akytec

CODESYS V3.5 CHEAT SHEET

COMMENTS

// single line comment	(* multi-line comment *)

VARIABLE DECLARATION

<Name> : <Type> := <initial_value>;

Naming rules for variables

If no initial value is specified, "null" is used

Spaces before and after ":", ":=" and ";" are not considered

Examples:

```
iVar: INT := 123;
rVar: REAL := 11.22;
wVar: WORD := 16#FFAB;
// maximum string length is 20 characters
sVar: STRING (20) := 'hello, world';
```

BASIC TYPES OF VARIABLE DECLARATION BLOCKS

Block type	Variables	Declaration place
VAR ... END_VAR	Local variables	In programs, function blocks, functions, methods
VAR_INPUT ... END_VAR	Input variables	
VAR_OUTPUT ... END_VAR	Output variables	
VAR_IN_OUT ... END_VAR	Input-output variables	
VAR CONSTANT ... END_VAR	Constants	
VAR_GLOBAL ... END_VAR	Global variables	In global variable list
VAR RETAIN ... END_VAR	RETAIN variables (non-volatile)	In programs and global variable lists
VAR_GLOBAL RETAIN ... END_VAR	Global RETAIN variables (non-volatile)	In global variable list
VAR_GLOBAL PERSISTENT RETAIN ... END_VAR	Global PERSISTENT variables (non-volatile)	In persistent variable list

DATA TYPES — Boolean and integer

Type	Size in bytes	Valid range
BOOL	1	TRUE/FALSE
BYTE/USINT	1	0...255
WORD/UINT	2	0...65535
DWORD/UDINT	4	$0...2^{32}-1$
LWORD/ULINT	8	$0...2^{64}-1$
SINT	1	-128...127
INT	2	-32768...32767
DINT	4	$-2^{31}...2^{31}-1$
LINT	8	$-2^{63}...2^{63}-1$

DATA TYPES	Boolean and integer

Notes:	1. Bitwise access is supported:
	// Write TRUE to the 15th bit of the variable of WORD type
	wVar.15 := TRUE;

DATA TYPES	Floating-point	
Type	**Size in bytes**	**Precision (number of significant digits after the decimal point)**
REAL	4	7-8
LREAL	8	15-17
Examples:	11.22 1.25E3 (exponential notation)	

DATA TYPES	Strings		
Type	**Character size in bytes**	**Encoding**	**Examples of literals (note the quotation marks)**
STRING	1	ASCII	sVar: STRING (20) := 'hello, world';
WSTRING	2	UTF-16	wsVar: WSTRING (20) := "hello, world"

Notes:

1. Strings in CODESYS are null-terminated. When they are declared, memory is automatically reserved for the string termination character, i.e. STRING(20) will take 21 bytes of memory, and WSTRING(20) will take 42 bytes of memory.

2. When declaring a string, its maximum length can be specified. There are no explicit restrictions for this, but consider the STANDARD LIBRARY table and the notes below.

3. To display Cyrillic characters in the visualization, only the WSTRING type should be used.

DATA TYPES		Date and time		
Type	**Size in bytes**	**Valid range**	**Description**	**Unit**
TIME	4	T#0ms...T#49d17h2m47s295ms	Time interval	Milliseconds
DT	4	DT#1970-1-1-0:0:0... DT#2106-02-07-06:28:15	Date and time	Seconds
DATE	4	DATE#1970-1-1-... DT#2106-02-07	Date	Seconds (time of day is not processed)
TOD	4	TOD#00:00:00... TOD#23:59:59.999	Time of day	Milliseconds

Notes:

1. Valid arithmetic operations

TIME + TIME = TIME	DT – TIME = DT
TIME – TIME = TIME	DT – DT = TIME
TIME x Integer = TIME	TOD + TIME = TOD
DT + TIME = DT	TOD – TIME = TOD

2. LTIME/LDT/LDATE/LTOD types differ in size (8 bytes), range and units (nanoseconds).

TYPE CONVERSION

```
TO_<type>
// conversion of a floating-point number to a string
sVar := TO_STRING(rVar);
// conversion of an integer variable into a time interval (in milliseconds)
tVar := TO_TIME(udiVar);
// other types are converted in the same way
```

COMPOSITE DATA TYPES (see CODESYS Help)

ARRAY	STRUCT	UNION	ENUMERATION

A LITTLE ABOUT ARRAYS (as the most used composite data type)

```
// Declaration of a one-dimensional array of four elements with initial values 10, 20, 30 and
// 40 respectively
VAR
aiData := ARRAY [0..3] OF INT := [10, 20, 30, 40];
END_VAR
// Addressing an array element in the program code
// iVar is assigned the value 10
iVar := aiData[0];
// Write the value 25 to the array element with the index 1
aiData[1] := 25;
```

