Reference manual

PL7Micro/Junior/Pro

Appendices

TLX DR PL7 40E eng V4.0

35002664 00

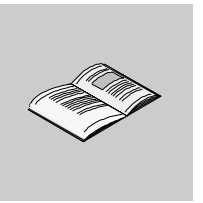
Related Documentation

At a Glance

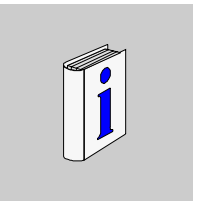
This manual is made up of three volumes:

- Volume 1: Description of the PL7 software
  - General
  - Ladder Language
  - Language Instruction List
  - Structured text language
  - Grafcet language
  - DFB function blocks
  - Functional modules
- Volume 2: Detailed description of the instructions and functions
  - Basic instructions
  - Advanced instructions
  - Bit objects and system words
- Volume 3: Appendices
  - Differences between PL7-2/3 and PL7-Micro/Junior
  - Checklist
  - List of reserved words
  - In conformance with the IEC standard 1131-3
  - OLE server automation
  - Performance characteristics

Table of Contents



About the book



At a Glance

Document Scope

This manual gives additional information for programming Micro, Premium and Atrium PLCs.

Validity Note

This publication has been updated to incorporate the PL7 V4.0 functions; however previous versions of PL7can still be implemented.

Revision History

Rev. No.	Changes
1	First version

Related Documents

Title of Documentation	Reference Number

Product Related Warnings

Contents

User Comments

We welcome your comments about this document. You can reach us by e-mail at TECHCOMM@modicon.com

Differences between PL7-2/3 and PL7 Micro/Junior

1

Introduction

Contents of this section

This section describes the differences in objects and instructions between PL7-2/ PL7-3 software and PL7 Micro/PL7 Junior software

What's in this Chapter?

This Chapter contains the following Maps:

Immediate values and labels

Immediate values

Table showing differences between PL7-2/3 and PL7-Micro/Junior

Objects	PL7-2/3	PL7 Micro/Junior
Base 10 integer	1234	1234
Base 2 integer	L'10011110'	2#10011110
Base 16 integer	H'ABCD'	16#ABCD
Floating point	-1.32e12 (PL7-3)	-1.32e12
Character string	M'aAbBcC'	'aAbBcC'

Labels

Table showing differences between PL7-2/3 and PL7-Micro/Junior

Objects	PL7-2/3	PL7 Micro/Junior
Label	Li i = 0 to 999	%Li i = 0 to 999

Bits

Input bits in rack Table showing differences between PL7-2/3 and PL7-Micro/Junior

Objects	PL7-2/3	PL7 Micro/Junior
Input bit in rack	`Ixy,i`	`%Ixy.i`
Input bit in an indexed rack	`Ixy,i(Wj)` (PL7-3)	`%Ixy.i[%MWj]`
Remote input bit	`RIx,y,i` (PL7-3)	`%I\chemin\mod.voie`
Indexed remote input bit	`RIx,y,i(Wj)` (PL7-3)	-

Output Bits in rack Table showing differences between PL7-2/3 and PL7-Micro/Junior

Objects	PL7-2/3	PL7 Micro/Junior
Output Bit in rack	`Oxy,i`	`%Qxy.i`
Output Bit in indexed rack	`Oxy,i(Wj)` (PL7-3)	`%Qxy.i[%MWj]`
Remote output bit	`ROx,y,i` (PL7-3)	`%Q\chemin\mod.voie`
Indexed remote output bit	`ROx,y,i(Wj)` (PL7-3)	-

I/O error bits in rack Table showing differences between PL7-2/3 and PL7-Micro/Junior

Objects	PL7-2/3	PL7 Micro/Junior
Module error bit	`Ixy,S` / `Oxy,S`	`%Ixy.MOD.ERR`
Channel bit	-	`%Ixy.i.ERR`

Remote I/O error bit Table showing differences between PL7-3 and PL7-Micro/Junior

Objects	PL7-3 (only)	PL7 Micro/Junior
Module error bit	-	`%I\chemin\mod.MOD.ERR`
Channel bit	`RDx,y,i/ERRORx,y,i`	`%I\chemin\mod.voie.ERR`
output channel tripped bit	`TRIPx,y,i`	-
output channel reactivation bit	`RSTx,y,i`	-

Internal bits and system bits

Table showing differences between PL7-2/3 and PL7-Micro/Junior

Objects	PL7-2/3	PL7 Micro/Junior
Internal bit	`Bi`	`%Mi`
Indexed internal bit	`Bi(Wj)` (PL7-3)	`%Mi[%MWj]`
System bit	`SYi`	`%Si`

Step bits

Table showing differences between PL7-2/3 and PL7-Micro/Junior

Objects	PL7-2/3	PL7 Micro/Junior
Step bit	`Xi`	`%Xi`
Macro-step bit	`XMj` (PL7-3)	`%XMj`
Step i bit of macro-step j	`Xj,i` (PL7-3)	`%Xj.i`
Input step bit of macro-step j	`Xj,I` (PL7-3)	`%Xj.IN`
Output step bit of macro-step j	`Xj,O` (PL7-3)	`%Xj.OUT`

Word bits

Table showing differences between PL7-2/3 and PL7-Micro/Junior

Objects	PL7-2/3	PL7 Micro/Junior
Bit j of internal word i	`Wi,j`	`%MWi:Xj`
Bit j of indexed internal word i	`Wi(Wk),j` (PL7-3)	`%MWi[%MWk]:Xj`
Bit j of constant word i	`CWi,j`	`%KWi:Xj`
Bit j of indexed constant word i	`CWi(Wk),j` (PL7-3)	`%KWi[%MWk]:Xj`
Bit j of register i	`IW/OWxy,i,j`	`%IW/%QWxy.i:Xj`
Bit k of common word j of station i	`COMi,j,k` `COMXi,j,k` (X = B, C, D)	`%NWi.j:Xk` `%NXWi.j:Xk`
Bit j of system word i	`SWi,j`	`%SWi:Xj`

Words

Internal words Table showing differences between PL7-2/3 and PL7-Micro/Junior

Objects	PL7-2/3	PL7 Micro/Junior
Single length internal word	`Wi`	`%MWi`
Indexed single length internal word	`Wi(Wj)` (PL7-3)	`%MWi[%MWj]`
Double length internal word	`DWi` (PL7-3)	`%MDi`
Indexed double length internal word	`DWi(Wj)` (PL7-3)	`%MDi[%MWj]`
Real internal word	-	`%MFi`
Indexed real internal word	-	`%MFi[%MWj]`

Constant words Table showing differences between PL7-2/3 and PL7-Micro/Junior

Objects	PL7-2/3	PL7 Micro/Junior
Single length constant word	`CWi`	`%KWi`
Indexed single length constant word	`CWi(Wj)`	`%KWi[%MWj]`
Double length constant word	`CDWi` (PL7-3)	`%KDi`
Indexed double length constant word	`CDWi(Wj)` (PL7-3)	`%KDi[%MWj]`
Real constant word	-	`%KFi`
Indexed real constant word	-	`%KFi[%MWj]`

Register words Table showing differences between PL7-2/3 and PL7-Micro/Junior

Objects	PL7-2/3	PL7 Micro/Junior
Single length input register word	`IWxy,i`	`%IWxy.i`
Double length input register word	-	`%IDxy.i`
Single length output register word	`OWxy,i`	`%QWxy.i`
Double length output register word	-	`%QDxy.i`
Remote input register word	`RIWx,y,i` (PL7-3)	`%IW\chemin\mod.voie`
Remote output register word	`ROWx,y,i` (PL7-3)	`%QW\chemin\mod.voie`

Other words Table showing differences between PL7-3 and PL7-Micro/Junior

Objects	PL7-3 (only)	PL7 Micro/Junior
System word	SWi	%SWi
Common word j of station	COMi,j COMXi,j (where X=B,C,D)	%NW{i}j %NW{[r.]i}j r= network number
Status word from a remote discrete module	STATUSAx,y,i (PL7-3) STATUSBx,y,i (PL7-3)	-
Status word from a remote discrete module channel	STSx,y,i (PL7-3)	%IW\chemin\mod.voie.ERR

Function blocks

Timer

Table showing differences between PL7-2/3 and PL7-Micro/Junior

Objects	PL7-2/3	PL7 Micro/Junior
Address	Ti	%Ti
Preset value (word)	TI,P	%Ti.P
Current value (word)	Ti,V	%Ti.V
Timer running (bit)	Ti,R	%Ti.R
Timer elapsed (bit)	Ti,D	%Ti.D

Monostable

Table showing differences between PL7-2/3 and PL7-Micro/Junior

Objects	PL7-2/3	PL7 Micro/Junior
Address	Mi	%MNi
Preset value (word)	Mi,P	%MNi.P
Current value (word)	Mi,V	%MNi.V
Monostable running (bit)	Mi,R	%MNi.R

Up/Down Counter

Table showing differences between PL7-2/3 and PL7-Micro/Junior

Objects	PL7-2/3	PL7 Micro/Junior
Address	Ci	%Ci
Preset value (word)	Ci,P	%Ci.P
Current value (word)	Ci,V	%Ci.V
Counting overrun (bit)	Ci,E	%Ci.E
Preset achieved (bit)	Ci,D	%Ci.D
Down counting overrun (bit)	Ci,F	%Ci.F

Register Table showing differences between PL7-2/3 and PL7-Micro/Junior

Objects	PL7-2/3	PL7 Micro/Junior
Address	Ri	%Ri
Input word (word)	Ri,I	%Ri.I
Output word (word)	Ri,O	%Ri.O
Full register (bit)	Ri,F	%Ri.F
Empty register (bit)	Ri,E	%Ri.E

Text Block Table showing differences between PL7-2/3 and PL7-Micro/Junior

Objects	PL7-2/3	PL7 Micro/Junior
Address	TXTi	No text block

Drum Table showing differences between PL7-2 and PL7-Micro/Junior

Objects	PL7-2	PL7 Micro/Junior
Address	Di (PL7-2)	%DRi
Number of active steps (word)	Di,S	%DRi.S
activity time of the current step (word)	Di,V	%DRi.V
16 order bits (word)	Di,Wj	%DRi.Wj
Final step in progress (bit)	Di,F	%DRi.F

Fast Counter/ Timer Table showing differences between PL7-2 and PL7-Micro/Junior

Objects	PL7-2	PL7 Micro/Junior
Address	FC (PL7-2)	-
Preset value (word)	FC,P	-
Current value (word)	FC,V	-
External reset (bit)	FC,E	-
Preset achieved (bit)	FC,D	-
Counting running (bit)	FC,F	-

Real time clock Table showing differences between PL7-2 and PL7-Micro/Junior

Objects	**PL7-2**	**PL7 Micro/Junior**
Address	H (PL7-2)	-
"WEEK" or "YEAR" type day selection MTWTFSS (word)	VD	-
Starting setpoint (word)	BGN	-
End setpoint (word)	END	-
Current value < setpoint (bit)	<	-
Current value = setpoint (bit)	=	-
Current value > setpoint (bit)	>	-

Bit and word tables

Bit strings

Table showing differences between PL7-2/3 and PL7-Micro/Junior

Objects	PL7-2/3	PL7 Micro/Junior
Internal bit string	Bi[L]	%Mi:L
Input bit string	Ixy,i[L] (PL7-3)	%Ixy.i:L
Output bit string	Oxy.i[L] (PL7-3)	%Qxy.i:L
Grafcet step bit string	Xi[L] (PL7-3)	%Xi:L
Macro-step bit string	XMi[L] (PL7-3)	-

Character strings

Table showing differences between PL7-2/3 and PL7-Micro/Junior

Objects	PL7-2/3	PL7 Micro/Junior
String	-	%MBi:L (with i pair)

Word tables

Table showing differences between PL7-2/3 and PL7-Micro/Junior

Objects	PL7-2/3	PL7 Micro/Junior
Internal word table	Wi[L]	%MWi:L
Indexed internal word table	Wi(Wj)[L]	%MWi[%MWj]:L
Double internal word table	DWi[L] (PL7-3)	%MDi:L
Indexed double internal word table	DWi(Wj)[L] (PL7-3)	%MDi[%MWj]:L
Constant word table	CWi[L]	%KWi:L
Indexed constant word table	CWi(Wj)[L]	%KWi[%MWj]:L
Constant double word table	CDWi[L] (PL7-3)	%KDi:L
indexed double constant word table	CDWi(Wj)[L] (PL7-3)	%KDi[%MWj]:L
Real table	-	%MFi:L
Indexed real table	-	%MFi[%MWj]:L
Constant real table	-	%KFi:L
Indexed constant real table	-	%KFi[%MWj]:L
Remote input element table	RIx,y,i[L] (PL7-3)	-
Remote output element table	ROx,y,i[L] (PL7-3)	-
Indexed remote input element table	RIx,y,i(Wj)[L] (PL7-3)	-
Indexed remote output element table	ROx,y,i(Wj)[L] (PL7-3)	-

Optional function blocks

OFB Table showing differences between PL7-3 and PL7-Micro/Junior

Objects	PL7-3 (only)	PL7 Micro/Junior
Address	< OFB >i	-
OFB Element	< OFB >i, <element>	-
Indexed OFB Element	< OFB >i, <element>(Wj)	-
OFB element table	< OFB >i, <element>[L]	-
Indexed OFB element table	< OFB >i, <element>(Wj)[L]	-

Instructions

Bit instructions

Table showing differences between PL7-2, PL7-3 and PL7-Micro/Junior

Objects	PL7-2	PL7-3	PL7 Micro/Junior
Logical inversion	-	NOT	NOT
AND	AND	*	AND
OR	OR	+	OR
Exclusive OR	XOR	-	XOR
Rising edge	-	RE	RE
Falling edge	-	FE	FE
Set to 1	-	SET	SET
Set to 0	-	RESET	RESET

Word and double word instructions

Table showing differences between PL7-2, PL7-3 and PL7-Micro/Junior

Objects	PL7-2	PL7-3	PL7 Micro/Junior
Addition	+	+	+
Subtraction	-	-	-
Multiplication	*	*	*
Division	/	/	/
Comparison	>,>=,<,<=,=,<>	>,>=,<,<=,=,<>	>,>=,<,<=,=,<>
Remainder of a division	MOD	REM	REM
Square root	-	SQRT	SQRT
Absolute value	-	-	ABS
logical AND	AND	AND	AND
logical OR	OR	OR	OR
exclusive logical OR	XOR	XOR	XOR
Logical complement	CPL	CPL	CPL
Incrementation	-	INC	INC
Decrementation	-	DEC	DEC
Logical shift left	-	SHL	SHL
Logical shift right	-	SHR	SHR
Rotate shift left	SLC	SLC	ROL
Rotate shift right	SRC	SRC	ROR

Floating point instructions

Table showing differences between PL7-3 and PL7-Micro/Junior

Objects	PL7-3	PL7 Micro/Junior
Addition	ADDF	+
Subtraction	SUBF	-
Multiplication	MULF	*
Division	DIVF	/
Square root	SQRTF	SQRT
Absolute value	-	ABS
Equality test	EQUF	=
Strict superiority test	SUPF	>
Strict inferiority test	INFF	<
Other tests	-	>=, <=, <>

Byte string instructions

Table showing differences between PL7-3 and PL7-Micro/Junior

Objects	PL7-3	PL7 Micro/Junior
Rotate shift	SLCWORD	-

Conversion instructions

Table showing differences between PL7-2, PL7-3 and PL7-Micro/Junior

Objects	PL7-2	PL7-3	PL7 Micro/Junior
BCD Binary file conversion	BCD	DTB	BCD_TO_INT
Binary BCD file conversion	BIN	BTD	INT_TO_BCD
ASCII Binary file conversion	ATB	ATB	STRING_TO_INT or STRING_TO_DINT
Binary ASCII file conversion	BTA	BTA	INT_TO_STRING or DINT_TO_STRING
Gray Binary file conversion	-	GTB	GRAY_TO_INT
Floating point Integer conversion	-	FTB	REAL_TO_INT or REAL_TO_DINT
Integer Floating point conversion	-	FTF	INT_TO_REAL or DINT_TO_REAL
BCD Floating point conversion	-	DTF	BCD_TO_REAL
Floating point BCD conversion	-	FTD	REAL_TO_BCD
ASCII Floating point conversion	-	ATF	STRING_TO_REAL
Floating point ASCII conversion	-	FTA	REAL_TO_STRING

Table instructions

Table showing differences between PL7-3 and PL7-Micro/Junior

Objects	PL7-3	PL7 Micro/Junior
Arithmetic operations	+, -, *, /, REM	+, -, *, /, REM
Logic operations	AND, OR, XOR	AND, OR, XOR, NOT
Addition of words from a table	+	SUM
Searching for the first different word	EQUAL	EQUAL
Searching for the first equivalent word	SEARCH	FIND_EQU

Program instructions

Table showing differences between PL7-3 and PL7-Micro/Junior

Objects	PL7-3	PL7 Micro/Junior
Jump	JUMP Li	JUMP %Li
Calling the subroutine	-	CALL SRi SRi
Return of subroutine	RET	RETURN
Stopping the application	HALT	HALT
Conditional sequence	IF/THEN/ELSE	IF/THEN/ELSE/END_IF
Iterative sequence	WHILE/DO	WHILE/DO/END_WHILE

Interruption instructions

Table showing differences between PL7-3 and PL7-Micro/Junior

Objects	PL7-3	PL7 Micro/Junior
Test	READINT	-
Masking	MASKINT	MASKEVT
Unmasking	DMASKINT	UNMASKEVT
Acknowledgment	ACKINT	-
Setting an IT on a module	SETIT	-

Explicit I/O instructions

Table showing differences between PL7-3 and PL7-Micro/Junior

Objects	PL7-3	PL7 Micro/Junior
Reading discrete inputs	READBIT	-
Writing discrete outputs	WRITEBIT	-
Reading registers	READREG	-
Writing registers	WRITEREG	-
Reading words	READEXT	-
Writing words	WRITEEXT	-

Function block instructions

Table showing differences between PL7-2, PL7-3 and PL7-Micro/Junior

Objects	PL7-3	PL7 Micro/Junior
Preset	PRESET Ti / Ci	PRESET %Ti / %Ci
Start	START Ti / Mi	START %Ti / %MNi
Task activation	START CTRLi	-
Reset to zero	RESET Ci / Ri / TXTi	RESET %Ci / %Ri
Task deactivation	RESET CTRLi	-
Counting	UP Ci	UP %Ci
Down counting	DOWN Ci	DOWN %Ci
Storing in a register	PUT Ri	PUT %Ri
Removal from register	GET Ri	GET %Ri
Receiving a message	INPUT TXTi	-
Sending a message	OUTPUT TXTi	-
Sending/Receiving a message	EXCHG TXTi	-
Execution of an OFB	EXEC < OFBi >	-
Reading telegrams	READTLG	-

Delimiters

Differences

Table showing differences between PL7-2/3 and PL7-Micro/Junior

Objects	PL7-2/3	PL7 Micro/Junior
Assignment	`->`	`:=`
Left bracket for indexing	(	[
Right bracket for indexing	)	]
Table length	`[length]`	`:length`

Memory aids

2

Introduction

Contents of this section

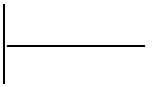
This section contains memory aids for PL7 language instructions

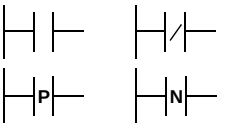
What's in this Chapter?

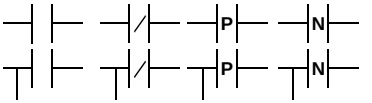
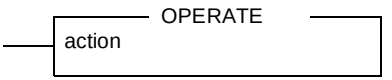
This Chapter contains the following Maps:

Boolean instructions

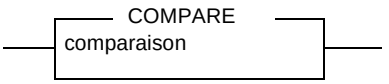
Memory aid Quick memory-aid for boolean instructions

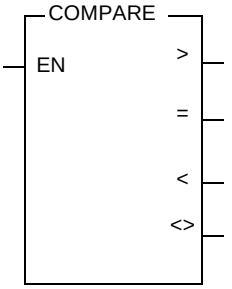
Boolean instructions	**LD**	**IL**
Accumulator or network intialization		`LD TRUE`

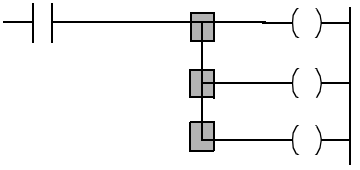
Direct (read) test, invert, rising edge, falling edge		`LD`
		`LDN`
		`LDR`
		`LDF`

And logic		`AND ANDN ANDR ANDF` `AND( AND(N AND(R AND(F`
Inversion	-	`N`
Or exclusive logic (direct, invert, rising edge, falling edge)	-	`XOR XORN` `XORR XORF`
Write (direct, inverse)		`ST` `STN`
Set Clear		`S` `R`
Operate block (contents: see following pages)		`[action]`

Boolean instructions	**LD**	**IL**

Horizontal block comparison (contents: see following pages)		`LD [comparison]`
		`AND [comparison]`
		`AND( [comparison]`
		`OR [comparison]`
		`OR( [comparison]`
		`XOR [comparison]`

Vertical comparison block		-

MemoryPusH
MemoryReaD
MemoryPOP
		`MPS`
		`MRD`
		`MPP`

ST Instructions

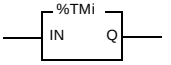
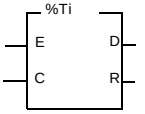
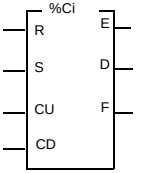
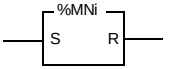
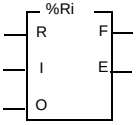
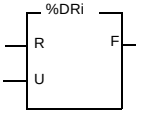
Memory aid

Quick memory-aid for ST instructions

Instructions	**ST**
Assignment	`:=`
OR boolean	`OR`
AND boolean	`AND`
OR boolean exclusive	`XOR`
Inversion	`NOT`
Rising edge, falling edge	`RE, FE`
Set on 1, set on 0	`SET, RESET`

LD and IL function blocks

Memory aid Quick memory-aid for LD and IL function blocks

Function blocks	LD	IL
IEC timers		IN **structure** BLK..END_BLK
PL-3 timer		-
Up/Down Counter		R S CU CD **structure** BLK..END_BLK
Monostable		S **structure** BLK..END_BLK
Register		R I O STN **structure** BLK..END_BLK
Drum		R U **structure** BLK..END_BLK

ST function blocks

Memory aid

Quick memory-aid for ST function blocks

Function blocks	**ST**
IEC timer	`START%TMi` `DOWN%TMi`

PL-3 timer	`PRESET%Ti`
	`START%Ti`
	`STOP%Ti`

Up/Down Counter	`RESET%Ci`
	`PRESET%Ci`
	`UP%Ci,DOWN%Ci`

Monostable	`START%MNi`

Register	`RESET%Ri`
	`PUT%Ri`
	`GET%Ri`

Drum	`RESET%DRi` `UP%DRi`

ST control structures

Memory aid

Quick memory-aid for ST control structures

Control structures	**ST**
Conditional action	`IF...THEN...ELSIF...THEN..ELSE...END_IF;`
Conditional iterative action	`WHILE...DO...END_WHILE;`
Conditional iterative action	`REPEAT...UNTIL...END_REPEAT;`
Repetitive action	`FOR...DO...END_FOR;`
Loop output instruction	`EXIT`

Full arithmetic (single and double length)

Memory aid

Quick memory-aid for full arithmetic instructions

Full arithmetic (single and double length)	**LD/IL/ST**
Transfer or initialization	`:=`
Comparisons	`= <> <= < > >=`
Addition, subtraction, multiplication, division, division remainder	`+ - * / REM`
ET, OU, OU exclusive, complement	`AND OR XOR NOT`
Absolute value, Square root	`ABS, SQRT`
Increment	`INC`
Decrement	`DEC`
Left shift	`SHL`
Right shift	`SHR`
Rotate shift left	`ROL`
Rotate shift right	`ROR`

Arithmetic on floating points

Memory aid Quick memory-aid for floating point arithmetic instructions

Arithmetic on floating points	**LD/IL/ST**
Transfer or initialization	`:=`
Comparisons	`= <> <= < > >=`
Addition, subtraction, multiplication, division, integer part	`+ - * / TRUNC`
Absolute value, square root	`ABS, SQRT`
Logarithm, exponentials	`LOG, LN, EXPT, EXP`
Sine, cosine, tangent	`SIN, COS, TAN`
Arc sine, arc cosine, arc tangent	`ASIN, ACOS, ATAN`
Degree <--> radian conversion	`DEG_TO_RAD, RAD_TO_DEG`

Digital conversions

Memory aid Quick memory-aid for digital conversion instructions

Digital conversions	LD/IL/ST
BCD conversion to single length integers	BCD_TO_INT
GRAY conversion to single length integers	GRAY_TO_INT
Single length integer conversion to BCD	INT_TO_BCD
Single length integer conversion to floating point	INT_TO_REAL
Double length integer conversion to floating point	DINT_TO_REAL
Floating point conversion to single length integers	REAL_TO_INT
Floating point conversion to double length integers	REAL_TO_DINT
32 bit BCD conversion to 32 bit integers	DBCD_TO_DINT
32 bit integer conversion to 32 bit BCD	DINT_TO_DBCD
32 bit BCD conversion to 16 bit integers	DBCD_TO_INT
16 bit integer conversion to 32 bit BCD	INT_TO_DBCD
Extracting the least significant word from a double word	LW
Extracting the most significant word from a double word	HW
Concatenation of 2 single words	CONCATW

Bit tables

Memory aid Quick memory-aid for instructions on bit tables

Bit tables	**LD/IL/ST**
Transfer or initialization	`:=`
Copy of a bits table in a bits table	`COPY_BIT`
AND between two tables	`AND_ARX`
OR between two tables	`OR_ARX`
OR exclusive between two tables	`XOR_ARX`
Negation in a table	`NOT_ARX`
Copy of a bits table in a word table	`BIT_W`
Copy of a bits table in a double word table	`BIT_D`
Copy of a word table in a bits table	`W_BIT`
Copy of a double word table in a bits table	`D_BIT`
Calculation of table length	`LENGTH_ARX`

Instructions on tables

Memory aid Quick memory aid for instructions on tables

Instructions on tables	**LD/IL/ST**
Transfer or initialization	`:=`
Arithmetic operations between tables	`+ - * / REM`
Logic operations between tables	`AND OR XOR`
Arithmetic operations between a table and an integer	`+ - * / REM`
Logic operations between a table and an integer	`AND OR XOR`
Table elements complement	`NOT`
Sum of all the table elements	`SUM`
Comparison of two tables	`EQUAL`
Search for the first table element which is equal to certain value	`FIND_EQW, FIND_EQD`
Search for the first table element which is equal to certain value from a set position	`FIND_EQWP, FIND_EQDP`
Search for the first table element which is greater than a certain value	`FIND_GTW, FIND_GTD`
Search for the first table element which is less than a certain value	`FIND_LTW, FIND_LTD`
Search for the greatest value in a table	`MAX_ARW, MAX_ARD`
Search for the smallest value in a table	`MIN_ARW, MIN_ARD`
Number of occurrences of value within a table	`OCCUR_ARW, OCCUR_ARD`
Left rotation shift of a table	`ROL_ARW, ROL_ARD`
Right rotation shift of a table	`ROR_ARW, ROL_ARW`
Table sort (increasing or decreasing direction)	`SORT_ARW, SORT_ARD`
Calculation of table length	`LENGTH_ARW, LENGTH_ARD`

Floating point table instructions

Memory aid Quick memory aid for instructions on floating point tables

Floating point table instructions	**LD/IL/ST**
Transfer and intialization	`:=`
Sum of all the table elements	`SUM_ARR`
Comparison of two tables	`EQUAL_ARR`
Search for the first table element which is equal to certain value	`FIND_EQR`
Search for the first table element which is greater than a certain value	`FIND_GTR`
Search for the first table element which is less than a certain value	`FIND_LTR`
Search for the greatest value in a table	`MAX_ARR`
Search for the smallest value in a table	`MIN_ARR`
Number of occurrences of a certain value within a table	`OCCUR_ARR`
Left rotation shift of a table	`ROL_ARR`
Right rotation shift of a table	`ROR_ARR`
Table sort (increasing or decreasing direction)	`SORT_ARR`
Calculation of table length	`LENGTH_ARR`

