SIEMENS S7 Series
S7NetPlus Provider

Version 1.0.1

User’s guide

October 26, 2020

[Remarks]

[Revision History]

Version	Date	Content
1.0.0	2019-07-11	First edition.
1.0.1	2019-09-17	Accessing anything other than DataBlock
	2020-10-26	Corrected typographical errors in Appendix A

[Tested Model]

Model	Version	Notes
S7-1500		Some functions are restricted.
S7-300		

Contents

1. Introduction

The SIEMENS S7 Series Provider is a ORiN2 CAO provider that accesses the SIEMENS S7 Series (S7-200,300,400,1200,1500).

This document gives an overview of the SIEMENS S7NetPlus provider and the implemented CAO interface (function specification).

2. 2. Outline of this provider

2.1. Installation

The SIEMENS S7NetPlus provider module consists of the following DLLs: If you installed using the ORiN2 SDK installer, the installation process is unnecessary. If you want to install manually, please execute as shown in Table 2-1.

Table 2-1 S7NetPlus provider

File name	CaoProvSIEMENSS7NetPlus.dll
ProgID	CaoProv.SIEMENS.S7NetPlus
Registry registration	RegAsm.bat CaoProvSIEMENSS7NetPlus.dll
Remove registry registration	UnregAsm.bat CaoProvSIEMENSS7NetPlus.dll

[Note] RegAsm.bat and UnregAsm.bat are in the DotNet/Bat folder under the ORiN2 installation folder.

2.2. Outline

The SIEMENS S7 series provider uses the open source library S7Net + to access variables defined in the PLC's internal data blocks.

Please define variables of data block using TIA Portal of SIEMENS.

2.3. Restrictions

This provider have the following restrictions according to the specifications of the library used.

• There are restrictions on the data types that can be accessed by each model.

• It is not possible to access optimized data blocks.

2.4. Method and Property

2.4.1. **CaoWorkspace::AddController method**

At the timing of AddController execution, SIEMENS S7NetPlus provider specifies parameters necessary for device connection based on the following format.

Syntax AddController (<bstrCtrlName:BSTR>, <bstrProvName:BSTR>, <bstrPcName:BSTR>, [<bstrOption:BSTR>])

< bstrCtrlName > : [in] Controller name (VT_BSTR)
Specify free format character strings.

< bstrProvName > : [in] Provider name =ProgID (VT_BSTR)
Specify the fixed character string of ” CaoProv.SIEMENS.S7NetPlus”.

<bstrPcName> : [in] Execution machine name (VT_BSTR)
Specify a computer name for the remote connection. Enter null (“”) for the local connection.

<bstrOption> : [in] Specify the following options separated by commas.

Conn : Specify the connection destination. (Required)
Ex.) “Conn = TCP: {IP address of connection destination}”

CPUType : Specify the CPU type to communicate without hyphens. (Required)
Five models, S7200, S7300, S7400, S71200 and S71500 can be specified.
Ex.) "CPUType = S71500"

Rack : Specify the rack number to connect.
The default is treated as 0.

Slot : Specify the slot number to connect.
The default is treated as 0.

Example

```
CCaoController ctrl = engine.Workspaces[0].AddController(
        "Test", "CaoProv.SIEMENS.S7NetPlus",
        null, "Conn=TCP:192.168.0.3,CPUType=S71500");
```

2.4.2. **CaoController::AddVariable method**

AddVariable method of CaoController class is a method to access variables.

Syntax AddVariable (<bstrName:BSTR>, [<bstrOption:BSTR>])

< bstrName > : [in] Variable name (VT_BSTR)

Specify any string other than the "@" start.

<bstrOption> : [in] Specify the following options separated by commas.

DataType : Specifies the type of memory to access.

DataBlock,Input,Output,Memory,Counter,Timer can be specified.

DataBlock is specified by default.

See the Restrictions (Link) for an accessible DataType for more information.

Type : Specify the type of value to be acquired.

