

欢迎参加 **OMRON** NJ专题班



第一章 NJ系列的结构

- 第一节 NJ系列概述
- 第二节 周边设备
- 第三节 Sysmac Studio

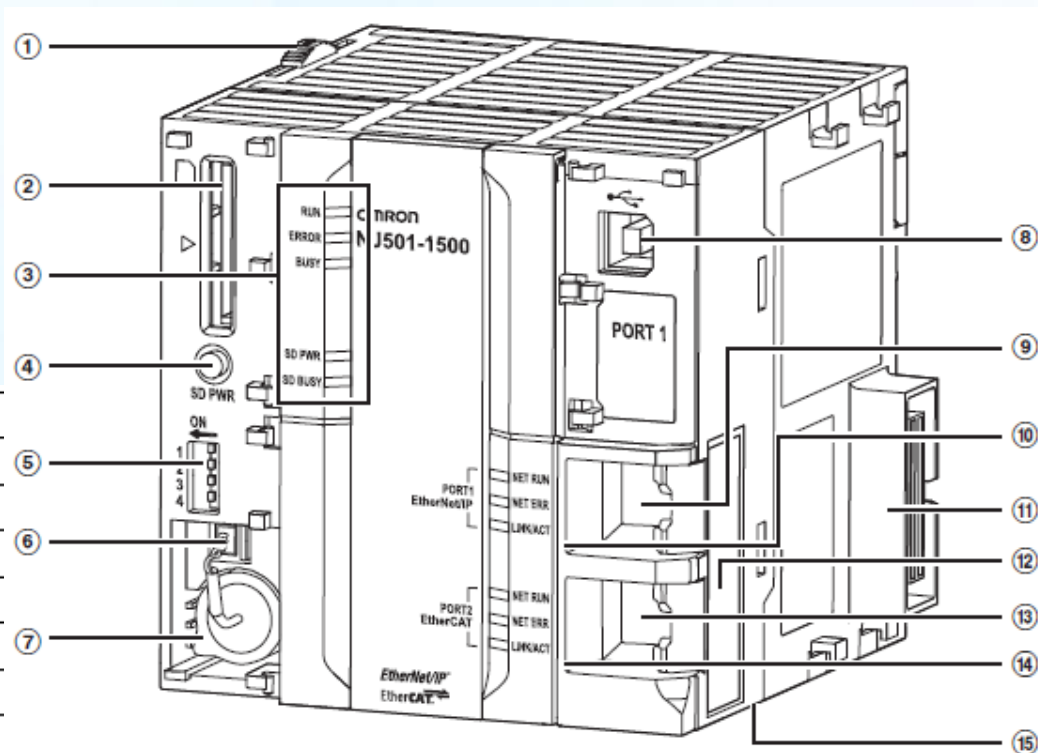
第一章 NJ系列的结构

NJ501主要特点总结

- 内置SD卡槽， EtherCAT口， USB口， Ethernet-IP口
- 内置运动控制模块
- LD: 1.9ns
- 程序容量大小以MB为单位， 总数为20MB。 约为400K步
- NJ变量大小4MB断电不保存， 2MB断电保存

第一章 NJ系列的结构

NJ外形，各部件位置和名称



1	卡扣
2	SD 内存卡接口
3	CPU 单元状态显示
4	SD 内存卡供电开关
5	DIP 开关
6	电池连接口
7	电池
8	USB 连接口
9	内置 EtherNet/IP 口 (port1)
10	内置 EtherNet/IP 状态显示
11	单元连接口
12	CPU ID 信息标签
13	内置 EtherCAT 口 (port2)
14	内置 EtherCAT 口状态显示
15	DIN 导轨卡槽

第一章 NJ系列的结构

- NJ用SD卡说明
- SD卡型号为HMC-SD291
- HMC-SD291容量为2GB
- HMC-SD291可擦写次数为10万次
- NJ用供电单元说明
- 供电单元型号为NJ-PA3001,NJ-PD3001
- 供电单元上有5VDC输出, 24VDC输出, **总功率不能超过30W**
- **其他: CJ系列的PA单元不能在NJ上使用**

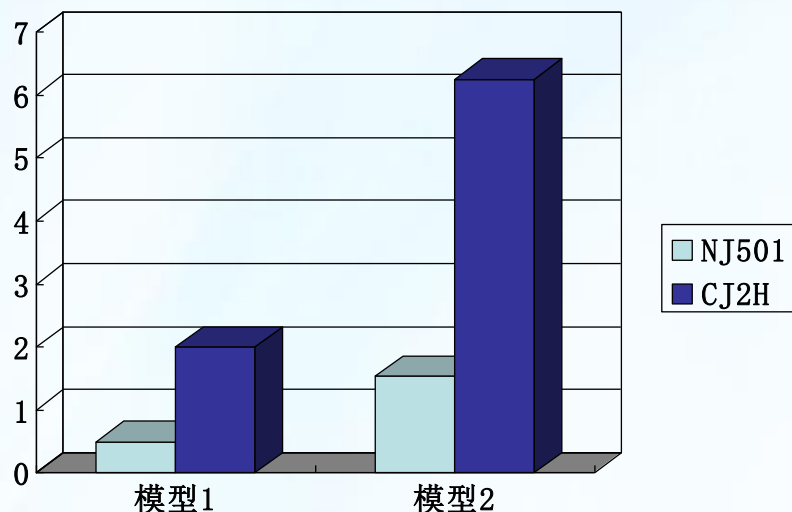
第一章 NJ系列的结构

NJ系列PLC优势

- NJ系列EtherCAT网络中从站交换数据时机可与主站（NJ机）的任务循环周期同步。
- 内置多种根据实际要求开发的功能块（100%欧姆龙独立开发）
- 适用于高端场合和大型系统的高速性

第一章 NJ系列的结构

- NJ高速性的数据说明
- 运动控制环境1为16轴
动作无I/O控制
- 运动控制环境2为32轴
60K步程序加50K步逻辑
程序



	NJ501	CJ2H
LD指令	1.9ns	16ns
实数型变量转换	26ns	6000ns
运动控制（16轴）	0.5ms	2ms
含运动控制的运用（32轴）	1.54ms	6.25ms

第一章 NJ系列的结构

- 在不使用附加模块的情况下，NJ可以支持的周边设备如下
- EtherNet/IP触摸屏
- EtherCAT伺服
- EtherCAT变频器
- EtherCAT I/O从站
- EtherCAT视觉传感器



第一章 NJ系列的结构

- EtherCAT网络简介
- EtherCAT(Ethernet Control Automation Technology)是基于以太网但比以太网更高速和高效的一种高性能工业网络系统。

The logo for EtherCAT, featuring the text "EtherCAT" in a bold, black, sans-serif font. To the right of the text is a stylized graphic consisting of a red arrow pointing right, positioned above a black arrow pointing left. A registered trademark symbol (®) is located to the right of the text.

EtherCAT®

第一章 NJ系列的结构

- EtherCAT网络的特点
 - 100Mbps的高速性
 - 与Ethernet存在共同性
 - 可以使用Ethernet网络用以太网电缆
 - 采用以太网帧，实现高速和实时数据传输
-
- NJ系列EtherCAT网络特点
 - 支持菊花链和分支连接两种方式
 - 最多支持192个从站
 - 从站间线缆长度最大100米
 - 使用专用分支器

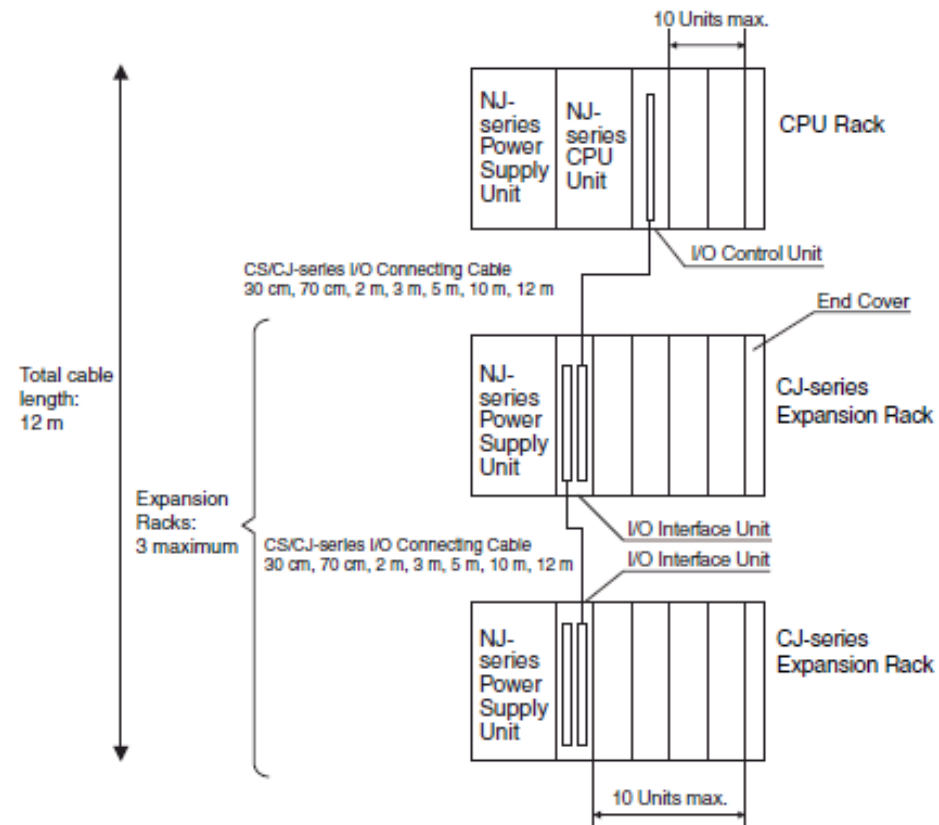
第一章 NJ系列的结构

- 分支器型号
- GX-JC03 3口
- GX-JC06 6口
- 欧姆龙分支器内置DC功能



第一章 NJ系列的结构

- NJ系列使用单元模块为CJ系列的单元模块
- NJ系列每个机架上最多支持10块模块
- NJ系列最多支持4个机架
- 整个系统最长距离12米
- 部分CJ模块不支持



第一章 NJ系列的结构

- EtherNet/IP简介
- EtherNet/IP不仅是控制器与控制器之间的网络，同样可以用于现场网络，另外由于使用标准的以太网技术，所以可以通用许多传统以太网的仪器设备

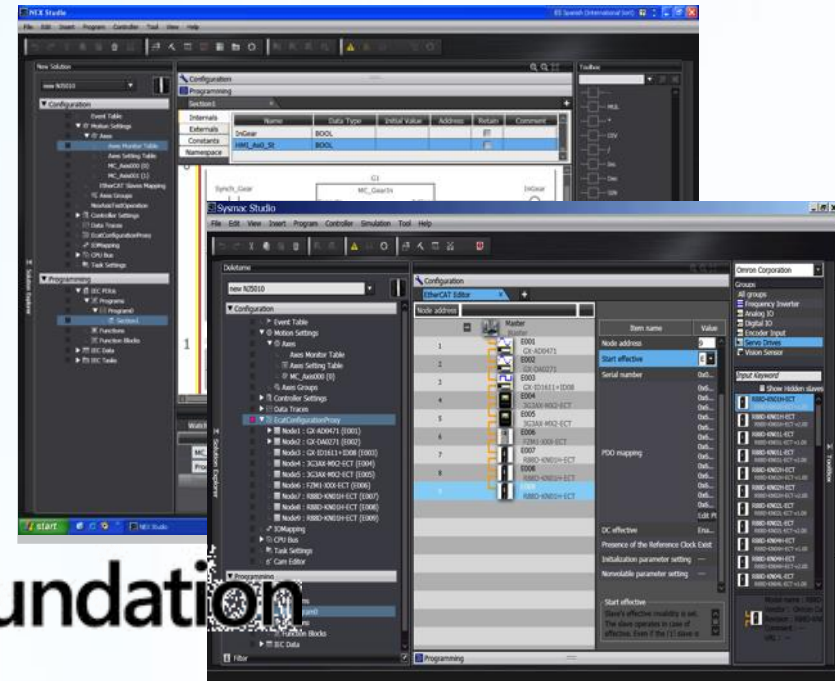


第一章 NJ系列的结构

- NJ系列EtherNet/IP网络特点
- 可以实现EtherNet/IP 通讯
- 网络变量更新周期时间和任务周期时间同步
- 标签中可以设置奇数字节
- 通过Tag data link实现灵活的通讯
- Auto IP功能
- 配置了专用的Socket通讯用功能块
- 支持NTP功能

第一章 NJ系列的结构

- Sysmac Studio是NJ系列PLC对应的软件
- 实现高速高效的开发
- 高度集成的测试环境
- 多种对象的高效反复利用率
- 对开发者而言方便的设计



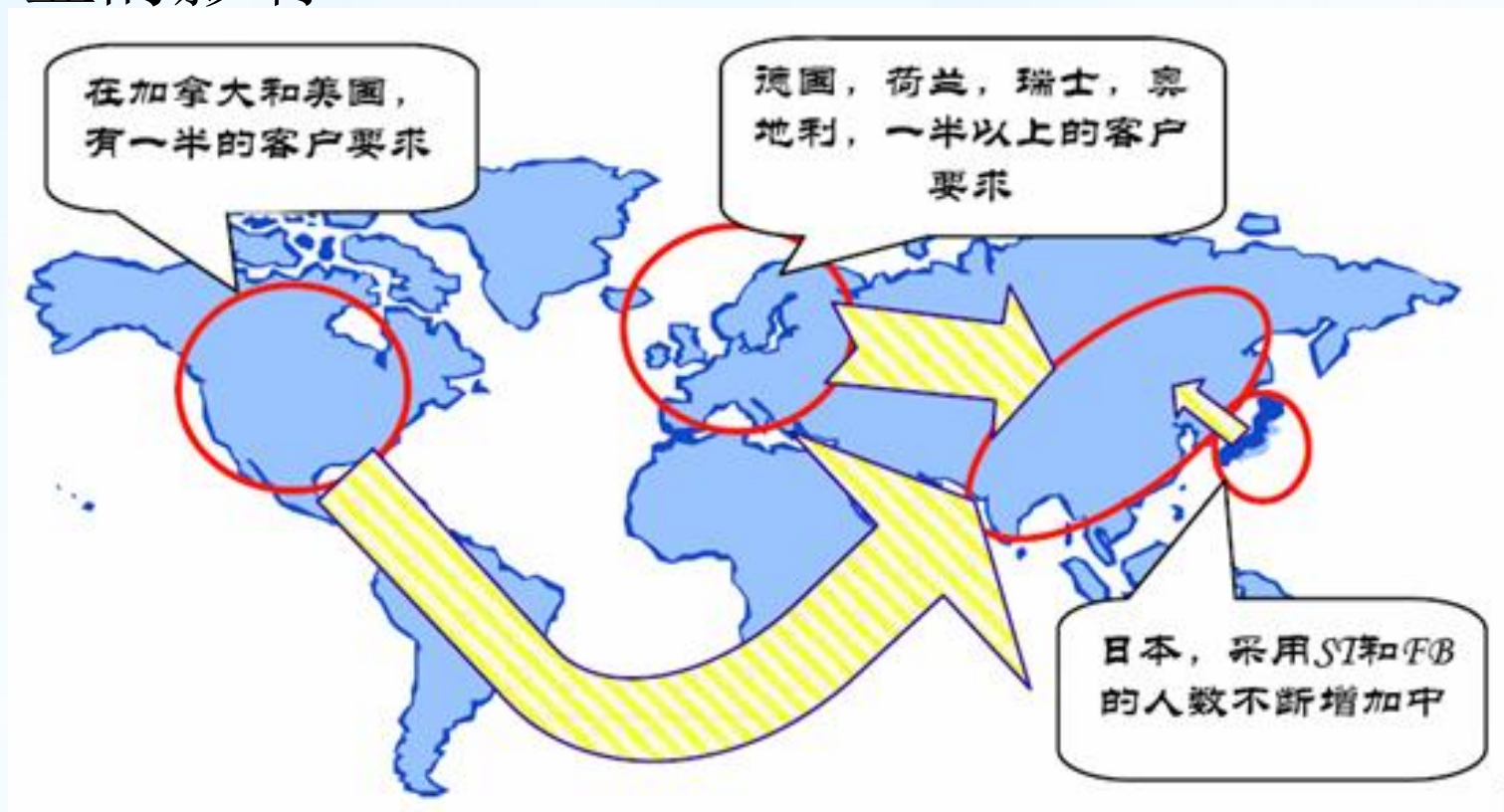
Windows
Presentation Foundation

第二章 软件概念

- 第一节 国际标准
- 第二节 变量
- 第三节 数据类型
- 第四节 程序
- 第五节 功能块 (FB)
- 第六节 功能 (FUN)
- 第六节 任务 (Task)