"Orpheus" instructions

Memory aid Quick memory aid for "Orpheus" instructions

"Orpheus" instructions	**LD/IL/ST**
Left shift on the word with shifted bit retrieval	`WSHL_RBIT, DSHL_RBIT`
Right shift on the word with sign extension and shifted bit retrieval	`WSHR_RBIT, DSHR_RBIT`
Right shift on the word, with filling using 0 and shifted bit retrieval	`WSHRZ_C, DSHRZ_C`
Up/down counting with overshoot signaling	`SCOUNT`
Left rotation shift	`ROLW, ROLD`
Right rotation shift	`RORW, RORD`

Explicit exchanges

Memory aid Quick memory aid for explicit exchanges

Explicit exchanges	**LD/IL/ST**
Read %M parameters using a logic channel	READ_PARAM
Read status %M using a logic channel	READ_STS
Restore %M parameters using a logic channel	RESTORE_PARAM
Save %M parameters using a logic channel	SAVE_PARAM
Write command %M using a logic channel	WRITE_CMD
Write %M parameters using a logic channel	WRITE_PARAM

Time management instructions

Memory aid Quick memory aid for time management instructions

Time management instructions	**LD/IL/ST**
Dater	SCHEDULE
Comparisons	= <> <= < > >=
Transfer	:=
Reading of date and code for last PLC stop	PTC
Reading of system date	RRTC
System date update	WRTC
Add time period to a complete date	ADD_DT
Add time period to a time of day	ADD_TOD
Date string conversion	DATE_TO_STRING
Day of the week	DAY_OF_WEEK
Interval between two dates	DELTA_D
Interval between two complete dates	DELTA_DT
Interval between two times of day	DELTA_TOD
Complete date string conversion	DT_TO_STRING
Remove time period from a complete date	SUB_DT
Remove time period from a time of day	SUB_TOD
Duration string conversion	TIME_TO_STRING
Time of day string conversion	TOD_TO_STRING
Format duration to hours-mn-secs	TRANS_TIME

Timing instructions

Memory aid Timing instructions quick memory aid

Timing instructions	**LD/IL/ST**
Timing on trigger event	FTON
Timing on trigger event	FTOF
Pulse timing	FTP
Rectangular signal generator	FPULSOR

Data storage instructions

Memory aid Data storage instructions rapid memory aid

Data storage instructions	**LD/IL/ST**
PCMCIA card storing zone setting	SET_PCMCIA
PCMCIA card data writing	WRITE_PCMCIA
PCMCIA card data reading	READ_PCMCIA

Character string instructions

Memory aid Character string instructions rapid memory aid

Character string instructions	**LD/IL/ST**
Comparisons	= <> <= < > >=
Transfer	:=
Converting a double integer into a string	DINT_TO_STRING
Converting a single integer into a string	INT_TO_STRING
Converting a string into a double integer	STRING_TO_DINT
Converting a string into a single integer	STRING_TO_INT
Floating point string conversion	STRING_TO_REAL
Floating point conversion into a string	REAL_TO_STRING
Two string concatenation	CONCAT
Deleting a substring	DELETE
Searching for a different initial character	EQUAL_STR
Searching for a substring	FIND
Inserting a substring	INSERT
Extracting the left part of a string	LEFT
String length	LEN
Extracting a substring	MID
Replacing a substring	REPLACE
Extracting the right part of a string	RIGHT

Multi-tasking and events

Memory aid Multi-tasking and event rapid memory aid

Multi-tasking and events	**LD/IL/ST**
Task activation / deactivation	`%Si position`
Adjusting the task cycle time	`%SWi position`
Global event masking.	`MASKEVT`
Global event unmasking.	`UNMASKEVT`

Communication

Memory aid Communication instructions rapid memory aid

Communication	**LD/IL/ST**
Stop request for function in action	CANCEL
Send / receive data	DATA_EXCH
Character string read request	INPUT_CHAR
Send / receive character string request	OUT_IN_CHAR
Send a character string:	PRINT_CHAR
Receiving a telegram	RCV_TLG
Basic language object reading	READ_VAR
Send / receive UNI-TE requests	SEND_REQ
Send a telegram	SEND_TLG
Basic language object writing	WRITE_VAR
Shifting a byte to the right in a table	ROR1_ARB
Swapping bytes in a word table	SWAP
Reading Modbus+ common data	READ_GDATA
Writing Modbus+ common data	WRITE_GDATA
Calling the modem	CALL_MODEM
Immediate server	SERVER
Writing 1 K messaging	WRITE_ASYN
Reading 1 K messaging	READ_ASYN

Reserved words

3

Reserved words

List of reserved words

The following reserved words must not be used as symbols.

Words from A to C

List of reserved words

ABS	ANY_REAL	BLOCK	CLOSED_CONTACT
ACCEPT	ARRAY	BODY	COIL
ACOS	AR_D	BOOL	COMMAND
ACTION	AR_DINT	BOTTOM	COMMENTS
ACTIVATE_PULSE	AR_F	BTI	COMP4
ACTIVE_TIME	AR_INT	BTR	COMPCH
ADD	AR_R	BY	CONCAT
ADDRESS	AR_W	BYTE	CONCATW
ADD_DT	AR_X	C	CONF
ADD_TOD	ASIN	CAL	CONFIGURATION
ADR	ASK	CALC	CONSTANT
AND	ASK_MSG	CALCN	CONTROL_LEDS
ANDF	ASK_VALUE	CALL	COPY_BIT
ANDN	ASSIGN_KEYS	CALL_COIL	COS
ANDR	AT	CANCEL	CTD
AND_ARX	ATAN	CASE	CTU
ANY	AUX	CD	CTUD
ANY_BIT	BCD_TO_INT	CHART	CU
ANY_DATE	BIT_D	CH_M	
ANY_INT	BIT_W	CLK	
ANY_NUM	BLK	CLOSE	

Words from D to E List of reserved words

D	DO	END	END_REPEAT
DATE	DOWN	ENDC	END_RESOURCE
DATE_AND_TIME	DR	ENDCN	END_RUNG
DAT_FMT	DRUM	END_ACTION	END_STEP
DAY_OF_WEEK	DS	END_BLK	END_STRUCT
DA_TYPE	DSHL_RBIT	END_BLOCK	END_TRANSITION
DEACTIVATE_PULSE	DSHRZ_C	END_CASE	END_TYPE
DEC	DSHR_RBIT	END_COMMENTS	END_VAR
DELETE	DSORT_ARD	END_CONFIGURATION	END_WHILE
DELTA_D	DSORT_ARW	END_FOR	EQ
DELTA_DT	DT	END_FUNCTION	EQUAL
DELTA_TOD	DTS	END_FUNCTION_BLOCK	EQUAL_ARR
DINT	DWORD	END_IF	ERR
DINT_TO_REAL	D_BIT	END_MACRO_STEP	EVT
DINT_TO_STRING	E	END_PAGE	EXCHG
DISPLAY_ALRM	EBOOL	END_PHRASE	EXCH_DATA
DISPLAY_GRP	ELSE	END_PROG	EXIT
DISPLAY_MSG	ELSIF	END_PROGRAM	EXP
DIV	EMPTY		EXPT
DMOVE	EMPTY_LINE		

Words from F to J List of reserved words

F	FIND_LTW	GR7	INFO
FALSE	FOR	GRAY_TO_INT	INITIAL_STEP
FAST	FPULSOR	GT	INIT_BUTTONS
FBD	FROM	GTI	INPUT
FE	FTOF	H	INPUT_CHAR
FIFO	FTON	HALT	INSERT
FIND	FTP	HALT_COIL	INT
FIND_EQ	FUNC	HASH_COIL	INTERVAL
FIND_EQD	FUNCTION	HW	INT_TO_BCD
FIND_EQDP	FUNCTION_BLOCK	H_COMPARE	INT_TO_REAL
FIND_EQR	F_B	H_LINK	INT_TO_STRING
FIND_EQW	F_EDGE	I	ITB
FIND_EQWP	F_TRIG	IF	ITS
FIND_GTD	GE	IL	JMP
FIND_GTR	GET	IN	JMPC
FIND_GTW	GET_MSG	INC	JMPCN
FIND_LTD	GET_VALUE	INCJUMP	JUMP
FIND_LTR	GLOBAL_COMMENT	INDEX_CH	JUMP_COIL

Words from L to M

List of reserved words

L	LIFO	MASKEVT	MIN_ARR
LAD	LIMIT	MAST	MIN_ARW
LANGAGE	LINT	MAX	MN
LANGUAGE	LIST	MAX_ARD	MOD
LD	LIT	MAX_ARR	MONO
LDF	LN	MAX_ARW	MOVE
LDN	LOCATION	MAX_PAGES	MPP
LDR	LOG	MAX_STEP	MPS
LE	LREAL	MCR	MRD
LEFT	LT	MCR_COIL	MS
LEN	LW	MCS	MUL
LENGTH_ARD	LWORD	MCS_COIL	MUX
LENGTH_ARR	M	MID	M_CH
LENGTH_ARW	MACRO_STEP	MIN	M_MACRO_STEP
LENGTH_ARX	MAIN	MIN_ARD	

Words from N to P

List of reserved words

N	NB_TRANSITIONS	OPEN_CONTACT	PID
N1	NE	OPERATE	PID_MMI
NAME	NIL	OR	PLC
NB_ACTIVE_STEPS	NO	ORF	POST
NB_ACTIVE_TIME	NON_STORED	ORN	PRESET
NB_BLOCKS	NOP	ORR	PRINT
NB_COMMON_WORDS	NOT	OR_ARX	PRINT_CHAR
NB_CONSTANT_WORDS	NOT_ARX	OTHERS	PRIO0
NB_CPT	NOT_COIL	OUT	PRIO1
NB_DRUM	NOT_READABLE	OUTIN_CHAR	PRIORITY
NB_INTERNAL_BITS	NO_GR7	OUTPUT	PRL
NB_INTERNAL_WORDS	NO_PERIOD	OUT_BLK	PROG
NB_MACRO_STEPS	N_CONTACT	P	PROGRAM
NB_MONO	O	P0	PROG_LANGAGE
NB_PAGES	OCCUR	P1	PROG_LANGUAGE
NB_REG	OCCUR_ARD	PAGE	PT
NB_TIMER	OCCUR_ARR	PAGE_COMMENT	PTC
NB_TM	OCCUR_ARW	PANEL_CMD	PUT
	OF	PERIOD	PV
	ON	PHRASE	PWM
	OPEN	PHRASE_COMMENT	P_CONTACT

Words from Q to R

List of reserved words

Q	REAL_TO_INT	RETURN	ROR_ARR
QUERY	REAL_TO_STRING	RET_COIL	ROR_ARW
R	REG	RIGHT	ROR_DWORD
R1	REM	ROL	ROR_WORD
RCV_TLG	REPEAT	ROLD	RRTC
RE	REPLACE	ROLW	RS
READ	REQ	ROL_ARD	RTB
READ_EVT_UTW	RESET	ROL_ARR	RTC
READ_ONLY	RESET_COIL	ROL_ARW	RTS
READ_PARAM	RESOURCE	ROL_DWORD	RUNG
READ_STS	RESTORE_PARAM	ROL_WORD	R_EDGE
READ_VAR	RET	ROR	R_TRIG
READ_WRITE	RETAIN	RORD	
REAL	RETC	RORW	
REAL_TO_DINT	RETCN	ROR_ARD	

Words from S to S

List of reserved words

S	SEND_REQ	SL	STN
S1	SEND_TLG	SLCWORD	STOP
SAVE	SERVO	SMOVE	STR
SAVE_PARAM	SET	SOFT_CONFIGURATION	STRING
SCHEDULE	SET_COIL	SORT	STRING_TO_DINT
SD	SFC	SORT_ARD	STRING_TO_INT
SEARCH	SHIFT	SORT_ARR	STRING_TO_REAL
SECTION	SHL	SORT_ARW	STRUCT
SEL	SHOW_ALARM	SQRT	SUB
SEMA	SHOW_MSG	SR	SUB_DT
SEND	SHOW_PAGE	ST	SUB_TOD
SENDER	SHR	STANDARD	SUM
SEND_ALARM	SHRZ	START	SU_TYPE
SEND_MBX_ALARM	SIN	STD	SWAP
SEND_MBX_MSG	SINGLE	STEP	S_T_AND_LINK
SEND_MSG	SINT	STI	S_T_OR_LINK

Words T to W

List of reserved words

T	TOP	UP	W
TAN	TP	USINT	WHILE
TASK	TRANSITION	USORT_ARD	WITH
TASKS	TRANS_TIME	USORT_ARW	WORD
THEN	TRUE	UTIN_CHAR	WRITE
TIME	TRUNC	VAR	WRITE_CMD
TIMER	TYPE	VAR_ACCESS	WRITE_PARAM
TIME_OF_DAY	TYPES	VAR_EXTERNAL	WRITE_VAR
TM	T_S_AND_LINK	VAR_GLOBAL	WRTC
TMAX	T_S_OR_LINK	VAR_INPUT	WSHL_RBIT
TMOVE	U	VAR_IN_OUT	WSHRZ_C
TO	UDINT	VAR_OUTPUT	WSHR_RBIT
TOD	UINT	VAR_PUBLIC	W_BIT
TOF	ULINT	VERSION	
TOFF	UNMASKEVT	V_COMPARE	
TON	UNTIL	V_LINK	

Words X, Y, and Misc

List of reserved words

XM	*_TO_* * = Letter
XM_MONO	SRi
XM_MULTI	AUXi
XOR	EVTi
XORF	XMi
XORN	i = integer
XORR	
XOR_ARX	
YES	

Compliance with IEC standard 1131-3

4

Introduction

Contents of this section

This section describes compliance with IEC standard 1131-3: "Programmable Controllers"

What's in this Chapter?

This Chapter contains the following Maps:

Compliance with IEC standard 1131-3

Introduction to IEC standard 1131-3

IEC Standard 1131-3 "Programmable controllers – Part 3: Programming languages" specifies the syntax and semantics of software elements used for programming programmable controllers.

This standard contains the description of 2 text languages, IL (Instruction List) and ST (Structured Text); of 2 graphic languages, LD (Ladder Diagram) and FDB (Function Block Diagram); and a graphic formalism, SFC (Sequential Function Chart) which is used to structure the internal organization of a programmed sequence.

PL7 programming software is used to program a programmable controller in accordance with the IEC standard. PL7 implements a sub-assembly of language elements defined by the standard, and defines extensions which are authorized within the framework of this standard.

The IEC standard 1131-3 does not define the interactivity rules for software supplied by a manufacturer claiming compliance with the standard. This allows marked flexibility for the introduction and entry of programming elements for the convenience of the user.

The elements of the standard implemented in PL7, information relating to implementation and error detection are all summarized in the compliance tables.

Common elements

General

Description of the characteristics common to languages which comply with IEC 1131-3
The following table lists implementations in PL7 languages. This information is in regard to tables of characteristics described in the IEC 1131-3 standard. This is for characteristics common to all PL7 languages.

Table of characteristics

Characteristics common to all languages:

Table No.	Characteristic No.	Description of characteristics
1	1	Character set required (see paragraph 2.1.1 of 1131-3)
1	2	Lower case characters
1	3a	Number sign (#)
1	4a	Dollar sign ($)
1	5a	Vertical line (|)
1	6a	Index delimiters Left and right square brackets "[ ]"
2	1	Upper case and numbers
2	2	Upper and lower case, numbers, integrated character underlining
3	1	Comments
4	1	Text integers (Note 1)
4	2	Text integers (Note 1)
4	3	Text integers with exponents
4	4	Literals in base 2 (Note 1)
4	6	Literals in base 16 (Note 1)
4	7	Zero and One Booleans
4	8	TRUE and FALSE Booleans
5	1	Text character string features
6	2	$$ Dollar sign
6	3	$' Apostrophe
6	4	$L or $l Change line
6	5	$N or $n New line
6	6	$P or $p Change page
6	7	$R or $r Cursor return
6	8	$T or $t Tabulation
7	1a	Time period text with t# short prefix (Note 2)

Table No.	Characteristic No.	Description of characteristics
10	1	BOOL -1 bit-
10	10	REAL -32 bits-
10	12	TIME -32 bits- (Note 3)
10	13	DATE -32 bits- (Note 3)
10	14	TIME_OF_DAY -32 bits- (Note 3)
10	15	DATE_AND_TIME -64 bits- (Note 3)
10	16	STRING
10	17	BYTE -8 bits-
10	18	WORD -16 bits-
10	19	DWORD -32 bits-
15	1	Prefix I for Input
15	2	Prefix Q for Output
15	3	Prefix M for Memory
15	4	Prefix X, bit size
15	5	No prefix, bit size
15	6	Prefix B, byte size (8 bits)
15	7	Prefix W, word size (16 bits)
15	8	Prefix D, size of double word (32 bits)
16	VAR VAR_INPUT VAR_OUTPUT VAR_IN_OUT VAR_EXTERNAL VAR_GLOBAL CONSTANT AT	Key words (Note 4)
17	2	Declaration of directly represented non-volatile variables (addresses) (Note 4)
17	3	Declaration of symbolic variables slots (symbols of addresses) (Note 4)
17	5	Automatic attribution of symbolic variables to memory (function block variables) (Note 4)
18	2	Initialization of directly represented non-volatile variables (addresses) (Note 4)
18	3	Slot assignation of initial values and symbolic variables (symbols on addresses) (Note 4)
18	5	Initialization of symbolic variables (function block variables) (Note 4)

Table No.	Characteristic No.	Description of characteristics
21	1	Overloaded PL7 functions are as follows; ABS, EQUAL, ROL, ROR, SHL, SHR, SQRT, SUM
21	2	In general, PL7 functions belong to this category.
22	3	BCD_TO_INT conversion function (Note 6)
22	4	INT_TO_BCD conversion function (Note 6)
23	1	ABS function: absolute value
23	2	SQRT function: square root
23	3	LN function: natural logarithm
23	4	LOG function: logarithm in base 10
23	5	EXP function : natural exponential
23	6	SIN function: sine in radians
23	7	COS function: cosine in radians
23	8	TAN function: tangent in radians
23	9	ASIN function: arc sine
23	10	ACOS function: arc cosine
23	11	ATAN function: arc tangent
25	1	SHL function: left shift
25	2	SHR function: right shift
25	3	ROR function: right rotation
25	4	ROL function: left rotation
29	1	LEN function: string length
29	2	LEFT function: n the most characters to the left
29	3	RIGHT function: n the most characters to the right
29	4	MID function: n characters from a given position
29	5	CONCAT function: string concatenation (Note 7)
29	6	INSERT function: inset one string into another
29	7	DELETE function: delete characters
29	8	REPLACE function: replace with other characters
29	9	FIND function: search for one string within another
32	Inputread Inputwrite Outputread Outputwrite	(Note 8)
33	1	RETAIN qualifier for function block internal variables.) (Note 9) (Note 4)
33	2	RETAIN qualifier for function block output (Note 9) (Note 4)

Table No.	Characteristic No.	Description of characteristics
33	4a	Declaring (literal) function block input/output. (Note 4)
37	1	Pulse timer: TP (Note 10)
37	2a	Engagement timer: TON (Note 10)
37	3a	Trigger timer: TOF (Note 10)
38	time diagrams	TP, TON, TOF
39	19	Usage of directly represented variables (addresses)
40	1	Step, graphical form Note: A step number replaces a step identifier
40	2	Step, textual format used in Grafcet source format only
41	1	Transition condition in ST language
41	2	Transition condition in LD language
42	2l	Action declaration in LD language
43	1	Action block
	2	Concatenated action blocks
45	2	Action N qualifier (not memorized)
45	11	Action P1 qualifier (Pulse rising edge)
45	12	Action P0 qualifier (Pulse falling edge)
46	1	Simple sequence, alternating between step/transition
46	2c	Divergence in "or": the user ensures that transition conditions are mutually exclusive.
46	3	Convergence in "or"
46	4	Divergence in "and", Convergence in "and"
46	5c	Sequence jump in a divergence in "or"
46	6c	Sequence loop: return to previous step
46	7	Directional arrows Note: Directional arrows rise and descend
48	40 41 42 43 44 45 46	Grafcet language fulfils the conditions required for the minimum level of compliance with SFC 1131-3 Graphic presentation
49	3	RESOURCE...ON...END_RESOURCE construction
49	5a	Periodic TASK construction in RESOURCE

Table No.	Characteristic No.	Description of characteristics
49	6a	PROGRAM declaration with PROGRAM-to-TASK association
49	7	Declaration of variables directly represented in VAR_GLOBAL
50	5b	Pre-emptive scheduling in multi-tasking model

Note:

- **Note 1:** The underlining characters (_) inserted between figures in numerical text are not accepted.
- **Note 2:** This text is only visible in the source application for expressing configured task times.
- **Note 3:** This type of data has not yet been set up for the user to be able to see. Nevertheless, this table specifies how their internal representation takes up the memory.
- **Note 4:** These key words are only used in the sources generated by PL7 and by the PL7-2 and PL7-3 application conversion tool.
- **Note 5:** Effects of limited conversions:
  DINT_TO_STRING: If the string receiving the result is less than 13 characters, %S15 is sectioned and re-positioned.
  INT_TO_STRING: If the string receiving the result is less than 7 characters, %S15 is sectioned and re-positioned.
  STRING_TO_DINT and STRING_TO_INT: If the string is not fully convertible, the result is undetermined and %S18 is re-positioned.
  DATE_TO_STRING: If the string receiving the result is less than 11 characters, %S15 is sectioned and re-positioned.
  DT_TO_STRING: If the string receiving the result is less than 20 characters, %S15 is sectioned and re-positioned.
  TIME_TO_STRING: If the string receiving the result is less than 15 characters, %S15 is sectioned and re-positioned.
  TOD_TO_STRING: If the string receiving the result is less than 9 characters, %S15 is sectioned and re-positioned.
  REAL_TO_STRING: If the string receiving the result is less than 15 characters, %S15 is sectioned and re-positioned.
  STRING_TO_REAL: If the string is not convertible to integers, the result is worth "1#NAN" (16#FFC0_0000) and %S18 is re-positioned.
  REAL_TO_INT: If the integer is not convertible within the limits of [-32768, =32767], the result is worth –32768 and both %S18 and %SW17 X0 are repositioned.
  REAL_TO_DINT: If the integer is not convertible within the limits of [-2147483648, =+2147483647], the result is worth -2147483648 and both %S18 and %SW17 X0 are repositioned.
  INT_TO_REAL: Conversion is always possible.
  DINT_TO_REAL: Conversion is always possible.
- **Note 6:** With the INT type not being formally implemented, but used all the same, these functions are used to change the format of a WORD coding.
- **Note 7:** CONCAT function limitations on concatenating 2 strings.
- **Note 8:** This paragraph applies to predefined PL7 function blocks.
- **Note 9:** The RETAIN qualifier is implicit.
- **Note 10:** Timers TP, TON and TOF respect the table 38 time graphs, but offer a I/O interface different to that of 1131-3.

IL language elements

General

Description of the characteristics of IL language elements which comply with IEC 1131-3
The following table lists implementations in PL7 languages. This information is in regard to tables of characteristics described in the IEC 1131-3 standard.

Table of characteristics

Characteristics of IL language elements:

Table No.	Characteristic No.	Description of characteristics
51	Instruction fields	Label, operator, operand, comment
52	1	LD
52	2	ST
52	3	S and R
52	4 5 6	AND OR XOR
52	18	JMP
52	20	RET
52	21	)
53	3	Usage of input operators for starting function blocks in IL
54	11	IN (see Note)
54	12	IN (see Note)
54	13	IN (see Note)

Note: PT operator has not been set up.

ST language elements

General

Description of the characteristics of ST language elements which comply with IEC 1131-3
The following table lists implementations in PL7 languages. This information is in regard to tables of characteristics described in the IEC 1131-3 standard.
This language is separately but fully used in ST modules. An ST sub-assembly is also used in IL and LD language OPERATE and COMPARISON blocks.

Table of characteristics

Characteristics of ST language elements:

Table No.	Characteristic No.	Description of characteristics
55	1	Between brackets
55	2	Function evaluation
55	3	- Negation
55	4	NOT Complement
55	5	JMP
55	6 7	* Multiplication / Division
55	9 10	+ Addition - Subtraction
55	11	<, >, <=, >= Comparison
55	12	= Equate
55	13	<> Does not equate
55	15	AND for the "and" boolean
55	16	XOR for the "exclusive or" boolean
55	17	OR for the "or" boolean
56	1	:= Assignment
56	3	RETURN structure
56	4	IF structure "if... then... elsif... then... else... end_if"
56	6	FOR structure "for... to... do... end_for" (see Note)
56	7	WHILE structure "while... do... end_while"
56	8	REPEAT structure "repeat ... until... end_repeat"
56	9	EXIT structure

Note: Set-up of FOR loop with an implicit step of 1 (by 1).

Common graphics elements

General

Description of the characteristics of common graphics elements which comply with IEC 1131-3
The following table lists implementations in PL7 languages. This information is in regard to tables of characteristics described in the IEC 1131-3 standard.

Table of characteristics

Characteristics of common graphics elements:

Table No.	Characteristic No.	Description of characteristics
57	2	Horizontal graphic lines
57	4	Vertical graphic lines
57	6	Horizontal graphic line / vertical graphic line junction point
57	8	Graphic crossing of lines without connections
57	10	Connected and unconnected graphic corners
57	12	Blocks with connected graphic lines
58	2	Unconditional jump in LD language
58	4	Conditional jump in unconditional LD language
58	5	Conditional return in LD language
58	8	Unconditional return in LD language

LD language elements

General

Description of the characteristics of LD language elements which comply with IEC 1131-3
The following table lists implementations in PL7 languages. This information is in regard to tables of characteristics described in the IEC 1131-3 standard.

Table of characteristics

Characteristics of LD language elements:

Table No.	Characteristic No.	Description of characteristics
59	1	Left power rail
59	2	Right power rail
60	1	Horizontal link
60	2	Vertical link
61	1	Open contact
61	3	Closed contact
61	5	Positive transition contact detector
61	7	Negative transition contact detector
62	1	Coil
62	2	Negated coil
62	3	SET (latch) coil
62	4	RESET (unlatch) coil

Implementation-dependent parameters

General

Description of PL7 parameters which depend on implementation.
The following table lists implementations in PL7 languages. This information is in regard to tables of characteristics described in the IEC 1131-3 standard.

