SIEMENS

SIMATIC

STEP 7
Standards compliance according to IEC 61131-3 (3rd Edition)

Function Manual

04/2015

A5E35932122-AA

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E35932122-AA
℗ 05/2015 Subject to change

Table of contents

Introduction 1

The **IEC61131** standard is applicable for the programmable logic controllers (PLC).

In accordance with the rules of the European Union, this international standard has been accepted in Germany as DIN EN 61131, in France as NF EN 61131, and in Britain as BS EN 61131.

The most important parts of the standard are quoted below. Quotes are in italics.

Part 3 of this standard defines the "Area of application" in Section 1.

“*This Part of IEC 61131 specifies syntax and semantics of programming languages for programmable controllers as defined in Part 1 of the IEC 61131*.

The functions of program entry, testing, monitoring, operating system, etc., are specified in Part 1.

This part of IEC 61131 specifies the syntax and semantics of a unified suite of programming languages for PLCs. These consist of textual languages, IL (Instruction List) and ST (Structured Text), and two graphical languages, LD (Ladder Diagram) and FBD (Function Block Diagram).

Sequential Function Chart (SFC) elements are defined for structuring the internal organization of programmable controller programs and function blocks. Also, configuration elements are defined which support the installation of programmable controller programs into programmable controller systems...."

The programming language elements defined in this part may be used in an interactive programming environment. The specification of such environments is beyond the scope of this standard; however, such an environment shall be capable of producing textual or graphic program documentation in the formats specified in this part.

Section 5 "Standards compliance" specifies:

"A programmable controller system, as defined in IEC 61131-1, which claims to comply, wholly or partially, with the requirements of this Part of IEC 61131 shall do so only as described below: … “

Section 5.3 "Compliance declaration of the manufacturer" specifies:

"The manufacturer may define any consistent subset of characteristics that are listed in the characteristic tables, and must make known the available subset in the "Compliance declaration of the manufacturer".

The compliance declaration of the manufacturer must be contained in the documentation that is included with the system, or it must be generated by the system itself.

The format of the compliance declaration of the manufacturer must provide the following information. Figure 4 in the standard shows an example.

- *The general information shall include the name and address of the manufacturer, the name and version of the product, the type and version of the controller and the revision date.*
- *The number of the corresponding characteristics table, the characteristics number and the applicable programming language must be specified for each implemented characteristic.*
- *The title and subtitle of the characteristics table, the description of the characteristic, examples, manufacturers remarks etc. are optional.*

Table and characteristics that are not implemented can be omitted."

Standards compliance in STEP 7 2

The programming languages of SIMATIC STEP 7 in TIA Portal meet the requirements of IEC 61131-3 in the characteristics described in the following table:

• Instruction List	AWL/STL	(corresponds to IEC 61131-3 language "AWL/STL")
• Ladder Logic	KOP/LAD	(corresponds to IEC 61131-3 language "KOP/LD")
• Function Block Diagram	FUP/FBD	(corresponds to IEC 61131-3 language "FUP/FBD")
• Structured Control Language (SCL)	SCL	(corresponds to IEC 61131-3 language "ST")
• S7-GRAPH	GRAPH	(corresponds to IEC 61131-3 language "AS/SFC")

.

The standard defines all standardized language elements in the form of tables, the rows of which refer to the realized feature with a number.

The language elements which are realized in STEP 7 according to the standard are specified below.

A good knowledge of the norm mentioned is a prerequisite for understanding the following tables.

The English version of **DIN EN 61131-3 : 2013-02 (3rd Edition)** is available from Beuth Verlag GmbH, 10787 Berlin, Fax +49 (030) 2601-1260.

IEC 61131-3 “PLC Programming Languages”
Implementer: Siemens AG.
Product: STEP 7 in TIA Portal
Date: 2014-07-21

This Product complies with the requirements of the standard for the following language features:

Feature No.	**Table Number and Title / Feature Description**			**Compliantly** implemented in the language (✓)				Implementer’s note
				LD	**FBD**	**IL**	**ST**	
	Table 1 – Character set							
1	“ISO/IEC 10646 2011			✓	✓	✓	✓	
2a	Lower case characters:	`a, b, c`		✓	✓	✓	✓	
2b	Number sign:	#	See Table 2	✓	✓	✓	✓	
2c	Dollar sign:	$	See Table 3	✓	✓	✓	✓	

	Table 2 - Identifiers					
1	Upper case letters and numbers: `IW215`	✓	✓	✓	✓	
2	Upper and lower case letters, numbers, embedded underscore	✓	✓	✓	✓	
3	Upper and lower case, numbers, leading or embedded underscore	✓	✓	✓	✓	

	Table 3 - Comments					
1	Single-line comment with // …			✓	✓	
2a	Multi-line comment with (* … *)				✓	
2b	Multi-line comment with /* … */					
3a	Nested comment with (* .. (* .. *) ..*)				✓	
3b	Nested comment with /* .. /* .. */ .. */					

	Table 4 - Pragma					
1	Pragma with { … } curly brackets			✓	✓	In source files of blocks

	Table 5 – Numeric literals					
1	Integer literal `-12`	✓	✓	✓	✓	
2	Real literal `-12.0`	✓	✓	✓	✓	
3	Real literals with exponent `1.34E-12`	✓	✓	✓	✓	
4	Binary literal `2#1111_1111`	✓	✓	✓	✓	
5	Octal literals `8#377`	✓	✓	✓	✓	
6	Hexadecimal literal `16#FF`	✓	✓	✓	✓	
7	Boolean zero and one	✓	✓	✓	✓	
8	Boolean FALSE and TRUE	✓	✓	✓	✓	
9	Typed literal `INT#-123`	✓	✓	✓	✓	

	Table 6 – Character string literals					
	Single-byte characters or character strings with ‘ ‘					
1a	Empty string (length zero)	✓	✓	✓	✓	
1b	String of length one or character `CHAR` containing a single character	✓	✓	✓	✓	
1c	String of length one or character `CHAR` containing the “space” character	✓	✓	✓	✓	
1d	String of length one or character `CHAR` containing the “single quote” character	✓	✓	✓	✓	Possible using feature 1g
1e	String of length one or character `CHAR` containing the “double quote” character	✓	✓	✓	✓	

	Table 6 – Character string literals					
1f	Support of two character combinations of Table 7	✓	✓	✓	✓	
1g	Support of a character representation with '$' and two hexadecimal characters	✓	✓	✓	✓	
	Double-byte characters or character strings with **" "**					
2a	Empty string (length zero)					
2b	String of length one or character `WCHAR` containing a single character					
2c	String of length one or character `WCHAR` containing the "space" character					
2d	String of length one or character `WCHAR` containing the "single quote" character					
2e	String of length one or character `WCHAR` containing the "double quote" character					
2f	Support of two character combinations of Table 7					
2g	Support of a character representation with '$' and four hexadecimal characters					
	Single-byte typed characters or string literals with #					
3a	Typed string	✓	✓	✓	✓	
3b	Typed character	✓	✓	✓	✓	
	Double-byte typed string literals with # (NOTE)					
4a	Typed double-byte string (using "double quote" charac-ter)					
4b	Typed double-byte character (using "double quote" char-acter)					
4c	Typed double-byte string (using "single quote" character)					
4d	Typed double-byte character (using "single quote" char-acter)					

	Table 7 – Two-character combinations in character strings					
1	Dollar sign	✓	✓	✓	✓	
2	Single quote	✓	✓	✓	✓	
3	Line feed	✓	✓	✓	✓	
4	Newline	✓	✓	✓	✓	
5	Form feed (page)	✓	✓	✓	✓	
6	Carriage return	✓	✓	✓	✓	
7	Tabulator	✓	✓	✓	✓	
8	Double quote	✓	✓	✓	✓	

	Table 8 – Duration literals					
	Duration abbreviations					
1a	`d`	✓	✓	✓	✓	
1b	`h`	✓	✓	✓	✓	
1c	`m`	✓	✓	✓	✓	
1d	`s`	✓	✓	✓	✓	
1e	`ms`	✓	✓	✓	✓	
1f	`us` (no µ available.)					
1g	`ns`					
	Duration literals without underscore					
2a	short prefix	✓	✓	✓	✓	
2b	long prefix	✓	✓	✓	✓	
	Duration literals with underscore					
3a	short prefix	✓	✓	✓	✓	
3b	long prefix	✓	✓	✓	✓	