第二章 软件概念

- IEC61131-3国际标准
- 在全球竞争中，要求使用IEC61131-3标准的要求比以往增加许多，而在亚洲，也已经收到来自欧洲企业的影响

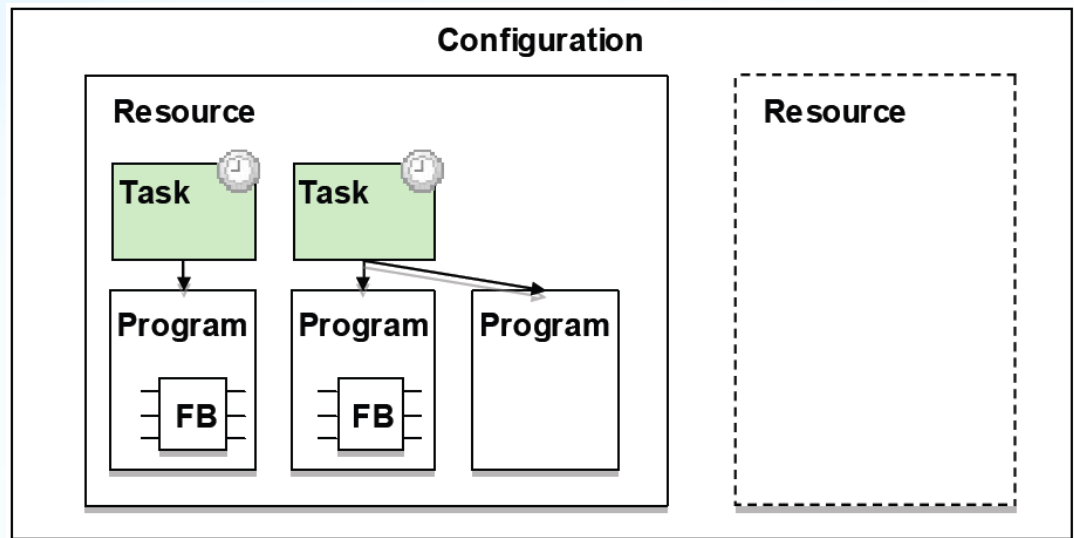


第二章 软件概念

- 现在就开始使用**IEC61131-3**的益处
- 可读性
- 可重复利用性
- 多种编程语言
- 可移植性（对于不同厂家而言）
- 可降低对应的教学经费
- 可对应学校的标准化教育
- 最终用户指定使用

第二章 软件概念

- 在 IEC61131-3标准下规定
- 将程序 (Program)与功能块 (FB), 功能 (Funtion)都基于一个POU (Program Organizatin Unit) 的概念下
- 调用POU就可以实现很高的重复利用率
- 在配置 (Configuration) 内, 你可以在Task里面添加 Program来做成一个运行程序。然后由资源 (Resource) 调用。



第二章 软件概念

- POU (Program Organization Unit)的概述
- POU (Program Organization Unit) 是基于 IEC61131-3的用户程序模块，它是建立一个用户程序的最基本单位，一般一个用户程序会由多个POU组成。
- POU的组件有以下三种类型。
- Programs程序
- Function Block (FB) 功能块
- Function(FUN)功能

第二章 软件概念

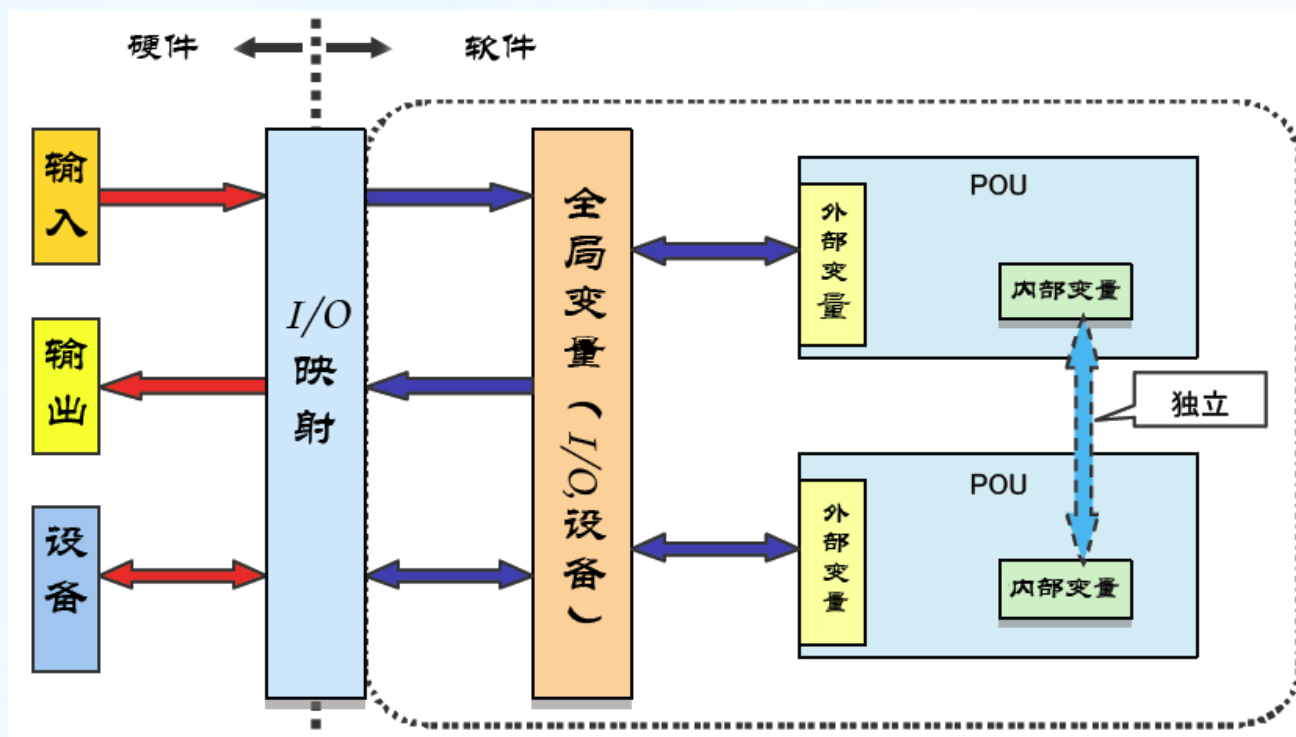
- **FUN**
- **FUN**是不需要设置名字的，因此输入很简单。
- **FUN**不占用内存。
- 没有数量限制。
- 当只需要简单的操作运算不需要对状态进行记忆的时候可以使用
- 一般只要输入相同的值就会输出一样的结果。
- **FB**
- **FB**需要设置名字，而且不同的**FB**要设置成不同的名字。
- 占用内存区，根据个数占用数量则不等。
- 当需要对一些状态进行记忆比如定时器指令，那么可以使用功能块。
- 能够实现即使是相同的输入也能产生不同的输出结果。

第二章 软件概念

- 变量
- **NJ**系列提供了一个完整的以变量为基础的编程环境
- **NJ**系列的变量比以往其他系列的变量在使用操作上更容易
- **NJ**系列的变量类型更全
- 变量需要分配一个固定的名字 (**name**)
- 最常见的对变量的叙述:变量=名字+类型+属性
- **NJ**的变量大致可以粗略分为全局变量 (**Global variables**)和本地变量(**Local variables**)

第二章 软件概念

- 对于每一个POU,在使用时需要登记变量, 登记在其变量表内的即是本地变量
- 具体FB和FUN内登记的本地变量会有所不同
- 不同的程序, 功能块, 功能的本地变量是不能彼此调用的
- 为了被所有的程序, 功能块, 功能调用, 此类变量被设置为全局变量
- 为了将硬件 (设备) 或者外部的信号可以被采用, 被转换成全局变量



第二章 软件概念

变量

- 变量名称。
- 数据类型。
- **AT**指定。
- 保持。
- 初始值。
- 常数。
- 网络公开。
- 微分。

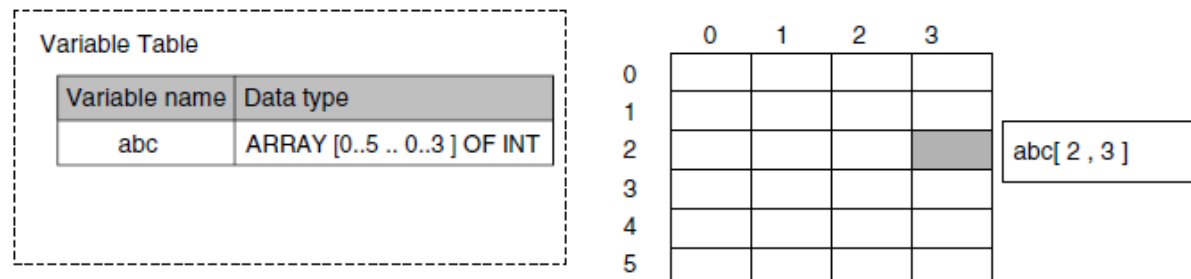
第二章 软件概念

- 数据类型
- 基本数据类型
 1. 布尔型
 2. 位字符串
 3. 文本字符串型
 4. 整数
 5. 无符号整数
 6. 实数型
 7. 持续时间型
 8. 日期型
 9. 时刻型
 10. 日期时刻型
- **FB**名变量
- 衍生数据类型（派生体，导出型）
 1. 结构体
 2. 共同体
 3. 枚举体

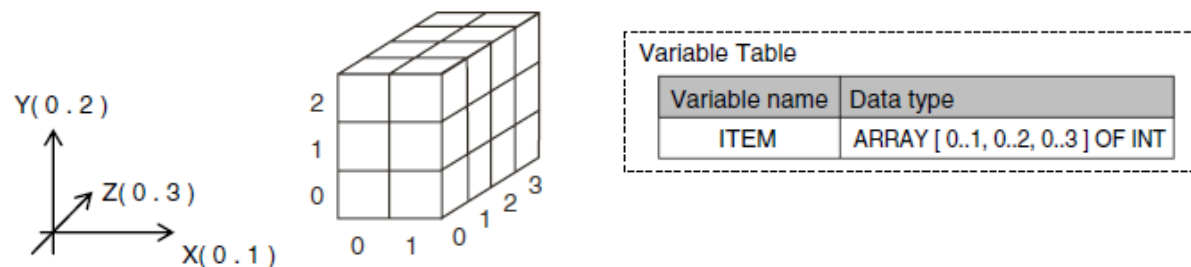
第二章 软件概念

- 数组的概念
- 数组可以将同样规格属性的数据合并在一起，然后作为一整个块来处理
- 最常见的运用是在运动控制中对于坐标的设定

Two-dimensional Array Specifications

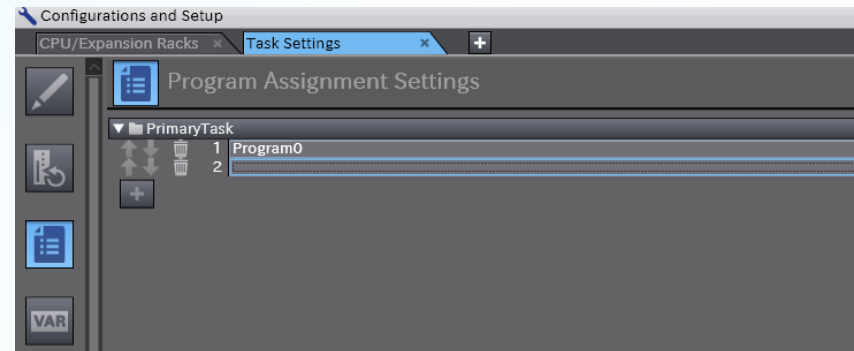
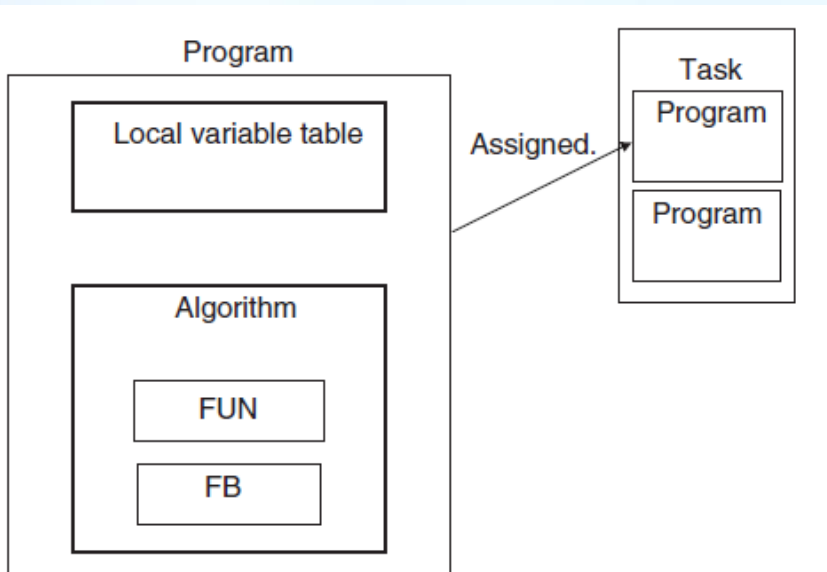


