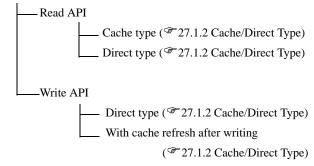
27 Designing Your Own Program

27.1	Using API Functions	27-2
27.2	Device Access APIs	27-20
27.3	Cache Buffer Control APIs	27-41
27.4	Queuing Access Control APIs	27-47
27.5	System APIs	27-50
27.6	SRAM Data Access APIs	27-57
27.7	CF Card / SD Card APIs	27-61
27.8	Binary Date and Time / Text Display Conversion	27-76
27.9	Other APIs	27-80
27.10	Precautions for Using APIs	27-85
27.11	Using APIs (Examples)	27-97

27.1 Using API Functions

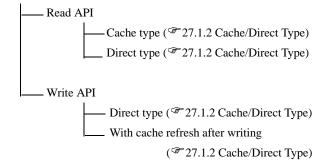
Reading and writing a Device/PLC

Single-handle functions (@27.1.1 Single-/Multi-Handle Functions)



PLC communication with multiple devices

Multi-handle functions (@27.1.1 Single-/Multi-Handle Functions)



For effective communication

- Group symbol access (@27.1.4 Group Access)
- Queuing access (@ 27.1.5 Queuing Access)

Other functions

- System APIs (→27.1.7 System APIs)
- SRAM Data Access APIs (→27.1.8 SRAM Data Access APIs)
- CF Card and SD Card APIs (\rightarrow 27.1.9 CF Card and SD Card APIs)
- Other APIs (\rightarrow 27.9 Other APIs)

27.1.1 Single-/Multi-Handle Functions

Single-Handle APIs

This API is used for sequential communications with target devices. During a call of an API, you cannot call another API.

To call an API, however, you need not perform a troublesome procedure such as 'Pro-Server EX' access handle acquisition.

Multi-Handle APIs

This API enables simultaneous use of single-handle API features for multiple devices. For differentiation from Single-Handle APIs, Multi-Handle APIs are identified with a capital "M" at the end of each API name. For example, a Multi-Handle API that provides the same feature as a Single-Handle API "ReadDeviceVariant()" is named "ReadDeviceVariantM()".

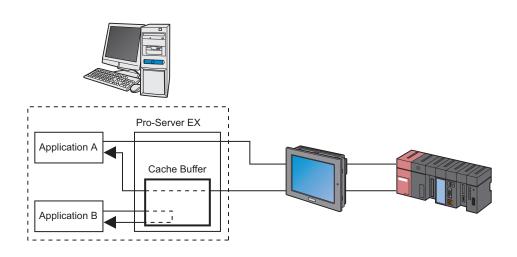
Multi-Handle APIs can be used for multi-thread applications, or for simultaneous access to multiple Devices/ PLCs.

27.1.2 Cache/Direct Type

Cache Read

When multiple applications send reading requests to the same device/PLC, it takes time if 'Pro-Server EX' accesses the Device/PLC to meet individual applications' reading request one by one.

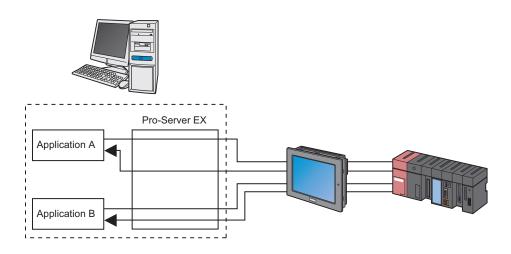
With the Cache Read feature, however, when two applications A and B send reading requests to the same Device/ PLC, 'Pro-Server EX' reads data from the Device/PLC according to the request of Application A first, stores the read data into the internal cache buffer, and sends the data to Application A in response to the reading request. Then, according to the request of Application B, 'Pro-Server EX' sends the data stored in the cache buffer to Application B, since the response data are already stored together with the data for Application A. 'Pro-Server EX' also provides cache buffer control APIs. Refer to "27.3 Cache Buffer Control APIs" for more details.



Direct Read

This feature always reads latest data from a Device/PLC, regardless of cache status.

