件），一边明示执行顺序一边编译，并查看和已经下载到运行时间引擎的程序间的比较结果的操作方法。

1 **打开＜维修＞－［14 PLC 程序编辑］，从显示的菜单当中选择［2 PLC 程序核对］。**
≫显示如下图所示的梯形图程序（显示图像文件）的一览。

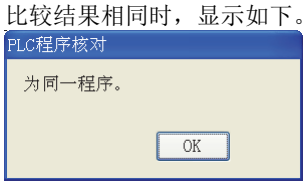




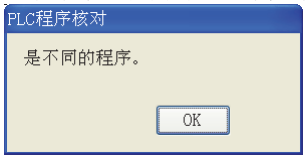
2 **通过［上下］选择要核对的梯形图程序，按下［Enter］。**
和校验（编译）时完全一样，根据执行的顺序依次按下［Enter］。

3 **所有的梯形图程序选择完毕后，按下 f12＜执行＞。**
≫和校验（编译）一样从编译执行到结合所生成的执行文件和运行时间引擎具备的程序相比较。

比较结果相同时，显示如下。



比较结果不相同时，显示如下。



4 **选中的梯形图程序发生编译错误时，无法核对。**
显示和校验（编译）相同的错误信息，修正后请再次执行核对。

5 **通过［复位 / R］退出菜单。**

5章 监视器

本章说明软件 PLC 的监视器功能。本产品备有以梯形监视器为首的多种监视器，请根据不同目的加以活用。

5.1 监视器的概要

软件 PLC 在“起动”状态时，可以实时显示接点或线圈的 ON/OFF 状态。可以在梯形图上显示接点或线圈的状态，显示特定的线圈等，具备复数种监视器功能。请根据不同目的加以活用。

监视器的刷新周期为 200msec。高速执行 ON/OFF 的接点或线圈有可能无法正确显示。

重点

软件 PLC 处于“分离”状态时，不显示所有的监视器。

重要

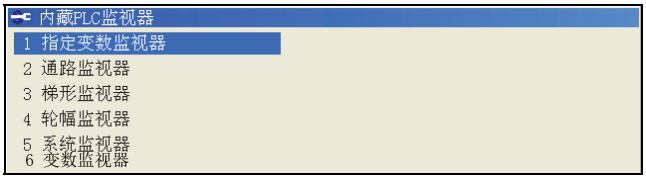
文件名称长度根据系统软件版本，将受到限制。

V3.21 以前梯形程序文件名称为 16 个文字以上时，无法执行有些监视器（梯形监视器、轮辐监视器）功能。文件名称应设为 16 个文字以内。

维修

1 选择＜维修＞－［3 监视器 1］～［6 监视器 4］，从显示的菜单当中选择［32 内藏 PLC 监视器］。

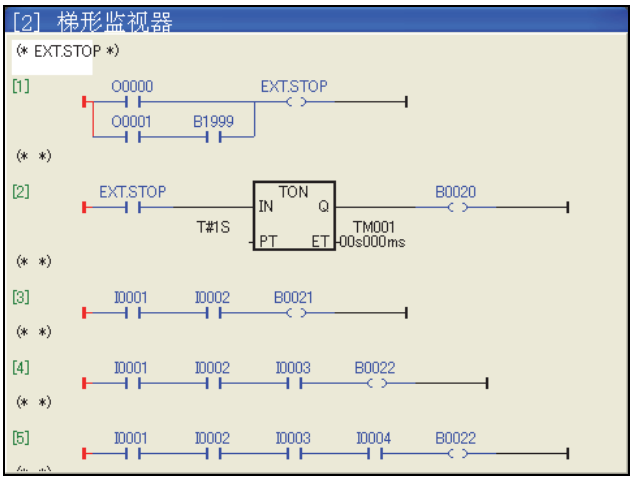
≫显示下图所示的内藏 PLC 监视器菜单的一览。



2 备有共 5 种监视器。

通过［上下］选择想要利用的监视器菜单，按下［Enter］。

≫例如梯形监视器显示如下。



各监视器菜单的内容在下一项说明。

3 关闭时，通过和其他的监视器相同的［画面移动］，将监视器画面设为活动状态后，同时按下［动作可能］和［画面移动］。

5.2 指定变数监视器

最多可以同时监视器 16 个任意变数。16 个变数和格式无关，可以自由指定。
监视器数据通过如下所示的文本显示。

①通用输入、通用输出、物理输入、物理输出

I0000 **TRUE**
IO 编号 TRUE/FALSE

②BOOL 变数（通常变数、保持变数）

B0000 **TRUE**
BOOL 编号 TRUE/FALSE

③实数变数（通常变数、保持变数）

R0001 **2345.0005**
实数变数编号 实数显示

④整数变数（通常变数、保持变数）
整数显示通过输入［左右］键，可以将第 3 个显示部分切换成 16 进制或 2 进制显示。
16 进制显示时

D0001 **12345** **00003039 H**
整数变数编号 整数显示 16 进制显示

2 进制显示时

D0001 **12345** **00000000 00000000 00110000 00111001 B**
整数变数编号 整数显示 2 进制显示

⑤短整数变数（通常变数、保持变数）
短整数显示通过输入［左右］键，可以将第 3 个显示部分切换成 16 进制或 2 进制显示。
16 进制显示时

DB0001 **123** **7B H**
短整数变数编号 整数显示 16 进制显示

2 进制显示时

DB0001 **12345** **01111011 B**
短整数变数编号 整数显示 2 进制显示

⑥定时器变数（通常变数、保持变数）

TM001 **00h00m12s345ms**
定时器变数编号 定时器显示

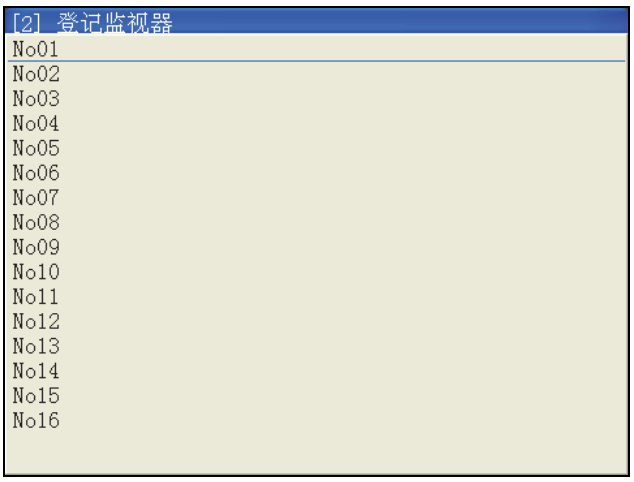
⑦文字变数（通常变数、保持变数）
字符串显示通过输入［左右］键，可以分别移动 1 个文字后显示文字显示部的显示部分。

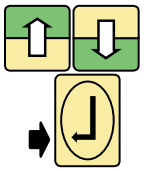
ST001 **ABCDEFGHIJKLMNOPQRSTUVWXYZ012345** **255**
字符串变数 文字显示 文字数

5.2.1 显示指定变数监视器

1 **从内藏 PLC 监视器菜单当中，选择［1 指定变数监视器］。**

≫显示出下图所示的监视器画面。

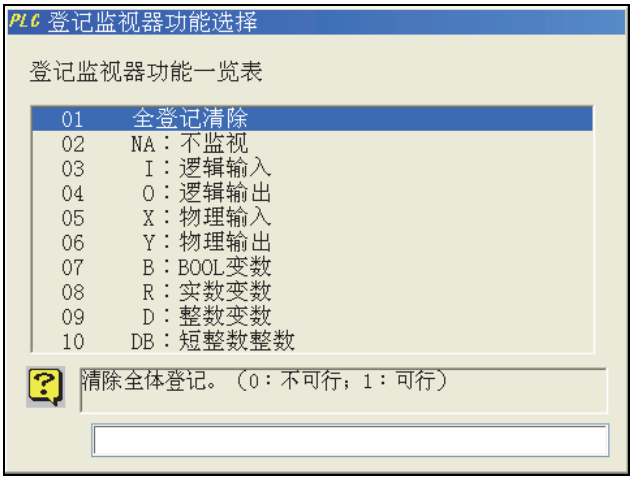


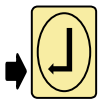


2 **通过［上下］选择适当的编号（例如 No.01），按下［Enter］。**

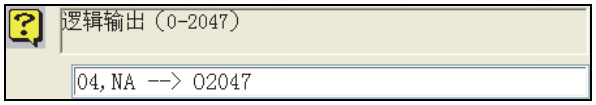
≫显示下图所示的变数种类的选择对话框。

在此对话框中选择想要监视器显示的变数。

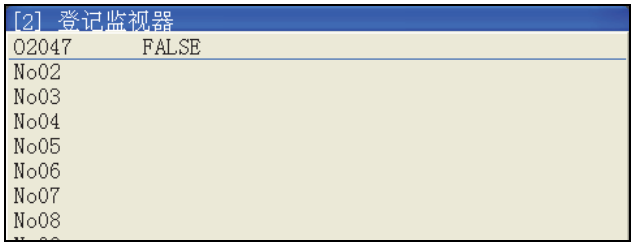


数值 

3 **例如，要对逻辑输出 2047 进行监视器显示时，将光标对齐“04 O：逻辑输出”，按下［Enter］，接着从数字键盘输入 2047，按下［Enter］。**

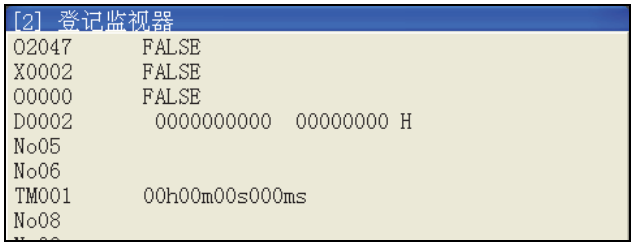


4 **变成如下的监视器画面。**



监视器编号 No.01 的部分变成物理输出 O2047，其右侧显示为监视器数据（此时为 FALSE）。

5 **同前面方法一样，可以依次指定 No.2～16 最多可达 16 个变数。**
编号跳越也没有关系。



即使切断电源，也可以保存监视器显示设定。

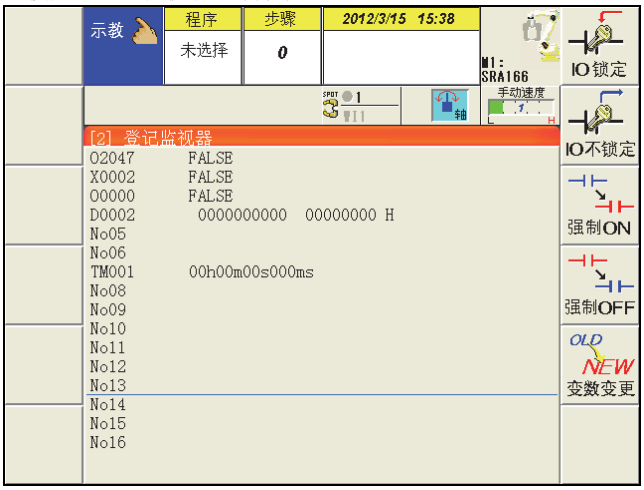
6 **想要将设定了监视器显示的编号恢复成不监视器的状态时，**
设定为“02 NA：不监视”。

7 **取消所有的监视器显示设定时，**
选择“01 全登记清除”，指定 1（可行）。

5.2.2 强制设定

在指定变数监视器画面可以强制更改变数的内容。本节说明其操作方法。

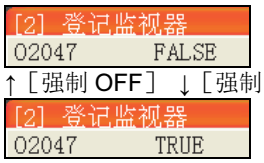
1 **将监视器画面设为活动状态下，按下［编辑］。**
≫标题栏转为表明编辑状态的红色，f 键显示变化如下。
使用这些 f 键执行强制设定。



2 **通过［上下］键将光标对齐要强制设定的变数。**

3 **强制设定逻辑输入输出、物理输入输出的 ON/OFF 状态时，**
首先按下 f7＜IO 锁定＞。
≫选中的变数暂时成为从 PLC 引擎分离的状态。扫描并未停止。

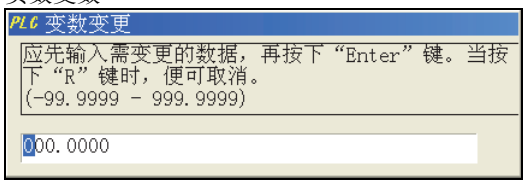
4 **强制 ON（设为 TRUE）时，按下 f9＜强制 ON＞;**
强制 OFF（设为 FALSE）时按下 f10＜强制 OFF＞。



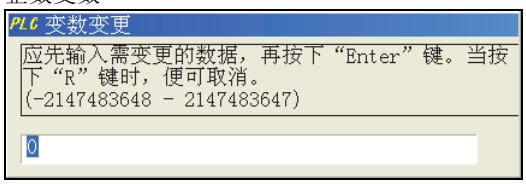
在此状态下关闭监视器画面，或打开其他的 PLC 监视器画面，强制设定状态（IO 锁定状态）都不变化。

5 **同上面方法一样，实数变数、整数变数、短整数变数、定时器变数、字符串变数都可使用 f11＜变数变更＞执行强制设定。**
分别显示出下述信息，根据显示的引导信息输入数值，最后再按下［Enter］。
字符串变数时会显示出软键盘，在该处输入新的字符串，按下 f12＜写入＞。

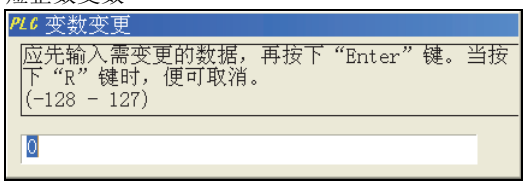
实数变数



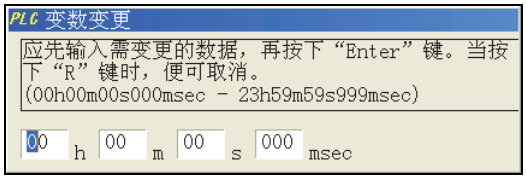
整数变数



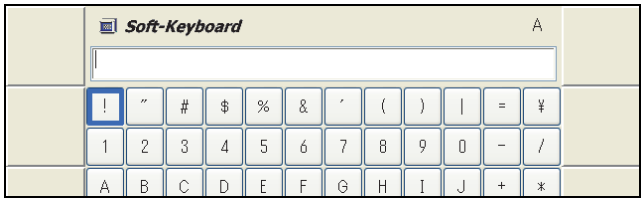
短整数变数



定时器变数



字符串变数



6 **取消输入时，按下［复位 / R］。**

7 **强制设定的操作结束后，按下 f8＜IO 不锁定＞。**
≫解除暂时性的分离状态，恢复通常的扫描状态。
切勿忘记执行此操作。

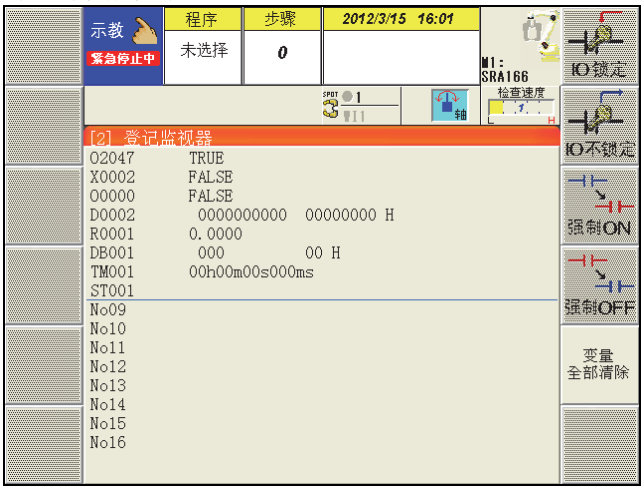
8 **按下［复位 / R］返回原先的监视器状态画面。**

5.2.3 变数的统一初始化

在指定变数监视器画面下，对于变数（停电保持软元件）的内容可以执行统一初始化。这里对其操作方法进行说明。

1 **将监视器画面设为活动状态下，按下［编辑］。**

2 **按下［动作可能］。**
≫f 键显示变化如下。



3 **按下 f11＜变数统一初始化＞。**
≫变数(停电保持软元件)执行初始化。

5.3 通道监视器

通道监视器时，能够仅将软件 PLC 当前执行中的程序所使用的变数，以一览方式显示。变数按“逻辑输入”“逻辑输出”“物理输入”“物理输出”“BOOL 变数”“短整数变数”“整数变数”“实数变数”“定时器变数”“字符串变数”的不同种类分别显示。

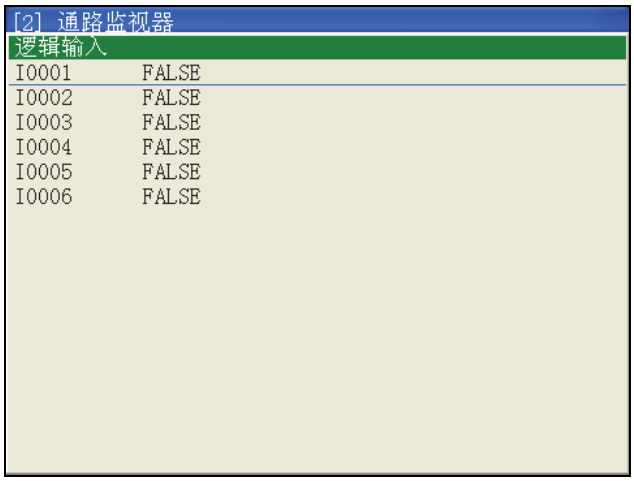
监视器数据以和指定变数监视器相同形式的文本加以显示。

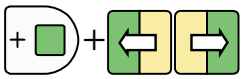
5.3.1 显示通道监视器

1 **从监视器菜单当中选择［2 通道监视器］。**

≫显示下图所示的监视器画面。（下图为全画面显示）

PLC 执行过程中的程序所使用的所有“逻辑输入”变数的状态以一览方式显示。





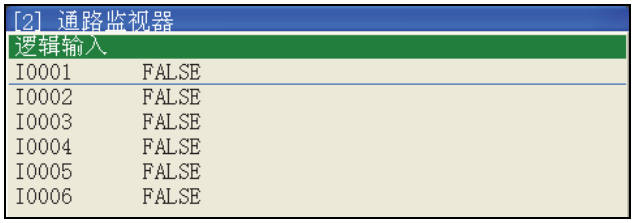
2 **按下［动作可能］＋［左右］。**

≫按照“逻辑输入”－“逻辑输出”－“物理输入”－“物理输出”－“BOOL 变数”－“整数变数”－“短整数变数”－“实数变数”－“定时器变数”－“字符串变数”的顺序切换监视器画面。

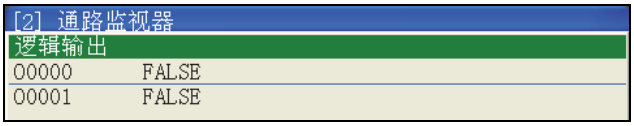
任一种画面中，均为仅以一览方式显示出 PLC 当前执行中的程序所使用变数的状态。

未作记录的变数时，则无任何显示。

“逻辑输入”

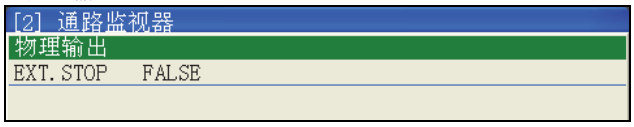


“逻辑输出”

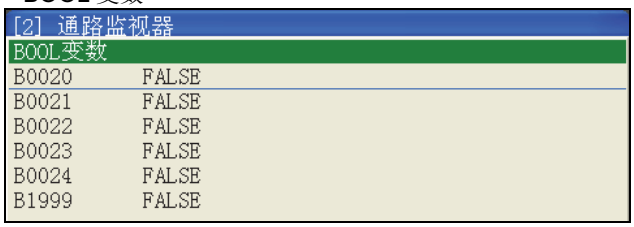


“物理输入”

“物理输出”



“BOOL 变数”

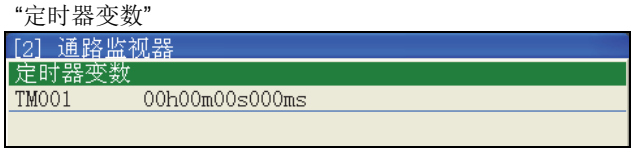


“实数变数”

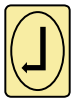
“整数变数”

“短整数变数”

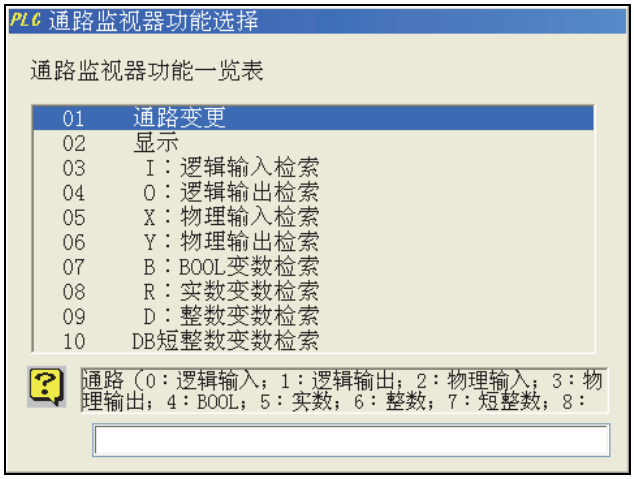
“定时器变数”

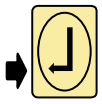


“字符串变数”

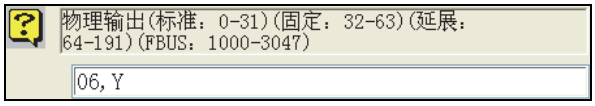


3 **可以在检索任意变数是否用于执行中程序的同时，跳转到相应的监视器画面。**
在任意通道的显示画面按下［Enter］。
≫显示如下所示的对话框。

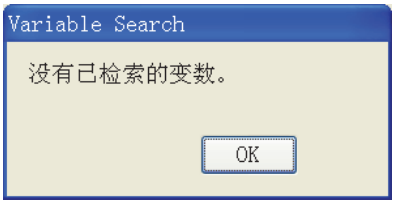


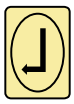
数值 

4 **在这些指定要检索的变数（＝想要监视器显示的变数）。**
检查（监视器显示）物理输出 Y0007 时，选择“06 Y:物理输出检索”，按下[7][Enter]。



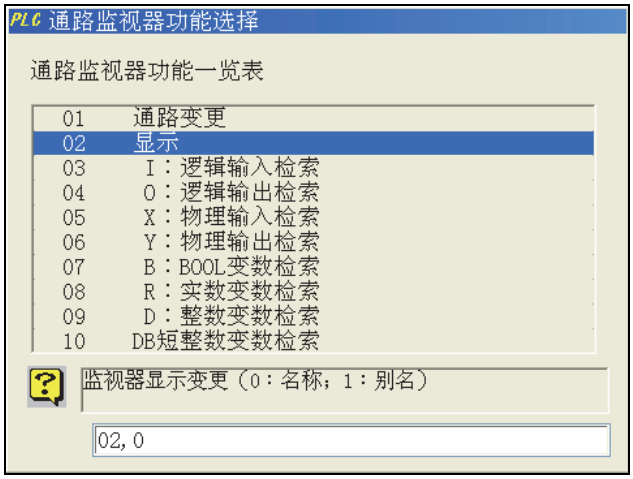
5 **监视器画面切换为“物理输出”，光标移到指定的 Y0007 的监视器显示行。**
如果物理输出 Y0007 在程序内部未使用时，显示如下。