Table of characteristics

Characteristics of IL language elements:

Parameters	PL7 limitations and behavior
Procedure for processing errors	Many errors can be indicated on execution by the way that bits and system words are arranged.
Nationally used characters	ÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÑÒÓÔ ÕÖØÙÚÛÜàáâãäåæçèéêëìí îïñòóôõöøùúûüßÿ #, $, |
Maximum identifier length	32
Maximum comment length	222
Duration value format	Note 1
TIME type value format More detail on the presentation of seconds in TIME_OF_DAY and DATE_AND_TIME types	Note 1 Note 2
Maximum number of table indexes Maximum table size	1 (Note 3) depends on the indexed zone (Note 3)
Maximum default length of STRING variables Maximum authorized length for STRING variables	does not apply 255
Maximum number of hierarchical levels Logical or physical configuration	3 Logic configuration
Maximum interval of index values	depends on the indexed zone (Note 3)
Initialization of system input	Variables are initialized by the system: ● to the initial value specified by the user, if need be ● if not to zero
Effect of conversions on more detail	cf table 22, characteristic 1
Maximum number of types and function block instances	Not the maximum number (limits are linked to the volume of the application)
Program size limit	Maximum program code volume = 64 Kbytes
More detail on time elapsed associated with a step	100ms

Parameters	PL7 limitations and behavior
Maximum number of steps per chart	96 on 3710 PLCs 128 on 3720 PLCs 1024 on 57xx V3.0 PLCs
Maximum number of transitions per chart and per step)	1024 transitions per chart 11 transition per step 1
Action control mechanism	Qualifiers P0, P1 and N1
Maximum number of action blocks per step	3 types of action are possible: on activation (P1), continue (N1) and on deactivation (P0)
Graphic representation of state of step	Active step is highlighted
Time for overshooting a transition (disabling of upstream steps, and enabling of downstream steps)	The overshoot time varies, and is never zero
Depth of divergent and convergent constructions	Limit given by entry grid
List of PLCs that can be programmed by PL7	TSX MICRO, TSX PREMIUM
Maximum number of tasks Task interval formats Pre-emptive or non-pre-emptive scheduling	1 periodic or cyclic task 1 periodic task 8 event tasks for 37 10 PLCs 16 event tasks for 37 20 PLCs 32 event tasks for 57 10 PLCs 64 event tasks for 57 20/30 PLCs from 1 ms to 225 ms Pre-emptive scheduling
Maximum length of an expression Partial evaluation of boolean expressions	variable no
Maximum length of command structures ST	variable
Value of command variable after complete execution of a FOR loop	The command variable value equals the limit value + 1 (as the step is 1)
Graphic/semi-graphic representation Restrictions on network topology	Graphic representation An LD network may extend over a maximum of 16 columns and 7 lines

Note:

- **Note 1:** This type of data has not yet been set up for the user to be able to see. Nevertheless, this table details their value formats in the IEC 1131-3 format.
  TIME : from T#0 to T#429496729.5s
  TIME_OF_DAY: from TOD#0:0:0 to TOD#23:59:59
  DATE_AND_TIME: from DT#1990-01-01:0:0:0 to DT#2099-12-31:23:59:59
  DATE: from D#1990-01-01 to D#2099-12-31DT#2099-12-3
- **Note 2:** Rounding is done as follows: from x.0 s to x.4 s, you round to x s and from x.5 s to x.9 s you round to x+1 s.
- **Note 3:** All types of directly represented variable can be indexed positively and negatively within the limit of their respective maximums defined in configuration.

Error situations

General

Description of PL7 parameters which depend on implementation.
The following table lists implementations in PL7 languages. This information is in regard to tables of characteristics described in the IEC 1131-3 standard.

Table of characteristics

Error situations:

Error situations	PL7 limitations and behavior
Type conversion errors	Indicated during execution via the arrangement of a system bit: cf Common elements table: table 22, characteristic 1
The digital result exceeds the format for the data type	Indicated during execution by the data type repositioning the system bit %S18
Position of specified characters is invalid	Indicated during execution via the arrangement of system bit %S18
The result exceeds the maximum string length	Indicated during execution via the string arranging system bit %S15
Edge effects during transition evaluation	Detected during programming
Execution delays are not respected Other task scheduling conflicts	Indicated on execution via the arrangement of system bit %S19 Detected during configuration
Division by 0 Data type invalid for an operation	Detected during programming if possible, if not it is indicated during execution via the arrangement of system bit %S18
FOR or WHILE iteration failure to be finished	On the occurrence of a fault, the PLC overruns the watchdog and the programming unit concerned is then indicated

OLE Automation Server

5

Introduction

Contents of this section

This section describes how the OLE Automation server works

What's in this Chapter?

This Chapter contains the following Sections:

5.1 Introduction

Introduction

Subject of this sub-section

This sub-section makes some general points concerning the OLE Automation server

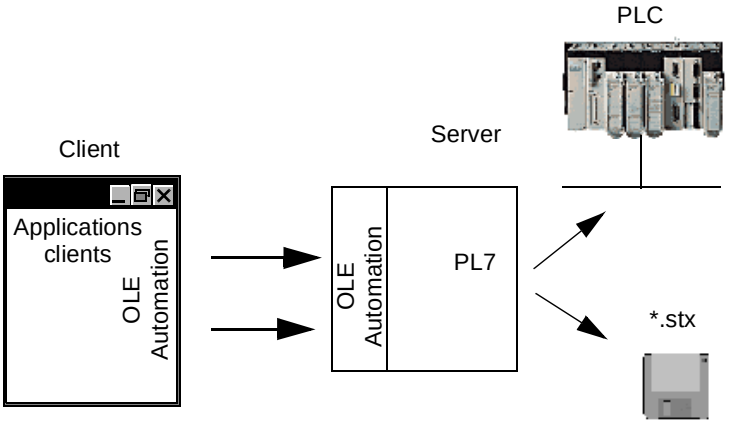
What's in this Section?

This Section contains the following Maps:

Introduction to the OLE Automation server

Introduction

The **OLE Automation Server** PL7 functionality offers the possibility of executing the PL7 in a programmed, non-interactive mode (i.e. without operator intervention) for applications external to the PL7. You use PL7 OLE Automation Server client application terminology.



Conforming fully with the standard defined by Microsoft in its OLE package to respond to this need, PL7 now features a standard, public interface, which is independent of the client application programming language.
The characteristic principles are:

- automation of repetitive tasks
- delegation to PL7 of processing tasks that only it can handle
- access to application information contained in the *.stx files
- more generally, development of higher level integrator type function, with encapsulation of PL7 specific implementation details.

Context of use

The OLE Automation server PL7 functionality is built into PL7 Pro. It is installed and used under the same installation and use conditions as PL7. The OLE Automation server is supported by the Microsoft OLE Automation run-time provided with Windows 95, 98 and NT

Note: To create an OLE server client application, you must know one of the following OLE Automation programming languages:

- Microsoft Visual Basic, version 5.0 or above
- Microsoft Visual C++, version 4.2 or above
- Microsoft VBA in Excel, version 5.0 or above.

OLE Automation server operating modes

General

In addition to the PL7 operating mode known up until the present which is designated as "interactive", where the PL7 reacts to operator inputs only, there is now the OLE automation server operating mode where the PL7 can also react to commands transmitted from an OLE client application.

Description

Operating mode selection is dependen on the PL7 start procedure:

- **Interactive mode** is initialized during PL7 start via Windows (Start menu/Programs/ Modicon Telemecanique/PL7 Pro.).
- **Server mode** is selected if the OLE Automation PL7 server is invoked in the programming of an OLE client application. An instance can not change the current mode.

Each time a client application invokes the OLE Automation PL7 server, a specific PL7 instance is started, independently of any other instances which may be in progress. Any number of PL7 instances in server mode or interactive mode can be running on the same terminal. The instances are all completely independent of each other and so can each run in their own context.

For all of these instances the same denial rules concerning concurrent access to an STX application or a PLC apply, these being: an STX application or a PLC can only be handled by one instance at a time.
This rule was developed for the OLE Automation PL7 server which can open an already open STX application but not save it. In the same way, it can perform a PLC —> PC transfer, on explicit demand, from a reserved PLC.
A client application can instantiate several competing "OLE Automation PL7 servers".

5.2 Implementation

Introduction

Subject of this sub-section

This sub-section describes the procedure for implementation of the OLE Automation server

What's in this Section?

This Section contains the following Maps:

Installation of OLE Automation

Description

Installation of the PL7 product OLE Automation Server is transparent for the user, as it self-installs during PL7 installation.
The installation includes:

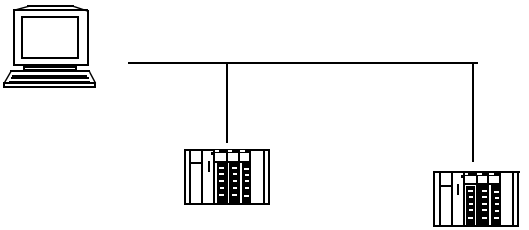
- Four OLE Automation client examples, complete with source code and README file supplied with the software.
  These examples are:
  - a "minimal" visual C++ 4.2 client: the minimum needed to be able to write an OLE client.
  - a "complete" visual C++ 4.2 client, which makes use of all the available input points.
  - a visual Basic 5.0 client, which makes use of all the available input points.
  - an Excel client.
- A *.h file which defines the value of the error codes generated by the server.
- A TLB interface file for a Visual C++ client.

Accessing the PL7 OLE Automation server

The PL7 OLE Automation server offers two ways to access its utilities.

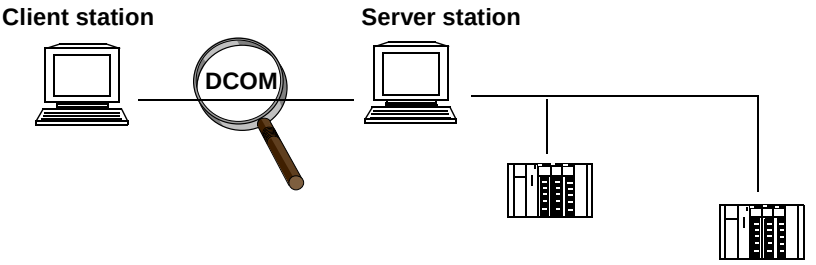
Local access

The client application and the PL7 OLE Automation server are on the same terminal.



Remote access via DCOM

The client application and the PL7 Automation server are on two separate terminals which are linked by the TCP/IP network:



Server start in local mode (COM mode)

Introduction

COM mode (Component object Model) defines an interface for communication between applications. It provides independence from the development tools.

Server start in local mode (COM mode)

To start the server in local mode, follow the procedure below:

1	declare an object " A " on the client application to communicate with the server application
2	Execute a Createdispatch function on object " A " from the client application in order to instantiate the server application
3	Save PL7Pro as OLE server in the register base for correct operation in server mode (the link between the client application and the server application is created if the latter is referenced in the register base).
4	The client application can now interrogate the server application via the interface accessed through object "A". E.g.: OpenStx("C:\appli.stx).

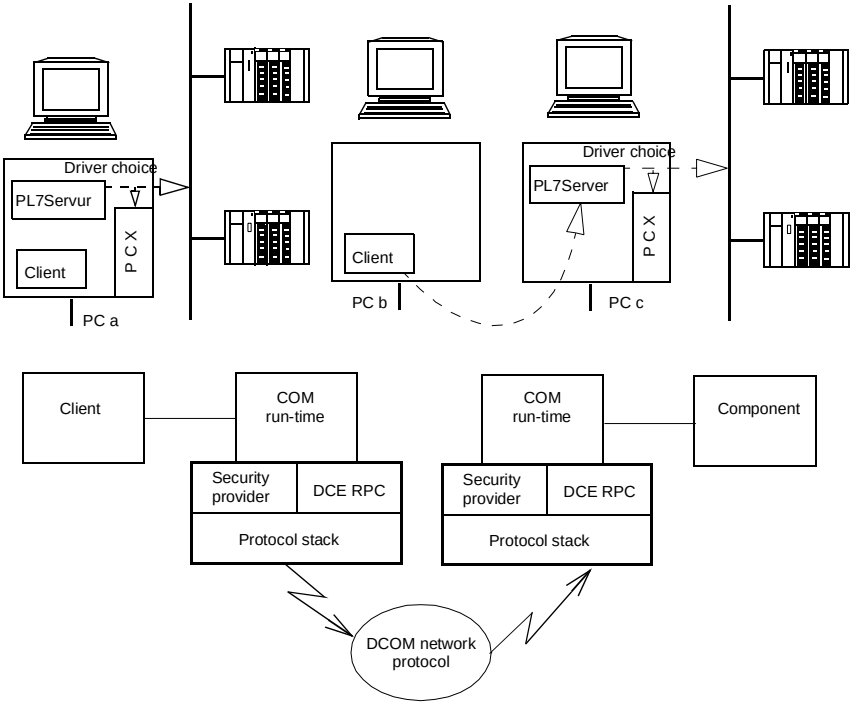
Server start in remote mode (DCOM)

Introduction

The PL7 server runs according to the information present in its register base. It can be located on the client machine or on a remote machine.

DCOM mode is an extension of COM (Component Object Model) mode. COM mode is used for a client application and a server application to communicate with each other on one machine.

DCOM mode is used for two remote machines to communicate with one another. It replaces inter-process communication protocols with network protocols.
Illustration of OLE Automation operation in remote server mode:



Configuration utility

To establish communication between two machines, we use a configuration utility called, "DCOMCNFG.EXE".
By default, the utility "DCOMCNFG.EXE" included in Windows 95. The user must connect to the following Internet address: " http://www.microsoft.com "
and search on this site to download this utility. This allows the server to operate in DCOM when running Windows 95.

Implementing the server in remote mode

Example of use

Let’s suppose a machine A (Windows NT 4) installed with a PL7Pro server.
The machine user must log in as "administrator" in order to manage **access rights**.
The client is machine B (Windows 95).

Procedure

Follow the procedure outlined below:

1	Start DCOMCNFG on machine B:
	● Select the «Default Properties» tab to view the following information:
	  • [X] "Enable Distributed COM on this computer"
	  • Default Authentication Level: CONNECT
	  • Default Impersonation Level: Identify
	● On the «Default Security» tab, check the option:
	  • [X] Enable remote connection
	● On the «Application» tab:
	  • Select PL7Pro server "PL7 server", then "properties"

2	Start DCOMCNFG on machine A:
	● Select the «Default Properties» tab to view the following information:
	  • [X] «Enable Distributed COM on this computer
	  • Default Authentication Level: CONNECT
	  • Default Impersonation Level: Identify

3	Machine B, the "Client" can control **PL7Pro server"** located on machine A.
4	Select the «Default security» tab, and add the users who are to have write authorization on the machine.

For more information

There are Newsgroups on the Internet, where you can ask questions about DCOM.

PL7 Server execution modes

Introduction

The OLE Automation PL7 server has two execution modes which can be chosen dynamically during execution (see `SetHMIServer`)

Execution modes

- Execution mode without HMI. The PL7 is started as a "background task", with no display or possibility of operator input. It is the mode used typically, for example, to automate repetitive tasks or to access application information contained in an stx file.
- Application mode with HMI. The PL7 is started as an "interactive" PL7 with display and the possibility of operator input but it remains receptive to commands from its client application. This mode has been developed to view the program or configuration elements of an application contained in an stx file using external tools such as DIAG Viewer.

Note: Utilities for displaying programs, tools, and modules can only be executed in the mode with HMI.
The OLE Automation PL7 server in HMI mode is subject to monitoring of user rights. It is set in "Read Only" user profile, which corresponds to its code display and PL7 I/O module display role.
When not in HMI mode the OLE Automation PL7 server is not subject to monitoring of user rights but the application cannot be modified with the utilities provided.

Input points: OLE Function

Introduction

There are four types of input points:

- **Execution context**
- **API monitoring**
- **Read information**
- **Application element display**

Execution context

Input points

Name	Description
OpenStx	Opening an application
SaveStx	Saving the active application
CloseStx	Closing the active application
Set DriverAndAddress	Setting the driver and the address of the accessed PLC
SetServerHMI	Making the OLE Automation PL7 server interactive or non-interactive
GetPL7HMI	Giving the status: open or closed application, local – online mode , API status
GetMessageError	Reads the error message associated with the error code

API monitoring

Input points

Name	Description
ConnectPLC	Enters online mode
DisconnectPLC	Exits online mode
SenCommandToPLC	Command sent to the PLC (RUN, STOP INIT)
DownloadToPLC	Loading the active application into a PLC
UploadFromPLCM	Copies a PLC application into the active application

Read information

Input points

Name	Description
ExportScyFile	Exports the symbols in the active application as an scy file
ExportFefFile	Exports the active application as an fef file
GetSymbol	Reads the symbol and the comment associated with an address
GetSTXAppIdentity	Reads the general application information contained in an STX file
GetPLCAppIdentity	Reads the general application information contained in a PLC
GetServerVersion	Reads the server version

Application element display

Input points

Name	Description
SetPosPL7Window	Sets the PL7 display characteristics (position and form)
ShowProgram	Opens an editor on a given program module
CloseProgram	Closes a program editor
ShowIOModule	Opens an editor on a given I/O module
CloseIOModule	Closes a given I/O module editor
ShowDFB	Opens an editor on a given DFB code
CloseDFB	Closes a editor on the given DFB code
OpenTool	Opens any MDI tool without context

Note: The input points concerning application element display do not work if the server is not in HMI mode.

5.3 OLE Functions

Introduction

Subject of this sub-section

This sub-section describes the OLE functions of the OLE Automation server

What's in this Section?

This Section contains the following Maps:

OpenStx

General

This function is used to open the stx file application type.

Description

Syntax:

integer **OpenStx**(String lpAppPathName)

- **Input:**
  Enter the string of characters containing the name of the file that needs to be opened.
- **Function called:**
  `OpenStation`: Station management function (gesta.dll).
- **Nominal context**:
  The PL7 server is active.
- **When there is an error:**

SRV_ERR_GEN_FILENOTFOUND	The file has not been found. Open Station
SRV_ERR_OPEN_BADZIP	There is a problem with opening the file
SRV_ERR_OPEN_BADFILE	There is a problem reading the stx file
SRV_ERR_OPEN_NOK_COMPATIBLE	The processor is not compatible with the PL7 being open.
SRV_ERR_OPEN_OPEN	An application is already open
SRV_ERR_GEN_PARAM_EMPTY	The Path Name is empty
SRV_ERR_GEN_ACTION	There is an error on opening the application
SRV_ERR_GEN_DRIVE FULL	There is no more free disk space for opening the application

- **Output:**
  `Short` type feedback code. Either 0 on a successful opening, or error code.

CloseStx

General

This function is used to close the current application

Description

Syntax:

integer **CloseStx**(integer p_bWithoutSave)

- **Input:**
  If an application has been modified, it can be closed without having to notify the user (`p_bWithoutSave a TRUE`).
- **Function called:**
  `CloseStation`: Station management function (gesta.dll).
- **Nominal context**:
  The PL7 server is active and an application is open.
- **When there is an error:**

SRV_ERR_GEN_NOTOPEN	No application is open.
SRV_ERR_CLOSE_NOTSAVE	The application has been modified, and changes must be saved before closing.

- **Output:**
  `Short` type feedback code. Either 0 on a successful closing, or error code.

ExportScyFile

General

This function is used to export symbols in the enabled application under the scy file format.

Description

Syntax:

integer **ExportScyFile**(String p_psScyFile)

- **Input:**
  Enter the character string containing the scy file name.
- **Function called:**
  `ExportScyFile`: Station management function (gesta.dll).
- **Nominal context**:
  The PL7 server is active. An application is open.
- **When there is an error:**

SRV_ERR_GEN_ACTION	Error on creation of sourcesymbol file.
SRV_ERR_GEN_NOTOPEN	No application is open.
SRV_ERR_GEN_PARAM_EMPTY	`p_ psScyFile` is empty.
SRV_EXPORT_ERRFILE	A write error has occurred in the source file.

- **Output:**
  `Short` type feedback code. Is either 0 on a successful export, or error code.

ExportFeFile

General

This function is used to export the active application under the fef file format.

Description

Syntax:

integer **ExportFefFile**(String p_psNamefile)

- **Input:**
  Enter the character string containing the fef file name.
- **Function called:**
  `ExportFefFile`: Station management function (gesta.dll).
- **Nominal context**:
  The PL7 server is active. An application is open.
- **When there is an error:**

SRV_ERR_GEN_ACTION	Error on creation of source application file.
SRV_ERR_GEN_PARAM_EMPTY	No application is open.
SRV_ERR_GEN_PARAM_EMPTY	`p_ psNamefile` is empty.
SRV_EXPORT_ERRFILE	A write error has occurred in the source file.

- **Output:**
  `Short` type feedback code. Is either 0 on a successful export, or error code.

DisconnectPLC

General

This function is used to carry out disconnection between the PLC and the PL7.

Description

Syntax:

integer **ExportFefFile**(String p_psNamefile)

- **Function called:**
  `DisconnectStation`: Station management function (gesta.dll).
- **Nominal context**:
  The PL7 server is active.
- **Output:**
  `Short` feedback code type. Either shows 0 on a successful disconnection, or error code (SRV_ERR_GEN_ACTION).

ConnectPLC

General

This function is used to connect up to a PLC.

Description

Syntax:

integer **ConnectPLC**(String p_lpDriver, String p_lpAddress)

- **Enter:**
  Enter the two character strings containing the driver and the PLC address.
- **Function called:**
  `ConnectStation`: Station management function (gesta.dll).
- **Nominal context:**
  The PL7 server is active. Wherever parameters consist of empty strings, the server carries out the function using the driver and/or the current address.
- **When there is an error:**

SRV_ERR_GEN_CONNECT	Connection was unsuccessful.
SRV_ERR_GEN_COMMUNICATION	There is a communication problem between the PL7pro and the PLC
SRV_ERR_GEN_RESERVED	The PLC has been reserved, and therefore cannot be connected to.
SRV_ERR_PLC_BLANK	The PLC does not contain any applications, and therefore connection cannot take place.
SRV_ERR_PC_BLANK	No applications are open, and therefore connection cannot take place.
SRV_ERR_DIFFERENCE_PLC_STX	The application which is open and the application contained in the PLC are different, and it is therefore impossible for connection to take place.
SRV_ERR_GEN_ADDRESS	The parameter address is invalid.
SRV_ERR_GEN_ACTION	An error has occurred on carrying out the function.
SRV_ERR_GEN_DRIVER	The parameter driver is invalid.
SRV_ERR_GEN_NOTOPEN	No application is open.
SRV_ERR_GEN_PROTECTEDAPPLI	Protected application.
SRV_COMPATIBLE_PLC	There is a compatibility problem with the PLC.

- **Output:**
  `Short` type feedback code. Either 0 on a successful connection, or error code.

SaveStx

General

This function is used to save the application which is open.

Description

Syntax:

integer **SaveStx**(String p_lpStxFile)

- **Enter:**
  Enter a character string containing the path and the name of the file to be saved.
- **Function called:**
  `SaveStx`: Station management function (gesta.dll).
- **Nominal context:**
  The PL7 server is active. An application is open and PL7 is in offline mode.
- **When there is an error:**

SRV_ERR_SAVEDENIED	Access to the file has been refused
SRV_ERR_SAVEERRZIP	An error has occurred on compacting the application
SRV_ERR_SAVEERRREN	Renaming has not been allowed
SRV_ERR_GEN_NOTOPEN	No application is open
SRV_ERR_GEN_PARAM_EMPTY	`p_ lpStxFile` is empty.
SRV_ERR_GEN_ACTION	There is an error on saving the application
SRV_ERR_PLC_CONNECT	Saving can only be carried out in offline mode.

- **Output:**
  `Short` type feedback code. Either 0 on a successful save, or error code.

DownloadToPLC

General

This function is used to download an application into the PLC memory.

Description

Syntax:

integer **DownloadToPLC**(String p_lpDriver, String p_lpAdresse)

- **Enter:**
  Enter the two character strings containing the driver and the PLC address.
- **Function called:**
  `DownloadStation`: Station management function (gesta.dll).
- **Nominal context:**
  The PL7 server is active. An application is open and PL7 is in offline mode. Wherever parameters consist of empty strings, the server carries out the function using the driver and/or the current address.
- **When there is an error:**

SRV_ERR_DOWNLOAD_CART	There is a memory cartridge compatibility problem between the application which is open and the PLC.
SRV_ERR_COMPATIBLE_PLC	There is a compatibility problem between the PLC and the application which is open.
SRV_ERR_GEN_ADDRESS	The `p_lpAdresse` parameter is invalid.
SRV_ERR_GEN_DRIVER	The `p_lpDriver` parameter is invalid.
SRV_ERR_GEN_NOTOPEN	No application is open.
SRV_ERR_GEN_RESERVED	The PLC has been reserved.
SRV_ERR_GEN_COMMUNICATION	Communication problem.
SRV_ERR_PLC_CONNECT	The PLC has already been connected.
SRV_ERR_PLC_ACTION	An error has occurred while downloading.

- **Output:**
  `Short` type feedback code. Either 0 on a successful download, or error code.

UploadFromPLC

General

This function is used to copy an application in a PLC over to the memory.

Description

Syntax:

integer **UploadFromPLC**(String p_lpDriver, String p_lpAdresse, integer p_iReservedMode)

- **Input:**
  Enter the two character strings containing the driver and the PLC address. An integer (`p_iReservedMode`) which can be used to upload onto a reserved PLC.
- **Function called:**
  `UploadStation`: Station management function (gesta.dll).
- **Nominal context:**
  The PL7 server is active. There are no applications open and the PL7 is in offline mode. If parameters consist of empty strings, the server carries out the function using the driver and/or the current address.
- **When there is an error:**

SRV_ERR_UPLOAD_TRANSFER	There is a problem in transferring the application contained in the PLC
SRV_ERR_UPLOAD_ABORT	Transfer is put on hold.
SRV_ERR_UPLOAD_RESERVED	The application has been reserved.
SRV_ERR_UPLOAD_NOAPPLI	The PLC is empty.
SRV_ERR_GEN_ADDRESS	Invalid address.
SRV_ERR_GEN_ACTION	An error has occurred in carrying out the function.
SRV_ERR_PLC_CONNECT	There is a connection error.
SRV_ERR_GEN_DRIVER	Invalid driver.
SRV_ERR_GEN_COMMUNICATION	There is a communication error.
SRV_ERR_PLC_CONNECT	The PLC has already been connected.
SRV_OPEN_NOT_COMPATIBLE	Compatibility problem.

- **Output:**
  `Short` type feedback code. Either 0 on a successful upload, or error code.

GetSymbol

General

This function is used to give the symbol and comment associated with an address.

Description

Summary:

integer **GetSymbol**(String p_lpRepere, String* p_bsSymbole, String* p_bsComment)

- **Input:**
  Enter the character string containing the address to be modified or completed.
- **Function called:**
  `GetSymbol`: Station management function (gesta.dll).
- **Nominal context:**
  The PL7 server is active. An application is open and in offline mode.
- **When there is an error:**

SRV_ERR_ADDRESSSYNTAX	the character string entered as a parameter does not match an address.
SRV_ERR_GEN_NOTOPEN	No application is open.
SRV_ERR_GEN_PARAM_EMPTY	`p_lpRepere` is empty.
SRV_ERR_GEN_ACTION	Error has occurred on carrying out the function.
SRV_ERR_GEN_ADDRESS	Invalid address.
SRV_ERR_GEN_ACTION	Error has occurred in carrying out the function.

- **Output:**
  `Short` type feedback code. Either 0 on a successful upload, or error code.
  Two character strings:
  - The associated symbol (`p_bsSymbol`).
  - The associated comment (`p_bsComment`)

SetServerHMI

General

This function can be used to display (or not display) the PL7 server. It switches from one operating mode to another. i.e. switch from a PL7 Pro server without HMI to a PL7 Pro server with HMI, and vice versa.

Description

Syntax:

integer **SetServerHMI** (integer p_bHMI)

- **Input:**
  A Boolean (HMI display or non-display).
- **Function called:**
  `SetIHMServer`: Windows application function (sawinapp.cpp).
- **Nominal context:**
  The PL7 server is active.
- **When there is an error:**

SRV_ERR_GEN_ACTION	Error has occurred on carrying out the function.
SRV_ERR_GEN_MODIFAPPLI	Application is in the process of being modified.

- **Output:**
  `Short` type feedback code. Either 0 on a successful upload, or error code.

GetPL7State

General

This function is used to obtain the state of the server.

Description

Syntax:

integer **GetPL7State**(String* p_lpStation, String * p_lpConnection)

- **Function called:**
  `GetPL7State`: Station management function (gesta.dll).
- **Nominal context:**
  The PL7 server is active.
- **When there is an error:**

SRV_ERR_GEN_ACTION	An error has occurred on carrying out the function.

- **Output:**
  `Short` feedback code type. Either 0 on a successful upload, or error code.
  Two character strings inform the user on:
  - The state of the station (open or closed).
  - The mode state (on- or offline). In online mode, the state of the PLC (run or stop) is indicated.

GetSTXAppIdentity

General

This function is used to find out general information about an application.

Description

Syntax:

integer **GetSTXAppIdentity**(String p_lpNameStx, VARIANT FAR* p_pVarInfo)

- **Enter:**
  Enter the character string (`p_1pNameStx`) containing the name of the application.
- **Function called:**
  `IdentAppliForm`: Station management function (gesta.dll).
- **Nominal context:**
  The PL7 server is active.
- **When there is an error:**

SRV_ERR_GEN_PARAM_EMPTY	`p_ lpNameStx` is empty.
SRV_ERR_GEN_ACTION	An error has occurred on carrying out the function.
SRV_ERR_APPLIINFO_NOK	Invalid information

- **Output:**
  `Short` feedback code type. Either 0 on a successful upload, or error code.