	Table 9 – Date and time of day literals					
1a	Date literal (long prefix)	✓	✓	✓	✓	
1b	Date literal (short prefix)	✓	✓	✓	✓	
2a	Long date literal (long prefix)	✓	✓	✓	✓	
2b	Long date literal (short prefix)	✓	✓	✓	✓	
3a	Time of day literal (long prefix)	✓	✓	✓	✓	
3b	Time of day literal (short prefix)	✓	✓	✓	✓	
4a	Long time of day literal (short prefix)	✓	✓	✓	✓	
4b	Long time of day literal (long prefix)	✓	✓	✓	✓	
5a	Date and time literal (long prefix)	✓	✓	✓	✓	
5b	Date and time literal (short prefix)	✓	✓	✓	✓	
6a	Long date and time literal (long prefix)	✓	✓	✓	✓	
6b	Long date and time literal (short prefix)	✓	✓	✓	✓	

	Tabelle 10 – Elementare Datentypen					
1	Boolean `BOOL`	✓	✓	✓	✓	
2	Short integer `SINT`	✓	✓	✓	✓	
3	Integer `INT`	✓	✓	✓	✓	
4	Double integer `DINT`	✓	✓	✓	✓	
5	Long integer `LINT`	✓	✓	✓	✓	
6	Unsigned short integer `USINT`	✓	✓	✓	✓	
7	Unsigned integer `UINT`	✓	✓	✓	✓	
8	Unsigned double integer `UDINT`	✓	✓	✓	✓	
9	Unsigned long integer `ULINT`	✓	✓	✓	✓	
10	Real numbers `REAL`	✓	✓	✓	✓	

	Tabelle 10 – Elementare Datentypen					
11	Long reals `LREAL`	✓	✓	✓	✓	
12a	Duration `TIME`	✓	✓	✓	✓	
12b	Long duration `LTIME`	✓	✓	✓	✓	
13a	Date (only) `DATE`	✓	✓	✓	✓	
13b	Long date (only) `LDATE`					
14a	Time of day (only) `TIME_OF_DAY` or `TOD`	✓	✓	✓	✓	
14b	Long time of day (only) `LTIME_OF_DAY or LTOD`	✓	✓	✓	✓	
15a	Date and time of Day) `DATE_AND_TIME` or `DT`	✓	✓	✓	✓	
15b	Long date and time of day `LDATE_AND_TIME` or `LDT`	✓	✓	✓	✓	
16a	Variable-length single-byte character string `STRING`	✓	✓	✓	✓	
16b	Variable-length double-byte character string `WSTRING`					
17a	Single-byte character `CHAR`	✓	✓	✓	✓	
17b	Double-byte character `WCHAR`					
18	Bit string of length 8 `BYTE`	✓	✓	✓	✓	
19	Bit string of length 16 `WORD`	✓	✓	✓	✓	
20	Bit string of length 32 `DWORD`	✓	✓	✓	✓	
21	Bit string of length 64 `LWORD`	✓	✓	✓	✓	

	Table 11 – Declaration of user-defined data types and initialization					
1a 1b	Enumerated data types					
2a 2b	Data types with named values					
3a 3b	Subrange data types					
4a 4b	Array data types					
5a 5b	FB types and classes as array elements					
6a 6b	Structured data type	✓	✓	✓	✓	
7a 7b	FB types and classes as structure elements					
8a 8b	Structured data type with relative addressing `AT`					
9a	Structured data type with relative addressing `AT` and `OVERLAP`					
10a 10b	Directly represented elements of a structure – partly specified using " *					

	Table 11 – Declaration of user-defined data types and initialization					
11a 11b	Directly derived data types					
12	Initialization using constant expressions					

	Table 12 – Reference operations					
	Declaration					
1	Declaration of a reference type					
	Assignment and comparison					
2a	Assignment reference to reference					
2b	Assignment reference to parameter of function, function block and method					
2c	Comparison with NULL					
	Referencing					
3a	`REF`(<variable>) Provides of the typed reference to the variable					
3b	`REF`(<function block instance>) Provides the typed reference to the function block or class instance					
	Dereferencing					
4	`<reference>^` Provides the content of the variable or the content of the instance to which the reference variable contains the reference					

	Table 13 – Declaration of variables					
1	Variable with elementary data type	✓	✓	✓	✓	
2	Variable with user-defined data type	✓	✓	✓	✓	
3	Array	✓	✓	✓	✓	
4	Reference					

	Table 14 – Initialization of variables					
1	Initialization of a variable with elementary data type	✓	✓	✓	✓	
2	Initialization of a variable with user-defined data type	✓	✓	✓	✓	
3	Array	✓	✓	✓	✓	
4	Declaration and initialization of constants	✓	✓	✓	✓	Global con-stants
5	Initialization using constant expressions					
6	Initialization of a reference					

	Table 15 – Variable-length ARRAY variables					
1	Declaration `using *` `ARRAY [*, *, . . . ] OF data type`					
	Standard functions LOWER_BOUND / UPPER_BOUND					
2a	Graphical representation					
2b	Textual representation					

	Table 16 – Directly represented variables					
	Location (NOTE 1)					
1	Input location `I`	✓	✓	✓	✓	
2	Output location `Q`	✓	✓	✓	✓	
3	Memory location `M`	✓	✓	✓	✓	
	Size					
4a	Single bit size X	✓	✓	✓	✓	
4b	Single bit size None	✓	✓	✓	✓	
5	Byte (8 bits) size `B`	✓	✓	✓	✓	
6	Word (16 bits) size `W`	✓	✓	✓	✓	
7	Double word (32 bits) size `D`	✓	✓	✓	✓	
8	Long (quad) word (64 bits) size `L`					
	Addressing					
9	Simple addressing `%IX1`					
10	Hierarchical addressing using “. `%QX7.5`	✓	✓	✓	✓	
11	Partly specified variables using asterisk ”*”					

	Table 17 – Partial access of ANY_BIT variables					
	Data Type - Access to					
1a	`BYTE –` bit `VB2.%X0`	✓	✓	✓	✓	
1b	`WORD –` bit `VW3.%X15`	✓	✓	✓	✓	
1c	`DWORD –` bit	✓	✓	✓	✓	
1d	`LWORD –` bit	✓	✓	✓	✓	
2a	`WORD –` byte `VW4.%B0`	✓	✓	✓	✓	
2b	`DWORD –` byte	✓	✓	✓	✓	
2c	`LWORD –` byte	✓	✓	✓	✓	
3a	`DWORD –` word	✓	✓	✓	✓	
3b	`LWORD –` word	✓	✓	✓	✓	
4	`LWORD –` dword `VL5.%D1`	✓	✓	✓	✓	

	Table 18 – Execution control graphically using EN and ENO					
1	Usage without `EN` and `ENO`	✓	✓		✓	Depends on the used function
2	Usage of `EN` only (without `ENO`)	✓	✓		✓	Depends on the used function
3	Usage of `ENO` only (without `EN`)					
4	Usage of `EN` and `ENO`	✓	✓		✓	Depends on the used function

	Table 19 – Function declaration					
1a	Without result `FUNCTION ... END_FUNCTION`	✓	✓	✓	✓	Void used to define

1b	With result
	`FUNCTION <name> : <data type>`
	`END_FUNCTION`
		✓	✓	✓	✓	

2a	Inputs `VAR_INPUT...END_VAR`	✓	✓	✓	✓	
2b	Outputs `VAR_OUTPUT...END_VAR`	✓	✓	✓	✓	
2c	In-outs `VAR_IN_OUT...END_VAR`	✓	✓	✓	✓	
2d	Temporary variables `VAR_TEMP...END_VAR`	✓	✓	✓	✓	
2e	Temporary variables `VAR...END_VAR`					
2f	External variables `VAR_EXTERNAL...END_VAR`					
2g	External constants `VAR_EXTERNAL_CONSTANT...END_VAR`					
3a	Initialization of inputs					
3b	Initialization of outputs					
3c	Initialization of temporary variables					