6 **按下［Enter］，消除信息。**

7 **在通道监视器可以用别名执行监视器显示。**
在任意通道的显示画面按下［Enter］，选择［02 显示］。



8 **要以变数名称显示时按下［0］［Enter］**
要以别名显示时按下［1］［Enter］。
≫显示的通道以变数名称（或别名）进行显示。
以字母表顺序排列显示，变数名称和别名时的显示顺序不相同。

5.3.2 强制设定

在通道监视器画面中，可以强制更改变数的内容。操作方法和指定变数监视器相同。

5.3.3 变数的统一初始化

在通道监视器画面中，可以将变数（停电保持软元件）的内容统一执行初始化。操作方法和指定变数监视器相同。

5.4 梯形监视器

对于 PLC 正在执行的程序，以梯形图显示的状态进行监视器显示。

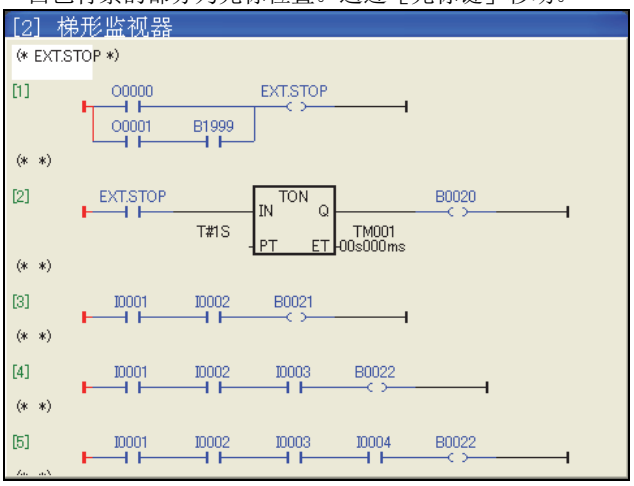
5.4.1 显示梯形监视器

1 **从监视器菜单当中选择［3 梯形监视器］。**

≫显示下图所示的监视器画面。（下图为全画面显示时）

从 PLC 当前执行的程序的前头开始显示。

白色背景的部分为光标位置。通过［光标键］移动。



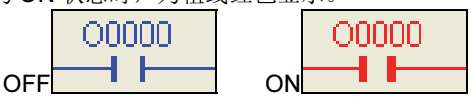
结合并执行有复数个程序时

1 个梯形监视器能显示的程序（显示图像文件；***.stf）仅为 1 个。结合复数个程序并加以执行时，会显示用于选择监视器显示的程序对话框。在对话框中选择要显示的程序。

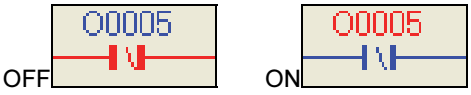
监视器画面最多可以同时显示（打开）4 个。要同时监视器复数个程序时，可在别的监视器画面显示梯形监视器，并选择别的程序。

接点和线圈的 ON/OFF、定时器等的当前值显示如下。

A 接点为 ON 状态时，为粗线红色显示。

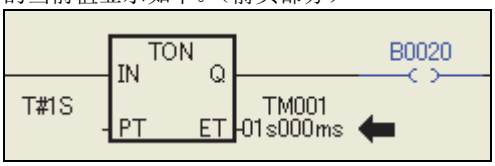


B 接点为 OFF 状态时，为粗线红色显示。



线圈为 ON 状态时，为粗线红色显示。

定时器的当前值显示如下。（箭头部分）

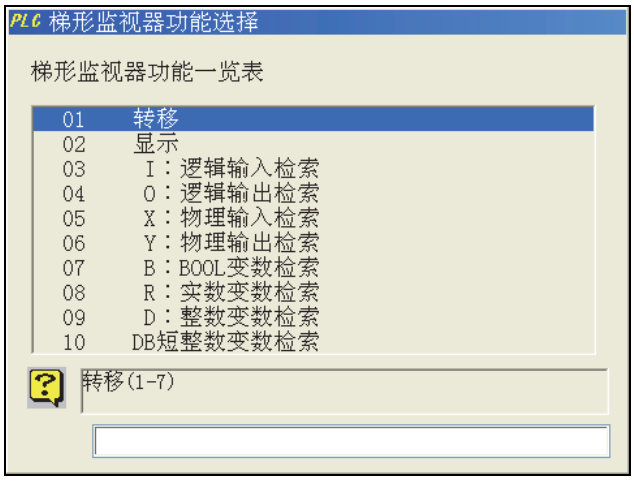


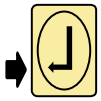
关于是否到时限，上述示例中时，对应于输出 Q 所连接的 BOOL 变数 B0020 的 ON/OFF 状态。

---

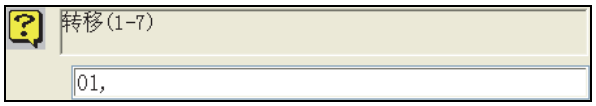
2 **梯形监视器画面可以指定轮幅编号或指定任意的变数，并跳转到其显示部。**
在任意的光标位置按下［Enter］。

≫显示如下所示的对话框

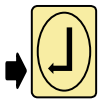


数值 

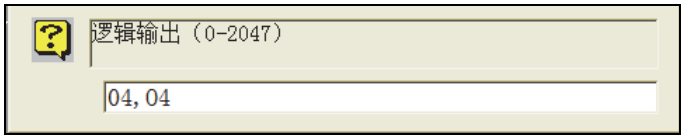
3 **指定轮幅编号跳转时**
选择“01 跳转”，按下［Enter］。
之后，输入想要显示的轮幅编号，按下［Enter］。



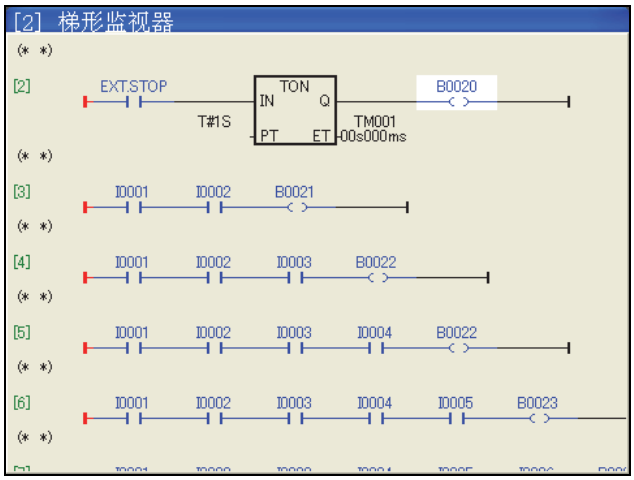
≫显示梯形监视器画面，光标定位到指定的轮幅（闭合回路）的前头。

数值 

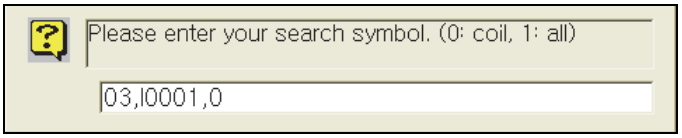
4 **检索逻辑输出、物理输入、实数变数、整数变数、短整数变数、定时器变数、字符串变数进行跳转时**
例如检索逻辑输出 B0004（向 B0004 跳转），则选择“04 O；逻辑输出检索”，输入“数值”＋［Enter］。



≫显示出梯形监视器画面，光标定位到指定的变数。

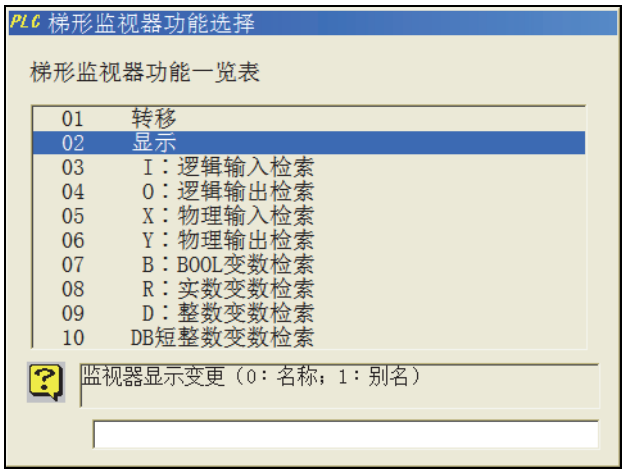


逻辑输入检索、物理输出检索、BOOL 变数检索的情况，可以选择只限于线圈中检索或者所有检索。
例如，只限于线圈中检索（向 I0001 跳转）的逻辑输入为 I0001 时，选择“03 I；逻辑输入检索”，输入“数值”＋［Enter］，并且输入检索符号“0”＋［Enter］。
此外，执行所有检索时，输入检索符号“1”＋［Enter］。

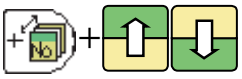


≫显示出梯形监视器画面，光标定位到指定的变数。

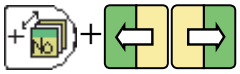
5 **梯形监视器时，能够用别名进行监视器显示。**
在梯形图显示画面按下［Enter］，选择［02 显示］。



6 **以变数名称显示时按下［0］[Enter]**
以别名显示时按下［1］[Enter]。



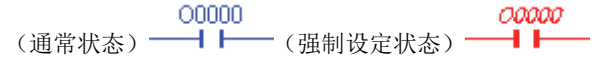
7 **按下［上档键］＋［上下]。**
≫执行向下方向的重新检索、向上方向的重新检索。
没有执行过检索时，无法执行重新检索。



8 **按下［上档键］＋［左右]。**
≫只限于线圈中执行向下方向的重新检索、向上方向的重新检索。
没有执行过检索时，无法执行重新检索。

5.4.2 强制设定

在梯形监视器画面中可以强制更改变数的内容。
操作方法和指定变数监视器、通道监视器相同。
强制设定中的文字为斜体。



5.4.3 变数的统一初始化

在梯形监视器画面中可以将变数（停电保持软元件）的内容统一执行初始化。
操作方法和指定变数监视器、通道监视器相同。

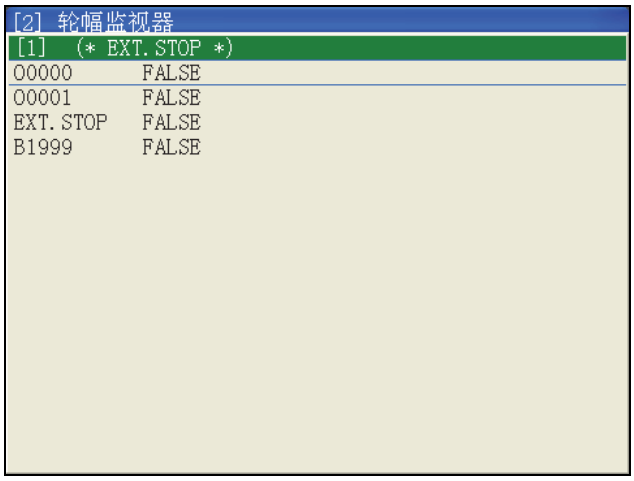
5.5 轮幅监视器

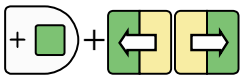
轮幅监视器时，可以将各条轮幅所使用的所有变数，按各轮幅（闭合回路）分别监视器显示。
监视器数据按照和指定变数监视器相同形式的文本显示。

5.5.1 显示轮幅监视器

1 **从监视器菜单当中，选择［4 轮幅监视器］。**

≫显示出下图所示的监视器画面。（下图为全画面显示时）
显示出 PLC 当前执行的程序的前头轮幅所使用的所有变数。
最上面的显示是轮幅编号和轮幅注释。

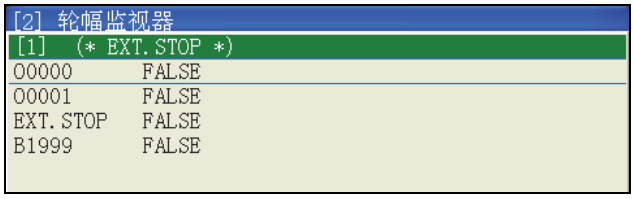




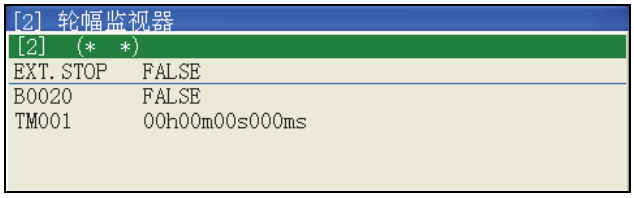
2 **按下［动作可能］＋［左右］。**

≫从前头的轮幅开始依次切换监视器画面。
任一画面下，均将以一览方式显示出该轮幅所使用的所有变数的状态。

轮幅［1］

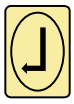


轮幅［2］



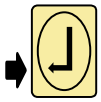
轮幅［3］
・・・
轮幅［7］

不从最末尾的轮幅返回前头的轮幅。

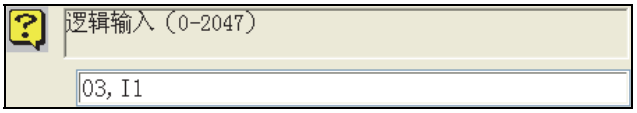


3 **可以在检索任意变数是否用于执行中的程序的同时，跳转到其监视器画面。**
在任意的轮幅监视器画面按下［Enter］。
≫显示下图所示的对话框。

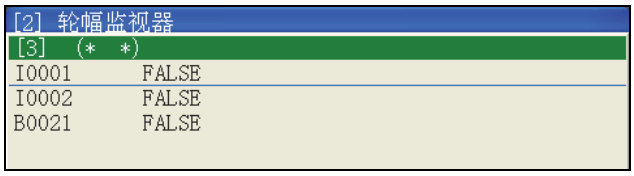
数字



4 **在这里指定要检索的变数（＝要监视器显示的变数）。**
检索（监视器显示）逻辑输入 I0001 时，选择“03 I：逻辑输入检索”，按下[1][Enter]。



5 **从当前显示的轮幅向后检索。**
切换为首先发现的轮幅的显示画面，光标移到指定的 I0001。



6 **轮幅监视器时能够以别名进行监视器显示。**
操作方法和通道监视器等相同。

5.5.2 强制设定

在轮幅监视器画面中可以强制更改变数的内容。
操作方法和指定变数监视器、通道监视器、梯形监视器相同。

5.5.3 变数的统一初始化

在轮幅监视器画面中可以将变数（停电保持软元件）的内容统一初始化。
操作方法和指定变数监视器、通道监视器、梯形监视器相同。

5.6 系统监视器

系统监视器是指诊断软件 PLC 的状态的监视器画面。

1 **从监视器菜单当中选择［5 系统监视器］。**
≫显示下图所示的监视器画面。

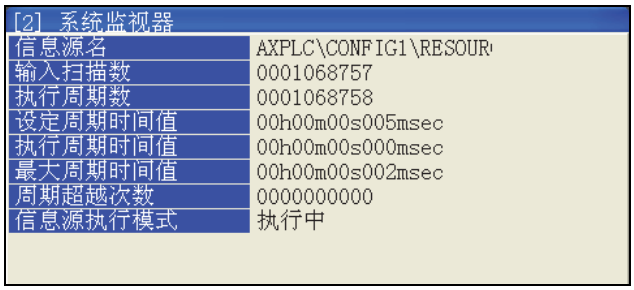


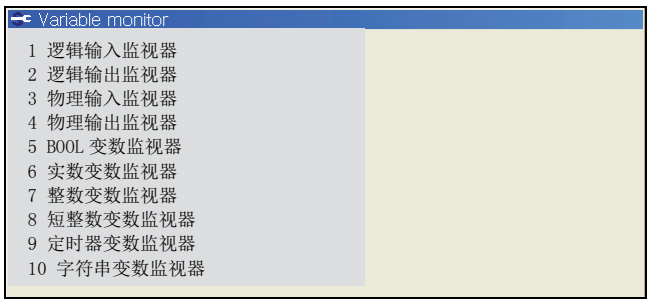
表 5.6.1 系统监视器的显示项目

显示项目	内容	
输入扫描数	是将前一次程序下载到运行时间引擎到现在为止的输入扫描次数。 以无符号 32 位数值进行显示，到达最大值后返回 0。	
执行周期数	是将前一次程序下载到运行时间引擎到现在为止的执行次数。 以无符号 32 位数值进行显示，到达最大值后返回 0。	
设定周期时间值	是设定的 PLC 扫描间隔（从开始扫描到开始下一次扫描为止的时间）。 工厂出厂时设定为 30msec。	
执行周期时间值	是实际扫描所花费的时间。	
最大周期时间值	是实际扫描所花费的时间的最大值。	
周期超越次数	是实际扫描所花费的时间超出“设定周期时间值”的次数的累计值。	
信息源执行模式	表示运行时间引擎的执行状态。	
	无资源	未下载资源（程序）。
	等待执行	资源（程序）的执行为停止状态。
	执行中	正在执行资源（程序）。
	中断点 （break point）	执行过程中在中断点处停止。(中断点仅可在 ISAGRAF 工作台使用。)
	异常	发生了致命的错误。

5.7 变数监视器

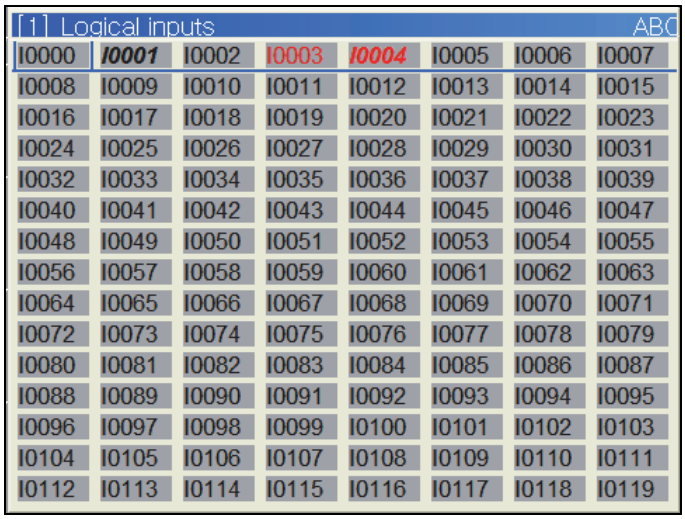
维修

1 **从监视器菜单当中选择 [6 变数监视器]。**
≫显示如下所示变数监视器菜单一览。



5.7.1 逻辑输入监视器

1 **从变数监视器菜单当中选择 [1 逻辑输入监视器]。**
≫显示如下所示监视器画面。（下图为全画面显示）
显示所有[逻辑输入]变数的状态。



根据变数状态将显示为如下所示。

信号状态：OFF	背景颜色：灰色	信号状态：ON	背景颜色：黄色
PLC 程序中未使用	文字：一般显示 I0002	PLC 程序中使用	文字：强调显示 I0001
IO 锁定：OFF	文字颜色：黑色	IO 锁定：ON	文字颜色：红色

重点

根据＜常数设定＞－［1 控制环境］－［6 内藏 PLC］画面的“PLC 逻辑输入输出继电器(编号)”设定，将显示内容有所不同。
（0 - 2047）：显示为 I0000 ～ I2047 。
（1 - 2048）：显示为 I0001 ～ I2048 。

5.7.2 逻辑输出监视器

1 **从变数监视器菜单当中选择 [2 逻辑输出监视器]。**

≫显示如下所示监视器画面。（下图为全画面显示）
显示所有[逻辑输出]变数的状态。

根据变数状态将显示以下内容。

信号状态：OFF	背景颜色：灰色	信号状态：ON	背景颜色：黄色
PLC 程序中未使用	文字：一般显示 O0002	PLC 程序中使用	文字：强调显示 O0001
IO 锁定：OFF	文字颜色：黑色	IO 锁定：ON	文字颜色：红色

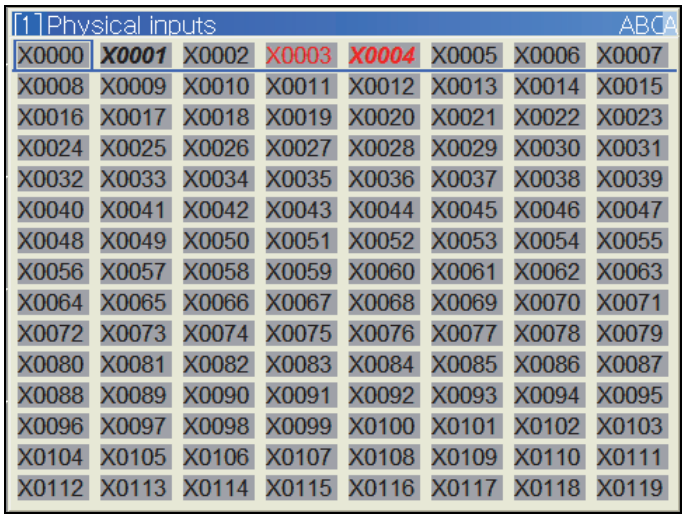
重点

根据＜常数设定＞－［1 控制环境］－［6 内藏 PLC］画面的“PLC 逻辑输入输出继电器(编号)”设定，将显示内容有所不同。
（0 - 2047）：显示为 I0000 ～ I2047 。
（1 - 2048）：显示为 I0001 ～ I2048 。

5.7.3 物理输入监视器

1 **从变数监视器菜单当中选择 [3 物理输入监视器]。**

≫显示如下所示监视器画面。（下图为全画面显示）
显示所有[物理输入]变数的状态。



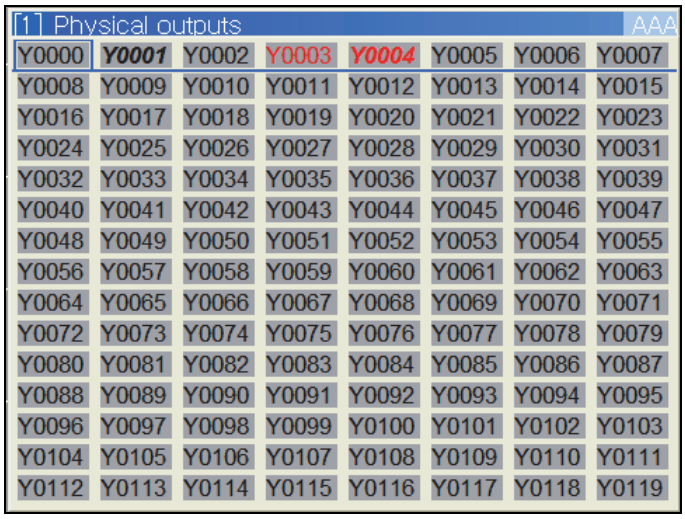
根据变数状态将显示以下内容。

信号状态：OFF	背景颜色：灰色	信号状态：ON	背景颜色：黄色
PLC 程序中未使用	文字：一般显示 X0002	PLC 程序中使用	文字：强调显示 **X0001**
IO 锁定：OFF	文字颜色：黑色	IO 锁定：ON	文字颜色：红色