Displayed information:

0	Station name,
1	Application version number,
2	Date and time of modification
3	Station application ID,
4	ID concerning local inputs/outputs configuration,
5	ID concerning remore inputs/outputs configuration,
6	PL7 code ID
7	Grafcet ID,
8	Constants ID,
9	Symbol ID,
10	ID for reservations,
11	The comment associated with the application.

GetPLCAppIdentity

General

This function is used to find out general information about an application in a PLC.

Description

Syntax:

integer **GetPLCAppIdentity**(String p_lpDriver, String p_lpAdresse, VARIANTFAR* p_pvInfoAppli)

- **Enter:**
  Enter the two character strings containing the driver and the PLC address.
- **Function called:**
  `IdentAppliOnPlcForm`: Station management function (gesta.dll).
- **Nominal context:**
  The PL7 sever is enabled and in offline mode.
- **When there is an error:**

SRV_ERR_APPLIINFO_NOK	Invalid information
SRV_ERR_GEN_ACTION	An error has occurred on carrying out the function.
SRV_ERR_GEN_ADDRESS	Invalid address.
SRV_ERR_COMPATIBLE_PLC	Compatibility problem between the API and PL7 software.
SRV_ERR_PLC_BLANK	There are no applications in the PLC.
SRV_ERR_GEN_ACTION	A problem has occurred while storing application information.
SRV_ERR_GEN_DRIVER	The `p_lpDriver` parameter is invalid.
SRV_ERR_GEN_COMMUNICATION	There is a communication error.

- **Output:**
  `Short` feedback code type. Either 0 on a successful upload, or there is a code error.

Displayed information:

0	Station name,
1	Application version number,
2	Date and time of modification
3	Station application ID,
4	ID concerning local inputs/outputs configuration,
5	ID concerning remote inputs/outputs configuration,
6	PL7 code ID

7	Grafcet ID,
8	Constants ID,
9	Symbol ID,
10	ID for reservations,
11	The comment associated with the application.

SendCommandToPLC

General

This function is used to send a command to the PLC.

Description

Syntax:

integer **SendCommandToPLC**(integer p_iCommand)

- **Enter:**
  The type of command that is to be executed
  There are 3 possible types:
  - SRV_COMMAND_INIT: Command to initialize PLC
  - SRV_COMMAND_STOP: Command to stop PLC
  - SRV_COMMAND_RUN: Command to run PLC
- **Function called:**
  `StationCommand`: Station management function (gesta.dll).
- **Nominal context:**
  The PL7 server is active. An application is open, and PL7 is online and in STOP mode.
- **When there is an error:**

SRV_ERR_GEN_ACTION	Invalid command.
SRV_ERR_GEN_NOTOPEN	There are no applications open.
SRV_ERR_COMMAND_NOTINLOCAL	The server is in offline mode.
SRV_COMMAND_ERRINIT	INIT function cannot be carried out as PLC is already in RUN mode.

- **Output:**
  `Short` feedback code type. Either 0 on a successful upload, or there is a code error.

SetDriverAndAddress

General

This function is used to change the driver and the address for the current station.

Description

Syntax:

integer **SetDriverAndAddress**(String p_lpDriver, String p_lpAddress)

- **Enter:**
  Enter the two character strings containing the driver and the PLC address.
- **Function called:**
  `DriverAndAddress`: Station management function (gesta.dll).
- **Nominal context:**
  The PL7 server is active. An application is open and in offline mode. If parameters are not fully completed or are invalid, their current values remain unchanged.
- **When there is an error:**

SRV_ERR_GEN_ADDRESS	The address put into the parameter is invalid.
SRV_ERR_GEN_DRIVER	The driver in the parameter is invalid.
SRV_ERR_PLC_CONNECT	The server is in online mode.
SRV_ERR_PLC_ACTION	An error has occurred on carrying out the function.
SRV_ERR_PARAM_EMPTY	The parameters are empty.

- **Output:**
  `Short` feedback code type. Either 0 on a successful upload, or there is a code error.

OpenTool

General

This function is used to open the tools present in the character string placed in the parameters.

Description

Syntax:

integer **OpenTool**(String p_lpListTool)

- **Enter:**
  The character strings containing the list of tools that the server must open.
  Formatting example: tools1;tools2;tools3
- **Function called:**
  `OpenTool`: Station management function (gesta.dll).
- **Nominal context:**
  The PL7 server with HMI is enabled. An application is open.
- **When there is an error:**

SRV_ERR_GEN_ACTION	A tool cannot be opened.
SRV_ERR_GEN_PARAM_EMPTY	Empty parameter.
SRV_ERR_GEN_NOTOPEN	No application is open.
SRV_ERR_GEN_WITHIHM	Function available using PL7 server in HMI mode.
SRV_ERR_GEN_OPENEDTVDATA	The "Animation Table" tool cannot be opened.

- **Output:**
  `Short` feedback code type.

SetPosPL7Windows

General

This function is used to set the position of the PL7 window.

Description

Syntax:

integer **PosPL7Windows**(integer CoordX, integer CoordY, integer CoordCX,integer CoordCY)

- **Enter:**
  Th X-Y coordinates for the window.
- **Function called:**
  `MoveWindow`: mfc. function
- **Nominal context:**
  The PL7 server with HMI is enabled.
- **When there is an error:**

SRV_ERR_GEN_ACTION	Command failed.
SRV_ERR_GEN_PARAM_EMPTY	Empty parameter.
SRV_ERR_GEN_WITHIHM	Function available using PL7 server in HMI mode.

- **Output:**
  `Short` feedback code type.

ShowProgram

General

This function is used to open the programs present in the parameter string placed in the parameters.

Description

Syntax:

integer **ShowProgram**(String p_lpListProgram)

- **Enter:**
  Enter the character strings containing the list of programs that the server must open.
  Formatting example: mast\lad1;mast\lit2;evt\evt0;sr1;mast\prl
- **Function called:**
  `ShowProgram`: Station management function (gesta.dll).
- **Nominal context:**
  The PL7 server with HMI is enabled.
  An application is open.
- **When there is an error:**

SRV_ERR_GEN_ACTION	Command failed.
SRV_ERR_GEN_NOTOPEN	There are no applications open.
SRV_ERR_GEN_PARAM_EMPTY	Empty parameter.
SRV_ERR_GEN_WITHIHM	Function available using PL7 server in HMI mode.
SRV_ERR_GEN_SHOWPROGRAM	The module cannot be opened.

- **Output:**
  `Short` feedback code type. Either 0 on a successful upload, or there is a code error.
- **Limitations:**
  This function is not able to open a non-instanced DFB. To display a non-instanced DFB code, the `ShowDFB` function must be used.

CloseProgram

General

This function is used to close the programs present in the character string placed in the parameters.

Description

Syntax:

integer **CloseProgram**(String p_lpListProgram)

- **Enter:**
  The character strings containing the list of programs that the server must close.
  Formatting example: mast\lad1;mast\lit2;evt\evt0;sr1;mast\prl.
- **Function called:**
  `CloseProgram`: Station management function (gesta.dll).
- **Nominal context:**
  The PL7 server with HMI is enabled.
  An application is open.
- **When there is an error:**

SRV_ERR_GEN_NOTOPEN	There are no applications open.
SRV_ERR_GEN_ACTION	Close failed.
SRV_ERR_GEN_PARAM_EMPTY	Empty parameter.
SRV_ERR_GEN_WITHIHM	Function available using PL7 server in HMI mode.
SRV_ERR_GEN_CLOSEPROGRAMM	The module cannot be closed.

- **Output:**
  `Short` feedback code type. Either 0 on a successful upload, or there is a code error.
- **Limitations:**
  This function cannot close a DFB which is open.

ShowIOModule

General

This function is used to open the input/output modules present in the character string placed in the parameters.

Description

- **Enter:**
  The character strings containing the list of modules that the server must open.
  The formatting for the character string is `rack,module; rack,module`
  Formatting example: 0,0;0,1;1,2
- **Function called:**
  `ShowIOModule`: Station management function (gesta.dll).
- **Nominal context:**
  The PL7 server with HMI is enabled.
  An application is open.
- **When there is an error:**

SRV_ERR_GEN_NOTOPEN	There are no applications open.
SRV_ACTION_NOK	There is a problem with enabling user rights.
SRV_ERR_GEN_PARAM_EMPTY	Empty parameter.
SRV_ERR_GEN_WITHIHM	Function available using PL7 server in HMI mode.

- **Output:**
  `Short` feedback code type. Either 0 on a successful upload, or there is a code error.
- **Limitations:**
  An IO module is opened in offline mode, while the module configuration screen is open. If the user changes from offline to online mode, the server does not switch from the configuration screen to the debugging screen. The user can either operate the PL7 Pro directly or open/close the module using a client.

CloseIOModule

General

This function is used to close the input/output modules present in the character string placed in the parameters.

Description

Syntax

CloseIOModule(String p_lpListIOModule) integer.

- **Enter:**
  The character strings containing the list of modules that the server must open.
  formatting example: 0,0;0,1;1,2
- **Function called:**
  `CloseIOModule`: Station management function (gesta.dll).
- **Nominal context:**
  The PL7 server with HMI is enabled.
  An application is open.
- **When there is an error:**

SRV_ERR_GEN_ACTION	Close command failed.
SRV_ERR_GEN_PARAM_EMPTY	Empty parameter.
SRV_ERR_GEN_NOTOPEN	There are no applications open.
SRV_ERR_GEN_WITHIHM	Function available using PL7 server in HMI mode.

- **Output:**
  `Short` feedback code type. Either 0 on a successful upload, or there is a code error.
- **Limitations:**
  This function does not close the configuration editor which is open using the `ShowIOModule` function.

ShowDFB

General

This function is used to display the code for one or many DFBs present in the character string placed in the parameters. Opening a DFB using this application does not require instancing.

Description

Syntax

integer **ShowDFB**(String p_lpListeDFB)

- **Enter:**
  The character strings containing the list of DFBs that the server must open.
- **Function called:**
  ShowDFB: Station management function (gesta.dll).
- **Nominal context:**
  The PL7 server with HMI is enabled.
  An application is open.
- **When there is an error:**

SRV_ERR_GEN_PARAM_EMPTY	Empty parameter
SRV_ERR_GEN_NOTOPEN	There are no applications open
SRV_ERR_GEN_ACTION	A problem has occurred while opening a DBF.
SRV_ERR_GEN_WITHIHM	Function available using PL7 server in HMI mode.

- **Output:**
  Short feedback code type. Either 0 on a successful upload, or there is a code error.

CloseDFB

General

This function is used to close the one or several DFBs present in the character string placed in the parameters.

Description

Syntax

integer **CloseDFB**(String p_lpListeDFB)

- **Enter:**
  The character strings containing the list of DFBs that the server must close.
- **Function called:**
  `CloseDFB`: Station management function (gesta.dll).
- **Nominal context:**
  The PL7 server with HMI is enabled.
  An application is open.
- **When there is an error:**

SRV_ERR_GEN_PARAM_EMPTY	Empty parameter
SRV_ERR_GEN_NOTOPEN	There are no applications open
SRV_ERR_GEN_ACTION	A problem has occurred while opening a DBF.
SRV_ERR_GEN_WITHIHM	Function available using PL7 server in HMI mode.

- **Output:**
  `Short` feedback code type. Either 0 on a successful upload, or there is a code error.

GetMessageError

General

This function is used to associate an error message with and according to the error code placed in the parameters.

Description

Syntax

String **GetMessageError**(integer p_iCodeError)

- **Enter:**
  The error code for one of the departments to return.
- **Nominal context:**
  The PL7 server is active.
- **Output:**
  A character string corresponding to the error label.

GetServerVersion

General

This function is used to find out the version number for the PL7 Pro server.

Description

Syntax

String **GetServerVersion**()

- **Nominal context:**
  The PL7 server is active.
- **Output:**
  A character string corresponding to the version number label for the PL7 Pro server.

Instruction times

6

At a Glance

Aim of this chapter

This chapter describes the PL7 language instruction times. Thus it can be used to calculate the execution time for an application and the memory size occupied.

What's in this Chapter?

This Chapter contains the following Sections:

6.1 General information

Calculation principles

Application program execution time

The execution time of the program can be calculated using the tables on the following pages to find the sum of all program instruction times.

Note: The time calculated is the maximum time.Operate blocks and subroutines are only processed if the condition of execution (logical equation determining the execution of a block or subroutine) is true, therefore it is possible that the actual time will be much shorter than the maximum time calculated.

Calculating the complete cycle time incorporates parameters which are specific to the PLC (overhead time, I/O Exchange time etc.). For the complete calculation procedure refer to the appropriate PLC operation manual (chapter on performance).

Application memory size

The size of the application is equal to the sum of the following elements:

Element	Calculation method
Program	Calculate the sum of each program instruction, and multiply by the appropriate coefficient corresponding to the language in use (see next page)
Advanced functions	See *Memory size for advanced functions, p. 195*
Configured PL7 objects	See *Object memory size, p. 185*
Configured input/output module	See *Review of the memory usage of the modules on MIcro, p. 186* and *Memory usage for the modules on Premium, p. 189*
Comments	The program comments occupy 1 byte per character.

In the tables on the following pages, the information on sizes refers to the instruction code size.

To find out the total size of an instruction or a program, use a multiplier coefficient that takes into account information typical of each language.

Language	**Size**
Ladder Language	Total size = 1.7 x Code size
Structured Text	Total size = 1.6 x Code size
Instruction List	For the Micro PLC: Total size = 1.4 x Code size
	For the Premium PLC: Total size = 1.6 x Code size
Grafcet	Chart size (in words) = 214 = 17 * number of chart steps = 2 * total number of configured steps + 4 * number of programmed actions

Note: The figures given in the following tables are average estimates, calculated using one application type. It is not possible to provide exact figures, as PL7 optimizes memory use according to the structure and contents of the application.

The chapter *Description of the memory zones, p. 184* gives a reminder of the various memory areas taken up by the application.

6.2 Instruction times on Micro PLCs

At a Glance

Aim of this section

This section describes the time taken by instructions executed on Micro PLCs.

What's in this Section?

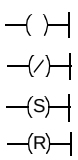
This Section contains the following Maps:

Boolean instruction performance characteristics

Performance characteristics

The table below shows the performance characteristics of the boolean instructions.

LD	IL	ST	Objects	Execution time (µs)			Size In words
				37 05/ 08/10	37 20 RAM	37 20 card	
				0,25	0,13	0,19	1
	LD LDN		%M1 (1)	0,25	0,13	0,19	1
			%M1[%MW2]	13.10	12.85	12.85	7
			%MW0:X0 (2)	6.06	5.75	5.75	4
			%IWi.j:Xk (3)	77.04	69.25	69.25	8
			%MW0[%MW8]:X0	16.29	15.55	15.55	8
			%KW0[%MW8]:X0	87.27	79.05	79.05	12
	LDR, LDF		%M1	0.50	0.25	0.38	2
			%M1[%MW2]	13.01	12.75	12.75	7
	AND, ANDN , AND (, AND (N , idem OR			idem LD,LDN			
	ANDR, ANDF, AND (R, AND (F, idem OR			idem LDR,LDF			
	XOR XORN		%M1	1.25	0.63	0.94	5
			%M1[%MW2]	26.94	26.08	26.26	13
			%MW0:X0	12.86	11.88	12.06	10
			%IWi.j:Xk	83.84	75.38	75.56	14
			%MW0[%MW8]:X0	33.33	31.48	31.66	14
			%KW0[%MW8]:X0	104.31	94.98	95.16	18
	XORR, XORF		%M1	2.25	1.13	1.69	9
			%M1[%MW2]	27.28	26.13	26.44	19

LD	IL	ST	Objects	Execution time (µs)			Size In words
				37 05/ 08/10	37 20 RAM	37 20 card	
	ST STN S R		%M1 (1)	0.50	0.25	0.38	2
			%M1[%MW2]	13.10	12.85	12.85	7
			%MW0:X0	5.88	5.60	5.60	4
			%NW{i}.j:Xk (3)	76.86	69.10	69.10	8
			%MW0[%MW8]:X0	16.41	15.65	15.65	8
Multiple coils on ladder, «cost» of the second and subsequent coils				0.25	0.13	0.19	1
Operate block	[action]		block executed	0.74	0.75	0.75	1
			Not executed	5.55	5.40	5.40	1
Horizontal comparison block	[LD [comparison]]		Additional comparison time	0.00	0.00	0.00	0
Vertical operate block			between 2 %MWi	12.38	11.85	11.85	4
Convergence	[ )]	[ )]	block executed	0.25	0.13	0.19	1
Untracked divergence of a convergence	[ ]		Ladder, 1 divergence	0.25	0.13	0.19	1
	[MPS, MPP, MRD]		List MPS+MPP	0.75	0.38	0.56	3
			List MRD	0.25	0.13	0.19	1

Legend

1. This applies to all forcible object bits: %I, %Q, %X, %M, %S,
2. other objects of the same kind: Output bits of function block %Tmi.Q …, Extract bits of system words %SWi:Xj
3. other objects of the same kind: Extract bits of shared words %NW{i}j:Xk, extract bits of I/O %IWi.j.Xk, %QWi.j.Xk words, extract bits of %KW, error bits %Ii.j.ERR.

Performance characteristics of function blocks

IEC timer

The table below describes the performance times of the IEC timer function block

LD	IL	ST	Conditions	Execution time (µs)		Size (words)
				3705/08/10	3720	
Rising edge on IN	IN %TM1 (rising edge)	START %TM1	Start timer	43.39	41.11	3
Falling edge on IN	IN %TM1 (falling edge)	DOWN %TM1	Stop timer	17.47	17.01	
IN =1	IN %TM1 (=1)		timer active	18.74	17.99	
IN =0	IN %TM1 (=0)		timer inactive	17.40	16.67	

PL7-3 timer

The table below describes the performance times of the PL7-3 timer function block

LD	IL	ST	Conditions	Execution time (µs)		Size (words)
				3705/08/10	3720	
		START %T1	start			3
		STOP %T1	stop	12.63	12.15	
I =0		RESET %T1	Reset	12.94	12.15	
			timer active	17.55	17.00	
			timer inactive			

Upcounter/ Downcounter

The table below describes the performance times of the upcounter/downcounter function block

LD	IL	ST	Conditions	Execution time (µs)		Size (words)
				3705/08/10	3720	
Reset, R=1	R %C8 (=1)	RESET %C8	Reset	18.69	17.92	3
Preset, S=1	S %C9 (=1)	PRESET %C9	Preset	20.42	19.73	
Rising edge on CU	CU %C8 (edge	UP %C8	Up	19.92	19.10	
Rising edge on CD	CD %C9 (edge	DOWN %C9	Down	19.92	19.10	
Inactive inputs	R/S/CU/CD inactive bit		No action	13.27	12,81	

Monostable

The table below describes the performance characteristics of the monostable function block

LD	IL	ST	Conditions	Execution time (µs)		Size (words)
				3705/08/10	3720	
Rising edge on S	S %MN0, rising edge	START %MN0	Start	35,08	33.16	3
O=1	O %MN0, O =1/0		Active monostable	11.64	11.17	

Register

The table below describes the performance times of the register function block

LD	IL	ST	Conditions	Execution time (µs)		Size (words)
Edge on I	I %R2 (edge	PUT %R2	Store	21.90	21.27	3
Edge on O	O %R2 (edge)	GET %R2	Remove from store	21.90	21.27	
R=1	R %R1 (=1)	RESET %R2	Reset	16.90	16.02	
Inactive inputs	I/O/R, bit inactive		no action	12.61	12.19	

Drum

The table below describes the performance times of the drum

LD	IL	ST	Conditions	Execution time (µs)		Size (words)
edge on U	U %DR0	UP %DR1	up, fixed	181.37	169.13	3
			per command bit	19.30	19.30	
R=1	R %DR1	RESET %DR2	reset, fixed	174.15	162.03	
			per command bit	19.30	19.30	
Inputs inactive	R/U, bit inactive		no action, fixed	175.92	164.00	
			per command bit	19.30	19.30	

Integer and floating point arithmetic

Corrections according to object type

The following time and size pages are given for objects of type %MW0, %MD0 or %MF0.
The table below gives corrections which should be applied to the values given in the arithmetic instruction times table if different types of objects are used.

Type of Object	Object position	Type of correction	Execution time (µs)		Size in words
			3705/08/10	3720	
Single length immediate value	-	Value to be subtracted from value mentioned for %MW	1.20	1.10	0
Double length immediate value	-	Value to be subtracted from value mentioned for %MD or %MF	0.75	1	0
Words, double Words or indexed floating points	After the := sign	Value to be added	10.52	10.05	4
	First operation, when the first operand is not indexed, or assignment	Value to be added	11.20	10.60	5
	Second operand if the first operand is equally indexed	Value to be added	13.37	12.60	5
%KWi, %KWi[%MWj] %KDi, %KFi shared words, input/output words	-	Value to be added	70.98	63.50	2

Correction according to the context of the operation

This table describes the corrections to be made to the values given in the arithmetic instruction times table according to the context of the operation.

Context of the operation	Type of Object	Type of correction	Execution time (µs)		Size
			3705/08/10	3720	37xx
The operation is in at least the second position in the statement E.g: %MW2 in:=%MW0*%MW1*%MW2	%MW	Value to be added to value mentioned for %MW	0.69	0.55	0
	%MD or %MF	Value to be added to value mentioned for %MD or %MF	0.99	0.75	0
Operation with result of an operation in parentheses or more urgent E.g: %MW0+%MW2+(...)	%MW	Value to be added to value mentioned for %MW	2.86	2.55	1
	%MD or %MF	Value to be added to value mentioned for %MD or %MF	3.60	3.15	1

Table of instruction times

The table below shows the arithmetic instruction times.

ST	Objects	Conditions	Execution time (µs)		Size (words) 37xx
			3705/08/10	3720	
object after the := sign	%MW0		4.81	4.50	2
	%MD0,%MF0		6.45	5.70	2
:=	%MW0		4.46	4.30	2
	%MD0 and %MF0		5.15	4.85	2
=, <>, <=, <, >, >=	%MW0		8.94	8.50	4
	%MD0		10.71	10.26	4
	%MF0		29.06	28.39	4
AND, OR, XOR	%MW0		7.29	6.90	3
	%MD0		9.21	8.55	3
+, -	%MW0		7.29	6.90	3
	%MD0		9.21	8.55	3
	%MF0		62.83	61.20	3

ST	Objects	Conditions	Execution time (µs)		Size (words) 37xx
			3705/08/10	3720	
*	%MW0		9.75	9.10	3
	%MD0		39.63	36.50	3
	%MF0		58.26	56.90	3
/, REM	%MW0		10.69	10.08	3
	%MD0		205.21	201.38	3
/	%MF0		62.47	60.25	3
ABS, -object	%MW0		7.20	6.95	3
	%MD0		9.97	9.53	3
	%MF0		13.01	12.50	3
NOT	%MW0		6.69	6.45	3
	%MD0		7.80	7.40	3
SQRT	%MW0		17.02	16.70	3
	%MD0		85.73	85.25	3
	%MF0		165.04	158.40	3
INC, DEC	%MW0		4.86	4.40	2
	%MD0		5.20	4.75	2
SHL, SHR, ROL, ROR	%MW0	for 1 bit	17.74	17.05	5
	%MD0	for 1 bit	20.58	19.15	5
		per supplementary bit	0.063		
LN	%MF0		1371.60	1270.00	3
LOG	%MF0		1458.00	1350.00	3
EXP	%MF0		1155.60	1070.00	3
EXPT	%MF0		2988.00	2490.00	3
TRUNC	%MF0		204.00	170.00	3
COS	%MF0		2829.60	2620.00	3
SIN	%MF0		2840.40	2630.00	3
TAN	%MF0		2937.60	2720.00	3
ACOS	%MF0		4082.40	3780.00	3
ASIN	%MF0		4082.40	3780.00	3
ATAN	%MF0		2786.40	2580.00	3
DEG_TO_RAD	%MF0		852.00	710.00	3
RAD_TO_DEG	%MF0		720.00	600.00	3

Instructions on the program and monitoring structures

Instruction time on the program

The table below shows the instruction time on the program.

ST	Execution time (µs)		Size (words) 37xx
	3705/08/10	3720	
Jump %Li	41.93	38.20	3
Maskevt	12.21	10.80	1
Unmaskevt	40.27	37.10	1
SRi	48.68	42.88	3
Return	42.18	38.33	3

Performance of the monitoring structures

The table below shows the instruction times of the monitoring structure type.

ST		Execution time (µs)		Size (words) 37xx
		3705/08/10	3720	
<cond>	condition evaluation			
forcible bit	see LD %M1 boolean instruction			
comparison	see comparisons =,<,> etc.			
if <cond > then <action> end_if;	the times and sizes below should be added to those of the action contained in the structure			
condition true		3.60	3.30	2
condition false (jump)		5.55	5.40	
If <cond> then <action1> else <action2> end_if;				
condition true		9.15	8.70	4
condition false		5.55	5.40	
while <cond> do.<action> end_while				
go into the loop with loop iteration		9.15	8.70	2
exiting the loop		5.55	5.40	
repeat <action> until <cond> end_repeat				
go into the loop with loop iteration		5.55	5.40	2
last entry		3.60	3.30	
for <word1:=word2>to <word3> do <action> end_for				

ST		Execution time (µs)		Size (words) 37xx
		3705/08/10	3720	
input in the for, executed only once		8.58	8.25	15
go into the loop with loop iteration		29.38	27.35	
exit the loop		20.42	19.40	

Digital conversions

Performance times

The table below shows the digital conversion instruction times.

ST	Execution time (µs)			Size (words)
	3705/08/10	3720 RAM	3720 card	37xx
BCD_TO_INT	25.03	24.55	24.55	3
INT_TO_BCD	21.66	21.15	21.15	3
GRAY_TO_INT	36.98	36.55	36.55	3
INT_TO_REAL	40.90	40.75	40.75	3
DINT_TO_REAL	33.32	32.55	32.55	3
REAL_TO_INT	58.75	58.55	58.55	3
REAL_TO_DINT	44.59	44.05	44.05	3
DBCD_TO_DINT	1 324.85	1 065.15	1 134.70	5
DBCD_TO_INT	1 265.54	925.70	986.15	5
DINT_TO_DBCD	1 124.85	825.15	879.10	5
INT_TO_DBCD	564.85	445.15	474.40	5

Instructions on a string of bits

Initializing a table of bits

This table shows the instruction times of a table of bits.

ST	Size (bit)	Execution time (µs)			Size (words) 37xx
		3705/08/10	3720 RAM	3720 card	
%M30:8:= 0	8	19.38	18.88	18.88	6
%M30:16:= 1	16	20.38	19.88	19.88	6
%M30:24:= 2	24	24.25	23.35	23.35	6
%M30:32:= 2	32	25.25	24.35	24.35	6

Copying a table of bits into a table of bits

This table shows the instruction times for copying a table of bits into another table of bits.

ST	Size (bit)	Execution time (µs)			Size (words) 37xx
		3705/08/10	3720 RAM	3720 card	
%M30:8:= %M20:8	8	25.54	24.79	24.79	6
%M30:16:= %M20:16	16	26.16	25.41	25.41	6
%M30:24:= %M20:24	24	33.41	32.26	32.26	6
%M30:32:= %M20:32	32	35.91	34.76	34.76	6
%M30:16:= COPY_BIT(%M20:16)	16	281.63	230.00	244.95	9
	32	440.82	360.00	383.40	9
	128	1261.22	1030.00	1096.95	9

Logic instructions on a table of bits

The table below shows the logic instruction times on a table of bits.

ST	Size (bits)	Execution time (µs)			Size (words) 37xx
		3705/08/10	3720 RAM	3720 card	
AND_ARX, OR_ARX, XOR_ARX					
%M0:16:= AND_ARX(%M30:16,%M50:16)	16	397.42	320.00	340.80	12
%M0:32:= AND_ARX(%M30:32,%M50:32)	32	620.97	500.00	532.50	12
%M0:128:= AND_ARX(%M30:128,%M50:128)	128	1 887.74	1 520.00	1 618.80	12

ST	Size (bits)	Execution time (µs)			Size (words) 37xx
		3705/08/10	3720 RAM	3720 card	
NOT_ARX					
%M0:16:= NOT_ARX(%M30:16)	16	281.63	230.00	244.95	9
	32	440.82	360.00	383.40	9
	128	1261.22	1030.00	1096.95	9

Copying a table of bits into a table of words

The table below shows the instruction times for copying a table of bits into a table of words.