	Table 20 – Function call					
1a	Complete formal call (textual only) NOTE This is used if EN/ENO is necessary in calls.	✓	✓	✓	✓	
1b	Incomplete formal call (textual only) NOTE This is used if EN/ENO is not necessary in calls.					
2	Non-formal call (textual only) (fix order and complete) NOTE This is used for call of standard functions without formal names.					
3	Function without function result	✓	✓	✓	✓	Void used to define
4	Graphical representation	✓	✓			
5	Usage of negated boolean input and output in graphical representation	✓	✓			
6	Graphical usage of `VAR_IN_OUT`					

	Table 21 – Typed and overloaded functions					
1a	Overloaded function `ADD` (`ANY`_Num to `ANY`_Num)				✓	
1b	Conversion of inputs `ANY_ELEMENT_TO_INT`					
2a	Typed functions: `ADD_INT`	✓	✓			Using the cor-rect function is supported by the editor
2b	Conversion: `WORD_TO_INT`	✓	✓	✓	✓	

	Table 22 – Data type conversion function					
1a	Typed conversion `input_TO_output`	✓	✓	✓	✓	
1b	Overloaded conversion `TO_output`					
2a	“Old” overloaded truncation `TRUNC`				✓	
2b	Typed truncation `input_TRUNC_output`	✓	✓			
2c	Overloaded truncation `TRUNC_output`					
3a	Typed `input_BCD_TO_output`	✓	✓		✓	Convert of BCD16 and BCD32
3b	Overloaded `BCD_TO_output`					

	Table 22 – Data type conversion function					
4a	Typed `input TO BCD output`					
4b	Overloaded `TO BCD output`					

	Table 23 – Data type conversion of numeric data types					
1	LREAL _TO_ REAL	✓	✓	✓	✓	
2	LREAL _TO_ LINT	✓	✓	✓	✓	
3	LREAL _TO_ DINT	✓	✓		✓	
4	LREAL _TO_ INT	✓	✓		✓	
5	LREAL _TO_ SINT	✓	✓		✓	
6	LREAL _TO_ ULINT	✓	✓	✓	✓	
7	LREAL _TO_ UDINT	✓	✓		✓	
8	LREAL _TO_ UINT	✓	✓		✓	
9	LREAL _TO_ USINT	✓	✓		✓	
10	REAL _TO_ LREAL	✓	✓	✓	✓	
11	REAL _TO_ LINT	✓	✓		✓	
12	REAL _TO_ DINT	✓	✓		✓	
13	REAL _TO_ INT	✓	✓		✓	
14	REAL _TO_ SINT	✓	✓		✓	
15	REAL _TO_ ULINT	✓	✓		✓	
16	REAL _TO_ UDINT	✓	✓		✓	
17	REAL _TO_ UINT	✓	✓		✓	
18	REAL _TO_ USINT	✓	✓		✓	
19	LINT _TO_ LREAL	✓	✓	✓	✓	
20	LINT _TO_ REAL	✓	✓		✓	
21	LINT _TO_ DINT	✓	✓	✓	✓	
22	LINT _TO_ INT	✓	✓		✓	
23	LINT _TO_ SINT	✓	✓		✓	
24	LINT _TO_ ULINT	✓	✓	✓	✓	
25	LINT _TO_ UDINT	✓	✓		✓	
26	LINT _TO_ UINT	✓	✓		✓	
27	LINT _TO_ USINT	✓	✓		✓	
28	DINT _TO_ LREAL	✓	✓		✓	
29	DINT _TO_ REAL	✓	✓		✓	
30	DINT _TO_ LINT	✓	✓	✓	✓	
31	DINT _TO_ INT	✓	✓		✓	
32	DINT _TO_ SINT	✓	✓		✓	
33	DINT _TO_ ULINT	✓	✓		✓	
34	DINT _TO_ UDINT	✓	✓		✓	
35	DINT _TO_ UINT	✓	✓		✓	
36	DINT _TO_ USINT	✓	✓		✓	

	Table 23 – Data type conversion of numeric data types					
37	INT _TO_ LREAL	✓	✓		✓	
38	INT _TO_ REAL	✓	✓		✓	
39	INT _TO_ LINT	✓	✓		✓	
40	INT _TO_ DINT	✓	✓		✓	
41	INT _TO_ SINT	✓	✓		✓	
42	INT _TO_ ULINT	✓	✓		✓	
43	INT _TO_ UDINT	✓	✓		✓	
44	INT _TO_ UINT	✓	✓		✓	
45	INT _TO_ USINT	✓	✓		✓	
46	SINT _TO_ LREAL	✓	✓		✓	
47	SINT _TO_ REAL	✓	✓		✓	
48	SINT _TO_ LINT	✓	✓		✓	
49	SINT _TO_ DINT	✓	✓		✓	
50	SINT _TO_ INT	✓	✓		✓	
51	SINT _TO_ ULINT	✓	✓		✓	
52	SINT _TO_ UDINT	✓	✓		✓	
53	SINT _TO_ UINT	✓	✓		✓	
54	SINT _TO_ USINT	✓	✓		✓	
55	ULINT _TO_ LREAL	✓	✓	✓	✓	
56	ULINT _TO_ REAL	✓	✓		✓	
57	ULINT _TO_ LINT	✓	✓	✓	✓	
58	ULINT _TO_ DINT	✓	✓		✓	
59	ULINT _TO_ INT	✓	✓		✓	
60	ULINT _TO_ SINT	✓	✓		✓	
61	ULINT _TO_ UDINT	✓	✓	✓	✓	
62	ULINT _TO_ UINT	✓	✓		✓	
63	ULINT _TO_ USINT	✓	✓		✓	
64	UDINT _TO_ LREAL	✓	✓		✓	
65	UDINT _TO_ REAL	✓	✓		✓	
66	UDINT _TO_ LINT	✓	✓		✓	
67	UDINT _TO_ DINT	✓	✓		✓	
68	UDINT _TO_ INT	✓	✓		✓	
69	UDINT _TO_ SINT	✓	✓		✓	
70	UDINT _TO_ ULINT	✓	✓	✓	✓	
71	UDINT _TO_ UINT	✓	✓		✓	
72	UDINT _TO_ USINT	✓	✓		✓	
73	UINT _TO_ LREAL	✓	✓		✓	
74	UINT _TO_ REAL	✓	✓		✓	
75	UINT _TO_ LINT	✓	✓		✓	
76	UINT _TO_ DINT	✓	✓		✓	
77	UINT _TO_ INT	✓	✓		✓	
78	UINT _TO_ SINT	✓	✓		✓	

	Table 23 – Data type conversion of numeric data types					
79	UINT _TO_ ULINT	✓	✓		✓	
80	UINT _TO_ UDINT	✓	✓		✓	
81	UINT _TO_ USINT	✓	✓		✓	
82	USINT _TO_ LREAL	✓	✓		✓	
83	USINT _TO_ REAL	✓	✓		✓	
84	USINT _TO_ LINT	✓	✓		✓	
85	USINT _TO_ DINT	✓	✓		✓	
86	USINT _TO_ INT	✓	✓		✓	
87	USINT _TO_ SINT	✓	✓		✓	
88	USINT _TO_ ULINT	✓	✓		✓	
89	USINT _TO_ UDINT	✓	✓		✓	
90	USINT _TO_ UINT	✓	✓		✓	