5.7.4 物理输出监视器

1 **从变数监视器菜单当中选择 [4 物理输出监视器]。**

≫显示如下所示监视器画面。（下图为全画面显示）
显示所有[物理输出]变数的状态。



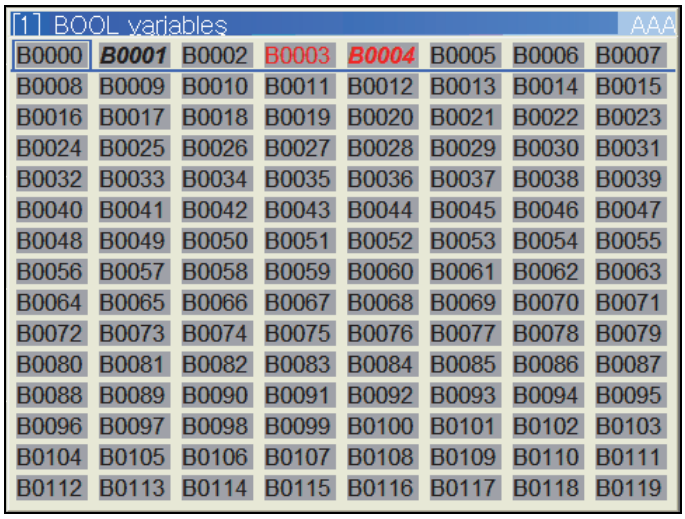
根据变数状态将显示以下内容。

信号状态：OFF	背景颜色：灰色	信号状态：ON	背景颜色：黄色
PLC 程序中未使用	文字：一般显示 	PLC 程序中使用	文字：强调显示 
IO 锁定：OFF	文字颜色：黑色	IO 锁定：ON	文字颜色：红色

5.7.5 BOOL 变数监视器

1 **从变数监视器菜单当中选择 [5 BOOL 变数监视器]。**

≫显示如下所示监视器画面。（下图为全画面显示）
显示所有[BOOL]变数的状态。



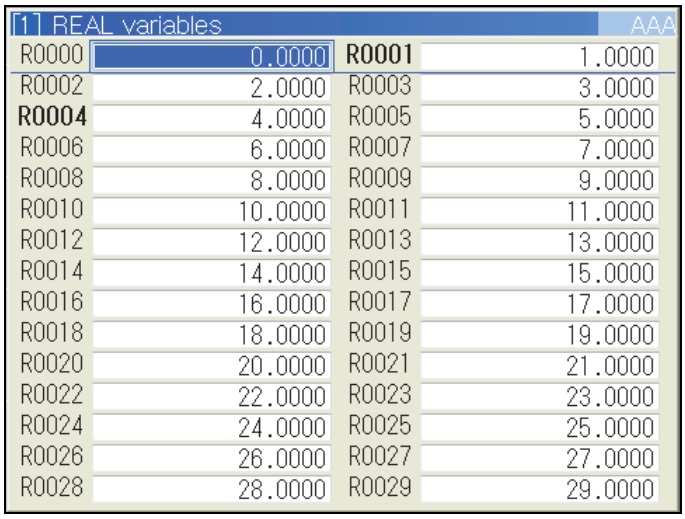
根据变数状态将显示以下内容。

变数值：FALSE	背景颜色：灰色	变数值：TRUE	背景颜色：黄色
PLC 程序中未使用	文字：一般显示 B0002	PLC 程序中使用	文字：强调显示 **B0001**

5.7.6 实数变数监视器

1 **从变数监视器菜单当中选择 [6 实数变数监视器]。**

≫显示如下所示监视器画面。（下图为全画面显示）
显示所有[实数]变数的状态。



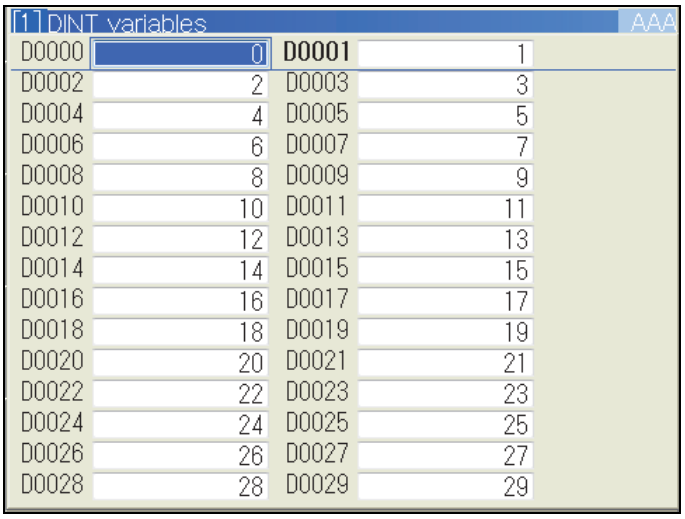
根据变数状态将显示以下内容。

PLC 程序中未使用	文字：一般显示	PLC 程序中使用	文字：强调显示
	R0000		**R0001**

5.7.7 整数变数监视器

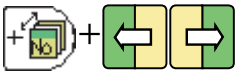
1 **从变数监视器菜单当中选择 [7 整数变数监视器]。**

≫显示如下所示监视器画面。（下图为全画面显示）
显示所有[整数]变数的状态。



根据变数状态将显示以下内容。

PLC 程序中未使用	文字：一般显示	PLC 程序中使用	文字：强调显示
	D0000		D0001



2 **按下［上档键］＋［左右］。**

≫将切换变数的显示形式。
10 进数显示形式时切换为 16 进数；16 进数显示形式时切换为 10 进数。
初始状态为 10 进数显示。

10 进数显示

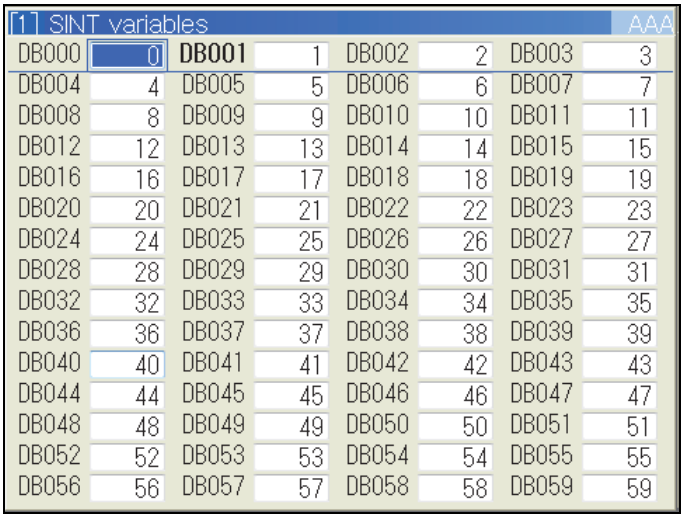
16 进数显示

5.7.8 短整数变数监视器

1 **从变数监视器菜单当中选择 [8 短整数变数监视器]。**

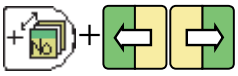
≫显示如下所示监视器画面。（下图为全画面显示）

显示所有[短整数]变数的状态。



根据变数状态将显示以下内容。

PLC 程序中未使用	文字：一般显示	PLC 程序中使用	文字：强调显示
	DB000		DB001



2 **按下［上档键］＋［左右］。**

≫将切换变数的显示形式。

10 进数显示形式时切换为 16 进数；16 进数显示形式时切换为 10 进数。

初始状态设定为 10 进数显示。

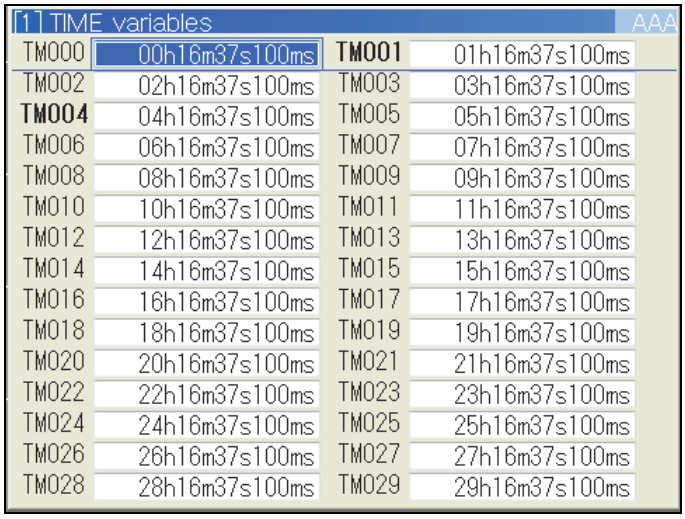
10 进数 DB056 56

16 进数 DB056 38

5.7.9 定时器变数监视器

1 **从变数监视器菜单当中选择 [9 定时器变数监视器]。**

≫显示如下所示监视器画面。（下图为全画面显示）
显示所有[定时器]变数的状态。



根据变数状态将显示以下内容。

PLC 程序中未使用	文字：一般显示	PLC 程序中使用	文字：强调显示
	TM000		TM001

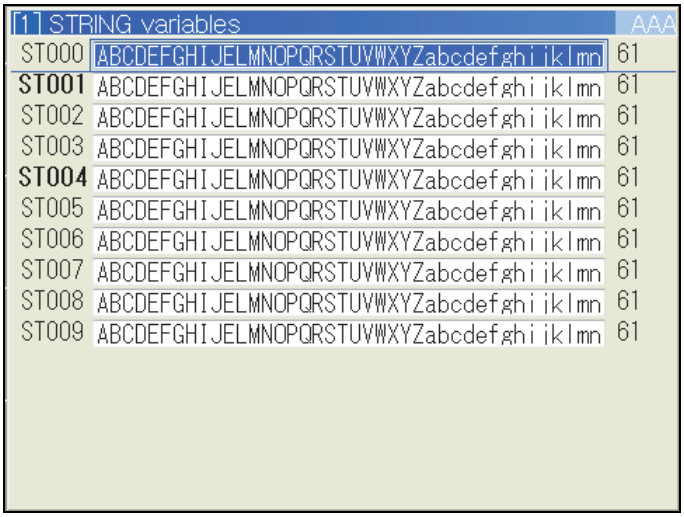
5.7.10 字符串变数监视器

1 **从变数监视器菜单当中选择 [10 字符串变数监视器]。**

≫显示如下所示监视器画面。（下图为全画面显示）

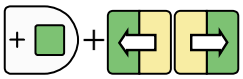
显示所有[字符串]变数的状态。

右侧数值为文字数。



根据变数状态将显示以下内容。

PLC 程序中未使用	文字：一般显示	PLC 程序中使用	文字：强调显示
	ST002		ST001



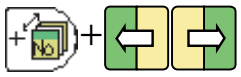
2 **按下［动作可能］＋［左右］。**

≫移动(1 文字分)开始显示位置。

如以下显示状态时，

按下［动作可能］＋［右］键。

将变更为如下显示画面。



2 **按下［动作可能］＋［左右］。**

≫移动(1 页分)开始显示位置。

如以下显示状态时，

按下［动作可能］＋［右］键。

将变更为如下显示画面。

NOTE

6章 输入输出继电器一览

本章详细说明逻辑输入输出信号及物理输入输出信号和软件 PLC 的输入输出继电器编号的对应关系。

6.1 逻辑输入输出继电器

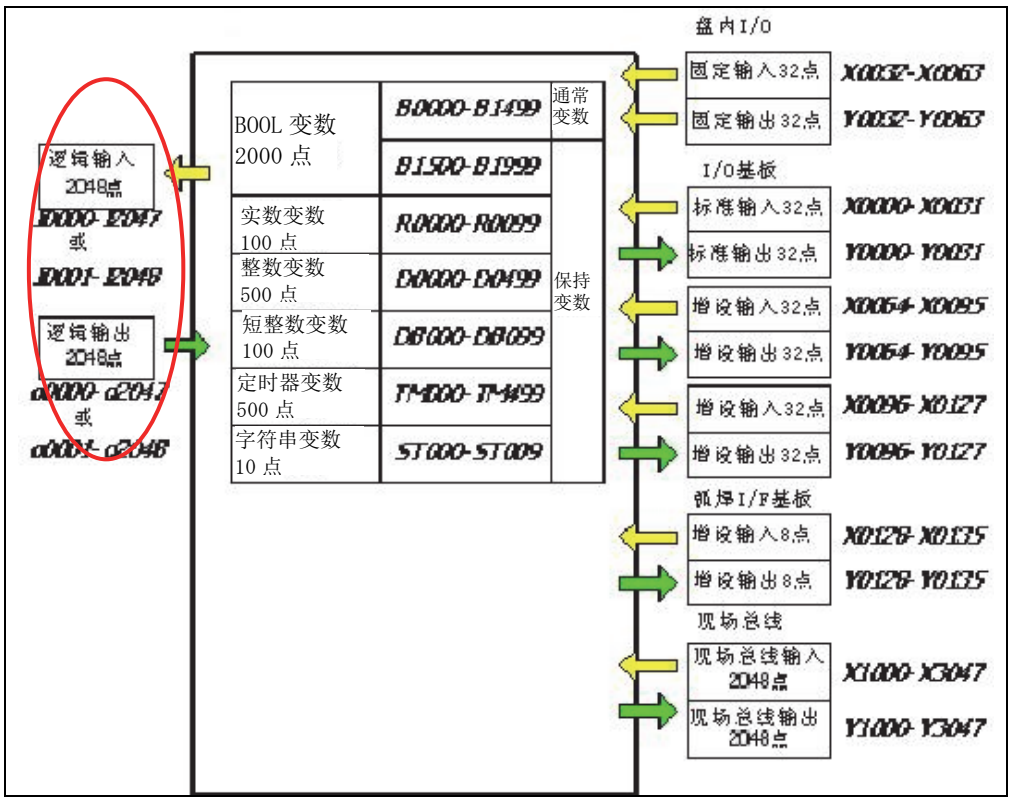


图 6.1.1 逻辑输入输出继电器

逻辑输入输出信号是指从软件 PLC 查看时的本控制装置侧的输入输出信号。
和实际在作业程序中记录的输入输出信号一样，逻辑输入继电器也是采用的是以 IO 开头的记述方法。

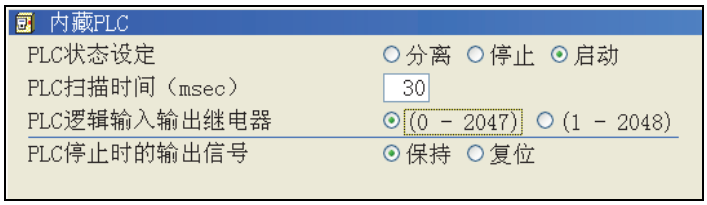
6.1.1 继电器编号

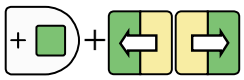
逻辑输入输出继电器编号可以在常数菜单进行设定。
从以下 2 个种类当中进行选择。工厂出厂设定时为“0 - 2047”。

表 6.1.1 输入输出逻辑

选项	说明
0－2047	在 0～2047 的范围内设定继电器编号。 信号编号为 1～2048，因此 “逻辑输入继电器编号 ＝ 逻辑输入信号编号 － 1” “逻辑输出继电器编号 ＝ 逻辑输出信号编号 － 1”。
1 －2048	继电器编号和信号编号一样，在 1～2048 范围内设定。 “逻辑输入继电器编号 ＝ 逻辑输入信号编号” “逻辑输出继电器编号 ＝ 逻辑输出信号编号”。

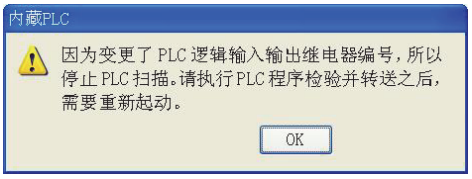
1 **在示教模式下，选择 f5＜常数设定＞－［1 控制环境］，从显示的菜单当中选择［4 内藏 PLC］。**
≫显示如下所示内藏 PLC 的相关设定菜单，将光标移动到 PLC 逻辑输入输出。





2 **同时按下［动作可能］＋［左右键］，选择 0-2047 或 1-2048。**

3 **按下 f12＜写入＞。**
≫PLC 状态设定为“启动”状态时，显示如下信息。



按下［Enter］键，结束设定画面。此时，PLC 状态设定为“停止”状态。
务必实施第 3 章所述的“程序校验”操作。

6.2 物理输入输出继电器

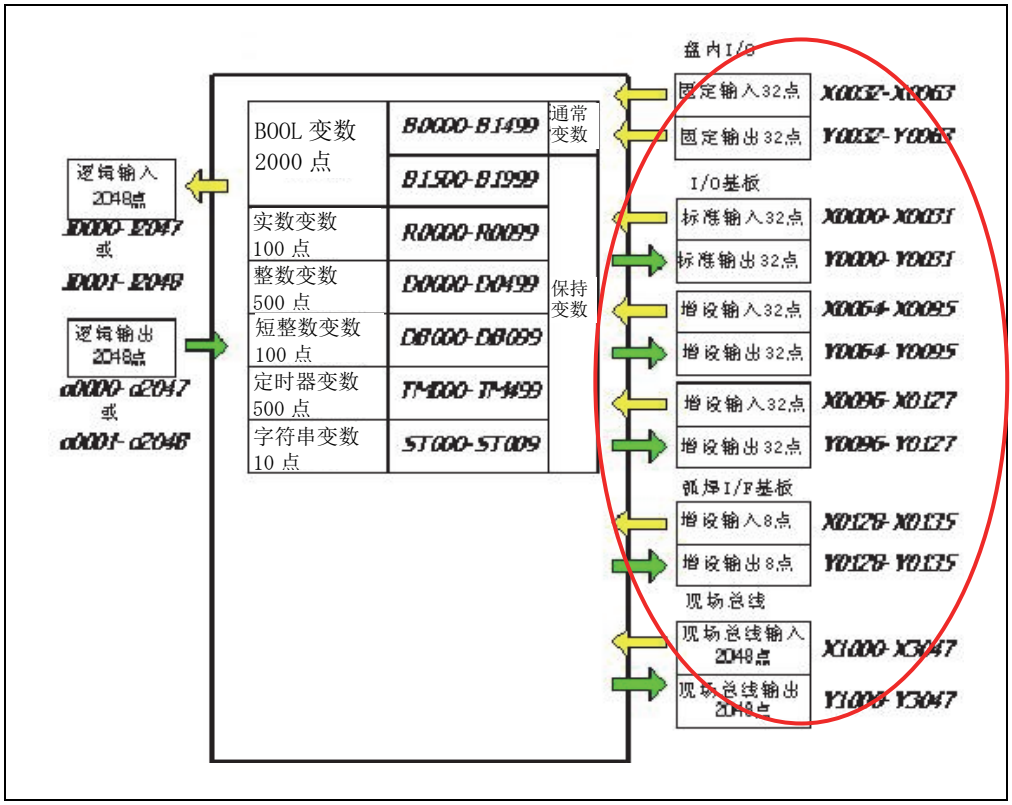


图 6.2.1　物理输入输出继电器

表 6.2.1　继电器编号与基板的关系

输入	
继电器编号	基板
X0000 – X0031	I/O 基板 1
X0032 – X0063	盘内输入（固定输入）
X0064 – X0095	I/O 基板 2
X0096 – X0127	I/O 基板 3
X0128 – X0135	电弧 IF 基板
X1000 – X1511	现场总线 CH1
X1512 – X2023	现场总线 CH2
X2024 – X2535	现场总线 CH3
X2536 – X3047	现场总线 CH4

出力	
继电器编号	基板
Y0000 – Y0031	I/O 基板 1
Y0032 – Y0063	盘内输出（固定输出）
Y0064 – Y0095	I/O 基板 2
Y0096 – Y0127	I/O 基板 3
Y0128 – Y0135	电弧 IF 基板
X1000 – X1511	现场总线 CH1
X1512 – X2023	现场总线 CH2
X2024 – X2535	现场总线 CH3
X2536 – X3047	现场总线 CH4

6.2.1 盘内 I/O(固定输入输出)

是伺服 ON/OFF 等的控制装置内部的控制所使用的输入输出信号。

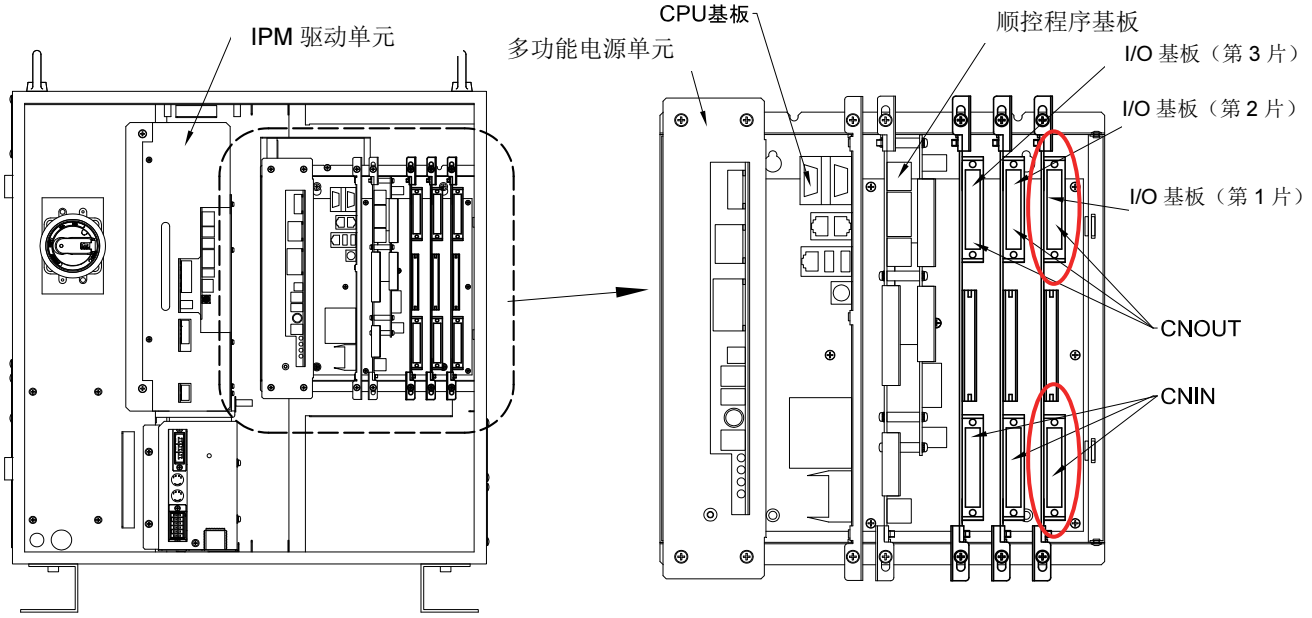
软件 PLC 中仅可参照固定输入输出信号。编制了输出到固定输入输出信号的程序时，会在编译时发生错误。

表 6.2.2 固定输入输出继电器编号一览

	固定输入 信号名称	继电器编号		固定输出 信号名称	继电器编号
1	运转准备入	***X0032***	1	运转准备入	***Y0032***
2	G-STOP	***X0033***	2	运转准备 ON 要求	***Y0033***
3	起动 1	***X0034***	3	起动中指示灯 1	***Y0034***
4	起动 2	***X0035***	4	起动中指示灯 2	***Y0035***
5	起动 3	***X0036***	5	起动中指示灯 3	***Y0036***
6	起动 4	***X0037***	6	起动中指示灯 4	***Y0037***
7	停止	***X0038***	7	暂停指示灯	***Y0038***
8	再生模式	***X0039***	8	TP Enable Release	***Y0039***
9	栅网开关	***X0040***	9	运转准备 ON 许可	***Y0040***
10	—	***X0041***	10	磁性开关 ON 许可	***Y0041***
11	高速示教	***X0042***	11	内部/外部选择	***Y0042***
12	P1 正常	***X0043***	12	WPS 紧急停止控制	***Y0043***
13	外部紧急停止输入	***X0044***	13	CPU 异常输出	***Y0044***
14	紧急停止输入	***X0045***	14	TP 模式选择	***Y0045***
15	安全插头	***X0046***	15	外部运转准备 ON 请求	***Y0046***
16	运转准备入确认	***X0047***	16	—	***Y0047***
17	TP 紧急停止	***X0048***	—	—	—
18	示教模式	***X0049***	—	—	—
19	—	***X0050***	—	—	—
20	启动开关 ON	***X0051***	—	—	—
21	冲击传感器	***X0052***	—	—	—
22	CRON	***X0053***	—	—	—
23	伺服 ON	***X0054***	—	—	—
24	伺服 ENABLE	***X0055***	—	—	—
25	—	***X0056***	—	—	—
26	—	***X0057***	—	—	—
27	—	***X0058***	—	—	—
28	磁性开关 ON	***X0059***	—	—	—
29	—	***X0060***	—	—	—
30	熔敷检测	***X0061***	—	—	—
31	不一致检测	***X0062***	—	—	—
32	—	***X0063***	—	—	—
33	不一致(GSTOP)	—	—	—	—
34	不一致(示教/再生)	—	—	—	—
35	不一致(MAT-SW)	—	—	—	—
36	不一致(高速示教)	—	—	—	—
37	不一致(外部 ES)	—	—	—	—
38	不一致(ES)	—	—	—	—
39	不一致(SP)	—	—	—	—
40	不一致(TP-ES)	—	—	—	—
41	不一致(TP ENABLE)	—	—	—	—
42	不一致(CRON)	—	—	—	—
43	—	—	—	—	—
44	—	—	—	—	—
45	—	—	—	—	—
46	—	—	—	—	—
47	—	—	—	—	—
48	—	—	—	—	—