ST	Size (bits)	Execution time (µs)			Size (words) 37xx
		3705/08/10	3720 RAM	3720 card	
%MW1:= %M30:8	8	14.84	14.36	14,36	5
%MW1:= %M30:16	16	16.34	15.86	15.86	5
%MD2:= %M30:24	24	14.54	14.23	14.23	5
%MD2:= %M30:32	32	16.04	15.73	15.73	5
%MW1:4:= BIT_W(%M40:80,0,17,2)	17	501.43	390.00	415.35	16
%MD1:4:= BIT_D(%M30:80,0,33,0)	33	379.53	530.00	564.45	16

Copying a table of words into a table of bits

This table shows the instruction times for copying a table of words into a table of bits.

ST	Size (bits)	Execution time (µs)			Size (words) 37xx
		3705/08/10	3720 RAM	3720 card	
%M30:8:= %MW1	8	19.28	18.68	18.68	5
%M30:16:= %MW2	16	20.28	19.68	19.68	5
%M30:24:= %MD1	24	21.20	20.37	20.37	5
%M30:32:= %MD3	32	22.20	21.37	21.37	5
%M30:32:= W_BIT(%MW200:2,0,2,0)	32	488.68	370.00	394.05	16
%M30:32:= D_BIT(%MD0:1,0,2,0)	32	567.33	460.00	489.90	16

Instructions on tables of words, double words and floating points

Initializing a table of words with one word

The table below shows the instruction times for initializing a table of words with one word.

ST	Size (of the table of words)	Execution time (µs)			Size (words) 37xx
		3705/08/10	3720 RAM	3720 card	
%MW0:10:= %MW100	10 words	47,46	42,15	42,15	7
	per word	0,34	0,20	0,20	
%MD0:10:= %MD100	10 double words	81,27	74,45	74,45	7
	per double word		2,87	2,65	2,65

Copying a table of words into a table of words

This table shows the instruction times for copying a table of words into another table of words.

ST	Size (of the table of words)	Execution time (µs)			Size (words) 37xx
		3705/08/10	3720 RAM	3720 card	
%MW0:10:=%MW20:10;	10 words	95,80	85,35	85,35	9
	per word	0,77	0,50	0,50	
%MD0:10:=%MD20:10;	10 double words	111,13	97,65	97,65	9
	per double word	1,54	1,00	1,00	

Arithmetic and logic instructions between 2 tables of words

The table below shows the arithmetic and logic instruction times between two tables of words.

ST	Size (of the table of words)	Execution time (µs)			Size (words) 37xx
		3705/08/10	3720 RAM	3720 card	
+, -					
%MW0:10:= %MW10:10 + %MW20:10;	10 words	168,04	151,95	151,95	14
	per word	7,13	6,35	6,35	
%MD0:10:= %MD10:10+%MD20:10;	10 double words	239,17	214,40	214,40	14
	per double word	13,84	12,25	12,25	
*					

ST	Size (of the table of words)	Execution time (µs)			Size (words) 37xx
		3705/08/10	3720 RAM	3720 card	
%MW0:10:= %MW10:10 * %MW20:10;	10 words	189,32	175,40	175,40	14
	per word	9,27	8,70	8,70	
%MD0:10:= %MD10:10 * %MD20:10;	10 double words	710,35	603,80	603,80	14
	per double word	61,64	51,20	51,20	
/, REM					
%MW0:10:= %MW10:10 / %MW20:10;	10 words	224,76	181,40	181,40	14
	per word	13,14	9,30	9,30	
%MD0:10:= %MD10:10 / %MD20:10;	10 double words	2 192,38	2 157,35	2 157,35	14
	per double word	209,16	206,55	206,55	
AND, OR, XOR					
%MW0:10:=%MW10:10 AND %MW20:10;	10 words	163,69	147,40	147,40	14
	per word	6,66	5,85	5,85	
%MD0:10:=%MD10:10 AND %MD20:10;	10 double words	240,14	215,90	215,90	14
	per double word	13,94	12,40	12,40	

Arithmetic and logic instructions between 1 table of words and 1 word

The table below shows the arithmetic and logic instruction times between 1 table of words and 1 word.

ST	Size (of the table of words)	Execution time (µs)			Size (words) 37xx
		3705/08/10	3720 RAM	3720 card	
+, -					
%MW0:10:=%MW10:10 + %MW20; or %MW0:10:= %MW20 + %MW10:10	10 words	119,12	108,55	108,55	12
	per word	2,87	2,65	2,65	

ST	Size (of the table of words)	Execution time (µs)			Size (words) 37xx
		3705/08/10	3720 RAM	3720 card	
%MD0:10:=%MD10:10 + %MD20;	10 double words	159,68	147,45	147,45	12
	per double word	6,57	6,25	6,25	
*					
%MW0:10:= %MW20*%MW10:10;	10 words	166,86	132,45	132,45	12
	per word	7,94	5,05	5,05	
%MD0:10:= %MD20*%MD10:10;	10 double words	587,01	522,95	522,95	12
	per double word	49,18	43,80	43,80	
/, REM					
%MW0:10:=%MW10:10 / %MW30;	10 words	196,69	155,85	155,85	15
	per word	10,86	7,30	7,30	
%MD0:10:= MD10:10 / %MD30	10 double words	2 230,17	2 173,95	2 173,95	12
	per double word	213,66	208,90	208,90	
AND, OR, XOR					
%MW0:10:=%MW10:10 AND %MW20;	10 words	117,20	106,45	106,45	12
	per word	2,64	2,40	2,40	
%MD0:10:=%MD20 AND %MD10:10;	10 double words	587,01	522,95	522,95	12
	per double word	6,47	6,15	6,15	
NOT					
%MW0:10:=NOT(%MW10:10);	10 words	110,28	100,25	100,25	9
	per word	2,96	2,75	2,75	
%MD0:10:=NOT(%MD10:10)	10 double words	126,39	114,00	114,00	9
	per double word	4,50	4,05	4,05	

Addition function on the table

The table below shows the instruction times for addition on a table.

ST	Size (of the table of words)	Execution time (µs)			Size (words) 37xx
		3705/08/10	3720 RAM	3720 card	
%MW20:=SUM(%MW0:10);	10 words	74,30	69,00	69,00	10
	per word	2,44	2,35	2,35	
%MD20:=SUM(%MD0:10);	10 double words	83,58	76,90	76,90	10
	per double word	3,17	2,95	2,95	
%MF20:=SUM_ARR(%MF0:10);	10 double words	1634	1257	1257	10
	per double word				

Table comparison function

The table below shows the instruction times for table comparison.

ST	Size (of the table of words)	Execution time (µs)			Size (words) 37xx
		3705/08/10	3720 RAM	3720 card	
%MW20:= EQUAL(%MW0:10;%MW10:10);	10 words	103,78	93,50	93,50	11
	per word	1,13	0,90	0,90	
%MD20:= EQUAL(%MD0:10;%MD10:10);	10 double words	116,17	103,40	103,40	11
	per double word	2,23	1,75	1,75	
%MF20:= EQUAL_ARR(%MF0:10;%MF10:10);	10 double words	741	570	607	11
	per double word				

Find function

The table below shows the instruction times for finding in a table.

ST	Size (of the table of words)	Execution time (µs)			Size (words) 37xx
		3705/08/10	3720 RAM	3720 card	
%MW20:= FIND_EQW(%MW0:10,%KW0)	10 words max poss	340,00	250,00	266,25	15
%MD20:= FIND_EQD(%MD0:10, %KD0)	10 double words, max poss	350,00	260,00	276,90	16
%MF20:= FIND_EQR(%MF0:10,%KF0)	10 double words	833	648	690,12	15
%MF20:= FIND_EQRP(%MF0:10, %KF0)	10 double words	845	650	692,25	15
%MD20:= FIND_GTR(%MF0:10, %KF0)	10 double words	836	643	684,79	15
%MD20:= FIND_LTR(%MF0:10, %KF0)	10 double words	836	643	684,79	15

Finding highest and lowest values

This table describes the instruction times for finding the highest and lowest values in a table.

ST	Size (of the table of words)	Execution time (µs)			Size (words) 37xx
		3705/08/10	3720 RAM	3720 card	
%MW20:= MAX_ARW(%MW0:10)	10 words	350,00	260,00	276,90	9
%MD20:= MAX_ARD(%MD0:10)	10 double words	410,00	300,00	319,50	9
%MF20:=MAX_ARR(%MF0:10);	10 double words	1366	1051	1119,31	9
%MF20:=MIN_ARR(%MF0:10)	10 double words	1270	977	1040,50	9

Calculating the number of occurrences

This table shows instruction times for the number of occurrences of a value in a table of words.

ST	Size (of the table of words)	Execution time (µs)			Size (words) 37xx
		3705/08/10	3720 RAM	3720 card	
%MW20:= OCCUR_ARW(%MW0:10, %KW0)	10 words	350,00	250,00	266,25	15
%MD20:= OCCUR_ARD(%MD0:10, %KD0)	10 double words	370,00	270,00	287,55	16
%MF20:= OCCUR_ARR(%MF0:10, %KF0)	10 double words	1265	973	1036,24	16

Rotate shift

The table below shows the rotate shift instruction times.

ST	Size (of the table of words)	Execution time (µs)			Size (words) 37xx
		3705/08/10	3720 RAM	3720 card	
ROL_ARW(word or value,%MWj:10)	10 words	550,00	400,00	426,00	9
ROL_ARD(%MDi,%MDj:10)	10 double words	590,00	430,00	457,95	9
ROL_ARR(%MFi,%MFj:10)	10 double words	585	450	479,25	9

Sort instruction

The table below shows the instruction times for sorting the elements in a table.

ST	Size (of the table of words)	Execution time (µs)			Size (words) 37xx
		3705/08/10	3720 RAM	3720 card	
SORT_ARW(%MWi,%MWj:10)	10 words max poss	970,00	700,00	745,50	9
SORT_ARD(%MDi,%MDj:10)	5 double words, max poss	610,00	450,00	479,25	9
SORT_ARR(%MFi,%MFj:10)	10 double words	1863	1433	1526,14	9

Time management instructions

Instructions on date, hour and period management

The table below shows the date, hour and period instruction times.

ST	Execution time (µs)			Size (words)
	3705/08/10	3720 RAM	3720 card	37xx
%MW2:4:= ADD_DT(%MW2:4,%MD8)	4 400.00	3 300.00	3 514.50	13
%MD2:= ADD_TOD(%MD2,%MD8)	2 100,00	1 550,00	1 650,75	9
%MB2:11:= DATE_TO_STRING(%MD40)	1 370,00	900,00	958,50	9
%MW5:= DAY_OF_WEEK()	220,00	280,00	298,20	5
%MD10:= DELTA_D(%MD2, %MD4)	1 520,00	1 130,00	1 203,45	9
%MD10:= DELTA_DT(%MD2:4,%MW6:4)	3 170,00	2 300,00	2 449,50	13
%MD10:= DELTA_TOD(%MD2,%MD4)	2 330,00	1 700,00	1 810,50	9
%MB2:20:= DT_TO_STRING(%MW50:4)	2 050,00	1 450,00	1 544,25	11
%MW2:4:= SUB_DT(%MW2:4,%MD8)	4 750,00	3 500,00	3 727,50	13
%MD2:= SUB_TOD(%MD2,%MD8)	2 330,00	1 700,00	1 810,50	9
%MB2:15:= TIME_TO_STRING(%MD40)	1 560,00	1 200,00	1 278,00	9
%MB2:9:= TOD_TO_STRING(%MD40)	1 270,00	800,00	852,00	9
%MD100:= TRANS_TIME(%MD2)	500,00	500,00	532,50	7

Real-time clock access

The table below shows the real time clock instruction times.

ST	Execution time (µs)			Size (words)
	3705/08/10	3720 RAM	3720 card	37xx
RRTC(%MW0:4)	93,60	84,80	84,80	5
WRTC(%MW0:4)	248,61	230,85	230,85	5
PTC(%MW0:5)	97,98	88,60	88,60	5
SCHED-ULE(%MW0,%MW1,%MW2,%MD10, %MD12,%M0)	1430	1100	1171,5	5

Instructions on strings of characters

Assigning and copying strings of characters

The table below shows the instruction times for assigning and copying strings of characters

ST	Size (characters)	Execution time (µs)			Size (words) 37xx
		3705/08/10	3720 RAM	3720 card	
%MB0:8:=%MB10:8	8 characters	105.16	93.80	93.80	9
	per character	1.65	1.30	1.30	
%MB0:8:='abcdefg'	8 characters	120.72	110.20	110.20	11
	per character	4.15	3.85	3.85	0.5

Word conversions<-> strings of characters

This table describes the instruction times for converting words and strings of characters.

ST	Execution time (µs)			Size (words) 37xx
	3705/08/10	3720 RAM	3720 card	
%MW1:= STRING_TO_INT(%MB0:7)	97.69	91.95	91.95	7
%MB0:7:= INT_TO_STRING(%MW0)	104.36	96.70	96.70	7

Double word conversions<-> strings of characters

This table describes the instruction times for converting double words and strings of characters.

ST	Execution time (µs)			Size (words) 37xx
	3705/08/10	3720 RAM	3720 card	
%MD1:= STRING_TO_DINT(%MB0:13)	1 070.53	965.62	965.62	7
%MB0:13:= DINT_TO_STRING(%MD0)	322.29	295.35	295.35	7

Floating point conversions<-> strings of characters

This table describes the instruction times for converting for floating points into strings of characters.

ST	Execution time (µs)			Size (words) 37xx
	3705/08/10	3720 RAM	3720 cards	
%MF1:= STRING_TO_REAL(%MB0:15)	1 783.70	1 634.53	1 634.53	7
%MB0:15:= REAL_TO_STRING(%MF0)	741.75	681.20	681.20	7

Character string manipulation instructions

This table describes the instruction times for manipulating strings of characters

ST	Execution time (µs)			Size (words) 37xx
	3705/08/ 10	3720 RAM	3720 card	
%MB10:20:= CON-CAT(%MB30:10,%MB50:10)	1 170.00	770.00	820.05	15
%MB10:20:= DELETE(%MB10:22,2,3);	950.00	600.00	639.00	15
%MW0:= EQUAL_STR(%MB10:20,%MB30:20); (the fifth character differs)	860.00	520.00	553.80	13
%MW0:= FIND(%MB10:20,%MB30:10);	1 610.00	1 000.00	1 065.00	13
%MB10:20:= IN-SERT(%MB30:10,%MB50:10,4);	1 270.00	800.00	852.00	17
%MB10:20:= LEFT(%MB30:30,20);	920.00	570.00	607.05	13
%MW0:= LEN(%MB10:20);	770.00	340.00	362.10	9
%MB10:20:= MID(%MB30:30,20,10);	1 080.00	700.00	745.50	15
%MB10:20:= RE-PLACE(%MB30:20,%MB50:10,10,10);	1 450.00	870.00	926.55	19
%MB10:20:= RIGHT(%MB30:30,20);	1 480.00	950.00	1 011.75	13

Application-specific and Orphée functions

Communication functions

The table below shows the communication function instruction times.

ST	Execution time (µs)			Size (words) 37xx
	3705/08/10	3720 RAM	3720 card	
SEND_REQ(%KW0:6,15,%MW0:1, %MW10:10,%MW30:4)	**2182**	**1818**	**1936**	**21**
SEND_TLG(%KW0:6,1,%MW0:5,%MW30:2)	1636	1364	1452	15

Operator dialog function

The table below shows the operator dialog instruction times.

ST	Execution time (µs)			Size (words) 37xx
	3705/08/10	3720 RAM	3720 card	
SEND_MSG(ADR#1.0,%MW0:2,%MW10:2)	2 240	2 000	2208	19
SEND_ALARM(ADR#1.0,%MW0:2,%MW10:2)	2 240	2 000	2208	19
GET_MSG(ADR#1.0,%MW0:2,%MW10:2)	2 240	2 000	2 208	19
GET_VALUE(ADR#1.0,%MW0,%MW10:2)	1 120	1 000	1 104	17
ASK_MSG(ADR#1.0,%MW0:2,%MW10:2,%MW20:2)	2 240	2 000	2 208	23
ASK_VALUE(ADR#1.0,%MW0,%MW10:2,%MW20:2)	2 240	2 000	2 208	21
DISPLAY_ALRM(ADR#1.0,%MW0,%MW10:2)	1 120	1 000	1 104	17
DISPLAY_GRP(ADR#1.0,%MW0,%MW10:2)	1 120	1 000	1 104	17
DISPLAY_MSG(ADR#1.0,%MW0,%MW10:2)	1 120	1 000	1 104	17
CONTROL_LEDS(ADR#1.0,%MW0:2,%MW10:2)	2 240	2 000	2 208	19
ASSIGN_KEYS(ADR#1.0,%MW0:2,%MW10:2)	2 240	2 000	2 208	19
PANEL_CMD(ADR#1.0,%MW0:2,%MW10:2)	2 240	2 000	2 208	19

Process control function

The table below shows the process control function instruction times.

ST	Condition	Execution time (µs)			Size (words) 37xx
		3705/08/10	3720 RAM	3720 card	
PID("PIDS1",'Unit',%IW3.5,%MW12,%M16,%MW284:43)	deval_mmi=0	1320	1100	1172	24
	deval_mmi=1	1080	900	958,5	
PWM(%MW11,%Q2.1,%MW385:5)		600	500	532,5	11
SERVO(%MW12,%IW3.6,%Q2.2,%Q2.3,%MW284:43,%MW390:10)		960	800	852	19
PID_MMI(ADR#0.0.4,%M1,%M2:5,%MW410:62)	EN=1	1140	950	1012	20

Orphée function

The table below shows the process control function instruction times.

ST	Condition	Execution time (µs)			Size (words) 37xx
		3705/08/10	3720 RAM	3720 card	
DSHR_RBIT(%MD102,16,%MD204,%MD206)	write 10 words	660	480	511	13
DSHRZ_C(%MD102,16,%MD204,%MD206)	request to copy 10 words	410	310	330	13
WSHL_RBIT(%MW102,8,%MW204,%MW206)	exchange 10 words	300	220	234	13
WSHR_RBIT(%MW102,8,%MW204,%MW206)	20 bytes	390	280	298	13
WSHRZ_C(%MW102,8,%MW204,%MW206)	20 bytes	300	220	234	13
SCOUNT(%M100,%MW100,%M101,%M102,%MW101,%MW102,%M200,%M201,%MW200,%MW201)	20 bytes	510	410	437	25

Explicit input/output instructions

Performance times

This table shows the explicit input/output instruction times.

ST	Execution time (µs)			Size (words) 37xx
	3705/08/10	3720 RAM	3720 card	
Read_Sts %CHi.MOD				
Any application except the processor communication channel	30	30	32	2
Read_Sts %CHi				
Analog input	180	180	216	6
Analog output	90	70	74	
Counting module CTZ	110	95	104	
Write_Param %CHi				
Analog input	790	570	790	6
Counting module CTZ	1127	1080	1083	
Read_Param %CHi				
Analog input	260	290	316	6
Counting module CTZ	338	295	300	
Save_Param %CHi				
Analog input	1234	1220	1240	6
Counting module CTZ	1370	1220	1240	
Restore_Param %CHi				
Analog input	550	510	535	6
Counting module CTZ	1160	1080	1097	
Write_Cmd %CHi				
Discrete output	50	47	52	6

6.3 instruction times on Premium PLCs

At a Glance

Aim of this section

This section describes the times for instructions carried out on Premium PLCs.

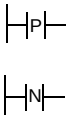
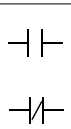
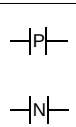
What's in this Section?

This Section contains the following Maps:

Boolean instruction performance characteristics

P57 1•/2• processor instruction times

The table below shows the boolean instruction times for the P57 1•/2• processors.

LD	IL	ST	Objects	Execution time (µs)								Size words
				57 1• ram	57 1• card	57 1• ram > 4 K	57 1• card > 4 K	57 2• ram	57 2• card	57 2• ram > 4 K	57 2• card > 4 K	
				0,37	0,50			0,06	0,21			1
	LD LDN		%M1 (1)	0,50	0,62	0,62	0,87	0,19	0,21	0,25	0,42	1
			%M1[%MW2]	1,50	2,25	1,50	2,25	0,62	1,25	0,62	1,25	6
			%MW0:X0 (2)	1,12	1,62	1,12	1,62	0,37	0,83	0,37	0,83	4
			%IWi.j:Xk (3)	1,75	2,50	1,75	2,50	0,62	1,25	0,62	1,25	6
			%MW0[%MW8] :X0	2,25	3,37	2,25	3,37	0,94	1,87	0,94	1,87	9
			%KW0[%MW8]: X0	2,25	3,37	2,25	3,37	0,94	1,87	0,94	1,87	9
	LDR, LDF		%M1	0,87	1,12	1,00	1,37	0,25	0,42	0,31	0,62	2
			%M1[%MW2]	1,87	2,75	1,87	2,75	0,69	1,46	0,69	1,46	7
	AND, ANDN, AND (, AND (N, same as OR			same as LD,LDN								
	ANDR, ANDF, AND (R, AND (F, same as OR			same as LDR,LDF								

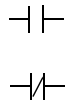
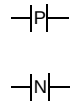
LD	IL	ST	Objects	Execution time (µs)								Size words
				57 1• ram	57 1• card	57 1• ram > 4 K	57 1• card > 4 K	57 2• ram	57 2• card	57 2• ram > 4 K	57 2• card > 4 K	
	XOR XORN		%M1	2,12	2,75	2,37	3,25	0,56	1,04	0,69	1,46	5
			%M1[%MW2]	3,37	4,75	3,37	4,75	0,94	2,29	0,94	2,29	12
			%MW0:X0	3,37	4,62	3,37	4,62	0,75	2,08	0,75	2,08	10
			%IWi.j:Xk	4,00	5,50	4,00	5,50	1,00	2,50	1,00	2,50	12
			%MW0[%MW8]:X0	4,50	6,37	4,50	6,37	1,31	3,12	1,31	3,12	15
			%KW0[%MW8]:X0	4,50	6,37	4,50	6,37	1,31	3,12	1,31	3,12	15
	XORR, XORF		%M1	3,62	4,75	3,87	5,25	0,81	1,87	0,94	2,29	9
			%M1[%MW2]	5,62	8,00	5,62	8,00	1,69	3,96	1,69	3,96	19
—( )— —(/)— —(S)— —(R)—	ST STN S R		%M1 (1)	1,00	1,25	1,12	1,50	0,37	0,46	0,44	0,62	2
			%M1[%MW2]	1,62	2,37	1,62	2,37	0,75	1,29	0,75	1,29	6
			%MW0:X0	1,25	1,75	1,25	1,75	0,50	0,83	0,50	0,83	4
			%NW{i}.j:Xk (3)	1,87	2,62	1,87	2,62	0,75	1,25	0,75	1,25	6
			%MW0[%MW8]:X0	2,37	3,50	2,37	3,50	0,75	1,92	0,75	1,92	9
multiple ladder coils, «cost» of the second and subsequent coils				0,62	0,75	0,75	1,00	0,31	0,25	0,37	0,42	1
operate block	[action]		executed block	0,25	0,50	0,25	0,50	0,12	0,42	0,12	0,42	2
			not executed	0,50	0,75	0,50	0,75	0,12	0,42	0,12	0,42	2
vertical operate block			between 2 %MWi	1,62	11,85	1,62	2,25	0,56	1,04	0,56	1,04	5
Convergence	[ )]	[ )]	executed block	0,37	0,13	0,37	0,50	0,06	0,21	0,06	0,21	1
Untracked divergence of a convergence	[ ]		Ladder, 1 divergence	0,37	0,50	0,37	0,50	0,06	0,21	0,06	0,21	1
	[MPS, MPP, MRD]		MPS+MPP list	1,12	1,50	1,12	1,50	0,19	0,62	0,19	0,62	3
			MRD List	0,37	0,50	0,37	0,50	0,06	0,21	0,06	0,21	1

Legend

1. This applies to all forcible bit objects: %I, %Q, %X, %M, %S,
2. other objects of the same kind: output bits of function block %Tmi.Q etc, extract bits of %SWi:Xj system words
3. other objects of the same kind: extract bits of shared words %NW{i}j:Xk, extract bits of I/O %IWi.j.Xk, %QWi.j.Xk words, extract bits of %KW, %Ii.j.ERR error bits.

P57 3•/4• processor instruction times

The table below shows the boolean instruction times for the P57 3•/4• processors.

LD	IL	ST	Objects	Execution time (µs)						Size in words
				57 3• ram	57 3• card	57 3• ram > 4 K	57 3• card > 4 K	57 4• ram	57 4• > 4K	
				0,04	0,17			0,02		1
	LD LDN		%M1 (1)	0,12	0,17	0,17	0,33	0,06	0,08	1
			%M1[%MW2]	0,42	1,00	0,42	1,00	0,21	0,21	6
			%MW0:X0 (2)	0,25	0,67	0,25	0,67	0,12	0,12	4
			%IWi.j:Xk (3)	0,42	1,00	0,42	1,00	0,21	0,21	6
			%MW0[%MW8]:X0	0,62	1,50	0,62	1,50	0,31	0,31	9
			%KW0[%MW8]:X0	0,62	1,50	0,62	1,50	0,31	0,31	9
	LDR, LDF		%M1	0,17	0,33	0,21	0,50	0,08	0,10	2
			%M1[%MW2]	0,46	1,17	0,46	1,17	0,23	0,23	7
	AND, ANDN, AND (, AND (N, same as OR			same as LD,LDN						
	ANDR, ANDF, AND (R, AND (F, same as OR			same as LD,LDN						

LD	IL	ST	Objects	Execution time (µs)						Size in words
				57 3• ram	57 3• card	57 3• ram > 4 K	57 3• card > 4 K	57 4• ram	57 4• > 4K	
	XOR XORN		%M1	0,37	0,83	0,46	1,17	0,19	0,23	5
			%M1[%MW2]	0,62	1,83	0,62	1,83	0,31	0,31	12
			%MW0:X0	0,50	1,67	0,50	1,67	0,25	0,25	10
			%IWi.j:Xk	0,67	2,00	0,67	2,00	0,33	0,33	12
			%MW0[%MW8]:X0	0,87	2,50	0,87	2,50	0,44	0,44	15
			%KW0[%MW8]:X0	0,87	2,50	0,87	2,50	0,44	0,44	15
	XORR, XORF		%M1	0,54	1,50	0,62	1,83	0,27	0,31	9
			%M1[%MW2]	1,12	3,17	1,12	3,17	0,56	0,56	19
—( )— —(/)— —(S)— —(R)—	ST STN S R		%M1 (1)	0,25	0,33	0,29	0,50	0,12	0,15	2
			%M1[%MW2]	0,50	1,00	0,50	1,00	0,25	0,25	6
			%MW0:X0	0,33	0,67	0,33	0,67	0,17	0,17	4
			%NW{i}.j:Xk (3)	0,50	1,00	0,50	1,00	0,25	0,25	6
			%MW0[%MW8]:X0	0,62	0,75	0,50	1,50	0,25	0,25	9
multiple ladder coils, «cost» of the second and subsequent coils				0,21	0,17	0,25	033	0,10	0,12	1
operate block	[action]		executed block	0,25	0,50	0,08	0,33	0,04	0,04	2
			not executed	0,50	0,75	0,08	0,33	0,04	0,04	2
vertical operate block			between 2 %MWi	1,62	11,85	0,37	0,83	0,19	0,19	5
Convergence	[ )]	[ )]	executed block	0,37	0,13	0,04	0,17	0,02	0,02	1
Untracked divergence of a convergence	[ ]		Ladder, 1 divergence	0,37	0,50	0,04	0,17	0,02	0,02	1
	[MPS, MPP, MRD]		MPS+MPP list	1,12	1,50	0,12	0,50	0,06	0,06	3
			MRD List	0,37	0,50	0,04	0,17	0,02	0,02	1

Legend

1. This applies to all forcible bit objects: %I, %Q, %X, %M, %S,
2. other objects of the same kind: output bits of function block %Tmi.Q etc, extract bits of %SWi:Xj system words
3. other objects of the same kind: extract bits of shared words %NW{i}j:Xk, extract bits of I/O %IWi.j.Xk, %QWi.j.Xk words, extract bits of %KW, %Ii.j.ERR error bits.