	Table 24 – Data type conversion of bit data types					
1	LWORD _TO_ DWORD	✓	✓		✓	
2	LWORD _TO_ WORD	✓	✓	✓	✓	
3	LWORD _TO_ BYTE	✓	✓		✓	
4	LWORD _TO_ BOOL	✓	✓		✓	
5	DWORD _TO_ LWORD	✓	✓		✓	
6	DWORD _TO_ WORD	✓	✓		✓	
7	DWORD _TO_ BYTE	✓	✓		✓	
8	DWORD _TO_ BOOL	✓	✓		✓	
9	WORD _TO_ LWORD	✓	✓	✓	✓	
10	WORD _TO_ DWORD	✓	✓		✓	
11	WORD _TO_ BYTE	✓	✓		✓	
12	WORD _TO_ BOOL	✓	✓		✓	
13	BYTE _TO_ LWORD	✓	✓		✓	
14	BYTE _TO_ DWORD	✓	✓		✓	
15	BYTE _TO_ WORD	✓	✓		✓	
16	BYTE _TO_ BOOL	✓	✓		✓	
17	BYTE _TO_ CHAR	✓	✓		✓	
18	BOOL _TO_ LWORD	✓	✓		✓	
19	BOOL _TO_ DWORD	✓	✓		✓	
20	BOOL _TO_ WORD	✓	✓		✓	
21	BOOL _TO_ BYTE	✓	✓		✓	
22	CHAR _TO_ BYTE	✓	✓		✓	
23	CHAR _TO_ WORD	✓	✓		✓	
24	CHAR _TO_ DWORD	✓	✓		✓	
25	CHAR _TO_ LWORD	✓	✓		✓	
26	WCHAR _TO_ WORD					

	Table 24 – Data type conversion of bit data types					
27	WCHAR _TO_ DWORD					
28	WCHAR _TO_ LWORD					

	Table 25 – Data type conversion of bit and numeric types					
1	LWORD _TO_ LREAL	✓	✓		✓	
2	DWORD _TO_ REAL	✓	✓		✓	
3	LWORD _TO_ LINT	✓	✓		✓	
4	LWORD _TO_ DINT	✓	✓		✓	
5	LWORD _TO_ INT	✓	✓		✓	
6	LWORD _TO_ SINT	✓	✓		✓	
7	LWORD _TO_ ULINT	✓	✓	✓	✓	
8	LWORD _TO_ UDINT	✓	✓		✓	
9	LWORD _TO_ UINT	✓	✓		✓	
10	LWORD _TO_ USINT	✓	✓		✓	
11	DWORD _TO_ LINT	✓	✓		✓	
12	DWORD _TO_ DINT	✓	✓		✓	
13	DWORD _TO_ INT	✓	✓		✓	
14	DWORD _TO_ SINT	✓	✓		✓	
15	DWORD _TO_ ULINT	✓	✓		✓	
16	DWORD _TO_ UDINT	✓	✓		✓	
17	DWORD _TO_ UINT	✓	✓		✓	
18	DWORD _TO_ USINT	✓	✓		✓	
19	WORD _TO_ LINT	✓	✓		✓	
20	WORD _TO_ DINT	✓	✓		✓	
21	WORD _TO_ INT	✓	✓		✓	
22	WORD _TO_ SINT	✓	✓		✓	
23	WORD _TO_ ULINT	✓	✓		✓	
24	WORD _TO_ UDINT	✓	✓		✓	
25	WORD _TO_ UINT	✓	✓		✓	
26	WORD _TO_ USINT	✓	✓		✓	
27	BYTE _TO_ LINT	✓	✓		✓	
28	BYTE _TO_ DINT	✓	✓		✓	
29	BYTE _TO_ INT	✓	✓		✓	
30	BYTE _TO_ SINT	✓	✓		✓	
31	BYTE _TO_ ULINT	✓	✓		✓	
32	BYTE _TO_ UDINT	✓	✓		✓	
33	BYTE _TO_ UINT	✓	✓		✓	
34	BYTE _TO_ USINT	✓	✓		✓	
35	BOOL _TO_ LINT	✓	✓		✓	
36	BOOL _TO_ DINT	✓	✓		✓	
37	BOOL _TO_ INT	✓	✓		✓	

	Table 25 – Data type conversion of bit and numeric types					
38	BOOL _TO_ SINT	✓	✓		✓	
39	BOOL _TO_ ULINT	✓	✓		✓	
40	BOOL _TO_ UDINT	✓	✓		✓	
41	BOOL _TO_ UINT	✓	✓		✓	
42	BOOL _TO_ USINT	✓	✓		✓	
43	LREAL _TO_ LWORD	✓	✓		✓	
44	REAL _TO_ DWORD	✓	✓		✓	
45	LINT _TO_ LWORD	✓	✓		✓	
46	LINT _TO_ DWORD	✓	✓		✓	
47	LINT _TO_ WORD	✓	✓		✓	
48	LINT _TO_ BYTE	✓	✓		✓	
49	DINT _TO_ LWORD	✓	✓		✓	
50	DINT _TO_ DWORD	✓	✓		✓	
51	DINT _TO_ WORD	✓	✓		✓	
52	DINT _TO_ BYTE	✓	✓		✓	
53	INT _TO_ LWORD	✓	✓		✓	
54	INT _TO_ DWORD	✓	✓		✓	
55	INT _TO_ WORD	✓	✓		✓	
56	INT _TO_ BYTE	✓	✓		✓	
57	SINT _TO_ LWORD	✓	✓		✓	
58	SINT _TO_ DWORD	✓	✓		✓	
59	SINT _TO_ WORD	✓	✓		✓	
60	SINT _TO_ BYTE	✓	✓		✓	
61	ULINT _TO_ LWORD	✓	✓	✓	✓	
62	ULINT _TO_ DWORD	✓	✓		✓	
63	ULINT _TO_ WORD	✓	✓		✓	
64	ULINT _TO_ BYTE	✓	✓		✓	
65	UDINT _TO_ LWORD	✓	✓		✓	
66	UDINT _TO_ DWORD	✓	✓		✓	
67	UDINT _TO_ WORD	✓	✓		✓	
68	UDINT _TO_ BYTE	✓	✓		✓	
69	UINT _TO_ LWORD	✓	✓		✓	
70	UINT _TO_ DWORD	✓	✓		✓	
71	UINT _TO_ WORD	✓	✓		✓	
72	UINT _TO_ BYTE	✓	✓		✓	
73	USINT _TO_ LWORD	✓	✓		✓	
74	USINT _TO_ DWORD	✓	✓		✓	
75	USINT _TO_ WORD	✓	✓		✓	
76	USINT _TO_ BYTE	✓	✓		✓	

	Table 26 – Data type conversion of date and time types					
1	LTIME _TO_ TIME	✓	✓		✓	
2	TIME _TO_ LTIME	✓	✓		✓	
3	LDT _TO_ DT	✓	✓		✓	
4	LDT _TO_ DATE	✓	✓		✓	
	LDT _TO_ LTOD	✓	✓		✓	
6	LDT _TO_ TOD	✓	✓		✓	
7	DT _TO_ LDT	✓	✓		✓	
8	DT _TO_ DATE	✓	✓		✓	
9	DT _TO_ LTOD	✓	✓		✓	
10	DT _TO_ TOD	✓	✓		✓	
11	LTOD _TO_ TOD	✓	✓		✓	
12	TOD _TO_ LTOD	✓	✓		✓	

	Table 27 – Data type conversion of character types					
1	WSTRING _TO_ STRING					
2	WSTRING _TO_ WCHAR					
3	STRING _TO_ WSTRING					
4	STRING _TO_ CHAR	✓	✓		✓	
5	WCHAR _TO_ WSTRING					
6	WCHAR _TO_ CHAR					
7	CHAR _TO_ STRING	✓	✓		✓	
8	CHAR _TO_ WCHAR					

	Table 28 – Numerical and arithmetic functions					
	General functions					
1	`ABS(x)`	✓	✓	✓	✓	
2	`SQRT(x)`	✓	✓	✓	✓	
	Logarithmic functions					
3	`LN(x)`	✓	✓	✓	✓	
4	`LOG(x)`					
5	`EXP(x)`	✓	✓	✓	✓	

	Table 28 – Numerical and arithmetic functions					
	Trigonometric functions					
6	`SIN(x)`	✓	✓	✓	✓	
7	`COS(x)`	✓	✓	✓	✓	
8	`TAN(x)`	✓	✓	✓	✓	
9	`ASIN(x)`	✓	✓	✓	✓	
10	`ACOS(x)`	✓	✓	✓	✓	
11	`ATAN(x)`	✓	✓	✓	✓	

12	`ATAN2(y, x)`
	`              +--------+`
	`              | ATAN2  |`
	`ANY_REAL--|Y       |--ANY_REAL`
	`ANY_REAL--|X       |`
	`              +--------+`					

