结构文本(ST)
TM246

贝加莱工业自动化
Perfection in Automation
www.br - automation.com

简介

前提

培训模块：	TM210 – Automation Studio基础 TM211 – Automation Studio在线通讯 TM213 – 自动化运行（Runtime）系统 TM223 – Automation Studio诊断
软件：	无
硬件：	无

目录

1、简介

结构文本是一种高级语言，如果你知道如何使用高级语言来编程，像：Basic、PACSAL或C，那么你会很轻松的掌握Structured Text（ST）编程；如果不知道，你会看到ST有着简单、标准的结构，保证程序高效、快速运行并简单易懂。

图. 1 书本印刷: 过去和现在

在下一章里，你会学习到ST的命令、关键字、语法和其它的主题。所有的这些你都可以做练习，我们有许多帮助你理解ST的简单例程。

简介

1.1 目的

课程参与者将熟悉使用结构文本给自动化目标编程。
课程参与者将学到每个独立的命令组和它们之间如何工作。
课程参与者将了解结构文本预留的关键字。

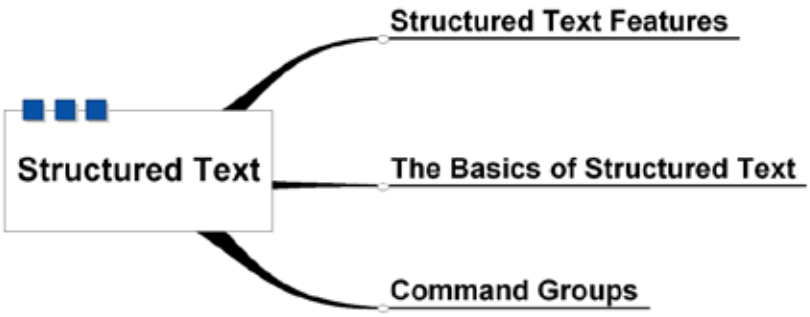


图. 2 综述

2、结构化文本的特点

2.1 概述

ST是针对自动化系统的高级文本编程语言。简单的标准结构确保快速、高效的编程。ST使用了高级语言的许多传统特性，包括：变量、操作符和控制流程语句。ST还能与其它的PLC编程语言一起工作。

那么什么是结构文本呢？"结构"是指高水平的结构化编程能力，象一个"结构化的编程"；"文本"是指应用文本而不是梯形图和顺序函功能表的能力。

ST语言不能代替其它的语言，每种语言都有它自己的优点和缺点。ST主要的一个优点就是能简化复杂的数学方程。

2.2 特点

结构化文本有以下特点:

高级文本编程语言

结构化的编程

简单的标准结构

快速高效的编程

使用直观灵活

与PASCAL类似

有计算机编程经验的人可以很容易地使用它

符合IEC 61131-3标准

2.3 可能性

Automation Studio提供以下功能:

数字量和模拟量I/O

逻辑操作

逻辑比较表达式

算术运算

判断语句

机器的状态语句

循环语句

功能块

可选用的动态变量

诊断工具

3、结构文本基础

3.1 表达式

表达式是指返回变量评估值的结构。表达式由操作符和操作数组成。操作数可以是常量,变量,调用函数或其它表达式。

例子:

```
b + c

(a - b + c)**COS(b)

SIN(a) * COS(b)
```

3.2 赋值操作符

通过一个表达式和一个值来给变量赋值。赋值语句包括位于左边的变量，赋值操作符":="，及后边需要计算的表达式。所有的语句，包括赋值语句，必须要以分号";"结尾。

例子:

```
Var1 := Var2 * 2; (*  Var1 <-- (Var2 * 2) *)
```

图 4 Assignment

当这行程序执行后，变量"Var1"的值是变量"Var2"的两倍。

3.3 注释

虽然注释经常被删掉，但它们是源代码中非常重要的一部分。它们解释了一部分代码，使程序更易读懂。注释帮助你或其他人读你的程序，即使过去了很长时间。注释不被编译，因此不会影响程序的执行。注释应该用一对星号和小括号括起来"(*comment*)"。