Instruction times for the function blocks

IEC timer The table below describes the instruction times for the IEC timer function block

LD	IL	ST	Conditions	Execution time (µs)				Size (words)
				571•	572•	573•	574•	
rising edge on IN	IN %TM1 (rising edge)	START %TM1	start timer	29	8,0	5,4	3,7	3
falling edge on IN	IN %TM1 (falling edge)	DOWN %TM1	stop timer	9	2,6	1,7	1,2	
IN =1	IN %TM1 (=1)		timer active	12	3,5	2,3	1,6	
IN =0	IN %TM1 (=0)		timer inactive	10	3,3	2,2	1,5	

PL7-3 timer The table below describes the instruction times for the PL7-3 timer function block

LD	IL	ST	Conditions	Execution time (µs)				Size (words)
				571•	572•	573•	574•	
		START %T1	start					3
		STOP %T1	stop	7	2,8	2,0	1,4	
E =0		RESET %T1	reset	7	3,1	2,2	1,6	
			timer active	11	3,4	2,3	1,7	
			timer inactive					

Upcounter/ downcounter

The table below describes the instruction times for the upcounter/downcounter function block

LD	IL	ST	Conditions	Execution time (µs)				Size (words)
				571•	572•	573•	574•	
reset, R=1	R %C8 (=1)	RESET %C8	reset	11	3,4	2,3	1,7	3
preset, S=1	S %C9 (=1)	PRESET %C9	preset	12	3,6	2,4	1,7	
rising edge on CU	CU %C8 (edge	UP %C8	up	12	3,7	2,5	1,8	
rising edge on CD	CD %C9 (edge	DOWN %C9	down	12	3,7	2,5	1,8	
inactive inputs	R/S/CU/CD inactive bit		no action	7	2,5	1,7	1,2	

Monostable

The table below describes the instruction times for the monostable function block

LD	IL	ST	Conditions	Execution time (µs)				Size (words)
				571•	572•	573•	574•	
rising edge on S	S %MN0, rising edge	START %MN0	start	24	7,2	4,9	3,4	3
S=1	S %MN0, S =1/0		active monostable	6	2,2	1,5	1,1	

Register

The table below describes the instruction times for the register function block

LD	IL	ST	Conditions	Execution time (µs)				Size (words)
				571•	572•	573•	574•	
edge on I	I %R2 (edge	PUT %R2	store	13	3,9	2,6	1,8	3
edge on O	O %R2 (edge)	GET %R2	remove from strorage	13	3,9	2,6	1,8	
R=1	R %R1 (=1)	RESET %R2	reset	9	3,3	2,3	1,6	
inactive inputs	I/O/R, inactive bit		no action	6	2,6	1,8	1,3	

Drum The table below describes the instruction times for the drum

LD	IL	ST	Conditions	Execution time (µs)				Size (words)
				571•	572•	573•	574•	
edge on U	U %DR0	UP %DR1	up, fixed	124	35	24	16	3
			per command bit	25	25	25	25	
R=1	R %DR1	RESET %DR2	reset, fixed	118	33	23	15	
			per command bit	25	25	25	25	
inactive inputs	R/U, inactive bit		no action, fixed	120	34	23	16	
			per command bit	25	25	25	25	

Integer and floating point arithmetic

Corrections according to object type

The times and volumes below are given for objects of the %MW0, %MD0 or %MF0 type.
This table describes the corrections which should be applied to the values given in the arithmetic instruction times table if other object types are used.

Object type	Object position	Type of correction	Execution time (µs)							Size words
			57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	
Single length immediate value	-	Value to be subtracted from or added to value mentioned for %MW	-0,12	-0,12	-0,12	0,00	-0,08	0,00	-0,04	0
Double length immediate value	-	Value to be subtracted from or added to value mentioned for %MD or %MF	-0,12	0,00	-0,19	0,21	-0,12	0,17	-0,06	1
Words, double Words or indexed floating points	After the := sign	Value to be added	1,12	1,75	0,56	1,04	0,37	0,83	0,19	5
	First operation, when the first operand is not indexed, or assignment	Value to be added	1,12	1,75	0,56	1,04	0,37	0,83	0,19	5
	Second operand if the first operand is also indexed	Value to be added	1,12	1,75	0,56	1,04	0,37	0,83	0,19	5
%KWi, %KWi[%MWj] %KDi, %KFi shared words, input/output words	-	Value to be added	0,62	0,87	0,25	0,42	0,17	0,33	0,08	2

Correction according to the context of the operation

This table describes the corrections to be made to the values given in the arithmetic instruction times table according to the context of the operation.

Context of the operation	Object type	Type of correction	Execution time (µs)							Size words
			57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	
The operation is in at least the second position in the statement E.g.: %MW2 in:=%MW0*%MW1*%MW2	%MW	Value to be added to value mentioned for %MW	0,50	0,62	0,19	0,33	0,12	0,25	0,06	1
	%MD or %MF	Value to be added to value mentioned for %MD or %MF	0,62	0,75	0,31	0,46	0,21	0,33	0,10	1
Operation with result of an operation in parentheses or more urgent E.g.: %MW0+%MW2+(...)	%MW	Value to be added to value mentioned for %MW	0,37	0,37	0,12	0,12	0,08	0,08	0,04	1
	%MD or %MF	Value to be added to value mentioned for %MD or %MF	0,50	0,50	0,25	0,25	0,17	0,17	0,08	1

Table of instruction times

The table below shows the arithmetic instruction times.

ST	Objects	Conditions	Execution time (µs)							Size words
			57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	
object after the := sign	%MW0	-	0,6	0,9	0,2	0,4	0,2	0,3	0,08	2
		%MW0+(...or %MW0 before *,/ or REM)	0,7	1,0	0,4	0,4	0,2	0,3	0,12	2
	%MD0	-	0,7	1,0	0,4	0,4	0,2	0,3	0,12	2
		%MW0+(...or %MW0 before *,/ or REM)	1,0	1,2	0,6	0,5	0,4	0,3	0,21	2
	%MF0		1,0	1,2	0,6	0,5	0,4	0,3	0,21	2
:=	%MW0		0,6	0,9	0,2	0,4	0,2	0,3	0,08	2
	%MD0 and %MF0		0,7	1,0	0,4	0,4	0,2	0,3	0,12	2

ST	Objects	Conditions	Execution time (µs)							Size words
			57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	
=, <>, <=, <, >, >=	%MW0		1,0	1,4	0,3	0,6	0,2	0,5	0,10	3
	%MD0		1,5	1,5	0,4	0,6	0,3	0,5	0,15	3
	%MF0		24	24	2,6	2,6	1,9	2,0	1,4	4
AND, OR, XOR	%MW0		0,6	0,9	0,2	0,4	0,2	0,3	0,08	2
	%MD0		0,7	1,0	0,4	0,4	0,2	0,3	0,12	2
+, -	%MW0		0,6	0,9	0,2	0,4	0,2	0,3	0,08	2
	%MD0		0,7	1,0	0,4	0,4	0,2	0,3	0,12	2
	%MF0		48	48	2,5	2,5	1,9	2,0	1,4	3
*	%MW0		6,3	6,5	2,0	2,2	1,5	1,6	1,2	3
	%MD0		29	29	9,3	9,3	6,3	6,4	4,7	3
	%MF0		44	44	2,5	2,5	1,9	2,0	1,4	3
/, REM	%MW0		6,9	7,2	2,1	2,3	1,5	1,6	1,2	3
	%MD0		148	149	34	34	21	21	15	3
/	%MF0		46	47	3,3	3,4	2,5	2,6	1,8	3
ABS, -object	%MW0		0,6	0,9	0,2	0,4	0,2	0,3	0,08	2
	%MD0		0,7	1,0	0,4	0,4	0,2	0,3	0,12	2
	%MF0		9	10	2,1	2,1	1,6	1,7	1,2	3
NOT	%MW0		0,6	0,9	0,2	0,4	0,2	0,3	0,1	2
	%MD0		0,7	1,0	0,4	0,4	0,2	0,3	0,1	2
SQRT	%MW0		19	19	3,5	3,7	2,1	2,2	1,5	3
	%MD0		62	62	10,2	10,3	5,7	5,8	4,4	3
	%MF0		117	117	2,8	2,8	2,1	2,1	1,5	3
INC, DEC	%MW0		0,7	1,0	0,4	0,4	0,2	0,3	0,12	2
	%MD0		1,0	1,2	0,6	0,5	0,4	0,3	0,21	2
SHL, SHR, ROL, ROR	%MW0	for 1 bit	2,0	2,9	0,8	1,5	0,5	1,2	0,27	7
	%MD0	for 1 bit	2,1	3,0	0,9	1,5	0,6	1,2	0,31	7
		per additional bit	0,042							
LN	%MF0		847	847	2,2	2,2	1,6	1,6	1,5	
LOG	%MF0		900	900	2,2	2,2	1,6	1,6	1,5	
EXP	%MF0		713	713	6,4	6,4	4,7	4,7	4,0	
EXPT	%MF0		1 747	1 747	2,2	2,2	1,6	1,6	1,5	
TRUNC	%MF0		1 753	1 753	2,2	2,2	1,6	1,6	1,5	
COS	%MF0		1 813	1 813	2,2	2,2	1,6	1,6	1,5	

ST	Objects	Conditions	Execution time (µs)							Size words
			57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	
SIN	%MF0		2 520	2 520	2,7	2,7	2,0	2,0	1,8	
TAN	%MF0		2 520	2 520	3,4	3,4	2,5	2,5	2,2	
ACOS	%MF0		1 720	1 720	2,1	2,1	1,6	1,6	1,4	
ASIN	%MF0		1 640	1 640	61	68	43	49	32	
ATAN	%MF0		103	142	32	36	23	26	17	
DEG_TO_RAD	%MF0		392	537	86	96	61	69	45	
RAD_TO_DEG	%MF0		380	522	86	96	61	69	46	

Instructions on the program and monitoring structures

Instruction times on the program

The table below shows the instruction times on the program.

ST	Execution time (µs)							Size (words)
	57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
Jump %Li	1,2	1,6	0,8	0,6	0,5	0,5	0,3	3
Maskevt	15,5	15,5	5,8	5,8	4,0	4,0	2,8	1
Unmaskevt	15,7	15,7	6,0	6,0	4,2	4,2	3,0	1
SRi	1,9	2,2	1,4	1,2	1,0	0,8	0,5	2
Return	0,6	0,9	0,2	0,4	0,2	0,3	0,1	2

Instruction times for the monitoring structures

The table below shows the instruction times for the monitoring structure-types.

ST		Execution time (µs)							Volume (words)
		57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
<cond>	condition evaluation								
forcible bit	see LD %M1 boolean instruction								
comparison	see comparisons =,<,> .etc.								
if <cond > then <action> end_if;	the times and volumes below should be added to those of the action contained in the structure								
true condition		0,2	0,5	0,1	0,4	0,1	0,3	0,04	2
false condition (jump)		0,5	0,7	0,1	0,4	0,1	0,3	0,04	
If <cond> then <action1> else <action2> end_if;									
true condition		0,7	1,2	0,2	0,8	0,2	0,7	0,08	4
false condition		0,5	0,7	0,1	0,4	0,1	0,3	0,04	
while <cond> do.<action> end_while									
go into the loop with loop iteration		0,7	1,2	0,2	0,8	0,2	0,7	0,08	2
exit the loop		0,5	0,7	0,1	0,4	0,1	0,3	0,04	
repeat <action> until <cond> end_repeat									

ST		Execution time (µs)							Volume (words)
		57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
go into the loop with loop iteration		0,5	0,7	0,1	0,4	0,1	0,3	0,04	2
last entry		0,2	0,5	0,1	0,4	0,1	0,3	0,04	
for <word1:=word2>to <word3> do <action> end_for									
input in the for, executed only once		1,2	1,7	0,5	0,8	0,3	0,7	0,17	15
go into the loop with loop iteration		3,5	5,0	1,2	2,5	0,8	2,0	0,42	
exit the loop		1,7	2,5	0,6	1,2	0,4	1,0	0,21	

Digital conversions

Instruction times The table below shows the instruction times for the digital conversions.

ST	Execution time (µs)							Volume (words)
	57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
BCD_TO_INT(%MW0)	17	17	3,3	3,4	1,9	2,1	1,5	3,0
INT_TO_BCD(%MW0)	14	14	2,8	3,0	1,7	1,9	1,3	3,0
GRAY_TO_INT(%MW0)	27	28	4,7	4,9	2,7	2,8	1,9	3,0
INT_TO_REAL(%MW0)	28	28	1,5	1,7	1,4	1,6	1,0	3,0
DINT_TO_REAL(%MD0)	24	24	1,7	1,8	1,6	1,7	1,2	3,0
REAL_TO_INT(%MF0)	41	42	1,6	1,7	1,5	1,6	1,1	3,0
REAL_TO_DINT(%MF0)	33	33	1,7	1,8	1,6	1,7	1,2	3,0
DBCD_TO_DINT(%MD0)	612	840	231	233	178	179	138	5
DBCD_TO_INT(%MD0)	537	737	203	204	156	157	121	5
DINT_TO_DBCD(%MD0)	512	702	193	195	149	150	115	5
INT_TO_DBCD(%MW0)	274	376	104	104	80	80	62	5

Instructions on a bit string

Initializing a bit table

This table shows the instruction times for initializing a bit table.

ST	Size (bit)	Execution time (µs)							Size (words)
		57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
%M30:8 := 0	8	3,6	4,5	2,4	3,2	1,6	2,3	0,8	7
%M30:16 := 1	16	5,6	6,5	4,4	5,2	3,0	3,7	1,5	7
%M30:24 := 2	24	12	14	5,6	6,8	3,7	4,8	2,4	12
%M30:32 := 2	32	14	16	7,6	8,8	5,1	6,1	3,1	12

Copying a bit table into a bit table

This table shows the instruction times for copying a bit table into another bit table.

ST	Size (bit)	Execution time (µs)							Size (words)
		57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
%M30:8 := %M20:8	8	6,9	7,9	5,4	6,0	3,6	4,2	1,8	8
%M30:16 := %M20:16	16	8,1	9,1	6,6	7,3	4,4	5,1	2,2	8
%M30:24 := %M20:24	24	22	23	14	16	10	11	5,4	13
%M30:32 := %M20:32	32	27	28	19	21	13	14	7,0	13
%M30:16 := COPY_BIT(%M20:16)	16	173	237	65	66	50	50	39	17
	32	263	360	99	100	76	77	59	17
	128	818	1 122	309	312	238	239	184	17

Logic instructions on a bit table

The table below shows the logic instruction times for a bit table.

ST	Size (bits)	Execution time (µs)							Size (words)
		57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
AND_ARX, OR_ARX, XOR_ARX									
%M0:16:= AND_ARX(%M30:16,%M50:16)	16	233	319	88	89	68	68	52	24
%M0:32:= AND_ARX(%M30:32,%M50:32)	32	368	504	139	140	107	107	83	24
%M0:128:= AND_ARX(%M30:128,%M50:128)	128	1 178	1 616	445	449	343	344	265	24
NOT_ARX									
%M0:16:= NOT_ARX(%M30:16)	16	173	237	65	66	50	50	39	17
	32	263	360	99	100	76	77	59	17
	128	818	1 122	309	312	238	239	184	17

Copying a bit table into a word table

This table shows the instruction times for copying bit tables into a word table.

ST	Size (bits)	Execution time (µs)							Size (words)
		57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
%MW1:= %M30:8	8	4,5	5,2	3,4	3,9	2,2	2,7	1,1	6
%MW1:= %M30:16	16	7,5	8,2	6,4	6,9	4,2	4,7	2,1	6
%MW2:= %M30:24	24	11	11	10	10	6,8	7,2	3,8	6
%MD2:= %M30:32	32	14	14	13	13	8,8	9,2	4,8	6
%MW1:4:= BIT_W(%M40:80,0,17,2)	17	231	317	87	88	67	68	52	23
%MD1:4:= BIT_D(%M30:80,0,33,0)	33	325	446	123	124	95	95	73	23

Copying a word table into a bit table

This table shows the instruction times for copying word tables into a bit table.

ST	Size (bits)	Execution time (µs)							Size (words)
		57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
%M30:8:= %MW1	8	3,6	4,4	2,5	3,0	1,7	2,2	0,8	6
%M30:16:= %MW2	16	5,6	6,4	4,5	5,0	3,0	3,5	1,5	6
%M30:24:= %MD1	24	12	13	6,1	6,9	4,3	5,1	3,1	11
%M30:32:= %MD3	32	14	15	8,1	8,9	5,7	6,4	3,7	11
%M30:32:= W_BIT(%MW200:2,0,2,0)	32	231	317	87	88	67	68	52	23
%M30:32:= D_BIT(%MD0:1,0,2,0)	32	275	377	104	105	80	80	62	23

Instructions on tables of words, double words and floating points

Initializing a table of words with one word

This table shows the instruction times for initializing a table of words with one word.

ST	Size (of the table of words)	Execution time (µs)							Size (words)
		57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
%MW0:10:= %MW100	10 words	34	35	14	15	10	11	6,7	10
	per word	0,16	0,16	0,15	0,15	0,12	0,12	0,08	
%MD0:10:= %MD100	10 double words	53	54	19	20	13	14	8,8	10
	per double word	1,98	1,98	0,57	0,57	0,37	0,37	0,26	

Copying a table of words into a table of words

This table shows the instruction times for copying a table of words into another table of words.

ST	Size (of the table of words)	Execution time (µs)							Size words
		57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
%MW0:10:=%MW20:10;	10 words	63	65	25	26	17	19	12	15
	per word	0,28	0,28	0,31	0,31	0,24	0,24	0,15	
%MD0:10:=%MD20:10;	10 double words	69	71	29	30	20	22	14	15
	per double word	0,79	0,79	0,71	0,71	0,53	0,53	0,36	

Arithmetic and logic instructions between 2 tables of words

The table below shows the arithmetic and logic instruction times between two tables of words.

ST	Size (of the table of words)	Execution time (µs)							Size words
		57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
+, -									
%MW0:10:= %MW10:10 + %MW20:10;	10 words	110	112	41	43	28	30	19	23
	per word	4,7	4,7	1,8	1,8	1,3	1,3	0,9	
%MD0:10:= %MD10:10+%MD20:10;	10 double words	154	156	60	62	41	43	28	23
	per double word	8,9	8,9	3,6	3,6	2,5	2,5	1,7	
*									
%MW0:10:= %MW10:10 * %MW20:10;	10 words	127	129	47	50	33	35	23	23
	per word	6,4	6,4	2,4	2,4	1,7	1,7	1,2	
%MD0:10:= %MD10:10 * %MD20:10;	10 double words	441	444	153	155	104	106	73	23
	per double word	37,6	37,6	12,9	12,9	8,7	8,7	6,2	
/, REM									
%MW0:10:= %MW10:10 / %MW20:10;	10 words	133	135	49	52	34	36	24	23
	per word	7,0	7,0	2,6	2,6	1,8	1,8	1,4	
%MD0:10:= %MD10:10 / %MD20:10;	10 double words	1 639	1 642	395	397	248	250	172	23
	per double word	157	157	37	37	23	23	16	
AND, OR, XOR									
%MW0:10:=%MW10:10 AND %MW20:10;	10 words	108	111	40	43	28	30	19	23
	per word	4,5	4,5	1,7	1,7	1,2	1,2	0,8	
%MD0:10:=%MD10:10 AND %MD20:10;	10 double words	155	158	61	63	42	44	29	23
	per double word	9	9	4	4	3	3	2	

Arithmetic and logic instructions between 1 table of words and 1 word

The table below shows the arithmetic and logic instruction times between 1 table of words and 1 word.

ST	Size (of the table of words)	Execution time (µs)							Size (words)
		57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
+, -									
%MW0:10 :=%MW10:10 + %MW20; or %MW0:10:= %MW20 + %MW10:10	10 words	86	88	30	32	21	22	14	18
	per word	2,8	2,8	0,8	0,8	0,5	0,5	0,4	
%MD0:10:=%MD10:10 + %MD20;	10 double words	112	114	41	43	28	30	19	18
	per double word	5,2	5,2	1,9	1,9	1,3	1,3	0,9	
*									
%MW0:10:= %MW20*%MW10:10;	10 words	113	115	38	40	26	27	18	18
	per word	5,6	5,6	1,6	1,6	1,1	1,1	0,7	
%MD0:10:= %MD20*%MD10:10;	10 double words	381	383	132	134	90	92	64	18
	per double word	32	32	11	11	7,4	7,4	5,3	
/, REM									
%MW0:10 :=%MW10:10 / %MW30;	10 words	140	142	46	48	31	33	21	18
	per word	8,4	8,4	2,4	2,4	1,6	1,6	1,1	
%MD0:10:= MD10:10 / %MD30	10 double words	1 585	1 587	375	377	235	236	163	18
	per double word	152	152	35	35	22	22	15	
AND, OR, XOR									
%MW0:10:=%MW10:10 AND %MW20;	10 words	86	88	30	32	21	22	14	18
	per word	2,8	2,8	0,8	0,8	0,5	0,5	0,4	

ST	Size (of the table of words)	Execution time (µs)							Size (words)
		57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
%MD0:10:=%MD20 AND %MD10:10;	10 double words	381	383	132	134	90	92	64	18
	per double word	4,2	4,2	1,5	1,5	1,0	1,0	0,7	
NOT									
%MW0:10:=NOT(%MW10:10);	10 words	74	75	26	28	18	19	12	15
	per word	1,9	1,9	0,5	0,5	0,4	0,4	0,2	
%MD0:10:=NOT(%MD10:10)	10 double words	84	86	31	33	22	23	15	15
	per double word	2,9	2,9	1,0	1,0	0,7	0,7	0,5	

Addition function on the table

The table below shows the instruction times for the addition function on the table.

ST	Size (of the table of words)	Execution time (µs)							Size (words)
		57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
%MW20:=SUM(%MW0:10);	10 words	51	53	17	18	12	12	8	16
	per word	1,6	1,6	0,4	0,4	0,3	0,3	0,2	
%MD20:=SUM(%MD0:10);	10 double words	58	59	19	20	13	14	9	16
	per double word	2,1	2,1	0,6	0,6	0,4	0,4	0,3	

Table comparison function

The table below shows the table comparison instruction times.

ST	Size (of the table of words)	Execution time (µs)							Size (words)
		57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
%MW20:= EQUAL(%MW0:10;%MW10:10);	10 words	67	69	26	28	18	20	13	27
	per word	0,6	0,6	0,4	0,4	0,3	0,3	0,2	
%MD20:= EQUAL(%MD0:10;%MD10:10);	10 double words	74	76	31	33	22	23	15	27
	per double word	1,2	1,2	0,9	0,9	0,7	0,7	0,5	

Find function

The table below shows the find in a table instruction times.

ST	Size (of the table of words)	Execution time (µs)							Size (words)
		57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
%MW20:= FIND_EQW(%MW0:10,%KW0)	10 words, max poss	150	206	57	57	44	44	34	14
%MD20:= FIND_EQD(%MD0:10, %KD0)	10 double words, max poss	163	223	61	62	47	48	37	15

Finding highest and lowest values

This table describes the instruction times for finding the highest and lowest values in a table.

ST	Size (of the table of words)	Execution time (µs)							Size (words)
		57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
%MW20:= MAX_ARW(%MW0:10)	10 words	163	223	61	62	47	48	37	12
%MD20:= MAX_ARD(%MD0:10)	10 double words	194	266	73	74	56	57	44	12

Calculating the number of occurrences

This table shows the instruction times for the number of occurrences of a value in a table of words.

ST	Size (of the table of words)	Execution time (µs)							Size (words)
		57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
%MW20:= OCCUR_ARW(%MW0:10, %KW0)	10 words	163	223	61	62	47	48	37	14
%MD20:= OCCUR_ARD(%MD0:10, %KD0)	10 double words	175	240	66	67	51	51	39	15

Rotate shift

The table below shows the rotate shift instruction times.

ST	Size (of the table of words)	Execution time (µs)							Size (words)
		57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
ROL_ARW(word or value,%MWj:10)	10 words	250	343	94	95	73	73	56	12
ROL_ARD(%MDi,%MDj:10)	10 double words	269	369	102	102	78	79	61	12

Sort instruction

The table below shows the instruction times for sorting the elements in a table.

ST	Size (of the table of words)	Execution time (µs)							Size (words)
		57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
SORT_ARW(%MWi,%MWj:10)	10 words max poss	450	618	170	172	131	132	101	12
SORT_ARD(%MDi,%MDj:10)	5 double words, max poss	275	377	104	105	80	80	62	12

Calculation of length

The table below shows the instruction times for calculating the length of a table.

ST	Size (of the table of words)	Execution time (µs)							Size (words)
		57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
LENGTH_ARW(tab_word)		31	43	12	12	9	9	7	6
LENGTH_ARD(tab_dword)		31	43	12	12	9	9	7	6
LENGTH_ARW(tab_reel)		31	43	12	12	9	9	7	6
LENGTH_ARW(tab_bit)		31	43	12	12	9	9	7	6

Floating point tables

The table below shows the instruction times on a floating point table.

ST	Size (of the table of words)	Execution time (µs)							Size (words)
		57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
SUM_ARR	10 integers	794	942	186	207	132	149	98	
FIND_EQR	10 integers, median poss	390	535	83	93	59	67	44	
FIND_EQRP	10 integers, median poss	391	536	83	93	59	67	44	
FIND_GTR	10 integers, median poss	390	535	83	93	59	67	44	
FIND_LTR	10 integers, median poss	390	535	83	93	59	67	44	
MAX_ARR	10 integers	648	889	160	179	114	128	85	
MIN_ARR	10 integers	601	825	148	164	105	118	78	
OCCUR_ARR	10 integers	598	821	147	164	104	118	78	
ROL_ARR	10 integers	273	374	67	75	48	54	35	
ROR_ARR	10 integers	264	363	65	72	46	52	34	
SORT_ARR	10 integers	896	1 229	220	245	156	176	116	
EQUAL_ARR	10 integers	344	472	84	94	60	68	45	

Time management instructions

Instructions for date, hour and period management

The table below shows the instruction times for date, hour and period management.

ST	Execution time (µs)							Size (words)
	57 1• ram	**57 1• card**	**57 2• ram**	**57 2• card**	**57 3• ram**	**57 3• card**	**57 4•**	**57• •**
%MW2:4:= ADD_DT(%MW2:4,%MD8)	1 951	2 677	737	744	568	571	440	19
%MD2:= ADD_TOD(%MD2,%MD8)	1 025	1 407	387	391	298	300	231	9
%MB2:11:= DATE_TO_STRING(%MD40)	606	832	229	231	176	177	137	12
%MW5:= DAY_OF_WEEK()	88	121	33	34	26	26	20	5
%MD10:= DELTA_D(%MD2, %MD4)	731	1 004	276	279	213	214	165	9
%MD10:= DELTA_DT(%MD2:4,%MW6:4)	1 506	2 067	569	574	438	441	339	19
%MD10:= DELTA_TOD(%MD2,%MD4)	1 113	1 527	421	424	324	325	251	9
%MB2:20:= DT_TO_STRING(%MW50:4)	707	970	267	269	206	207	159	17
%MW2:4:= SUB_DT(%MW2:4,%MD8)	2 344	3 216	886	893	682	685	528	19
%MD2:= SUB_TOD(%MD2,%MD8)	1 113	1 527	421	424	324	325	251	9
%MB2:15:= TIME_TO_STRING(%MD40)	794	1 089	300	303	231	232	179	12
%MB2:9:= TOD_TO_STRING(%MD40)	519	712	196	198	151	152	117	12
%MD100:= TRANS_TIME(%MD2)	331	455	125	126	96	97	75	7

Real-time clock access

The table below shows the instruction times for the real-time clock.