Bool, Char, Byte, Int, Word, DInt, DWord, LInt, LWord, Real, LReal, String, WString, CharArray, WCharArray can be specified.

Required if a DataType other than Counter or Timer is specified.

If Counter and Timer are specified for DataType, the values specified in the Type options are ignored; for Counter, LReal is specified for WORD,Timer.

For details of each type, refer to Details of Variable Types.

The types of values that can be specified are limited by the model.

See the Restrictions (Link) for Types of Values that You Can Specify for more information.

Address : Specify the address of the memory to access.

When Bool is specified for Type, specify in the form of {address}. {Bit offset}.

If you specify something other than Bool in Type, specify only the address.

(See Appendix A)

DBNo : Specify the data block number to access.
If a value other than DataBlock is specified for DataType, it is ignored.
The default is treated as 1.
(See 2.4.2.1)

Elem : Specify the number of elements to be acquired.
Specify as an integer of 0 or more.
The default is treated as 0.
Also, if String and WString are specified in Type, Elem is ignored and 0 is specified.
If 0 is specified or if CharArray or WCharArray is specified for Type, the value is acquired as the specified type.
Otherwise it gets the value as an array of the specified type.

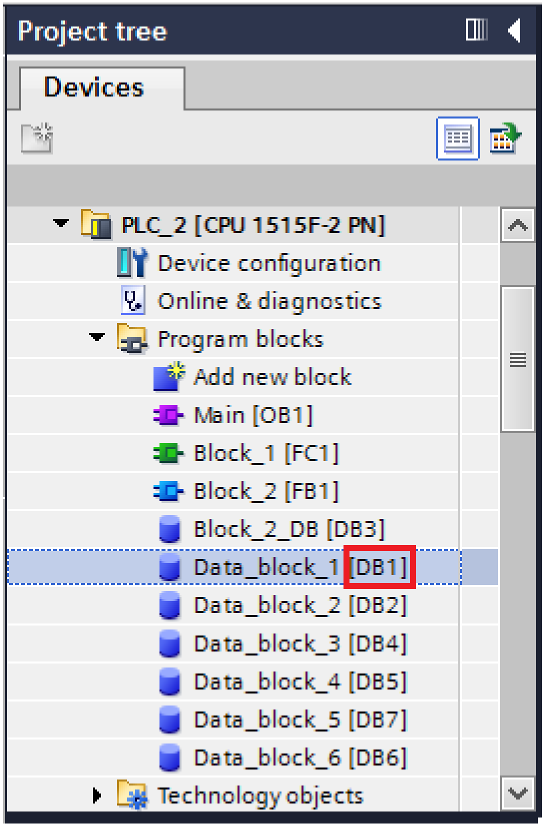
Example

```
    CCaoVariable variable = ctrl.AddVariable(
        "VAR1", "DBNo=1,Type=DINT,Address=0,Elem=0");
```

