欢迎参加 OMRON NJ专题班

第一章 软件实际操作的知识点

- 第一节 编程的组件化和再利用
- 第二节 最适合编程语言的选择
- 第三节 国际标准IEC61131-3的优点
- 第四节 编程基本单位和任务
- 第五节 FUN
- 第六节 FB
- 第七节 变量
- 第八节 数据类型

第一章 软件实际操作的知识点

- 最适合编程语言的选择

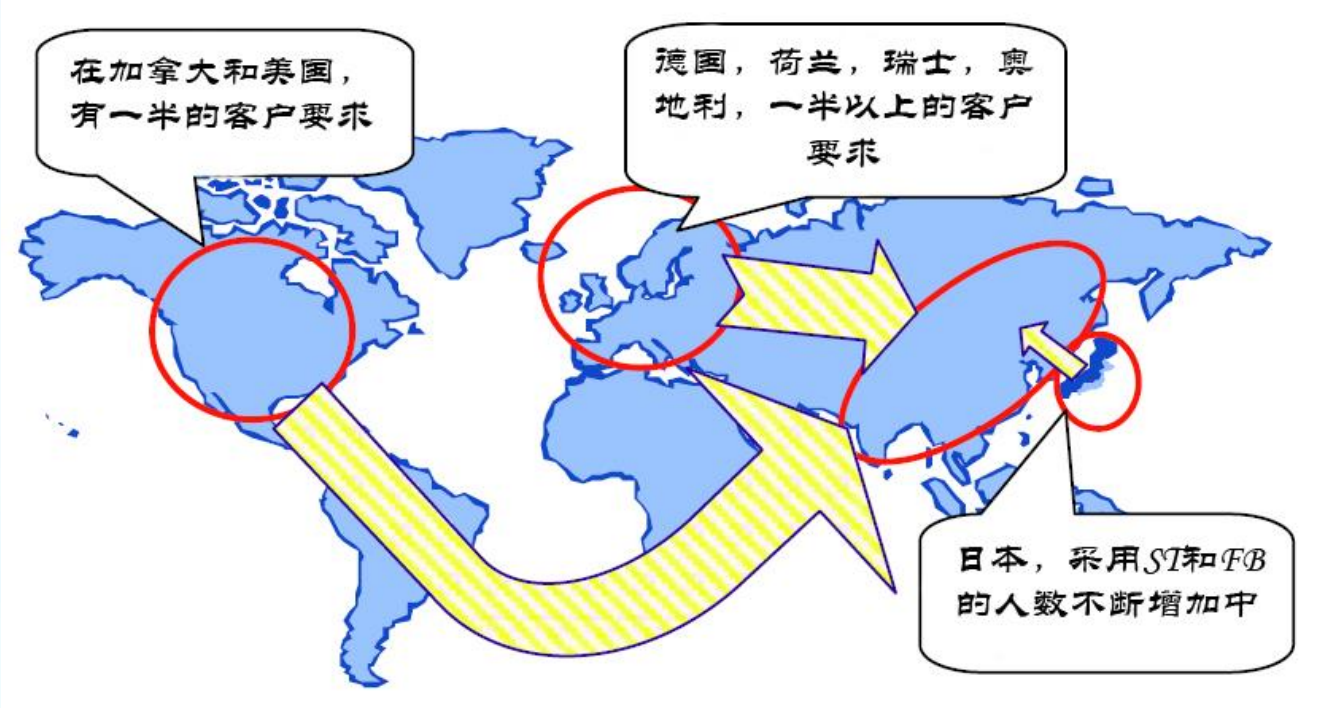
（1）高效率

（2）对应IEC61131-3规格

（3）数据变量化

第一章 软件实际操作的知识点

- IEC6113-3的优点
- 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

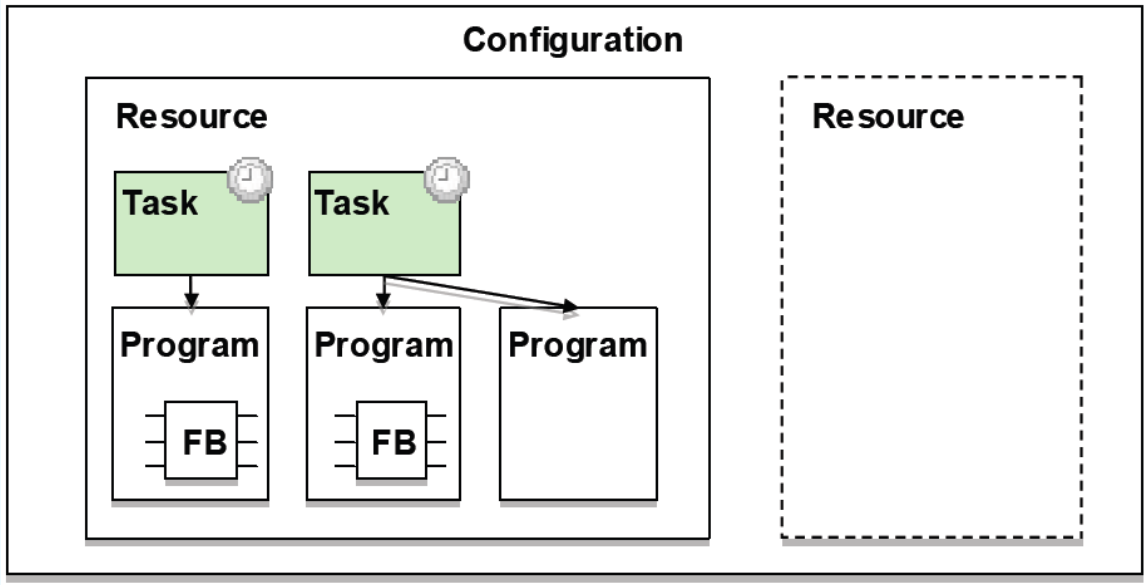


第一章 软件实际操作的知识点

- 可读性。
- 可重复利用性。
- 多种编程语言（IL/LD/ST/SFC/FBD）。
- 可移植性（对于不同厂家间的程序移植）。
- 可降低对应的教学经费。
- 可使学校教育标准化。
- 全球化的标准。
- 许多情况下最终用户会指定使用该标准。

第一章 软件实际操作的知识点

- 由IEC61131-3所决定，任务（Task）下挂入一个或者多个程序来实现用户所需要的逻辑命令。在程序中可以使用功能或者功能块
- NJ系统下标准的POU（Program Organization Unit）可以由程序（Program）,功能（Function）,功能块（Function Block）组成。它是最基本的单元用以建立用户程序。



第一章 软件实际操作的知识点

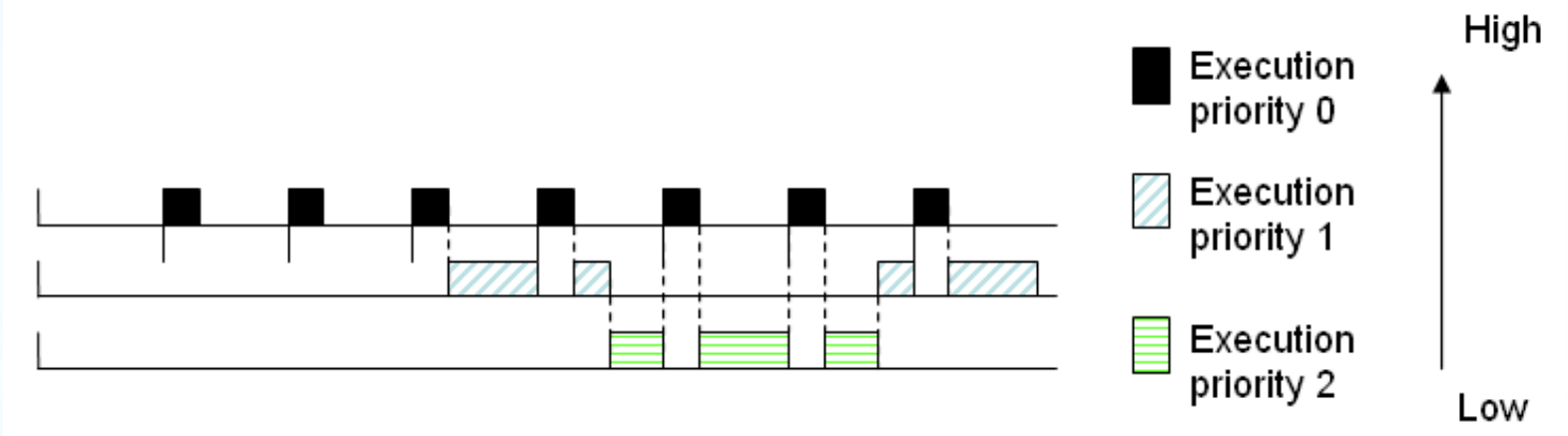
- 任务（Task）
- 在标准IEC61131-3下，任务是用一种来执行用户程序的功能。
- （1）任务的类别

