Tani Item Syntax

Tani GmbH

Networks in industry

.

The name Tani stands for communication in industrial production.
The focus is on communication systems:

- *OPC Server for widespread PLCs*
- *Equipment and software for connecting PLCs, SCADA systems and databases.*
- *Fieldbus diagnostic systems.*

Tani Item Syntax

Created: Juli 2023 in Nuremberg, Germany

Tani GmbH
Freiligrathstrasse 12
D-90482 Nuremberg
Internet: https://tanindustrie.de
Email: info@tanindustrie.de
Tel.: +49 911 980 373 54
HRB: Amtsgericht Nürnberg 29562
USt-Id: DE 289 906 852

Table of contents

1 Item Syntax

This chapter describes the Item syntax for the corresponding connection (Access Path) that can be used to create new items (tags) .

General information

- Browsing Tree
- System Tree
- Redundancy

Item Syntax for different PLCs

- S7 Item Syntax
- S5 Item Syntax
- Modbus Item-Syntax
- PLC-5 / SLC Item Syntax
- MELSEC-Q Item Syntax
- IEC 60870-5 Item Syntax
- KNX Item Syntax
- Send/Receive Item Syntax
- Configuration Item Syntax

Information for PLCs with online symbols

- S7-1200/S7-1500, Rockwell CompactLogix/ControlLogix, BACnet

The following chapters are generally:

- Bit mask
- Arrays
- Suffixes

1.1 S7 Item Syntax

The S7 item syntax is set up as shown below:

<Area><Data type><Start address>[.Array size][Suffix]

If the data type is BOOL, the bit number is required:

<Area><Data type><Start address><.Bit number>[.Array size]

If the data type is STRING, the string length is required:

<Area><Data type><Start address><.String length>[.Array size][Suffix]

Legend: <> mandatory [ ] optional

<Area>

	Syntax	Orientation[1]	Access Rights	Notes
Data block	**DBx.** **V synonym for DB1**	BYTE	Read / write	With data blocks, the specification of a block number x is required (x = 1 to 65535). A dot must appear after the block

Instance block	**DIx.**	BYTE	Read / write	number.
Flag	**M or F**	BYTE	Read / write	
Timer	**T**	WORD	Read	directly followed by the timer number. The data type is REAL, with suffix S5T STRING
Counter	**Z or C**	WORD	Read / write	directly followed by the counter number.
Input	**E or I**	BYTE	Read	
Output	**A or O or Q**	BYTE	Read / write	
I/O Input	**PE or PI**	BYTE	Read	
I/O Output	**PA or PO or PQ**	BYTE	Read / write	

[1]BYTE-oriented means that a byte is addressed for each physical address.
WORD-oriented means that a word (16 bits) is addressed for each physical address.

<Data type>

Type	**Syntax**	DB / DI	M	I/O	PI /PO	C/T	with Array	useful Suffixes
BIT VT_BOOL	**X**	DB5.X4.3	MX1.3	---	---	---	---	---
BIT VT_BOOL		DB5.4.3	M1.3	E4.3 I4.3 A4.3 O4.3	PE4.5 PI4.5 PA1.3 PO1.3	---	---	---
BYTE VT_UI1	**B** **Byte**	DB5.B2 DB5.Byte2	MB4 MByte4 FB4 FByte4	EB4 EByte4 IB4 IByte4 AB5 AByte5 OB5 OByte5	PEB4 PEByte4 PIB4 PIByte4 PAB5 PAByte5 POB5 POByte5	---	DB5.B2.4 DB5.Byte2.4 MB4.3 MByte4.3 FB4.4 FByte4.4 POB5.3 etc.	KF BCD
WORD VT_UI2 VT_I4*	**W** **Word**	DB5.W3 DB5.Word3	MW4 MWord4 FW4 FWord4	EW4 EWord4 IW4 IWord4 AW5 AWord5 OW5 OWord5 QWord5	PEW4 PEWord4 PIW4 PIWord4 PAWord5 PAWord5 POW5 POWord5	C5 Z5 T5	DB5.W3.2 DB5.Word3.2 MW4.2 EWord4.2 PAWord5.3 C5.3 Z5.10 T5.2 etc.	KF BCD KT S5T TR D SWAP
INT VT_I2	**I** **Int**	DB5.I3 DB5.Int3	MI4 MInt4 FI4 FInt4	EI4 EInt4 II4 IInt4 AI5 AInt5 OI5	PEI4 PEInt4 PII4 PIInt4 PAI5 PAInt5 POI5	---	DB5.I3.2 DB5.Int3.2 MI4.4 FInt4.3 AInt5.3 OI5.2 OInt5.5	SWAP

				OInt5	POInt5		POInt5.4 etc.	
REAL VT_R4	**R** **Real**	DB5.R2 DB5.REAL 2	MR4 MREAL4	ER4 EREAL4 IR4 IREAL4 AR4 AREAL4 OR4 OREAL4	PER4 PEREAL4 PIR4 PIREAL4 PAR4 PAREAL4 POR4 POREAL4	----	DB5.R2.2 DB5.REAL 2.4 MR4.5 ER4.4 AREAL4.5 PER4.2 PAR4.7 POR4.3 etc.	KG SWAP
STRING VT_BSTR	**S** **String**	DB5.S1.80 DB5.String 1.80	MS2.80 MString2.8 0	---	---	---	DB5.S1.80 .5 DB5.String 1.80.5 MS2.80.3 MString2.8 0.3	KA[1]
S7- STRING VT_BSTR	**G** **SS**	DB5.G2.80 DB5.SS2. 80	MG2.80 MSS2.80	---	---	---	DB5.G2.80 .5 DB5.SS2. 80.5 MG2.80.5 MSS2.80. 5	---
S5- STRING VT_BSTR	**SF**	DB5.SF2.8 0	FSF2.80	---	---	---	DB5.SF2.8 0.5 MSF2.80.5	---
DOUBLE WORD VT_UI4 VT_R8*	**DW** **Dword**	DB5.DW3 DB5.Dwor d3	MDW4 MDWord4 FDW4 FDWord4	EDW4 EDWord4 IDW4 IDWord4 ADW5 ADWord5 ODW5 ODWord5 QDW5	PEDW4 PEDWord 4 PIDW4 PIDWord4 PADW5 PADWord 5 PODW5 PODWord 5	---	DB5.DW3 MDW4.2 FDW4.2 EDW4.4 ADWord5. 8 PEDW4.8 PADW5.4 PODW5.4 etc.	KF BCD KG T TOD SWAP
DOUBLE INT VT_I4	**D** **DI** **DInt**	DB5.D3 [2] DB5.DI3 DB5.DInt3	MDI4 MDInt4 FDI4 FDInt4	EDI4 EDInt4 IDI4 IDInt4 ADI5 ADInt5 ODI5 ODInt5 QDI5	PEDI4 PEDInt4 PIDI4 PIDInt4 PADI5 PADInt5 PODI5 PODInt5	---	DB5.D3.2 DB5.DI3.2 DB5.DInt3. 4 MDI4.5 EDI4.5 ADI5.2 PEDI4.5 PADInt5.2 etc.	BCD T TOD SWAP
QUAD WORD	**QW** **Qword**	DB5.QW3 DB5.Qwor	MQW4 MQWord4	EQW4 EQWord4	PEQW4 PEQWord	---	DB5.QW3 MQW4.2	KF BCD

VT_UI8		d3	FQW4 FQWord4	IQW4 IQWord4 AQW5 AQWord5 OQW5 OQWord5 QQW5	4 PIQW4 PIQWord4 PAQW5 PAQWord 5 POQW5 POQWord 5		FQW4.2 EQW4.4 AQWord5. 8 PEQW4.8 PAQW5.4 POQW5.4 etc.	SWAP
QUAD INT VT_I8	**Q** **QI** **QInt**	DB5.Q3 DB5.QI3 DB5.QInt3	MQI4 MQInt4 FQI4 FQInt4	EQI4 EQInt4 IQI4 IQInt4 AQI5 AQInt5 OQI5 OQInt5 QQI5	PEQI4 PEQInt4 PIQI4 PIQInt4 PAQI5 PAQInt5 POQI5 POQInt5	---	DB5.Q3.2 DB5.QI3.2 DB5.QInt3. 4 MQI4.5 EQI4.5 AQI5.2 PEQI4.5 PAQInt5.2 etc.	SWAP
DOUBLE VT_R8	**QR** **QReal**	DB5.QR2 DB5.QRE AL2	MQR4 MQREAL4	EQR4 EQREAL4 IQR4 IQREAL4 AQR4 AQREAL4 OQR4 OQREAL4	PEQR4 PEQREAL 4 PIQR4 PIQREAL4 PAQR4 PAQREAL 4 POQR4 POQREAL 4	----	DB5.QR2. 2 DB5.QRE AL2.4 MQR4.5 EQR4.4 AQREAL4. 5 PEQR4.2 PAQR4.7 POQR4.3 etc.	SWAP
Date and Time VT_DATE	**DT**	DB5.DT3	MDT4	EDT4 IDT4 ADT5 ODT5 QDI5	PEDT4 PIDT4 PADT5 PODT5	---	DB5.DT3.2 MDT4.2 EDT4.2 PEDT4.2 etc.	ISO

[1] KA Suffix is not possible with string arrays

[2] Caution! Danger of mix up with S5 syntax for bit (DB5D1.1)

Notes on counters and timers
Counters and timers are always addressed by words. For this reason, the specification of a data type is not required!
The start address directly follows the “T” or “C/Z” area. Timers can only be read! Counters can be read and written.
Timer values are indicated in seconds (e.g., T = 0.7 => T = 0.7 s = 700 ms).
Counters are represented in decimals (0 to 999).

<Start address>

The start address specifies the addresses starting at which can be read or written.
Example: DB5.DW6: Double word 6 of the data block 5 is the start address.
Example: MB17: Flag byte 17 is the start address.
If the start address is a certain bit, the bit number must also be specified.

<.Bit number>

The bit number must always be specified when the data type is BOOL.
Example: I4.3: bit 3 of input byte 4 – an input bit is addressed here.
Example: MX12.1: Bit 1 of flag byte 12 – a flag bit is addressed here.

[.Array size]

An array (i.e., field, row, data area) is a series of equal elements. An array combines several units of one data type into a field. If, for example, several words are read out from a data block, this is called an array of words. To create an array, the length of the array is added to the standard syntax separated by a dot.
Example: DB10.REAL2.5.
For more information on arrays, see also Arrays.

[Suffix]

A value can be represented in another format with the aid of a suffix.
For more information on suffixes, see also Suffixes.