Direct Read APIs are identified with a capital "D" or "DM" at the end of each API name.



Direct Write

This API writes values. Direct Write APIs are identified with a capital "D" or "DM" at the end of each API name.

Write with Cache Refresh

When caching data from a device, 'Pro-Server EX' rereads the relevant device data after writing values, to refresh the cache data.

The processing speed of this API is lower than that of Direct Write APIs. When 'Pro-Server EX' has cache-read device data, use Write with Cache Refresh.

27.1.3 Cache Buffer Control APIs

Cache Buffer Control APIs allow you to know whether cache data for a target device has been updated or not.

• Cache Buffer Control APIs are not intended to rewrite a network project file, but used to add data to or change data in the internal memory of 'Pro-Server EX'.

Cache Buffer

When caching device data, 'Pro-Server EX' manages multiple devices as a whole. The unit of the management is called "cache buffer".

(1) One cache buffer is comprised of multiple records.

(2) One record can be specified by direct specification of addresses of consecutive multiple devices, by symbol

specification, or by group symbol specification.

(3) You can assign a unique name to each cache buffer.

For cache buffer registration, the following two methods are available:

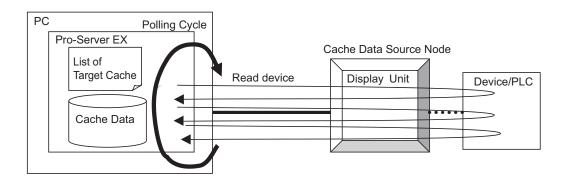
 (1) Registration using 'Pro-Studio EX' (Create a cache buffer in "Device Cache" on the feature screen, and register it in a network project file.)
 (2) Registration using API

Cache buffer updating procedure

To update a cache buffer, "Polling" and "Constant monitoring" methods are available.

The principle of polling method

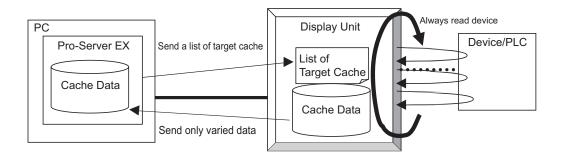
According to a list of target devices in the cache buffer, 'Pro-Server EX' reads device data to update the cache buffer when the cycle specified in cache buffer registration is reached.



◆ The principle of constant monitoring method

At the start of cache buffer updating, 'Pro-Server EX' sends a list of target devices to a data source node. According to the list, the data source node constantly reads device data (as fast as possible), and sends only changed data to 'Pro-Server EX'.

'Pro-Server EX' receives the data, and handles it as cache data.



• When the cache data source node is in the GP Series, the constant monitoring method cannot be used.

Selecting constant monitoring method or polling method

If a large volume of device data are monitored with the constant monitoring method, then 'Pro-Server EX' is engaged in monitoring, resulting in deterioration of the whole system performance.

To prevent this, it is recommended to select the constant monitoring method only for highly-urgent items, and to use the polling method for other items.

With the polling method, the cache buffer may not be updated according to the update cycle, depending on your PC or network conditions, types of Device/PLCs, and performance of your system. In this case, use Direct Read APIs.

As standard data volume acceptable with each method, the constant monitoring method can handle up to tens of bytes to hundreds of bytes, and the polling method can handle up to several kilobytes. For a larger data volume, use Direct Read APIs.

Note that the allowable number of bytes varies depending on performance of your system.

Starting and Stopping Caching

'Pro-Server EX' caching start/stop timing is described below.

(1) Caching starts or stops by cache buffer.

(2) To register a cache buffer in a network project file with 'Pro-Studio EX', the following three types of

registration methods can be selected for each cache buffer. The caching start timing for each method is as follows.

1) At start of 'Pro-Server EX'

After 'Pro-Server EX' starts and a network project is loaded, 'Pro-Server EX' starts caching.

When a network project is reloaded, 'Pro-Server EX' also starts caching.

2) Starting caching automatically when a pre-registered device is read

When a Device Read API is issued for a cache device registered in the cache buffer, 'Pro-Server EX' starts caching.