6.2.2 I/O 基板 1（选购件）

是作为选购件装备的 I/O 基板 1 的连接器 CNIN（输入）及 CNOUT（输出）的输入输出信号。



＜正面图（内部）＞　　＜机架单元放大图＞

图 6.2.2　I/O 基板输入输出连接器的位置

表 6.2.3 [I/O 基板 1]继电器编号一览

输入连接器 CNIN		输出连接器 CNOUT	
连接器针编号	继电器编号	连接器针编号	继电器编号
1	***X0000***	1	***Y0000***
2	***X0001***	2	***Y0001***
3	***X0002***	3	***Y0002***
4	***X0003***	4	***Y0003***
5	***X0004***	5	***Y0004***
6	***X0005***	6	***Y0005***
7	***X0006***	7	***Y0006***
8	***X0007***	8	***Y0007***
9	-	9	-
10	***X0008***	10	***Y0008***
11	***X0009***	11	***Y0009***
12	***X0010***	12	***Y0010***
13	***X0011***	13	***Y0011***
14	***X0012***	14	***Y0012***
15	***X0013***	15	***Y0013***
16	***X0014***	16	***Y0014***
17	***X0015***	17	***Y0015***
18	-	18	-
19	***X0016***	19	***Y0016***
20	***X0017***	20	***Y0017***
21	***X0018***	21	***Y0018***
22	***X0019***	22	***Y0019***
23	***X0020***	23	***Y0020***
24	***X0021***	24	***Y0021***
25	***X0022***	25	***Y0022***
26	***X0023***	26	***Y0023***
27	-	27	-
28	-	28	-
29	-	29	-
30	-	30	-
31	-	31	-
32	-	32	-
33	***X0024***	33	***Y0024***
34	***X0025***	34	***Y0025***
35	***X0026***	35	***Y0026***
36	***X0027***	36	***Y0027***
37	***X0028***	37	***Y0028***
38	***X0029***	38	***Y0029***
39	***X0030***	39	***Y0030***
40	***X0031***	40	***Y0031***
41	-	41	-
42	-	42	-
43	-	43	-
44	-	44	-
45	-	45	-
46	-	46	-
47	-	47	-
48	-	48	-
49	-	49	-
50	-	50	-

I/O 基板 1、2、3 为选购件。

6.2.3 I/O 基板 2、3（选购件）

作为选购件装备的第 2 片往后的 I/O 基板的输入输出信号，对应于下述的继电器编号。

表 6.2.4 [I/O 基板 2] [I/O 基板 3]继电器编号一览

基板	继电器编号	种类
I/O 基板 2	***X0064-X0095***	**输入**
	Y0064-Y0095	**输出**
I/O 基板 3	***X0096-X0127***	**输入**
	Y0096-Y0127	**输出**

6.2.4 电弧 I/F 板（选购件）

以选购件形式装备的电焊 I/F 板的输入输出信号如下所示。

表 6.2.5 [电焊 I/F 板]继电器编号一览

输入连接器 TBIN		输出连接器 TBOUT	
连接器针编号	继电器编号	连接器针编号	继电器编号
1	***X00128***	1	**-**
2	***X00129***	2	**-**
3	***X00130***	3	***Y00128***
4	***X00131***	4	***Y00129***
5	**-**	5	***Y00130***
6	***X00132***	6	***Y00131***
7	***X00133***	7	**-**
8	***X00134***	8	***Y00132***
9	***X00135***	9	***Y00133***
10	**-**	10	***Y00134***
11	**-**	11	***Y00135***
12	**-**	12	**-**
13	**-**	**-**	**-**
14	**-**	**-**	**-**

6.2.5 现场总线输入输出（选购件）

现场总线输入输出区域是以选购件形式装备的 DeviceNet、Profibus、RIO 等的现场总线专用的输入输出信号区域。现场总线最多可以安装 4 个通道，例如对上位控制器用分配为从属通道、对低位控制装置用分配为主通道等，可以分别改变各通道的用途进行分配。

使用复数个通道时，各通道所使用的输入输出信号区域如图 6.2.3 所示。各通道的开始地址（最小的输入输出信号的编号）是固定的，从该处设定使用的点数。

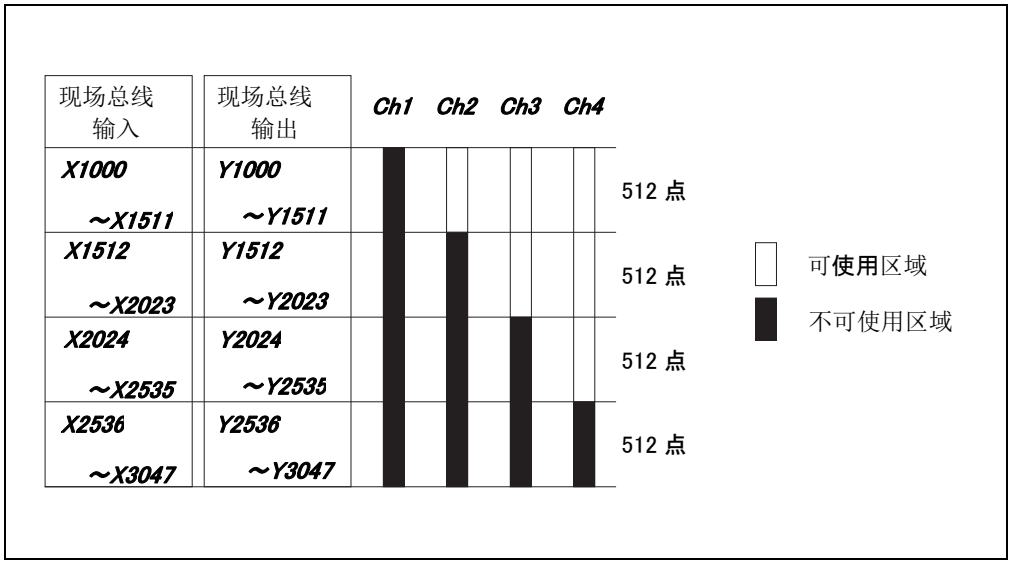


图 6.2.3 现场总线输入输出

例如想要在 DeviceNet 同时使用主 / 从两个通道时，按如下方式分配，分别可以利用 1024 点。

表 6.2.6 同时使用 2 个通道的设定示例

通道编号	设定	点数	输入	输出
Ch1	DeviceNet / 从	1024 点	X1000-X2023	Y1000-Y2023
Ch2	无效	0	-	-
Ch3	DeviceNet / 主	1024 点	X2024-X3047	Y2024-Y3047
Ch4	无效	0	-	-

详情敬请参见选购件操作说明书如“DeviceNet 功能”等现场总线相关的操作说明书。

7章 指令一览

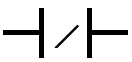
本章对于软件 PLC 可以使用的基本命令和应用命令（LD 块）的详情进行描述。

7.1 基本命令

7.1.1 A 接点

接点的右侧连接线的状态是接点左侧的连接线和分配给接点的变数之间的逻辑积。

7.1.2 B 接点



接点的右侧连接线的状态是接点左侧的连接线和分配给接点的变数的反转值之间的逻辑积。

7.1.3 上升沿接点

接点的右侧连接线的状态是仅在左侧连接线为 TRUE、且 BOOL 型变数从 FALSE 转为 TRUE 的上升沿时才成为 TRUE。除此以外为 FALSE。

7.1.4 下降沿接点

接点的右侧连接线的状态是仅在左侧连接线为 TRUE、且 BOOL 型变数从 TRUE 转为 FALSE 的下降沿时才为 TRUE。除此以外为 FALSE。

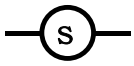
7.1.5 线圈

左侧连接线的状态用于代入分配给线圈的变数。
左侧连接线的状态直接传递给右侧连接线。右侧连接线通常连接到右母线。

7.1.6 反转线圈

左侧连接线的反转状态用于代入分配给线圈的变数。
左侧连接线的状态直接传递给右侧连接线。右侧连接线通常连接到右母线。

7.1.7 设置线圈

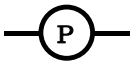


当左侧连接线的状态为 TRUE 时，分配变数的值被设定成 TRUE。输出变数的这个状态一直延续到通过复位线圈反转为止。
左侧连接线的状态直接传递给右侧连接线。右侧连接线通常连接到右母线。

7.1.8 复位线圈

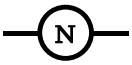
当左侧连接线的状态为 TRUE 时，分配变数的值被复位成 FALSE。输出变数的这个状态一直延续到通过设置线圈反转为止。
左侧连接线的状态直接传递给右侧连接线。右侧连接线通常连接到右母线。

7.1.9 带上升沿边缘检测的线圈



当左侧连接线的状态处于从 FALSE 转为 TRUE 的上升沿时，分配变数被设置成 TRUE。除此以外时全部复位成 FALSE。
左侧连接线的状态直接传递给右侧连接线。右侧连接线通常连接到右母线。

7.1.10 带下降沿边缘检测的线圈



当左侧连接线的状态处于从 TRUE 转为 FALSE 的下降沿时，分配变数被设置成 TRUE。除此以外时全部复位成 FALSE。
左侧连接线的状态直接传递给右侧连接线。右侧连接线通常连接到右母线。

7.1.11 转移

转移符号的左侧连接线的 BOOL 状态为 TRUE 时，程序跳转到对应的标签符号，从其后开始执行。

7.1.12 返回

RETURN 标签的左侧连接线的 BOOL 状态为 TRUE 时，程序不执行以后的程序，直接返回。

7.2 应用命令（LD 块）

本软件 PLC 中把比较命令或定时器命令等标准配备的应用命令，称之为“LD 块”，通过编号或名称进行输入。LD 块用四方的块（功能块）显示，总是具备输入和输出。

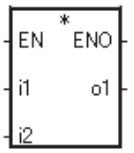
例如“定时器命令”是 LD 块 No.24＝TON（上升沿延迟时间）。功能块“TON”是和 ICS Triplex ISaGRAF 公司 ISaGRAF（ISaGRAF-PRO）标准支持的程序块相同的程序块。“LD 块 No.24”是考虑到示教的方便性，由本控制装置独立赋予的编号，是 ICS Triplex ISaGRAF 公司 ISaGRAF（ISaGRAF-PRO）中不存在的编号。

重要

投入电源时及启动 PLC 时，保存在 LD 块内的变数将会被清除。
因此，投入电源时及启动 PLC 时分配在 LD 块输出的变数有可能没有被保存。？？
ブロック内で保持される変数は電源投入時や PLC 起動時にクリアされます。
そのため、LD ブロックの出力に割り当てられる変数は電源投入時や PLC 起動時にクリアされ、保持されない場合があります。

下面，对于各命令进行详细描述。
本文已经获得 ICS Triplex ISaGRAF 公司的认可，几乎原样登载了 ISaGRAF（ISaGRAF-PRO）用户指南的内容。说明文字中还登载了较多的 LD 语言（梯形图）以外的语言（ST、IL 等），这些是用于说明各命令的具体动作的语言，不能通过本控制装置的悬式示教作业操纵按钮台进行显示、编辑。通过 LD 语言（梯形图）以外的语言进行编程时，请使用 ISaGRAF 工作台。

7.2.1 No.1；*（乘法运算）

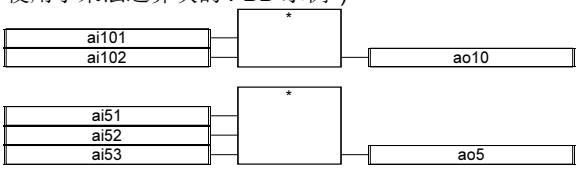


注意: 此命令的输入数量可改成 3 个以上。

自变数:
输入 **i*** 整数、实数型 (全部输入为相同类型)
输出 **o1** 整数、实数型 输入的带符号乘法运算
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

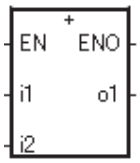
说明:
复数个整数、实数型变数的乘法运算

(* 使用了乘法运算块的 FBD 示例*)




```
(* 等价的 ST 语言: *)
ao10 := ai101 * ai102;
ao5 := (ai51 * ai52) * ai53;

(* 等价的 IL 语言: *)
LD      ai101
MUL     ai102
ST      ao10
LD      ai51
MUL     ai52
MUL     ai53
ST      ao5
```

7.2.2 No.2；+（加法运算）

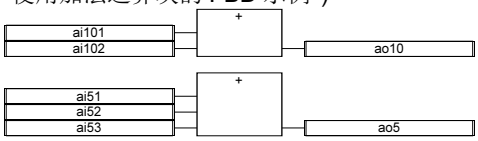


注意: 此命令的输入数量可改成 3 个以上。

自变数:
输入 **i*** 整数、实数型 (全部输入为相同类型)
输出 **o1** 整数、实数型 全部输入的带符号加法运算
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
复数的整数、实数型变数的带符号加法运算

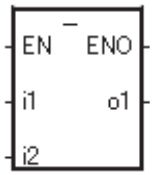
(* 使用加法运算块的 FBD 示例*)




```
(* 等价的 ST 语言: *)
ao10 := ai101 + ai102;
ao5 := (ai51 + ai52) + ai53;
```

```
(* 等价的 IL 语言: *)
LD      ai101
ADD     ai102
ST      ao10
LD      ai51
ADD     ai52
ADD     ai53
ST      ao5
```

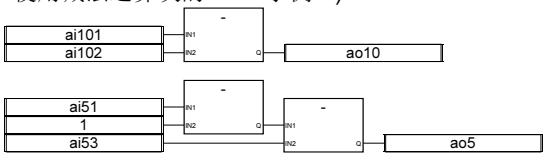
7.2.3 No.3；-（减法运算）



自变数:
输入 **i1,i2** 整数、实数型 (IN1 和 IN2 为相同类型)
输出 **o1** 整数、实数型 减法运算结果 (IN1 - IN2)
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
2 个整数、实数型变数的减法运算（第 1 个－第 2 个）。

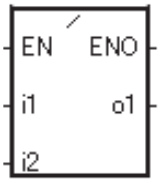
(* 使用减法运算块的 FBD 示例 *)



(* 等价的 ST 语言: *)
```
ao10 := ai101 - ai102;
ao5 := (ai51 - 1) - ai53;
```

(* 等价的 IL 语言: *)
```
LD      ai101
SUB     ai102
ST      ao10
LD      ai51
SUB     1
SUB     ai53
ST      ao5
```

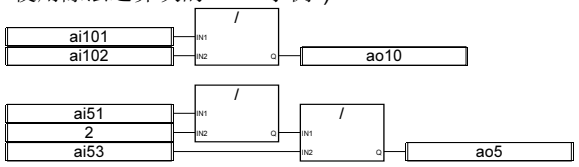
7.2.4 No.4；/（除法运算）



自变数:
输入 **i1,i2**　　整数、实数型　　除数不得为 0 。(IN1 和 IN2 为相同类型)
输出 **o1**　　整数、实数型　　带符号的除法运算（IN1 / IN2）
EN　仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
2 个整数、实数型变数的除法运算（第 1 个 / 第 2 个）

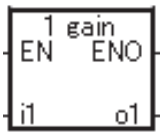
(* 使用除法运算块的 FBD 示例*)



(* 等价的 ST 语言: *)
```
ao10 := ai101 / ai102;
ao5 := (ai5 / 2) / ai53;
```

(* 等价的 IL 语言: *)
```
LD      ai101
DIV     ai102
ST      ao10
LD      ai51
DIV     2
DIV     ai53
ST      ao5
```

7.2.5 No.5；1 gain（代入）

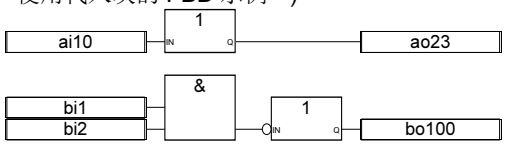


自变数:
输入 **i1**　　所有类型
输出 **o1**　　所有类型
EN　仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
将某个变数代入别的变数。

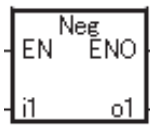
此块是将输入和输出直接连接的块。

(* 使用代入块的 FBD 示例 *)




```
(* 等价的 ST 语言: *)
ao23 := ai10;
bo100 := NOT (bi1 AND bi2);

(*等价的 IL 语言: *)
LD      ai10
ST      ao23
LD      bi1
AND     bi2
LDN bo100
```

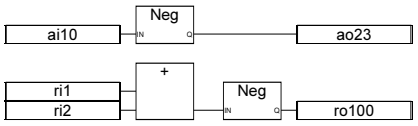
7.2.6 No.6；Neg（整数的符号反转）



自变数:
输入 **i1**　　整数、实数型　　输入、输出参数为相同类型
输出 **o1**　　整数、实数型　　输入、输出参数为相同类型
EN　仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
变数的反转（符号反转）的代入

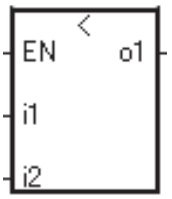
(* 使用 Negation 块的 FBD 示例 *)




```
(* 等价的 ST 语言: *)
ao23 := - (ai10);
ro100 := - (ri1 + ri2);
(*等价的 IL 语言: *)
LD      ai10
```

```
MUL     -1
ST      ao23
LD      ri1
ADD     ri2
MUL     -1.0
LD      ro100
```

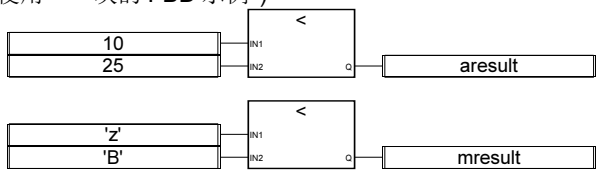
7.2.7 No.7；<（小于）



自变数:
输入 **i1,i2**　　整数型、实数型、定时器型、字符串型　　2 个输入为同一类型
输出 **o1**　　BOOL 型　　如果 I1 < I2 则为 TRUE
EN　仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。

说明:
比较某个变数是否比别的变数小。

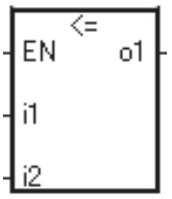
(*使用"< " 块的 FBD 示例*)




```
(* 等价的 ST 语言: *)
aresult := (10 < 25); (* aresult 为 TRUE *)
mresult := ('z' < 'B'); (* mresult 为 FALSE *)
```

```
(* 等价的 IL 语言: *)
LD      10
LT      25
ST      aresult
LD      'z'
LT      'B'
ST      mresult
```

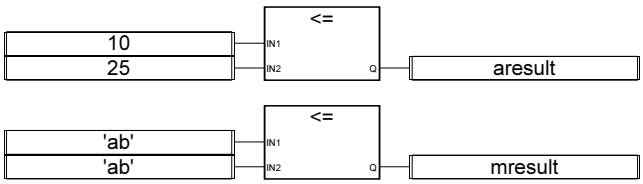
7.2.8 No.8；<=（小于或等于）



自变数:
输入 **i1,i2**　　整数型、实数型、字符串型　　2 个输入为相同类型
输出 **o1**　　BOOL 型　　如果 I1 <= I2 则为 TRUE
EN　仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。

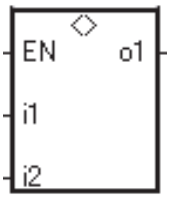
说明:
比较某个变数是否小于或等于别的变数。

(*使用"<= " 块的 FBD 示例*)




```
(* 等价的 ST 语言: *)
aresult := (10 <= 25); (* aresult 为 TRUE *)
mresult := ('ab' <= 'ab'); (* mresult 为 TRUE *)

(* 等价的 IL 语言: *)
LD      10
LE      25
ST      aresult
LD      'ab'
LE      'ab'
ST      mresult
```

7.2.9 No.9；<>（不相等）



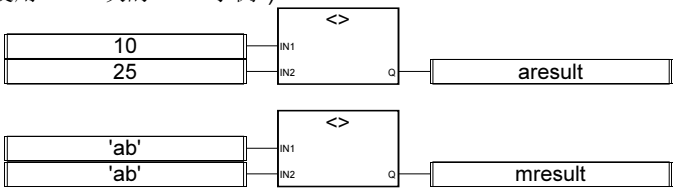
自变数:

输入 **i1,i2**	整数型、实数型、字符串型	2 个输入为相同类型
输出 **o1**	BOOL 型	如果 I1 <> I2 则为 TRUE

EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。

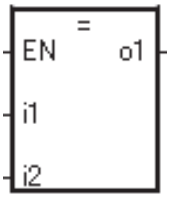
说明:
比较某个变数是否和别的变数不相等。

(*使用"<> " 块的 FBD 示例*)




```
(* 等价的 ST 语言: *)
aresult := (10 <> 25); (* aresult 为 TRUE *)
mresult := ('ab' <> 'ab'); (* mresult 为 FALSE *)

(* 等价的 IL 语言: *)
LD      10
NE      25
ST      aresult
LD      'ab'
NE      'ab'
ST      mresult
```

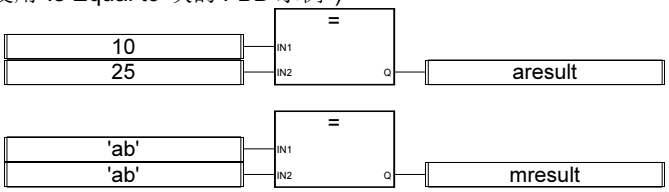
7.2.10 No.10；=（相等）



自变数:
输入 **i1,i2**　　整数型、实数型、字符串型　　2 个输入为相同类型
输出 **o1**　　BOOL 型　　如果 IN1 = IN2 则为 TRUE
EN　仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。

说明:
比较某个变数是否和别的变数相等。

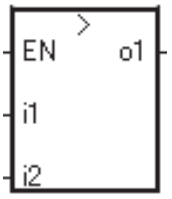
(*使用"Is Equal to"块的 FBD 示例*)



(* 等价的 ST 语言: *)
aresult := (10 = 25); (* aresult 为 FALSE *)
mresult := ('ab' = 'ab'); (* mresult 为 TRUE *)

(* 等价的 IL 语言: *)
```
LD      10
EQ      25
ST      aresult
LD      'ab'
EQ      'ab'
ST      mresult
```