  NJ系统内的任务包含两种类别：周期任务和事件任务

  其中事件任务功能尚未开放。

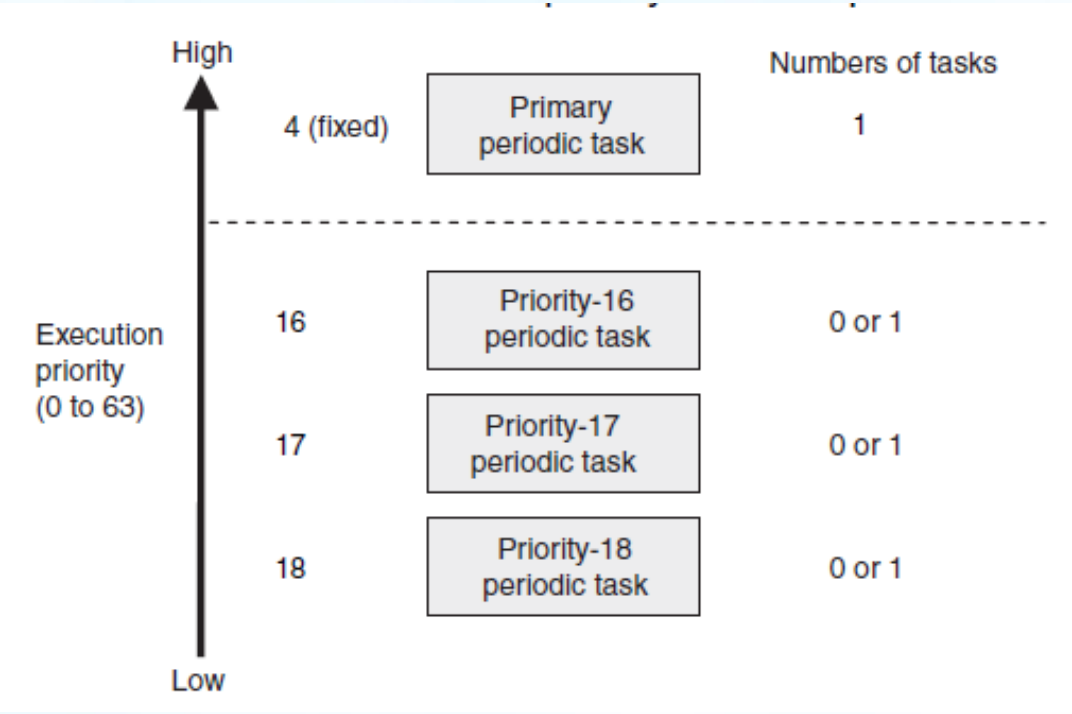
第一章 软件实际操作的知识点

- （2）优先度
- 每个任务的循环周期和优先度可以单独被设置。
- 根据所分配的优先度的高低，在高优先度的任务的循环周期结束后，开始动作低优先度的任务。
- 当低优先度任务尚未完成时，如果高优先度任务循环时间又到了后，则会打断低优先度的任务。



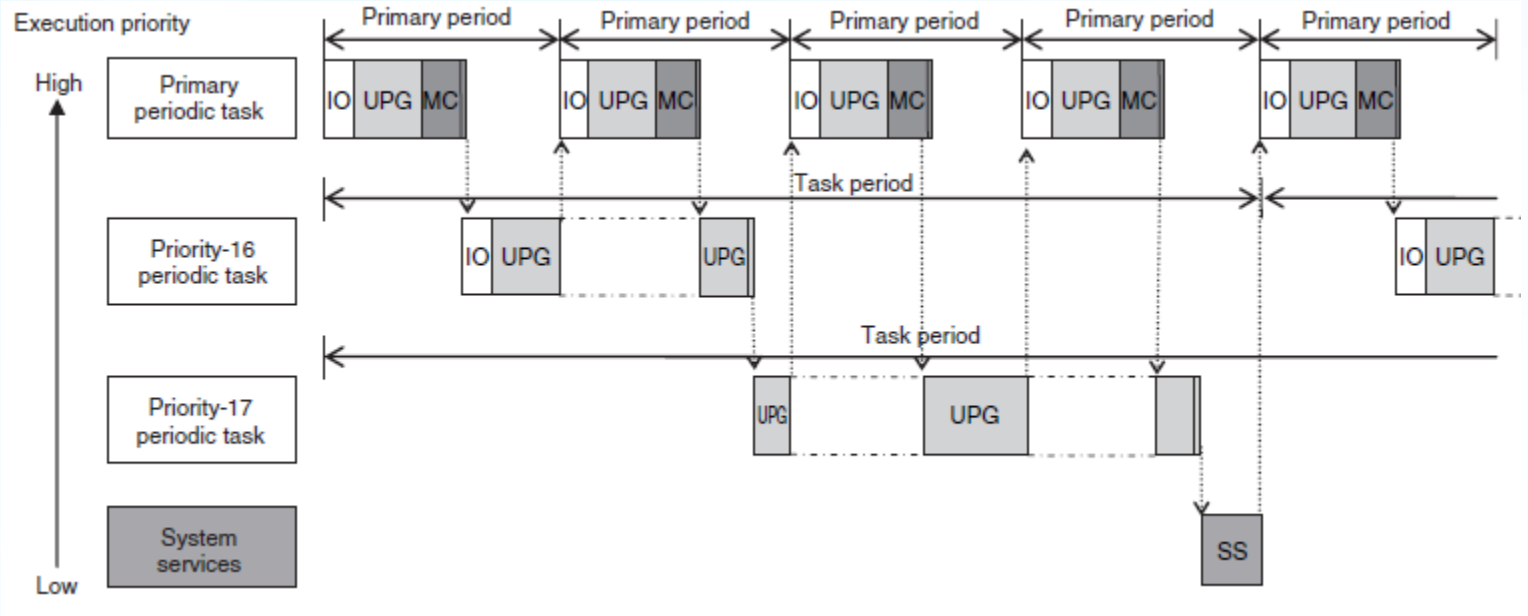
第一章 软件实际操作的知识点

- NJ系统的任务和原本CJ的任务概念有了比较大的区别。在原本的CJ系统中，高优先级的周期任务是不会打断低优先级的循环任务。



第一章 软件实际操作的知识点

- 另外一点很大的区别在于以前无论有几个周期任务，一个扫描周期里I/O刷新只有一次但是现在I/O刷新会有不同的任务内执行，而用户可以根据自己的实际程序来决定将不同的I/O分配到不同的任务下。（4和16号优先度）



第一章 软件实际操作的知识点

- 周期任务又分为主要周期任务和周期任务两种

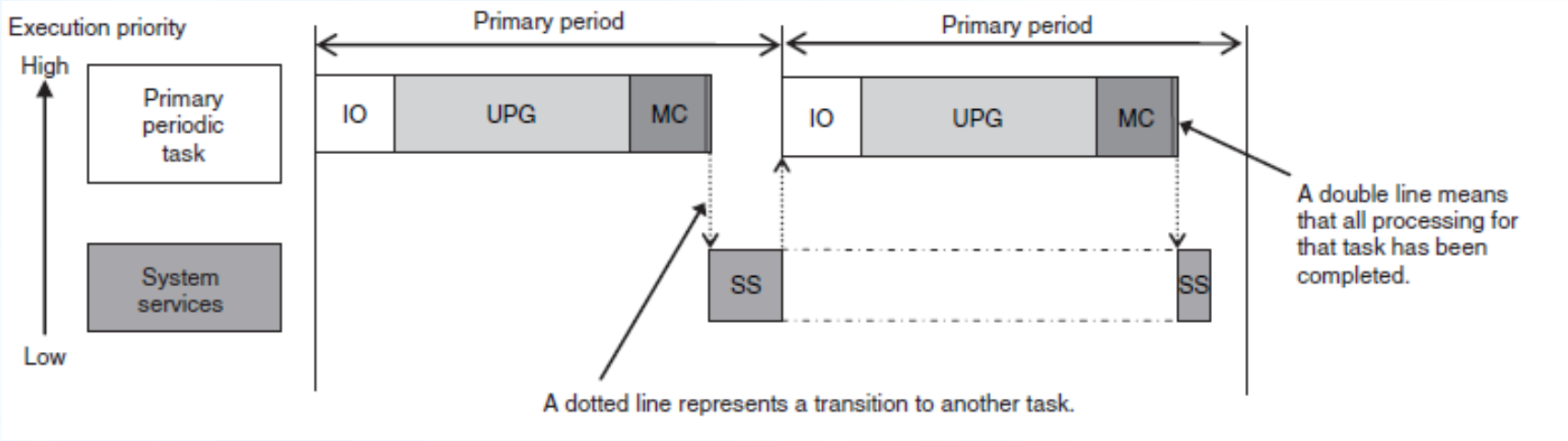
Item	Specification
Type of task	• Primary periodic task • Periodic task
Numbers of tasks	• Primary periodic task: 1 • Periodic tasks: 0 to 3 tasks
Number of programs per task	128 max.
Task period of the primary periodic task	500 µs, 1 ms, 2 ms, or 4 ms
Task periods of periodic tasks	Set the task period of each periodic task to an integer multiple of the task period of the primary periodic task. Any of the following can be set.

Task period of the primary periodic task	Task periods that you can set for periodic tasks
500 µs	1 ms, 2 ms, 3 ms, 4 ms, 5 ms, 8 ms, 10 ms, 15 ms, 20 ms, 25 ms, 30 ms, 40 ms, 50 ms, 60 ms, 75 ms, or 100 ms
1 ms	1 ms, 2 ms, 3 ms, 4 ms, 5 ms, 8 ms, 10 ms, 15 ms, 20 ms, 25 ms, 30 ms, 40 ms, 50 ms, 60 ms, 75 ms, or 100 ms
2 ms	2 ms, 4 ms, 8 ms, 10 ms, 20 ms, 30 ms, 40 ms, 50 ms, 60 ms, or 100 ms
4 ms	4 ms, 8 ms, 20 ms, 40 ms, 60 ms, or 100 ms

第一章 软件实际操作的知识点

- 主要周期任务的特点

1. 拥有最高级的优先度。
2. NJ系统通过主要周期任务来不间断工作。
3. 每个周期会处理I/O刷新，系统处理，用户程序，执行运动控制。



第一章 软件实际操作的知识点

- 周期任务的特点

1. 周期任务的执行周期都是主要周期任务的执行周期时间的整倍。
2. 周期任务和主要周期任务使用不同的时期来实行控制。
3. 可以将一个整体内容分成0～3个周期任务里。
4. 根据客户要求对不同周期任务的优先度进行分配。（优先度16，17，18）