例子:

```
(* This is one line comment *)
```

图. 6 单行注释

```
(* This
is more
lines
comment *)
```

3.4 操作符优先级

如果在一个表达式中使用几个操作符，就会出现优先级的问题（执行的顺序）。操作符按优先级的顺序来执行。

在任何一个表达式中，首先执行最高级别的操作符，接着执行低一级的操作符，等等，直到执行完所有的操作符。具有相同级别的操作符按照书写顺序从左至右依次执行。

操作符	符号 / 语法:	
括号	()	最高优先级
函数调用		
例子	Call argument(s)	
LN(A), MAX(X), 等.		
注释	**	
取反	NOT	
乘		
除		
取模 (取除法的余数)	*	
/		
MOD		
加		
减	+	
-		
比较	<, >, <=, >=	
等于		
不等于	=	
<>		
逻辑与	AND	
逻辑异或	XOR	
逻辑或	OR	最低优先级

执行顺序:

例 1:

```
Result := 6 + 7 * 5 - 3;     (*The multiplication first; higher precendence *)
Result := 6 +   35  - 3;     (*The addition; rule from left to right *)
Result := 41 - 3;            (*Substraction at the end *)
Result := 38;
```

图. 7 例 1: 执行顺序

首先做乘法，然后是加法，最后是减法。
使用小括号（最高优先级），可以得到你想要的执行顺序。看下面的例子。

例 2:

如下所示，将操作符放到小括号里可能影响执行的顺序。

```
Result := (6 + 7) * (5 - 3);(*operations inside the parentheses first *)
Result :=    13   *    2;   (*then the multiplication *)
Result :=    26;
```

图. 8 例 2: 执行顺序

表达式从左至右执行。先执行小括号里的操作，接着是乘法。因为小括号的优先级高于乘法的优先级。可以看出，这两个例子看起来很相似，但结果不同。

4、命令组

ST有下面的命令组:
布尔逻辑操作
算术操作
比较操作
判断
Case语句

4.1 布尔逻辑操作

操作数不需要是BOOL类型。
布尔逻辑操作:

符号	逻辑操作	例子
NOT	取反	a := NOT b;
AND	逻辑与	a := b AND c;
OR	逻辑或	a := b OR c;
XOR	异或	a := b XOR c;

真值表:

输入		AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

这些操作符可以形成一个逻辑表达式和条件语句，结果是真（TRUE）或假（FALSE）。

命令组

例 1:

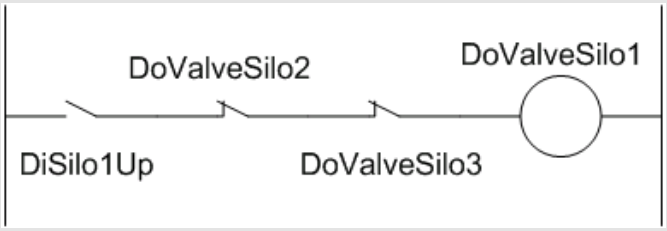


图. 10 电气图

ST编辑器允许任何数的分枝。

例 2:

```
IF (Level >= MaxLevel) OR (E_Stop = 1) THEN
  Pump := 0;
END_IF
```

图. 10 电气图

练习:

当按下"BtnLigntOn"开关后，输出"DoLight"应该亮起，直到"BtnLightOff"按下后才关闭。
使用布尔逻辑操作编写该任务。

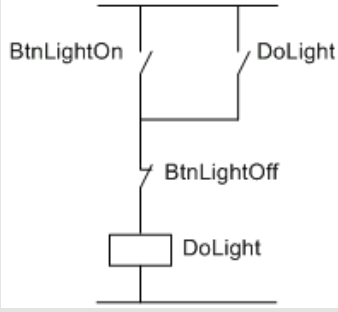


图. 11 例子, 逻辑操作

4.2 算术运算

使用高级语言的决定性因素是看它处理算术运算的简单程度。

4.2.1 基本的算术运算

ST为应用程序提供了以下基本的算术运算:

符号	算术操作	例子
:=	赋值	a := b;
+	加	a := b + c;
-	减	a := b - c;
*	乘	a := b * c;
/	除	a := b / c;
MOD	取模 (显示余数)	a := b mod c;

数据类型是非常重要的参数。看下面的表格。:

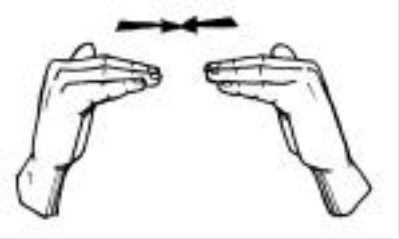
语法	数据类型	结果		
	Res	Op 1	Op 2	
Res := 8 / 3;	INT	INT	INT	2
Res := 8 / 3;	REAL	INT	INT	2.0
Res := 8.0 / 3;	REAL	REAL	INT	2.6667
Res := 8.0 / 3;	INT	REAL	INT	Error

* 编译器出错信息: Type mismatch: Cannot convert REAL to INT.

你可以看到，结果也依赖于语法和数据类型。
表达式左边的数据类型应该等同于（或大于）右边的数据类型。

备注:

左面数据类型 := 右面数据类型;



4.2.2 隐性数据类型转换

该类型的转换由编译器完成。编译器将表达式中低的数据类型转换成高的数据类型。如果有两种或多个类型的变量参与运算，那么必须将它们转换成相同的类型以便执行运算。

Data type	BOOL	SINT	INT	DINT	USINT	UINT	UDINT	REAL
BOOL	BOOL	x	x	x	x	x	x	x
SINT	x		INT	DINT	USINT	UINT	UDINT	REAL
INT	x	INT		DINT	INT	UINT	UDINT	REAL
DINT	x	DINT	DINT		DINT	UDINT	UDINT	REAL
USINT	x	USINT	INT	DINT		UINT	UDINT	REAL
UINT	x	UINT	UINT	DINT	UINT		UDINT	REAL
UDINT	x	UDINT	UDINT	UDINT	UDINT	UDINT		REAL
REAL	x	REAL	REAL	REAL	REAL	REAL	REAL	

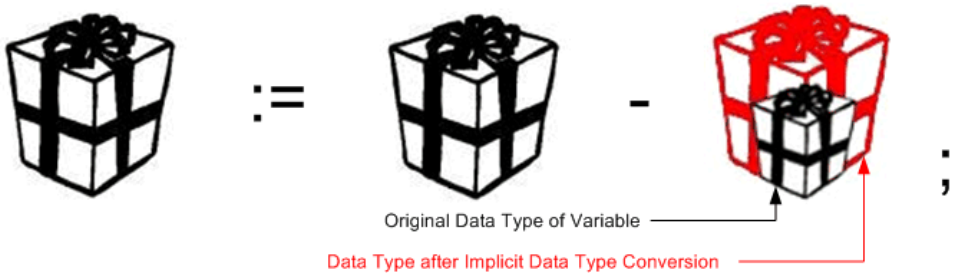


图. 12 隐性数据类型转换

例子:

```
INT_Result := INT_Var1 + SINT_Var2
(* [INT]        [INT]      [SINT] *)
```

图. 13 固有数据类型转换例子

SINT_Var2 首先转换成INT类型。

4.2.3 显性数据类型转换

显性数据类型转换也是数据类型转换问题。我们知道，表达式的左右两边要有相同的数据类型，但还需注意…

例子:

```
INT_TotalWeight := INT_Weight1 + INT_Weight2
(*   [INT]              [INT]          [INT]  *)
```

第一眼看上去好像没什么问题，但和（INT_Weight1+INT_Weight2）超过了INT的取值范围。在这种情况下，必须使用显性数据类型转换。

例子:

```
DINT_TotalWeight := INT_TO_DINT(INT_Weight1) + INT_Weight2
(*   [DINT]                        [INT]          [INT]  *)
```