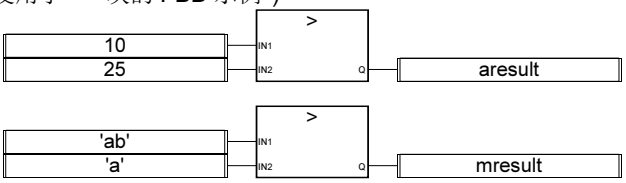
7.2.11 No.11；>（大于）



自变数:
输入 **i1,i2**　　整数型、实数型、定时器型、字符串型　　2 个输入为相同类型
输出 **o1**　　BOOL 型　　如果 IN1 > IN2 则为 TRUE
EN　仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。

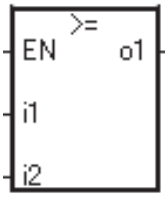
说明:
比较某个变数是否大于别的变数。

(*使用了">" 块的 FBD 示例*)




```
(* 等价的 ST 语言: *)
aresult := (10 > 25); (* aresult 为 FALSE *)
mresult := ('ab' > 'a'); (* mresult 为 TRUE *)

(* 等价的 IL 语言: *)
LD      10
GT      25
ST      aresult
LD      'ab'
GT      'a'
ST      mresult
```

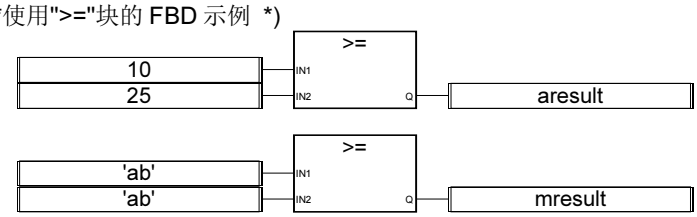
7.2.12 No.12；>=（大于或等于）



自变数:
输入 **i1,i2**　　整数型、实数型、字符串型　　2 个输入为相同类型
输出 **o1**　　BOOL 型　　如果 IN1 >= IN2 则为 TRUE
EN　仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。

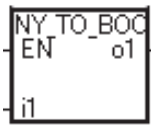
说明:
比较某个变数是否大于或等于别的变数。

(*使用">="块的 FBD 示例 *)




```
(* 等价的 ST 语言: *)
aresult := (10 >= 25); (* aresult 为 FALSE *)
mresult := ('ab' >= 'ab'); (* mresult 为 TRUE *)

(* 等价的 IL 语言: *)
LD      10
GE      25
ST      aresult
LD      'ab'
GE      'ab'
ST      mresult
```

7.2.13 No.13；ANY_TO_BOOL（转换为 BOOL 型）



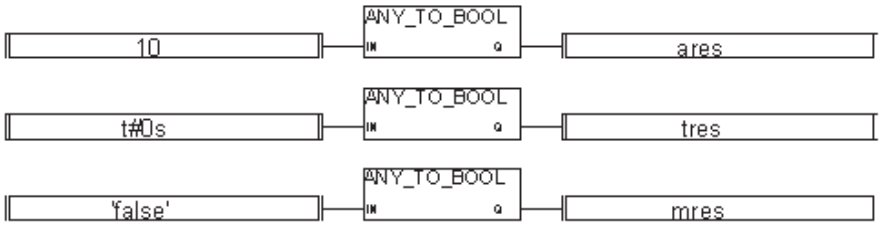
自变数:

输入 **i1**	ANY	全部的非整数值
输出 **o1**	BOOL 型	TRUE： 0 以外的数值
		FALSE： 0
		TRUE： 'TRUE' 字符串
		FALSE： 'FALSE' 字符串

EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。

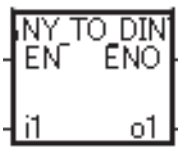
说明:
将变数转换为 BOOL 型。

(*使用 BOOL 型转换块的 FBD 示例*)




```
(* 等价的 ST 语言: *)
ares := ANY_TO_BOOL (10);          (* ares 为 TRUE *)
tres := ANY_TO_BOOL (t#0s);        (* tres 为 FALSE *)
mres := ANY_TO_BOOL ('FALSE');     (* mres 为 FALSE *)

(* 等价的 IL 语言: *)
LD      10
ANY_TO_BOOL
ST      ares
LD      t#0s
ANY_TO_BOOL
ST      tres
LD      'FALSE'
ANY_TO_BOOL
ST      mres
```

7.2.14 No.14；ANY_TO_DINT（转换为整数型）



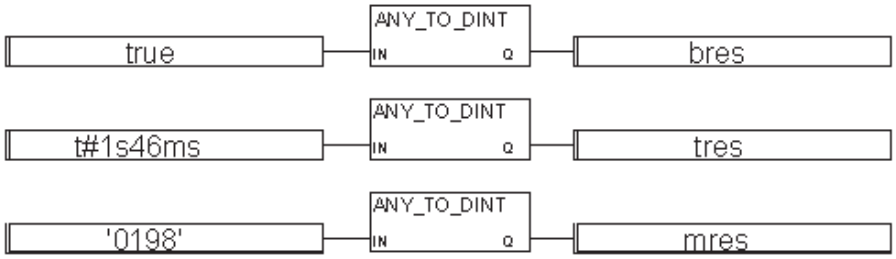
自变数:

输入 **i1**	ANY	全部的非整数值
输出 **o1**	整数型	0 ： IN 为 FALSE 时
		1 ： IN 为 TRUE 时
		定时器的ｍｓ数值
		实数变数的整数部分
		字符串的１０进制数表述

EN　仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

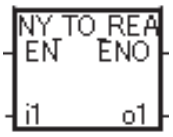
说明:
将变数转换为整数型。

(*使用整数型转换块的 FBD 示例 *)




```
(* 等价的 ST 语言: *)
bres := ANY_TO_DINT (TRUE);       (* bres 为 1 *)
tres := ANY_TO_DINT (t#1s46ms);   (* tres 为 1046 *)
mres := ANY_TO_DINT ('0198');     (* mres 为 198 *)

(* 等价的 IL 语言: *)
LD      TRUE
ANY_TO_DINT
ST      bres
LD      t#1s46ms
ANY_TO_DINT
ST      tres
LD      '0198'
ANY_TO_DINT
ST      mres
```

7.2.15　No.15；ANY_TO_REAL（转换为实数型）



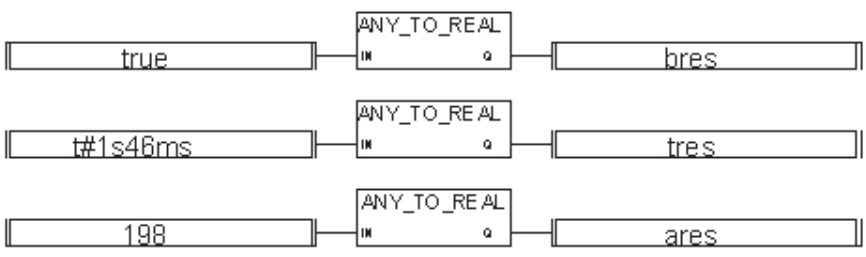
自变数:

输入 **i1**	ANY	全部的非整数值
输出 **o1**	实数型	0.0 ： IN 为 FALSE
		1.0 ： IN 为 TRUE
		定时器的毫秒级单位的数值
		和整数型相同的数值

EN　仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
将变数转换为实数型。

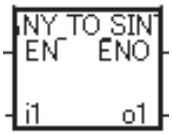
(*使用实数型转换块的 FBD 示例 *)




```
(* 等价的 ST 语言: *)
bres := ANY_TO_REAL (TRUE);       (* bres 为 1.0 *)
```

```
tres := ANY_TO_REAL (t#1s46ms);   (* tres 为 1046.0 *)
ares := ANY_TO_REAL (198);        (* ares 为 198.0 *)

(* 等价的 IL 语言: *)
LD      TRUE
ANY_TO_REAL
ST      bres
LD      t#1s46ms
ANY_TO_REAL
ST      tres
LD      198
ANY_TO_REAL
ST      ares
```

7.2.16 No.16；ANY_TO_SINT（转换为短整数型）



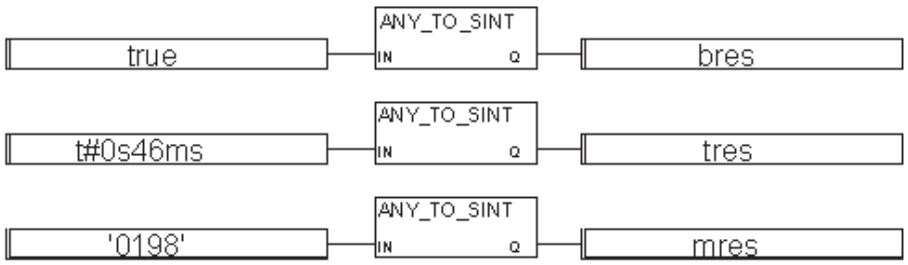
自变数:

输入 **i1**	ANY	全部的非整数值
输出 **o1**	实数型	IN 为 FALSE 时为 0 / IN 为 TRUE 时为 1 定时器的毫秒级单位的数值 实数型时为整数部分 字符串表示的 10 进制数

EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

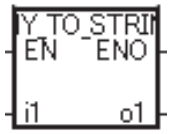
说明:
将变数转换为短整数型。

(使用了"转换为短整数"运算符的 FBD 的示例 *)




```
(* 等价的 ST 语言: *)
bres := ANY_TO_SINT (TRUE);       (* bres 为 1 *)
tres := ANY_TO_SINT (t#0s46ms); (* tres 为 46 *)
mres := ANY_TO_SINT ('0198');     (* mres 为 198 *)
```

7.2.17 No.17；ANY_TO_STRING（转换为字符串型）

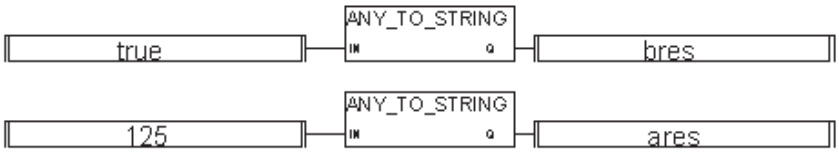


自变数:

输入 **i1**	any	全部的非字符串值
输出 **o1**	字符串型	I1 为 BOOL 型时'FALSE' 或'TRUE'
		I1 为整数、实数型时 10 进制数表述

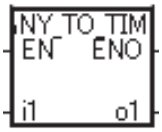
说明:
将变数转换为字符串型。

(*使用了可变长度字符串型转换块的 FBD 示例 *)




```
 (* 等价的 ST 语言: *)
bres := ANY_TO_STRING (TRUE); (* bres 为 'TRUE' *)
ares := ANY_TO_STRING (125); (* ares 为 '125' *)

(* 等价的 IL 语言: *)
LD    TRUE
ANY_TO_STRING
ST    bres
LD    125
ANY_TO_STRING
ST    ares
```

7.2.18 No.18；ANY_TO_TIME（转换为定时器型）



自变数:

输入 **i1**	any	全部的非定时器型值
		毫秒单位的数值
输出 **o1**	定时器型	以 I1 表述的定时器值
		I1 为实数时为其整数部分

EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

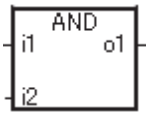
说明:
将整数型、实数型变数转换为定时器值。

(*使用定时器型转换块的 FBD 示例*)


```
 (* 等价的 ST 语言: *)
ares := ANY_TO_TIME (1256);        (* ares := t#1s256ms *)
```

```
rres := ANY_TO_TIME (1256.3);       (*rres := t#1s256ms *)

(* 等价的 IL 语言: *)
LD      1256
ANY_TO_TIME
ST      ares
LD      1256.3
ANY_TO_TIME
ST      rres
```

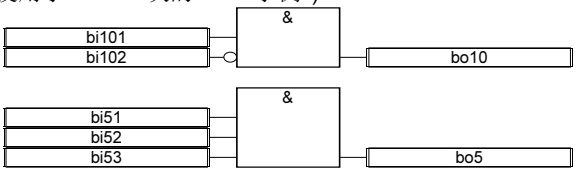
7.2.19 No.19；AND（BOOL 型 AND）



注意: 此命令的输入数量可改成 3 个以上。
自变数:
输入 **i1,i2**　　BOOL 型
输出 **o1**　　BOOL 型　输入参数的逻辑积

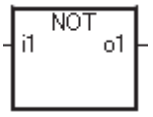
说明:
2 个以上的输入参数的逻辑积

(*使用了"AND" 块的 FBD 示例*)




```
(* 等价的 ST 语言: *)
bo10 := bi101 AND NOT (bi102);
bo5 := (bi51 AND bi52) AND bi53;

(* 等价的 IL 语言: *)
  LD      bi101       (*  现在结果  := bi101 *)
  ANDN    bi102       (*  现在结果  := bi101 AND not(bi102) *)
  ST      bo10        (* bo :=  现在结果  *)
  LD      bi51        (*  现在结果  := bi51;
  &       bi52        (*  现在结果  := bi51 AND bi52 *)
  &       bi53        (*  现在结果  := (bi51 AND bi52) AND bi53 *)
  ST      bo5         (* bo :=  现在结果  *)
```

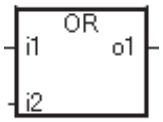
7.2.20 No.20；NOT（BOOL 型 NOT）



自变数:
输入 **i1**　　BOOL 型
输出 **o1**　　BOOL 型　I1 为 FALSE 时则为 TRUE、I1 为 TRUE 时则为 FALSE

说明:
返回 BOOL 型的否定

7.2.21 No.21；OR（BOOL 型 OR）

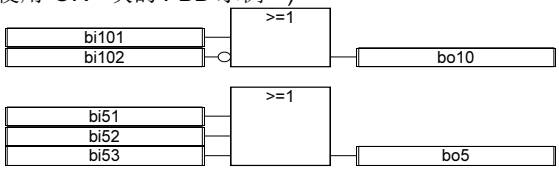


注意: 此命令的输入数量可改成 3 个以上。

自变数:
输入 **i1,i2**　　BOOL 型
输出 **o1**　　BOOL 型　输入参数的逻辑和

说明:
2 个以上的输入参数的逻辑和

(*使用"OR" 块的 FBD 示例 *)



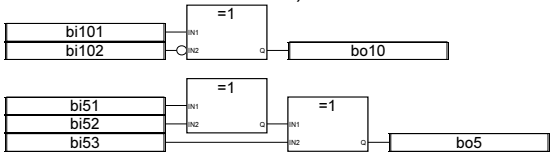

```
(* 等价的 ST 语言: *)
bo10 := bi101 OR NOT (bi102);
bo5 := (bi51 OR bi52) OR bi53;

(* 等价的 IL 语言: *)
LD      bi101
ORN     bi102
ST      bo10
LD      bi51
OR      bi52
OR      bi53
ST      bo5
```

7.2.22 No.22；XOR（BOOL 型 Excusive OR）

自变数:
输入 **i1,i2**　　BOOL 型
输出 **o1**　　BOOL 型　2 个 BOOL 输入的异或

说明:
2 个 BOOL 值的异或

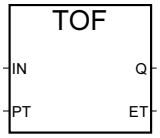
(*使用"XOR"块的 FBD 示例 *)




```
(* 等价的 ST 语言: *)
bo10 := bi101 XOR NOT (bi102);
bo5 := (bi51 XOR bi52) XOR bi53;

(* 等价的 IL 语言: *)
LD      bi101
```

```
XORN    bi102
ST      bo10
LD      bi51
XOR     bi52
XOR     bi53
ST      bo5
```

7.2.23 No.23；TOF（下降沿延迟时间）



自变数:

IN BOOL 型 检测到下降沿时，开始定时器值递增
检测到上升沿时，定时器值停止、复位

PT 定时器型 设定定时器值

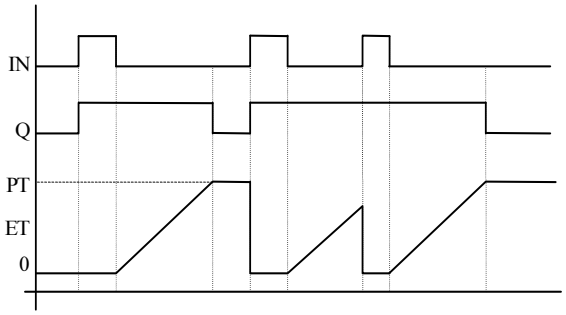
Q BOOL 型 TRUE 时，经过时间未到达设定定时器值

ET 定时器型 现在经过的定时器值

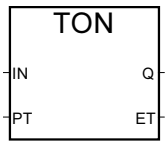
说明:

内部定时器值递增至指定的值（PT）为止

时间图:



7.2.24 No.24；TON（上升沿延迟时间）



自变数:

IN BOOL 型 检测到上升沿时开始定时器值的递增
检测到下降沿时，定时器值停止、复位

PT 定时器型 到时限设定值

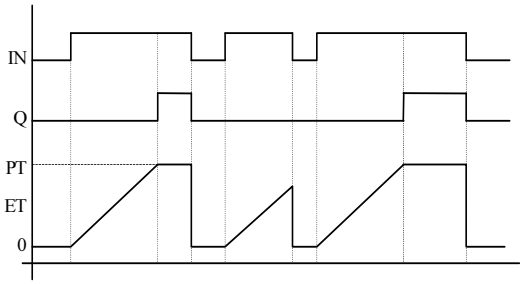
Q BOOL 型 TRUE 时，经过了时限设定值

ET 定时器型 现在经过的定时器值

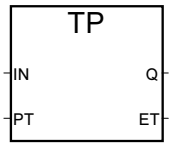
说明:

内部定时器值递增至指定的值（PT）为止

时间图:



7.2.25 No.25；TP（脉冲时机）



自变数:

IN BOOL 型 检测到上升沿时，开始定时器值的递增（如果未处于递增中）。
如果在 FALSE 状态下，定时器值经过了设定定时器值时，对定时器值进行复位。
定时器递增过程中，忽视 IN 的变化。

PT 定时器型 设定定时器值

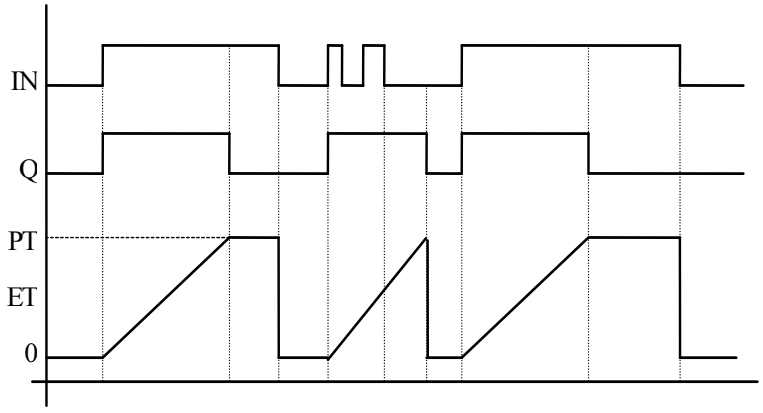
Q BOOL 型 TRUE 时，定时器为递增中

ET 定时器型 现在经过的定时器值

说明:

内部定时器值递增至指定的值（PT）为止

时间图:

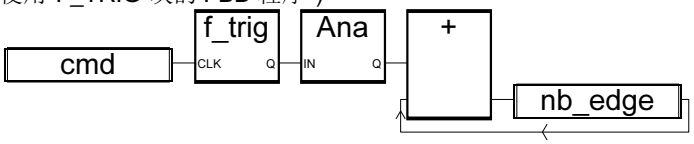


7.2.26 No.26；F_TRIG（下降沿边缘检测）

自变数:
CLK BOOL 型
Q BOOL 型 TRUE：CLK 从 TRUE 下降为 FALSE 时
FALSE：除此以外时

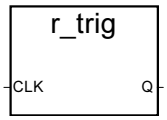
说明:
检测 BOOL 型变数的下降沿（和 1 次循环前的值进行比较）。

(*使用"F_TRIG"块的 FBD 程序*)




```
(*等价的 ST 语言: F_TRIG1 为 F_TRIG 块的实例 *)
F_TRIG1(cmd);
nb_edge := ANA(F_TRIG1.Q) + nb_edge;

(* 等价的 IL 语言: *)
LD      cmd
ST      F_TRIG1.clk
CAL     F_TRIG1
LD      F_TRIG1.Q
ANA
ADD     nb_edge
ST      nb_edge
```

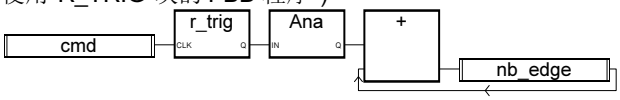
7.2.27 No.27；R_TRIG（上升沿边缘检测）



自变数:
CLK BOOL 型
Q BOOL 型 TRUE ： CLK 从 FALSE 上升为 TRUE 时
FALSE：除此以外时

说明:
检测 BOOL 型变数的上升沿（和 1 次循环前的值进行比较）。

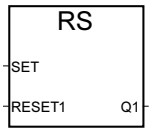
(*使用"R_TRIG"块的 FBD 程序*)




```
(*等价的 ST 语言: R_TRIG1 为 R_TRIG 块的实例 *)
R_TRIG1(cmd);
nb_edge := ANA(R_TRIG1.Q) + nb_edge;
```

```
(* 等价的 IL 语言: *)
LD      cmd
ST      R_TRIG1.clk
CAL     R_TRIG1
LD      R_TRIG1.Q
ANA
ADD     nb_edge
ST      nb_edge
```

7.2.28 No.28；RS（RS 双稳态电路）



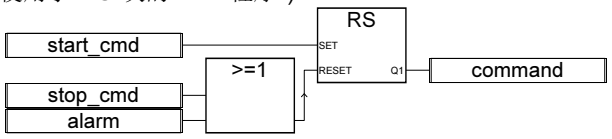
自变数:

SET	BOOL 型 TRUE 则将 Q1 设为 TRUE。
RESET1	BOOL 型 TRUE 时将 Q1 复位成 FALSE（优先）
Q1	BOOL 型 BOOL 型存储器状态

说明:
复位优先双稳定：参照下表

Set	Reset1	Q1	result Q1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

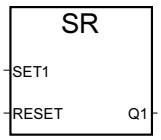
(*使用了"RS"块的 FBD 程序*)




```
(*等价的 ST 语言: RS1 为 RS 块的实例 *)
RS1(start_cmd, (stop_cmd OR alarm));
command := RS1.Q1;
```

```
(* 等价的 IL 语言: *)
LD      start_cmd
ST      RS1.set
LD      stop_cmd
OR      alarm
ST      RS1.reset1
CAL     RS1
LD      RS1.Q1
ST      command
```

7.2.29 No.29；SR（SR 双稳态电路）



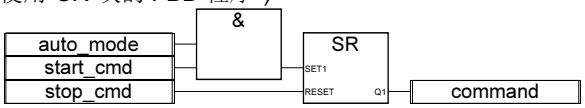
自变数:

SET1		BOOL 型 TRUE 则将 Q1 设为 TRUE (优先)
RESET	BOOL 型	TRUE 则将 Q1 复位成 FALSE。
Q1	BOOL 型	BOOL 型的存储器状态

说明:
设置优先双稳定：参照下表

Set1	Reset	Q1	result Q1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

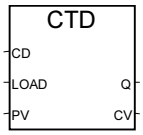
(*使用"SR"块的 FBD 程序*)




```
(*等价的 ST 语言: SR1 为 SR 块的实例 *)
SR1((auto_mode & start_cmd), stop_cmd);
command := SR1.Q1;
```

```
(* 等价的 IL 语言: *)
LD      auto_mode
AND     start_cmd
ST      SR1.set1
LD      stop_cmd
ST      SR1.reset
CAL     SR1
LD      SR1.Q1
ST      command
```