Three-dimensional Array Specifications



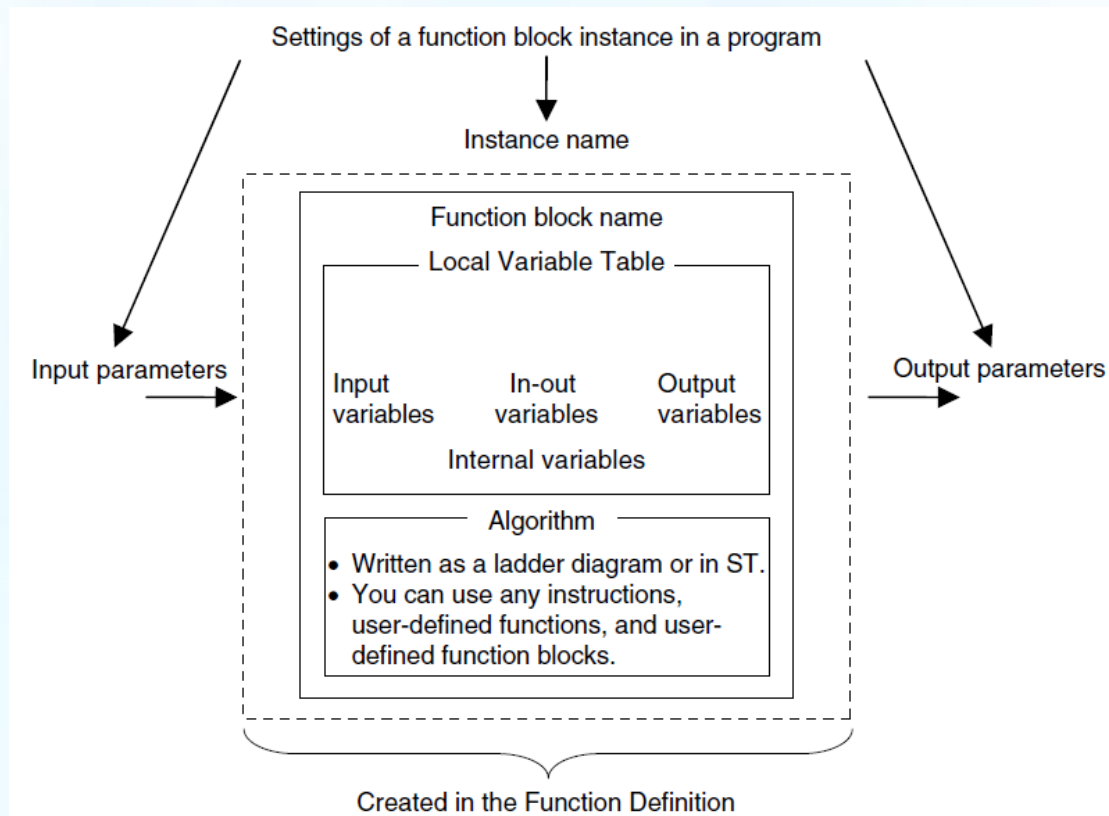
第二章 软件概念

- 程序的概念
- 程序包括本地变量表和算法，算法则会由功能和功能块来实现。
- 根据登记进的任务级别不同，执行的序列会有先后



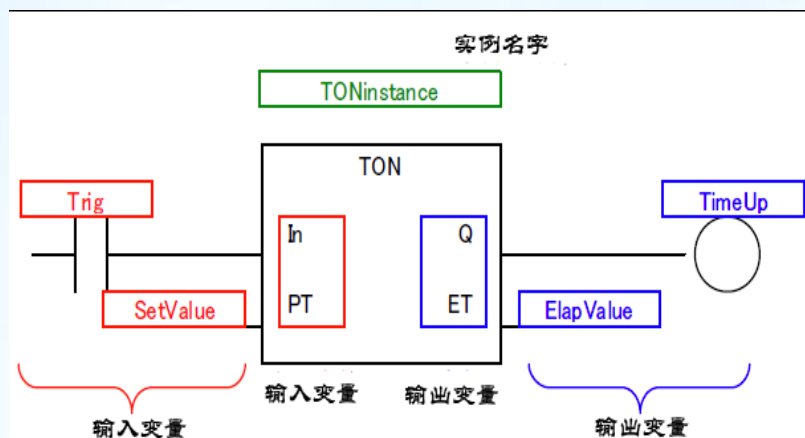
第二章 软件概念

- 功能块概述
- 功能块在使用的时候需要设定一个名字



第二章 软件概念

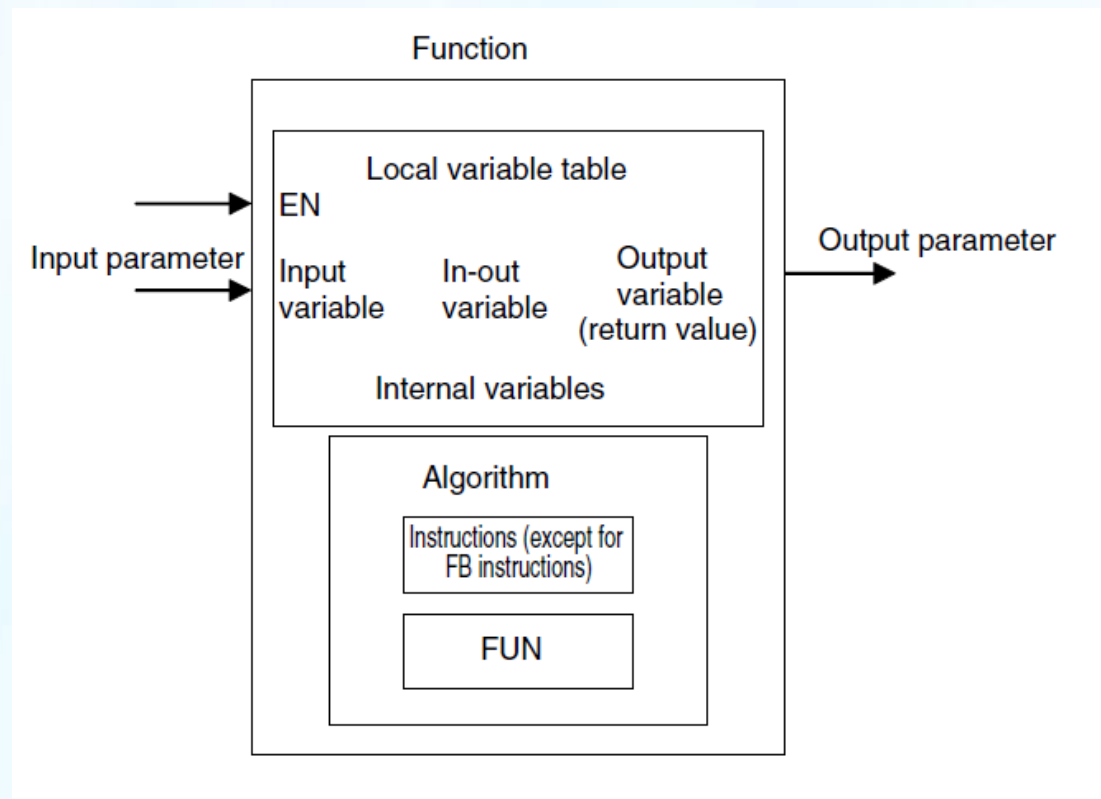
- 功能块在梯级编程环境
- 功能块在**ST**编程环境



```
1 TON_instance(In:=Trig, PT:=SetValue, Q=>TimeUp, ET =>ElapValue);
```

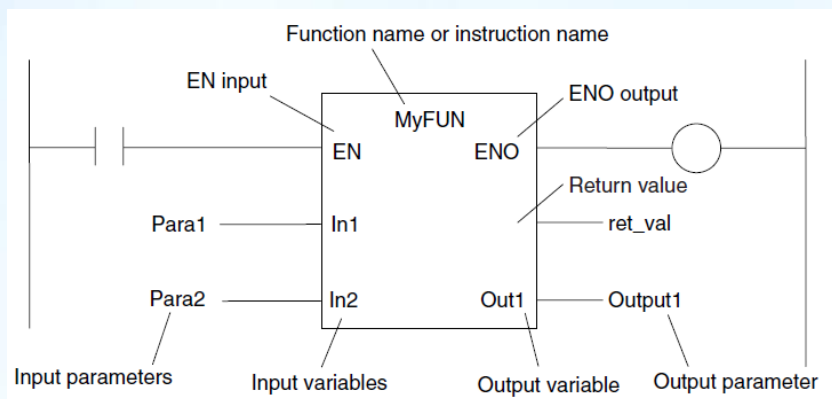
第二章 软件概念

- 功能概述
- 功能在使用的时候不需要设定一个名字



第二章 软件概念

- 功能在梯级编程环境
- 功能在ST编程环境



```
1 Result := MAX(EN:=Trig, In1:=Value1, In2:=Value2,  
2 ENO=>Done);  
3
```

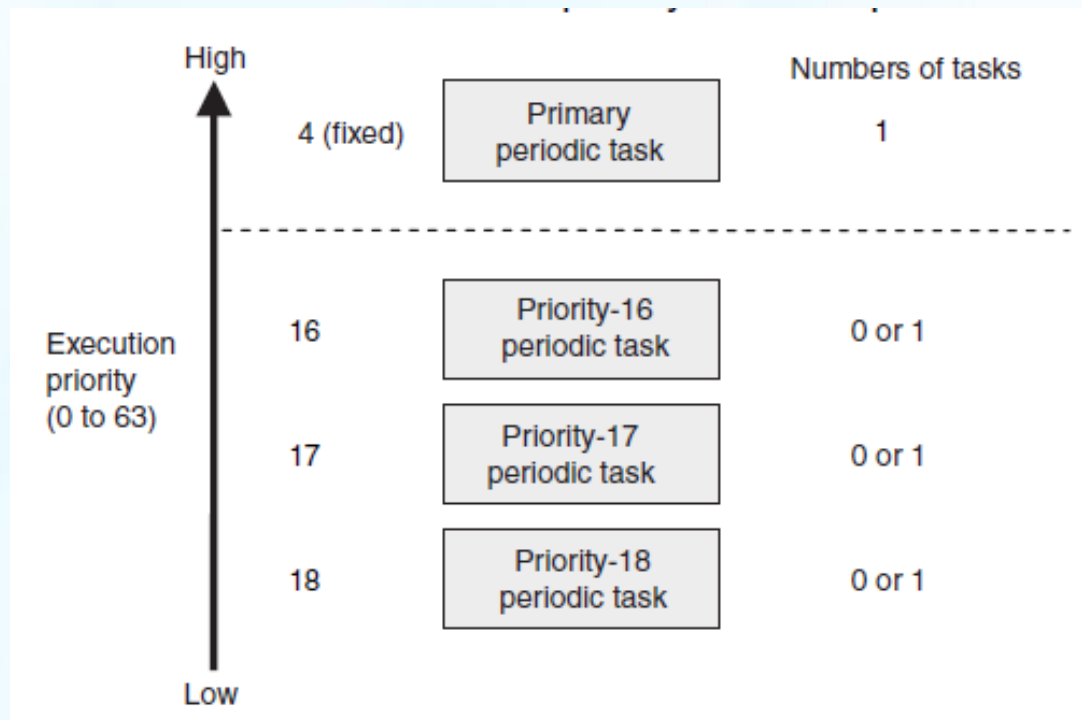
第二章 软件概念

NJ系列任务的特点

- 支持**4**个周期任务和最多**32**个事件任务。（目前）
- **EtherCAT**的通讯周期和任务的循环周期完全同步。
- 所有种类的任务周期时间都是主要任务的周期时间的倍数。
- 即便是低优先级的任务周期也能被保证。
- 不同的**I/O**被分配在不同的任务里执行。
- 可以设置不同的任务锁定。设置指令为**LOCK/UNLOCK**。

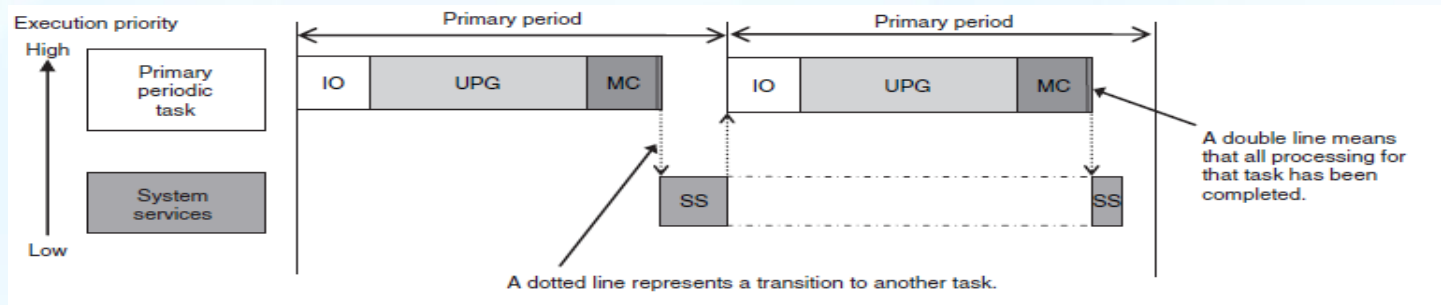
第二章 软件概念

- 任务根据任务号会有不同的优先级
- 任务分为主要周期任务（**Primary periodic task**）和周期任务（**Periodic task**）
- 任务号（或者说优先级）目前有4，16，17，18

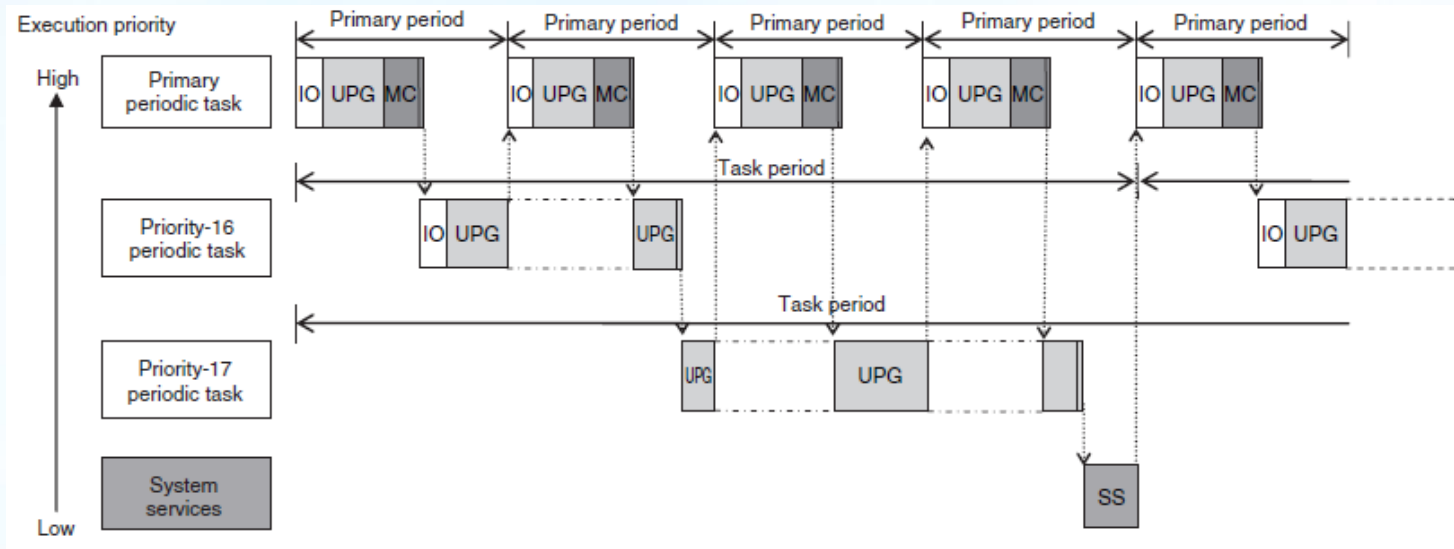


第二章 软件概念

主要周期任务的循环情况

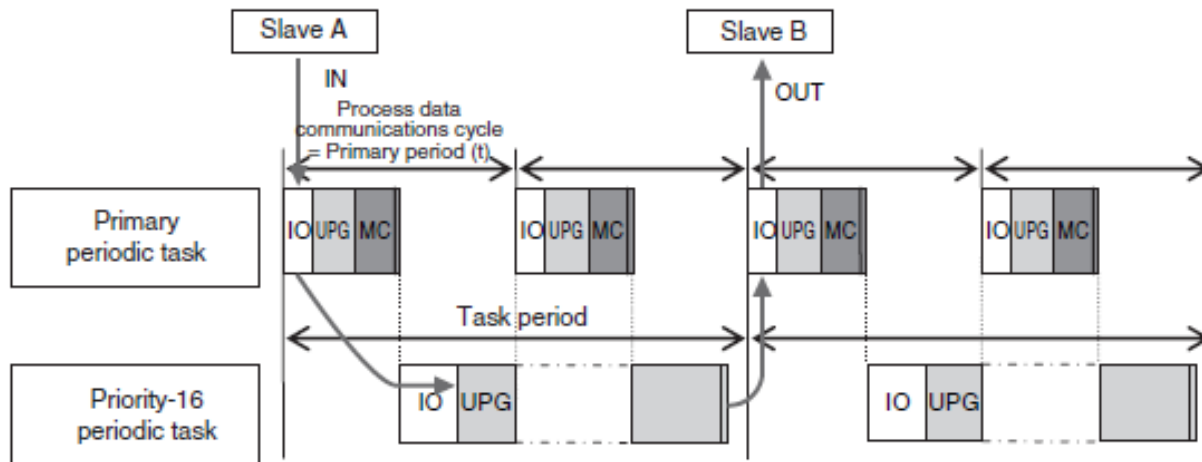


周期任务的循环情况



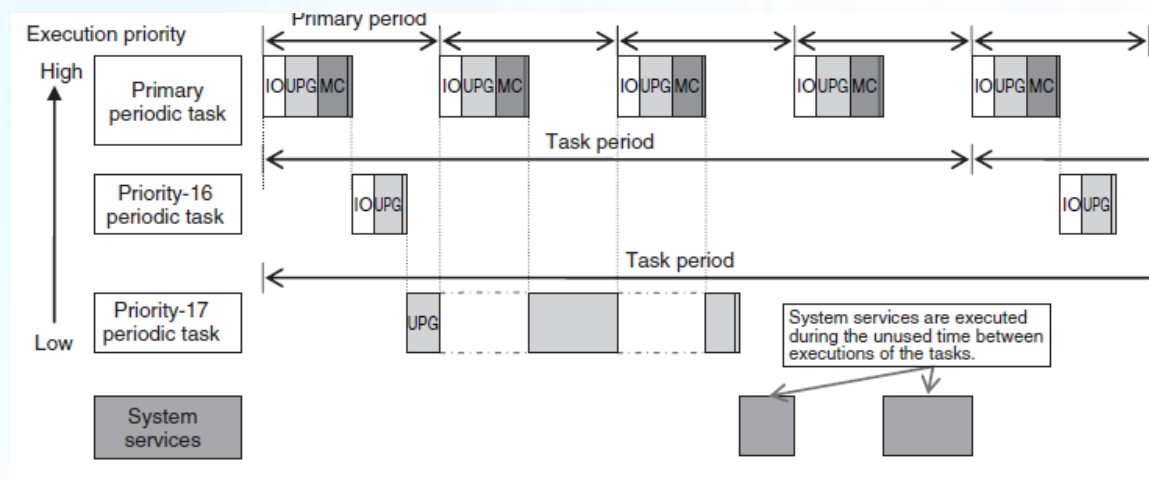
第二章 软件概念

- NJ系列的I/O刷新
- NJ的外部I/O情况分为来自模块与来自EtherCAT网络从站两种情况
- 根据客户的设置I/O刷新可以在主要周期任务内或是在16号周期任务内