第一章 软件实际操作的知识点

- 系统服务
- 系统服务简称SS（System services）
- 对于一个CPU单元运行的过程当中，那些不在任务中进行的过程动作和处理就被归类为系统服务

System service	Description
USB port service	• Processing of service requests from the Sysmac Studio or an HMI, such as CIP commands
Built-in EtherNet/IP port service	• Processing of message service requests, such as CIP commands, from the Sysmac Studio, an HMI, host computers, or other Controllers • EtherNet/IP tag data link communications processing **Note** If there is communications processing for EtherNet/IP tag data links and refreshing tasks are set for the tags (i.e., variables with a Network Publish attribute), variable access processing is performed as part of system common processing 2 for the task that is set as the refreshing task and not as a system service.
Service for CJ-series Special Units	• Event servicing for CJ-series Special Units • Execution of communications instructions (CIP) **Note** The CPU Unit exchanges data between CJ-series Special Units and their allocated memory words during I/O refreshing.
SD Memory Card service	• Access from FTP client • SD Memory Card operations from the Sysmac Studio • Execution of SD Memory Card instructions
Self-diagnosis	• Hardware error detection

第一章 软件实际操作的知识点

- I/O刷新
- 需要处理的对象数据主要分成CJ系列的单元和EtherCAT Slaves的从站数据（包括网络型伺服产品）
- 刷新的时间段取决于用户的任务分配分为 分配给主要周期任务和周期任务

