Basics of ST

Excerpt of tutorial developed at University of Auckland by Gulnara Zhabelova

Based on Dr. Valeriy Vyatkin's book

"IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design", Second Edition

Basics of ST

- Each must end with a semi-colon (";")
- Basic statement
  - Assignment - :=
    - Q:=IN;
    - Q:=sin(angle);
    - Q := (IN1 + (IN2 / IN 3)) * IN4;
  - Conditional statement
    - IF / THEN / ELSE (simple binary switch)
    - CASE (enumarated switch)

```
1  ALGORITHM REQ IN ST:
2  (* Add your comment (as per IEC 61131-3) here
3   Normally executed algorithm
4   *)
5   ;
6   OUT := (X-Y)*(X+Y);
7  END_ALGORITHM
```

Basics of ST

- Conditional statement
  - IF / THEN / ELSE (simple binary switch)

Syntax

```
IF <BOOL expression> THEN
    <statements>
ELSIF <BOOL expression> THEN
<statements>
ELSE
<statements>
END_IF;
```

examples

```
(* simple condition *)
IF bCond THEN
Q1 := IN1;
Q2 := TRUE;
END_IF;
```

```
(* binary selection *)
IF bCond THEN
Q1 := IN1;
Q2 := TRUE;
ELSE
Q1 := IN2;
Q2 := FALSE;
END_IF;
```

Basics of ST

- Conditional statement
  - CASE (enumarated switch)

Syntax

CASE <DINT expression> **OF**
<value> :
<statements>
<value> , <value> :
<statements>;
<value> .. <value> :
<statements>;
ELSE
<statements>
END_CASE;

example

CASE iNumber **OF**
0 :
Alarm := TRUE;
AlarmText := '0 gives no result';
1 .. 3, 5 :
bPrime := TRUE;
4, 6 :
bPrime := FALSE;
ELSE
Alarm := TRUE;
AlarmText := 'I don't know after 6 !';
END_CASE;

DINT: 32 bit (default) integer

Basics of ST

- Loops
  - While
  - Repeat
  - For

Syntax
WHILE <BOOL expression> **DO**
<statements>
END_WHILE ;

example

```
iPos := 0;
WHILE iPos < iMax DO
MyArray[iPos] := 0;
iNbCleared := iNbCleared + 1;
END_WHILE;
```

Basics of ST

- Loops
  - While
  - Repeat
  - For

Syntax
REPEAT
<statements>
UNTIL <BOOL expression> **END_REPEAT**;

example

```
iPos := 0;
REPEAT
MyArray[iPos] := 0;
iNbCleared := iNbCleared + 1;
iPos := iPos + 1;
UNTIL iPos = iMax END_REPEAT;
```

Basics of ST

- Loops
  - While
  - Repeat
  - For

Syntax
FOR <*index*> := <*minimum*> **TO** <*maximum*> **BY** <*step*> **DO**
<statements>
END_FOR;

The "BY <*step*>" statement can be omitted. The default value for the step is 1.

example

```
iArrayDim := 10;
(* resets all items of the array to 0 *)
FOR iPos := 0 TO (iArrayDim - 1) DO
MyArray[iPos] := 0;
END_FOR;
(* set all items with odd index to 1 *)
FOR iPos := 1 TO 9 BY 2 DO
MyArray[ipos] := 1;
END_FOR;
```

Basics of ST

- Math operators

MIN	get the minimum of two integers
MAX	get the maximum of two integers
LIMIT	bound an integer to low and high limits
MOD	modulo (finds the remainder of division of one number by another.)
ODD	test if an integer is odd

Examples

```
Q := MIN (IN1, IN2);
Q := MAX (IN1, IN2);
Q := LIMIT (IMIN, IN, IMAX);  (* IMIN if IN < IMIN; IMAX if IN > IMAX; IN otherwise*)
Q := MOD (IN, BASE);
Q := ODD (IN);  (*TRUE if IN is odd. FALSE if IN is even.*)
```

Basics of ST

- Math operation (more)

ABS	absolute value
TRUNC	integer part
LOG	logarithm
POW,EXPT	power
SQRT	square root

SIN	sine
COS	cosine
TAN	tangent
ASIN	arc-sine
ACOS	arc-cosine
ATAN	arc-tangent
ATAN2	arc-tangent of Y / X

Examples

```
Q := ABS (IN);
Q := TRUNC (IN); (*integer part of IN*)
Q := LOG (IN); (*logarithm (base 10) of IN*)
Q := EXPT (IN, EXP); (*IN at the 'EXP' power*)
Q := SQRT (IN); (*square rot of IN*)
```

Examples

```
Q := SIN (IN);
```

Basics of ST

- Boolean operations

AND	performs a boolean AND
OR	performs a boolean OR
XOR	performs an exclusive OR
NOT	performs a boolean negation of its input

Examples

```
Q := IN1 AND IN2;
Q := IN1 OR IN2;
Q := IN1 XOR IN2;
Q := NOT IN;
```

Basics of ST

- Comparison

<	less than
>	greater than
<=	less or equal
>=	greater or equal
=	is equal
<>	is not equal

Basics of ST

- String operations

+	concatenation of strings
MLEN	get string length
DELETE	delete characters in a string
INSERT	insert characters in a string
FIND	find characters in a string
REPLACE	replace characters in a string
LEFT	extract a part of a string on the left
RIGHT	extract a part of a string on the right
MID	extract a part of a string
CHAR	build a single character string
ASCII	get the ASCII code of a character within a string
ATOH	converts string to integer using hexadecimal basis
HTOA	converts integer to string using hexadecimal basis
CRC16	CRC16 calculation
ArrayToString	copies elements of an SINT array to a STRING
StringToArray	copies characters of a STRING to an SINT array

Basics of ST

- Type conversion functions

ANY_TO_BOOL	converts to boolean
ANY_TO_SINT	converts to small (8 bit) integer
ANY_TO_INT	converst to 16 bit integer
ANY_TO_DINT	converts to integer (32 bit - default)
ANY_TO_LINT	converts to long (64 bit) integer
ANY_TO_REAL	converts to real
ANY_TO_LREAL	converts to double precicion real
ANY_TO_TIME	converts to time
ANY_TO_STRING	converts to character string

Examples

```
Q := ANY_TO_BOOL (IN);
```

Basics of ST

- Arrays

Syntax
FOR <*index*> := <*minimum*> **TO** <*maximum*> **BY** <*step*> **DO**
<statements>
END_FOR;

example

```
TheArray[1,7] := value;
result := SingleArray[i + 2];
```