第二章 软件概念

- NJ系列的系统服务
- 系统服务包含
- USB端口服务
- 内置EIP口服务
- CJ单元模块服务
- SD内存卡服务
- 自诊断功能

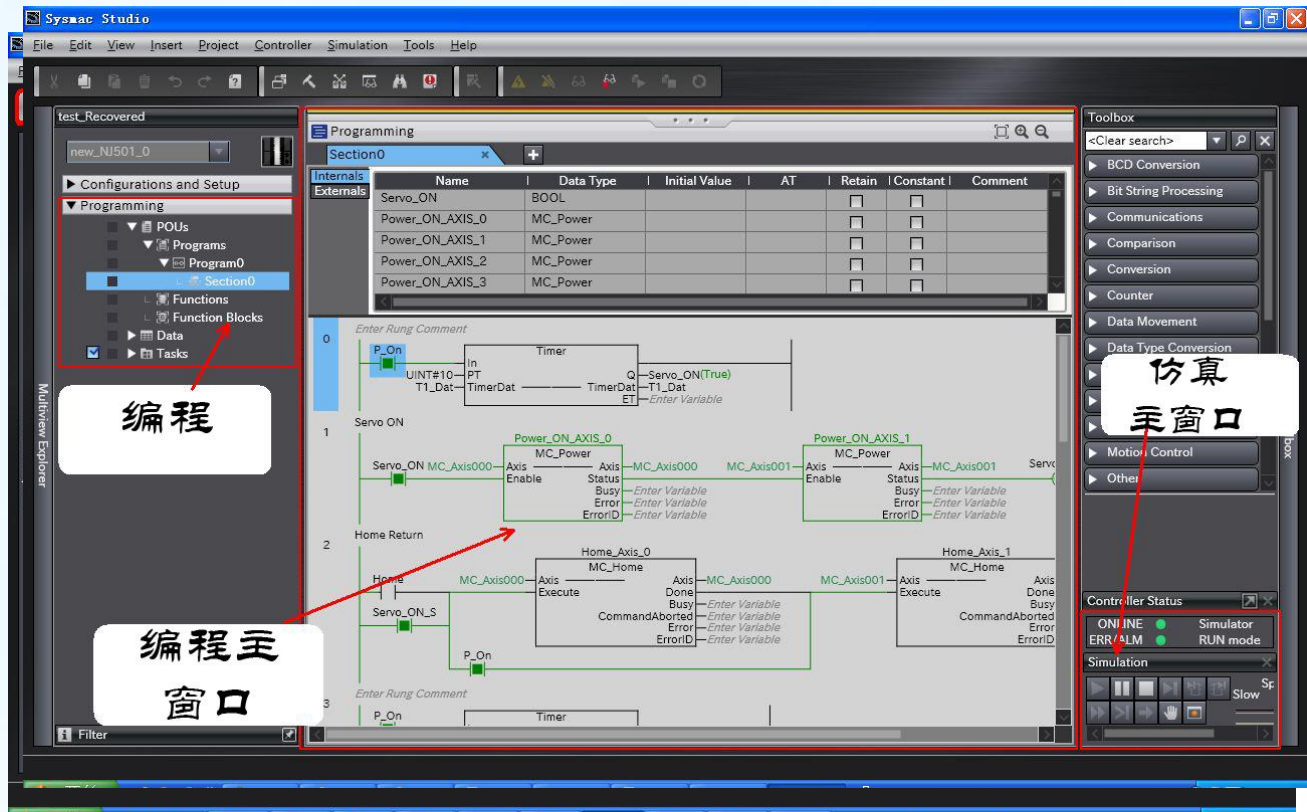


第三章 NJ系统的启动

- 第一节 Sysmac Studio的启动
- 第二节 CPU机架的构成和设定
- 第三节 控制器的设定
- 第四节 EtherCAT的设定
- 第五节 Motion控制的设定
- 第六节 I/O MAP的设定
- 第七节 POU的登录
- 第八节 变量的登录

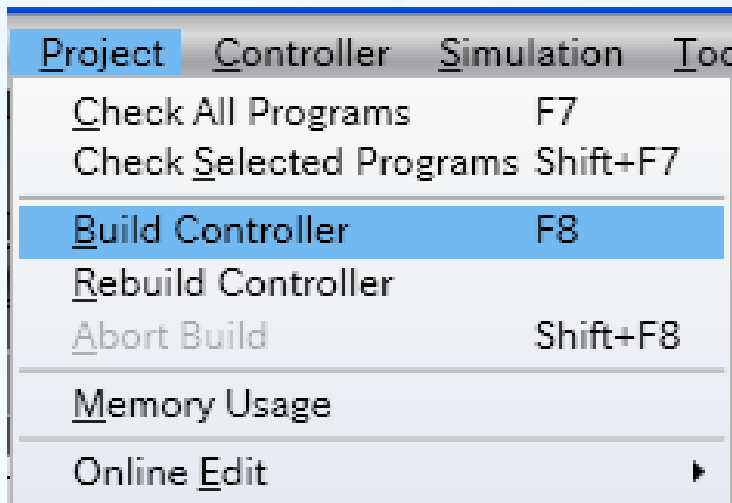
第三章 NJ系统的启动

- 启动软件及建立新项目
- 项目窗口的分布



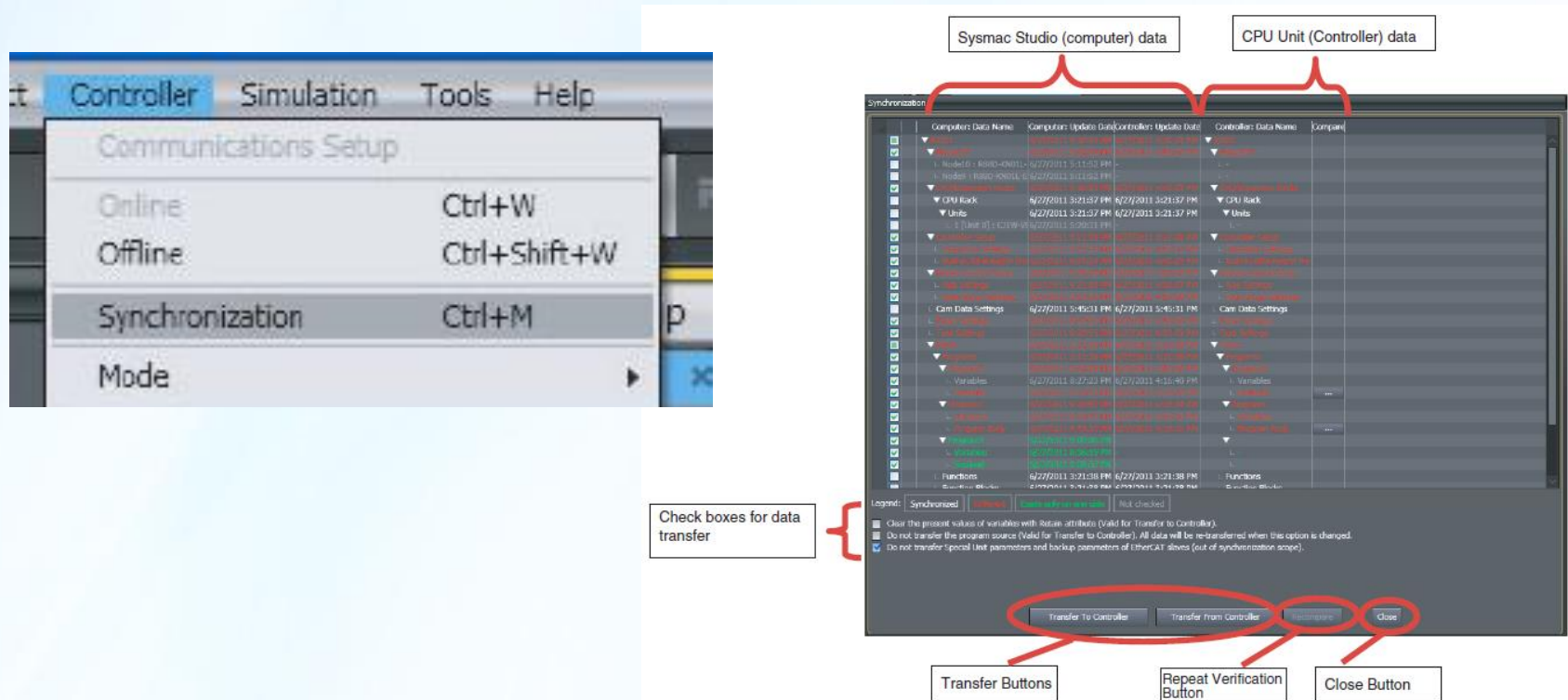
第三章 NJ系统的启动

- 建立(Build)的概念
- 建立的目的是为了将用户所编译的程序转换成CPU能够执行的格式。
- 当变更POU,数据类型, 全局变量5秒后建立会自动执行



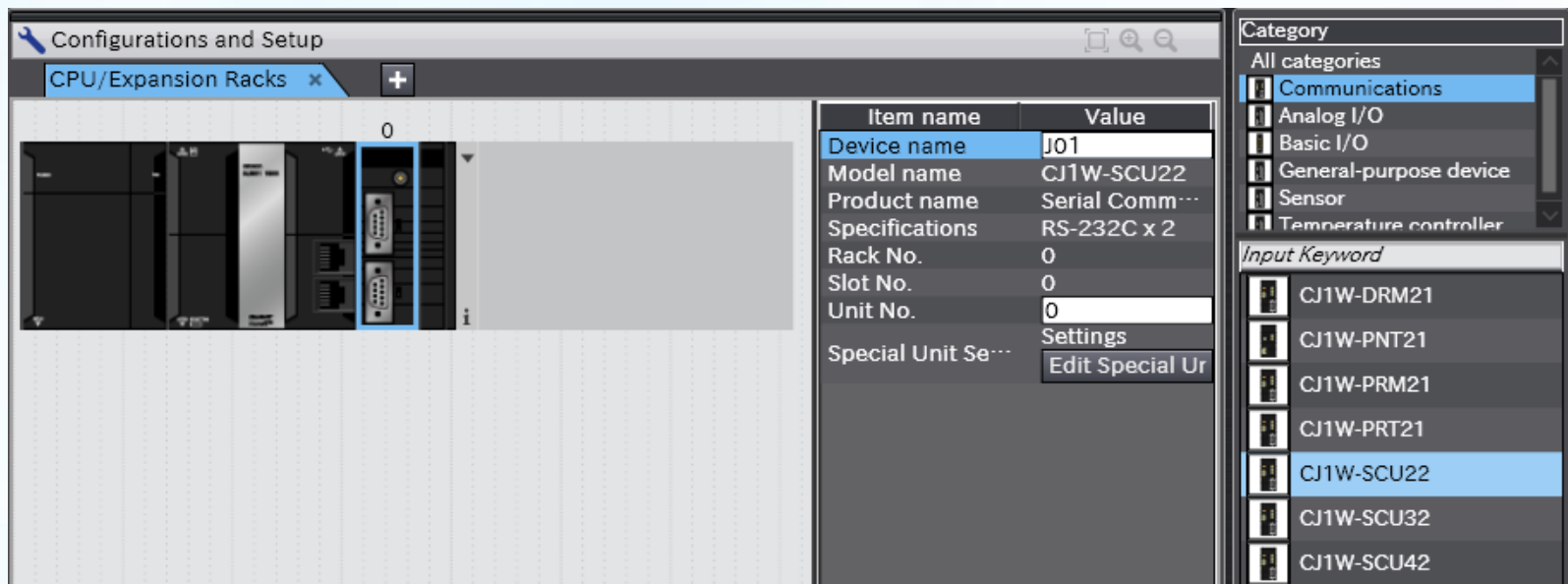
第三章 NJ系统的启动

- 同期/同期比较（Synchronization）
- 同期比较功能是将Sysmac Studio的数据和NJ系统控制器的数据进行比较的一种功能。



第三章 NJ系统的启动

- CPU机架的构成和设定
 - (1) 双击CPU/Expansion Racks
 - (2) 插入单元
 - (3) 改变电源模块型号

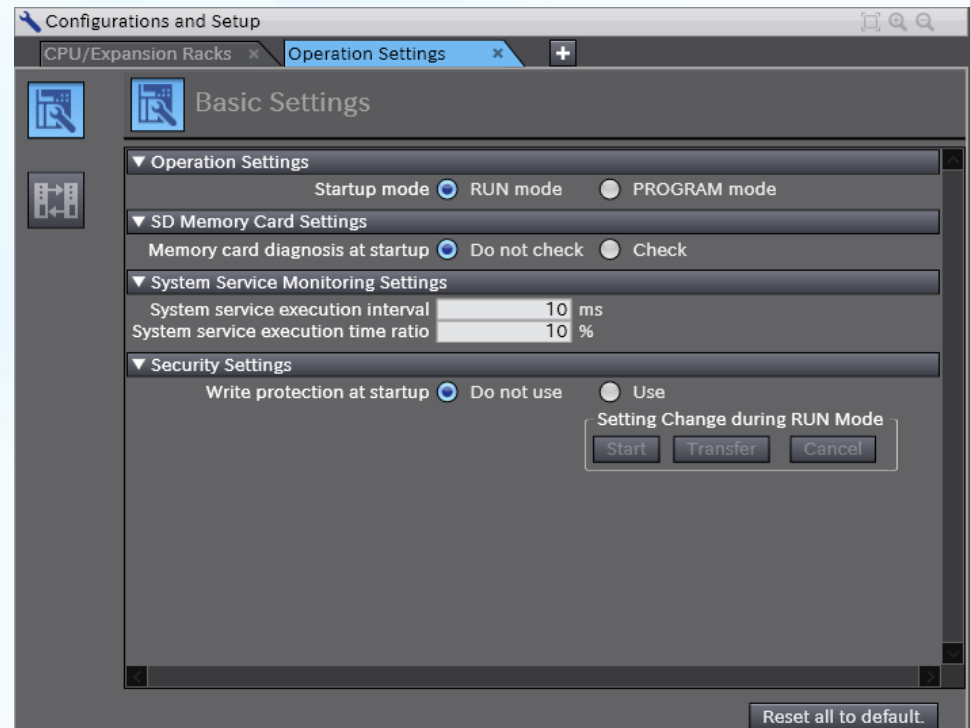


第三章 NJ系统的启动

- 单元的更改
- 单元的复制和粘贴
- 高性能单元的设定

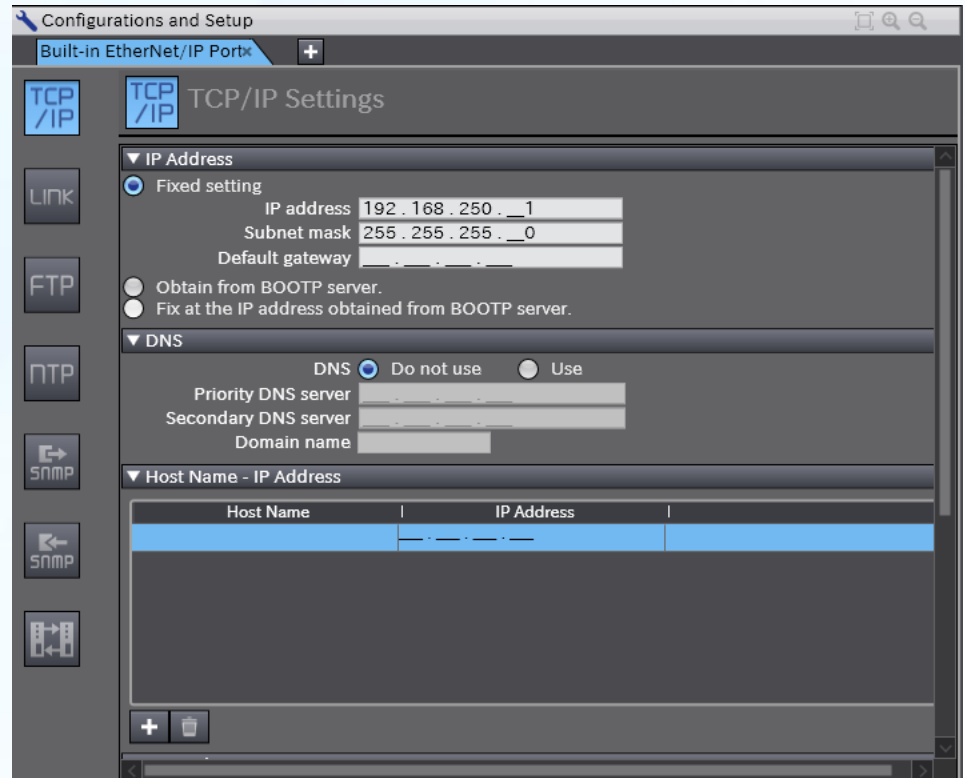
第三章 NJ系统的启动

- 控制器的设定
 1. 默认启动模式
 2. SD卡上电检测
 3. 系统服务监控时间及动作时间
 4. 写保护



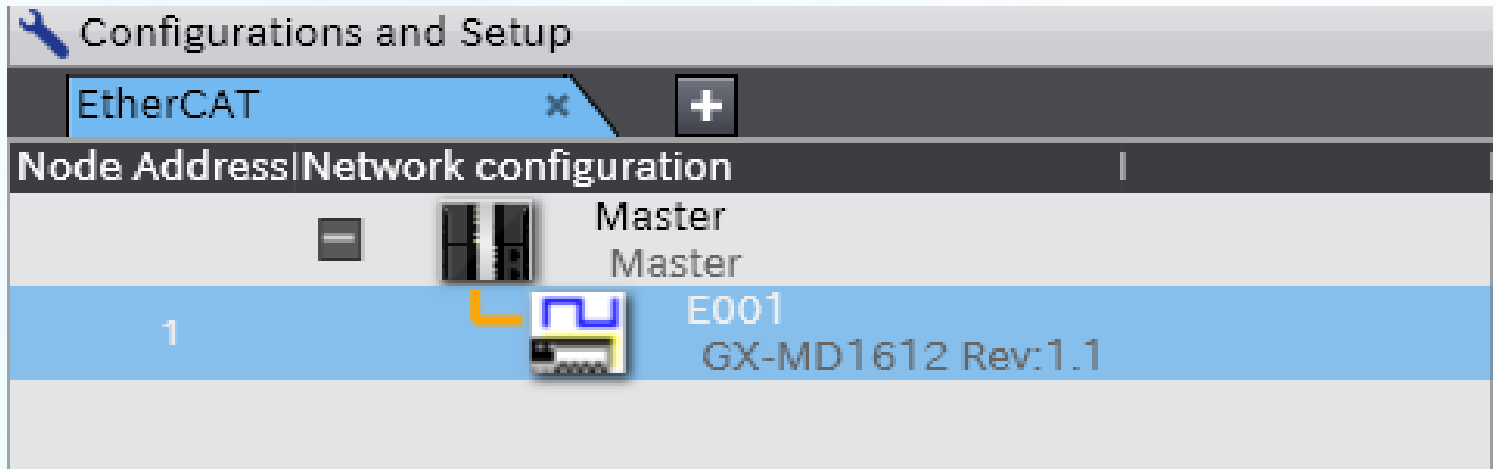
第三章 NJ系统的启动

- 内置EtherNet/IP口的设置
- TCP/IP口设置
- 连接设置
- FTP功能设置
- NTP服务设置
- SNMP功能设置
- SNMPTrap功能设置
- FINS设置