1.2 S5 Item Syntax

The S5 item syntax is set up as shown below:

<Area><Data type><Start address>[.Array size][Suffix]

If the data type is BOOL, the bit number is required:

<Area><Data type><Start address><.Bitnr.>[.Array size]

If the data type is STRING, the string length is required:

<Area><Data type><Start address><.String length>[.Array size][Suffix]

Legend: <> mandatory [ ] optional

<Area>

	Syntax	Orientation[1]	Access rights	Notes
Data block	**DBx**	WORD	Read / write	With blocks, the specification of a block number x is required (x = 1 to 65535). After the block number no dot is required.
Extended data block	**DXx**	WORD	Read / write	
Flag	**M oder F**	BYTE	Read / write	
Timer	**T**	WORD	Read	
Counter	**Z oder C**	WORD	Read	
Input	**E oder I**	BYTE	Read	
Output	**A oder O oder Q**	BYTE	Read / write	
Periphery	**P**	BYTE	Read / write	
Extended Periphery	**OB**	BYTE	Read / write	
System area		WORD	Read	
Absolute memory	**AS**	WORD	Read	

cells				

[1]BYTE-oriented means that a byte is addressed for each physical address.
WORD-oriented means that a word (16 bits) is addressed for each physical address.

<Data type> for Data blocks and extended Data blocks

Type VT_Type	**Syntax**	Sample	with Array	useful Suffixes
BIT VT_BOOL	**D**	DB5D4.12	---	---
BYTE VT_UI1	**DB**	DB5DB3	DB5DB3.5	KF, BCD
LEFT BYTE VT_UI1	**DL**	DB5DL4	DB5DL4.2	---
RIGHT BYTE VT_UI1	**DR**	DB5DR2	DB5DR2.5	---
WORD VT_UI2	**DW**	DB5DW4	DB5DW4.5	KF, BCD, KT, S5T, TR, SWAP
DOUBLE WORD VT_UI4	**DD**	DB5DD3	DB5DD3.2	KF, BCD, KG, T, TOD, SWAP
QUAD WORD VT_UI8	**DQ**	DB5DQ3	DB5DQ3.2	KF, BCD, SWAP
STRING VT_BSTR	**S**	DB5S2.3	DB5S2.80.3	KA [1]
S7-STRING VT_BSTR	**SS**	DB5SS2.3	DB5SS2.80.3	---
S5-STRING VT_BSTR	**SF**	DB5SF2.3	DB5SF2.80.3	---

<Data type> for all other areaes

	Syntax	M or F	I/O	P / OB	C / T / RS / AS	with Array	useful Suffixes
BIT VT_BOOL		M4.3 F4.3	E4.3 I4.3 A4.3 Q4.3	---	---	---	---
BYTE VT_UI1	**B**	MB4 FB4	EB4 IB4 AB5 QB5	PB4 OB4	---	MB4.3 FB4.4 EB4.4 IB4.5 AB5.2 QB5.5 PB4.2 OB4.3	KF BCD
WORD VT_UI2	**W**	MW4 FW4	EW4 IW4 AW5 QW5	PW2 OW2	C5 T5 RS4 BS4 AS5	MW4.2 FW4.4 EW4.5 IW4.2 AW5.5 QW5.2 PW2.10 OW2.3	KF BCD KT S5T TR SWAP

						C5.5 T5.3 RS4.2 BS4.7 AS5.2	
DOUBLE WORD VT_UI4	**D**	MD4 FD4	ED4 ID4 AD5 QD5	PD5 OD5	---	MD4.4 FD4.2 ED4.6 ID4.2 AD5.6 QD5.2 PD5.4 OD5.9	KF BCD KG T TOD SWAP
QUAD WORD VT_UI8	**Q**	MQ4 FQ4	EQ4 IQ4 AQ5 QQ5	PQ5 OQ5	---	MQ4.4 FQ4.2 EQ4.6 IQ4.2 AQ5.6 QQ5.2 PQ5.4 OQ5.9	KF BCD SWAP
STRING VT_BSTR	**S**	MS4.80 FS4.80	---	---	---	MSS4.80.3 FSS4.80.3	KA [1]
S7-STRING VT_BSTR	**SS**	MSS4.80 FSS4.80	---	---	---	MSS4.80.3 FSS4.80.3	---
S5-STRING VT_BSTR	**SF**	MSF4.80 FSF4.80	---	---	---	MSF4.80.3 FSF4.80.3	---

[1] KA Suffix is not possible with string arrays

Notes on counters and timers
Counters and timers are always addressed by words. For this reason, the specification of a data type is not required!
The start address directly follows the “T” or “C/Z” area. Timers can only be read! Counters can be read and written.
Timer values are indicated in seconds (e.g., T = 0.7 => T = 0.7 s = 700 ms).
Counters are represented in decimals (0 to 999).

<Start address>

The start address specifies the addresses starting at which can be read or written.
Example: DB5.DW6: Double word 6 of the data block 5 is the start address.
Example: MB17: Flag byte 17 is the start address.
If the start address is a certain bit, the bit number must also be specified.

<.Bit number>

The bit number must always be specified when the data type is BOOL.
Example: I4.3: bit 3 of input byte 4 – an input bit is addressed here.
Example: FX12.1: Bit 1 of flag byte 12 – a flag bit is addressed here.

Note:
Writing to bits of a S5 PLC is not permitted. Please read the byte/word, set the bit and write the byte/word back.

[.Array size]

An array (i.e., field, row, data area) is a series of equal elements. An array combines several units of one data type into a field. If, for example, several words are read out from a data block, this is called an array of words. To create an array, the length of the array is added to the standard syntax separated by a dot.
Example: DB10DW2.5.
For more information on arrays, see also Arrays.

[Suffix]
A value can be represented in another format with the aid of a suffix.
For more information on suffixes, see also Suffixes.

1.3 Modbus Item Syntax

The Modbus item syntax is set up as shown below:

<Area><Data type><Start address>[.Array size][Suffix]

If the data type is BOOL, the bit number is required:

<Area><Data type><Start address><.Bit number>[.Array size]

The following item syntax allows to address a different UnitID as set up at the connection parameters:

[UnitID.]<Area><Data type><Start address><.Bit number>[.Array size][Suffix]

Legend: <> mandatory [ ] optional

[UnitID.]
The UnitId item syntax is specified for an item with the literals "ID" followed by a number and a dot.
The range is 0 - 255.
If the UnitID is not present in the Itemsyntax, the parameterized UnitID is transferred to the PLC.
Samples:
Id1.40001
Id2.R2
Id3.S5.30

<Area>

	Syntax	Syntax with number	Orientation[1]	Access Rights
Discrete Inputs	**I** **E** **DI** **DE**	1xxxxx	BIT	Read
Discrete Outputs	**A** **O** **Q** **DA** **DO** **DQ**	0xxxxx	BIT	Read / write
Input Register	**ER** **IR**	3xxxxx	WORD	Read
Register (Holding Register)	**R** **HR**	4xxxxx	WORD	Read / write
Discrete Inputs Oktal [2]	**J**	-	BIT	Read

Discrete Outputs Oktal [2]	**P**	-	BIT	Read / write

[1] BIT-oriented means that one bit is addressed for each physical address.
WORD-oriented means that one word (16 bits) is addressed for each physical address.
[2] Entry of the start address is octal and the numbers 8 and 9 are invalid characters. Internally, the address is handled decimally and must be considered for logger and status entries.

Areas can either be addressed via the above stated **alphabetic sequence** or via a **number**. This means that a discrete input can be addressed by the abbreviation "E" the same as with the number "1", discrete outputs by letter "O" or number 0, Input registers by "IR" or "3", Registers by "R" or "4".

<Data type>

Data type VT-Typ	**Syntax**	Syntax R	Syntax I / O	Syntax ER	with Array	useful Suffixes
BIT VT_BOOL	**X** [1]	RX5.2 [1] HRX5.2 [1] 4X5.2 [1]	E5 I5 DE5 DI5 100005 A5 O5 Q5 DA5 DO5 DQ5 000005	ERX5.2 [1] 3X5.2 [1]	I1.10 O2.5	
LEFT CHAR RIGHT CHAR VT_I1	**LC** **RC**	RLC5 RRC5	--- ---	ERLC5 ERRC5	RLC5.2 RRC5.2	
LEFT BYTE RIGHT BYTE VT_UI1	**LB** **RB**	RLB5 RRB5	--- ---	ERLB5 ERRB5	RLB5.2 RRB5.2	
INT VT_I2	**ohne** **I**	R5 HR5 45 RI5 HRI5 4I5	---	ER5 IR5 35 ERI5 IRI5 3I5	R5.2 HR5.2 45.2 ERI5.2 IRI5.2 3I5.2	SWAP [2]
WORD VT_UI2	**W**	RW5 HRW5 4W5	----	ERW5 IRW5 3W5	RW5.2 HRW5.2 4W5.2 ERW5.2 IRW5.2 3W5.2	SWAP [2]
DOUBLE INT VT_I4	**D** **DI**	RD5 HRD5 4D5 RDI5 HRDI5 4DI5	---	ERD5	RD5.2 HRD5.2 4D5.2	TOD SWAP [2]
DOUBLE	**D**	RDW5	---	ERDW5	RDW5.2	TOD

WORD VT_UI4		HRDW5 4DW5		IRDW5 3DW5	HRDW5.2 4DW5.2 ERDW5.2 IRDW5.2 3DW5.2	SWAP [2]
QUAD INT VT_I8	**Q** **QI**	RQ5 HRQ5 4Q5 RQI5 HRQI5 4QI5	---	ERQ5 IRQ5 3Q5	ERQ5.2 IRQ5.2 3Q5.2	SWAP [2]
QUAD WORD VT_UI8	**QW**	RQW5 HRQW5 4QW5	---	ERQW5 IRQW5 3QW5	RQW5.2 HRQW5.2 4QW5.2 ERQW5.2 IRQW5.2 3QW5.2	SWAP [2]
REAL VT_R4	**R**	RR5	----	ERR5	RR5.2 ERR5.2	SWAP [2]
DOUBLE VT_R8	**QR**	RQR5	---	ERQR5	RQR5.2 ERQR5.2	SWAP [2]
STRING VT_BSTR [2]	**S**	RS5.80	----	ERS5.80	RS5.80.3	SWAP [2] KA [3]

[1] Diskrete Inputs and Outputs are one single bit in the PLC. When reading from Register or Input Registers, the whole register is read and the bit is extracted.

Note:
Writing of individual bits in the register and input register area is possible if the writing of bits in the word is activated in the configuration of the connection.
The entire register is then read, the bit is set or deleted and the register is written back again.

[2] The suffix SWAP is possible for a string to give the bytes the correct order. For the other data types all bytes are changed in the order.

[3] KA Suffix is not possible with string arrays

<Start address>

The start address specifies the address starting at which read or write accesses begins.
Example:
ER5 -> Input Register 5
O12 -> Output 12

<.Bit number>

The bit number must always be specified when the data type is BOOL!
Example: HRX5.2: Bit 2 of holding register 5

[.Array size]

Arrays are created to combine several units of one data type together in one field.
Example: HRD5.3
For more information on arrays, see also Arrays.

[Suffix]

A value can be represented in another format with the aid of a suffix.
For more information on suffixes, see also Suffixes.