	Table 29 – Arithmetic functions					
	Extensible arithmetic functions					
1	Addition	✓	✓		✓	
2	Multiplication	✓	✓		✓	
	Non-extensible arithmetic functions					
3	Subtraction	✓	✓		✓	
4	Division	✓	✓		✓	
5	Modulo	✓	✓		✓	
6	Exponentiation	✓	✓		✓	
7	Move	✓	✓		✓	

	Table 30 – Bit shift functions					
1	Shift left SHL	✓	✓	✓	✓	
2	Shift right SHR	✓	✓	✓	✓	
3	Rotation left ROL	✓	✓	✓	✓	
4	Rotation right ROR	✓	✓	✓	✓	

	Table 31 – Bitwise Boolean functions					
1	And (&)	✓	✓	✓	✓	
2	Or (>=1)	✓	✓	✓	✓	
3	Exclusive Or	✓	✓	✓	✓	
4	Not	✓	✓	✓	✓	

	Table 32 – Selection functions						
1	Move (assignment)	MOVE	✓	✓	✓	✓	
2	Binary selection	SEL	✓	✓	✓	✓	
3	Extensible maximum function	MAX	✓	✓	✓	✓	
4	Extensible minimum function	MIN	✓	✓	✓	✓	
5	Limiter	LIMIT	✓	✓	✓	✓	
6	Extensible multiplexer	MUX	✓	✓	✓	✓	

	Table 33 – Comparison functions							
1	Decreasing sequence	`GT`	`>`	✓	✓	✓	✓	
2	Monotonic sequence	`GE`	`>=`	✓	✓	✓	✓	
3	Equality	`EQ`	`=`	✓	✓	✓	✓	
4	Monotonic sequence	`LE`	`<=`	✓	✓	✓	✓	
5	Increasing sequence	`LT`	`<`	✓	✓	✓	✓	
6	Inequality	`NE`	`<>`	✓	✓	✓	✓	

	Table 34 – Selection functions						
1	String length	`LEN`	✓	✓	✓	✓	
2	Left	`LEFT`	✓	✓	✓	✓	
3	Right	`RIGHT`	✓	✓	✓	✓	
4	Middle	`MID`	✓	✓	✓	✓	
5	Extensible concatenation	`CONCAT`	✓	✓	✓	✓	
6	Insert	`INSERT`	✓	✓	✓	✓	
7	Delete	`DELETE`	✓	✓	✓	✓	
8	Replace	`REPLACE`	✓	✓	✓	✓	
9	Find	`FIND`	✓	✓	✓	✓	

	Table 35 – Numerical functions of time and duration data types					
1a	`ADD`				✓	
1b	`ADD_TIME`	✓	✓			
1c	`ADD_LTIME`	✓	✓			
2a	`ADD`				✓	
2b	`ADD_TOD_TIME`	✓	✓			
2c	`ADD_LTOD_LTIME`	✓	✓			
3a	`ADD`				✓	
3b	`ADD_DT_TIME`	✓	✓			

	Table 35 – Numerical functions of time and duration data types					
3c	ADD_LDT_LTIME	✓	✓			
4a	SUB				✓	
4b	SUB_TIME	✓	✓			
4c	SUB_LTIME	✓	✓			
5a	SUB				✓	
5b	SUB_DATE_DATE	✓	✓			
5c	SUB_LDATE_LDATE	✓	✓			
6a	SUB				✓	
6b	SUB_TOD_TIME	✓	✓			
6c	SUB_LTOD_LTIME	✓	✓			
7a	SUB				✓	
7b	SUB_TOD_TOD	✓	✓			
7c	SUB_TOD_TOD	✓	✓			
8a	SUB				✓	
8b	SUB_DT_TIME	✓	✓			
8c	SUB_LDT_LTIME	✓	✓			
9a	SUB				✓	
9b	SUB_DT_DT					
9c	SUB_LDT_LDT	✓	✓			
10a	MUL				✓	
10b	MUL_TIME					
10c	MUL_LTIME					
11a	DIV				✓	
11b	DIV_TIME					
11c	DIV_LTIME					

	Table 36 – Additional functions of time data types CONCAT and SPLIT					
1a	CONCAT_DATE_TOD	✓	✓		✓	
1b	CONCAT_DATE_LTOD	✓	✓		✓	
2	CONCAT_DATE					
3a	CONCAT_TOD					
3b	CONCAT_LTOD					
4a	CONCAT_DT					
4b	CONCAT_LDT					

	Table 36 – Additional functions of time data types CONCAT and SPLIT					
	Split time data types					
5	`SPLIT_DATE`					
6a	`SPLIT_TOD`					
6b	`SPLIT_LTOD`					
7a	`SPLIT_DT`					
7b	`SPLIT_LDT`					
	Get day of the week					
8	`DAY_OF_WEEK`					

	Table 37 – Function for endianess conversion						
1	`TO_BIG_ENDIAN`	`TO_BIG_ENDIAN`					
2	`TO_LITTLE_ENDIAN`	`TO_LITTLE_ENDIAN`					
3	`BIG_ENDIAN_TO`	`FROM_BIG_ENDIAN`					
4	`LITTLE_ENDIAN_TO`	`FROM_LITTLE_ENDIAN`					

	Table 38 – Functions of enumerated data types					
1	`SEL`					
2	`MUX`					
3	`EQ`					
4	`NE`					

	Table 39 – Validate functions					
1	`IS_VALID`					
2	`IS_VALID_BCD`					

	Table 40 – Function block type declaration					

1	Declaration of function block type
	`FUNCTION_BLOCK ...`
	`END_FUNCTION_BLOCK`
		✓	✓	✓	✓	

2a	Declaration of inputs `VAR_INPUT ... END_VAR`	✓	✓	✓	✓	
2b	Declaration of outputs `VAR_OUTPUT ... END_VAR`	✓	✓	✓	✓	
2c	Declaration of in-outs `VAR_IN_OUT ... END_VAR`	✓	✓	✓	✓	
2d	Declaration of temporary variables `VAR_TEMP ... END_VAR`	✓	✓	✓	✓	
2e	Declaration of static variables `VAR ... END_VAR`	✓	✓	✓	✓	
2f	Declaration of external variables `VAR_EXTERNAL ... END_VAR`					

	Table 40 – Function block type declaration					
2g	Declaration of external variables `VAR_EXTERNAL CONSTANT ... END_VAR`					
3a	Initialization of inputs	✓	✓	✓	✓	
3b	Initialization of outputs	✓	✓	✓	✓	
3c	Initialization of static variables	✓	✓	✓	✓	
3d	Initialization of temporary variables					
-	`EN/ENO` inputs and outputs					See table 18
4a	Declaration of `RETAIN` qualifier on input variables	✓	✓	✓	✓	
4b	Declaration of `RETAIN` qualifier on output variables	✓	✓	✓	✓	
4c	Declaration of `NON_RETAIN` qualifier on input variables	✓	✓	✓	✓	
4d	Declaration of `NON_RETAIN` qualifier on output variables	✓	✓	✓	✓	
4e	Declaration of `RETAIN` qualifier on static variables	✓	✓	✓	✓	
4f	Declaration of `NON_RETAIN` qualifier on static variables	✓	✓	✓	✓	
5a	Declaration of `RETAIN` qualifier on local FB instances					
5b	Declaration of `NON_RETAIN` qualifier on local FB instances					
6a	Textual declaration of - rising edge inputs					
6b	- falling edge inputs (textual)					
7a	Graphical declaration of - rising edge inputs (>)					
7b	Graphical declaration of - falling edge inputs (<)					

	Table 41 – Function block instance declaration					
1	Declaration of FB instance(s)	✓	✓	✓	✓	
2	Declaration of FB instance with initialization of its variables					

	Table 42 – Function block call					
1	Complete formal call (textual only) Is used if EN/ENO is necessary in calls.			✓	✓	
2	Incomplete formal call (textual only)			✓	✓	
3	Graphical call	✓	✓			
4	Graphical call with negated boolean input and output	✓	✓			
5a	Graphical call with usage of `VAR_IN_OUT`					