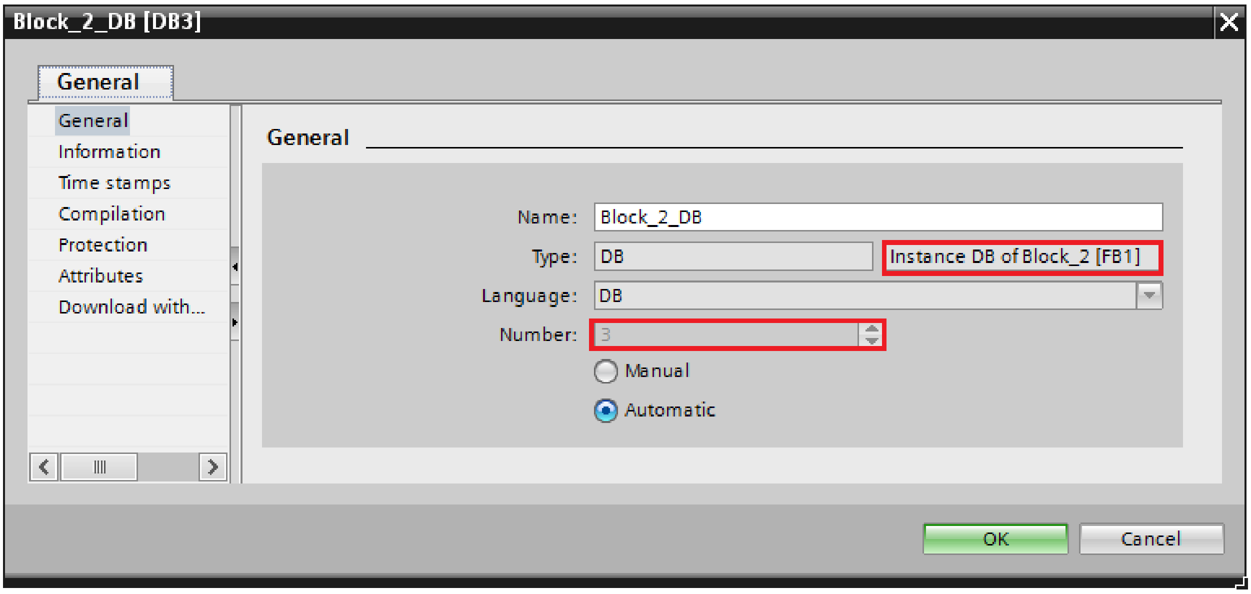
2.4.2.1. DBNo option

Please specify the number described on the right side of the data block name of TIA Portal.

In the example below, 1 is specified.



Instance data blocks can not be checked on the project tree, so check the data block by selecting [Right-click]-[Properties].



General

Name: Block_2_DB

Type: DB Instance DB of Block_2 [FB1]

Language: DB

Number: 3

○ Manual

● Automatic

2.4.3. **CaoVariable::get_Value method**

Reads the value specified by AddVariable.

Example

```
        object value = variable.Value;
```

2.4.4. **CaoVariable::put_Value method**

Writes the value specified in AddVariable.

When CharArray and WCharArray are specified for Type, it is necessary to specify a string or an array of 1-element strings. If you specify anything other than CharArray and WCharArray in Type, you must specify an array of elements specified in Elem. However, when Elem is 0 or 1, a direct value can be specified.

Example

```
        variable.Value = “ABC”;
```

3. Details of Variable Type

3.1. Bool

Bool reads and writes bitwise values.
Each parameter has the following meaning.

3.1.1. **Address**

The left side of the period represents the read start address, and the right side represents the bit offset of the read start address.

3.1.2. **Elem**

Specifies the number of values to read in bits.

3.1.3. **Attention**

If the bit offset is other than 0 or Elem is not a multiple of 8, data in the affected range is read in byte units, and then the specified position is updated and writing is performed. Because of this, bits that do not write may lose the changes made between read and write.

3.2. Char, Byte, Int, Word, DInt, DWord, LInt, LWord, Real, LReal

Char, Byte, Int, Word, Dint, DWord, LInt, LWord, Real, and LReal read / write big-endian memory images, and Elem is the number of elements in each type of size unit.

The meaning of each Type is as follows.

Value	Type	Size(bytes)	Sign
Char	Integer	1	Signed
Byte	Integer	1	Unsigned
Int	Integer	2	Signed
Word	Integer	2	Unsigned
DInt	Integer	4	Signed
DWord	Integer	4	Unsigned
LInt	Integer	8	Signed
LWord	Integer	8	Unsigned
Real	Real number	4	-
LReal	Real number	8	-

3.3. String, Wstring

String and WString are types for reading / writing memory defined as String and WString on the S7 series as a string.

[Attention]

Since Strings and WStrings are recorded in a special format in the S7 series, do not use Strings and WStrings when accessing areas other than the areas declared as Strings and WStrings on the S7 series.