ST	Execution time (µs)							Size (words)
	57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
RRTC(%MW0:4)	88	89	30	30	21	21	14	8
WRTC(%MW0:4)	69	70	25	25	17	17	11	8
PTC(%MW0:5)	74	75	26	27	18	19	12	8
SCHED-ULE(%MW0,%MW1,%MW2,%MD10, %MD12,%M0)ÿ	88	89	30	30	21	21	14	8

Timer functions

The table below shows the instruction times for the timer functions.

ST	Execution time (µs)							Size (words)
	57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
FTON	53	53	28	32	21	24	12	
FTOF	53	53	28	32	21	24	12	
FTP	53	53	28	32	21	24	12	
FPULSOR	181	249	69	69	53	53	41	

Character string instructions

Character string assignment and copying

The table below describes the instruction times for assigning and copying character strings

ST	Size (characters)	Execution time (µs)							Size (words)
		57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
%MB0:8:=%MB10:8	8 characters	66	67	27	27	18	19	14	15
	per character	0,39	0,39	0,30	0,30	0,23	0,23	0,16	
%MB0:8:='abcdefg'	8 characters	85	85	29	29	20	20	14	14
	per character	2,37	2,37	0,68	0,68	0,47	0,47	0,36	0,5

Converting words <-> strings of characters

This table describes the instruction times for converting words and character strings.

ST	Execution time (µs)							Size (words)
	57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
%MW1:= STRING_TO_INT(%MB0:7)	69	71	23	23	16	16	12	10
%MB0:7:= INT_TO_STRING(%MW0)	74	75	23	23	15	16	12	10

Converting double word<-> character strings

This table describes the instruction times for converting double words and character strings.

ST	Execution time (µs)							Size (words)
	57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
%MD1:= STRING_TO_DINT(%MB0:13)	706	707	237	237	160	160	115	10
%MB0:13:= DINT_TO_STRING(%MD0)	215	216	66	67	44	45	33	10

Converting character string<-> floating points

This table describes the instruction times for converting floating points into character strings.

ST	Execution time (µs)							Size (words)
	57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
%MF1:= STRING_TO_REAL(%MB0:15)	1 912	1 913	344	344	237	237	155	10
%MB0:15:= REAL_TO_STRING(%MF0)	500	501	140	140	96	96	63	10

Instructions for manipulating character strings

This table describes the instruction times for manipulating character strings

ST	Execution time (µs)							Size (words)
	57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
%MB10:20:= CONCAT(%MB30:10,%MB50:10)	494	678	187	188	144	144	111	24
%MB10:20:= DELETE(%MB10:22,2,3);	400	549	151	152	116	117	90	21
%MW0:= EQUAL_STR(%MB10:20,%MB30:20); (the fifth character differs)	338	463	128	129	98	99	76	19
%MW0:= FIND(%MB10:20,%MB30:10);	650	892	246	248	189	190	146	19
%MB10:20:= INSERT(%MB30:10,%MB50:10,4);	519	712	196	198	151	152	117	26
%MB10:20:= LEFT(%MB30:30,20);	369	506	139	141	107	108	83	19
%MW0:= LEN(%MB10:20);	219	300	83	83	64	64	49	12
%MB10:20:= MID(%MB30:30,20,10);	444	609	168	169	129	130	100	21
%MB10:20:= REPLACE(%MB30:20,%MB50:10,10,10);	556	763	210	212	162	163	125	28
%MB10:20:= RIGHT(%MB30:30,20);	606	832	229	231	176	177	137	19

Extracting words The table below shows the instruction times for extracting words

ST	**Execution time (µs)**							**Size (words)**
	57 1• ram	**57 1• card**	**57 2• ram**	**57 2• card**	**57 3• ram**	**57 3• card**	**57 4•**	**57• •**
LW	32	44	12	12	9	9	7	
HW	32	44	12	12	9	9	7	
CONCATW	32	44	12	12	9	9	7	

Application-specific and Orphee functions

Communication functions

The table below shows the instruction times for the communication functions.

ST	Execution time (µs)							Size (words)
	57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
SEND_REQ: instruction execution time, time to be added in the communication system								
SEND_REQ(%KW0:6,15,%MW0:1, %MW10:10,%MW30:4)	1 250	1 715	472	476	364	366	282	33
SEND_TLG: processing is simultaneous with the instruction, no time needs to be added in the communication system								
SEND_TLG(%KW0:6,%MW0:5,%MW30:2)	938	1 287	354	357	273	274	211	24
SERVER for 120 octets	3 825	4 244	2 225	2 229	1 677	1 679	1 427	16
WRITE_ASYN for 500 words	2 975	3 301	1 731	1 734	1 305	1 306	1 110	16
READ_ASYN for 500 words	2 975	3 301	1 731	1 734	1 305	1 306	1 110	16

Dialog operator function

The table below shows the instruction times for the dialog operator.

ST	Execution time (µs)							Size (words)
	57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	574•	57• •
SEND_MSG(ADR#1.0,%MW0:2,%MW10:2)	1250	1715	472	476	364	366	282	25
SEND_ALARM(ADR#1.0,%MW0:2,%MW10:2)	1250	1715	472	476	364	366	282	25
GET_MSG(ADR#1.0,%MW0:2,%MW10:2)	1250	1715	472	476	364	366	282	25
GET_VALUE(ADR#1.0,%MW0,%MW10:2)	625	858	236	238	182	183	141	20
ASK_MSG(ADR#1.0,%MW0:2,%MW10:2,%MW20:2)	1250	1715	472	476	364	366	282	32
ASK_VALUE(ADR#1.0,%MW0,%MW10:2,%MW20:2)	1250	1715	472	476	364	366	282	27
DISPLAY_ALRM(ADR#1.0,%MW0,%MW10:2)	625	858	236	238	182	183	141	20
DISPLAY_GRP(ADR#1.0,%MW0,%MW10:2)	625	858	236	238	182	183	141	20
DISPLAY_MSG(ADR#1.0,%MW0,%MW10:2)	625	858	236	238	182	183	141	20

ST	Execution time (µs)							Size (words)
	57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
CONTROL_LEDS(ADR#1.0,%MW0:2,%MW10:2)	1250	1715	472	476	364	366	282	25
ASSIGN_KEYS(ADR#1.0,%MW0:2,%MW10:2)	1250	1715	472	476	364	366	282	25
PANEL_CMD(ADR#1.0,%MW0:2,%MW10:2)	1250	1715	472	476	364	366	282	25

Process control function

The table below shows the instruction times for the process control functions.

ST	Condition	Execution time (µs)							Size (words)
		57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
PID("PIDS1",'Unit',%IW3.5,%MW12,%M16,%MW284:43)	deval_mmi=0	688	943	260	262	200	201	155	32
	deval_mmi=1	563	772	213	214	164	165	127	
PWM(%MW11,%Q2.1,%MW385:5)		313	429	118	119	91	91	70	17
SERVO(%MW12,%IW3.6,%Q2.2,%Q2.3,%MW284:43,%MW390:10)		500	686	189	191	145	146	113	31
PID_MMI(ADR#0.0.4,%M1,%M2:5,%MW410:62)	EN=1	625	858	236	238	182	183	141	30

Data storage

The table below shows the instruction times for the data storage functions.

ST	Condition	Execution time (µs)							Size (words)
		57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
SET_PCMCIA	fixed		350		70		40	30	
	per word		0,4		0,3		0,2	0,2	
WRITE_PCMCIA	fixed		350		70		40	30	
	per word		0,8		0,3		0,3	0,2	
READ_PCMCIA	fixed		350		70		40	30	
	per word		0,7		0,4		0,3	0,4	

Orphee function The table below shows the instruction times for the process control functions.

ST	Execution time (µs)							Size (words)
	57 1• ram	**57 1• card**	**57 2• ram**	**57 2• card**	**57 3• ram**	**57 3• card**	**57 4•**	**57• •**
DSHL_RBIT(%MD102,16,%MD204,%MD206)	200	274	76	76	58	58	45	17
DSHR_RBIT(%MD102,16,%MD204,%MD206)	319	437	120	122	93	93	72	17
DSHRZ_C(%MD102,16,%MD204,%MD206)	194	266	73	74	56	57	44	17
WSHL_RBIT(%MW102,8,%MW204,%MW206)	138	189	52	52	40	40	31	17
WSHR_RBIT(%MW102,8,%MW204,%MW206)	181	249	69	69	53	53	41	17
WSHRZ_C(%MW102,8,%MW204,%MW206)	138	189	52	52	40	40	31	17
SCOUNT(%M100,%MW100,%M101,%M102, %MW101,%MW102,%M200,%M201,%MW200, %MW201)	263	360	99	100	76	77	59	38

Explicit input/output instructions

Performance times

This table shows the explicit input/output instruction times.

ST	Execution time (µs)							Size (words)
	57 1• ram	**57 1• card**	**57 2• ram**	**57 2• card**	**57 3• ram**	**57 3• card**	**57 4•**	**57• •**
Read_Sts %CHi.MOD								
Any application except the processor communication channel	552	651	291	292	220	220		2
Read_Sts %CHi								
Discrete input	296	317	180	181	136	136	117	6
Discrete output	386	426	227	227	171	171	145	
Analog input	334	363	201	201	151	151	129	
Analog output	327	354	197	197	148	148	127	
CTY 2A/4A counting module	327	354	197	197	148	148	127	
CTY 2C counting module	311	335	189	189	142	142	122	
CFY 11/22 step by step module	448	505	254	255	192	192	163	
CAY 21/41 axis command module	334	363	201	201	151	151	129	
Write_Param %CHi								
Analog input	499	574	274	275	207	207	174	6
Analog output	474	540	265	265	200	200	169	
CTY 2A/4A counting module	603	731	302	303	229	230	190	
CTY 2C counting module	400	444	233	234	176	176	150	
CFY 11/22 step by step module	577	691	297	298	225	226	187	
CAY 21/41 axis command module	461	522	260	260	196	196	166	
Read_Param %CHi								
Analog input	115	118	75	75	56	56	49	6
Analog output	115	118	75	75	56	56	49	
CTY 2A/4A counting module	334	363	201	201	151	151	129	
CTY 2C counting module	349	381	209	209	157	157	134	
CFY 11/22 step by step module	393	435	230	230	173	173	148	
CAY 21/41 axis command module	386	426	227	227	171	171	145	
Save_Param %CHi								

ST	Execution time (µs)							Size (words)
	57 1• ram	57 1• card	57 2• ram	57 2• card	57 3• ram	57 3• card	57 4•	57• •
Analog input	635	787	306	307	232	233	191	6
Analog output	640	795	306	307	233	233	191	
CTY 2A/4A counting module	752	1 049	271	274	209	211	160	
CTY 2C counting module	379	417	223	223	168	168	143	
CFY 11/22 step by step module	421	470	243	243	183	183	155	
CAY 21/41 axis command module	421	470	243	243	183	183	155	
Restore_Param %CHi								
Analog input	467	531	262	263	198	198	167	6
Analog output	467	531	262	263	198	198	167	
CTY 2A/4A counting module	608	739	303	304	230	230	190	
CTY 2C counting module	349	381	209	209	157	157	134	
CFY 11/22 step by step module	588	707	299	300	227	227	188	
CAY 21/41 axis command module	480	548	267	268	202	202	170	
Write_Cmd %CHi								
Discrete output	288	308	176	176	133	133	114	6
Analog inputs							0	
Input forcing	134	138	86	86	65	65	56	
Input recalibration	691	895	303	304	231	232	186	
Analog outputs							0	
Forcing	143	147	92	92	69	69	60	
Smove %CH1.0(%MW1,%MW2,%MW3,%MD4,%MD5,%MW6)								
CFY 11/22 step by step module	617	755	304	305	231	231	190	19
CAY 21/41 axis command module	608	739	303	304	230	230	190	

DFB function block

Size occupied by the DFB type

The following formula is used to calculate the size occupied by the DFB type:

DFB type size = Size of DFB variables and parameters + DFB Code size

Size of DFB variables and parameters

The size of the DFB variables and parameters is calculated as follows:

Size of DFB variables and parameters = 110 + sum of variable and parameter descriptors + sum of sizes occupied by each variable or parameter

with:

Descriptor of'a variable or of'a parameter = 5.5 + (Number of characters in the name of the variable or parameter)/2

and

Size occupied by each variable or parameter:

Type	IN	IN/OUT	OUT	PUBLIC	PRIVATE
EBOOL	0,5	2	0,5	0,5	0,5
BOOL	0,5	2	0,5	0,5	0,5
WORD	1	2	1	1	1
DWORD	2	2	2	2	2
REAL	2	2	2	2	2
AR_X	3	3	0.5*N	0.5*N	0.5*N
AR_W	3	3	N	N	N
AR_D	3	3	2*N	2*N	2*N
AR_R	3	3	2*N	2*N	2*N
STRING	3	3	0.5*N	0.5*N	0.5*N

N = number of'table elements or character string length (STRING)

DFB code size

The size of the DFB variables and parameters is calculated as follows:

DFB size code = 11 + sum of each of the instruction sizes(1)
(1) The following volumes, depending on the variable or parameter contained in the'instructions, are to be added to the instruction size:

Nature	Type	Volume
IN	EBOOL	0,5
	BOOL, WORD, DWORD, REAL	0
	AR_X,AR_W,AR_D,AR_R,STRING	3
IN/OUT	EBOOL	3
	BOOL, WORD, DWORD, REAL	3
	AR_X,AR_W,AR_D,AR_R,STRING	3
OUT, PUBLC PRIVATE	EBOOL	0,5
	BOOL, WORD, DWORD, REAL	0
	AR_X,AR_W,AR_D,AR_R,STRING	0

Indexed object:

Nature	Type	Volume
IN	AR_X,AR_W,AR_D,AR_R	7
IN/OUT	AR_X,AR_W,AR_D,AR_R	7
OUT, PUBLIC, PRIVATE	AR_X,AR_W,AR_D,AR_R	6

Size occupied by ' DFB usage

Calling'one instance of the DFB with no parameter = 6 words

Calculation for a parameter

Nature	Type	Volume
IN	EBOOL, BOOL, WORD, DWORD, REAL	same assignment:=
	AR_X,AR_W,AR_D,AR_R, STRING	14
IN/OUT	EBOOL, WORD, DWORD, REAL	10
	BOOL, AR_X,AR_W,AR_D,AR_R	14
OUT,	All types	same assignment:=

Using'one variable of'one instance: add one word per report

Execution time

The total DFB execution time is given by the following formula:

Total time of'DFB execution = DFB maximum transmission rate code + Sum of times of'access to DFB variables and parameters + DFB call (no parameter) + Sum of time of'access for each parameter

The following table gives the execution time in µs.

Element	**Type**	**571• ram**	**571• card**	**572• ram**	**572• card**	**573• ram**	**573• card**	**574•**
Projection for DFB code		13,0	16,3	4,8	4,8	3,4	3,8	1,1
Accessing a DFB variable or parameter (1)								
Indexed objects								
IN	EBOOL	0,2	0,3	0,09	0,12	0,06	0,10	0,02
	BOOL,WORD,DWORD,REAL	0	0	0	0	0	0	0
	AR_X,AR_W,AR_D,AR_R,STRING	1,4	1,7	0,5	0,8	0,4	0,6	0,1
IN/OUT		1,4	1,7	0,5	0,8	0,4	0,6	0,1
OUT, PUBLIC, PRI-VATE	EBOOL	0,2	0,3	0,09	0,12	0,06	0,10	0,02
	BOOL,WORD,DWORD,REAL	0	0	0	0	0	0	0
	AR_X,AR_W,AR_D,AR_R,STRING	0	0	0	0	0	0	0
Non-indexed objects								
IN, IN/ OUT	AR_X,AR_W,AR_D,AR_R,	2,8	3,5	1,0	0,9	0,7	0,7	0,2
OUT, PUBLIC, PRI-VATE	AR_X,AR_W,AR_D,AR_R,	2,1	2,6	0,8	1,1	0,5	0,9	0,2
DFB call (no parameter)		3,8	4,8	1,5	1,8	1,0	1,4	0,3
Calculating by parameter (1)								
IN	EBOOL	same:=						
	BOOL,WORD,DWORD,REAL	same:=						
	AR_X,AR_W,AR_D,AR_R,STRING	3,5	4,4	1,5	1,9	1,0	1,6	0,3
IN/OUT	EBOOL,WORD,DWORD,REAL	2,8	3,5	1,0	1,4	0,7	1,1	0,2
	BOOL,AR_X,AR_W,AR_D,AR_R,STRING	3,5	4,4	1,5	1,9	1,0	1,6	0,3
OUT	All types	same:=						

(1) value to add relating to operations applying to %M-type objects.

6.4 Advanced functions

At a Glance

Aim of this section

This section gives the amount of memory occupied by the advanced functions, and the method of calculating the number of instructions.

What's in this Section?

This Section contains the following Maps:

Description of the memory zones

Recap

The application is divided into several memory zones:

- bit memory zone:
  - this zone is specific to the TSX 37 PLCs and is restricted to 1280 bits
  - this zone is part of the data memory zone for the TSX 57 PLCs,
- data memory zone (words),
- application memory zone, comprising:
  - the configuration,
  - the program,
  - the constants.

The bit memory and data zones are always stored in the internal RAM while the application memory zone can be stored in the internal RAM or on the memory card.

Object memory size

Description

This table shows the memory size occupied by each type of PL7 language object.

Object type	Bit memory (in words)	Data (in words)	Application (in words)
Grafcet steps (%Xi, %Xi.T)	0,5	1	
%Mi	0,5		
Words (%MWi)		1	
Constants (%KWi)			1,25
%NWi		1	
%Ti		4	2
%TMi		5	2
%MNi		4	2
%Ci		3	1
%Ri (length lg)		6+lg	2
%DRi		6	49

Grafcet interpreter data = 355 + 2 x No. of active configured steps + (No. of valid configured transitions) / 2

Review of the memory usage of the modules on MIcro

General

Note: This information is for a specific processor version. They can be subject to 'slight' variations according to how the product develops.

The tables below give the size occupied in each of the zones and a fixed size to be added to the memory usage the first time an application-specific function is used, for each module type.

Processors

The table below gives the memory usage for the TSX 37 processor modules.

Processors	Bit memory (words)	Data (words)	Application zone (words)
TSX 37-05/08/10	70	1560	920
TSX 37-21	70	1570	930
TSX 37-22	70	2110	1280
Using FAST task (TSX 37)		260	
Using first event (TSX 37)		520	

All or nothing modules

The table below gives the memory usage for the all or nothing modules.

The all or nothing family	Bit memory (words)	Data (words)	Application zone (words)
8 discrete inputs	4	12	40
16 discrete inputs	8	12	50
4 discrete outputs	2	12	40
8 discrete outputs	4	12	40
8I/8O discrete	4	12	40
16I/12O discrete	16	20	100
32I/32O discrete	32	20	142

Analog modules

The table below gives the memory usage for the analog modules.

Analog family	Bit memory (words)	Data (words)	Application zone (words)
4 analog input module			
AEZ414	0	156	56
Additional cost for the first 4-analog input family module			120
8 analog input module			
AEZ801/AEZ802	0	212	72
Additional cost for the first 8-analog input family module			120
Analog output module			
ASZ200	0	52	40
ASZ401	0	100	59
Additional cost for the first analog output family module			120

Counting modules

The table below gives the memory usage for the counting modules.

Counting family	Bit memory (words)	Data (words)	Application zone (words)
CTY1A	16	108	64
CTY2A	32	212	106
Additional cost for the first counting channel			144
Additional cost for the first down counting channel			144
Additional cost for the first channel in CPT/DCPT			144

Communication modules

The table below gives the memory usage for the communication modules.

Communication family	Bit memory (words)	Data (words)	Application zone (words)
STZ010	0	36	168
SCP111/SCP112/SCP114 (on UC UTW)	0	40	763
FPP 20 on UC (Channel 0 UTW)	0	40	755
MDM 10	0	2528	12880

Memory usage for the modules on Premium

General

Note: This information is for a specific processor version. They can be subject to «slight» variations according to product developments.

The tables below give the size occupied in each of the zones and a fixed size to be added to the memory usage the first time an application-specific function is used, for each module type.

Processors

The table below gives the memory usage for the TSX 57 processor modules.

Processors	Bit memory (words)	Data (words)	Application zone (words)
P 57-1•	70	4714	1720
P 57-2•/3•/4•	70	4714	1784
Using FAST task (TSX 57)		520	
Additional cost for first module in configuration		600	
P 57-1•: per process control loop		500	
Additional cost for first loop			25000
P 57-2•/3•/4•: per process control loop		500	
Additional cost for first loop			5000

Discrete modules

The table below gives the memory usage for the all or nothing modules.

Discrete family	Bit memory (words)	Data (words)	Application zone (words)
Single discrete input family			
8 discrete inputs	4	100	100
16 discrete inputs	8	130	110
32 discrete inputs	16	230	120
64 discrete inputs	32	430	190
Additional cost for the first input family module			610
Single discrete output family			
8 discrete outputs	4	110	100
16 discrete outputs	8	160	110

Discrete family	Bit memory (words)	Data (words)	Application zone (words)
32 discrete outputs	16	280	120
64 discrete outputs	32	550	190
Additional cost for the first output family module			570
Discrete event input family			
16 discrete inputs (DEY 16FK)	8	220	130
Additional cost for the first input family module			680
Discrete Input/Output family safety			
12I/4O or 12I/4O(PAY)	16	128	200
Additional cost for the first DIS EVT input family module			1320
Mixed discrete input/output family			
16 inputs/12 outputs (DMY 28FK)	16	304	152
Additional cost for the first family module			1432
Mixed discrete			
Reflex 16 I/12 O (DMY 28RFK)	32	976	656
Additional cost for the first reflex mixed discrete family module			5596

Analog modules

The table below gives the memory usage for the analog modules.

Analog family	Bit memory (words)	Data (words)	Application zone (words)
Analog input families			
AEY414	4	430	160
AEY800	8	840	240
AEY1600	16	1670	430
Additional cost for the first analog input family module (AEY 414/800/1600)			2990
AEY810	8	888	248
AEY1614	16	1768	432
Additional cost for the first analog input family module (AEY 810/1614)			3056
AEY420	4	476	168

Analog family	Bit memory (words)	Data (words)	Application zone (words)
Additional cost for the first analog input family module (AEY 810/1614)			2080
Analog output family			
ASY410	4	430	160
Additional cost for the first ASY410 analog output module			1700
ASY800	8	744	248
Additional cost for the first ASY800 analog output module			1760

Counting modules

The table below gives the memory usage for the counting modules.

Counting family	Bit memory (words)	Data (words)	Application zone (words)
CTY2A module	32	410	170
CTY4A module	64	800	250
Additional cost for the first configured counting channel			1740
CTY2C module	48	672	184
Additional cost for the first configured counting channel			1992

Servo-motor modules

The table below gives the memory usage for the servo-motor modules.

Servo-motor family	Bit memory (words)	(words)	(words)
CAY•1	78	520	140
CAY•2	78	376	232
CAY33 channel 3	78	264	170
Additional cost for the first CAY•1 configured channel			2130
Additional cost for the first CAY•2/33 configured channel			3600
Additional cost for the first CAY•33 configured channel 3			3600

Step by step modules

The table below gives the memory used for the step by step modules.

Step by step family	Bit memory (words)	Data (words)	Application zone (words)
CFY11	29	323	104
CFY21	58	646	152
Additional cost for the first configured step by step channel			2368

Communication modules

The table below gives the memory usage for the communication modules.

Communication module family	Bit memory (words)	Data (words)	Application zone (words)
SCY21600 (Channel 0 UTW)	1	230	80
on SCY21600 (Channel 1 UTW)	1	450	40
Additional cost for the first channel configured in UTW			1280
ETY 110	1	431	256
Additional cost for the first ETY 110 configured channel			1984
ETY 120	1	48	136
Additional cost for the first ETY 120 configured channel			1368
ETY 210	1	434	400
Additional cost for the first ETY 210 configured channel			3424
IBY 100	1	450	40

Communication sub-modules

The table below gives the memory usage for the communication sub-modules.

Communication sub-module	Bit memory (words)	Data (words)	Application zone (words)
SCP111/ SCP112/ SCP114 (UTW)	1	60	580
on UC (Channel 0 UTW)			
FPP 20 on UC (Channel 0 UTW)	1	60	580
FPP 10 on UC (Channel 0 UTW)	1	40	870

AS-i module

The table below gives the memory usage for the AS-i module.

AS-i family	Bit memory (words)	Data (words)	Application zone (words)
SAY	3	373	176
Additional cost for the first ASi channel			2272

Weighing modules

The table below gives the memory usage for the weighing modules.

Weighing family	Bit memory (words)	Data (words)	Application zone (words)
AWY001	1	170	120
Additional cost for the first configured weighing channel			3920

TBX remote input/output modules

The table below gives the memory usage for the TBX remote input/output modules.

Remote input/output family	Bit memory (words)	Data (words)	Application zone (words)
Discrete inputs	8	152	88
Additional cost for the first configured base			1400
Discrete outputs	8	176	88
Additional cost for the first configured base			1320
Programmable	8	160	88
Additional cost for the first configured base			2304
Latch'state	8	160	88
Additional cost for the first configured base			1400
AES 400	2	270	104
ASS 200	2	270	104
AMS 620	4	508	112
Additional cost for the first configured base			3968

Momentum modules

The table below gives the memory usage for the Momentum modules

Momentum family	Bit memory (words)	Data (words)	Application zone (words)
Inputs	16	96	72
Additional cost for the first configured base			1384
Output	16	112	72
Additional cost for the first configured base			1256
Mixed	16	104	72
Additional cost for the first configured base			1424

Remote X bus modules

The table below gives the memory usage for the remote X bus module.

Remote X bus	Bit memory (words)	Data (words)	Application zone (words)
TSX REY 200 module	0	0	56

Memory size for advanced functions

Description

The tables below show the size of the code taken into the application (application zone) for each advanced function (OF) when one is called.

Functions in the same family share code (shared code). This shared code is taken into the PLC the first time a function from this family is called. The code specific to a function is taken in the first time this function is called.

Example

- The first time a function of the numerical conversion family is called, with the case of DBCD_TO_DINT, the code is taken into the application zone:
  - Shared code = 154 words
  - OF DBCD_TO_INT code = 149 words
- When another function of the numerical conversion family is called, in the case of DINT_TO_DBCD, the code is taken into the application zone:
  - OF DINT_TO_DBCD code = 203 words
  - If a function in the digital conversion family already called is called (DBCD_TO_DINT or DINT_TO_DBCD): no code taken in

Numerical conversions

The table below gives the memory usage for advanced conversion functions.

Numerical conversions	OF	Code size (in words)
Converting a 32 bit BCD number into a 32 bit integer	DBCD_TO_DINT	203
Converting a 32 bit BCD number into a 16 bit integer	DBCD_TO_INT	149
Converting a 32 bit integer into a BCD 32 bit number	DINT_TO_DBCD	203
Converting a 16 bit integer into a BCD 32 bit number	INT_TO_DBCD	75
Extracting the least significant word from'a double word	LW	33
Extracting the most significant word from'a double word	HW	33
Forming 'a double word with 2 words	CONCATW	33
	shared code	154

Instructions for bit strings

The table below gives the memory used for advanced bit string functions.

Bit strings	OF	Code size
Logic AND between two tables	AND_ARX	209
Copying a bit table into a double word table	BIT_D	248
Copying a bit table into a word table	BIT_W	205
Copying a bit table into a bit table	COPY_BIT	146
Copying a double word table into a bit table	D_BIT	196

Bit strings	OF	Code size
Add-in at one of a table	NOT_ARX	157
Logic OR between two tables	OR_ARX	209
Copying a word table into a bit table	W_BIT	195
Exclusive OR between two tables	XOR_ARX	209
Length in number of'elements	LENGTH_ARX	20
	shared code	427

Instructions for Word table

The table below gives the memory usage for advanced conversion functions on word tables.