	Table 42 – Function block call					
5b	Graphical call with assignment of `VAR_IN_OUT` to a variable					
6a	Textual Call with separate assignment of input `FB_Instance.Input := x;`			✓	✓	
6b	Graphical call separate assignment of input	✓	✓			
7	Textual Output read after FB call `x:= FB_Instance.Output;`			✓	✓	
8a	Textual output assigned in FB call			✓	✓	
8b	Textual output assigned in FB call with negation					
9a	Textual call with function block instance name as input					
9b	Graphical call with function block instance name as input					
10a	Textual call with function block instance name as `VAR_IN_OUT`					
10b	Graphical call with function block instance name as `VAR_IN_OUT`					
11a	Textual call with function block instance name as external variable					
11b	Graphical call with function block instance name as external variable					

	Table 43 – Standard bistable function blocks					
1a	Bistable function block (set dominant): `SR(S1,R,Q1)`					

	`       +-----+`
	`       | SR  |`
	`BOOL---|S1 Q1|---BOOL`
	`BOOL---|R    |`
	`       +-----+`
		✓	✓	✓	✓	

1b	Bistable function block (set dominant) with long input names: `SR(SET1, RESET, Q1)`					

	`       +--------+`
	`       | SR     |`
	`BOOL---| SET1 Q1|---BOOL`
	`BOOL---|RESET   |`
	`       +--------+`					

2a	Bistable function block (reset dominant): `RS(S, R1, Q1)`					

	`       +-----+`
	`       | RS  |`
	`BOOL---|S  Q1|---BOOL`
	`BOOL---|R1   |`
	`       +-----+`
		✓	✓	✓	✓	

	Table 43 – Standard bistable function blocks					
2b	Bistable function block (reset dominant) with long input names: `RS(SET,RESET1, Q1)`					

	`        +---------+`
	`        | RS      |`
	`BOOL---|SET    Q1|---BOOL`
	`BOOL---|R1       |`
	`        +---------+`					

	Table 44 – Standard edge detection function blocks					
1	Rising edge detector: `R_TRIG(CLK, Q)`					

	`        +---------+`
	`        | R_TRIG |`
	`BOOL --|CLK     Q|-- BOOL`
	`        +---------+`
		✓	✓		✓	

2	Falling edge detector: `F_TRIG(CLK, Q)`					

	`        +---------+`
	`        | F_TRIG |`
	`BOOL --|CLK     Q|-- BOOL`
	`        +---------+`
		✓	✓		✓	

	Table 45 – Standard counter function blocks					
	Up-Counter					
1a	`CTU_INT(CU, R, PV, Q, CV)` or `CTU(..)`	✓	✓	✓	✓	

	`       +-------+`
	`       | CTU   |`
	`BOOL--->CU    Q|---BOOL`
	`BOOL---|R      |`
	` INT---|PV   CV|---INT`
	`       +-------+`
	and also:
	`       +-------------+`
	`       | CTU_INT     |`
	`BOOL--->CU          Q|---BOOL`
	`BOOL---|R            |`
	` INT---|PV         CV|---INT`
	`       +-------------+`					

1b	`CTU_DINT PV, CV: DINT`	✓	✓	✓	✓	
1c	`CTU_LINT PV, CV: LINT`	✓	✓	✓	✓	
1d	`CTU_UDINT PV, CV: UDINT`	✓	✓	✓	✓	
1e	`CTU_ULINT(CD, LD, PV, CV) PV, CV: ULINT`	✓	✓	✓	✓	

	Table 45 – Standard counter function blocks					
	Down-counters					
2a	`CTD_INT(CD, LD, PV, Q, CV) or CTD`	✓	✓	✓	✓	

```
        +-----+
        | CTD |
 BOOL--->CD  Q|---BOOL
 BOOL---|LD   |
  INT---|PV CV|---INT
        +-----+
```

and also:

```
        +-----------+
        | CTD_INT   |
 BOOL--->CD        Q|---BOOL
 BOOL---|LD         |
  INT---|PV       CV|---INT
        +-----------+
```

2b	`CTD_DINT PV, CV: DINT`	✓	✓	✓	✓	
2c	`CTD_LINT PV, CV: LINT`	✓	✓	✓	✓	
2d	`CTD_UDINT PV, CV: UDINT`	✓	✓	✓	✓	
2e	`CTD_ULINT PV, CV: UDINT`	✓	✓	✓	✓	
	Up-down counters					
3a	`CTUD_INT(CD, LD, PV, Q, CV) or CTUD(..)`	✓	✓	✓	✓	

```
        +-----------+
        | CTUD      |
 BOOL--->CU       QU|---BOOL
 BOOL--->CD       QD|---BOOL
 BOOL---|R          |
 BOOL---|LD         |
  INT---|PV       CV|---INT
        +-----------+
```

and also:

```
        +-----------+
        | CTUD_INT  |
 BOOL--->CU       QU|---BOOL
 BOOL--->CD       QD|---BOOL
 BOOL---|R          |
 BOOL---|LD         |
  INT---|PV       CV|---INT
        +-----------+
```

3b	`CTUD_DINT PV, CV: DINT`	✓	✓	✓	✓	
3c	`CTUD_LINT PV, CV: LINT`	✓	✓	✓	✓	
3d	`CTUD_UDINT PV, CV: UDINT`	✓	✓	✓	✓	
3e	`CTUD_ULINT PV, CV: ULINT`	✓	✓	✓	✓	

	Table 46 – Standard timer function blocks					
1a	Pulse, overloaded `TP`					
1b	Pulse using TIME	✓	✓	✓	✓	
1c	Pulse using LTIME	✓	✓	✓	✓	
2a	On-delay, overloaded `TON`					
2b	On-delay using TIME	✓	✓	✓	✓	
2c	On-delay using LTIME	✓	✓	✓	✓	
2d	On-delay, overloaded (Graphical)					
3a	Off-delay, overloaded `TOF`					
3b	Off-delay using TIME	✓	✓	✓	✓	
3c	Off-delay using LTIME	✓	✓	✓	✓	
3d	Off-delay, overloaded (Graphical)					

	Table 47 – Program declaration					
1	Declaration of a program `PROGRAM ... END PROGRAM`					
2a	Declaration of inputs `VAR INPUT ... END VAR`					
2b	Declaration of outputs `VAR OUTPUT ... END VAR`					
2c	Declaration of in-outs `VAR IN OUT ... END VAR`					
2d	Declaration of temporary variables `VAR TEMP ... END VAR`					
2e	Declaration of static variables `VAR ... END VAR`					
2f	Declaration of external variables `VAR EXTERNAL ... END VAR`					
2g	Declaration of external variables `VAR EXTERNAL CONSTANT ... END VAR`					
3a	Initialization of inputs					
3b	Initialization of outputs					
3c	Initialization of static variables					
3d	Initialization of temporary variables					
4a	Declaration of `RETAIN` qualifier on input variables					
4b	Declaration of `RETAIN` qualifier on output variables					
4c	Declaration of `NON_RETAIN` qualifier on input variables					
4d	Declaration of `NON_RETAIN` qualifier on output variables					
4e	Declaration of `RETAIN` qualifier on static variables					
4f	Declaration of `NON_RETAIN` qualifier on static variables					

	Table 47 – Program declaration					
5a	Declaration of `RETAIN` qualifier on local FB instances					
5b	Declaration of `NON_RETAIN` qualifier on local FB instances					
6a	Textual declaration of - rising edge inputs					
6b	Textual declaration of - falling edge inputs (textual)					
7a	Graphical declaration of - rising edge inputs (>)					
7b	Graphical declaration of - falling edge inputs (<)					
8a	`VAR_GLOBAL...END_VAR` declaration within a `PROGRAM`					
8b	`VAR_GLOBAL CONSTANT` declarations within `PROGRAM` type declarations					
9	`VAR_ACCESS...END_VAR` declaration within a `PROGRAM`					

	Table 48 – Class					
1	`CLASS ... END_CLASS`					
1a	`FINAL` specifier					
	Adapted from function block					
2a	Declaration of variables `VAR ... END_VAR`					
2b	Initialization of variables					
3a	`RETAIN` qualifier on internal variables					
3b	`NON_RETAIN` qualifier on internal variables					
4a	`VAR_EXTERNAL` declarations within class type declarations					
4b	`VAR_EXTERNAL CONSTANT` declarations within class type declarations					
	Methods and specifiers					
5	`METHOD...END_METHOD`					
5a	`PUBLIC` specifier					
5b	`PRIVATE` specifier					
5c	`INTERNAL` specifier					
5d	`PROTECTED` specifier					
5e	`FINAL` specifier					