3.4. CharArray, WCharArray

CharArray and WCharArray are types for retrieving arrays of Char and Int as strings.

3.4.1. **Read operation**

CharArray reads Elem bytes and converts it into an ASCII string.

WCharArray reads Elem * 2 bytes and converts it into big endian Unicode.

3.4.2. **Write operation**

CharArray converts the set value into an array of bytes as an ASCII string and writes it.

WCharArray converts the set value into a byte array as big endian Unicode and writes it.

4. Restrictions for each model

4.1. Restrictions on accessible DataType

Due to the limitations of the libraries used, accessible DataType is limited depending on the model.
The DataType that can be accessed for each model is as follows.
If you specify a DataType that is not accessible, variables can be added successfully, but errors will occur when reading or writing values.

DataType	Corresponding address	S7-300	S7-1500
DataBlock	DB	✓	✓
Input	I, E	✓	✓
Output	Q, A	✓	✓
Memory	M	✓	✓
Counter	C, Z	Readable Write disable	-
Timer	T	Readable Write disable	-

4.2. Restrictions on the types of possible values

• By S7-300, specifying WString in the type specification causes errors when reading and writing values.

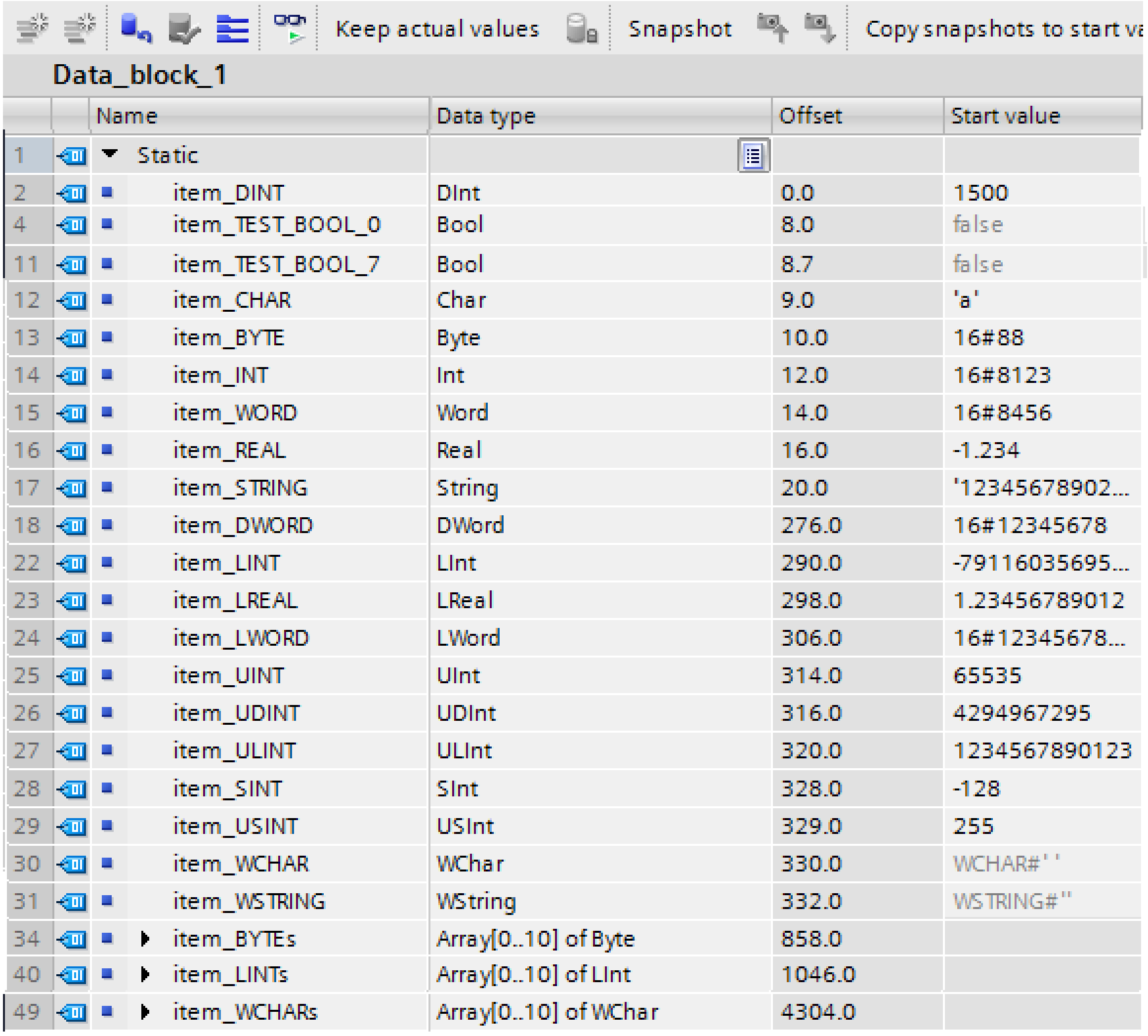
5. Indication of license

5.1. S7.Net+

Appendix A. Addressing examples

The following shows an example of address specification for accessing each data when the following definitions are made on the controller.

[Data of data block defined by TIA Portal]



Data_block_1

	Name	Data type	Offset	Start value
1	Static			
2	item_DINT	DInt	0.0	1500
4	item_TEST_BOOL_0	Bool	8.0	false
11	item_TEST_BOOL_7	Bool	8.7	false
12	item_CHAR	Char	9.0	'a'