Instructions for word tables	OF	Code size (in words)
Searching for the first element in a table equal to one value	FIND_EQW	75
Searching for the first element in a table greater than one value	FIND_GTW	75
Searching for the first element in a table less than one value	FIND_LTW	78
Searching for the highest value in a table	MAX_ARW	78
Searching for the smallest value in a table	MIN_ARW	74
Number of times a value occurs in a table	OCCUR_ARW	145
Rotate shift on the left of a table	ROL_ARW	150
Rotate shift on the right of a table	ROR_ARW	144
Table sorting (ascending or descending)	SORT_ARW	164
Partial search for the first element of'a table equal to one value	FIND_EQWP	77
Length in number of'elements	LENGTH_ARW	20
	shared code	162

Instructions for tables of double words

The table below gives the memory used for advanced conversion functions on tables of double words

Instructions for double word tables	OF	Code size (in words)
Searching for the first element in a table equal to one value	FIND_EQD	79
Searching for the first element in a table greater than one value	FIND_GTD	80
Searching for the first element in a table less than one value	FIND_LTD	95
Searching for the highest value in a table	MAX_ARD	95
Searching for the smallest value in a table	MIN_ARD	78
Number of times a value occurs in a table	OCCUR_ARD	163
Rotate shift on the left of a table	ROL_ARD	170
Rotate shift on the right of a table	ROR_ARD	178
Table sorting(ascending or descending)	SORT_ARD	
Partial search for the first element of'a table equal to one value	FIND_EQWP	77
Length in number of'elements	LENGTH_ARW	20
	shared code	162

Instructions for floating point tables

The table below gives the memory usage for advanced conversion functions on floating point tables.

Instructions for floating point tables	OF	Code size (in words)
Sum of the elements of'a table of real values	SUM_ARR	152
Searching for the first element equal to one value in a table	FIND_EQR	134
Searching for the first element equal to one a value in a table starting from'a row	FIND_EQRP	135
Searching for the first element greater than a value in a table	FIND_GTR	134
Searching for the first element less than a value in a table	FIND_LTR	134
Searching for the greatest value in a table	MAX_ARR	161
Searching for the smallest value in a table	MIN_ARR	162
Number of times a value occurs in a table	OCCUR_ARR	132
Rotate shift on the left of a table	ROL_ARR	167

Instructions for floating point tables	OF	Code size (in words)
Rotate shift on the right of a table	ROR_ARR	173
Table sorting (ascending or descending)	SORT_ARR	271
comparing 2 tables of real values	EQUAL_ARR	173
Table sorting (ascending or descending)	LENGTH_ARR	20
	shared code	124

Time management instructions

The table below gives the memory usage for advanced time management functions.

Dates, hours and periods	OF	Code size
Adding a period to a complete date	ADD_DT	519
Adding a period to a time of day	ADD_TOD	188
Converting a date into a string	DATE_TO_STRING	150
Day of the week	DAY_OF_WEEK	99
Deviation between two dates	DELTA_D	374
Deviation between two complete dates	DELTA_DT	547
Deviation between two times of day	DELTA_TOD	110
Converting data completely into a string	DT_TO_STRING	266
Subtracting a period from a complete date	SUB_DT	548
Subtracting a period from a time of day	SUB_TOD	186
Converting a period into a string	TIME_TO_STRING	413
Converting a time of day into a string	TOD_TO_STRING	156
Transferring a period into the form of hours-mins-secs	TRANS_TIME	211
Real-time clock function	SCHEDULE	700
	shared code	1703

Instructions for character strings

The table below gives the memory usage for advanced character string functions.

Dates, hours and periods	OF	Code size
Instructions for character strings		code size
Concatenating two strings	CONCAT	
Deleting a sub-string	DELETE	279
Searching for the first different character	EQUAL_STR	212
Finding a sub-string	FIND	225

Dates, hours and periods	OF	Code size
Inserting a sub-string	INSERT	287
Extracting from the left of a string	LEFT	38
String length	LEN	70
Extracting a sub-string	MID	44
Replacing a sub-string	REPLACE	365
Extracting from the right of a string	RIGHT	55
	shared code	418

Orphee functions

The table below gives the memory usage for Orphee functions.

Orphee functions	OF	Code size
Move to the left on 32, recovering moved bits	DSHL_RBIT	152
Move to the right on 32 with sign extension, recov. moved bits	DSHR_RBIT	152
Move to the right on 32 with 0 filling, recov. moved bits	DSHRZ_C	133
Move to the left on 16, with moved bit recovery	WSHL_RBIT	91
Move to the right on 16 with sign extension, recov. moved bits	WSHR_RBIT	103
Move to the right on 16 with 0 filling, recov. moved bits	WSHRZ_C	90
	shared code	173
Up/down counting with overshoot signaling	SCOUNT	617
Rotating to the left of'a word	ROLW	41
Rotating to the right of a word	RORW	
Rotating to the left of a double word	ROLD	49
Rotating to the left of a double word	RORD	49

Time delay functions

The table below gives the memory usage for the time delay functions.

Time delay functions	OF	Code size (in words)
Pulse output	FPULSOR	215
Trigger time delay	FTOF	272
Time delay trigger	FTON	217
Time delay pulse	FTP	245

Logarithmic, exponential and trigonometric functions

The table below gives the memory usage for logarithmic, exponential and trigonometric functions.

Logarithmic, exponential and trigonometric functions	OF	Code size (in words)
Natural logarithm	NL	0
Decimal logarithm	LOG	0
Exponential	EXP	0
Exponentiation of'a real by an integer	EXPT	523
Whole part	TRUNC	128
Cosine of an angle in radians	COS	0
Sine of an angle in radians	SIN	0
Tangent of an angle in radians	TAN	0
Arc cosine (result between 0 and pi)	ACOS	0
Arc sine (result between -pi/2 and pi/2)	ASIN	0
Arc tangent (result between -pi/2 and pi/2)	ATAN	0
Converting degrees into radians	DEG_TO_RAD	257
Converting radians into degrees	RAD_TO_DEG	247
	shared code	392

Process control functions

The table below gives the memory usage for process control functions.

Process control functions	OF	Code size (in words)
Mixed PID regulator	PID	1800
Pulse width modulation of a numerical value	PWM	600
PID output stage for discrete valve command	SERVO	1200
Dedicated operator dialog management on CCX17 of the PID	PID_MMI	4400
	shared code	573

Dialog operator functions

The table below gives the memory usage for the operator dialog functions.

Operator dialog functions	OF	Code size (in words)
Blocking entry of a variable on CCX17	Ask_msg,	46,5
Blocking entry of a variable on msg contained in CCX17	Ask_value,	46,5
Dynamic command key assigning	Assign_keyS,	46,5

Operator dialog functions	OF	Code size (in words)
Command of LED control	Control_leds,	46,5
Displaying an alarm contained in the CCX17	Display_alarm,	46,5
Displaying a group of messages contained in the CCX17	Display_GRP,	46,5
Displaying a message contained in the CCX17	Display_MSG,	46,5
Multiple entry of a variable on CCX17	GET_MSG,	46,5
Multiple entry of a variable on msg contained in CCX17	GET_VALUE,	46,5
Sending a command to the CCX17	PANEL_CMD,	46,5
Displaying an alarm msg contained in the PLC memory	SEND_alarm,	46,5
Displaying a message contained in the PLC memory	Send_msg	46,5
	shared code	573

Communication functions

The table below gives the memory usage for the communication functions.

Communication functions	OF	Code size (in words)
Reading standard language objects	READ_VAR	617
Writing standard language objects	WRITE_VAR	500
Sending/receiving UNI-TE requests	SEND_REQ	438
Sending and/or receiving data	DATA_EXCH	375
Sending a character string	PRINT_CHAR	476
Reading request for a character string	INPUT_CHAR	625
Sending and/or receiving a character string	OUT_IN_CHAR	531
Sending a telegram	SEND_TLG	219
Receiving a telegram	RCV_TLG	172
Request to stop a function in progress	CANCEL	
	shared code	506
Right shift of 1 octet in a table of octets	ROR1_ARB	235
Immediate server	SERVER	32
	shared code	648
Write 1K of message handling		32
	shared code	936
Read 1K of message handling	READ_ASYN	32
	shared code	920

Movement command functions

The table below gives the memory usage for advanced movement functions.

Movement command functions	OF	Code size (words)
1 axis automatic movement command	SMOVE	24
Multi-axis automatic movement command	XMOVE	32

Data storage

The table below gives the memory usage for the data storage functions.

Data storage	OF	Code size (words)
Initializing the zone for'storage on the PCMCIA card	SET_PCMCIA	24
Write data on the PCMCIA card	WRITE_PCMCIA	24
Read data on the PCMCIA card	READ_PCMCIA	24
	shared code	288

User-defined exchange function

The table below gives the memory usageage for the user-defined exchange functions.

User-defined exchanges	OF	Code size (words)
Read status parameters	READ_STS	0
Read adjustment parameters	READ_PARAM	0
Update adjustment parameters	WRITE_PARAM	0
Save adjustment parameters	SAVE_PARAM	0
Restore adjustment parameters	RESTORE_PARAM	0
Update command parameters	WRITE_CMD	0
(1) OF specific, the code is counted in the volume of the I/O module.		

DFB of diagnostics

The first time one of the diagnostics' DFBs is programmed, 200 words are reserved in the application program zone.

The table below shows the size of the code taken into the application for each type of diagnostics' DFB (in the program zone) and the size occupied per instance in the data and program zones.

Diagnostics' DFB (sizes in words)	DFB type size	DFB type code size	Data size per instance
IO_DIA	800	64	72
ALRM_DIA	608	40	48
NEPO_DIA	15184	128	136
TEPO_DIA	10896	128	136
EV_DIA	1144	48	56
MV_DIA	2616	80	88
ASI_DIA	7912	304	312

Method for calculating the number of instructions

General

This method is used to calculate the number of basic (assembler level) Boolean or numeric instructions.

Calculating the number of Boolean instructions

The number of the following elements is taken into account in this calculation:

- unitary Boolean operations: load (LD), AND, OR, XOR,ST,etc
- closing brackets (or ladder convergences: vertical convergence links)
- comparison (AND[...], OR[...]...) blocks and operate ([...])

The operators NOT, RE and FE should not be counted as a boolean instruction.

Example:

```
LD %M0
AND (%M1
OR %M2
)
ST %M3
```

= 5 boolean instructions

Calculating the number of Numeric instructions

The number of the following elements is taken into account in this calculation:

- assignments (:=)
- load of the first value after:=
- arithmetic instructions (+, -, *, /, <, =,...), operations on words or tables of words, double words, floating points)
- logic instructions on words
- functions(OF, EQUAL,…) irrespective of the number of parameters
- function blocks (or function block instruction)

Example:

```
%MW0:=(%MW1+%MW2)*%MW3;
```

counted instructions:

:=

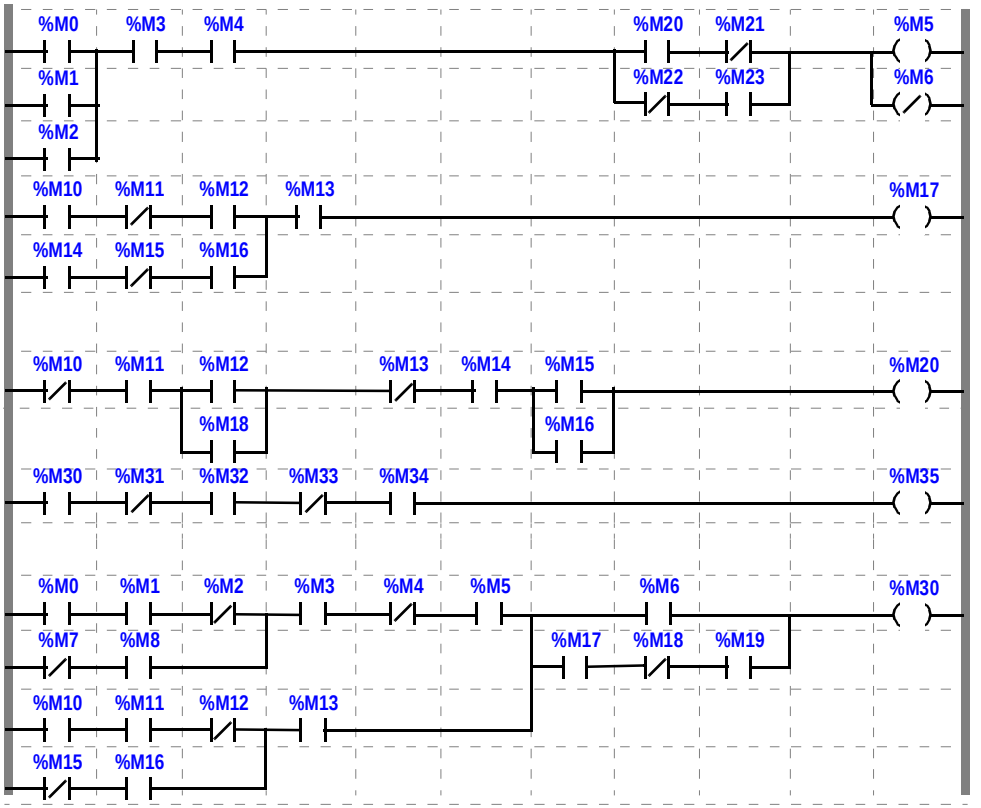
%MW1 (corresponds to the load instruction in the accumulator)

+

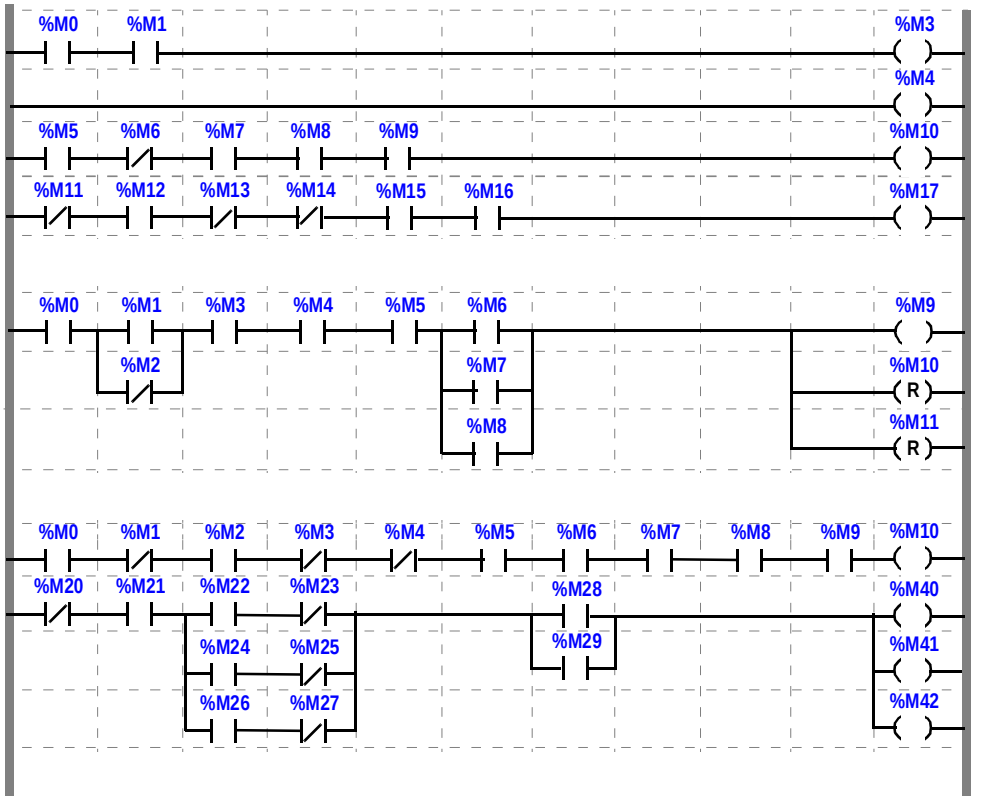
*

i.e. 4 instructions.

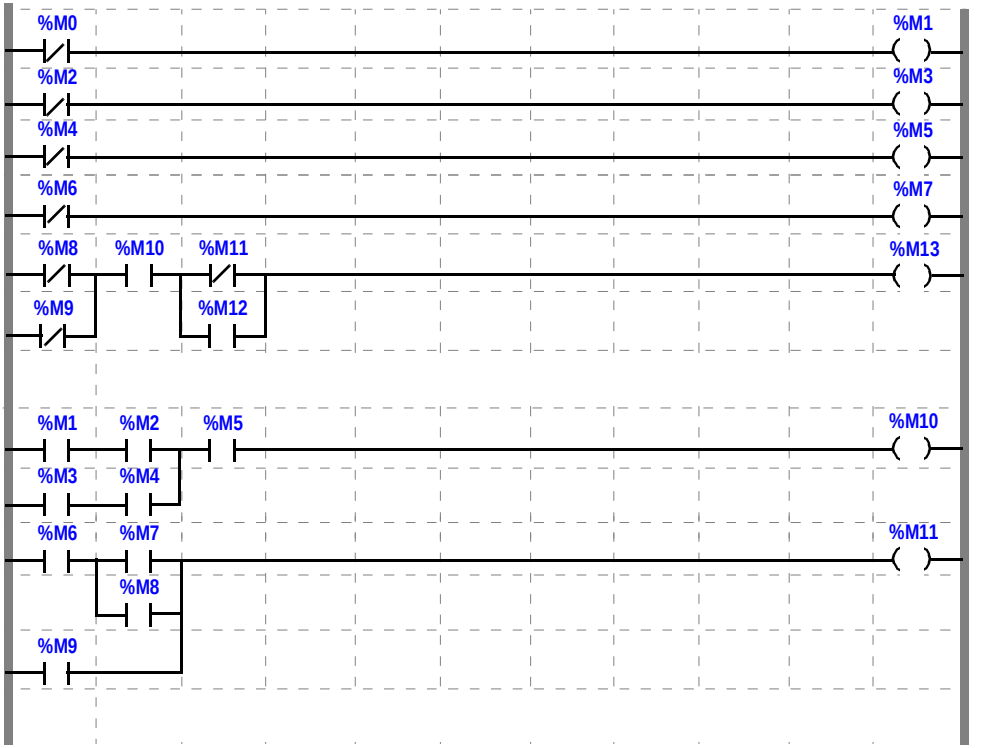
Example Example of program comprising 65% boolean and 35% numeric:



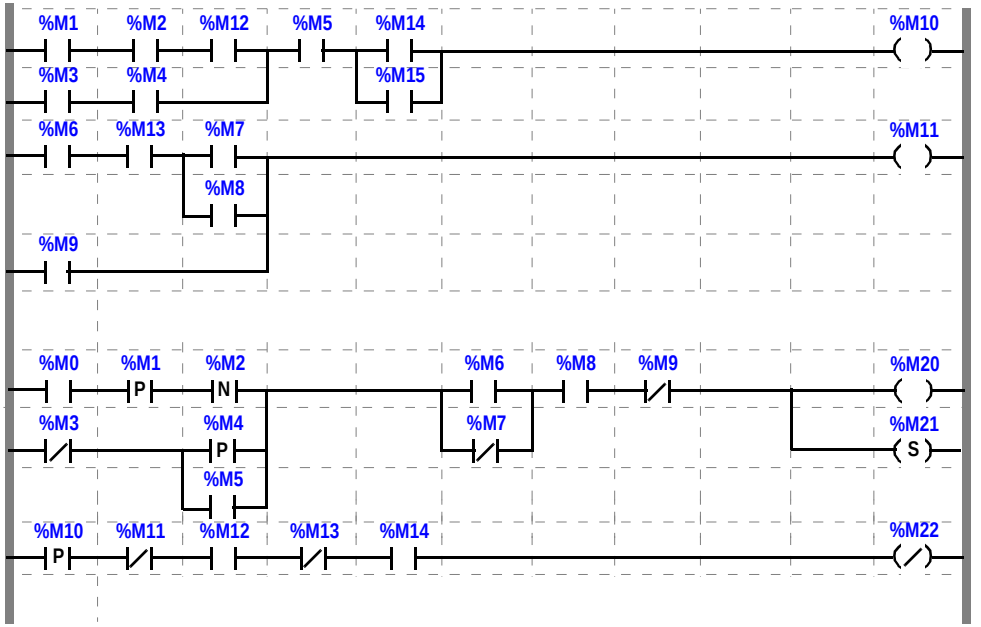
Example (continued)



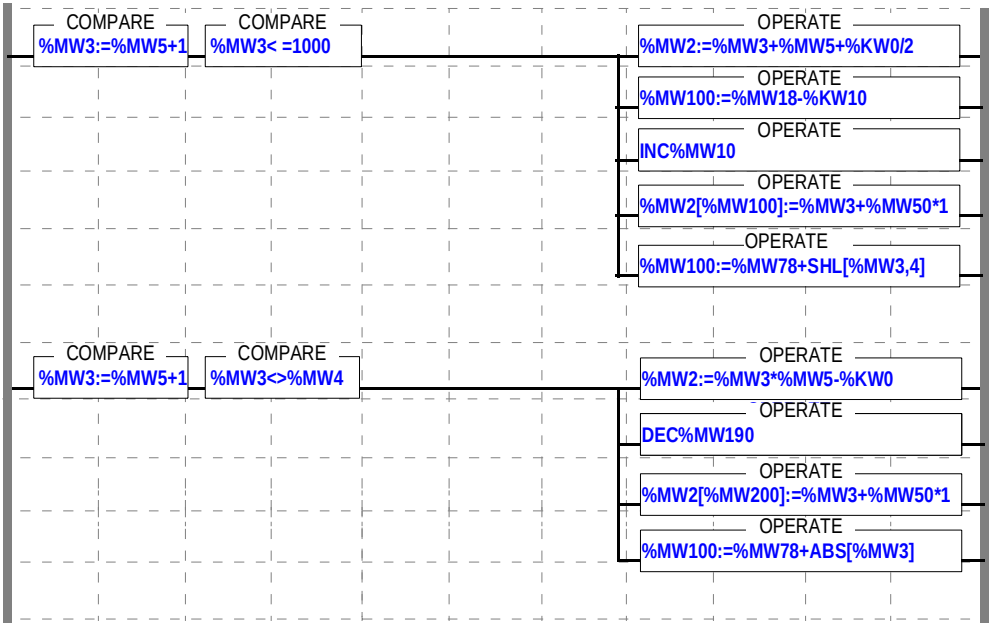
Example (continued)



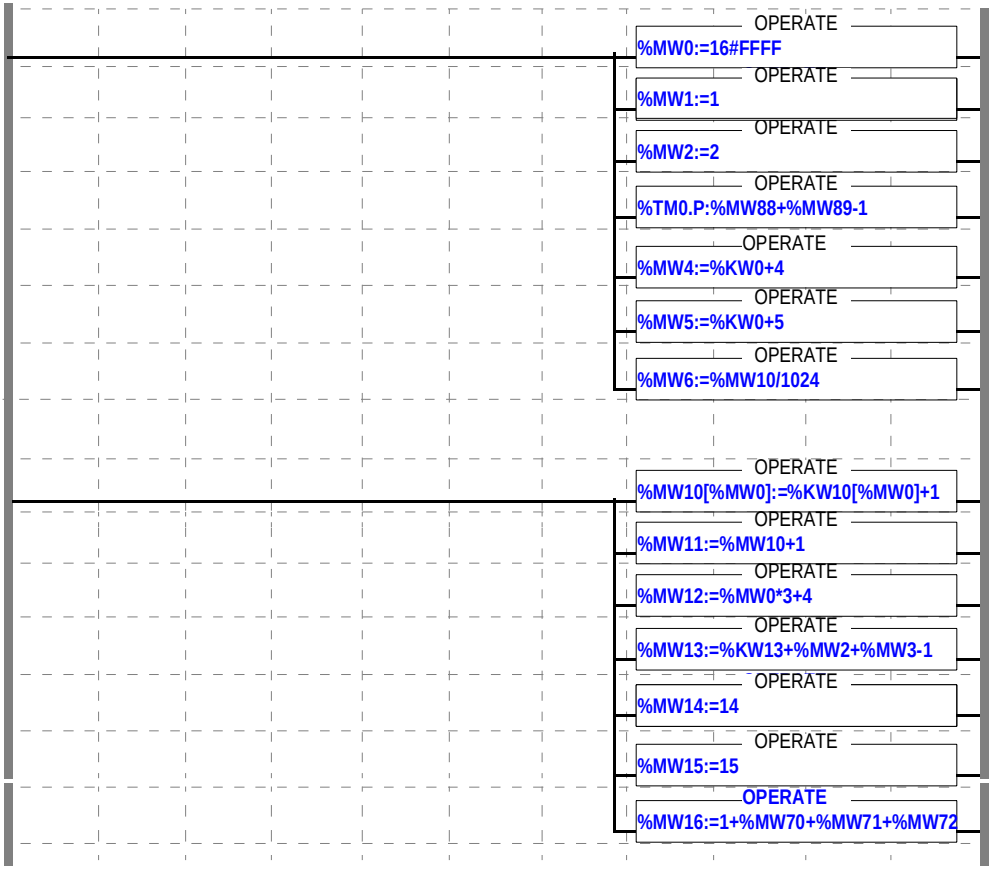
Example (continued)



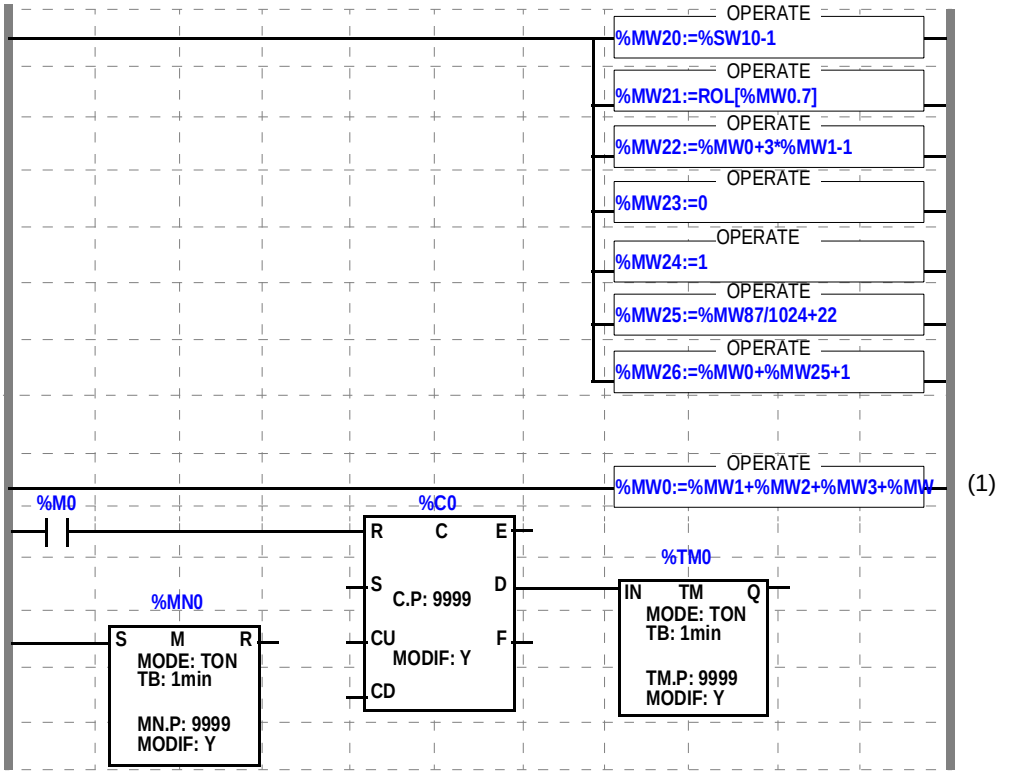
Example (continued)



Example (continued)



Example (continued)



(1):%MW0:=%MW1+%MW2+%MW3+%MW4+%MW5+%MW6+%MW7+%MW8+%MW9+%MW10+1

Memory Usage

	Number of instructions	%	
Boolean without edge	187	54,05%	64,16%
Boolean with edge	4	1,16%	
Operate block	31	8,96%	

	Number of instructions	**%**	
Function block	3	0,87%	35,84%
Simple arithmetic (+,-,:=,AND,etc)	111	32,08%	
Indexed arithmetic	4	1,16%	
*, /	6	1,73%	
Immediate values	24		
Total	346		100,00%

Index