1.4 PLC-5 / SLC Item Syntax

The item syntax for PLC-5 and SLC is set up as shown below:

<Area>[File number]<Start address>[Array size][Tani-Suffix]

<Area>[File number]<Start address>[Array size][RSLinX-Suffix]

Legend: <> mandatory [ ] optional

<Area>

File Type	**Syntax**	Orientation [1]	Access rights	Default File number	Address
Output	**O**	BIT	Read / write	0	octal
Input	**I**	BIT	Read	1	octal
Integer	**N**	WORD	Read / write	7	decimal
Binary	**B**	WORD	Read / write	3	decimal
Float	**F**	DOUBLE WORD	Read / write	8	decimal
String	**ST**	SLC-String	Read / write	9	decimal
SFC-Status	**SC**	WORD	Read / write	3	decimal
Status	**S**	WORD	Read / write	2	decimal
Timer	**T**	WORD	Read / write	4	decimal
Counter	**C**	WORD	Read / write	5	decimal
Control	**R**	WORD	Read / write	6	decimal
ASCII	**A**	WORD	Read / write	9	decimal
Long Integer	**L**	DOUBLE WORD	Read / write	9	decimal

[1] BIT-oriented means that one bit is addressed for each physical address.
WORD-oriented means that one word (16 bits) is addressed for each physical address.
DOUBLE WORD-oriented means that one double word (32 bits) is addressed for each physical address.

[File number]

Specification of the file number is optional. If it is not specified, the default file number is used. See column marked **Default File Number** in the table.

<Start address>

The start address specifies the address starting at which read or write accesses begin. The start address can consist of 2 pieces of information. Word (floating) number and when a single bit is accessed, then the bit number. The word number can be omitted with a bit. The word number or the floating number is introduced by a colon (:). The bit number is then introduced with a slash (/) <:word> or <:float> or </bit>.

The address is octal for some areas. It is decimal for others. See column labeled Address.

Either the word number, the float number or the bit number is specified as the start address.
<:word> or <:float> or </bit>

The following syntax is used to address a certain bit within a word.

<:word/bit>

NOTE:
When bits are write-accessed, the whole word is written!

Syntax	Description
O:0	Word 0 in Output file 0
O:0/12	Bit 10 (12 octal = 10 decimal) in output file 0
O/12	Bit 10 (12 octal = 10 decimal) in output file 0
I:37	Word 31 (37 octal = 31 decimal) in input file 1
I4:37/2	Bit 2 in Word 31 (37 octal = 31 decimal) in input file 4
I:1/0	Bit 0 in Word 1 in input file 1
B3/26	Bit 26 in binary file 3
B12:5/15	Bit 15 in word 5 of binary file 12
F8:0	Float 0
N23:4	word 4 of integer file 23
N23:4/2	Bit 2 in word 4 of integer file 23 = Bit 66 in integer file 23
N23/66	Bit 66 in integer file 23

[Array size]

Arrays are created to combine several units of one data type in a field together. Arrays are only possible for word areas and float areas. The array size is initiated with a #.
Examples: N23:4#10

[Tani-Suffix]

A value can be represented in another format with the aid of a suffix. If no suffix is specified, the formats from the column Orientation apply.
For more information on suffixes, see also Suffixes.

[RsLinx-Suffix]

The alternative syntax allows you to use one or more suffixes. Every suffix is initiated. The order is arbitrary.

NOTE:
The combination of RSLINX-Suffixes and Tani-Suffixes is not possible.

Suffix	Description	File Types
,N	Values are represented with sign. (Default is: no sign)	I, O, N, B, S, A
,M	Values are represented in Motorola-Format. (Default is: Intel-Format)	I, O, N, B, S, A, F
,L<xxx>	The len specifier for RSLinX Format. xxx is the count of array length	all
,SC<xxx>	Values are interpreted as a zero-terminated string. The maximum length is XXX.	N, A

1.5 MELSEC-Q Item Syntax

Two syntax versions are available for setting up items.

1. Simple version:

<Area><Start address>[.Array size][Suffix]

2. Expanded version:

<Area><.Type><Start address>[.Array size][Suffix]

Legend: <> mandatory [ ] optional

REMEMBER :

- With the expanded version, a period is required between the <area> and the <type>. If the period is omitted, the syntax uses the simple version. The <type> then corresponds to the default type (for bit area BIT, for word area WORD – see table <Bereich> (Area)).
- If the representation of the start address is HEX, all numbers for this area are also HEX.
- The HEX / decimal numbers can be changed with the following prefixes.
  Conversion of HEX -> DEC: Input of 0d (number zero + the letter d) before the decimal address
  Conversion of DEC -> HEX Input of 0x (number 0 + the letter x) before the hexadecimal address
- When words, double words or strings are registered in a bit area, the start address is a bit address and only possible on word boundaries (dec. 0/16/32... or hex: 0/10/20...) (e.g., Y.D10.3).
- Bit arrays in bit areas are not possible.
- Bit arrays in WORD areas with HEX representation, are not possible.

<Area>

	Syntax	Orientation[1]	Representation of the Start address
Special Relay	**SM**	BIT	decimal
Special Register	**SD**	WORD	decimal
Input Relay	**X**	BIT	HEX
Output Relay	**Y**	BIT	HEX
Internal Relay	**M**	BIT	decimal
Latch Relay	**L**	BIT	decimal
Annunciator	**F**	BIT	decimal
Edge Relay	**V**	BIT	decimal
Link Relay	**B**	BIT	HEX
Data Register	**D**	WORD	decimal
Link Register	**W**	WORD	HEX
Timer Contact	**TS**	BIT	decimal
Timer Coil	**TC**	BIT	decimal
Timer Current Value	**TN**	WORD	decimal
Retentive Timer Contact	**STS**	BIT	decimal
Retentive Timer Coil	**STC**	BIT	decimal
Retentive Timer Current Value	**STN**	WORD	decimal
Counter Contact	**CS**	BIT	decimal
Counter Coil	**CC**	BIT	decimal
Counter Current Value	**CN**	WORD	decimal

Special Link Relay	**SB**	BIT	HEX
Special Link Register	**SW**	WORD	HEX
Step Relay	**S**	BIT	decimal
Direct Input	**DX**	BIT	HEX
Direct Output	**DY**	BIT	HEX
Index Register	**Z**	WORD	decimal
File Register (Normal Access by block Switching)	**R**	WORD	decimal
File Register (Serial No. Access)	**ZR**	WORD	HEX

[1] BIT-oriented means that one bit is addressed for each physical address. WORD-oriented means that one word (16 bits) is addressed for each physical address.

<.Type> <,Type>

Typ VT_Typ	**Syntax**	Simple Syntax	ex:Bit area	Ex: Word area	with Array in Bit area	with Array in Word area	useful Suffixes
BIT VT_BOOL	**X**	DY1	-----	D.X1.2	-----	-----	-----
BIT [1] VT_BOOL	**-----**	DY1	-----	D.1.2	-----	-----	-----
BYTE VT_UI2	**B** **BYTE**	-----	Y.B10 Y.BYTE10	R.B1 R.BYTE1	Y.B10.5 Y.BYTE10.5	R.B1.3 R.BYTE1.3	-----
INT VT_I2	**I** **INT**	Y.I10 Y.INT10	Y.I10 Y.INT10	R.I2 R.INT2	Y.I10.3 Y.INT10.3	R.I2.3 R.INT2.3	-----
WORD VT_UI2	**W** **WORD**	R20	Y.W10 Y.WORD10	R.W2 R.WORD2	Y.W10.3 Y.WORD10.3	R.W2.3 R.WORD2.3	-----
DOUBLE WORD VT_UI4	**D** **DW** **DWORD**	-----	Y.D10 Y.DWORD10	R.D2 R.DWORD2	Y.D10.3 Y.DWORD10.3	R.D2.3 R.DWORD2.3	-----
DOUBLE INT VT_I4	**DI** **DINT**	-----	Y.DI10 Y.DINT10	R.DI2 R.DINT2	Y.DI10.3 Y.DINT10.3	R.DI2.3 R.DINT2.3	-----
REAL VT_R4	**R** **REAL**	-----	Y.R10 Y.REAL10	R.R2 R.REAL2	Y.R10.3 Y.REAL10.3	R.R2.3 R.REAL2.3	-----
STRING VT_BSTR	**S** **STRING**	-----	Y.S10.20 Y.STRING10.20	R.S2 R.STRING2.20	-----	-----	-----

[1] Careful: With HEX addresses, it's better to select the version with the X.

[.Array size]

Arrays are created to combine several units of one data type into one field.
Examples:
D20.300
For more information on arrays, see also Arrays.

[Suffix]

Suffixes can be used to represent a value in another format.
Example: D20.300KF
For more information on suffixes, see also Suffixes.

1.6 KNX Item Syntax

The KNX item syntax is set up as shown below:

<Address>[$<Data type>]

Legend: <...> mandatory [<...>] optional

<Address>

KNX group address to access.
Possible syntax:
· <main>/<middle>/<subgroup> - 3-level address, main = 0 .. 31, middle = 0 .. 7, subgroup = 0 .. 255
· <main>/<subgroup> - 2-level address, main = 0 .. 31, subgroup = 0 .. 2047
· <group> - 1-level address, group = 0 .. 65535
Example: 1/0/1

<Data type>

The data type to use. If not given, the data is handled as byte array. Exactly one of the numeric code, encoding description or the name shall be used.

Numeric	Encoding	Name	Datatype
1.001	B1	DPT_Switch	Bool
1.002	B1	DPT_Bool	Bool
1.003	B1	DPT_Enable	Bool
1.004	B1	DPT_Ramp	Bool
1.005	B1	DPT_Alarm	Bool
1.006	B1	DPT_BinaryValue	Bool
1.007	B1	DPT_Step	Bool
1.008	B1	DPT_UpDown	Bool
1.009	B1	DPT_OpenClose	Bool
1.010	B1	DPT_Start	Bool
1.011	B1	DPT_State	Bool
1.012	B1	DPT_Invert	Bool
1.013	B1	DPT_DimSendStyle	Bool
1.014	B1	DPT_InputSource	Bool
1.015	B1	DPT_Reset	Bool
1.016	B1	DPT_Ack	Bool
1.017	B1	DPT_Trigger	Bool
1.018	B1	DPT_Occupancy	Bool
1.019	B1	DPT_Window_Door	Bool
1.021	B1	DPT_LogicalFunction	Bool
1.022	B1	DPT_Scene_AB	Bool
1.023	B1	DPT_ShutterBlinds_Mode	Bool
1.100	B1	DPT_Heat/Cool	Bool
2.001	B2	DPT_Switch_Control	Bool[2]