第三章 NJ系统的启动

- EtherCAT的设定
 - (1) 双击选择EtherCAT
 - (2) 从工具箱中选择需要的从站。
- 对于主站的设置
- 对于从站的设置
- 对于伺服的设置



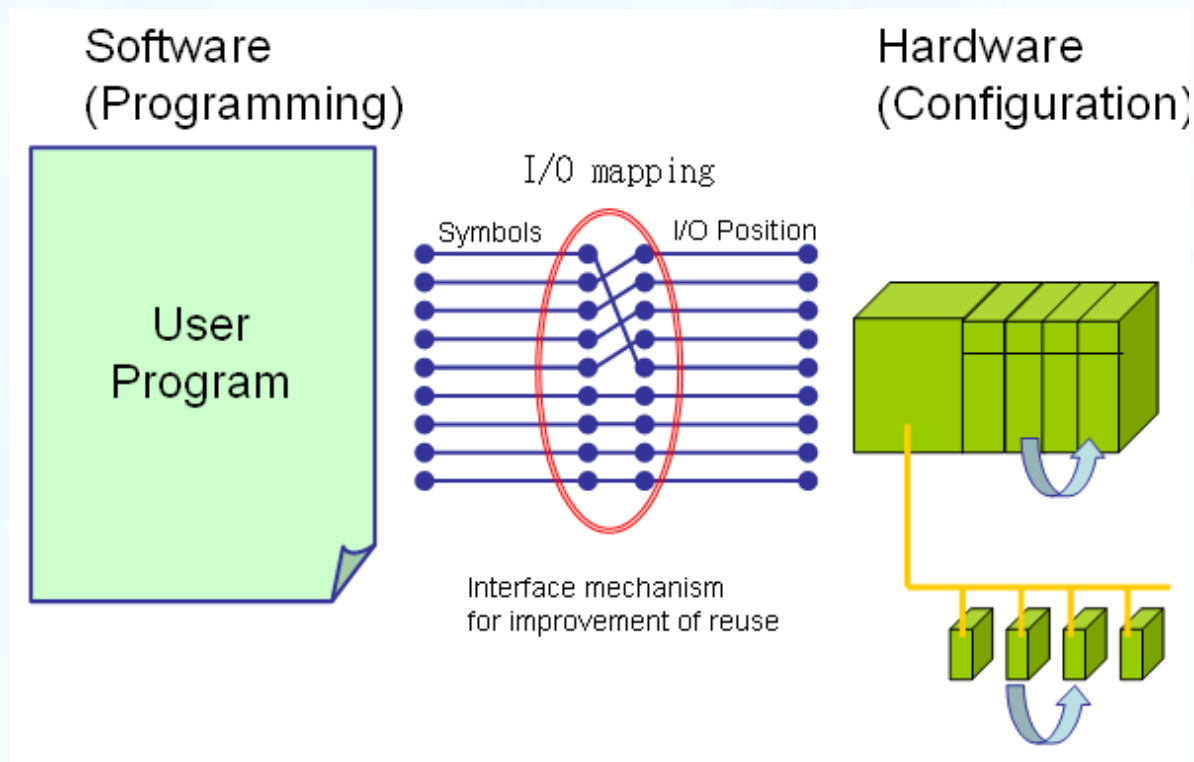
第三章 NJ系统的启动

- 运动控制的设定
- 轴的登记
- 轴组的设置

Axis number	0
Axis use	Used axis ▼
Axis type	Servo axis ▼
Feedback control	No control loop ▼
Input device	Node: 2 Device: R88D-KN01H-ECT ▼
Output device	<Not assigned> ▼
▶ Detailed Settings	

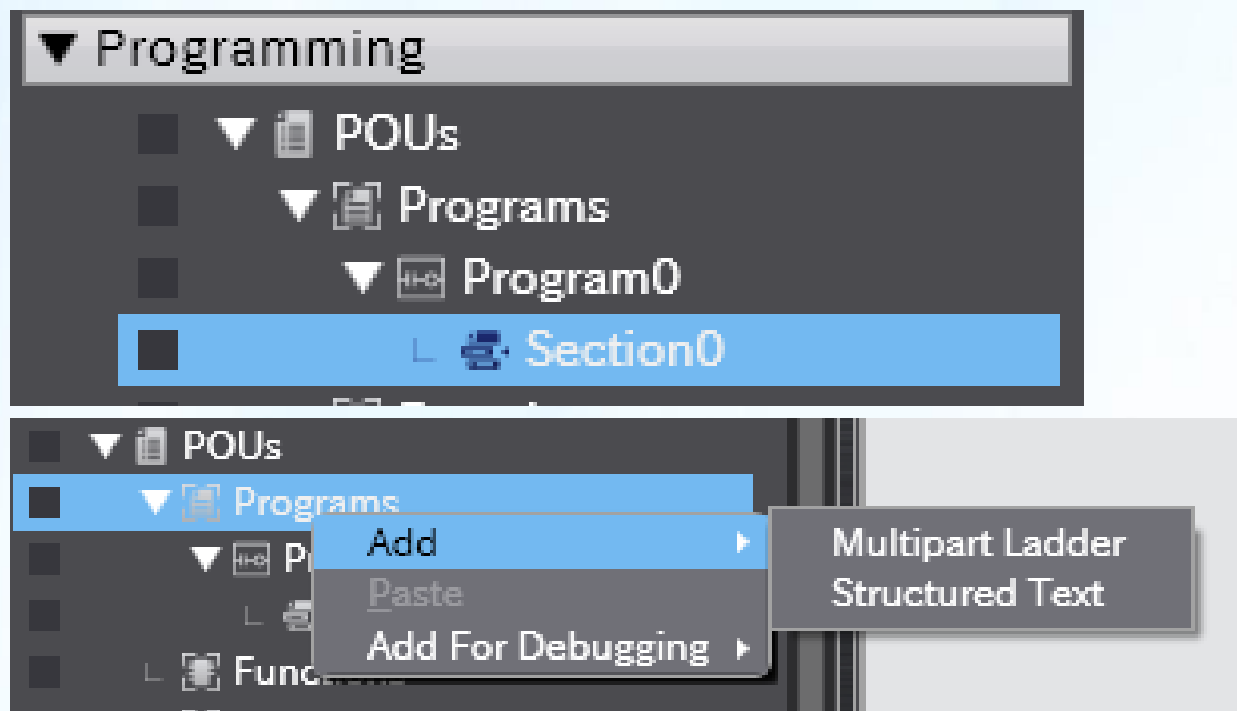
第三章 NJ系统的启动

- I/O MAP的概述
- I/O MAP是一种将软件和硬件联合起来的机制
- NJ系列的I/O MAP可以手动生成或者自动生成
- 生成的设备变量名称遵循一定的规律，从站为E,单元模块为J



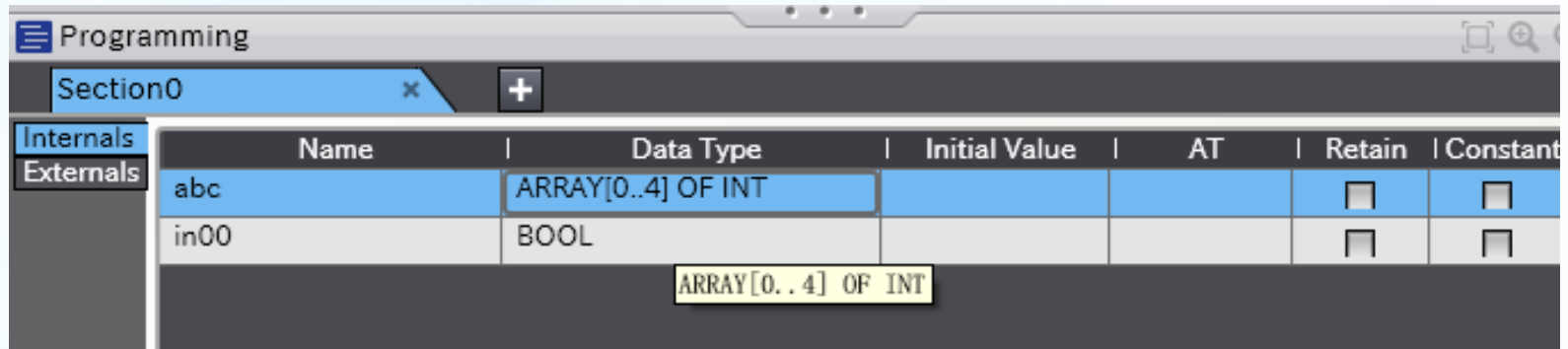
第三章 NJ系统的启动

- POU的登录
- NJ系列的POU由程序，功能，功能块组成



第三章 NJ系统的启动

- 变量的登录
- 本地变量的登录
- 本地变量中的外部变量（**Externals Variables**）登录后也会出现在全局变量表内
- 对于数组变量的登录



The screenshot shows the 'Programming' window with a tab for 'Section0'. On the left, there are two tabs: 'Internals' (selected) and 'Externals'. The main area displays a table of variables with the following columns: Name, Data Type, Initial Value, AT, Retain, and Constant.

Name	Data Type	Initial Value	AT	Retain	Constant
abc	ARRAY[0..4] OF INT			<input type="checkbox"/>	<input type="checkbox"/>
in00	BOOL			<input type="checkbox"/>	<input type="checkbox"/>

Below the table, there is a text box containing the declaration: `ARRAY[0..4] OF INT`.

第四章 逻辑编程（基础篇）

- 第一节 梯级基本指令的描述方法
- 第二节 **FUN**的使用方法
- 第三节 程序输入练习
- 第四节 程序检测
- 第五节 在线编辑

第四章 逻辑编程（基础篇）

- 梯级编程环境下程序执行是从上至下的原则
- 同一梯级的元素则遵循从左往右的原则
- 输入不能写在最右边
- 梯级不能为空
- **NJ**系列的梯级允许一个线圈后添加一个线圈
- 不需要一个开关接在一个线圈后面
- **NJ**系列允许线圈，功能，功能块前不添加开关输入



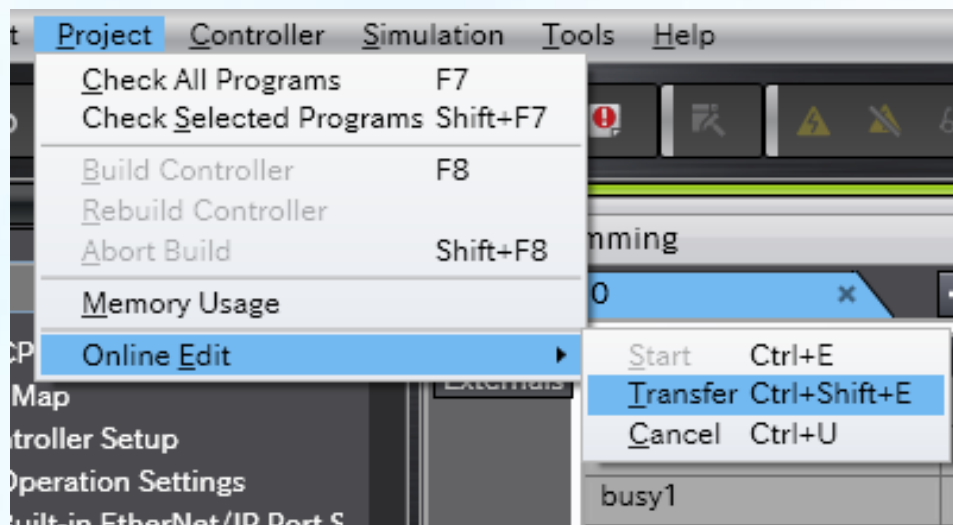
第四章 逻辑编程（基础篇）

简单程序练习

- (1) 新建项目
- (2) 配置机架
- (3) 分配I/O MAP
- (4) 输入程序
- (5) 程序检测
- (6) 同步后下载

第四章 逻辑编程（基础篇）

- 在线编辑功能
 - (1) 在线
 - (2) 选择在线编辑
 - (3) 修改后选择发送
- 备注 只有同一个POU下才能修改



第五章 逻辑编程

- 第一节 计数指令
- 第二节 定时指令
- 第三节 数据处理指令I
- 第四节 数据处理指令II

第五章 逻辑编程

- 微分指令

(1)R_TRIG

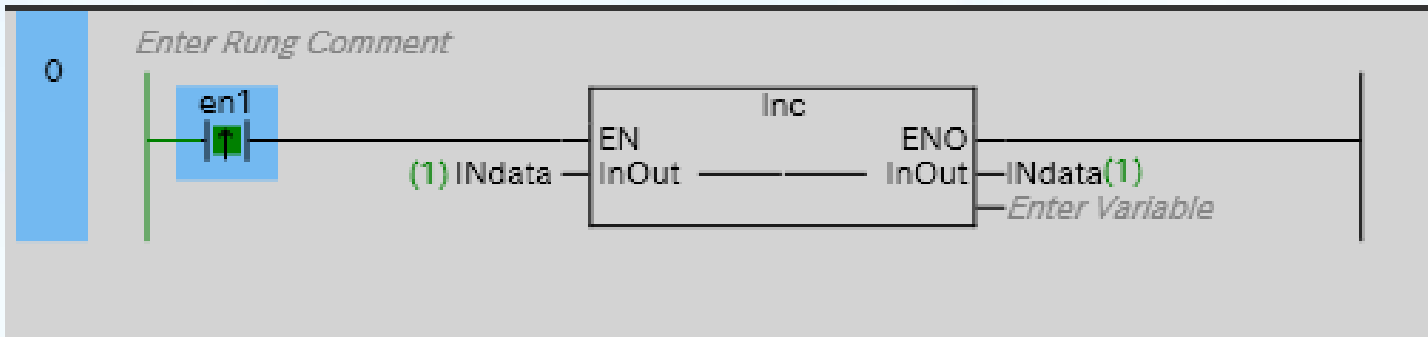
(2)F_TRIG

(3)微分符号



第五章 逻辑编程

- INC自加指令
- 使用微分的输入参数
- 不同的数据类型范围



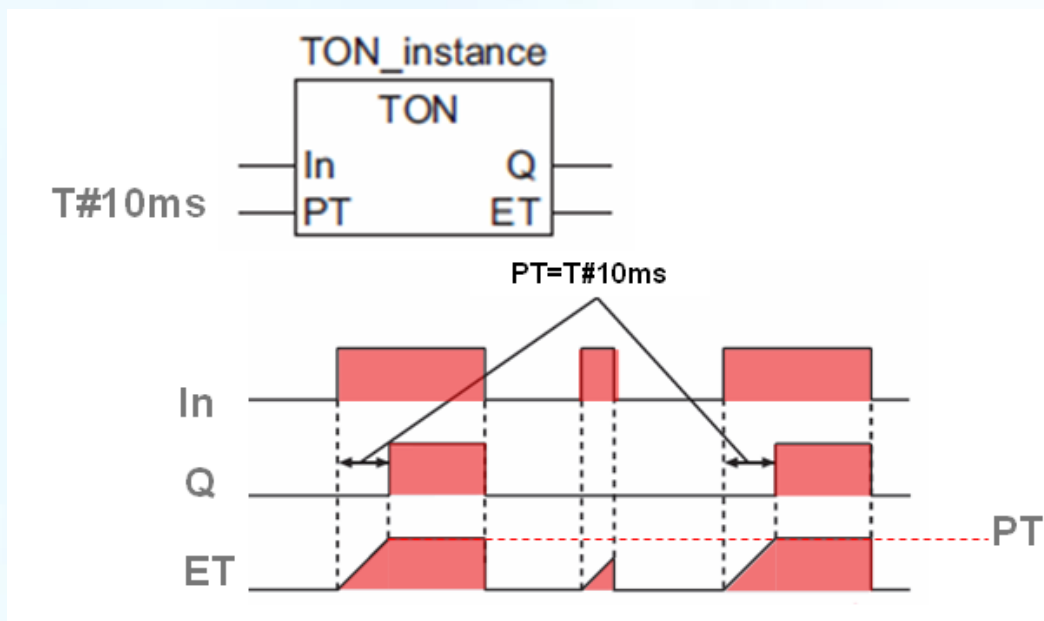
	Boolean	Bit strings					Integers							Real numbers	Times, durations, dates, and text strings					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InOut						OK	OK	OK	OK	OK	OK	OK								
Out	OK																			