第一章 软件实际操作的知识点

- 任务间的接口
- 当一个系统内有多个任务并存的时候就要考虑任务间的接口问题。
- NJ系列有两个方法来标准多任务情况接口的正常运行

1 对变量进行独家控制功能的设置

2 通过指令对相关段落的变量进行保护

第一章 软件实际操作的知识点

- FUN

（1）FUN是不需要设置名字的，因此输入很简单。

（2）FUN是不占用内存。

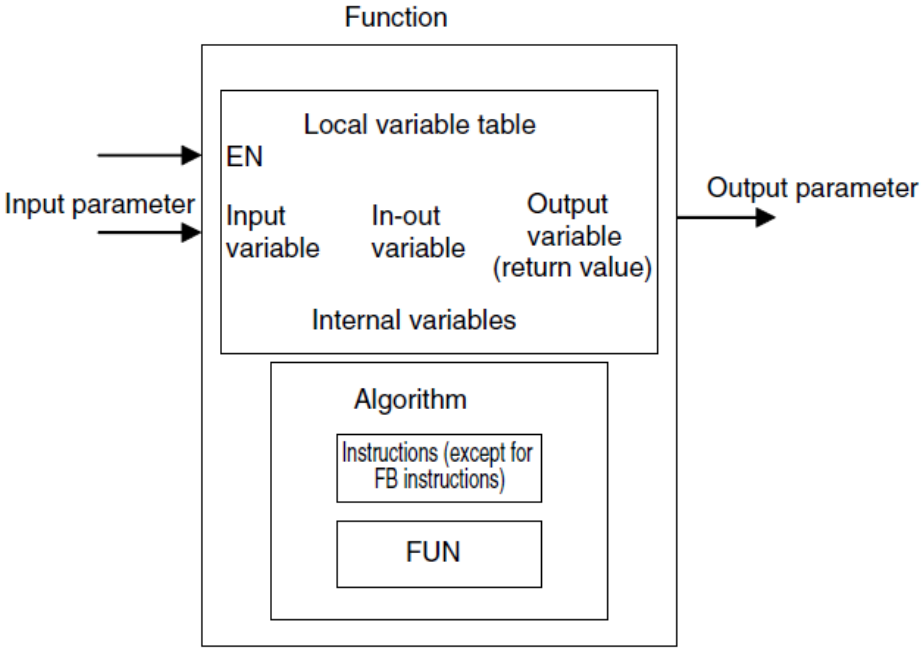
(3) 没有数量限制。

(4) 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

(5) 一般只要输入相同的值就会输出一样的结果。

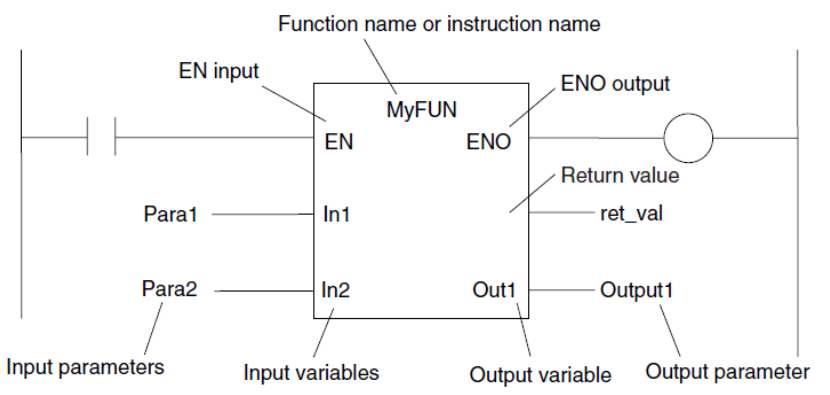
第一章 软件实际操作的知识点

- FUN的结构
- FUN内部包括一张完整的变量表（一个EN变量必须被设置）和必要的算法（不包括FB）



第一章 软件实际操作的知识点

- 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
```

第一章 软件实际操作的知识点

- F B的特点

（1）FB需要设置名字，而且不同的FB要设置不同的名字。

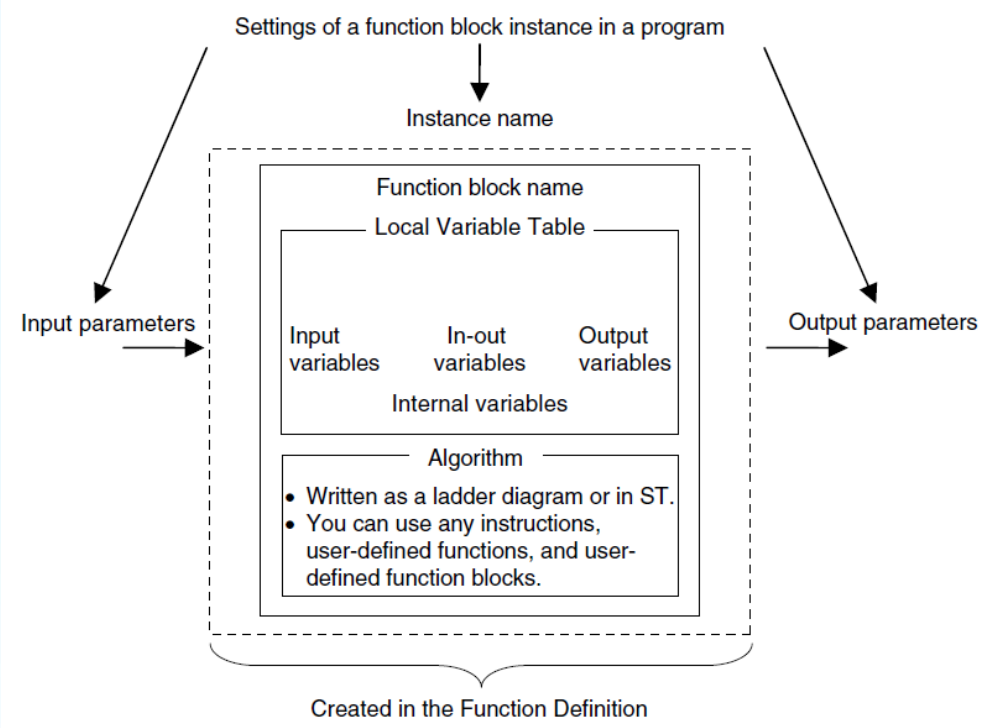
（2）占用内存区，根据个数占用数量则不等。

（3）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。

（4）能够实现即使是相同的输入也能产生不同的输出结果。

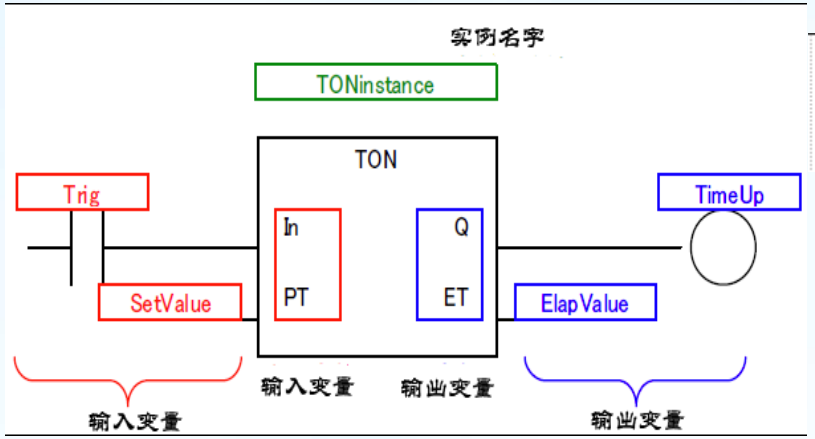
第一章 软件实际操作的知识点

- FB概述
- FB内部包括一张完整的变量表和必要的算法
- FB在使用的时候需要设定一个名字



第一章 软件实际操作的知识点

- FB在梯级编程环境



- FB在ST编程环境

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

第一章 软件实际操作的知识点

- 变量的概念
- 变量需要分配一个固定的名字，以及类型和属性
- NJ系列的变量类型现在比以往的PLC变量类型更齐全
- 变量分为本地变量和全局变量
- 全局变量当要被一个具体的POU调用的时候，必须登记进POU内部的外部变量表中

第一章 软件实际操作的知识点

- 全局变量（Global variables）

  全局变量的使用范围：

  全局变量可以被所有的POU(programs,FB,FUN)访问，但是当POU被外部访问的时候为了保护安全性防止被误修改需要改该部分变量设定为外部变量才能访问。

备注：一般全局变量都会挂钩设备变量。

NJ系列具有将设备变量自动生成为全局变量的功能。

第一章 软件实际操作的知识点

- 本地变量（local variables）
- 本地变量根据不同的对象，会有不同的区别，有内部变量，外部变量，输入变量，输出变量，输入输出变量之分
- 一个POU的本地变量对于其他POU而言是不可访问的
- 一个POU内部变量名字不可重复，但是不同POU 内部变量名字可重复
- 注意：本地变量不可以作为网络中使用的变量

第一章 软件实际操作的知识点

- 变量

（1）变量名称。

（2）数据类型。

（3）AT指定。

（4）保持。

（5）初始值。（注意初始值和保持两项的区别）

（6） 常数。

（7）网络公开。

（8）微分。

第一章 软件实际操作的知识点

- 数据类型
- NJ能处理的数据类型有三种
- 基本数据类型
- POU实例名型变量
- 导出数据类型

第一章 软件实际操作的知识点

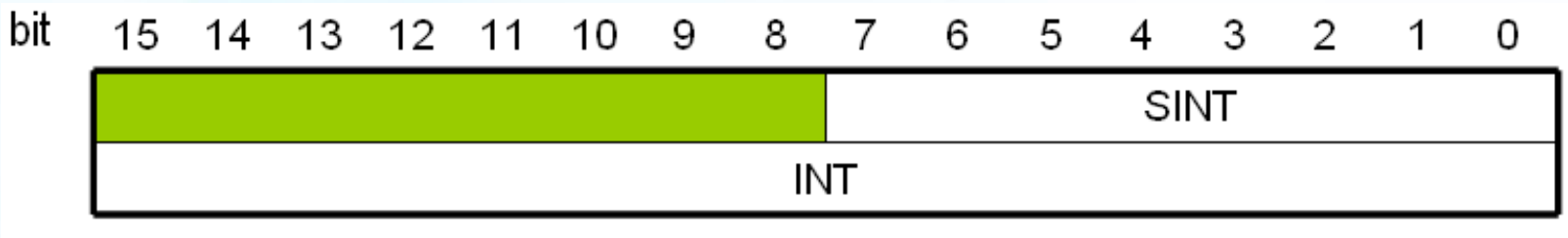
- 基本数据类型
- 布尔型
- 位字符串
- 整数型（带符号/不带符号）
- 实数型
- 持续时间型
- 日期型
- 时刻型
- 日期时刻型
- 文本字符串型

第一章 软件实际操作的知识点

- 部分新增数据类型的说明
- BYTE型
- NJ系列中新加入了BYTE数据类型，和以前的WORD数据类型两者都能在位层面用来表示数值。
- 表述范围00～FF/0000 0000~1111 1111

第一章 软件实际操作的知识点

- SINT/USINT型
- 只占一个字节的长度
- SINT的范围是-128~+127，USINT的范围是0~+255



第一章 软件实际操作的知识点

- TIME型数据
- 这种数据类型是用来表达时间范围的，总共占用8个字节
- 时间范围在#-9223372036854.775808ms
- ~#+9223372036854.775807ms
- 有多种写法
- 例如T#6d_10M
- T#12d3.5h
- T#61m(等同于#1h1m)

第一章 软件实际操作的知识点

- DATE型
- 对时间做描述的，占用8个字节
- 格式为yyyy-mm-dd，范围为1970-1-1到2106-2-6
- 例如D#2011-11-29

第一章 软件实际操作的知识点

- TIME_OF_DAY型
- 占用8个字节
- 格式为hh-mm-ss，范围为00:00:000000000~23：59：59.999999999
- 例如；Tod#12:14:59 代表12点14分59秒

第一章 软件实际操作的知识点

- DATE_AND_TIME
- 可以看为DATE与TIME_OF_DAY数据类型的合体
- 格式为yyyy-mm-dd-hh:mm:ss，范围为
- 1970-01-01:00:00:000000000~2106-02-06:23:59:999999999

第一章 软件实际操作的知识点

- 练习
- 制作一段可以计算设备运行时间的指令
- 使用到的指令有：
- GetTime获取当前时间
- SUB_DT_DT比较DATE_AND_TIME差值用指令
- TimeToSec 转换为秒的指令

第一章 软件实际操作的知识点

- 导出数据类型主要分为

1. 结构体（Structure）
2. 共同体 (Union)
3. 枚举体（Enumeration）
4. 数组

第一章 软件实际操作的知识点

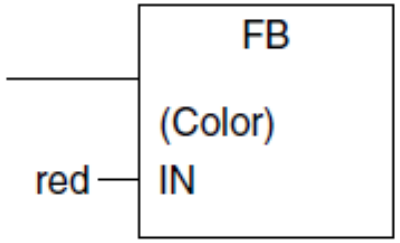
- 结构体
- 结构体是内部包含了两种或者两种以上不同数据类型的一种变量类型（和数组的最大区别）
- 通过这种结合，这些不同数据类型的变量可以被当作一个整体来使用

第一章 软件实际操作的知识点

- 联合体
- 联合体可以实现同一个数据以不同的变量类型来被访问

第一章 软件实际操作的知识点

- 枚举体
- 枚举体实现了将文本和数值联系起来的解决方案
- 使用文本就代表调用了数值



Enumeration Table

Data type	
Color	ENUM
Enumerator	Value
red	0
yellow	1
green	2

Variable Table

Variable name	Data type
DiscColor	Color

第一章 软件实际操作的知识点

- 数组
- 一个数组可以由同一类型数据变量组成。并将它们看成一个整体
- 当要处理多个同样数据类型的变量时候，数组会显得十分方便。比如在作一些位置控制的数据设定时
- 派生体也可以作为数组的成员

第一章 软件实际操作的知识点

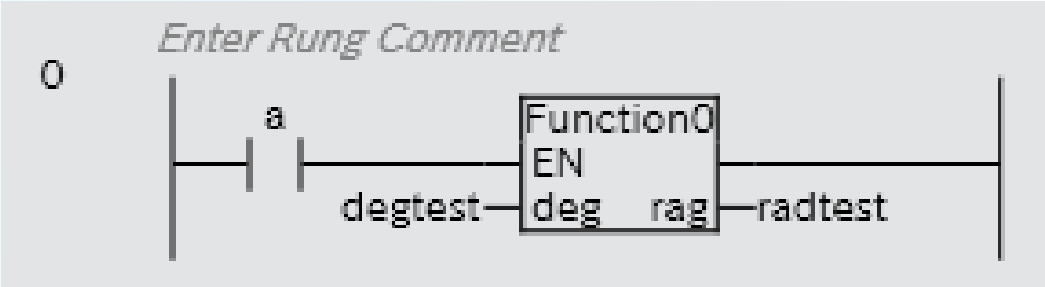
- 范围的概念
- NJ系统允许用户进行范围设定（当然要小于这种变量类型本身的范围）
- 写法上（A..B） A和B就分别为范围的两个阀值

第二章 功能和功能块的制作

- 第一节 功能的制作
- 第二节 功能块的制作
- 第三节 编程

第二章 功能和功能块的制作

- 功能的制作（使用梯级编程）
- 演示&练习
- 需要制作有如下功能的一个功能，当输入角度的度数后，输出会自动转换成π角度
- 完成效果

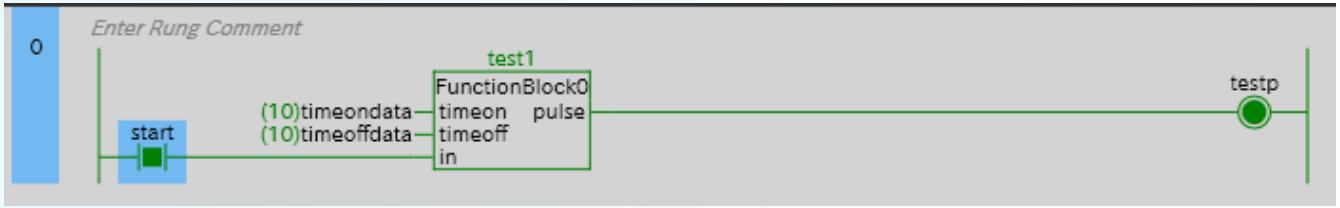


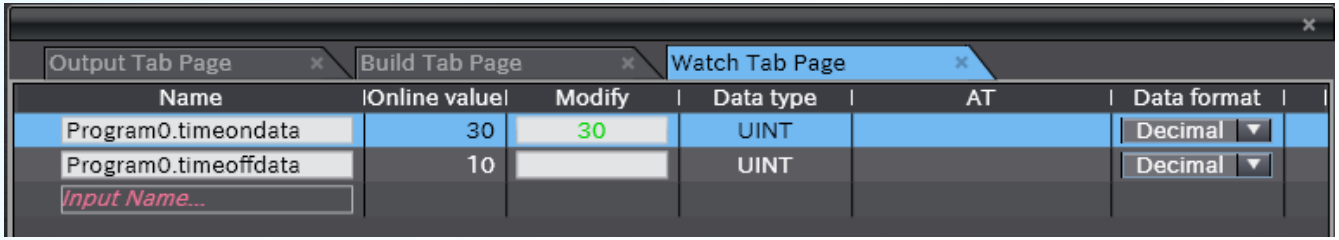
Build Tab Page | Output Tab Page | Watch Tab Page

Name	Online value	Modify	Data type	AT	Data format
Program0.degtest	180	180	LREAL		Real
Program0.radtest	3.141592		LREAL		Real
Input Name...					

第二章 功能和功能块的制作

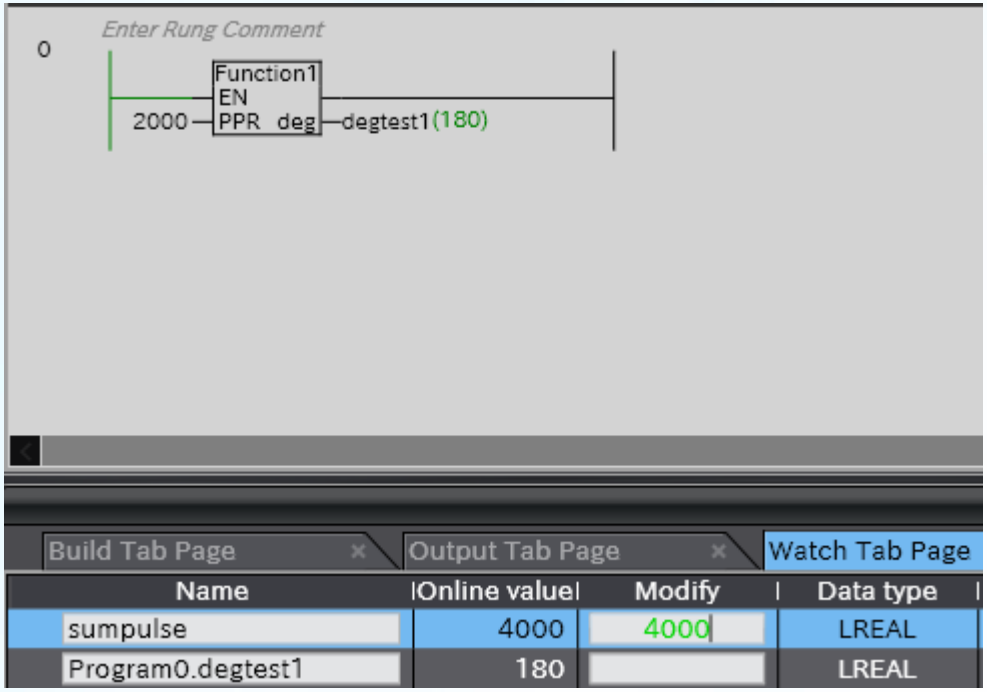
- 功能块的制作
- 需要制作有如下功能的一个功能块，按照输入的ON和OFF时间来输出不同占空比的脉冲点，如果ON和OFF时间相同就是标准脉冲信号
- 完成效果





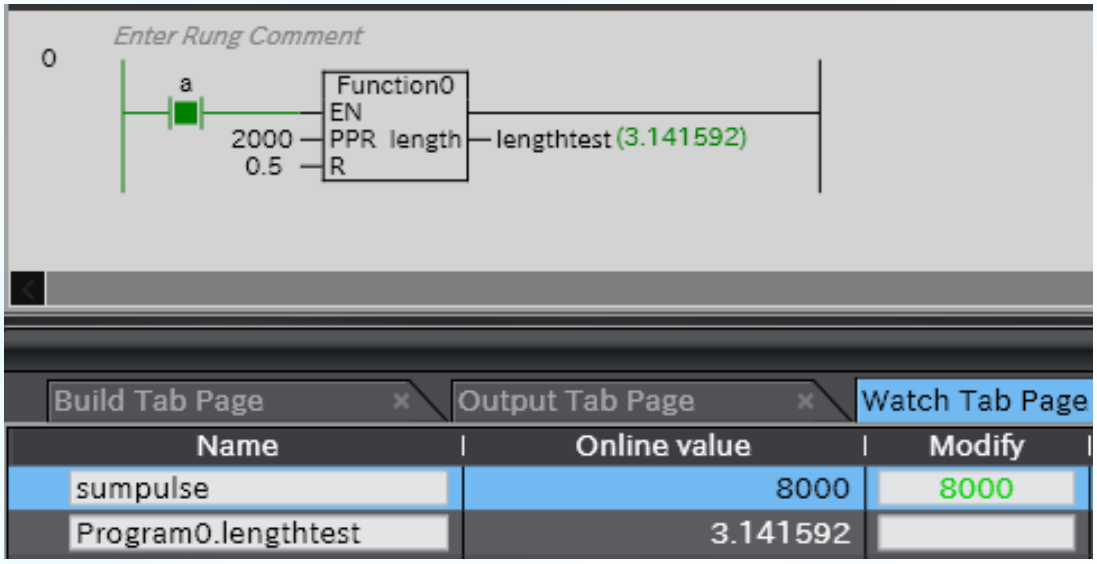
第二章 功能和功能块的制作

- 练习一
- 要求实现根据已知旋转编码器每转脉冲数，根据当前脉冲个数显示当前走过的角度
- 完成效果



第二章 功能和功能块的制作

- 练习二
- 要求实现：当已知道旋转编程器每转脉冲数和外挂轮分度圆半径。求走过的弧长距离
- 完成效果



第三章 使用ST编程

第一节 ST编程的概念

第二节 ST编程演示（四则运算）

第三节 ST编程命令 （IF,CASE,FOR)

第四节 ST控制文本程序的练习

第五节 ST编程下调用功能和功能块（以及和梯级编程的区别)

第三章 使用ST编程

- ST概述
- ST(Strctured Text)中译为结构文本。是一种基于文本的高级语言；它采用了一些描述语言，来描述系统中各种变量之间的关系，执行需要的操作。

第三章 使用ST编程