13	item_BYTE	Byte	10.0	16#88
14	item_INT	Int	12.0	16#8123
15	item_WORD	Word	14.0	16#8456
16	item_REAL	Real	16.0	-1.234
17	item_STRING	String	20.0	'12345678902...
18	item_DWORD	DWord	276.0	16#12345678
22	item_LINT	LInt	290.0	-79116035695...
23	item_LREAL	LReal	298.0	1.23456789012
24	item_LWORD	LWord	306.0	16#12345678...
25	item_UINT	UInt	314.0	65535
26	item_UDINT	UDInt	316.0	4294967295
27	item_ULINT	ULInt	320.0	1234567890123
28	item_SINT	SInt	328.0	-128
29	item_USINT	USInt	329.0	255
30	item_WCHAR	WChar	330.0	WCHAR#' '
31	item_WSTRING	WString	332.0	WSTRING#''
34	item_BYTEs	Array[0..10] of Byte	858.0	
40	item_LINTs	Array[0..10] of LInt	1046.0	
49	item_WCHARs	Array[0..10] of WChar	4304.0	

Example 1) BOOL type definition (item_TEST_BOOL_0)

DBNo = 1, Type = BOOL, Address = 8.0

Example 2) BOOL type definition (item_TEST_BOOL_7)

DBNo = 1, Type = BOOL, Address = 8.7

Example 3) Byte type definition (item_BYTE)

DBNo = 1, Type = BYTE, Address = 10

Example 4) Int type definition (item_INT)

DBNo = 1, Type = INT, Address = 12

Example 5) DInt type definition (item_DINT)

DBNo = 1, Type = DINT, Address = 0

Example 6) DWord type definition (item_DWORD)

DBNo = 1, Type = DWORD, Address = 276

Example 7) Real type definition (item_REAL)

DBNo = 1, Type = REAL, Address = 16

Example 8) LInt type definition (item_LINT)

DBNo = 1, Type = LINT, Address = 290

Example 9) ULInt type definition (item_ULINT)

DBNo = 1, Type = LWORD, Address = 320

Example 10) LReal type definition (item_LREAL)

DBNo = 1, Type = LREAL, Address = 298

Example 11) Char type definition (item_CHAR)

DBNo = 1, Type = CHAR, Address = 9

DBNo = 1, Type = CharArray, Address = 9

Example 12) WChar type definition (item_WCHAR)