第五章 逻辑编程

- 基本定时指令
- Timer/TON（注意ET值计算方式不同）
- 其他定时指令
- TOF
- TP（Pulse Timer）
- Accumulation Timer(Retentive Timer)累加定时器

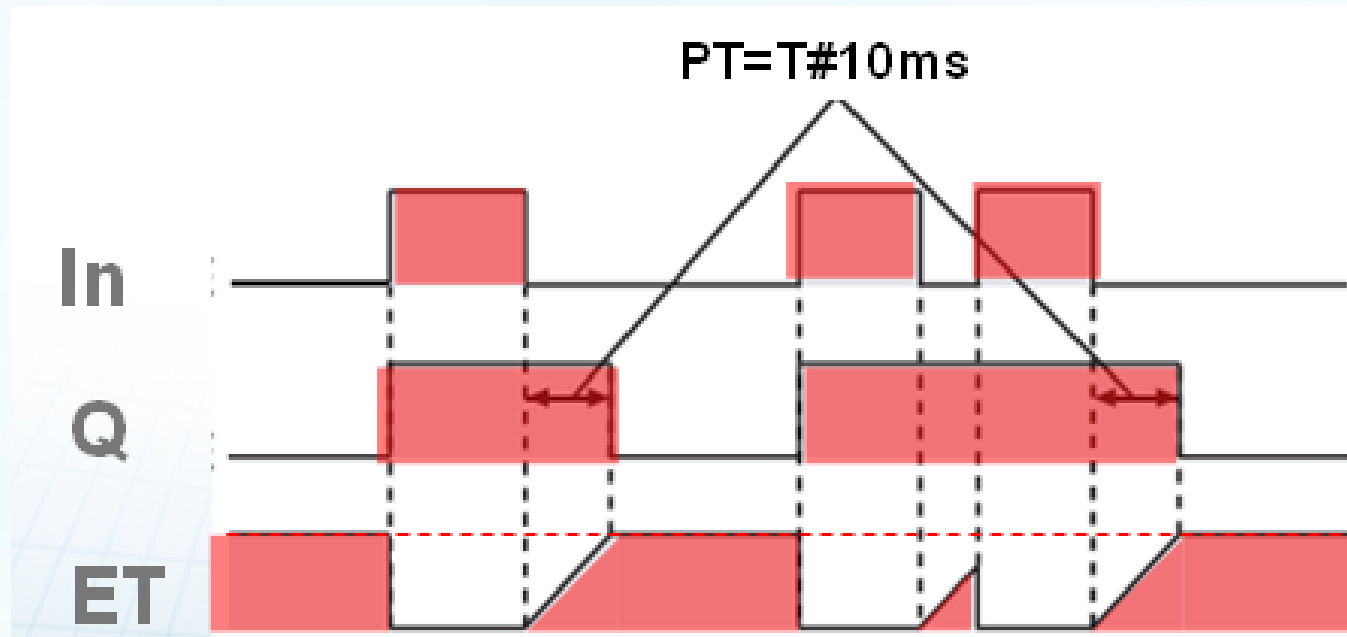
第五章 逻辑编程

- TON指令
- TON指令是定时器输入信号ON延时指令
- 当ET时间到达PT设定值的时候，输出Q上的变量转为TRUE。
- 当输入IN 信号切断后，输出Q的变量转为FALSE。



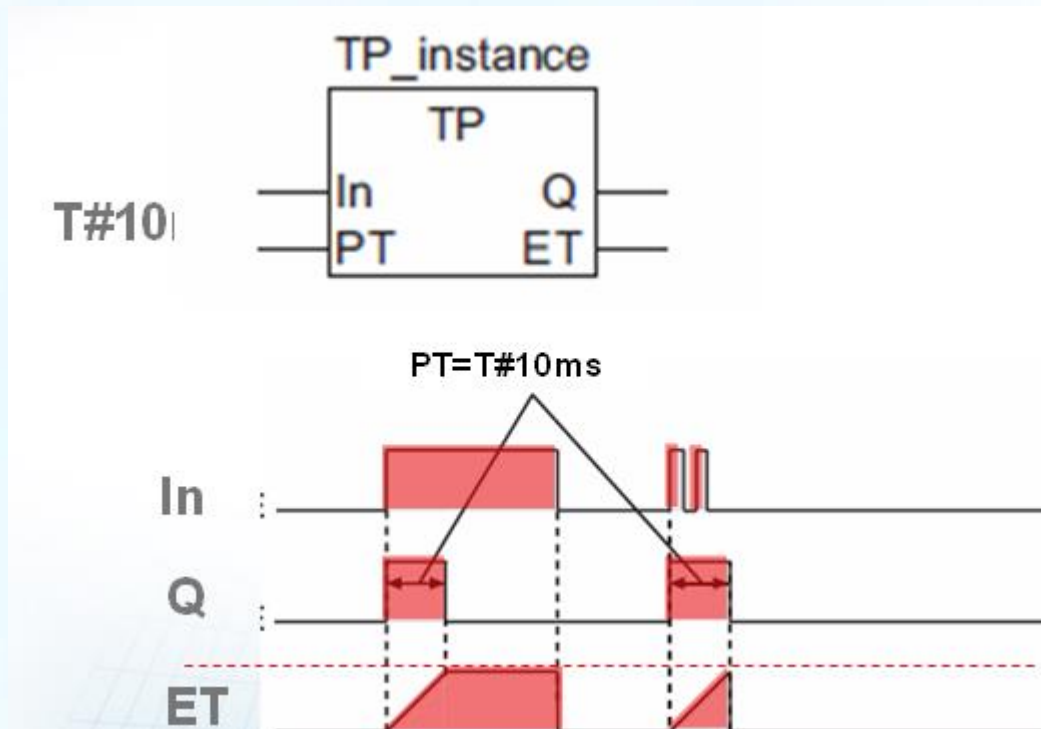
第五章 逻辑编程

- TOF
- OF指令是当输入信号ON后Q专为TRUE,当输入信号OFF后开始计时, 到达PT值后,Q断开输出转为FALSE。



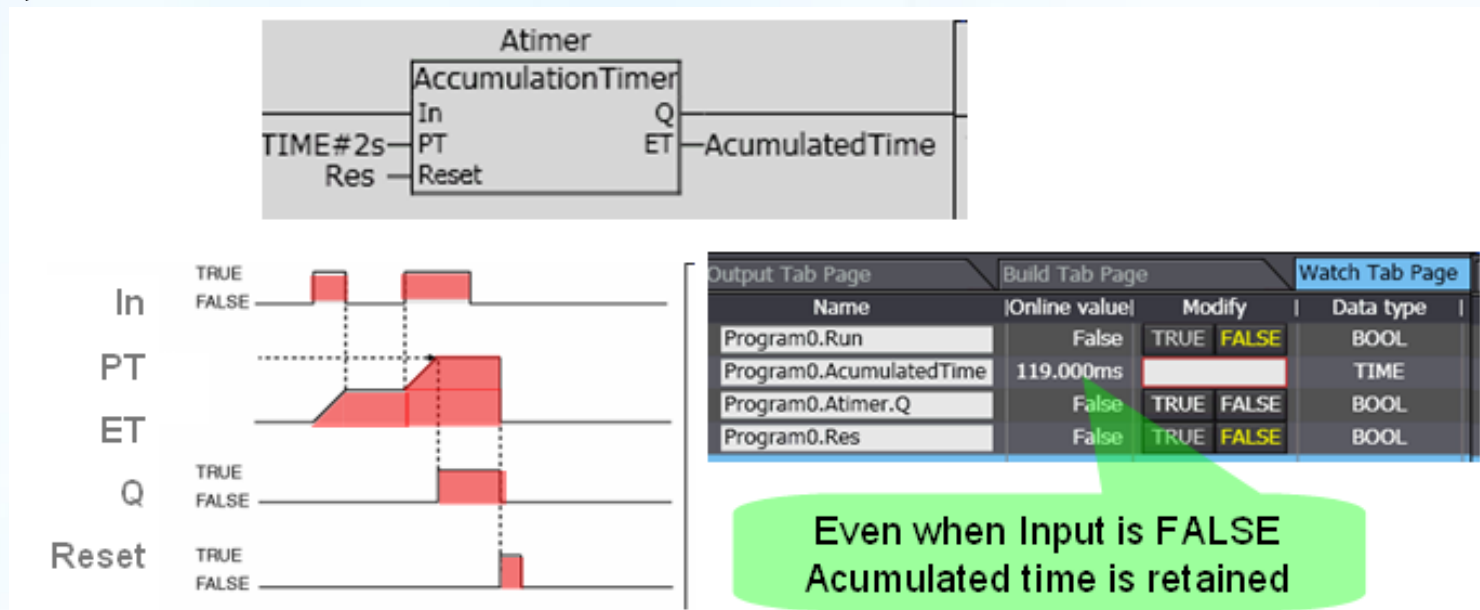
第五章 逻辑编程

- TP(Pulse Timer)
- 当输入In信号触发后，Q保持一个脉冲信号，脉冲宽度由用户设置。



第五章 逻辑编程

- AccumulationTimer (Retentive Timer)
- 和普通TON指令的区别在于，当输入IN信号断开后，ET值依然保持，因此当IN继续输入后，能够保持。当ET值等于设定值后，Q转为True



第五章 逻辑编程

- 基本数据传输指令
- MOVE
- MoveBit
- MoveDight
- 其他数据传输指令
- TransBits
- MemCopy
- SetBlock
- Exchange
- AryExchange
- AryMove
- Clear
- Copy**ToNum
- Copy**To**
- CopyNumTo**

第五章 逻辑编程

MOVE指令

MOVE指令能够将输入参数复制到输出参数。

	Boolean	Bit strings					Integers							Real numbers		Times, durations, dates, and text strings				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	LINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
	An enumeration, array, array element, structure, or structure member can also be specified.																			
Out	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
	Must be the same data type as <i>In</i> if <i>In</i> is an enumeration, array element, structure, or structure member. Must be an array with the same data type, size, and subscripts if <i>In</i> is an array.																			

0 Enter Rung Comment

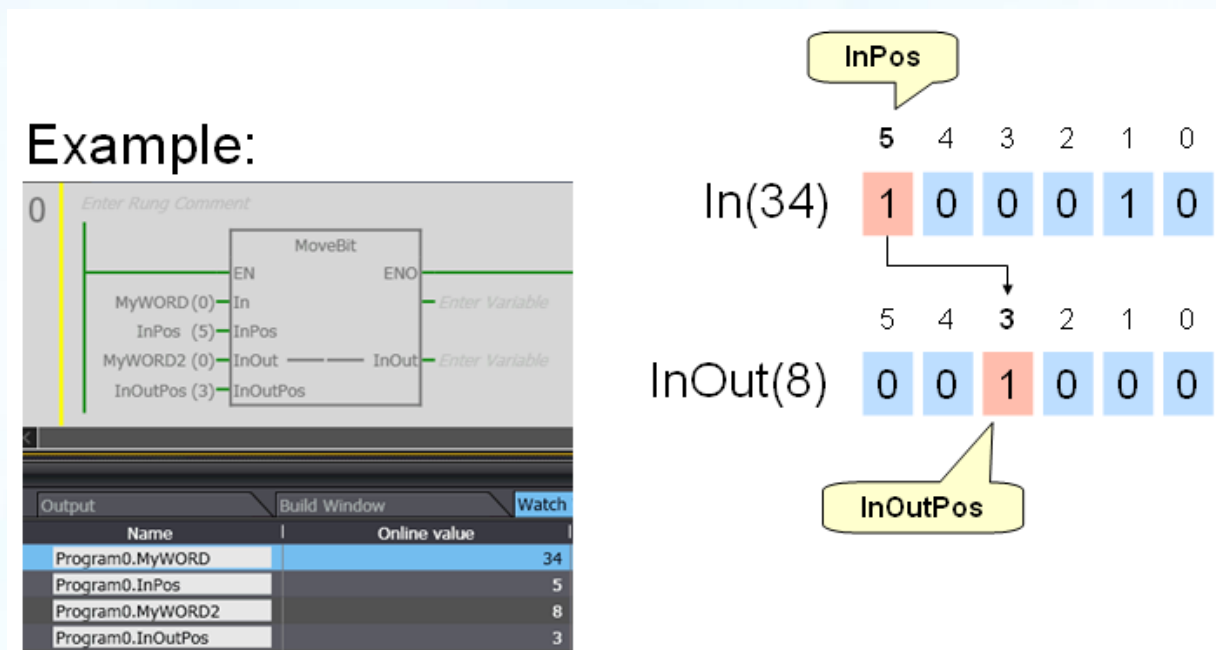
Name	Base Type
BOX	STRUCT
BOX_ID	INT
WEIGHT	REAL

Output Build Window Watch Window

Name	Online value	Modify	Data type
Program0.Box1.WEIGHT	29.2999992370605	29.3	REAL
Program0.Box2.WEIGHT	29.2999992370605		REAL
Program0.Box1.BOX_ID	3	3	INT
Program0.Box2.BOX_ID	3		INT

第五章 逻辑编程

- MoveBit
- MoveBit指令能将作为输入的位字符中某一个指定位传递给输出的位字符指定位。



第五章 逻辑编程

- MoveDigit
- MoveDigit指令能将位字符数据类型变量内部一个或更多个数字转移给指定的另一个对象，并且可以决定起始通道。

Example:

The diagram illustrates the MoveDigit instruction in a ladder logic program. The instruction is triggered by a 'DoIt' contact. It moves data from 'MyWORD (4660)' to 'MyWORD2 (8960)'. The 'InPos' parameter is set to 1, and the 'InOutPos' parameter is set to 2. The 'DSize' parameter is set to 2. The 'In' and 'InOut' labels are shown on the right side of the instruction.