7.2.30 No.30；CTD（递减计数）



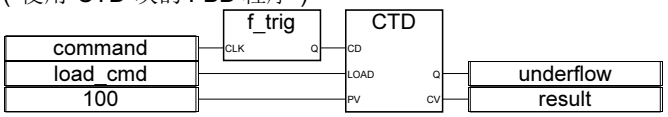
自变数:

CD	BOOL 型	计数输入 (CD 为 TRUE 时递减计数)
LOAD		BOOL 型 LOAD 指令 (优先) （LOAD 为 TRUE 时，CV = PV）
PV	整数型	计数器的初始值
Q	BOOL 型	下溢： CV = 0 时为 TRUE
CV	整数型	计数器结果

注意：
CTD 不检测输入 CD 的上升沿、下降沿。必须使用 "R_TRIG" 或 "F_TRIG" 制作脉冲计数器。

说明:
计数数值（整数型）从 PV 值开始，逐一减少至 0 为止。

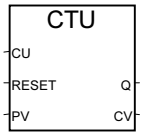
(*使用"CTD"块的 FBD 程序*)




```
 (*等价的 ST 语言: F_TRIG1 为 F_TRIG 块的实例，CTD1 为 CTD 块的实例*)
CTD1(F_TRIG1(command),load_cmd,100);
underflow := CTD1.Q;
result := CTD1.CV;
```

```
(* 等价的 IL 语言: *)
LD      command
ST      F_TRIG1.clk
CAL     F_TRIG1
LD      F_TRIG1.Q
ST      CTD1.cd
LD      load_cmd
ST      CTD1.load
LD      100
ST      CTD1.pv
CAL     CTD1
LD      CTD1.Q
ST      underflow
LD      CTD1.cv
ST      result
```

7.2.31 No.31；CTU（递增计数）



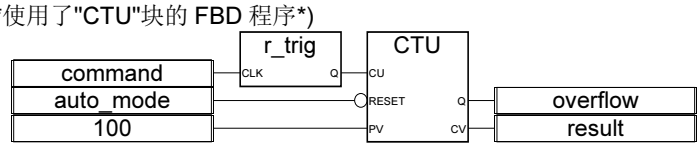
自变数:

CU	BOOL 型	计数输入（CU 为 TRUE 时，进行计数)
RESET	BOOL 型	复位指令（优先）
PV	整数型	最大计数值
Q	BOOL 型	上溢： CV = PV 时为 TRUE
CV	整数型	现在的计数结果

注意：
CTU 不检测输入 CU 的上升沿、下降沿。必须使用 "R_TRIG" 或 "F_TRIG" 制作脉冲计数器。

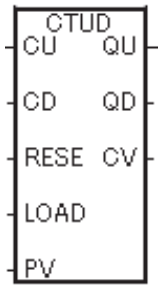
说明:
计数数值（整数型）从 0 开始逐一增加

(*使用了"CTU"块的 FBD 程序*)



```
(*等价的 ST 语言: R_TRIG1 为 R_TRIG 块的实例， CTU1 为 CTU 块的实例*)
CTU1(R_TRIG1(command),NOT(auto_mode),100);
overflow := CTU1.Q;
result := CTU1.CV;

(* 等价的 IL 语言: *)
LD      command
ST      R_TRIG1.clk
CAL     R_TRIG1
LD      R_TRIG1.Q
ST      CTU1.cu
LDN     auto_mode
ST      CTU1.reset
LD      100
ST      CTU1.pv
CAL     CTU1
LD      CTU1.Q
ST      overflow
LD      CTU1.cv
ST      result
```

7.2.32 No.32；CTUD（上下计数器）



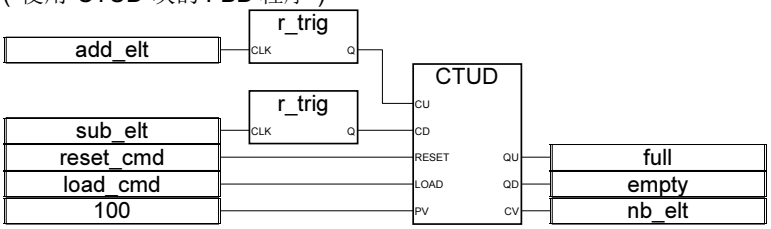
自变数:

CU	BOOL 型	递增计数器输入 (CU 为 TRUE 时计数)
CD	BOOL 型	递减计数器输入 (CD 为 TRUE 时计数)
RESET	BOOL 型	复位指令 (优先) (RESET 为 TRUE 时， CV = 0)
LOAD		BOOL 型 LOAD 指令(LOAD 为 TRUE 时, CV = PV)
PV	整数型	最大计数值
QU	BOOL 型	上溢： CV = PV 时为 TRUE
QD	BOOL 型	下溢： CV = 0 时为 TRUE
CV	整数型	计数结果

注意:
CTUD 不检测输入 CU、CD 的上升沿、下降沿。必须使用 "R_TRIG" 或 "F_TRIG" 制作脉冲计数器。

说明:
计数数（整数型）从 0 逐一增加至 PV 值为止，
或从当前值逐一减少至 0 为止。

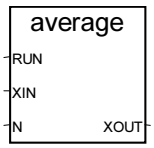
(*使用"CTUD"块的 FBD 程序*)




```
(*等价的 ST 语言: R_TRIG1 和 R_TRIG2 为 R_TRIG 块的实例， CTUD1 为 CTUD 块的实例*)
CTUD1(R_TRIG1(add_elt), R_TRIG2(sub_elt), reset_cmd, load_cmd,100);
full := CTUD1.QU;
empty := CTUD1.QD;
nb_elt := CTUD1.CV;
```

```
(* 等价的 IL 语言: *)
LD      add_elt
ST      R_TRIG1.clk
CAL     R_TRIG1
LD      R_TRIG1.Q
ST      CTUD1.cu
LD      sub_elt
ST      R_TRIG2.clk
CAL     R_TRIG2
LD      R_TRIG2.Q
ST      CTUD1.cd
LD      load_cmd
ST      CTUD1.load
LD      100
ST      CTUD1.pv
CAL     CTUD1
LD      CTUD1.QU
ST      full
LD      CTUD1.QD
ST      empty
LD      CTUD1.CV
ST      nb_elt
```

7.2.33 No.33；AVERAGE（N 样本的平均）



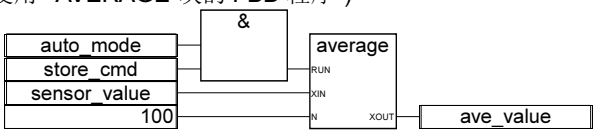
自变数:

RUN BOOL 型	TRUE：执行 / FALSE：复位	
XIN	实数型	实数值输入
N	整数型	应取平均的样本数（每次循环 1 个输入）
XOUT	实数型	XIN 值的样本个数（N 个）的移动平均值

说明:
每次循环载入 XIN，求取过去的样本值（样本数 N）的平均值。存储最新的 N 个数据。

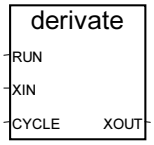
样本数**N最大为 128**。"**RUN**"指令为 **FALSE** 时（复位模式），输出值为和输入值相同的值。
存储的样本数量达到 N 后，最初存储的值将被删除。

(使用*"AVERAGE"块的 FBD 程序*)




```
(*等价的 ST 语言:AVERAGE1 为 AVERAGE 块的实例 *)
AVERAGE1((auto_mode & store_cmd), sensor_value, 100) ;
ave_value := AVERAGE1.XOUT;

(* 等价的 IL 语言: *)
LD      auto_mode
AND     store_cmd
ST      AVERAGE1.run
LD      sensor_value
ST      AVERAGE1.xin
LD      100
ST      AVERAGE1.N
CAL     AVERAGE1
LD      AVERAGE1.XOUT
ST      ave_value
```

7.2.34 No.34；DERIVATE（时间轴上的微分）



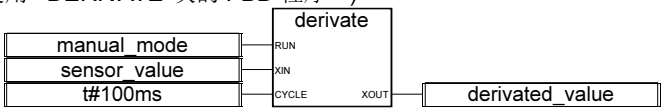
自变数:

RUN	BOOL 型	模式，TRUE=微分执行 / FALSE=复位
XIN	实数型	输入值
CYCLE	定时器型	采样周期
XOUT	实数型	微分结果

说明:
执行实数值的微分。

"**CYCLE**"参数值比ＩＳａＧＲＡＦ的循环时间短时，采样间隔将按照循环时间执行。

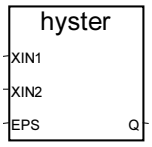
(使用*"DERIVATE"块的 FBD 程序 *)




```
(*等价的 ST 语言: DERIVATE1 为 DERIVATE 块的实例 *)
DERIVATE1(manual_mode, sensor_value, t#100ms);
derivated_value := DERIVATE1.XOUT;
```

```
(* 等价的 IL 语言: *)
LD      manual_mode
ST      DERIVATE1.run
LD      sensor_value
ST      DERIVATE1.XIN
LD      t#100ms
ST      DERIVATE1.CYCLE
CAL     DERIVATE1
LD      DERIVATE1.XOUT
ST      derivated_value
```

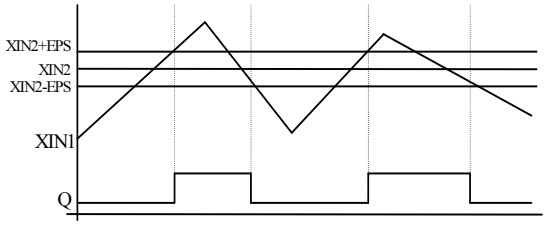
7.2.35 No.35；HYSTER（2 个实数值的滞后的 BOOL 型值）



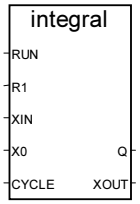
自变数:
XIN1 实数型　实数
XIN2 实数型　实数（测试 XIN1 是否超过 XIN2+EPS）
EPS 实数型　滞后值(＞ 0)
Q　BOOL 型　XIN1 超过 XIN2+EPS 且不低于 XIN2-EPS 时，为 TRUE

说明:
用于实数值的上限值的滞后

时间图例:



7.2.36 No.36；INTEGRAL（时间轴上的积分）

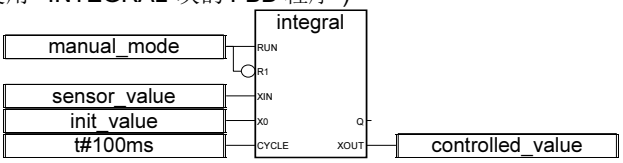


自变数:
RUN BOOL 型　模式、TRUE=积分 / FALSE=锁定
R1　BOOL 型　覆写复位
XIN　实数型　输入值
X0　实数型　初始值
CYCLE　定时器型　采样周期
Q　BOOL 型　R1 的反转
XOUT　实数型　积分结果

说明:
对实数值执行积分。

每隔指定的采样周期，对输入（XIN）的值乘上前一次采样后经过的时间(ms 单位)，加上前一次的输出后，从 XOUT 进行输出。
"**CYCLE**"参数值比ＩＳａＧＲＡＦ的循环时间短时，采样间隔按照循环时间执行。

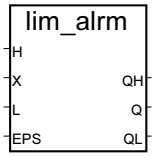
(使用*"INTEGRAL"块的 FBD 程序*)



(*等价的 ST 语言: INTEGRAL1 为 INTEGRAL 块的实例 *)
INTEGRAL1(manual_mode, NOT(manual_mode), sensor_value, init_value, t#100ms);
controlled_value := INTEGRAL1.XOUT;

(* 等价的 IL 语言: *)
```
LD      manual_mode
ST      INTEGRAL1.run
STN     INTEGRAL1.R1
LD      sensor_value
ST      INTEGRAL1.XIN
LD      init_value
ST      INTEGRAL1.X0
LD      t#100ms
ST      INTEGRAL1.CYCLE
CAL     INTEGRAL1
LD      INTEGRAL1.XOUT
ST      controlled_value
```

7.2.37 No.37；LIM_ALRM（具备滞后的上下限报警）

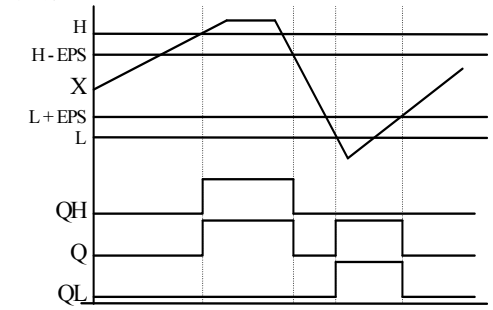


自变数:
H 实数型 上限设定值
X 实数型 输入值
L 实数型 下限设定值
EPS 实数型 滞后值（＞0）
QH BOOL 型 上限报警：X 超过上限值 H 时，为 TRUE
Q BOOL 型 报警：超出上下限时，为 TRUE
QL BOOL 型 下限报警：X 低于下限值 L 时，为 TRUE

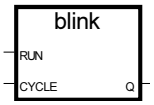
说明:
实数值的相对于上下限的滞后。

给滞后附加上限、下限限位。用于上限、下限的滞后量为 EPS 参数的一半。

时间图 ：



7.2.38 No.38；BLINK（闪烁 BOOL 型信号）

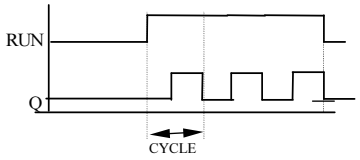


自变数:

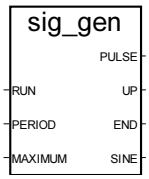
RUN	BOOL 型	模式、TRUE=闪烁 / FALSE=输出复位
CYCLE	定时器型	闪烁周期
Q	BOOL 型	闪烁的输出信号

说明:
生成闪烁信号。

时间图:



7.2.39 No.39；SIG_GEN（正弦信号发生器）



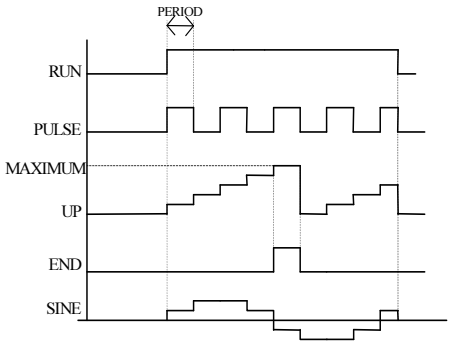
自变数:

RUN	BOOL 型	模式 TRUE=执行 / FALSE=复位成 FALSE
PERIOD	定时器型	1 次采样的时间（脉冲的宽度）
MAXIMUM	整数型	最大计数值
PULSE	BOOL 型	每 1 次采样时间分别反转。
UP	整数型	每 1 次采样时间递增计数
END	BOOL 型	递增计数结束时，为 TRUE
SINE	实数型	正弦波形信号（波形半周期为递增计数时间）

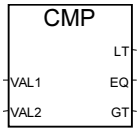
说明:
同时产生各种信号（BOOL 型矩形、整数型递增计数、实数型正弦波）。

计数器达到最大值后，从 0 重新开始。END 信号仅在 1 PERIOD 的时间为 TRUE。

时间图:



7.2.40 No.40；CMP（比较功能块）

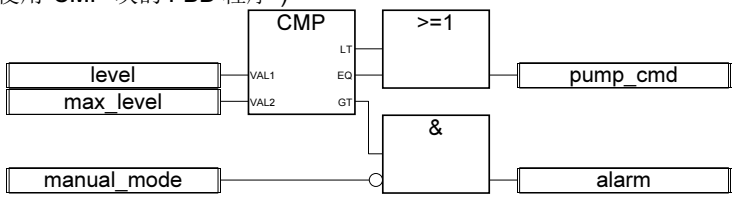


自变数:

VAL1		整数型	带符号整数
VAL2		整数型	带符号整数
LT	BOOL 型	VAL1 < VAL2 时为 TRUE	
EQ	BOOL 型	VAL1 = VAL2 时为 TRUE	
GT	BOOL 型	VAL1 > VAL2 时为 TRUE	

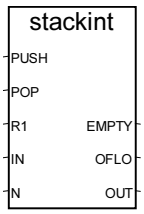
说明:
比较 2 个整数型的值。取 3 种结果之一。（<、=、>）

(*使用"CMP"块的 FBD 程序*)




```
(*等价的 ST 语言: CMP1 为 CMP 块的实例名 *)
CMP1(level, max_level);
pump_cmd := CMP1.LT OR CMP1.EQ;
alarm := CMP1.GT AND NOT(manual_mode);

(* 等价的 IL 语言: *)
LD      level
ST      CMP1.val1
LD      max_level
ST      CMP1.val2
CAL     CMP1
LD      CMP1.LT
OR      CMP1.EQ
ST      pump_cmd
LD      CMP1.GT
ANDN    manual_mode
ST      alarm
```

7.2.41 No.41；STACKINT（整数堆栈）



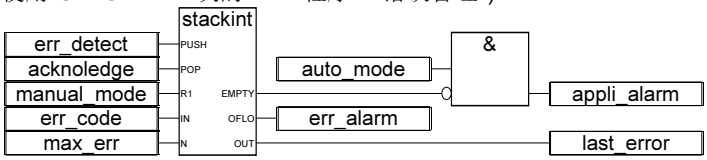
自变数:
PUSH BOOL 型 PUSH 指令：通过上升沿检测，将 **IN** 值追加到堆栈的顶部
POP BOOL 型 POP 指令 ：通过上升沿检测，删除堆栈的前头（最后压入的值）
R1 BOOL 型 将堆栈复位成空的状态
IN 整数型 压入的值
N 整数型 堆栈尺寸
EMPTYBOOL 型 堆栈为空时，为 TRUE
OFLO BOOL 型 上溢：堆栈已满时，为 TRUE
OUT 整数型 堆栈的前头的值

说明:
管理整数型值的堆栈。

STACKINT 功能块针对 PUSH、POP 两种输入，检测上升沿。**最大的堆栈尺寸为 128**。堆栈尺寸 **N 必须是 1 到 128 范围内**的值。

OFLO 值仅在执行一次复位之后才有效（也就是说 R1 一旦设为 TRUE 后有必要返回 FALSE）。

(*使用"STACKINT" 块的 FBD 程序 ： 错误管理*)

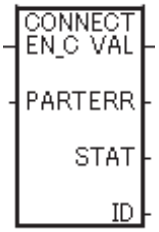



```
(*等价的 ST 语言: STACKINT1 为 STACKINT 块的实例 *)
STACKINT1(err_detect, acknoledge, manual_mode, err_code, max_err);
appli_alarm := auto_mode AND NOT(STACKINT1.EMPTY);
err_alarm := STACKINT1.OFLO;
last_error := STACKINT1.OUT;
```

```
(* 等价的 IL 语言: *)
LD      err_detect
ST      STACKINT1.push
LD      acknoledge
ST      STACKINT1.pop
LD      manual_mode
ST      STACKINT1.r1
LD      err_code
ST      STACKINT1.IN
LD      max_err
ST      STACKINT1.N
CAL     STACKINT1
LD      auto_mode
ANDN    STACKINT1.empty
ST      appli_alarm
LD      STACKINT1.OFLO
ST      err_alarm
LD      STACKINT1.OUT
ST      last_error
```

7.2.42 No.42；CONNECT（面向资源的连接）

目前不可利用。



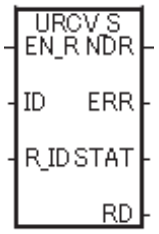
自变数:

EN_C		BOOL 可以连接
PARTNER	STRING	远程的通信伙伴的名称
VALID	BOOL	TRUE 时，连接 ID 为有效
ERROR	BOOL	TRUE 时，接收新的非零状态
STATUS	DINT	最后检测到的状态
ID	DINT	通信通道的标识符（ID）

说明:
执行(当前的项目或别的项目的)远程或本地的资源间的连接，管理基于 USEND_S 块和 URCV_S 块的交换。
由此，制作出通信通道标识符(ID)。
其他的全部的通信功能块(URCV_S 或 USEND_S)都需要此标识符。

7.2.43 No.43；URCV_S（面向资源发送信息）

目前不可利用。



自变数:

EN_R		BOOL 可以接收数据
ID	DINT	通信通道的标识符
R_ID	STRING	通道内的远程 SFB 的标识符
NDR	BOOL	TRUE 时，将新字符串接收到 RD
ERROR	BOOL	TRUE 时，接收新的非零的 STATUS
STATUS	DINT	最后检测到的状态
RD	STRING	接收到的字符串

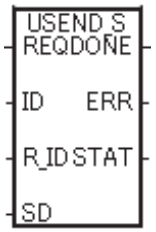
说明:
从（当前项目或别的项目的)远程或本地的资源，接收字符串。

注: 在当前的循环中，调用"URCV_S"前有必要预先调用"Connect"块。

此功能块是从 1 个 USEND_S 实例接收字符串。此时，之前接收到的字符串被覆写。正常接收字符串后，NDR 在 1 次循环间设定为 TRUE。发生错误后，输出参数 ERROR 为 TRUE，状态设定到 STATUS 参数中。

7.2.44 No.44；USEND_S（从资源接收信息）

目前不可利用。



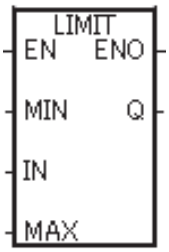
自变数:

REQ		BOOL 在上升沿边缘发送请求
ID	DINT	通信通道的标识符
R_ID	STRING	通道内的远程 CFB 的标识符
SD	STRING	发送的字符串
DONE		BOOL TRUE 时为正常结束
ERROR	BOOL	TRUE 时，接收新的非零 STATUS
STATUS	DINT	最后检测到的状态

说明:
向（当前的项目或别项目的)远程或本地的资源发送字符串。

注: 在当前的循环中，调用"USEND_S"之前，有必要预先调用"Connect"块。
此 CFB 在 REQ 的上升沿边缘，向 1 个 URCV_S 实例发送字符串。字符串正常发送后，设定为 DONE。
发生错误后，输出参数 ERROR 为 TRUE，状态设定到 STATUS 参数中。

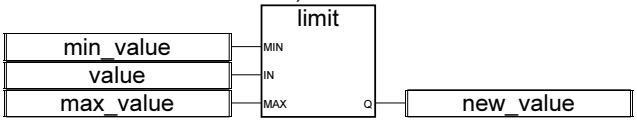
7.2.45 No.45；LIMIT（限制值）



自变数:
MIN 整数型 最小值设定值
IN 整数型 整数输入值
MAX 整数型 最大值设定值
Q 整数型 输入值控制在最大、最小值的范围内
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
给输入值设置上下限。输入值在最大值和最小值之间时保持输入值状态。超过最大值时设为最大值，低于最小值时设为最小值。

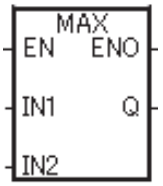
(*使用"LIMIT"块的 FBD 程序 *)



(*等价的 ST 语言:*)
new_value := LIMIT (min_value, value, max_value);
(* 将值控制在 min_value～max_value 的范围内 *)

(* 等价的 IL 语言: *)
LD　　min_value
LIMIT　value, max_value
ST　　new_value

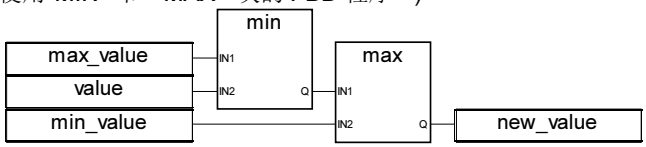
7.2.46 No.46；MAX（最大值）



自变数:
IN1 整数型　整数值
IN2 整数型　整数值
Q 整数型　2 个输入的最大值
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
给出 2 个整数值的最大值

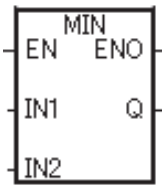
(*使用"MIN" 和 "MAX" 块的 FBD 程序 *)