```
//Program Overview
IF NOT  fault THEN
      Ready:=TRUE; //Green Light in ON state
      AlarmSignal:=FALSE;
      (*Turn OFF the red light
            AND enable machine FOR operation*)

      IF NOT HomingDone THEN
            ExecuteHoming:=TRUE;//Make homing FOR 1st TIME;
            OpenGrip:=TRUE; //OpenGrip grip FOR Enable grasping
      END_IF;

ELSE
      Ready:=FALSE;
      ExecuteHoming:=FALSE;
      AlarmSignal:=TRUE;
      (*turn red light ON*)
END_IF;
```

蓝色部分是流程控制程序语言和运算符。

绿色部分是用户注释

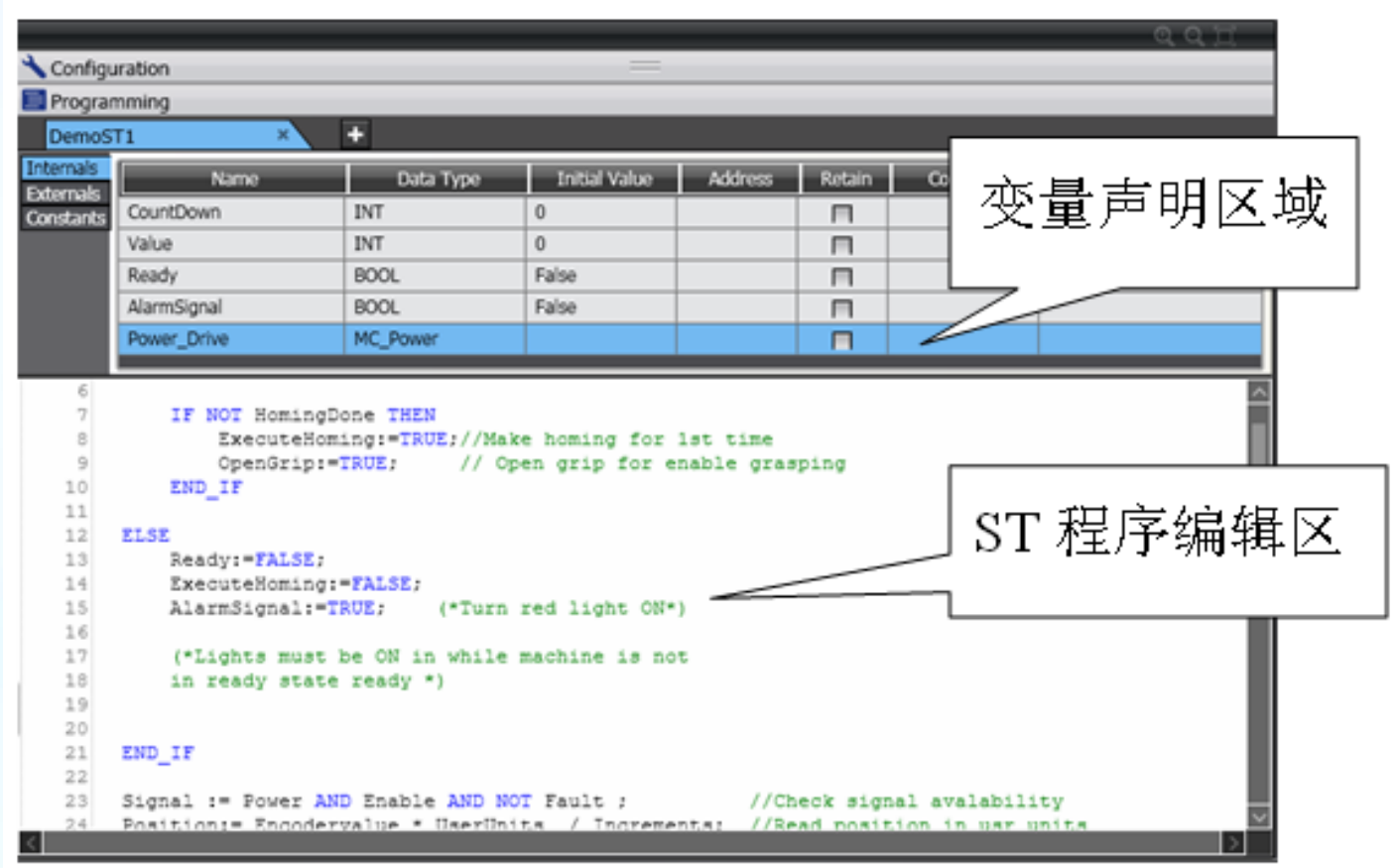
黑色部分是用户变量，用户功能，用户功能块

第三章 使用ST编程

- ST的优点：
- （1）对使用者而言，代码有更强的可读性
- （2）如果是都基于IEC61131-3的ST代码，那么彼此就会有很强的可移植性
- （3）可以被文本编辑器编辑
- （4）比LD（梯形图）和FBD（功能块图）编程更有效率
- （5）能够灵活运用结构体数据类型
- （6）当改用ST后，开发时间将被降低
- （7）在一个POU内部可以将梯级中嵌入ST编程（Inline ST）

第三章 使用ST编程

- ST编程环境



第三章 使用ST编程

- 变量赋值
- “：=”
- 使用“；”结尾
- 使用“（*...*）进行标识
- 或者“//”进行标识
- 变量名的大小写没有区别

```
Ready:=FALSE;
AlarmSignal:=TRUE;
OpenGrip:=FALSE;
```

```
AlarmSignal:=TRUE;     (*Turn red light ON*)