Diagram illustrating the MoveDigit instruction parameters and data flow:

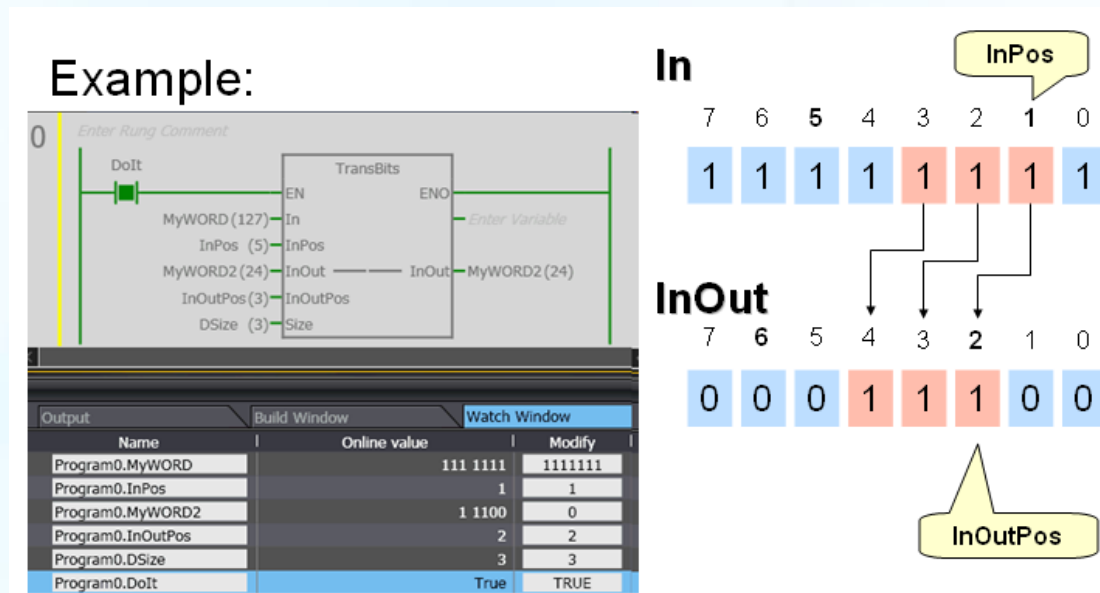
- InPos:** 3 2 1 0
- In:** 1 2 3 4
- InOut:** 3 2 1 0
- InOutPos:** 2 3 0 0

Watch Window:

Name	Online value	Modify	Data type	Data format
Program0.MyWORD	1234	1234	WORD	Hexadecimal
Program0.InPos	1	1	USINT	Decimal
Program0.MyWORD2	2300	0	WORD	Hexadecimal
Program0.InOutPos	2	2	USINT	Decimal
Program0.DSize	2	2	USINT	Decimal

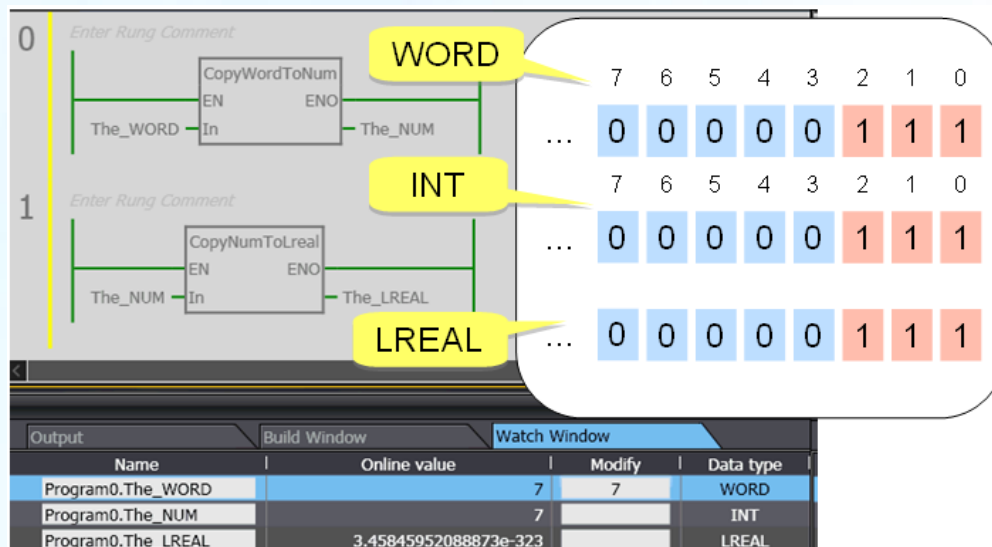
第五章 逻辑编程

- TransBits
- TransnBits指令能将指输入上位字符串数据类型变量的一个或多个位复制给输出位字符串数据类型变量上的一个或者多个位。



第五章 逻辑编程

- Clear
- Clear指令会将值清空
- Copy*To*
- 可以根据用户的设置在不同的数据变量之间，但是必须是同样数量级。



第五章 逻辑编程

- 四则运算指令

(1) 基本算法指令。

(2) 三角函数类指令。

(3) 平方、LOG类指令。

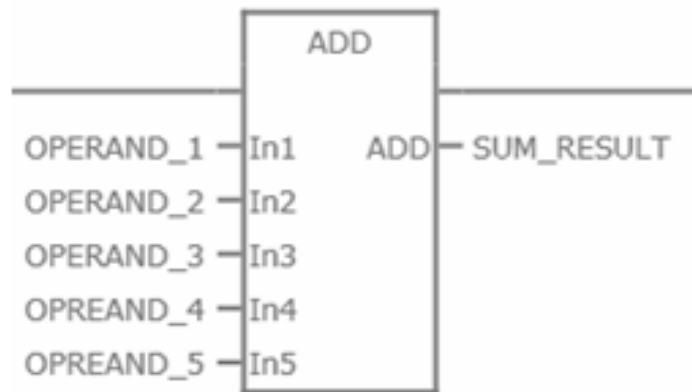
(4) 数组管理类指令。

BASIC ARITMETIC	POWER, LOGARITHMIC, RANDOM, FRACTIONS	ARRAY MANAGEMENT
ADD (+) AddOU (+OU) SUB () SubOU (OU) MUL (*) MulOU (*OU) DIV (/) MOD ABS Inc and Dec	SQRT LN and LOG EXPT Rand ModReal Fraction CheckReal	AryAdd AryAddV ArySub ArySubV AryBCD AryBin AryMean ArySD
TRIGONOMETRIC		
RadToDeg and DegToRad SIN, COS, and TAN ASIN, ACOS, and ATAN		

第五章 逻辑编程

- 基本算法指令举例
- **ADD**等同于+指令
- 支持字符串变量

Example (ADD):



Name	Data Type
IAM_	ANY_ELEMENTARY
SUM_RESULT	ANY_ELEMENTARY
OPERAND_1	ANY_ELEMENTARY
OPERAND_2	ANY_ELEMENTARY
OPERAND_3	ANY_ELEMENTARY
OPREAND_4	ANY_ELEMENTARY
OPREAND_5	ANY_ELEMENTARY

第五章 逻辑编程

- 三角函数指令举例
- COS指令

The screenshot displays a software interface for programming a PLC. At the top, a window titled "Section0" contains a table for internal variables:

Name	Data Type	Initial Value
d	LREAL	
e	LREAL	

Below the table, a ladder logic diagram for rung 0 is shown. It features a "COS" instruction block with "EN" (Enable) and "ENO" (Enable Out) terminals. The input terminal "In" is connected to variable "d", and the output terminal "Out" is connected to variable "e".

At the bottom, a monitoring table is visible, showing the online values of the variables:

Name	Online value	Modify
Program0.d	0	
Program0.e	1	

第五章 逻辑编程

- 平方LOG类指令举例
- EXPT指令

The screenshot shows the 'Programming' window in GX Developer. The 'Internals' table is visible, listing variables 'a', 'b', and 'c' with their respective data types and initial values. Below the table, a ladder logic diagram for the EXPT instruction is shown. The instruction is labeled 'EXPT' and has two inputs: 'In' (connected to variable 'a') and 'Pwr' (connected to variable 'b'). The output is labeled 'ENO' and is connected to variable 'c'. The diagram is labeled '0' and 'Enter Rung Comment'.

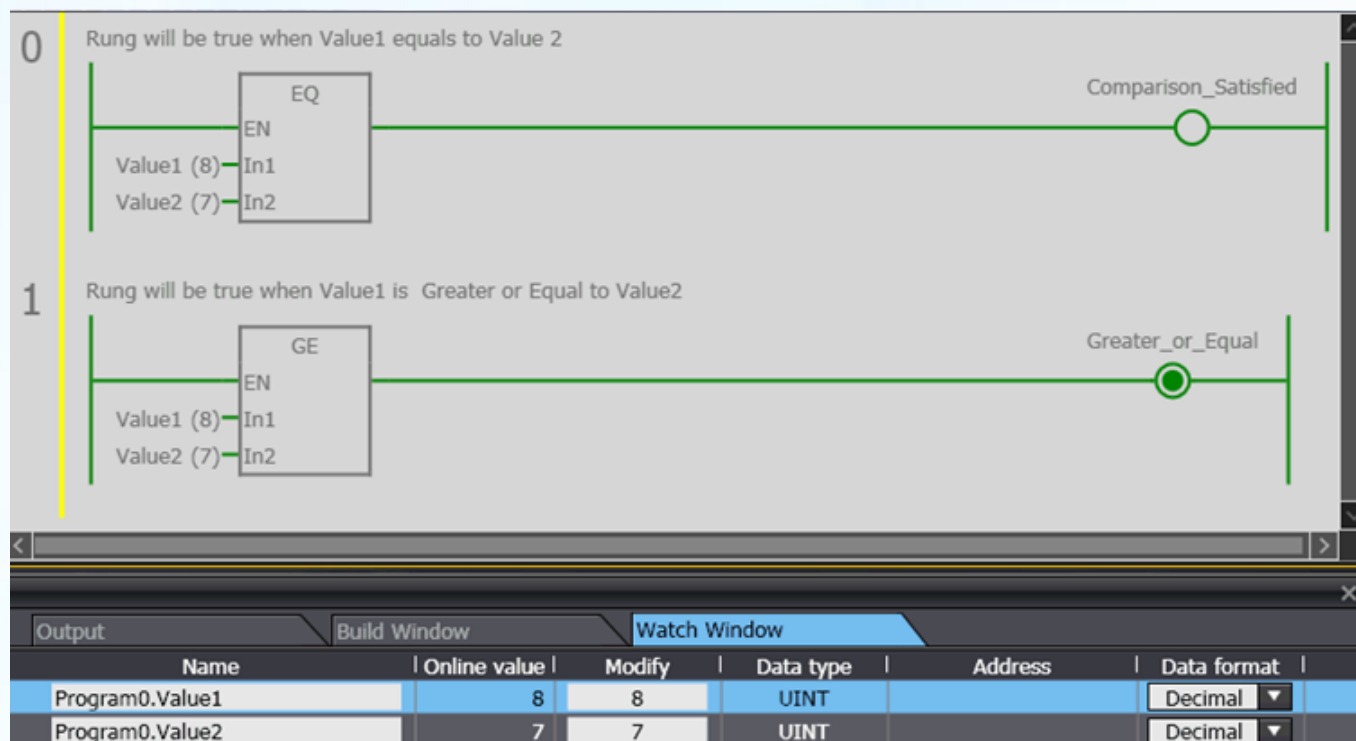
Name	Data Type	Initial Value
a	REAL	
b	real	2
c	LREAL	

The screenshot shows the 'Watch' tab page in GX Developer. It displays a table with three columns: 'Name', 'Online value', and 'Modify'. The table shows the online values for variables 'a' and 'c'.

Name	Online value	Modify
Program0.a	2	2
Program0.c	4	

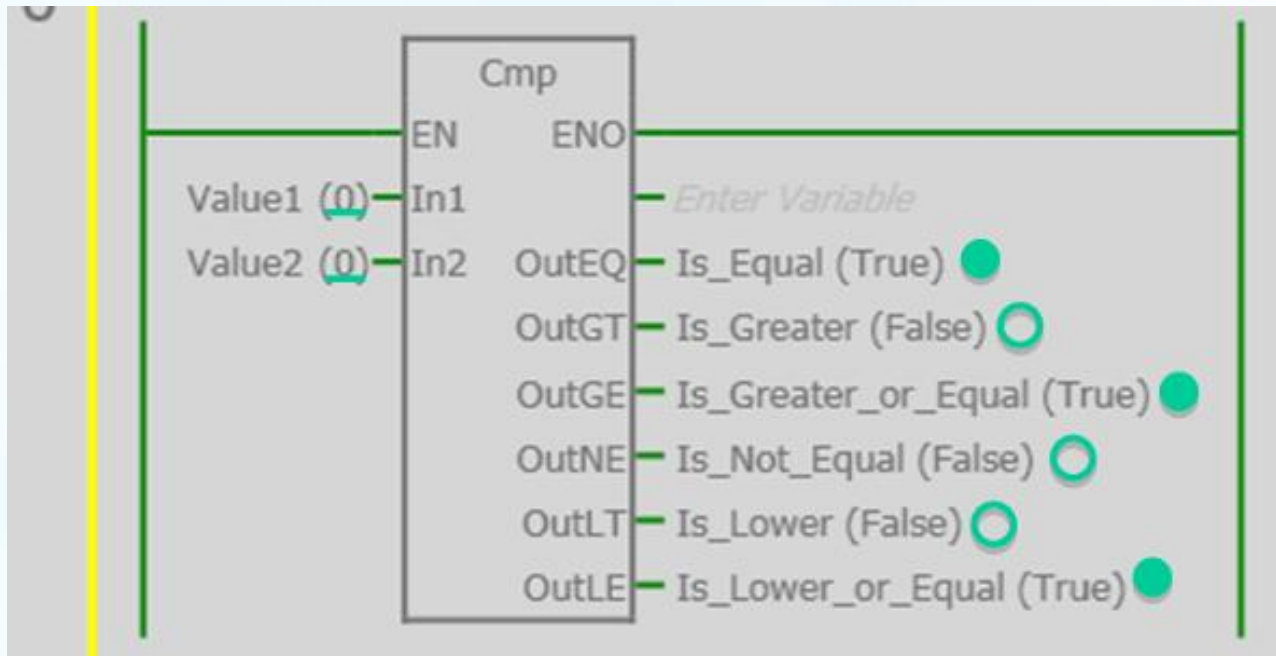
第五章 逻辑编程

- 比较类指令举例
- GE(>=)



第五章 逻辑编程

- Cmp指令
- NJ系列的CMP与CJ不同，现在会将所有的比较结果集合在一起



第五章 逻辑编程

- ZoneCmp
- 区域比较，当目标值处于范围内即输出结果
- 编程开发比以前更简单化

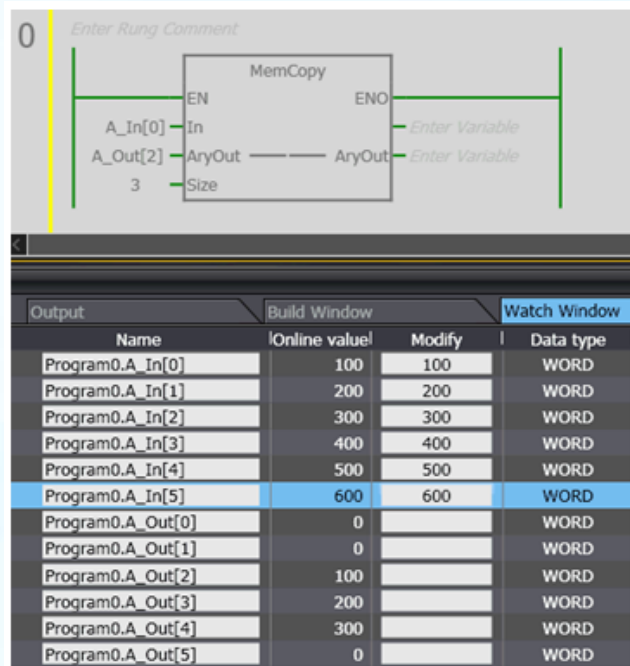


第五章 逻辑编程

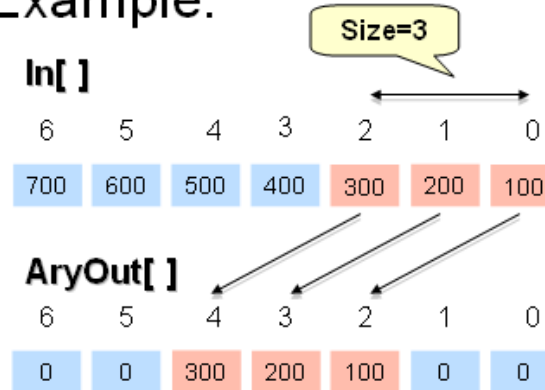
- 数组类指令
- 数组类和一般的指令区别在于其将数组作为一个整体，将一个个变量作为其中的个体，与以往欧姆龙**PLC**处理指令相比较差异较大

第五章 逻辑编程

- MemCopy
- MemCopy指令复制两个或者两个以上元素在不同的数组之间。
- 可以将一个数组内的元素转移给该数组内其他位置的元素
- 转移源数据和转移对象必须是同样的数据类型