变量DINT_TotalWeight应该为DINT类型，右边的变量中至少有一个应转换成DINT类型。这种转换用的是OPERATOR库中的函数。

练习:

在两个不同的地方检测玻璃钢的温度，编写程序来计算平均温度，并以模拟量显示输出。
注意模拟量输入和输出必须是INT 类型。

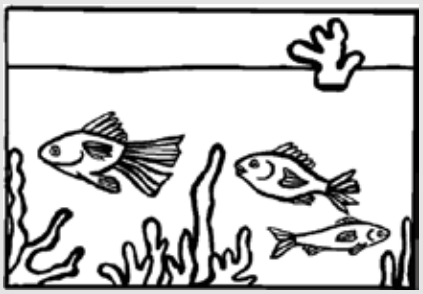


图. 14 玻璃钢

4.3 比较操作

高级编程语言ST或以允许比较操作的简单结构分枝。比较的结果是真（TRUE）或假（FALSE）。

符号	逻辑比较含义	例子
=	等于	IF a = b THEN
<>	不等于	IF a <> b THEN
>	大于	IF a > b THEN
>=	大于等于	IF a >= b THEN
<	小于	IF a < b THEN
<=	小于等于	IF a <= b THEN

比较操作作为一个逻辑条件用在IF, ELSE, WHILE 和UNTIL语句中。

4.4 判断

用IF语句表示判断，这里还要用到比较操作。判断分三部分:

简单IF语句

IF – ELSE语句

IF – ELSIF语句

嵌套的IF

判断	语法	描述
IF THEN	**IF a > b THEN**	**1. 比较**
	Result := 1;	**1. 语句(s)**
ELSIF THEN	**ELSIF a > c THEN**	**2. 比较 (可选)**
	Result := 2;	**2. 语句(s)**
ELSE	**ELSE**	**前面IF语句都不满足(可选)**
	Result : = 3;	**3. 语句(s)**
END_IF	**END_IF**	**判断结束**

4.4.1 IF

最简单的IF判断语句。

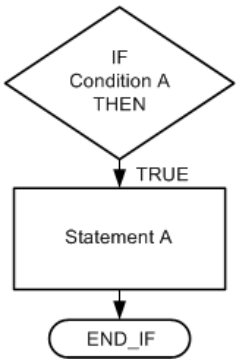


图. 15 简单 IF 语句流程图

```
(*********************************)
(******    Nested  IF     ********)
(*********************************)
IF V1 > V2 THEN
  IF V2 > V4 THEN
    V3 := 99;          (* comment *)
  END_IF
END_IF
(*********************************)
```

例子:

```
IF (Level >= MaxLevel) OR (E_Stop = 1) THEN
  Pump := 0;
END_IF
```

图. 16 简单 IF 语句程序

基本上，如果条件表达式的结果为TRUE就执行语句。如果条件表达式的结果是FALSE，程序就执行END_IF后面的语句。条件表达式可以是简单关联的语句或用运算符（and,or等）连接的复合语句。

4.4.2 ELSE

它是简单IF语句的扩展。在IF结构中应该只有一个ELSE。

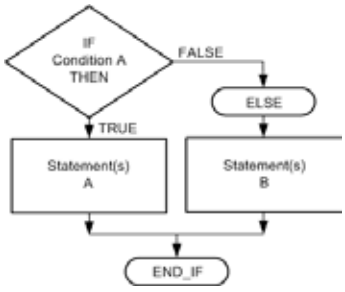


图. 17 IF- ELSE 语句

```
(*******************************)
(******     IF ELSE      ********)
(*******************************)
IF V1 > V2 THEN
  V3 := 99;          (* comment *)
ELSE
  V4 := 66;          (* comment *)
END_IF
(*******************************)
```