DBNo = 1, Type = WORD, Address = 330

DBNo = 1, Type = WCharArray, Address = 330

Example 13) String type definition (item_STRING)

DBNo = 1, Type = String, Address = 20

Example 14) WString type definition (item_WSTRING)

DBNo = 1, Type = WString, Address = 332

Example 15) Byte type array (11 elements) (item_BYTEs)

DBNo = 1, Type = BYTE, Address = 858, Elem = 11

Example 16) LInt type array (11 elements) (item_LINTs)

DBNo = 1, Type = LINT, Address = 1046, Elem = 11

Example 17) WChar type array (11 elements) (item_WCHAR)

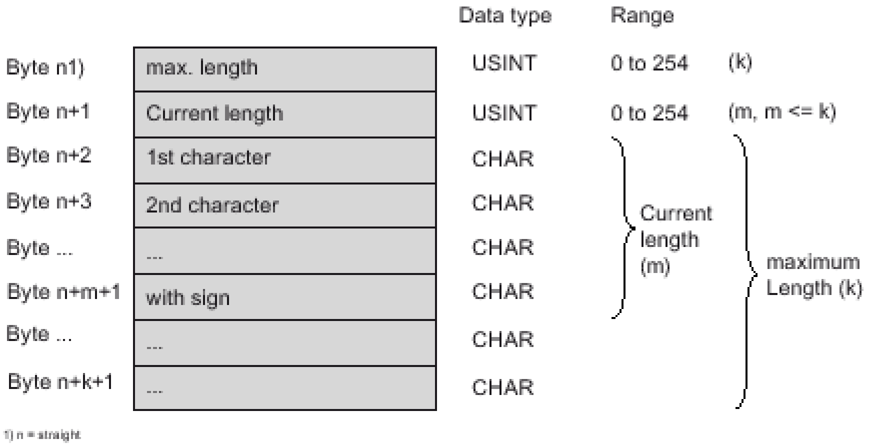
DBNo = 1, Type = WORD, Address = 4304, Elem = 11

DBNo = 1, Type = WCharArray, Address = 4304, Elem = 11

Appendix B. About accessing string types

String types (STRING,WSTRING) are structured as shown in Figure5-1 .

(WSTRING type have 2 bytes for each element).



		Data type	Range		
Byte n1)	max. length	USINT	0 to 254	(k)	
Byte n+1	Current length	USINT	0 to 254	(m, m <= k)	
Byte n+2	1st character	CHAR		Current length (m)	maximum Length (k)
Byte n+3	2nd character	CHAR			
Byte ...	...	CHAR			
Byte n+m+1	with sign	CHAR			
Byte ...	...	CHAR			
Byte n+k+1	...	CHAR			