(*等价的 ST 语言:*)
new_value := MAX (MIN (max_value, value), min_value);
(* 将值控制在 min_value～max_value 的范围内 *)

(* 等价的 IL 语言: *)
LD　　max_value
MIN　　value
MAX　　min_value
ST　　new_value

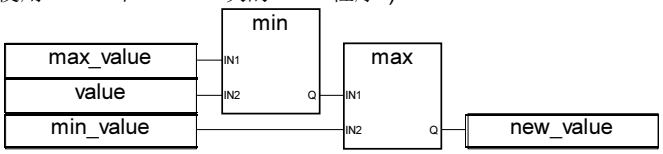
7.2.47 No.47；MIN（最小值）



自变数:
IN1 整数型　整数值
IN2 整数型　整数值
Q 整数型　2 个输入的最小值
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
给出 2 个整数值的最小值。

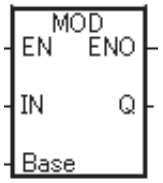
(*使用"MIN" 和 "MAX"块的 FBD 程序*)



(*等价的 ST 语言:*)
new_value := MAX (MIN (max_value, value), min_value);
(* 将值控制在 min_value～max_value 的范围内 *)

(* 等价的 IL 语言: *)
LD max_value
MIN value
MAX min_value
ST new_value

7.2.48 No.48；MOD（余数运算）



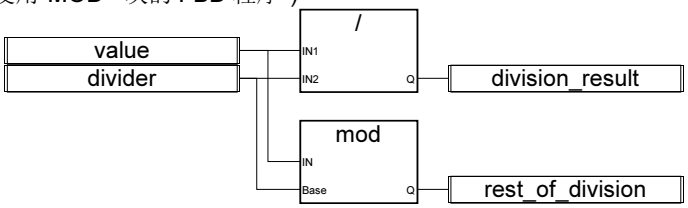
自变数:

IN	整数型	输入值
Base	整数型	输入值（大于 0 的值）
Q	整数型	余数运算的计算结果 Base <= 0 时，为-1

EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
计算出整数值的余数。

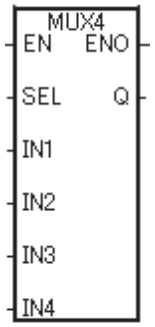
(*使用"MOD" 块的 FBD 程序*)



(*等价的 ST 语言:*)
division_result := (value / divider); (* 整数的除法运算*)
rest_of_division := MOD (value, divider); (* 余数*)

(* 等价的 IL 语言: *)
LD value
DIV divider
ST division_result
LD value
MOD divider
ST rest_of_division

7.2.49 No.49；MUX4（多路复用器 4）



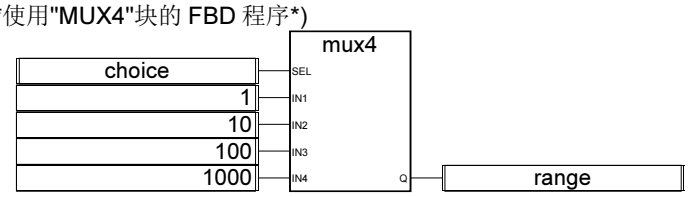
自变数:

SEL	整数型	选择输入值 [0～3]
IN1..IN4	整数型	输入值
Q	整数型	= SEL = 0 时 IN1
		= SEL = 1 时 IN2
		= SEL = 2 时 IN3
		= SEL = 3 时 IN4
		= SEL 为上述以外时则为 0

EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

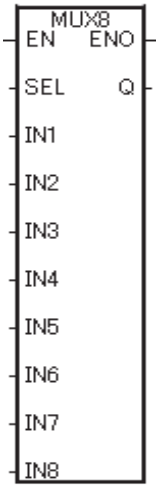
说明:
4 点输入用多路复用器：从 4 个整数输入值当中选择 1 个。

(*使用"MUX4"块的 FBD 程序*)




```
(*等价的 ST 语言:*)
range := MUX4 (choice, 1, 10, 100, 1000);
(* 从 4 个输入当中选择 1 个。choice 为 1 时，range 为 10 *)

(* 等价的 IL 语言: *)
LD      choice
MUX4    1,10,100,1000
ST      range
```

7.2.50 No.50；MUX8（多路复用器 8）



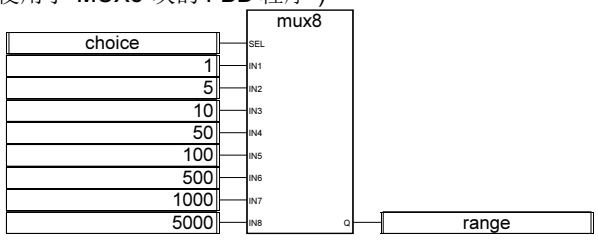
自变数:

SEL	整数型	选择输入值 [0～7]
IN1..IN8	整数型	输入值
Q	整数型	= SEL = 0 时 IN1
		= SEL = 1 时 IN2
		= SEL = 2 时 IN3
		.....
		= SEL = 7 时 IN8
		= SEL 为上述以外时则为 0

EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

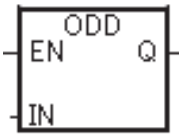
说明:
8 点输入用多路复用器：从 8 个整数输入值当中选择一个。

(*使用了"MUX8"块的 FBD 程序*)




```
(*等价的 ST 语言:*)
range := MUX8 (choice, 1, 5, 10, 50, 100, 500, 1000, 5000);
(* 从 8 个输入当中选择一个。choice 为 3 时，range 为 50 *)

(* 等价的 IL 语言: *)
LD      choice
MUX8    1,5,10,50,100,500,1000,5000
ST      range
```

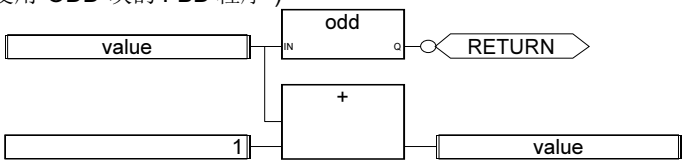
7.2.51 No.51；ODD（奇偶校验）



自变数:
IN 整数型 整数输入
Q BOOL 型 TRUE：输入值为奇数时
FALSE：输入值为偶数时
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。

说明:
整数值的奇偶校验：判断是奇数还是偶数。

(*使用"ODD"块的 FBD 程序*)




```
(*等价的 ST 语言:*)
If Not (ODD (value)) Then Return; End_if;
value := value + 1;
(* value  始终为偶数*)
```

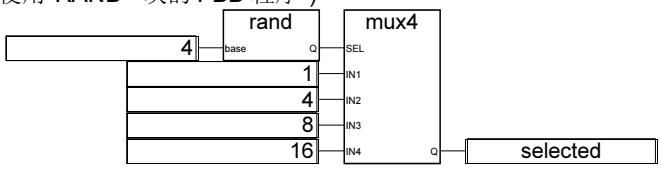
```
(*  等价的 IL 语言: *)
LD      value
ODD
RETNC
LD      value
ADD     1
ST      value
```

7.2.52 No.52；RAND（随机值）

自变数:
base 整数型 随机值的最大值
Q 整数型 [0～base-1] 的随机值
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

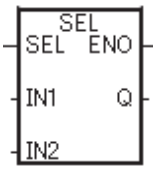
说明:
给出用整数值指定范围的随机值。

(*使用"RAND" 块的 FBD 程序*)




```
(*等价的 ST 语言:*)
selected := MUX4 ( RAND (4), 1, 4, 8, 16 );
(*
根据产生的随机值，从 4 个输入当中任选其一。
RAND 产生的数值为 [0～3]。结果在 MUX4 的 'selected' 产生以下的输出。
1： RAND 输出为 0
4： RAND 输出为 1
8： RAND 输出为 2
16： RAND 输出为 3
*)

(* 等价的 IL 语言: *)
LD      4
RAND
MUX4    1,4,8,16
ST      selected
```

7.2.53 No.53；SEL（二进制选择器）



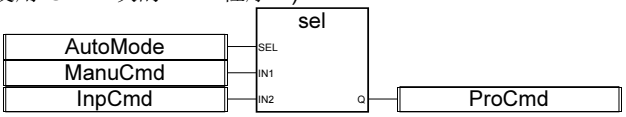
自变数:

SEL	BOOL 型	选择用
IN1, IN2	整数型	整数值
Q	整数型	= IN1： SEL 为 FALSE 时
		= IN2： SEL 为 TRUE 时

EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

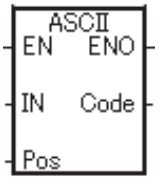
说明:
二进制选择：从 2 个整数值当中选择一个。

(*使用"SEL" 块的 FBD 程序 *)




```
(*等价的 ST 语言:*)
ProcCmd := SEL (AutoMode, ManuCmd, InpCmd);
(* 选择处理的指令*)

(* 等价的 IL 语言: *)
LD      AutoMode
SEL     ManuCmd,InpCmd
ST      ProCmd
```

7.2.54 No.54；ASCII（文字 → ASCII 代码）



自变数:

IN 字符串型 非空的字符串

Pos 整数型 转换的文字的字符串中的位置
[1～ len] (len 为字符串 IN 的长度)

Code 整数型 选中的文字的 ASCII 代码
[0 ～ 255]
Pos 超过字符串长度时，为 0

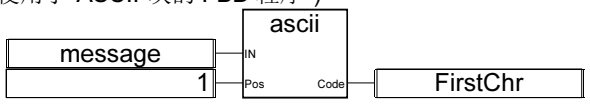
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。

ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:

给出字符串中所指定的文字相对应的 ASCII 代码。

(*使用了"ASCII"块的 FBD 程序*)




```
(*等价的 ST 语言:*)
FirstChr := ASCII (message, 1);
(* FirstChr 为字符串的第 1 个文字的 ASCII 代码*)

(* 等价的 IL 语言: *)
LD      message
ASCII   1
ST      FirstChr
```

7.2.55 No.55；CHAR（ASCII 代码 → 文字）

自变数:

Code 整数型 ASCII 代码 [0 ～ 255]

Q 字符串型 1 个文字的字符串。
文字是 ASCII 代码（Code）所对应的文字。ASCII 代码使用的是用 256 相除后的余数。

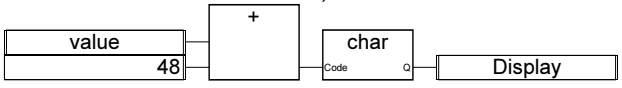
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。

ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:

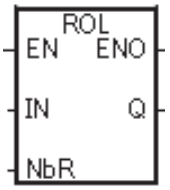
给出 ASCII 代码所对应的文字。

(*使用了"CHAR"块的 FBD 程序*)



```
(*等价的 ST 语言:*)
Display := CHAR ( value + 48 );
(* value 为 0～9 *)
(* 48 为'0'的 ASCII 代码*)
(* 结果为'0' ～ '9'的文字*)

(* 等价的 IL 语言: *)
LD      value
ADD     48
CHAR
ST      Display
```

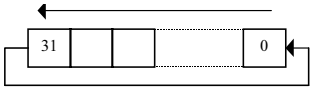
7.2.56 No.56；ROL（向左循环）



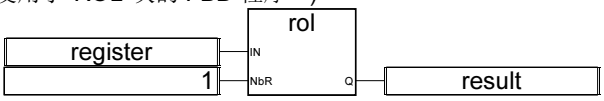
自变数:

IN 整数型 整数值
NbR 整数型 要循环的位数 [1～31]
Q 整数型 向左循环后的结果
nb_rotation <= 0 时，无变化
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
对整数值的位实施向左循环。按照 3 2 位执行循环。

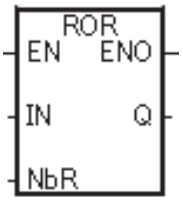


(*使用了"ROL"块的 FBD 程序 *)



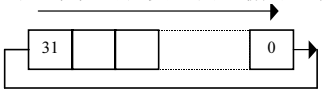

```
(*等价的 ST 语言:*)
result := ROL (register, 1);
(* register = 2#0100_1101_0011_0101*)
(* result   = 2#1001_1010_0110_1010*)

(* 等价的 IL 语言: *)
LD      register
ROL     1
ST      result
```

7.2.57 No.57；ROR（向右循环）

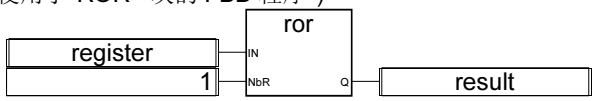


自变数:
IN 整数型 整数值
NbR 整数型 要循环的位数 [1～31]
Q 整数型 循环后的结果
nb_rotation <= 0 时，无变化
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
对整数值的位实施向右循环。按照 32 位执行循环。

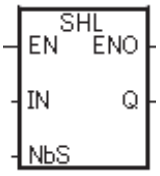


(*使用了"ROR" 块的 FBD 程序*)



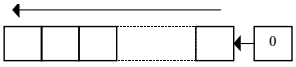

```
(*等价的 ST 语言:*)
result := ROR (register, 1);
(* register = 2#0100_1101_0011_0101 *)
(* result   = 2#1010_0110_1001_1010 *)

(* 等价的 IL 语言: *)
LD    register
ROR   1
ST    result
```

7.2.58 No.58；SHL（向左偏移）

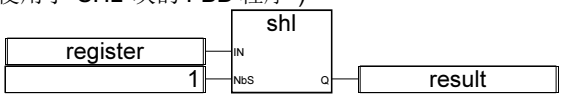


自变数:
IN 整数型 整数值
NbS 整数型 位偏移数 [1～31]
Q 整数型 向左偏移后的结果
nb_shift <= 0 时，无变化。
在最低位插入 0。
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
对整数值的位实施向左偏移。按照 3 2 位单位执行偏移。



(*使用了"SHL"块的 FBD 程序*)




```
(*等价的 ST 语言:*)
result := SHL (register,1);
(* register = 2#0100_1101_0011_0101 *)
(* result   = 2#1001_1010_0110_1010 *)

(*  等价的 IL 语言: *)
LD      register
SHL     1
ST      result
```

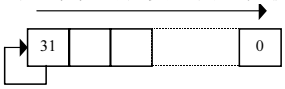
7.2.59 No.59；SHR（向右偏移）

自变数:

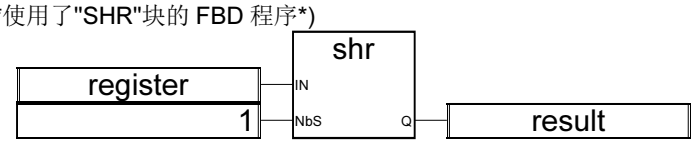
IN 整数型 整数值
NbS 整数型 位偏移数 [1～31]
Q 整数型 向右偏移后的结果
nb_shift <= 0 时，无变化。
最高位不变化。

EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
对整数值的位实施向右偏移。按照 3 2 位单位执行偏移。



(*使用了"SHR"块的 FBD 程序*)



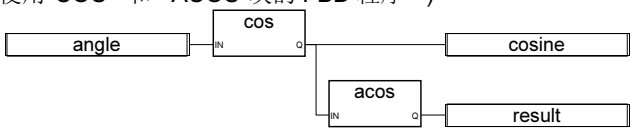

```
(*等价的 ST 语言:*)
result := SHR (register,1);
(* register = 2#1100_1101_0011_0101 *)
(* result   = 2#1110_0110_1001_1010 *)

(*  等价的 IL 语言: *)
LD      register
SHR     1
ST      result
```

7.2.60 No.60；ACOS（反余弦）

自变数:
IN 实数型 必须为-1.0 ～ +1.0
Q 实数型 输入值的反余弦[0.0～π]
= 0.0 ：不正确的输入值时
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
计算实数值的反余弦。

(*使用"COS" 和 "ACOS"块的 FBD 程序 *)



(*等价的 ST 语言:*)
cosine := COS (angle);
result := ACOS (cosine); (* result 等同于 angle*)

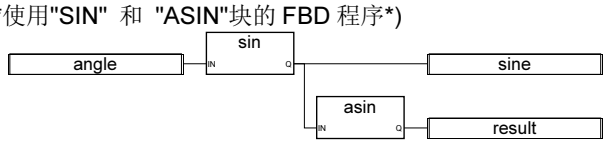
(* 等价的 IL 语言: *)
LD angle
COS
ST cosine
ACOS
ST result

7.2.61 No.61；ASIN（反正弦）

自变数:
IN 实数型 必须为-1.0 ～ +1.0
Q 实数型 输入值的反余弦[-π/2～+π/2]
= 0.0 ：不正确的输入值时
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
计算实数值的反正弦。

(*使用"SIN" 和 "ASIN"块的 FBD 程序*)



(*等价的 ST 语言:*)
sine := SIN (angle);
result := ASIN (sine); (* result 等同于 angle *)

(* 等价的 IL 语言: *)
LD angle

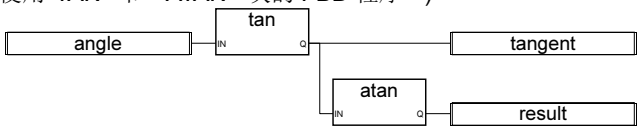
```
SIN
ST      sine
ASIN
ST      result
```

7.2.62 No.62；ATAN（反正切）

自变数:
IN 实数型 实数值
Q 实数型 输入值的反正切[-π/2～+π/2]
= 0.0：不正确的输入值时
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
计算实数值的反正切。

(*使用"TAN" 和 "ATAN" 块的 FBD 程序 *)



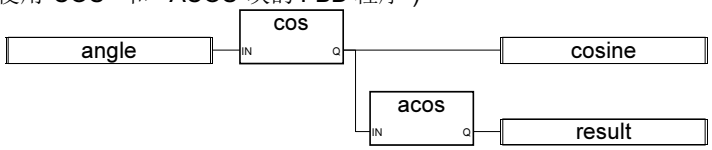

```
(*等价的 ST 语言:*)
tangent := TAN (angle);
result := ATAN (tangent); (* result 等同于 angle *)

(* 等价的 IL 语言: *)
LD      angle
TAN
ST      tangent
ATAN
ST      result
```

7.2.63 No.63；COS（余弦）

自变数:
IN 实数型 实数值
Q 实数型 输入值的余弦[-1.0 ～ +1.0]
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
计算实数值的余弦。

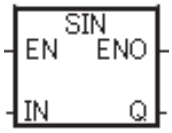
(*使用"COS" 和 "ACOS"块的 FBD 程序*)



(*等价的 ST 语言:*)
cosine := COS (angle);
result := ACOS (cosine); (* result 等同于 angle *)

(* 等价的 IL 语言: *)
LD angle
COS
ST cosine
ACOS
ST result

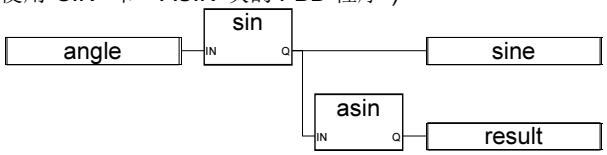
7.2.64 No.64；SIN（正弦）



自变数:
IN 实数型 实数值
Q 实数型 输入值的正弦[-1.0 ～ +1.0]
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
计算实数值的正弦。

(*使用"SIN" 和 "ASIN"块的 FBD 程序*)



(*等价的 ST 语言:*)
sine := SIN (angle);
result := ASIN (sine); (* result 等同于 angle *)

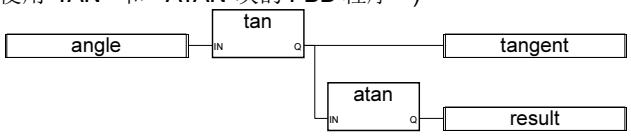
(* 等价的 IL 语言: *)
LD angle
SIN
ST sine
ASIN
ST result

7.2.65 No.65；TAN（正切）

自变数:
IN 实数型 用 π 相除后的余数不得为 π/2。
Q 实数型 输入值的正切。
相对于不正确的输入，为 1E+38。
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

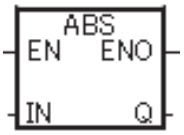
说明:
计算实数值的正切。

(*使用"TAN" 和 "ATAN"块的 FBD 程序 *)




```
(*等价的 ST 语言:*)
tangent := TAN (angle);
result := ATAN (tangent); (* result 等同于 angle *)

(* 等价的 IL 语言: *)
LD      angle
TAN
ST      tangent
ATAN
ST      result
```

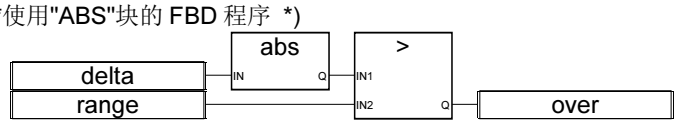
7.2.66 No.66；ABS（绝对值）



自变数:
IN 实数型 实数值输入
Q 实数型 绝对值（始终为 0 以上）
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
给出实数的绝对值。

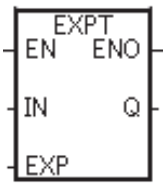
(*使用"ABS"块的 FBD 程序 *)




```
(*等价的 ST 语言: *)
over := (ABS (delta) > range);
```

```
(* 等价的 IL 语言: *)
LD      delta
ABS
GT      range
ST      over
```

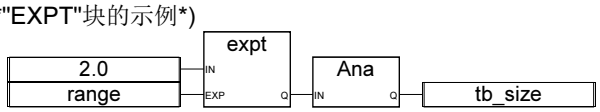
7.2.67 No.67；EXPT（指数）



自变数:
IN 实数型 带符号实数
EXP 整数型 整数（指数部）
Q 实数型 (IN ^{EXP})
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

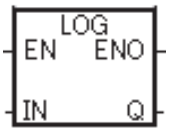
说明:
以实数给出(base ^{exponent})的运算结果。'base' 为最初的自变数， 'exponent' 为第 2 个自变数，为整数。

(*"EXPT"块的示例*)




```
(*等价的 ST 语言:*)
tb_size := ANA (EXPT (2.0, range) );

(* 等价的 IL 语言: *)
LD      2.0
EXPT    range
ANA
ST      tb_size
```

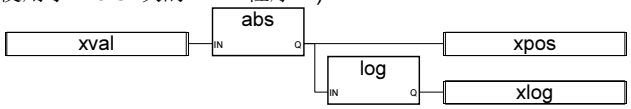
7.2.68 No.68；LOG（常用对数）



自变数:
IN 实数型 大于 0 的实数值
Q 实数型 输入值的常用对数（以 10 为底数）
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

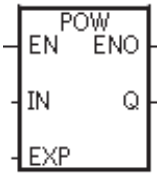
说明:
给出实数输入的常用对数。

(*使用了"LOG"块的 FBD 程序 *)




```
(*等价的 ST 语言:*)
xpos := ABS (xval);
xlog := LOG (xpos);

(* 等价的 IL 语言: *)
LD      xval
ABS
ST      xpos
LOG
ST      xlog
```

7.2.69 No.69；POW（乘方计算）



自变数:

IN 实数型 实数值（非运算数）
EXP 实数型 实数值（乘方数）
Q 实数型 (IN ^{EXP})

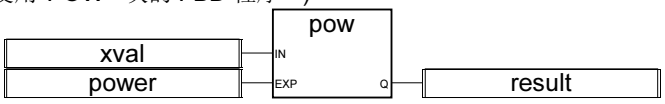
1.0 ： IN<> 0.0、 EXP = 0.0 时
0.0 ： IN= 0.0 、 EXP < 0.0 时
0.0 ： IN =0.0 、 EXP = 0.0 时
0.0 ： IN < 0 时

EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:

给出实数的乘方(base ^{exponent})。'base' 为最初的自变数， 'exponent' 为第 2 个自变数，为实数。

(*使用"POW" 块的 FBD 程序 *)