	Table 48 – Class					
	Inheritance					
6	`EXTENDS`					
7	`OVERRIDE`					
8	`ABSTRACT`					
	Access reference					
9a	`THIS`					
9b	`SUPER`					
	Variable access specifiers					
10a	`PUBLIC` specifier					
10b	`PRIVATE` specifier					
10c	`INTERNAL` specifier					
10d	`PROTECTED` specifier					
	Polymorphism					
11a	with VAR_IN_OUT					
11b	with reference					

	Table 49 – Class instance declaration					
1	Declaration of class instance(s) with default initialization					
2	Declaration of class instance with initialization of its public variables					

	Table 50 – Textual call of methods – Formal and non-formal parameter list					
1a	Complete formal call (textual only) Shall be used if `EN/ENO` is necessary in calls.					
1b	Incomplete formal call (textual only) Shall be used if EN/ENO is not necessary in calls.					
2	Non-formal call (textual only) (fix order and complete)					

	Table 51 – Interface					
1	`INTERFACE ... END_INTERFACE`					
	Methods and specifiers					
2	`METHOD...END_METHOD`					
	Inheritance					
3	`EXTENDS`					
	Usage of interface					
4a	`IMPLEMENTS` interface					
4b	`IMPLEMENTS multi-interfaces`					
4c	Interface as type of a variable					

	Table 52 – Assignment Attempt					
1	Assignment attempt with interfaces using `?=`					
2	Assignment attempt with references using `?=`					

	Table 53 – Object oriented function block					
1	Object oriented Function block					
1a	`FINAL` specifier					
	Methods and specifiers					
5	`METHOD...END_METHOD`					
5a	`PUBLIC` specifier					
5b	`PRIVATE` specifier					
5c	`INTERNAL` specifier					
5d	`PROTECTED` specifier					
5e	`FINAL` specifier					
	Usage of interface					
6a	`IMPLEMENTS` interface					
6b	`IMPLEMENTS multi-interfaces`					
6c	Interface as type of a variable					
	Inheritance					
7a	`EXTENDS`					
7b	`EXTENDS`					
8	`OVERRIDE`					
9	`ABSTRACT`					
	Access reference					
10a	`THIS`					
10b	`SUPER`					
10c	`SUPER()`					

	Table 53 – Object oriented function block					
	Variable access specifiers					
11a	`PUBLIC` specifier					
11b	`PRIVATE` specifier					
11c	`INTERNAL` specifier					
11d	`PROTECTED` specifier					
	Polymorphism					
12a	with `VAR_IN_OUT` with equal signature					
12b	With `VAR_IN_OUT` with compatible signature					
12c	with reference with equal signature					
12d	with reference with compatible signature					

	Table 54 – SFC step					**Valid for SFC (Graph)**
1a	Step – graphical form with directed links					✓
1b	Initial step – graphical form with directed link					✓
2a	Step – textual form without directed links					
2a	Initial step – textual form without directed links					
3a	Step flag – general form `***.X = BOOL#1` when `***` is active, `BOOL#0` otherwise					✓
3b	Step flag – direct connection of Boolean variable `***.X` to right side of step					✓
4	Step elapsed time – general form `***.T =` a variable of type `TIME`					✓

	Table 55 – SFC transition and transition condition					**Valid for SFC (Graph)**
1	Transition condition physically or logically adjacent to the transition using ST language					
2	Transition condition physically or logically adjacent to the transition using LD language					✓
3	Transition condition physically or logically adjacent to the transition using FBD language					✓
4	Use of connector					
5	Transition condition: Using LD language					
6	Transition condition: Using FBD language					
7	Textual equivalent of feature 1 using ST language					
8	Textual equivalent of feature 1 using IL language					
9	Use of transition name					✓
10	Transition condition using LD language					

	Table 55 – SFC transition and transition condition					**Valid for SFC (Graph)**
11	Transition condition using FBD language					
12	Transition condition using IL language					
13	Transition condition using ST language					

	Table 56 – SFC declaration of actions					**Valid for SFC (Graph)**
1	Any Boolean variable declared in a `VAR` or `VAR_OUTPUT` block, or their graphical equivalents, can be an action.					✓
2l	Graphical declaration in LD language					
2s	Inclusion of SFC elements in action					
2f	Graphical declaration in FBD language					
3s	Textual declaration in ST language					
3i	Textual declaration in IL language					

	Table 57 – Step/action association					**Valid for SFC (Graph)**
1	Action block physically or logically adjacent to the step					✓
2	Concatenated action blocks physically or logically adja-cent to the step					✓
3	Textual step body					
4	Action block "d" field					

	Table 58 – Action block					**Valid for SFC (Graph)**

```
    +-----+---------------+-----+
 ---| "a" |      "b"      | "c" |---
    +-----+---------------+-----+
    |            "d"            |
    |                           |
    +---------------------------+
```

1	`"a"` : Qualifier as per 6.7.4.5					
2	`"b"` : Action name					
3	`"c"` : Boolean "indicator" variables (**deprecated**)					
	"d" : Action using:					
4i	IL language					
4s	ST language					
4l	LD language					
4f	FBD language					
5l	Use of action blocks LD					
5f	Use of action blocks in FBD					

	Table 59 – Action qualifiers						**Valid for SFC (Graph)**
1	Non-stored (null qualifier)	None					
2	**N**on-stored	`N`					✓
3	overriding **R**eset	`R`					✓
4	**S**et (**S**tored)	`S`					✓
5	time **L**imited	`L`					✓
6	time **D**elayed	`D`					✓
7	**P**ulse	`P`					
8	**S**tored and time **D**elayed	`SD`					
9	**D**elayed and **S**tored	`DS`					
10	**S**tored and time **L**imited	`SL`					
11	**P**ulse (rising edge)	`P1`					
12	**P**ulse (falling edge)	`P0`					

	Table 60 – Action control features					
1	With final scan					✓
2	Without final scan					

	Table 61 – Sequence evolution – graphical					**Valid for SFC (Graph)**
1	Single sequence					✓
2a	Divergence of sequence with left to right priority					✓
2b	Divergence of sequence with numbered branches					
2c	Divergence of sequence with mutual exclusion					
3	Convergence of sequence					✓
4a	Simultaneous divergence after a single transition					✓
4b	Simultaneous divergence after conversion					✓
4c	Simultaneous convergence before one transition					✓
4d	Simultaneous convergence before a sequence selection					✓
5a,b,c	Sequence skip					✓
6a, b, c	Sequence loop					✓
7	Directional arrows					✓

	Table 62 – Configuration and resource declaration					
1	CONFIGURATION...END_CONFIGURATION					VAR_GLOBAL >> Definition als PLCVariable
2	VAR_GLOBAL...END_VAR within CONFIGURATION					
3	RESOURCE...ON ...END_RESOURCE					
4	VAR_GLOBAL...END_VAR within RESOURCE					
5a	Periodic TASK					Tasks are provided in form of organization blocks (OBs) in STEP 7
5b	Non-periodic TASK					
6a	WITH for PROGRAM to TASK association					
6b	WITH for FUNCTION_BLOCK to TASK association					
6c	PROGRAM with no TASK association					
7	Directly represented variables in VAR_GLOBAL					
8a	Connection of directly represented variables to PROGRAM inputs					
8b	Connection of GLOBAL variables to PROGRAM inputs					
9a	Connection of PROGRAM outputs to directly represented variables					
9b	Connection of PROGRAM outputs to GLOBAL variables					
10a	VAR_ACCESS...END_VAR					
10b	Access paths to directly represented variables					
10c	Access paths to PROGRAM inputs					
10d	Access paths to GLOBAL variables in RESOURCEs					
10e	Access paths to GLOBAL variables in CONFIGURATIONs					
10f	Access paths to PROGRAM outputs					
10g	Access paths to PROGRAM internal variables					
10h	Access paths to function block inputs					
10i	Access paths to function block outputs					
11a	VAR_CONFIG...END_VAR to variables. This feature shall be supported if the feature "partly defined" with "*" in Table 16 is supported.					
11b	VAR_CONFIG...END_VAR to components of structures					
12a	VAR_GLOBAL CONSTANT in RESOURCE					
12b	VAR_GLOBAL CONSTANT in CONFIGURATION					
13a	VAR_EXTERNAL in RESOURCE					
13b	VAR_EXTERNAL CONSTANT in RESOURCE					