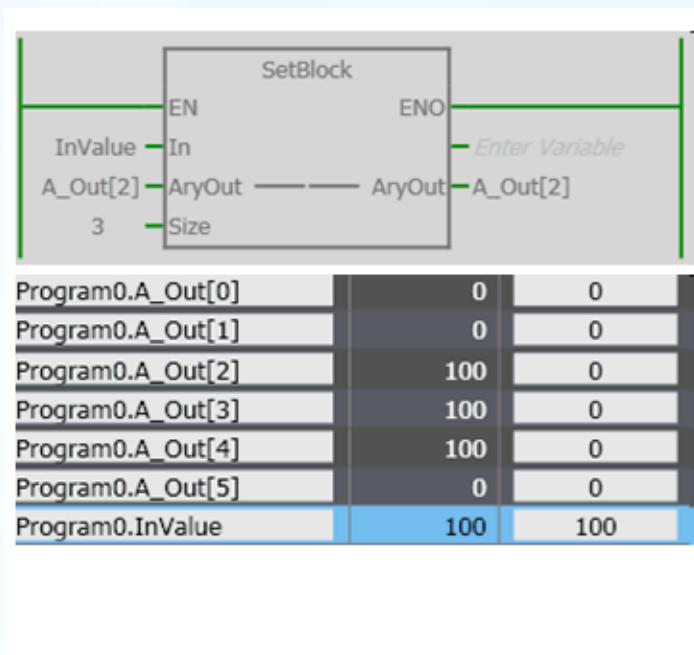


Example:

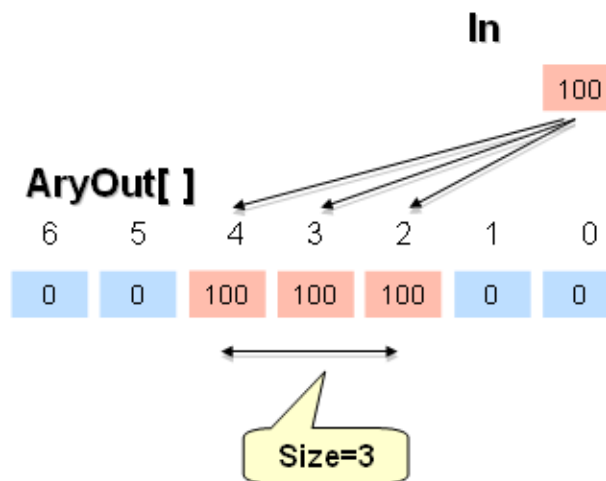


第五章 逻辑编程

- **SetBlock**
- **SetBlock**会根据用户设置的长度将输入值赋值给数组变量的内部元素。

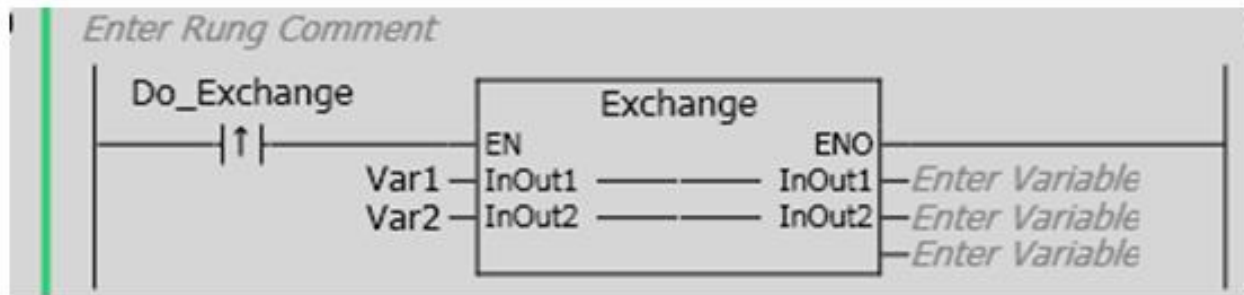


Example:



第五章 逻辑编程

- Exchange
- 将两个通道内的数据进行交换



Name	Online value	Modify	Data type
Program0.Var1	3	3	INT
Program0.Var2	0		INT
Program0.Do_Exchange	False	TRUE FALSE	BOOL

Name	Online value	Modify	Data type
Program0.Var1	0	3	INT
Program0.Var2	3		INT
Program0.Do_Exchange	True	TRUE FALSE	BOOL

第五章 逻辑编程

- AryExchange
- AryExchange将两个数组参数数值对换。

The image displays two side-by-side screenshots of the Omron GX Developer software's Watch Tab Page, illustrating the execution of the AryExchange function.

Left Screenshot (Initial State):

- Do_Exchange: False
- Program0.Ararray1[0]: 5
- Program0.Ararray1[1]: 5
- Program0.Ararray1[2]: 5
- Program0.Ararray1[3]: 5
- Program0.Ararray2[0]: 13
- Program0.Ararray2[1]: 13
- Program0.Ararray2[2]: 13
- Program0.Ararray2[3]: 13

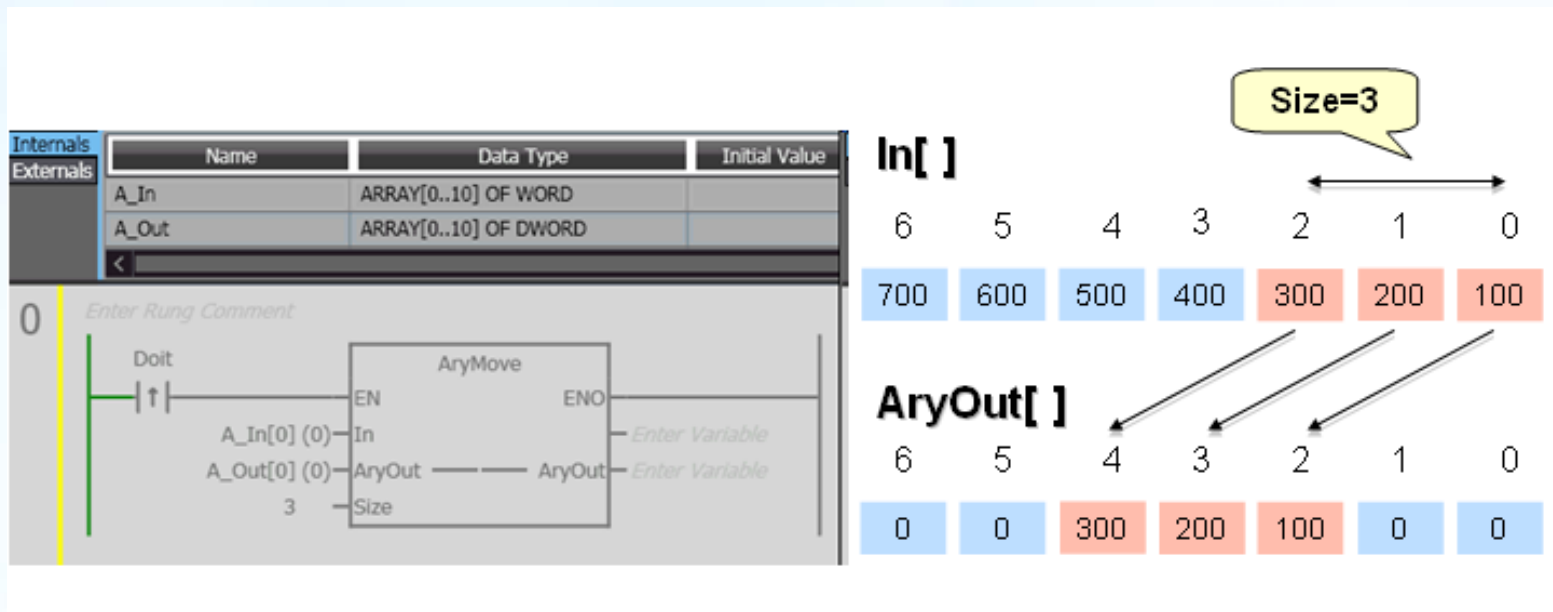
Right Screenshot (After Execution):

- Do_Exchange: True
- Program0.Ararray1[0]: 13
- Program0.Ararray1[1]: 13
- Program0.Ararray1[2]: 13
- Program0.Ararray1[3]: 5
- Program0.Ararray2[0]: 5
- Program0.Ararray2[1]: 5
- Program0.Ararray2[2]: 5
- Program0.Ararray2[3]: 13

Red arrows indicate the swap of values between the two arrays: the value 5 from Ararray1[3] moves to Ararray2[3], and the value 13 from Ararray2[0] moves to Ararray1[0].

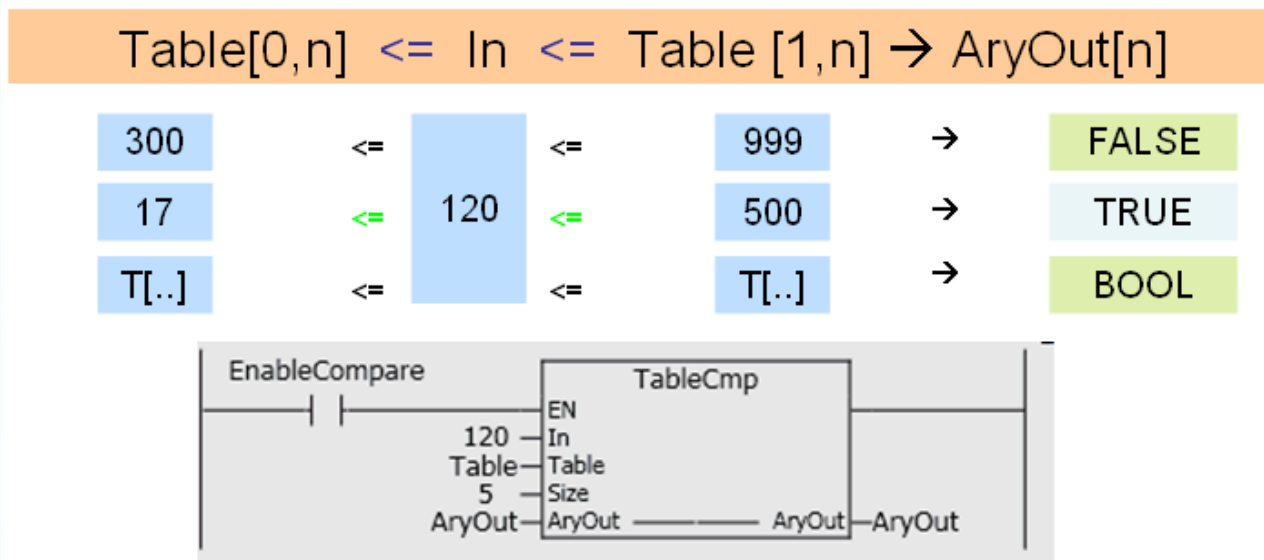
第五章 逻辑编程

- AryMove
- 指令复制两个或者两个以上元素在不同的数组之间
- 转移源数据和转移对象可以是不同的数据类型



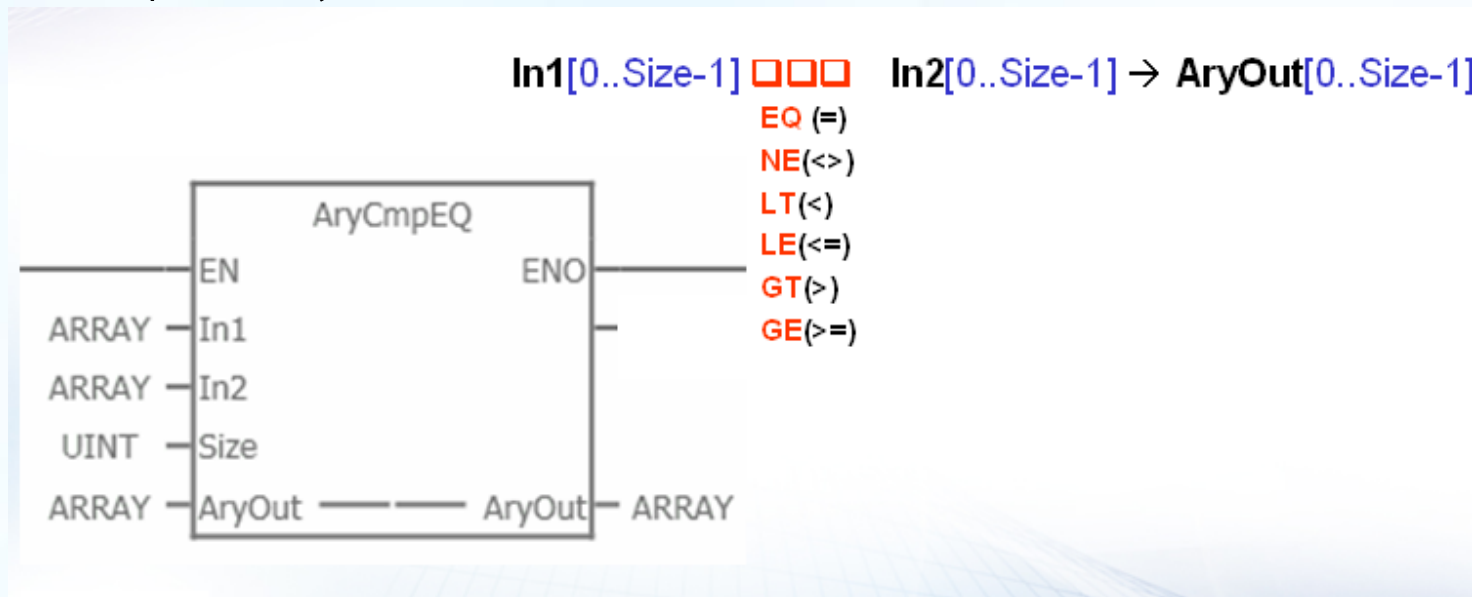
第五章 逻辑编程

- TableCmp
- TableCmp 指令是一个能对表格数据做区域比较的指令。
- 这种表格数据是由一个二维数组变量来扮演的。
- 然后将输出设置给一个一维数组。



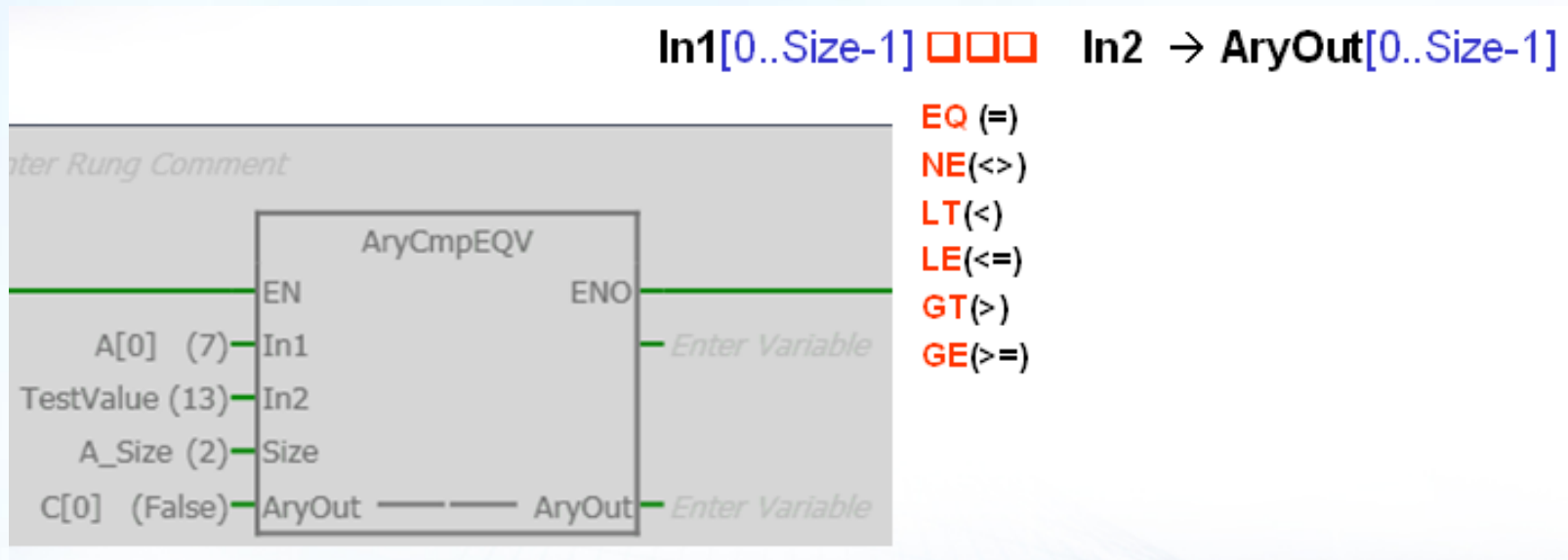
第五章 逻辑编程

- **AryCmp****
- **AryCmp****指令是比较根据用户设定制定元素个数的2个不同数组之间进行比较，并将结果给予AryOut变量上设置的数组。******里面可以是EQ,NE等。



第五章 逻辑编程

- **AryCmp**V**
- **AryCmp**V**指令是用于一个数组变量和一个值之间做比较，并把比较的输出结果给一个数组变量。******里面可以是**EQ,NE**等



END

欧姆龙客户服务中心

WWW.fa.omron.com.cn

免费声讯：400-820-4535