BASIC OPERATORS

Mnemonics in ST language	Mnemonics in graphical languages	Description
Arithmetic operators		
:=	MOVE	Assignment of values of one variable (or literal) to another. In ST, assignment is performed " right to left", in graphical languages - "left to right".
+	ADD	Addition
-	SUB	Subtraction
*	MUL	Multiplication
/	DIV	Integer division
MOD		Non-negative integer remainder of division
Bitwise operators (applicable to BOOL and all integer types)		
NOT		Bitwise "NOT"
AND		Bitwise "AND"
OR		Bitwise "OR"
XOR		Bitwise "Exclusive OR"
Relational operators		
=	EQ	Comparison on equality
<>	NE	Comparison on inequality
>	GT	Comparison on "greater than"
<	LT	Comparison on "less than"
>=	GE	Comparison on "greater than or equal to"
<=	LE	Comparison on "less than or equal to"
Bitshift operators		
SHL		Left shift, padding with zeros
SHR		Right shift, padding with zeros
ROL		Cyclic left shift (left rotation)
ROR		Cyclic right shift (right rotation)

Selection operators	
SEL	Selection from two values of the same type
MUX	Selection from an arbitrary number of values of the same type (multiplexer)
LIMIT	Range limitation
Mathematical operators	
MIN	Selection of a minimum from an arbitrary number of values of the same type
MAX	Selection of a maximum from an arbitrary number of values of the same type
ABS	Absolute value
SQRT	Square root extraction
LN	Natural logarithm
LOG	Decimal logarithm
EXP	Exponential function
EXPT	Power function
TRUNC	Truncation of REAL value to DINT with discarding of fractional part
TRUNC INT	Truncation of REAL value to INT with discarding of fractional part
Trigonometric operators	
SIN	Sine
COS	Cosine
TAN	Tangent
ASIN	Arcsine
ACOS	Arccosine
ATAN	Arctangent
TIME	Time in milliseconds elapsed since system boot
SIXEOF, XSIZEOF	Object size in bytes
Notes:	1. The order in which operators are executed in an expression is determined by their priority and the parenthesis placing.

CONTROL STATEMENTS OF ST LANGUAGE

Statement	Description
IF...THEN...ELSIF...THEN...ELSE...END_IF	Conditional selection (check a set of conditions and execute subsequent operations for the first one that turns out to be true)
CASE...OF...:...ELSE...END_CASE	Multiple-choice operator (check an integer variable for a match with a set of values and execute subsequent operations in case of a match)
FOR...TO...BY...DO...END_FOR	Cycle with specified number of iterations
WHILE...DO...END_WHILE	Cycle with precondition (condition check before iteration)
REPEAT...UNTIL...END_REPEAT	Cycle with postcondition (condition check after iteration)
CONTINUE	Go to the next iteration of the loop
EXIT	Exit cycle
RETURN	Exit POU

BASIC TYPES OF PROGRAM OBJECTS (POU)

Name	Own memory and saving of variable values between calls	Declaration of instances
Function (FUN)	No	No
Function Block (FB)	Yes	Yes
Program (PRG)	Yes	No

STANDARD LIBRARY (basic functions and function blocks)

Name	Description
String functions	
CONCAT	Merges two strings to a single string
DELETE	Deletes a substring from a string
FIND	Search for a substring in a string
INSERT	Inserts a substring into a string
LEFT	Returns a specified number of characters of a string, starting from left
LEN	Returns the length of a string
MID	Returns a specified number of characters of a string, starting from a specified position
REPLACE	Replaces a specific number of characters of a string by another string
RIGHT	Returns a specified number of characters of a string, starting from right

Notes:

1. Functions to handle strings of WSTRING type and timers with LTIME type are available in the Standard64 library.
2. Other frequently used blocks (linear scaling, pulse generator, BLINK, PID controller, etc.) are available in the Util library.
3. String functions can handle strings up to 255 characters in length only. To work with longer strings use functions of StringUtils library.
4. Advanced functions for string processing (parsing, formatting, etc.) are available in the AkytecStringUtils library.

Edge detectors	
R_TRIG	Generates a single pulse on the rising edge (R_TRIG) or falling edge (F_TRIG) of the specified Boolean variable
F_TRIG	
Triggers	
SR	Set-dominant trigger
RS	Reset-dominant trigger
Counters	
CTU	Incremental pulse counter
CTD	Decremental pulse counter
CTUD	Pulse counter with support for both counting directions
Timers	
TP	Generator of a logic signal of a specified length
TON	Timer with turn-on delay
TOF	Timer with turn-off delay

OTHER USEFUL LIBRARIES FROM THE CODESYS DISTRIBUTION PACKAGE

Library	Description	Documentation
CAA Memory	Memory operations (memory blocks copying, comparing, filling etc.)	Open
CAA File	File operations (reading/writing)	Open
CAA SerialCom	Working with COM port to implement non-standard exchange protocols	Open
CAA NetBaseServices	Working with TCP/UDP to implement non-standard exchange protocols	Open
Visu Utils	Visualization control from the program code (switch between screens, open windows, etc.)	Open

USEFUL LINKS

CODESYS Online Help