2.002	B2	DPT_Bool_Control	Bool[2]
2.003	B2	DPT_Enable_Control	Bool[2]
2.004	B2	DPT_Ramp_Control	Bool[2]
2.005	B2	DPT_Alarm_Control	Bool[2]
2.006	B2	DPT_BinaryValue_Control	Bool[2]
2.007	B2	DPT_Step_Control	Bool[2]
2.008	B2	DPT_Direction1_Control	Bool[2]
2.009	B2	DPT_Direction2_Control	Bool[2]
2.010	B2	DPT_Start_Control	Bool[2]
2.011	B2	DPT_State_Control	Bool[2]
2.012	B2	DPT_Invert_Control	Bool[2]
3.007	B4	DPT_Control_Dimming	Bool[4]
3.008	B4	DPT_Control_Blinds	Bool[4]
4.001	A8	DPT_Char_ASCII	Uint 8
4.002	A8	DPT_Char_8859_1	Uint 8
5.001	U8	DPT_Scaling	Uint 8
5.003	U8	DPT_Angle	Uint 8
5.004	U8	DPT_Percent_U8	Uint 8
5.005	U8	DPT_DecimalFactor	Uint 8
5.006	U8	DPT_Tariff	Uint 8
5.010	U8	DPT_Value_1_Ucount	Uint 8
6.001	V8	DPT_Percent_V8	Int 8
6.010	V8	DPT_Value_1_Count	Int 8
6.020	B5N3	DPT_Status_Mode3	Structure
7.001	U16	DPT_Value_2_Ucount	Uint 16
7.002	U16	DPT_TimerPeriodMsec	Uint 16
7.003	U16	DPT_TimerPeriod10MSec	Uint 16
7.004	U16	DPT_TimerPeriod100MSec	Uint 16
7.005	U16	DPT_TimerPeriodSec	Uint 16
7.006	U16	DPT_TimerPeriodMin	Uint 16
7.007	U16	DPT_TimerPeriodHrs	Uint 16
7.010	U16	DPT_PropDataType	Uint 16
7.011	U16	DPT_Length_mm	Uint 16
7.012	U16	DPT_UElCurrentmA	Uint 16
7.013	U16	DPT_Brightness	Uint 16
8.001	V16	DPT_Value_2_Count	Int 16
8.002	V16	DPT_DeltaTimeMsec	Int 16
8.003	V16	DPT_DeltaTime10MSec	Int 16
8.004	V16	DPT_DeltaTime100MSen	Int 16
8.001	V16	DPT_DeltaTimeSec	Int 16
8.006	V16	DPT_DeltaTimeMin	Int 16
8.007	V16	DPT_DeltaTimeHrs	Int 16
8.010	V16	DPT_Percent_V16	Int 16
8.011	V16	DPT_Rotation_Angle	Int 16

9.001	F16	DPT_Value_Temp	Float
9.002	F16	DPT_Value_Tempd	Float
9.003	F16	DPT_Value_Tempa	Float
9.004	F16	DPT_Value_Lux	Float
9.005	F16	DPT_Value_Wsp	Float
9.006	F16	DPT_Value_Pres	Float
9.007	F16	DPT_Value_Hunidity	Float
9.008	F16	DPT_Value_AirQuality	Float
9.010	F16	DPT_Value_Time1	Float
9.011	F16	DPT_Value_Time2	Float
9.020	F16	DPT_Value_Volt	Float
9.021	F16	DPT_Value_Curr	Float
9.022	F16	DPT_PowerDensity	Float
9.023	F16	DPT_KelvinPerPercent	Float
9.024	F16	DPT_Power	Float
9.025	F16	DPT_Value_Volume_Flow	Float
9.026	F16	DPT_Rain_Amount	Float
9.027	F16	DPT_Value_Temp_F	Float
9.028	F16	DPT_Value_Wsp_kmh	Float
10.001	N3U5r2U6r2U6	DPT_TimeOfDay	Structure
11.001	r3U5r4U4r1U7	DPT_Date	Structure
12.001	U32	DPT_Value_4_Ucount	Uint 32
13.001	V32	DPT_Value_4_Count	Int 32
13.002	V32	DPT_FlowRate_m3/h	Int 32
13.010	V32	DPT_ActiveEnergy	Int 32
13.011	V32	DPT_ApparantEnergy	Int 32
13.012	V32	DPT_ReactiveEnergy	Int 32
13.013	V32	DPT_ActiveEnergy_kWh	Int 32
13.014	V32	DPT_ApparentEnergy_kVAh	Int 32
13.015	V32	DPT_ReactiveEnergy_kVARh	Int 32
13.100	V32	DPT_LongDeltaTimeSec	Int 32
14.000	F32	DPT_Value_Acceleration	Float
14.001	F32	DPT_Value_Acceleration_Angular	Float
14.002	F32	DPT_Value_Activation_Energy	Float
14.003	F32	DPT_Value_Activity	Float
14.004	F32	DPT_Value_Mol	Float
14.005	F32	DPT_Value_Amplitude	Float
14.006	F32	DPT_Value_AngleRad	Float
14.007	F32	DPT_Value_AngleDeg	Float
14.008	F32	DPT_Value_AngularMomentum	Float
14.009	F32	DPT_Value_Angular_Velocity	Float

14.010	F32	DPT_Value_Area	Float
14.011	F32	DPT_Value_Capacitance	Float
14.012	F32	DPT_Value_Charge_DensitySurface	Float
14.013	F32	DPT_Value_Charge_DensityVolume	Float
14.014	F32	DPT_Value_Compressibility	Float
14.015	F32	DPT_Value_Conductance	Float
14.016	F32	DPT_Value_Electrocal_Conductivity	Float
14.017	F32	DPT_Value_Density	Float
14.018	F32	DPT_Value_Electric_Charge	Float
14.019	F32	DPT_Value_Electric_Current	Float
14.020	F32	DPT_Value_Electric_CurrentDensity	Float
14.021	F32	DPT_Value_Electric_DipoleMoment	Float
14.022	F32	DPT_Value_Electric_Displacement	Float
14.023	F32	DPT_Value_Electric_FieldStrength	Float
14.024	F32	DPT_Value_Electric_Flux	Float
14.025	F32	DPT_Value_Electric_FluxDensity	Float
14.026	F32	DPT_Value_Electric_Polatization	Float
14.027	F32	DPT_Value_Electric_Potential	Float
14.028	F32	DPT_Value_Electric_PotentialDifference	Float
14.029	F32	DPT_Value_ElectromagneticMoment	Float
14.030	F32	DPT_Value_Electromotive_Force	Float
14.031	F32	DPT_Value_Energy	Float
14.032	F32	DPT_Value_Force	Float
14.033	F32	DPT_Value_Frequency	Float
14.034	F32	DPT_Value_AngularFrequency	Float
14.035	F32	DPT_Value_Heat_Capacity	Float
14.036	F32	DPT_Value_Heat_FlowRate	Float
14.037	F32	DPT_Value_Heat_Quantity	Float
14.038	F32	DPT_Value_Impedance	Float

14.039	F32	DPT_Value_Length	Float
14.040	F32	DPT_Value_Light_Quantity	Float
14.041	F32	DPT_Value_Luminance	Float
14.042	F32	DPT_Value_Luminous_Flux	Float
14.043	F32	DPT_Value_Luminous_Intensity	Float
14.044	F32	DPT_Value_Magnetic_FieldStrength	Float
14.045	F32	DPT_Value_Magnetic_Flux	Float
14.046	F32	DPT_Value_Magnetic_FluxDensity	Float
14.047	F32	DPT_Value_Magnetic_Moment	Float
14.048	F32	DPT_Value_Magnetic_Polarization	Float
14.049	F32	DPT_Value_Manetization	Float
14.050	F32	DPT_Value_MagnetomotiveForce	Float
14.051	F32	DPT_Value_Mass	Float
14.052	F32	DPT_Value_MassFlux	Float
14.053	F32	DPT_Value_Momentum	Float
14.054	F32	DPT_Value_Phase_AngleRad	Float
14.055	F32	DPT_Value_Phase_AngleDeg	Float
14.056	F32	DPT_Value_Power	Float
14.057	F32	DPT_Value_Power_Factor	Float
14.058	F32	DPT_Value_Pressure	Float
14.059	F32	DPT_Value_Reactance	Float
14.060	F32	DPT_Value_Resistance	Float
14.061	F32	DPT_Value_Resistivity	Float
14.062	F32	DPT_Value_SelfInductance	Float
14.063	F32	DPT_Value_SolidAngle	Float
14.064	F32	DPT_Value_Sound_Intensity	Float
14.065	F32	DPT_Value_Speed	Float
14.066	F32	DPT_Value_Stress	Float
14.067	F32	DPT_Value_Surface_Tension	Float
14.068	F32	DPT_Value_Common_Temperature	Float
14.069	F32	DPT_Value_Absolute_Temperature	Float
14.070	F32	DPT_Value_Temperature	Float

		Difference	
14.071	F32	DPT_Value_Thermal_Capacity	Float
14.072	F32	DPT_Value_Thermal_Conductivity	Float
14.073	F32	DPT_Value_ThermoelectricPower	Float
14.074	F32	DPT_Value_Time	Float
14.075	F32	DPT_Value_Torque	Float
14.076	F32	DPT_Value_Volume	Float
14.077	F32	DPT_Value_Volume_Flux	Float
14.078	F32	DPT_Value_Weight	Float
14.079	F32	DPT_Value_Work	Float
15.000	U4U4U4U4U4U4B4N4	DPT_Access_Data	Structure
16.000	A112	DPT_String_ASCII	String
16.001	A112	DPT_String_8859_1	String
17.001	r2U6	DPT_SceneNumber	Structure
18.001	B1r1U6	DPT_SceneControl	Structure
19.001	U8r4U4r3U5U3U5r2U6r2U6B16	DPT_DateTime	Structure
20.001	N8	DPT_SCLOMode	Uint 8
20.002	N8	DPT_BuildingMode	Uint 8
20.003	N8	DPT_OccMode	Uint 8
20.004	N8	DPT_Priority	Uint 8
20.005	N8	DPT_LightApplicationMode	Uint 8
20.006	N8	DPT_ApplicationArea	Uint 8
20.007	N8	DPT_AlarmClassType	Uint 8
20.008	N8	DPT_PSUMode	Uint 8
20.011	N8	DPT_ErrorClass_System	Uint 8
20.012	N8	DPT_ErrorClass_HVAC	Uint 8
20.013	N8	DPT_Time_Delay	Uint 8
20.014	N8	DPT_Beaufort_Wind_Force_Scale	Uint 8
20.017	N8	DPT_SensorSelect	Uint 8
20.020	N8	DPT_ActuatorConnectType	Uint 8
20.100	N8	DPT_FuelType	Uint 8
20.101	N8	DPT_BurnerType	Uint 8
20.102	N8	DPT_HVACMode	Uint 8
20.103	N8	DPT_DHWMode	Uint 8
20.104	N8	DPT_LoadPriority	Uint 8
20.105	N8	DPT_HVACContrMode	Uint 8
20.106	N8	DPT_HVACEmergMode	Uint 8
20.107	N8	DPT_ChangeoverMode	Uint 8
20.108	N8	DPT_ValveMode	Uint 8
20.109	N8	DPT_DamperMode	Uint 8
20.110	N8	DPT_HeaderMode	Uint 8