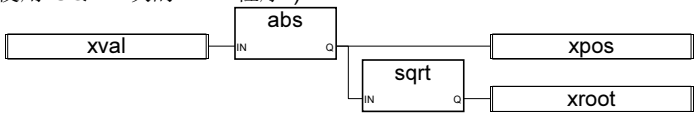

```
(*等价的 ST 语言:*)
result := POW (xval, power);

(* 等价的 IL 语言: *)
LD      xval
POW     power
ST      result
```

7.2.70 No.70；SQRT（平方根）

自变数:
IN 实数型 实数值（>= 0）
Q 实数型 输入值的平方根
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
计算实数值的平方根。

(*使用"SQRT"块的 FBD 程序*)



(*等价的 ST 语言:*)
xpos := ABS (xval);
xroot := SQRT (xpos);

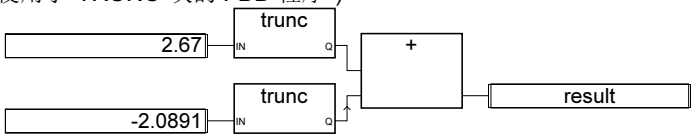
(* 等价的 IL 语言: *)
LD xval
ABS
ST xpos
SQRT
ST xrout

7.2.71 No.71；TRUNC（舍去小数部分）

自变数:
IN 实数型 实数值
Q 实数型 IN>0，小于输入值的最大的整数值
IN<0，大于输入值的最小的整数值
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
给出实数值的整数部分。

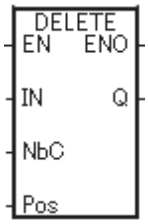
(*使用了"TRUNC"块的 FBD 程序*)



(*等价的 ST 语言:*)
result := TRUNC (+2.67) + TRUNC (-2.0891);
(* 意义: result := 2.0 + (-2.0) := 0.0; *)

```
(* 等价的 IL 语言: *)
LD      2.67
TRUNC
ST      temporary       (*  1 个目的 TRUNC 的结果  *)
LD      -2.0891
TRUNC
ADD     temporary
ST      result
```

7.2.72 No.72；DELETE（字符串的删除）

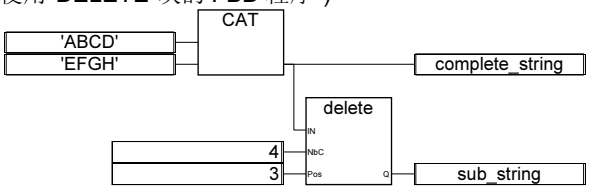


自变数:

IN 字符串型 非空的字符串
NbC 整数型 从字符串删除的文字数
Pos 整数型 删除的最初的字符串的位置
(字符串的前头的位置为 1)
Q 字符串型 删除指定字符串后的最终字符串
Pos < 1 时，空字符串
Pos > IN 的字符串时，最初的文字
NbC <= 0 时，最初的文字
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

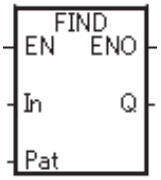
说明:
删除字符串的一部分。

(*使用"DELETE"块的 FBD 程序*)




```
(*等价的 ST 语言:*)
complete_string := 'ABCD' + 'EFGH'; (* complete_string  为  'ABCDEFGH' *)
sub_string := DELETE (complete_string, 4, 3); (* sub_string  为  'ABGH' *)

(* 等价的 IL 语言: *)
LD      'ABCD'
ADD     'EFGH'
ST      complete_string
DELETE4,3
ST      sub_string
```

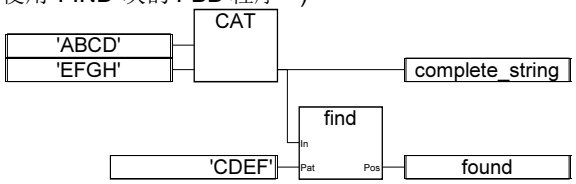
7.2.73 No.73；FIND（字符串检索）



自变数:
In 字符串型 输入字符串
Pat 字符串型 检索的字符串图形。非空。
Pos 整数型 = 0：未发现字符串图形时
= 最早发现检索字符串的位置
(如为第 1 个文字则为 1)
区分**大写文字、小写文字。**
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
从字符串当中检索指定的字符串，发现时返回该部位。

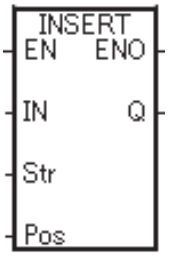
(*使用"FIND"块的 FBD 程序 *)



(*等价的 ST 语言:*)
complete_string := 'ABCD' + 'EFGH'; (* complete_string 为 'ABCDEFGH' *)
found := FIND (complete_string, 'CDEF'); (* found 为 3 *)

(* 等价的 IL 语言: *)
LD 'ABCD'
ADD 'EFGH'
ST complete_string
FIND 'CDEF'
ST found

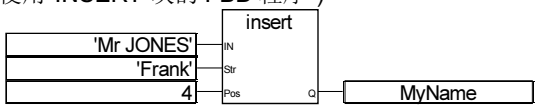
7.2.74 No.74；INSERT（字符串的插入）



自变数:
IN 字符串型 源字符串
Str 字符串型 插入的字符串
Pos 整数型 插入位置。插入字符串在编号的前面。字符串的前头为 1。
Q 字符串型 插入后的最终字符串
Pos <= 0 时，空的字符串
Pos 超过字符串 IN 的长度时，为 2 个字符串相连后的字符串。
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

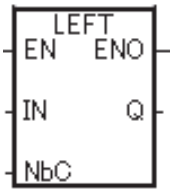
说明:
在字符串的指定位置插入其他的字符串。

(*使用"INSERT"块的 FBD 程序*)




```
(*等价的 ST 语言:*)
MyName := INSERT ('Mr JONES', 'Frank ', 4);
(* MyName ： 'Mr Frank JONES' *)

(* 等价的 IL 语言: *)
LD      'Mr JONES'
INSERT 'Frank',4
ST      MyName
```

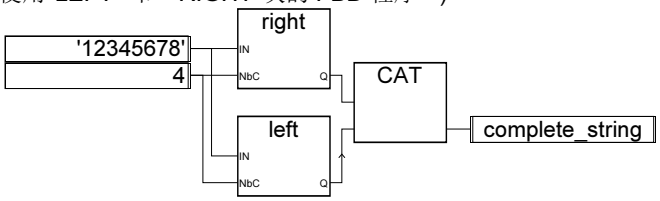
7.2.75 No.75；LEFT（字符串的左侧部分的获取）



自变数:
IN 字符串型 非空的字符串
NbC 整数型 取出的文字数
（小于字符串 IN 的长度）
Q 字符串型 字符串 IN 的左侧字符串（字符串长度为 NbC）
NbC <= 0 时，空的字符串。
NbC >= IN 的长度时，字符串 IN 整体。
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
从源字符串的左侧按照指定的文字数取出。

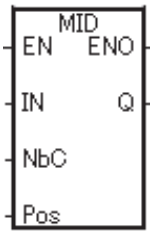
(*使用"LEFT" 和 "RIGHT"块的 FBD 程序 *)




```
(*等价的 ST 语言:*)
complete_string := RIGHT ('12345678', 4) + LEFT ('12345678', 4);
(* complete_string ： '56781234'
按照 RIGHT 取出的字符串为 '5678'
按照 LEFT 取出的字符串为 '1234'
*)
```

```
(* 等价的 IL 语言: 首先调用 LEFT。 *)
LD      '12345678'
LEFT    4
ST      sub_string      (* 中间结果 *)
LD      '12345678'
RIGHT   4
ADD     sub_string
ST      complete_string
```

7.2.76 No.76；MID（字符串的中间部分的获取）

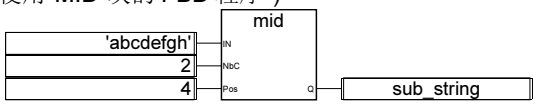


自变数:

IN 字符串型 非空的字符串
NbC 整数型 取出的文字数
（小于字符串 IN 的长度）
Pos 整数型 取出的位置
(前头为 1)
Q 字符串型 取出的字符串 (字符串长度为 NbC)。
指定的自变数不正确时，为空的字符串。
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
从指定位置取出指定长度的字符串。

(*使用"MID"块的 FBD 程序*)



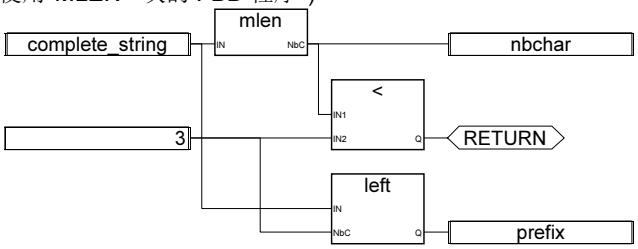

```
(*等价的 ST 语言:*)
sub_string := MID ('abcdefgh', 2, 4);
(* sub_string ： 'de' *)

(*等价的 IL 语言: *)
LD      'abcdefgh'
MID     2,4
ST      sub_string
```

7.2.77 No.77；MLEN（字符串长度的获取）

自变数:
IN 字符串型 字符串
NbC 整数型 字符串 IN 的字符串长度
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
计算字符串的字符串长度。

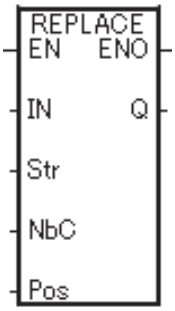
(*使用"MLEN" 块的 FBD 程序*)



(*等价的 ST 语言:*)
nbchar := MLEN (complete_string);
If (nbchar < 3) Then Return; End_if;
prefix := LEFT (complete_string, 3);
(* 此程序是从字符串提取左侧 3 个文字，加入 prefix 字符串。如果字符串长度不足 3 时，不作任何处理。 *)

(*等价的 IL 语言: *)
```
LD      complete_string
MLEN
ST      nbchar
LT      3
RETC
LD      complete_string
LEFT    3
ST      prefix
```

7.2.78 No.78；REPLACE（字符串置换）

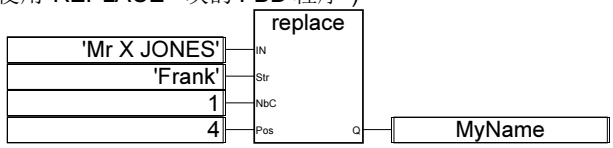


自变数:
IN 字符串型 源字符串
Str 字符串型 插入的字符串(置换 NbC 文字部分)
NbC 整数型 删除的文字数
Pos 整数型 置换的最初文字的位置
(前头的位置为 1)
Q 字符串型 置换后的最终字符串
- 首先，从 Pos 的部位删除 NbC 文字部分。
- 接着，字符串 Str 插入此部位。
Pos <= 0 时，为空的字符串。
Pos >字符串 IN 的长度时，为 2 个字符串的结合(IN + Str)。
NbC <= 0 时，为源字符串。

EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

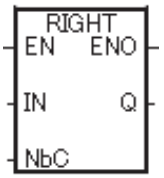
说明:
将字符串的一部分用别的字符串进行置换。

(*使用"REPLACE" 块的 FBD 程序*)




```
(*等价的 ST 语言:*)
MyName := REPLACE ('Mr X JONES', 'Frank', 1, 4);
(* MyName 为 'Mr Frank JONES'  *)

(*等价的 IL 语言: *)
LD      'Mr X JONES'
REPLACE     'Frank',1,4
ST      MyName
```

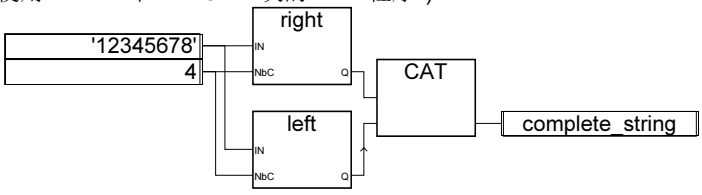
7.2.79 No.79；RIGHT（字符串的右侧部分的获取）



自变数:
IN 字符串型 非空的字符串
NbC 整数型 取出的文字数（小于字符串 IN 的长度）
Q 字符串型 字符串 IN 的右侧字符串（字符串长度为 NbC）
NbC <= 0 时，空的字符串。
NbC >=字符串 IN 的长度时，IN 整体。
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
从字符串的右侧按照指定的文字数取出。

(*使用"LEFT" 和 "RIGHT"块的 FBD 程序*)




```
(*等价的 ST 语言:*)
complete_string := RIGHT ('12345678', 4) + LEFT ('12345678', 4);
(* complete_string 为 '56781234'
按照 RIGHT 取出的字符串 '5678'
按照 LEFT 取出的字符串 '1234'
*)
```

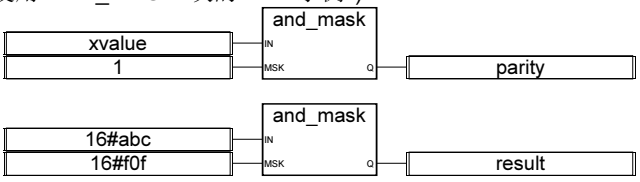
```
(*等价的 IL 语言: First done is call to LEFT *)
LD      '12345678'
LEFT    4
ST      sub_string          (*  中间结果*)
LD      '12345678'
RIGHT   4
ADD     sub_string
ST      complete_string
```

7.2.80 No.80；AND_MASK（按各整数位执行 AND MASK）

自变数:
IN 整数型
MSK 整数型
Q 整数型 IN 和 MSK 的各位的逻辑积
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
是整数型变数的逻辑积，是 IN 的各位和 MSK 的各位之间的逻辑积

(*使用 AND_MASK 块的 FBD 示例*)



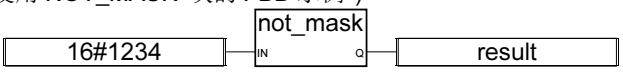

```
(*  等价的 ST 语言: *)
parity := AND_MASK (xvalue, 1); (* xvalue 为奇数则为  1*)
result := AND_MASK (16#abc, 16#f0f); (*  为 16#a0c *)

(*  等价的 IL 语言: *)
LD      xvalue
AND_MASK    1
ST      parity
LD      16#abc
AND_MASK    16#f0f
ST      result
```

7.2.81 No.81；NOT_MASK（按各整数位执行否定）

自变数:
IN 整数型
Q 整数型 3 2 位表述的 IN 的各位的反转
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
整数型变数的各位的反转

(*使用 NOT_MASK 块的 FBD 示例*)



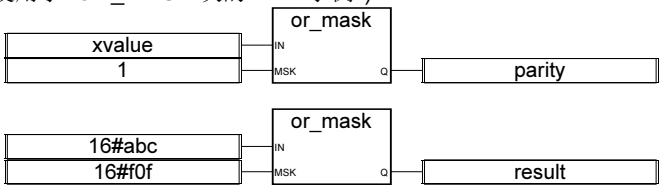

```
(* 等价的 ST 语言: *)
result := NOT_MASK (16#1234);
(* result 为 16#FFFF_EDCB *)

(* 等价的 IL 语言: *)
LD      16#1234
NOT_MASK
ST      result
```

7.2.82 No.82；OR_MASK（按各整数位执行 OR MASK）

自变数:
IN 整数型
MSK 整数型
Q 整数型 IN 和 MSK 的各位的逻辑和
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
是整数型变数的逻辑和，是 IN 的各位和 MSK 的各位之间的逻辑和

(*使用了 OR_MASK 块的 FBD 示例*)



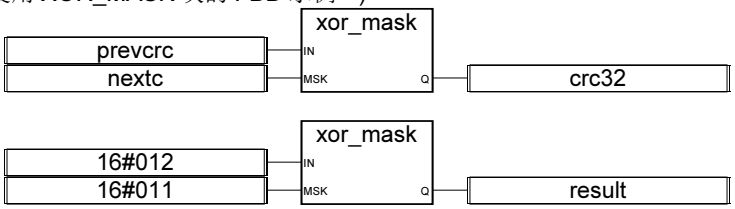

```
(* 等价的 ST 语言: *)
is_odd := OR_MASK (xvalue, 1); (* 必定为奇数*)
result := OR_MASK (16#abc, 16#f0f); (* 为 16#fbf *)

(* 等价的 IL 语言: *)
LD      xvalue
OR_MASK     1
ST      is_odd
LD      16#abc
OR_MASK     16#f0f
ST      result
```

7.2.83 No.83；XOR_MASK（按各整数位执行 XOR MASK）

自变数:
IN 整数型
MSK 整数型
Q 整数型 IN 和 MSK 的各位的异或
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
是整数型变数的异或，是 IN 的各位和 MSK 的各位之间的异或

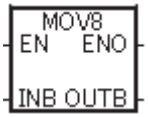
(*使用 XOR_MASK 块的 FBD 示例 *)




```
(* 等价的 ST 语言: *)
crc32 := XOR_MASK (prevcrc, nextc);
result := XOR_MASK (16#012, 16#011); (* 为 16#003 *)

(* 等价的 IL 语言: *)
LD      prevcrc
XOR_MASK    nextc
ST      crc32
LD      16#012
XOR_MASK    16#011
ST      result
```

7.2.84 No.84；MOV8（传送 8 个 BOOL 数据）

是本控制装置独特的命令。

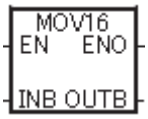


自变数:
INB BOOL 型
OUTBBOOL 型
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
将以 INB 为前头的连续的 8 个 BOOL 变数的值，代入以 OUTB 为前头的 8 个 BOOL 变数。

7.2.85 No.85；MOV16（传送 16 个 BOOL 数据）

是本控制装置独特的命令。

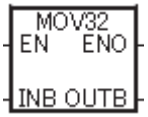


自变数:
INB BOOL 型
OUTB BOOL 型
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
将以 INB 为前头的连续的 16 个 BOOL 变数的值，代入以 OUTB 为前头的 16 个 BOOL 变数。

7.2.86 No.86；MOV32（传送 32 个 BOOL 数据）

是本控制装置独特的命令。



自变数:
INB BOOL 型
OUTB BOOL 型
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
将以 INB 为前头的连续的 32 个 BOOL 变数的值，代入以 OUTB 为前头的 32 个 BOOL 变数。

7.2.87 No.87；BTOD8（向整数变数传送 8 个 BOOL 数据）

是本控制装置独特的命令。

自变数:
INB BOOL 型
OUTD 整数型
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
将以 INB 为前头的连续的 8 个 BOOL 变数的值，向用 OUTD 表示的整数变数，将 INB 作为 LSB 代入。

7.2.88 No.88；BTOD16（向整数变数传 16 个 BOOL 数据送）

是本控制装置独特的命令。

自变数:
INB BOOL 型
OUTD 整数型
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
将以 INB 为前头的连续的 16 个 BOOL 变数的值，向用 OUTD 表示的整数变数，将 INB 作为 LSB 代入。

7.2.89 No.89；BTOD32（向整数变数传送 32 个 BOOL 数据）

是本控制装置独特的命令。

自变数:
INB BOOL 型
OUTD 整数型
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
将以 INB 为前头的连续的 32 个 BOOL 变数的值，向用 OUTD 表示的整数变数，将 INB 作为 LSB 代入。。

7.2.90 No.90；DTOB8（将整数变数向 8 个 BOOL 代入）

是本控制装置独特的命令。

自变数:
IND 整数型
OUTB BOOL 型
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
将用 IND 表示的变数的低位 8bit 的值，向以 OUTB 为前头的 8 个 BOOL 变数，将 OUTB 作为 LSB 代入。

7.2.91 No.91；DTOB16（将整数变数向 16 个 BOOL 代入）

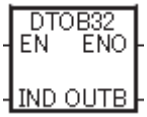
是本控制装置独特的命令。

自变数:
IND 整数型
OUTBBOOL 型
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
将用 IND 表示的变数的低位 16bit 的值，向以 OUTB 为前头的 16 个 BOOL 变数，将 OUTB 作为 LSB 代入。

7.2.92 No.92；DTOB32（将整数变数向 32 个 BOOL 代入）

是本控制装置独特的命令。



自变数:
IND 整数型
OUTBBOOL 型
EN 仅在输入为 TRUE 时才执行。每次扫描执行时，直接连接到母线。
ENO 输出始终是和块的最初的输入相同的状态。请连接 BOOL 型变数。

说明:
将用 IND 表示的变数的值，向以 OUTB 为前头的 32 个 BOOL 变数，将 OUTB 作为 LSB 代入。

NOTE

8章 故障排除

本章将对故障排除进行说明。

8.1 故障排除

重要

发生下列异常时，有可能在内藏 PLC 动作环境及梯形程序中发生了错误，因此内藏 PLC 状态设定应设定为“停止”。请确认内藏 PLC 动作环境后，状态设定请设为“启动”。

错误

E367 内藏 PLC 检测到错误。

［原因］ 在内藏 PLC 检测到错误时，会发生此错误。

［措施］ 应确认内藏 PLC 的动作环境。

错误

E550 为内藏 PLC 的扫描时间超时。

［原因］ 在内藏 PLC 的扫描时间过长时检测。

［措施］ 请修改 PLC 扫描时间的设定。或者，修改梯形程序使扫描时间变短。

重点

发生“E117 检测出停电。”的同时发生 E550 时，因停电而有可能发生了扫描时间超过错误。

警告

E2367 内藏 PLC 的扫描已停止。

［原因］ 内藏 PLC 状态设定虽处于“分离”以外，但在机器人动作时，PLC 的扫描却已停止时，会发生此错误。

［措施］ 先以常数模式将内藏 PLC 状态设定设定为“启动”，再开始扫描。

NOTE

NACHI
株式会社 不二越

http://www.nachi-fujikoshi.co.jp/

总公司　东京都港区东新桥1-9-2　汐留住友大厦17层　邮编 105-0021
Tel +81-3-5568-5245 Fax +81-3-5568-5236

中国

那智不二越（上海）贸易有限公司
上海市普陀区丹巴路98弄7号　龙裕财富中心11层　邮编 200062
Tel 021-6915-2200　Fax 021-6915-5427

重庆分公司
重庆市江北区红鼎国际名苑C座17-18, 17-19　邮编 400020
Tel 023-8816-1967　Fax 023-8816-1968

沈阳分公司
辽宁省沈阳市沈河区悦宾街1号方圆大厦304室　邮编 110000
Tel 024-3120-2252　Fax 024-2250-5316

北京分公司
北京市朝阳区朝外大街乙12号 昆泰国际大厦 0-1110室　邮编 100020
Tel 010-5879-0181　Fax 010-5879-0182

长春事务所
长春市绿园区普阳街1688号长融大厦B座707室　邮编 130061
Tel 0431-8507-8700　Fax 0431-8507-8701

广州事务所
广州市番禺区东环路431号港信城B座505室　邮编 510120
Tel 020-2293-9503　Fax 020-2293-9503

那智不二越（江苏）精密机械有限公司
江苏省张家港市经济技术开发区(南区)南园路39号　邮编 215618
Tel 0512-3500-7616　Fax 0512-3500-7615

上海不二越精密轴承有限公司
上海市嘉定区马陆镇丰茂路258号易通工业园　邮编 201801
Tel 021-6915-6200　Fax 021-6915-6202

耐锯（上海）精密刀具有限公司
上海市嘉定区马陆镇丰茂路258号易通工业园　邮编 201801
Tel 021-6915-5899　Fax 021-6915-5898

东莞建越精密轴承有限公司
东莞市洪梅镇凼涌村
Tel 0769-8843-1300　Fax 0769-8843-1330

机器人事业部
富山市不二越本町1-1-1, JAPAN 邮编930-8511
Tel +81-76-423-5137
Fax +81-76-493-5252