1) n = straight

Figure5-1 Structure of String types

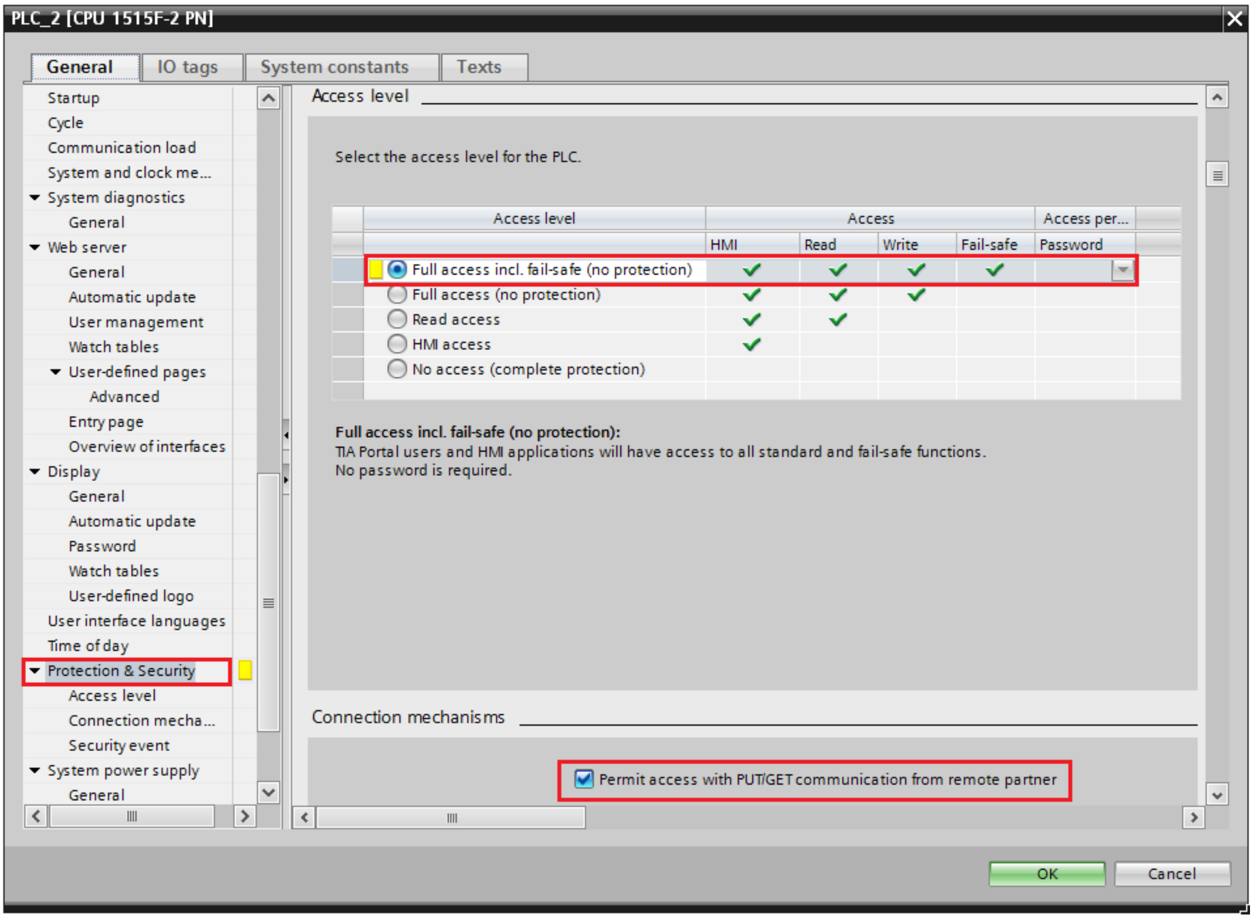
When writing with put_Value, the "maximum length" is acquired once, and the actual data part is written so that the size is not exceeded. After successful writing, write "current length".

When reading by get_Value, the "current length" is acquired, and the actual data portion of the size is read.

Appendix C. Permission for remote access

In the S7-1500[1] series, the [Permit access with PUT / GET communication from remote partner] option is disabled.
In the disabled state, communication from this provider can not be performed, so enable it.

Set the access level to "Full access including fail-safe (no protection)" from "Protection & security[2]" in Fig. 4-2 and check "Permit access with PUT / GET communication from remote partner".



Select the access level for the PLC.

Access level	Access				Access per...
	HMI	Read	Write	Fail-safe	Password
● Full access incl. fail-safe (no protection)	✓	✓	✓	✓	
○ Full access (no protection)	✓	✓	✓		
○ Read access	✓	✓			
○ HMI access	✓				
○ No access (complete protection)					

Full access incl. fail-safe (no protection):
TIA Portal users and HMI applications will have access to all standard and fail-safe functions.
No password is required.

☑ Permit access with PUT/GET communication from remote partner

Figure5-2 Protection and security

[1] There may be a model whose setting is invalidated even in other models. In that case, please enable it in the same way.
[2] The setting screen can be displayed by right-clicking [Target PLC] and selecting [Property].