20.111	N8	DPT_FanMode	Uint 8
20.112	N8	DPT_MasterSlaveMode	Uint 8
20.113	N8	DPT_StatusRoomSetp	Uint 8
20.120	N8	DPT_ADAType	Uint 8
20.121	N8	DPT_BackupMode	Uint 8
20.122	N8	DPT_StartSynchronization	Uint 8
20.600	N8	DPT_Behaviour_Lock_Unlock	Uint 8
20.601	N8	DPT_Behaviour_Bus_Power_Up_Down	Uint 8
20.602	N8	DPT_DALI_Fade_Time	Uint 8
20.603	N8	DPT_BlinkingMode	Uint 8
20.604	N8	DPT_LightControlMode	Uint 8
20.605	N8	DPT_SwitchPBModel	Uint 8
20.606	N8	DPT_PBAction	Uint 8
20.607	N8	DPT_DimmPBModel	Uint 8
20.608	N8	DPT_SwitchOnMode	Uint 8
20.609	N8	DPT_LoadTypeSet	Uint 8
20.610	N8	DPT_LoadTypeDetected	Uint 8
20.801	N8	DPT_SABExcept-Behaviour	Uint 8
20.802	N8	DPT_SABBehaviour_Lock_Unlock	Uint 8
20.803	N8	DPT_SSSBMode	Uint 8
20.804	N8	DPT_BlindsControlMode	Uint 8
20.1000	N8	DPT_CommMode	Uint 8
20.1001	N8	DPT_AddInfoTypes	Uint 8
20.1002	N8	DPT_RF_ModeSelect	Uint 8
20.1003	N8	DPT_RF_FilterSelect	Uint 8
21.001	B8	DPT_StatusGen	Bool[8]
21.002	B8	DPT_Device_Control	Bool[8]
21.100	B8	DPT_ForceSign	Bool[8]
21.101	B8	DPT_ForceSignCool	Bool[8]
21.102	B8	DPT_StatusRHC	Bool[8]
21.103	B8	DPT_StatusSDHWC	Bool[8]
21.104	B8	DPT_FullTypeSet	Bool[8]
21.105	B8	DPT_StatusRCC	Bool[8]
21.106	B8	DPT_StatusAHU	Bool[8]
21.601	B8	DPT_LightActuatorErrorInfo	Bool[8]
21.1000	B8	DPT_RF_ModeInfo	Bool[8]
21.1001	B8	DPT_RF_FilterInfo	Bool[8]
21.1010	B8	DPT_Channel_Activation_16	Bool[8]
22.100	B16	DPT_StatusDHWC	Bool[16]
22.101	B16	DPT_StatusRHCC	Bool[16]
22.1000	B16	DPT_Media	Bool[16]

22.1010	B16	DPT_Media	Bool[16]
23.001	N2	DPT_OnOff_Action	Bool[2]
23.002	N2	DPT_Alarm_Reaction	Bool[2]
23.003	N2	DPT_UpDown_Action	Bool[2]
23.102	N2	DPT_HVAC_PB_Action	Bool[2]
24.001	A[n]	DPT_VarString_8859_1	String
27.001	B32	DPT_CombinedInfoOnOff	Bool[32]
28.001	A[n]$UTF8	DPT_UTF-8	String
29.010	V64	DPT_ActiveEnergy_V64	Int 64
29.011	V64	DPT_ApparentEnergy_V64	Int 64
29.012	V64	DPT_ReactiveEnergy_V64	Int 64
--	U64	--	Uint 64
30.1010	B24	DPT_Channel_Activation_24	Bool[24]
31.101	N3	DPT_PB_Action_HVAC_Extended	Bool[3]
231.001	A8A8A8A8	DPT_Locale_ASCII	String
234.001	A8A8	DPT_LanguageCodeAlpha2_ASCII	String
234.002	A8A8	DPT_RegionCodeAlpha2_ASCII	String

Examples:
1/0/1$DPT_Switch - Monitor the light switch group 1/0/1, or switch all lights belonging to that group
1/0/1$B1 - the same, using the encoding syntax
1/0/1$1.001 - the same, using the numeric syntax

1.7 IEC60870-5-104 Item Syntax

The IEC60870-5-104 item syntax is set up as shown below:

[<CA>.]<Data type>[.<IOA>][.<Option>][.<more Options ...>]

Legend: <...> mandatory [<...>] optional

<CA>

Specifies the CA (Common Address) to use. Required if the connection is set to "Allow all CAs". Not used if a CA is specified in the connection settings.

<IOA>

Specifies the IOA (Information Object Address) to use. Required for ordinary Read and Write operations. Not used for special commands.

<Data type> (read = monitor direction)

	Syntax[1]	Type[2]	Access Rights	Options[3]
Single Point Information	**M_SP** **M_SP_NA_1** **M_SP_TA_1** **M_SP_TB_1**	Bool	Read	Value/SPI - 0 OFF, 1 ON BL - 0 Not Blocked, 1 Blocked SB - 0 Not Substituted, 1 Substituted NT - 0 Topical, 1 Not

				Topical IV - 0 Valid, 1 Invalid Timestamp - transmitted timestamp, ununsed for M_SP_NA_1
Double Point Information	**M_DP** **M_DP_NA_1** **M_DP_TA_1** **M_DP_TB_1**	Bool[2]	Read	Value/DPI - 00 intermediate state, 01 OFF, 10 ON, 11 invalid state BL - 0 Not Blocked, 1 Blocked SB - 0 Not Substituted, 1 Substituted NT - 0 Topical, 1 Not Topical IV - 0 Valid, 1 Invalid Timestamp - transmitted timestamp, ununsed for M_DP_NA_1
Step Position Information	**M_ST** **M_ST_NA_1** **M_ST_TA_1** **M_ST_TB_1**	Int 8	Read	Value - Position Value T - 0 Not Topical, 1 Topical OV - 0 No Overflow, 1 Overflow BL - 0 Not Blocked, 1 Blocked SB - 0 Not Substituted, 1 Substituted NT - 0 Topical, 1 Not Topical IV - 0 Valid, 1 Invalid Timestamp - transmitted timestamp, ununsed for M_ST_NA_1
Binary State Information	**M_BO** **M_BO_NA_1** **M_BO_TA_1** **M_BO_TB_1**	Bool[32]	Read	Value/BSI - Binary State OV - 0 No Overflow, 1 Overflow BL - 0 Not Blocked, 1 Blocked SB - 0 Not Substituted, 1 Substituted NT - 0 Topical, 1 Not Topical IV - 0 Valid, 1 Invalid Timestamp - transmitted timestamp, ununsed for M_BO_NA_1
Measured Value, Normalized	**M_ME_NV** **M_ME_ND_1** **M_ME_NA_1** **M_ME_TA_1** **M_ME_TD_1**	Float	Read	Value/NVA - Normalized Value OV - 0 No Overflow, 1 Overflow, ununsed for M_ME_ND_1

				BL - 0 Not Blocked, 1 Blocked, ununsed for M_ME_ND_1 SB - 0 Not Substituted, 1 Substituted, ununsed for M_ME_ND_1 NT - 0 Topical, 1 Not Topical, ununsed for M_ME_ND_1 IV - 0 Valid, 1 Invalid, ununsed for M_ME_ND_1 Timestamp - transmitted timestamp, ununsed for M_ME_ND_1, M_ME_NA_1
Measured Value, Scaled	**M_ME_SV** **M_ME_NB_1** **M_ME_TB_1** **M_ME_TE_1**	Int 16	Read	Value/SVA - Scaled Value OV - 0 No Overflow, 1 Overflow BL - 0 Not Blocked, 1 Blocked SB - 0 Not Substituted, 1 Substituted NT - 0 Topical, 1 Not Topical IV - 0 Valid, 1 Invalid Timestamp - transmitted timestamp, ununsed for M_ME_NB_1
Measured Value, Short Float	**M_ME_FV** **M_ME_NC_1** **M_ME_TC_1** **M_ME_TF_1**	Float	Read	Value/FVA - Floating-point Value OV - 0 No Overflow, 1 Overflow BL - 0 Not Blocked, 1 Blocked SB - 0 Not Substituted, 1 Substituted NT - 0 Topical, 1 Not Topical IV - 0 Valid, 1 Invalid Timestamp - transmitted timestamp, ununsed for M_ME_NC_1
Binary Counter	**M_IT** **M_IT_NA_1** **M_IT_TA_1** **M_IT_TB_1**	Int 32	Read	Value/BCR - Binary Counter Reading SQ - Sequence Counter CY - 0 No Carry, 1 Carry SA - 0 Not Adjusted, 1 Counter Adjusted IV - 0 Valid, 1 Invalid Timestamp - transmitted

				timestamp, ununsed for M_IT_NA_1
Single Event of Protection Equipment	**M_EP_EV** **M_EP_TA_1** **M_EP_TD_1**	Bool[2]	Read	Value/ES - Event State EI - 0 Elapsed Time Valid, 1 Elapsed Time Invalid BL - 0 Not Blocked, 1 Blocked SB - 0 Not Substituted, 1 Substituted NT - 0 Topical, 1 Not Topical IV - 0 Valid, 1 Invalid ElapsedTime - Elapsed Time Timestamp - transmitted timestamp
Start Events of Protection Equipment	**M_EP_SE** **M_EP_TB_1** **M_EP_TE_1**	Bool	Read	Value/GS - General Start of Operation SL1 - Start of Operation Phase L1 SL2 - Start of Operation Phase L2 SL3 - Start of Operation Phase L3 SIE - Start of Operation Earth Current SRD - Start of Operation in Reverse Direction EI - 0 Elapsed Time Valid, 1 Elapsed Time Invalid BL - 0 Not Blocked, 1 Blocked SB - 0 Not Substituted, 1 Substituted NT - 0 Topical, 1 Not Topical IV - 0 Valid, 1 Invalid RelayDurationTime - Relay Duration Time Timestamp - transmitted timestamp
Output Circuit Information of Protection Equipment	**M_EP_OC** **M_EP_TC_1** **M_EP_TF_1**	Bool	Read	Value/GC - General Command to Output Circuit CL1 - Command to Output Circuit Phase L1 CL2 - Command to Output Circuit Phase L2 CL3 - Command to Output Circuit Phase L3

				EI - 0 Elapsed Time Valid, 1 Elapsed Time Invalid BL - 0 Not Blocked, 1 Blocked SB - 0 Not Substituted, 1 Substituted NT - 0 Topical, 1 Not Topical IV - 0 Valid, 1 Invalid RelayOperatingTime - Relay Operating Time Timestamp - transmitted timestamp
Status and Change Detection	**M_PS M_PS_NA_1**	Bool[16]	Read	Value/State - Current State CD - Change Detection OV - 0 No Overflow, 1 Overflow BL - 0 Not Blocked, 1 Blocked SB - 0 Not Substituted, 1 Substituted NT - 0 Topical, 1 Not Topical IV - 0 Valid, 1 Invalid

[1]First line captures all opcodes for the type, other lines capture only the specific opcode.
[2]The data type of the Value element. Most types support additional information which may have different types.
[3]For read, only one option may be given. If missing, all data is returned as a structure.