(*Lights must be ON in while machine is not
in ready state ready *)
```

```
IF NOT HomingDone THEN
    ExecuteHoming:=TRUE;//Make homing for 1st time
    OpenGrip:=TRUE;      // Open grip for enable grasping
END_IF
```

第三章 使用ST编程

- 允许使用空格和标签使得ST更有可读性

```
Signal:=      Power AND Enable AND NOT fault;
Position:=    EncoderPosition  *  UserUnits  /  Increments;
```

- 对变量的分配定制二进制和十六进制，BIN码

```
ValueINT:=125;              //value expressed in integer
ValueHEX:=16#7D;            //value expressed in hexadecimal
ValueBIN:=2#1111101;        //value expressed in binary
```

第三章 使用ST编程

- 对于一个较长或较复杂的表达式，允许分成多行来表达，通过使用换行符“↵”来实现。另外注意在结尾部分不要漏掉分号

```
SafetyChain:=   (Sensor_1 AND Sensor_2 AND NOT Sensor3)
                AND (PowerEnable AND NOT StandStill OR MotorDisabled OR Compressor)
                OR(MainsSupply AND PowerSupply OR SecondarySupply)
                AND NOT (SafetyRelays OR SecurityLatch OR SafetyStop) ;
```

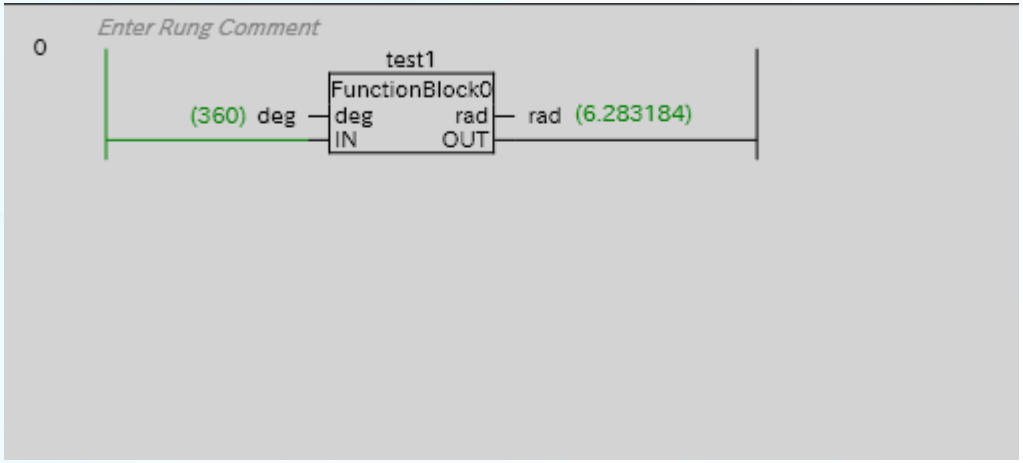
第三章 使用ST编程

- 变量名称
- 关键字都被禁止使用，包括AND,BY,CASE,DO,ELSE,ELSIF,EXIT,FALSE,FOR,IF,NOT,OF,OR,REPEAT,RETURN,THEN,TO,TRUE,UNTIL,WHILE,XOR,END_IF,END_WHILE,END_CASE,END_REPEAT
- 特殊符号不可以使用，包括

  <=,>=,<>,:=,..,&,(*,*),%,$,@...
- 一些数据类型也不可以使用 包括
- USINT,SINT,BYTE,UINT,INT,WORD,REAL,DINT,UDINT,DWORD,LREAL,LINT,ULINT,LWORD…

第三章 使用ST编程

- 编程练习（使用ST编程环境）
- 演示&练习
- 要求当输入角度的度数后，输出会自动转换成π角度
- 完成效果



第三章 使用ST编程

- ST编程演示（四则运算）
- ST常用的运算符

( )	Operation Execution Order	Value:=(1+2) *(3+4) // Value is 21 Precedence order : ( ), * / , + -
**	Exponent	Value:= 2**8 ; // Value is 256
NOT	Negation of a logical condition	Value:=NOT TRUE; //Value is FALSE
*	Multiplication	Value:=8 * 100; // Value is 800
/	Division	Value:=200 / 25; // Value is 8
+	Addition	Value:=200 + 25; // Value is 225
-	Subtraction	Value:=200 - 25; // Value is 175
MOD	Remainder	Value:=10 MOD 6; // Value is 4
<,>,<=,>=	Comparison	Value:= 60 > 10; // Value is TRUE
=	Equal condition	Value:= 8=7; // Value is FALSE
<>	Not equal condition	Value:= 8<>7; // Value is TRUE
&, AND	Logical AND	Value:=2#1001 AND 2#1100; //Value is 2#1000
XOR	Logical Exclusive OR	Value:=2#1001 XOR 2#1100; //Value is 2#0101
OR	Logical OR	Value:=2#1001 XOR 2#1100; //Value is 2#1101

第三章 使用ST编程

- ST标准命令主要包含以下三类

（1）条件语句

- IF..THEN…END_IF
- IF..THEN…ELSE…END_IF
- IF..THEN…ELSIF..THEN..END_IF

（2）分歧语句

- CASE..OF…END_CASE

（3）条件循环语句

- FOR..(BY)..DO..END_FOR
- WHILE..DO…END_WHILE
- REPEAT..UNTIL…END_REPEAT
- EXIT

第三章 使用ST编程

- IF指令
- IF指令的基本结构
- <condition_expression>部分是负责评价决定是否执行<statement_1>的依据。当condition为真的时候，则执行。
- 如果条件不为真，则会跳到END_IF后以去继续其他的动作。

```
IF <condition_expression> THEN
    <statement_1>;
```

第三章 使用ST编程

- 举例：
- 当Enable变量为真且PowerON变量为非的时候，State变量等于10。其他情况下State变量等于0

```
State:=0;
IF Enable AND NOT PowerON THEN
    State:=10;
END_IF

Value:=State;(*if previous condition would be TRUE, Value will be 10.
              If not,  Value will be 0 *)
```

第三章 使用ST编程

- IF指令二
- 当<condition_expression>部分为真时，执行THEN和ELSE之间的<statement_1>部分
- 当<condition_expression>部分为非的时候，执行ELSE和END_IF之间的<statement_2>部分

```
IF <condition_expression> THEN
    <statement_1>;
ELSE
    <statement_2>;
END_IF;
```

第三章 使用ST编程

- IF 指令三
- 当<condition_expression2>条件部分为真时，会动作<statement_2>声明部分。
- 完成该步后，会继续检测下一个ELSIF的条件部分，所以使用ELSIF就可以允许对多个条件进行检测。
- 如果这些条件都不满足，则执行ELSE下面的句子。

```
IF <condition_expression_1> THEN <statement_1>;
 ELSIF <condition_expression_2> THEN <statement_2>;
 ELSIF <condition_expression_3> THEN <statement_3>;

      ...
  ELSIF <condition_expression_n> THEN <statement_n>;
ELSE <statement_m>;
END_IF;
```

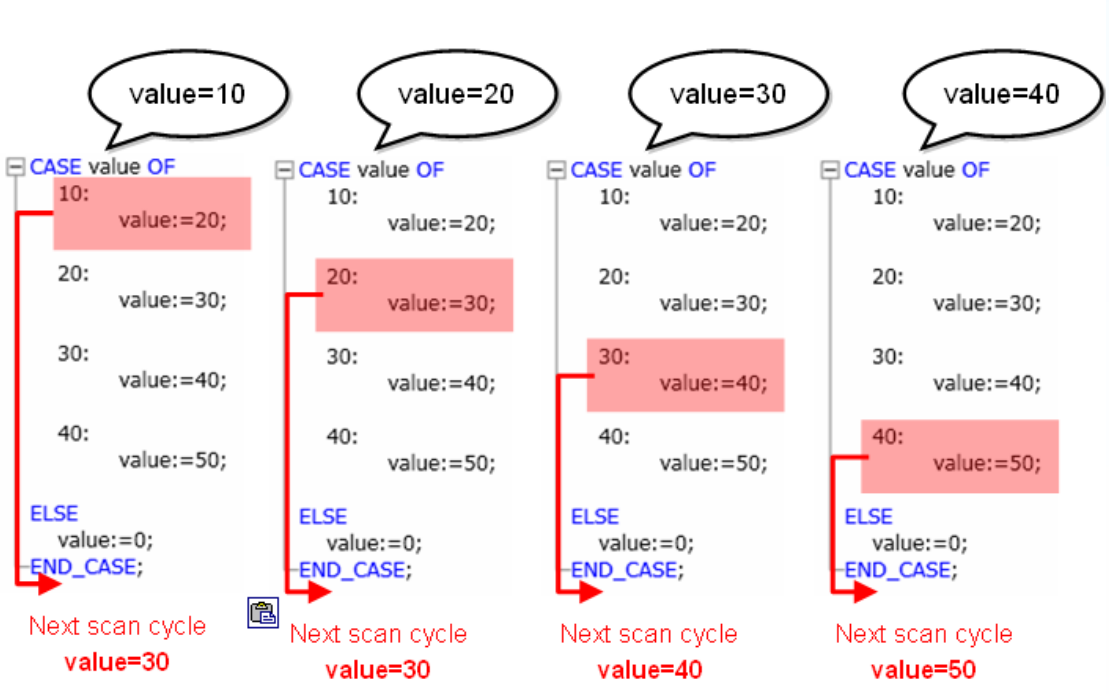
第三章 使用ST编程

- CASE指令
- CASE指令的基本结构
- <integer_expression>是对其内部的值进行评估，根据这个结果然后执行其对应的<integer_expression_value_n>里面的部分进执行。
- 如果这些条件都不满足，则执行ELSE下面的命令。

```
CASE <integer_expression> OF
    <integer_expression_value_1>:<statement_1>;
    <integer_expression_value_2>:<statement_2>;
            ...
    <integer_expression_value_n>:<statement_n>;
ELSE<statement_m>;
END_CASE;
```

第三章 使用ST编程

- 时序问题
- Value初始值为10，所以第一个循环后value为20。
- 从第二个循环开始后value为20，所以满足“20：”，因此第二个循环后value等于20。
- 从第三个循环开始后value为30，所以满足“30：”,因此第三个循环后value等于40。
- 从第四个循环开始后value为30，所以满足“30：”,因此第四个循环后value等于50。



第三章 使用ST编程

- 支持“x..y”的写法，代表x到y之间的所有数。
- 支持“x,y”的写法，代表当条件等于x或y的时候，则执行之后的动作。