	Table 63 – Task					
1a	Textual declaration of periodic TASK					
1b	Textual declaration of non-periodic TASK					
	Graphical representation of TASKs (general form)					Tasks are provided in form of organization blocks (OBs) in STEP 7
2a	Graphical representation of periodic TASKs (with INTERVAL)					
2b	Graphical representation of non-periodic TASK (with SINGLE)					
3a	Textual association with PROGRAMs					
3b	Textual association with function blocks					
4a	Graphical association with PROGRAMs					
4b	Graphical association with function blocks within PROGRAMs					
5a	Non-preemptive scheduling					
5b	Preemptive scheduling					

	Table 64 – Namespace					
1a	Public namespace (without access specifier)					
1b	Internal namespace (with INTERNAL specifier)					
2	Nested namespaces					
3	Variable access specifier INTERNAL					
4	Method access specifier INTERNAL					
5	Language element with access specifier INTERNAL: • User-defined data types - using keyword TYPE • Functions • Function block types • Classes • Interfaces					

	Table 65 – Nested namespace declaration options					
1	Lexically nested namespace declaration Equivalent to feature 2 of Table 64					
2	Nested namespace declaration by fully qualified name					
3	Mixed lexically nested namespace and namespace nested by fully qualified name					

	Table 66 – Namespace directive USING					
1	`USING` in global namespace					
2	`USING` in other namespace					
3	`USING` in POUs • Functions • Function block types • Classes • Methods • Interfaces					

	Table 67 – Parenthesized expression for IL language					
1	Parenthesized expression beginning with explicit opera-tor:					
2	Parenthesized expression (short form)					

	Table 68 – Instruction list operators						
1	`LD`	`N`					
2	`ST`	`N`					
3	`S , R`						
4	`AND`	`N, (`					
5	`&`	`N, (`					
6	`OR`	`N, (`					
7	`XOR`	`N, (`					
8	`NOT`						
9	`ADD`	`(`					
10	`SUB`	`(`					
11	`MUL`	`(`					
12	`DIV`	`(`					
13	`MOD`	`(`					
14	`GT`	`(`					
15	`GE`	`(`					
16	`EQ`	`(`					
17	`NE`	`(`					
18	`LE`	`(`					
9	`LT`	`(`					
20	`JMP`	`C, N`					
21	`CAL`	`C, N`					
22	`RET`	`C, N`					
23	`)`						
24	`ST?`						

	Table 69 – Calls for IL language					
1a	Function block call with non-formal parameter list					
1b	Function block call with formal parameter list					
2	Function block call with load/store of standard input parameters					
3a	Function call with formal parameter list					
3b	Function call with non-formal parameter list					
4a	Method call with formal parameter list					
4b	Method call with non-formal parameter list					

	Table 70 – Standard function block operators for IL language							
1	SR	S1,R	Q					
2	RS	S,R1	Q					
3	F/R_TRIG	CLK	Q					
4	CTU	CU,R,PV	CV,Q, also RESET					
5	CTD	CD,PV	CV,Q					
6	CTUD	CU,CD,R,PV	CV,QU,QD, also RESET					
7	TP	IN,PT	CV,Q					
8	TON	IN,PT	CV,Q					
9	TOF	IN,PT	CV,Q					

	Table 71 – Operators of the ST language						
1	Parentheses	(expression)				✓	
2	Evaluation of result of function and method – if a result is declared	Identifier (parameter list)				✓	
3	Dereference	^					
4	Negation	-				✓	
5	Unary Plus	+				✓	
5	Complement	NOT				✓	
7	Exponentiationb	**				✓	
8	Multiply	*				✓	
9	Divide	/				✓	
10	Modulo	MOD				✓	
11	Add	+				✓	
12	Subtract	-				✓	
13	Comparison	< , > , <= , >=				✓	
14	Equality	=				✓	
15	Inequality	<>				✓	

	Table 71 – Operators of the ST language						
16a	Boolean `AND`	`&`				✓	
16b	Boolean `AND`	`AND`				✓	
17	Boolean Exclusive `OR`	`XOR`				✓	
18	Boolean `OR`	`OR`				✓	

	Table 72 – ST language statements					
	Assignment					
1	Variable := expression;				✓	
1a	Variable and expression of elementary data type				✓	
1b	Variables and expression of different elementary data types with implicit type conversion according Figure 11				✓	
1c	Variable and expression of user-defined type				✓	
1d	Instances of function block type					
	Call					
2a	Function call				✓	
2b	Function block call and FB output usage				✓	
2c	Method call					
3	`RETURN`				✓	
	Selection					

4	`IF ...`
	`THEN ...`
	` ELSIF ... THEN ...`
	`ELSE ...END IF`
					✓	

5	`CASE ... OF`
	` ...`
	` ELSE ...`
	`END CASE`
					✓	

	Iteration					

6	`FOR ... TO ... BY ... DO`
	` ... END FOR`
					✓	

7	`WHILE ... DO`
	`...`
	`END WHILE`
					✓	

8	`REPEAT ...`
	` UNTIL ...`
	`END REPEAT`
					✓	

9	`CONTINUE`				✓	
10	`EXIT` an iteration				✓	
11	Empty Statement				✓	

	Table 73 – Graphic execution control elements						
	Unconditional jump						
1a	FBD language	`1---->>LABELA`		✓			

1b	LD language	`|`
		`+---->>LABELA |`
			✓				

	Conditional jump						

2a	FBD language	Example:
		jump condition, jump target
		`X---->>LABELB`
		`        +---+`
		`bvar0---| & |--->>NEXT`
		`bvar50--|   |`
		`        +---+`
		`NEXT:`
		`        +---+`
		`bvar5---|>=1|---bOut0`
		`bvar60--|   |`
		`        +---+`
				✓			

2b	LD language	Example:
		jump condition, jump target
		`|  X`
		`+-| |---->>LABELB`
		`|`
		`|`
		`|   bvar0   bvar50`
		`+---| |----| |--->>NEXT`
		`|`
		`|`
		`NEXT:`
		`|   bvar5      bOut0  |`
		`+---| |----+----( )---+`
		`|   bvar60 |           |`
		`+---| |----+           |`
		`|                       |`
			✓				

	Conditional return						

3a	LD language	`|  X`
		`+--| |---<RETURN>`
		`|`
			✓				

3b	FBD language	`X---<RETURN>`		✓			

	Table 73 – Graphic execution control elements						
	Unconditional return						
4	LD language	| +---<RETURN> |	✓				

	Table 74 – Power rails and link elements						
1	Left power rail (with attached horizontal link)	| +--- |	✓				
2	Right power rail (with attached horizontal link)	| ---+ |					
3	Horizontal link	-----------	✓				
4	Vertical link (with attached horizontal links)	| ----+---- ----+ | +----	✓				

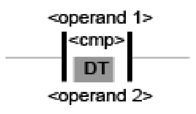
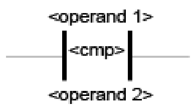
	Table 75 – Contacts						
	Static contacts						
1	Normally open contact	*** --| |--	✓				
2	Normally closed contact	*** --|/|--	✓				
	Transition-sensing contacts						
3	Positive transition-sensing contact	*** --|P|--	✓				
4	Negative transition-sensing contact	*** --|N|--	✓				
5a	Compare contact (typed)		✓				
5b	Compare contact, (over-loaded)		✓				

	Table 76 – Coils						
1	Coil	`*** --( )—`	✓				
2	Negated coil	`*** --(/)—`	✓				
	Latched coils						
3	Set (latch) coil	`*** --(S)—`	✓				
4	Reset (unlatch) coil	`*** --(R)—`	✓				
	Transition-sensing coils						
8	Positive transition-sensing coil	`*** --(P)—`	✓				
9	Negative transition-sensing coil	`*** --(N)—`	✓				