<Data type> (write = command direction, simple values)

	Syntax[1]	Type[2]	Access Rights	Options[3]
Single Point Command	**C_SC C_SC_NA_1 C_SC_TA_1**	Bool	Write	write mode (required if any options are given): D/Direct - one-step write SE/SelectExecute - two-step write Timestamp mode (only if not specified via type, default NTS): TS/Timestamp - send timestamp (use C_xx_Tx_1 opcode) NTS/NoTimestamp - don't send timestamp (use C_xx_Nx_1 opcode) Write Qualifier (default 0 - unspecified): SP/ShortPulse - send the Short Pulse qualifier LP/LongPulse - send the

				Long Pulse qualifier P/Persistent - send the Persistent qualifier QU0 .. QU31 - specify numerically
Double Point Command	**C_DC** **C_DC_NA_1** **C_DC_TA_1**	Bool[2]	Write	write mode (required if any options are given): D/Direct - one-step write SE/SelectExecute - two-step write Timestamp mode (only if not specified via type, default NTS): TS/Timestamp - send timestamp (use C_xx_Tx_1 opcode) NTS/NoTimestamp - don't send timestamp (use C_xx_Nx_1 opcode) Write Qualifier (default 0 - unspecified): SP/ShortPulse - send the Short Pulse qualifier LP/LongPulse - send the Long Pulse qualifier P/Persistent - send the Persistent qualifier QU0 .. QU31 - specify numerically
Regulating Command	**C_RC** **C_RC_NA_1** **C_RC_TA_1**	Bool[2]	Write	write mode (required if any options are given): D/Direct - one-step write SE/SelectExecute - two-step write Timestamp mode (only if not specified via type, default NTS): TS/Timestamp - send timestamp (use C_xx_Tx_1 opcode) NTS/NoTimestamp - don't send timestamp (use C_xx_Nx_1 opcode) Write Qualifier (default 0 - unspecified): SP/ShortPulse - send the Short Pulse qualifier LP/LongPulse - send the Long Pulse qualifier P/Persistent - send the Persistent qualifier QU0 .. QU31 - specify numerically

Binary State Command	**C_BO** **C_BO_NA_1** **C_BO_TA_1**	Bool[32]	Write	write mode (required if any options are given): D/Direct - one-step write Timestamp mode (only if not specified via type, default NTS): TS/Timestamp - send timestamp (use C_xx_Tx_1 opcode) NTS/NoTimestamp - don't send timestamp (use C_xx_Nx_1 opcode)
Set-Point Command, Normalized Value	**C_SE_NV** **C_SE_NA_1** **C_SE_TA_1**	Float	Write	write mode (required if any options are given): D/Direct - one-step write SE/SelectExecute - two-step write Timestamp mode (only if not specified via type, default NTS): TS/Timestamp - send timestamp (use C_xx_Tx_1 opcode) NTS/NoTimestamp - don't send timestamp (use C_xx_Nx_1 opcode) Write Qualifier (default 0 - unspecified): QL0 .. QL127 - specify numerically
Set-Point Command, Scaled Value	**C_SE_SV** **C_SE_NB_1** **C_SE_TB_1**	Int 16	Write	write mode (required if any options are given): D/Direct - one-step write SE/SelectExecute - two-step write Timestamp mode (only if not specified via type, default NTS): TS/Timestamp - send timestamp (use C_xx_Tx_1 opcode) NTS/NoTimestamp - don't send timestamp (use C_xx_Nx_1 opcode) Write Qualifier (default 0 - unspecified): QL0 .. QL127 - specify numerically
Set-Point Command, Floating-Point Value	**C_SE_FV** **C_SE_NC_1** **C_SE_TC_1**	Float	Write	write mode (required if any options are given): D/Direct - one-step write SE/SelectExecute - two-

				step write Timestamp mode (only if not specified via type, default NTS): TS/Timestamp - send timestamp (use C_xx_Tx_1 opcode) NTS/NoTimestamp - don't send timestamp (use C_xx_Nx_1 opcode) Write Qualifier (default 0 - unspecified): QL0 .. QL127 - specify numerically

[1]First line selects the opcode based on the presence/absence of the Timestamp option, other lines use the specific opcode.
[2]The data type to write.
[3]For write, multiple options may be given, depending on the opcode, minimum is D or SE. If none are present, write a structure that gives the options.

<Data type> (command direction, special commands)

	Syntax[1]	Type[2]	Access Rights	Options[3]
Interrogation Command	**C_IC** **C_IC_NA_1**	Bool	Write	interrogation mode: G/Global - global interrogation INRO1 .. INRO16 - group interrogation QOI0 .. QOI255 - specify numerically
Counter Interrogation Command	**C_CI** **C_CI_NA_1**	Bool	Write	interrogation mode: G/Global - global interrogation REQCO1 .. REQCO4 - group interrogation RQT0 .. RQT63 - specify numerically counter freeze mode (default 0 - read only): F/Freeze - freeze counters R/Reset - reset counters FR/FreezeAndReset - freeze and reset counters
Clock Synchronication Command	**C_CS** **C_CS_NA_1**	Bool	Write	
Test Command	**C_TS** **C_TS_NA_1** **C_TS_TA_1**	Bool	Write	write mode (required if any options are given): D/Direct - one-step write Timestamp mode (only if not specified via type, default NTS):

				TS/Timestamp - send timestamp (use C_xx_Tx_1 opcode) NTS/NoTimestamp - don't send timestamp (use C_xx_Nx_1 opcode)

[1]First line selects the opcode based on the presence/absence of the Timestamp option, other lines use the specific opcode.
[2]The data type to write. For commands, any written value will execute the command.
[3]For write, multiple options may be given, depending on the opcode, minimum is D or SE. If none are present, write a structure that gives the options.
[4]For C_CS, write a Bool value to Item "C_CS".

Examples
M_SP.1000.SPI - Monitor the Single-Point value at IOA 1000; returns a Bool value. Monitors opcodes M_SP_NA_1, M_SP_TA_1, M_SP_TB_1
M_SP_NA_1.1000.SPI - Monitor the Single-Point value at IOA 1000; returns a Bool value. Monitors opcode M_SP_NA_1 only
M_SP.1000.IV - Monitor the Invalid bit of Single-Point value 1000
M_SP.1000.Timestamp - Monitor the Timestamp of Single-Point value 1000
M_SP.1000 - Monitor the Single-Point value 1000; returns a structure SIQ_TS that contains the value, timestamp and all quality bits
C_SC.1000.D - Write a Bool value to execute a Single Point Command to IOA 1000 (uses C_SC_NA_1)
C_SC.1000.D.TS - Write a Bool value to execute a Single Point Command to IOA 1000, include a timestamp (uses C_SC_TA_1)
C_SC.1000.SE - Write a Bool value to execute a Single Point Command to IOA 1000 (uses C_SC_NA_1 and the Select-and-Execute semantic)
C_SC.1000.D.SP - Write a Bool value to execute a Single Point Command to IOA 1000 (uses C_SC_NA_1 and the Short-Pulse qualifier)
C_SC.1000 - Write a structure SCO to execute a Single Point Command to IOA 1000. Value, opcode and write options are specified by the structure contents.
C_IC.G - Write a Bool value to execute a Global Interrogation Command

1.8 Send/Receive Item Syntax

Send/Receive frames will be handled as byte arrays.
The name is raw.
The frames will be sent and received as one piece. In connections without a protocol like PLC Header, RFC1006 or OSI/H1 it can not be guaranteed that the frames will not be concentrated or splitted in pieces.
Ideally RAW data will be handled by PLC Engine Collect. It contains lot of RAW data access functions as search, change and more.

1.9 Browsing Tree

The address space can be browsed online.

The address space on OPC Classic and OPCPipe starts in the root.
Example: <**PLC**>.**flag0**. Or "**System**.**Topics**.<**PLC**>.**Status**

On OPC UA the root starts with "**Objects**". Under this topic there are multiple elements depending on the OPC configuration. The most important element is the element "**Topics**". All elements described on this manual are under this item.
Example: **Objects**.**Topics**.<**PLC**>.**flag0**. Or "**Objects**.**Topics**.**System**.**Topics**.<**PLC**>.**Status**

If browsing of items is not used registering an item needs a prefix. Which prefix your OPC UA server requires can be found in the documentation of your OPC UA server. No standard for a prefix exist, some servers are using names, others UUID or numbers, but it can be a binary field also. There exist no limitation in length. It is not guaranteed that the prefix will be the same after reconnect to the UA server.

The OPC UA server needs if browsing will not be used "**#ns=1;s=**" in front of an item. Leave the "**Objects.Topics**" away.

Example the OPC UA server: Read the flag word zero from a Siemens S7 300, the OPC connection is named "OPC", the controller connection is named "s7": "**OPC.#ns=1;s=s7.mw0"**

1.10 System Tree

The **System tree** contains the s**pecial items** the server supports. They allow accessing internal variables, their status and a lot more.

The system tree **"System"** lies in
- **OPC-DA and OpcPipe**: directly in the root.
- **OPC-UA** under **"Objects.Topic.System"**

With writing to the special item **"AddTopic"** it is possible adding more connections to PLC without using the configuration software.

The system topic can be switched off with the configuration software under "Station" - "General system settings" "Use OPC system topic".

Examples:
- SPS connection status: If a connection named "s7" is configured the connection status lies under **System**.**Topics**.s7.**Status (**over OPC UA **server.System**.**Topics**.s7.**Status)**
  The PLC operation mode lies under
  System.**Topics**.s7.**PlcDetails**.**PlcMode**
- Add a new connection with writing to **System**.**AddTopic**.
  "s7.tcp://192.168.2.200:102?name=\"my name \",typ=client,ownTSAP=0101,destinationTSAP=0302,slave=9,start1=1,WriteAllowed=1"

Elements in the root

- OPCPipe and OPC DA the start floor lies in the root floor.
- Under OPC-UA the start floor lies under "**Objects.Topics**"

Topic name	Description
System	System data as the list of topics, version information
Memory	Temporary variables and structures which relays on the system memory.
<**user defined**>	User defined topics created with the configuration software or with the browser or over writing to **System**.**AddTopic**

System

Topic name	Description
Topics	The predefined item "**System.Topics**" is containing the list of configured connections to controllers. The elements allows accessing information about stati of the OPC server itself and the connections to the controllers. Additional details from the controllers can be read as the name of the controller, its operation mode and more.
AddTopic	Adds connections to controllers. With writing of a string in the defined syntax connections will be added. The syntax is described in AddTopic.
Licenses	information about the current licenses
Versions	Versions of the software and its parts from the OPC server or the PlcEngine.
Internal	Please do not use this. It is made for special applications.
Platform	allows tags for different Platform things, eg: inputs and outputs on Phytec Regor
CurrentTime **...** and other tags	the current time and time information is available for further processing.