```
CASE A OF

   1:
      X:=1;

   2,5:    //if X value is 2 or 5

      X:=2;

   6..10:  //if X value is within 6 to 10 range

      X:=3;

   11,12,15..20:   //if X value is 11 0r 12, or within 15 and 20

      X:=4;
ELSE

   X:=0;

END_CASE;
```

第三章 使用ST编程

- FOR指令
- FOR指令的基本格式
- 在由<initial_value>和<end_value_expression>决定的范围内，循环执行FOR..END_FOR之间的内容。
- 当计数次数满足后，则跳到END_FOR之后。
- BY指令是可选的，BY可以通过<increment_expression>内的内容来决定如何计数。

```
FOR <FOR_variable> := <initial_value>  TO  <end_value_expression>  BY <increment_expression>
      DO
   <statement>;
END_FOR;
```

第三章 使用ST编程

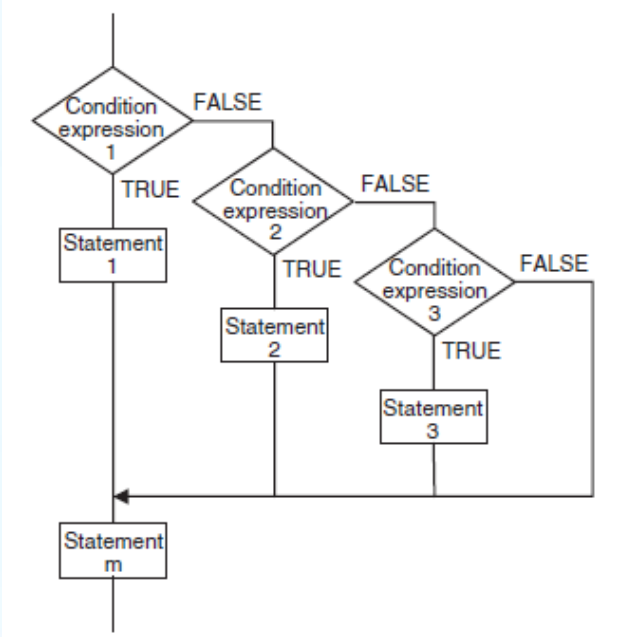
- BY的使用
- 默认不使用BY的情况下，每次循环+1
- 也可根据客户具体要求进行修改

```
FOR CountDown:=3 TO 56 DO
    Value:= Value + CountDown;
END_FOR;
```

```
FOR CountDown:=100 TO 0 BY -1 DO
    Value:= Value + 5;
    Toggle:= NOT Toggle;
END_FOR;
```

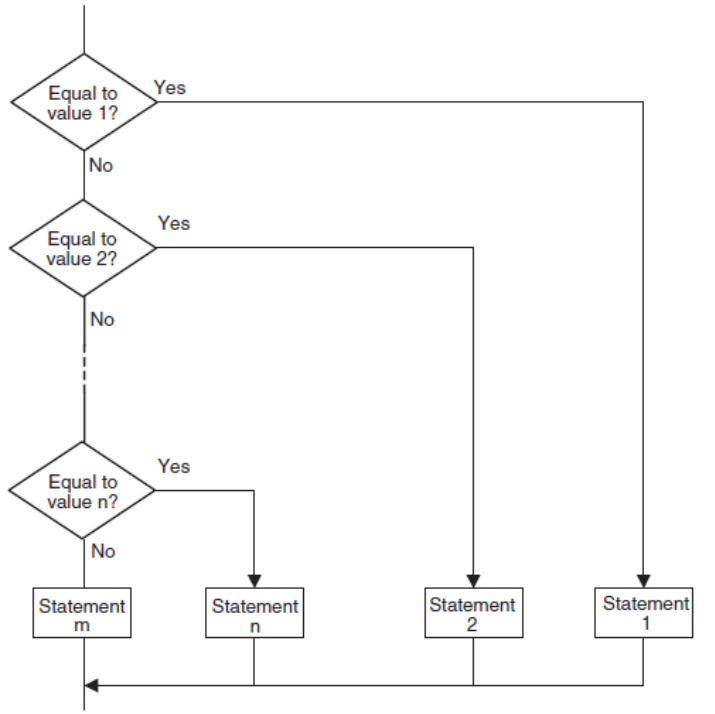
第三章 使用ST编程

- 编程练习（ ST环境下）
- 要求：根据图示求写出对应的ST指令
- 使用IF指令



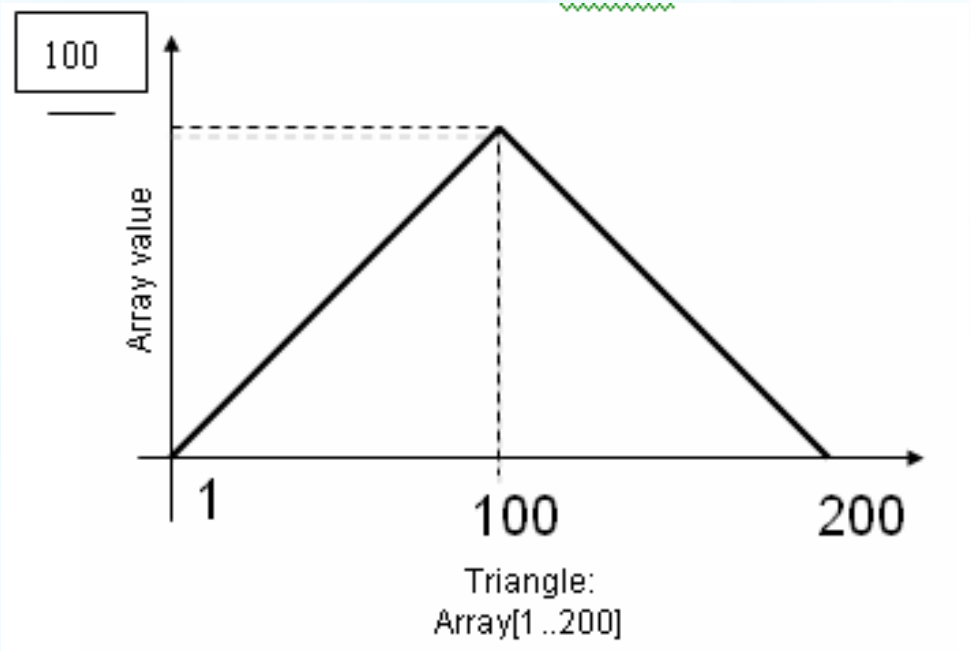
第三章 使用ST编程

- 编程练习（ ST环境下）
- 要求：根据图示求写出对应的ST指令
- 使用CASE指令



第三章 使用ST编程

- 编程练习（ ST环境下）
- 要求：根据图示求写出对应的ST指令
- 使用FOR指令

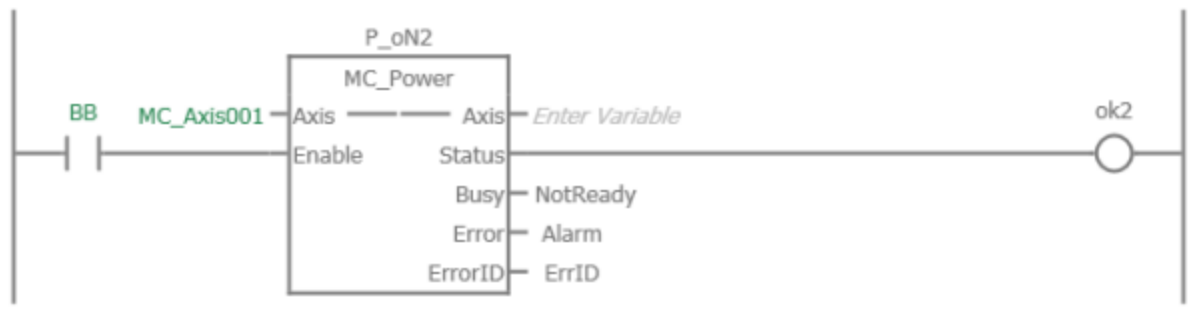


第三章 使用ST编程

- ST 编程下调用功能和功能块
- 调用FB，需要使用其在变量表中登记的名字
- 输入赋值使用“：=”
- 输出赋值使用“=>”
- 输入输出赋值两种都可以用（一般用“：=”）

第三章 使用ST编程

- 举例
- 原本在梯级下MC_Power功能块定义改为ST的写法
- 未使用的输入或者输出可以不写




```
P_oN2(
   Axis := MC_Axis001,
   Enable := BB,
   Axis =>,
   Status =>ok2,
   Busy =>NotReady,
   Error =>Alarm,
   ErrorID =>ErrID);
```

- 也可以改为变量名：= 功能块名.参数的写法

```
p_oN (
    Axis := MC_Axis000,
    Enable := AA,
    Axis =>,
    Status => ok,
    Busy => NotReady,
    Error =>,
    ErrorID =>);
```

```
ok:= p_oN.Status ;
```

```
NotReady:= p_oN.Busy ;
```

第三章 使用ST编程

- 调用FUN
- ST内调用功能是直接使用它自己的名字，因为它和功能块不一样是不需要设置名字
- 不需要设置的参数也可以省略不写

```
Production_1:=ProductionSpeed(
             Enable:=TRUE,
             SetPoint:=45,
             Diameter:=345.6);

Production_2:=ProductionSpeed(
             Enable:=TRUE,
             SetPoint:=46,
             Diameter:=220);
```

第三章 使用ST编程

- 调用特殊数据类型
- 结构体这样的变量类型可以用“.”来表示其中的元素