图. 18 IF – ELSE 程序

如果条件为TRUE，执行语句A。如果条件为FALSE，执行语句B。

4.4.3 ELSIF

运用一个或多个ELSE_IF语句可以实现多个不同的条件，而不用使用多个简单的IF语句创建复杂的程序逻辑。

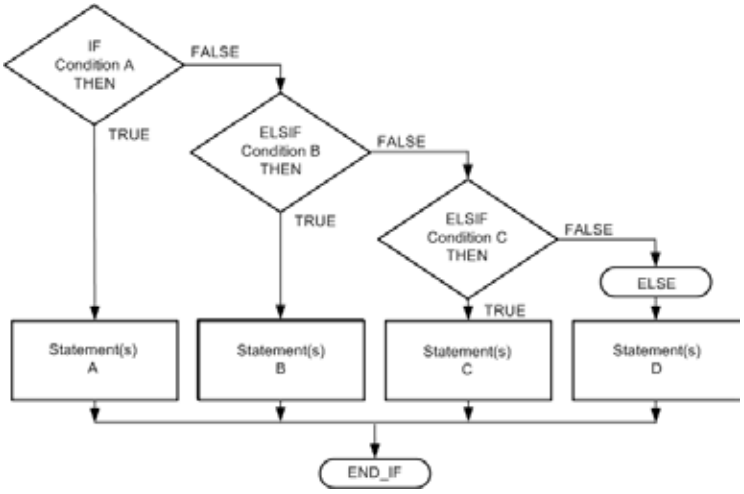


图. 19 IF-ELSIF-ELSE 流程图

```
(*****************************)
(****** IF, ELSIF, ELSE *******)
(*****************************)
IF V1 > V2 THEN
  V3 := 99;          (* comment *)
ELSIF V1 > V4 THEN
  V5 := 88;          (* comment *)
ELSIF V1 > V6 THEN
  V7 := 77;          (* comment *)
ELSE
  V8 := 66;          (* comment *)
END_IF
(*****************************)
```

图. 20 IF-ELSIF-ELSE 程序

处理器自上而下地执行判定。如果条件的结果为TRUE，那么就执行属于这个条件的指令和命令，之后处理器就跳到判断语句的结尾（END_IF）。在程序的一次循环中，无论下一个条件是否为TRUE，只有上面属于第一个条件TRUE的语句被执行。如果IF或ELSIF条件都不为TRUE，那么就执行属于ELSE下的指令。

任务: 气象站 – 第 I部分

用温度计来测量室外的温度，温度通过模拟量读取(1°=10)，并且要以文本的形式在房间里显示。

- 当温度在18°C以下，显示"cold"（冷）。
- 当温度是在18°C到25°C之间，显示"opt"（最佳）。
- 当温度是25°C以上，显示"hot"（热）。

用IF，ELSIF和ELSE语句实现这个任务。

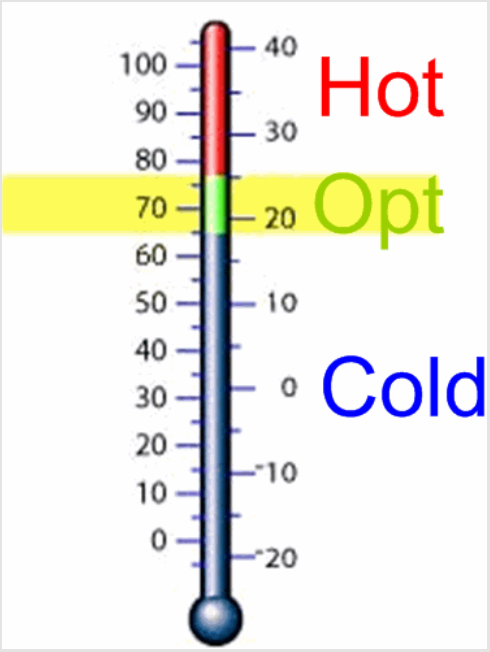


图. 21 温度计, 例子, IF

备注:

在ST中, 按如下指定一个字符串文本:
StringVar := ’ COLD’