Values for System.Topics.<topicName>

Topic name	Description
Status	Brings the status of the connection. 0-> ok 3 -> no connection 5 -> access denied 6 -> wait for data 7 -> wait for send ready 23 -> The communication partner denies the port. No program did open it 26 -> ARP can not resolve the address. Mostly the partner is switched off or disconnected 29 -> Suspended. No OPC client or logic table used one item. 30 -> Osi reset by peer. Mostly a TSAP is wrong 31 -> The port is already in use, mostly by another program 32 -> The domain name can not be resolved. The domain server is unreachable, or the name was not found. 33 -> The network is unreachable. Mostly the computers network interfaces are configured badly.
ReadCount	Number of handled synchronous read jobs. This variable can be written.
WriteCount	Number of handled write jobs. This variable can be written.
ActiveCount	Number of active items.
ServerCycle	Number of cycles with changed PLC data. This variable can be written.
Redundancy	This will exist in connections configured for redundancy only. Details here.
AddVariable	This element only exists in the Memory topic. Adds a variable in the memory variables. This element is write only. The syntax is <variable name>=<variable type><[><array size><]><;comment> Supported variable types are: u8, u16, u32, u64. i8, i16, i32, i64, f32, f64, bit, string. Example: MyVariable=u16[10];Array of UINT 16 with 10 elements

	If a variable name contains dots all named before the last dot are nodes. Hint: The symbols created with **AddVariable** will be deleted if the OPC or PLC Engine Collect software is stopped.
DeleteVariable	Deletes a variable which was created with **AddVariable**. The name of the variable need to be given. This element is write only. Each variable and node need to be deleted separately. Hint: If a variable will be deleted which was not created with **AddVariable** the access will be denied.
AddSymbol	The element only exists in controller topics which does not support symbols in the controller itself. Mostly this are Siemens S7 300/400 controllers, Modbus devices or Mitsubishi controllers. Adds a symbol. The item syntax of the desired controller type of the topic will be used. SymbolName=<ItemSyntax><;comment> Example for a S7 300: MyFlag=mw0;Flag word zero The symbol name can contain dots. So multiple floors can be created. Hint: The symbols created with **AddSymbol** will be deleted if the OPC or PLC Engine Collect software is stopped.
DeleteSymbol	Deletes a with **AddSymbol** created symbol.
AddStructure	Not available today
DeleteStructure	Not available today
DeleteTopic	If a connection has been added dynamically, this item is visible. If this is written to, this dynamically created connection is deleted.

Values for System.Topics.<topicName>.PlcDetail

If the information can not be readied from the controller the quality becomes bad (sensor failed).

Topic name	Description
PlcMode	0 -> PLC in Stop, 1 -> PLC in RUN/STOP, 2 -> PLC in RUN.
KeySwitch	0 -> Key switch in position Stop, 1 -> Run/Prog, 2 -> Run, 3 -> Mres
PlcName	The name given to the controller. Not all controllers does contain a name.
PlcType	The PLC type the manufacturer did give to it. Not all controllers does contain this.
OrderNumber	The order number of the controller. Not all controllers does contain this.
Force	Information if forces will exist in the controllers program code. Some p,c deliver the number of forces here.
Battery	Status of the battery. 0 -> Ok, 1 -> empty, 2 -> no battery, 3 -> not supported

Not all controller types and controller firmware versions are supporting all items under **PlcDetails**. So the **OrderNumber** will be available in Siemens S7 only. The details can be found in the controllers system manuals.

Values for System.AddTopic

Warning: System.AddTopic is deprecated. Pleas use the configuration topic which has much more functionality.

Topics can be created over this item.

Important: With **AddTopic** created connections are existing non permanent. On power off they will be lost. They must be created again after power on.

A topic created with **System**.**Topics**.<created topic> can be deleted with **System**.**Topics**.<created topic>**DeleteTopic**.

<table>
<tr><th>Topic name</th><th>Description</th></tr>
<tr><td>Name</td><td><Topic Name>. The name of the created connection. Please note the rules for topic names which will existing from various OPC clients.</td></tr>
<tr><td>PlcType</td><td>Valid are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>s7</td><td>Siemens S7-200, 300, 400 and compatible as Speed7. Can be used for accessing the controller over MPI adapter from Hilscher, Process Informatik, IBH Softec, Softing, Helmholz.</td></tr>
<tr><td>tia</td><td>Siemens S7 1200 and 1500. This will support the optimized data blocks.</td></tr>
<tr><td>s5</td><td>Siemens S5 with network CP All racks are supported 135, 155, 188. All CPU types.</td></tr>
<tr><td>compactlogics</td><td>Rockwell Control Logix and Compact Logix.</td></tr>
<tr><td>slc</td><td>Rockwell SLC family.</td></tr>
<tr><td>modbus</td><td>Modbus TCP and compatible systems as Wago, Beckhoff, Modicon, Omron and more..</td></tr>
<tr><td>melsecQ</td><td>Mitsubishi Melsec family.</td></tr>
<tr><td>raw</td><td>Raw data.</td></tr>
</table></td></tr>
<tr><td>NetworkProtocol</td><td>Valid are:
<table>
<tr><th>Value</th><th>Alternative</th><th>Description</th></tr>
<tr><td>ip</td><td>tcp</td><td>TCP/IP TCP over IpV4 (192.168.1.1) or IpV6 <235b:34aa::0001:0030) or domain names</td></tr>
<tr><td>udp</td><td></td><td>TCP/IP UDP over IpV4 (192.168.1.1) or IpV6 <235b:34aa::0001:0030) or domain names</td></tr>
<tr><td>h1</td><td>osi</td><td>OSI/H1 over MAC Addresses (080006010001) or (08:00:06:01:00:01)</td></tr>
</table></td></tr>
<tr><td>NetworkAddress</td><td>IpV4 (192.168.1.1) or
IpV6 (235b:34aa::0001:0030) or
domain names (s7.mydomain.com)</td></tr>
<tr><td>Port</td><td>1 .. 65534. This is needed for IP connections only.</td></tr>
<tr><td>ConnectType</td><td>Valid are:
• Server
• Client</td></tr>
<tr><td>OwnTSAP</td><td>in hexadecimal. This is valid on OSI/H1 or Port 102 (RFC1006) connections only. (0101)</td></tr>
<tr><td>DestinationTSAP</td><td>in hexadecimal. This is valid on OSI/H1 or Port 102 (RFC1006) connections only. (0302)</td></tr>
<tr><td>Slave</td><td>In Modbus connections this is the slave or node address. In Rockwell connections this is the CPU number. All other connections do not use this value.</td></tr>
<tr><td>Start1</td><td>I Modbus connection this defines the assumed first register. False means the first register is zero otherwise the register count starts with one.</td></tr>
<tr><td>RockwellRoutin</td><td>This is for reaching specific parts of the controller or access controllers in</td></tr>
</table>

g	underlaying connections. This element requires text. Three options exists: • No routing (empty path). This is for reaching CPU Damit with logical integrated network modules in the PLC. Example are Micro800 or L16. • Simple routing. This is for reaching a specific CPU in multi CPU controllers.. **CPU(0)** will access CPU 0. In controllers with logical separate network adapters this is required. The L35E-models will need this. • Extended routing. This is for accessing controllers over underlaying netwoks as DH+ **Port(1;2)** goes over port 1 to the destination 2. **PortEx(4;192.168.2.212)** accesses controllers in an underlaying ip network. In this example the network adapter has the port number 4, the next controller is reached over 192.168.2.212. **DHPlus(1;0;0;15)** reaches a controller over a DHplus on channel A(1), Source Link 0 the plc destination Link 0 Node 15. An example for a longer routing path:**Port(1;0)-Port(2;1)-PortEx(4;192.168.2.212)** The port and slot numbers are taken from the graphical configuration or RsWho. Connection routing paths can be very long if this runs over multiple underlaying busses. Some controllers as the Micro800 models can not have a routing path if they need to be accessed. In case of route over a Micro800 model to other controllers the path is required.
RockwellCharset	This can be: **UTF8** (default) or **ANSI**.
RockwellHeader	Allowed is **ENCAP** (default) or **CSP**.
RockwellProtocol	Allowed is **CIP** (default) or **PCCC**.
RockwellShowHidden	Possible are **None** (default), **Hidden**, **Underscore**, **All**.
Create	Writing a value not zero creates the connection. This variable is write only.
CreateStatus	0 -> Topic is created. This variable is read only. 1 -> Invalid connection parameters. Mostly this will happen on too long TSAP values. 5 -> Invalid adapter. The given adapter is not available. Mostly on variable adapters (USB to Ethernet) which was disconnected. 17 -> No free memory available. 101 -> not supported. This will happen on plc protocols which are not available. 1319 -> not supported. Mostly this happens on unsupported network protocols as IpV6 which is not configured. 1324 -> License limit. The maximum of connections are created already. 1325 -> not licensed.
AddTopic	can be used instead of the variables **Name**, **PlcType**, **NetworkProtocol**, **NetworkAddress**, **Port**, **ConnectType**, **OwnTSAP**, **DestinationTSAP** for creation of connections. This variable is write only. The syntax on an example of an s7 connection with the name "s7" "s7.tcp://192.168.2.200:102?name=\"my name \",typ=client,ownTSAP=0101,destinationTSAP=0302" The element **Create** is not used in this case.

Values for System.Versions

A version number of zero says that the component is not loaded.

Topic name	Description
PlcEngine	Version number of the core of the PlcEngine or OPC server.
Wmk	Version number of the Wmk library. This library contains general functions.
IpLib	Version number of the TCPIP library. This library contains all IP socket functions.
H1	Version number of the Osi/H1 driver.
OpcDA	Version number of the OPC DA library. This library contains Classic OPC and is available for Windows only.
OpcUA	Version number of the OPC UA library. This library contains OPC UA.
OpcPipe	Version number of the OpcPipe library. This library contains the item management.
PlcLib	Version number of the PLC protocol library. This library contains all PLC protocols and its management.
ItemSyntax	Version number of the item syntax library. The library converts symbols in elements.
ConfigSubs	Version number of the configuration general functions. The library handles the configuration of the core.
..	Depending of the software or the release more software component version numbers are shown.

Values for System.Platform

Platform offers variables which are different from platform to platform. Mostly this are I/O variables.

Variables for the Regor devices from the manufacturer Phytec:

Topicname	Beschreibung
Digital1Dir	Sets the function of pin 1. "true" defines it as output, "false" as input.
Digital1In	If pin 1 is an input "true" signals a closed connection between pin 1 and ground.
Digital1Out	If pin 1 is an output "true" signals a closed switch between pin 1 and ground.
	Digital2Dir, Digital2In, Digital2Out are for pin 2. Same to pin 3 and 4.
ErrorLED	"true" signals the LED on. Writing "false" switches it off, "true" on again.
RunStopLED	"true" signals the LED on. Writing "false" switches it off, "true" on again.

Values for System.Licenses

The licenses and their details are here.

Topic name	Description
BaseLicense	Order number of the product. The comment of the item contains the product name.
BaseLicenseExpires	End date for time limited and test licenses in the format yyyy-mm-dd hh:mm for example: 2017-12-31 23:59
BaseLicenseState	Bit coded value: Bit 0: A permanent license is present and it is OK. Bit 1: The dongle is recognized and valid. Bit 2: The dongle was recognized in the past, but now it can not be found.