```
<variable>:= <Type> . <Type element>;
```

- 一个结构体包含另一个结构体，那么还是可以用“.”来做

```
<variable>:= <Type_1> . <Type_2> . <Type_2 element>;
```

第三章 使用ST编程

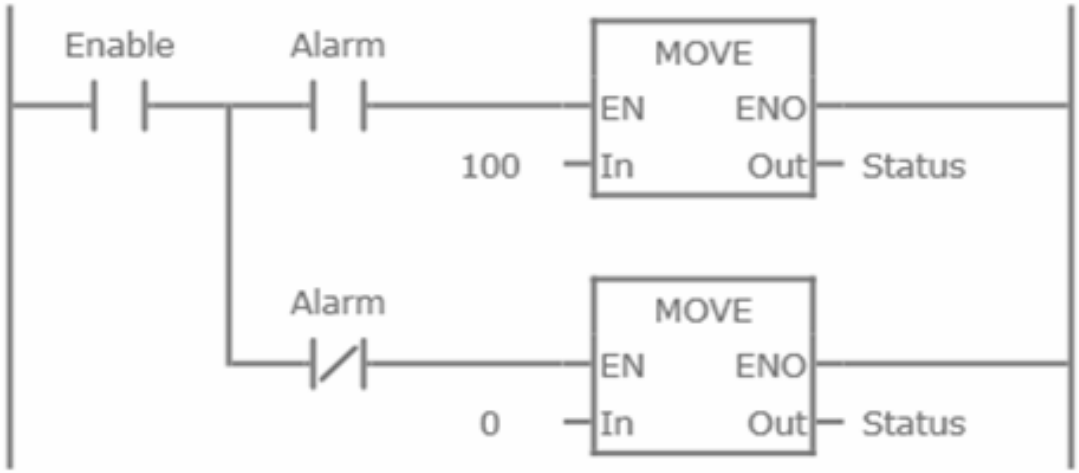
- LD梯级编程和ST结构文本编程的比较
- （1）基本逻辑导通线圈


```
Starter:=TRUE;
```

- （2）多重逻辑导通线圈

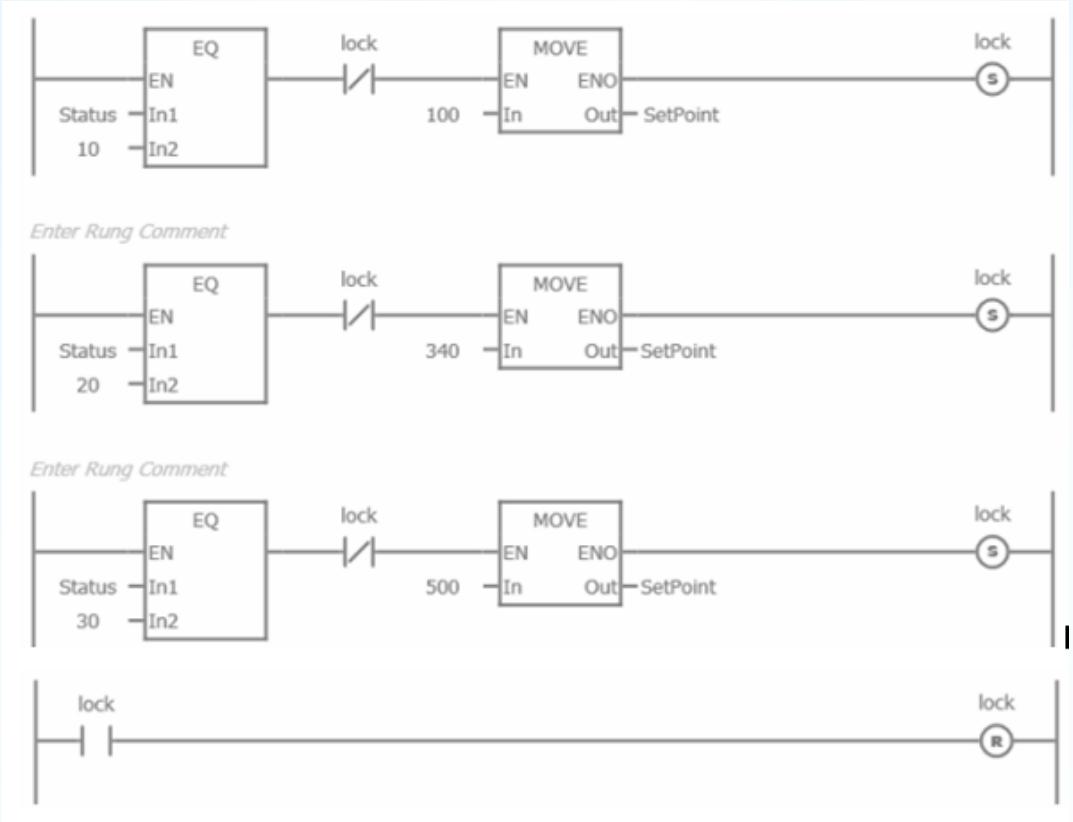

```
MotorRun:=(Enable OR ConveyorON)
          AND MotorStopped
          AND NOT Alarm;
```

- （3）简单逻辑触发功能




```
IF Enable THEN
    IF Alarm THEN
        Status:=100;
    ELSE
        Status:=0;
    END_IF;
END_IF;
```

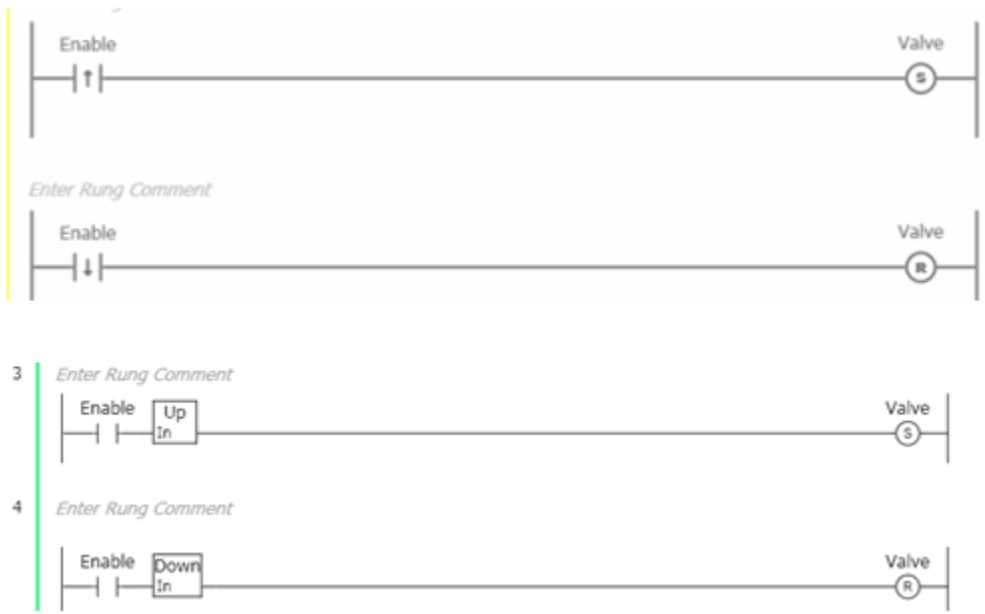
- （4）复杂逻辑触发功能




```
CASE Status OF

    10:  Setpoint:=100;

    20:  Setpoint:=340;

    30:  Setpoint:=500;

END_CASE;
```

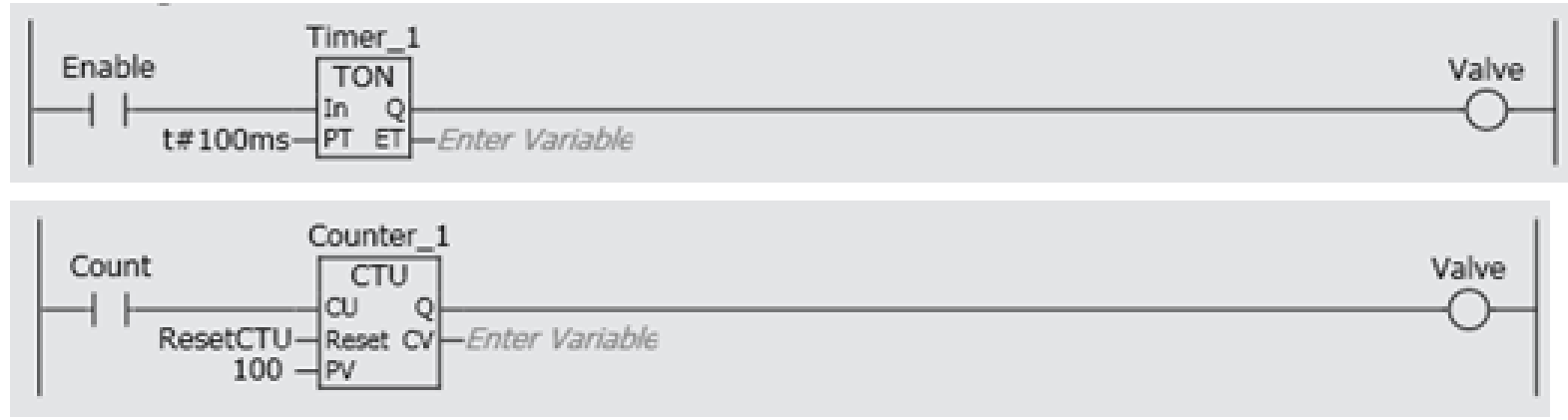
第三章 使用ST编程

- （5）微分触发




```
Rising_edge(clk:=Enable);
Falling_edge(clk:=Enable);

IF Rising_edge.q THEN
    Valve:=TRUE;
END_IF;

IF Falling_edge.q THEN
    Valve:=FALSE;
END_IF;
```

- （6）定时器和计数器类




```
Timer_1(In:=Enable,PT:=t#100ms,Q=>Valve);

Counter_1(   CU:=Count,Reset:=ResetCTU,
             PV:=100,Q=>Valve);
```

END

欧姆龙客户服务中心

WWW.fa.omron.com.cn

免费声讯：400-820-4535