	Bit 3: Licensing via software confirm code. Bit 4: A software confirm code is detected and OK. Bit 5: A time limited confirm code has been detected. The expiration time is shown in item BaseLicenseExpires. Bit 6: A time limited confirm code has been detected and expired. Bit 7: The hardware (network card) to which the license was attached is no longer present. Bit 8: No license is recognized, it is a test version. The expiration time is shown in item BaseLicenseExpires. Bit 9: The test phase has expired, a restart is required. Bit 10: Free license product is set, no licensing required
AddonLicenseX	Order number of an addon. X is a consecutive number and starts with 1. If multiple addons exist more AddonLicenseX elements will exist.The comment of the item contains the addon product name.

1.11 Configuration Tree

With the Configuration Topic all configurations can be done es the graphical configuration does. The topic can be created with the graphical configuration. If the software is installed on a new system the installer asks for the creation of this topic.

All details for the configuration is available in the Connection subtree.
It is separated to

Base	Name of the connection, rights, time details
OPC	All for OPC server and clients
Transport	IP Addresses, TSAPs, OSI
Operations	Handling for read and write connections, assistances

und viele Weitere die sich selbst erklären

The configuration topic knows some dynamic handling.

Example 1: Read the details of a connection.
Write the connection name to Configuration.Connection.Base.ConnectionName and the group name in Configuration.Connection.Base.ConnectionGroup.
After this write true to Configuration.Connection.Operations.Read.The job starts the content of the variable Configuration.Connection.Base.Status will change to 6 (running). If the variable changes zu zero the list of connections is read. In Configuration.Connection.Base.ListResults the results will be delivered. In Configuration.Connection.Base.ListResultMaxLen the number of connections will return. In case of an error the variable Configuration.Connection.Base.ListStatus is containing the error code, Configuration.Connection.Base.ListErrorText is containing an error text for this.
If this is handled over OPC UA so Configuration.Connection.Base.ListResult is an array with the desired length.

Example 2: Read the list of network adapters.
Write the variable Configuration.Operations.Adapter.Start to true. The variable Configuration.Operations.Assistants.Adapter.Status will return 6 (running). If the job finishes without an error the variable will become zero. In Configuration.Operations.Assistants.Adapter.Result the recognized network adapters of the station will return..In case of an error a human readable text will be in Configuration.Operations.Assistants.Adapter.ErrorText, in Configuration.Operations.Assistants.Adapter.Status the corresponding error code is delivered.

Example 3: Read the list of available Rockwell PLC.
Please search the network adapters as in example 2. Copy the needed adapter to Configuration.Operations.Assistants.RockwellHost.Configuration.Adapter. Write true to Configuration.Operations.Assistants.RockwellHost.Start. The job will start, The variable Configuration.Operations.Assistants.RockwellHostStatus will change to 6 (running). If the job will finish (in Configuration.Operations.Assistants.RockwellHost.Status zero comes) the found Rockwell controllers will be in Configuration.Operations.Assistants.RockwellHost.Result die Liste der gefundenen Steuerungen. Searching PLC will need time, some PLC will respond very late. So the variable Configuration.Operations.Assistants.RockwellHost.Start need set to true multiple times.
If this is handled over OPC UA so Configuration.Operations.Assistants.RockwellHost.Result is an array with the desired length.

Please import the structures from the Configuration topic if your OPC client can handle structures.

1.12 Redundancy

Connections with redundancy only provide these values.

Element name	Description
CurrentMaster	Reading brings the current master. 1 is the first connection. Writing switches to the given connection. Hint: Writing an invalid value does nothing. In **dynamic master** this switches to the given connection. If the new connection will work stable it will remain. Otherwise an other connection is choose. In **static master** it will be switched back after one minute if the master works stable. If the slave connection breaks it will be switched to the master immediately it it will work stable. Otherwise an outer connection is choose.
Connection<n>	This variable will exist in the number of redundancy connections. . Each Connection<n> Variable will contain **Status** **ServerCycle** **ReadCount** **WriteCount**. These variables are working in the same manner of in connections without redundancy. Details here.

1.13 Bit mask

With a bit mask several bits of a data type can be read or written, by combining them to one decimal value. Bit maskign of arrays is possible as well.

Syntax

8 bits of data word 0 of data block 5 shall be read beginning with bit 2. Therefore the following syntax is used:
- db5.w0#2.8 or
- db5.w0#2,8 or

- db5.w0#2#8

Examples

DB10.W03.4
DB10.DWord2#20.10
DB10.DWord6#20.10KF
DB10.DInt10#20#10

Note:
Does not work with Left Byte or Right Byte (DL, DR) and Suffix BA

Note:
Do not mask bits of REAL or KG

1.14 Arrays

The word array means a series of equal elements (field, row, data area). An array combines several units of one data type in a field. To create an array, the length of the array is added to the standard syntax, separated by a period.

Arrays are impossible in:
• with all suffixes that are string to the client: ○ ISO

1.15 Suffixes

Using a suffix, a value can be displayed in a different format.

Suffixes	**Syntax**	Used for	Area	Data type	Variant Data Type	Comment
Date and Time in String Format	**ISO**	DT	1990-1-1-00:00:00.000 bis 2098-12-31-24:59:59.999*	STRING	VT_BSTR	The suffix DT is used to show the data saved in the PLC as a combined data type DATE_AND_TIME and is transfered as a string. The data type DATE_AND_TIME has 8 bytes (64 bits) in the PLC. The year, the month, the day, the hour, the minutes, the seconds and the milliseconds are included. Remember to use the correct separators (hyphen, colon and period)!
BCD	**BCD**	Byte Word DWord QWord	Byte: 0 to 99 Word: 0 to 999 DWord: 0 to 9999999 QWord: 0 to 9999999999999999	VT_I1 VT_I2 VT_I4 VT_I8	VT_I1 VT_I2 VT_I4 VT_I8	With the BCD suffix, the data stored in the PLC is represented as an unsigned, binary-encoded value. For example, the decimal value “65535” will be represented as “9999”.

ASCII to Hex	**KA**	String	HEX: 0 to 9, A to F	STRING	VT_BSTR	With the KA suffix, the data stored in the PLC is represented as HEX characters.
Signed	**KF**	Byte Word DWord QWord	Byte: -128 to 127 Word: -32768 to 32767 DWord: -2147483648 to 2147483647 QWord −9.223.372.036.854.775.808 to 9.223.372.036.854.775.807	CHAR SHORT LONG LLONG	VT_I1 VT_I2 VT_I4 VT_I8	With the KF suffix, the data stored in the PLC is represented as a signed fixed point number.
S5-KG	**KG**	DWord	0.1469368E-38 to 0.1701412E39 Attention: The KG area in the PLC is larger than in the PC!	REAL	VT_R4	With the KG suffix, the data stored in the PLC is treated as a 4-byte floating point number.
S5-KT-Format	**KT**	Word	000.0 to 999.3	STRING	VT_BSTR	With the KT suffix, the data stored in the PLC is represented as a 2-byte time constant. The time base is included in addition to the time value. The value range is from 000.0 to 999.3
S5 Time	**S5T**	Word	0ms to 2h46m30	STRING	VT_BSTR	With the S5T suffix, the data stored in the PLC is represented as S5TIME (SIMATIC time). The S5TIME data type occupies one 16-bit word and is the product of the time value and the time interval (time base). The time duration is given in hours, minutes, seconds and milliseconds. The BCD number format is used for internal representation. The value range is from 0 ms to 2h46m30s. The smallest value is 10ms.
Date	**D**	Word	1990-01-01 bis 2168-12-31	STRING	VT_BSTR	The suffix D is used to show the data saved on the PLC as data type DATE. The DATE data type occupies one word. The content corresponds to the number of days since 01.01.1990. The representation contains the year, the day and the month, separated by a hyphen.September 1, 2006 is shown as 2006-01-09. The value range is from 0 (0 days since 01.01.1990: 1990-01-01) to 65378 (65378 days since 01.01.1990:

						2168-12-31). Rules for use of suffix D: • The years 1990 up to and including 2089 can be specified with 2 or 4 positions.The years starting with 2090 must be entered with 4 positions. 90 to 99 ==> 1990 to 1999 00 to 89 ==> 2000 to 2089 • The months and days can be specified with either 1 or 2 positions. • Anything but numbers can be used as a separator (e.g., 89/09/17). Any number of separators can be used.
Time	**T**	DWord	-24d20h31m23s648ms to 24d20h31m23s647ms	STRING	VT_BSTR	With the T suffix, the data stored in the PLC is represented as the TIME data type. The TIME data type occupies one double word (32 bits). The representation contains the days (d), hours (h), minutes (m), seconds (s) and milliseconds (ms). Milliseconds can be omitted. The value range is from -2147483648 (-24d20h31m23s648ms) to 2147483647 (24d20h31m23s647ms)
Time of Day	**TOD**	DWord, DInt	0:0:0.0 to 23:59:59.999	STRING	VT_BSTR	The TOD suffix is used to show the data saved on the PLC as data type TIME_OF_DAY. The data type TIME_OF_DAY occupies one double word (32 bits). The representation contains the information for hours:minutes:seconds and .milliseconds. Milliseconds can be omitted. The value range goes from 0:0:0.0 to 23:59:59.999.
TimeReal	**TR**	Word	0.01 to 9990.0	REAL	VT_R4	With the TR suffix, the data stored in the PLC is represented as the TIME REAL data type. The value range is from 0.01 to 9990.0 (s)
Swap bytes	**SWAP**	String Word, Int DWord, DInt QWord, QInt Real, Double	Exchange high byte and low byte	original data type	original data type	High byte and low byte of the tags stored in the PLC are exchanged. Important for strings in Modbus PLCs.
Comment	**_comment**	all items	gives the comment of the item	STRING	VT_BSTR	Only for OPC DA returns the comment of the item. It is write protected.

* ms can be omitted.

1.16 Item Syntax neutral

Modern PLCs don't need an item syntax. All symbols are available on the controller. They will be read online. This applies to all Rockwell ControlLogic and CompactLogix PLCs, all Siemens S7 1500 and 1200 models, and BACnet devices.
Some subsystems as MQTT do not know variables with defined data types. For these the neutral item syntax is offered also.
For MQTT the usage of this item syntax need to be configured in the connection parameters.

A tag is fully defined with the tag name, the data type is defined by the PLC. Normally, nothing needs to be added.

However, there are some exceptions:

- Addressing a single element of an array:
  the zero-based index is specified in square brackets after the item name.

<Tagname>[<index>]

If you need offline symbols, you can add them. This may be required:
- to have symbols without a controller online,
- to rename symbols
- for simulation connections.

The "item syntax" is the real symbol name in the controller.

Because no data type is available without online access to the PLC an optional name extension exists for specifying the type.
The rule is:

<Symbolname>{{<Datatype>,<Endian>}}

or for arrays:

<Symbolname>{{<Datatype>,<Arraylength>,<Endian>}}

The following data types are allowed (the spaces are important):
Bool
Int 8
Int 16
Int 32
Int 64
Uint 8
Uint 16
Uint 32
Uint 64
Float
Double
Timestamp
String

These is possible for endian:
LittleEndian, some times known as INTEL format
BigEndian, known as Motorola format

If non endian is given the format of the locally working cpu is used.

The symbol name (without the {{ }} name extension) must be present in the controller (except for simulation connections).