Even if reading is executed for some of the devices registered in the cache buffer, 'Pro-Server EX' starts caching for all registered devices.

Caching can be started by all the reading methods as well as Device Read APIs. (For example, when a device is specified as a data source for a data transfer function, or when a device is subjected to start condition check, caching starts.)

However, only when caching is started with the method 2), 'Pro-Server EX' stops caching if there is no access to the target device in the cache buffer for a specified period.

3) Starting caching with a program using Cache Buffer Start API (PS_StartCache)

(3) In the following conditions, 'Pro-Server EX' stops caching.

1) When 'Pro-Server EX' is closed, the cache buffer stops, and discards cache data.

2) Immediately before a network project is reloaded, the cache buffer stops, and discards cache data.

3) When the function of "Automatically start when a registered device is read" is enabled, and the cache buffer is not accessed within a specified stop time after start of caching, the cache buffer stops. (Cache data will not be discarded.)

4) When the cache buffer is stopped with a program by using the Cache Stop API (PS_StopCache).

27.1.4 Group Access

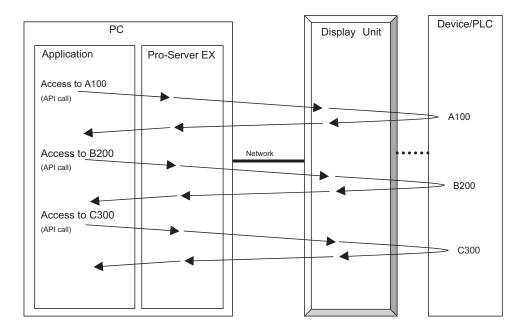
Some APIs use a group symbol to specify a device address.

With a group symbol, 'Pro-Server EX' can efficiently access multiple devices with a single call of an API.

- When 'Pro-Server EX' accesses devices by using a group symbol comprised of multiple devices, each access speed becomes high, and 'Pro-Server EX' and display unit internally optimize the processing. Therefore, you cannot specify the device access order. (The registration order of symbols in group symbol registration does not mean the access order.) If an access error occurs with any one of the multiple devices, the processing will stop. 'Pro-Server EX' recognizes it as the whole group access error, and will not execute access to the remaining devices.
 - The maximum group symbol data size available with a single call of an API is 1 Mbyte.

When calling API individually for each device:

Every time the API is called, 'Pro-Server EX' communicates with the device.

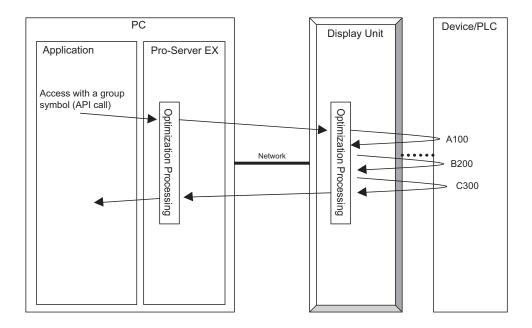


When accessing group symbols

Operation differs depending on the type of node.

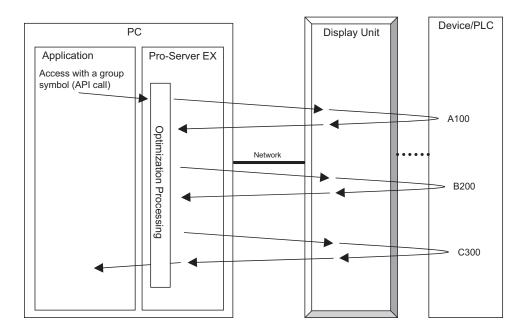
For SP-5B40/WinGP node, SP-5B10 node, GP4000/LT4000 Series node, GP3000 Series node or LT3000 node

'Pro-Server EX' sends a request for each node only once. The node internally divides the request to access each device separately. Thus, 'Pro-Server EX' can efficiently communicate with the devices on the network.



• For GP Series node

The API is called only once, and 'Pro-Server EX' internally divides the request to access each GP Series node separately. However, if the group has several consecutive symbols, 'Pro-Server EX' accesses these symbols at once.

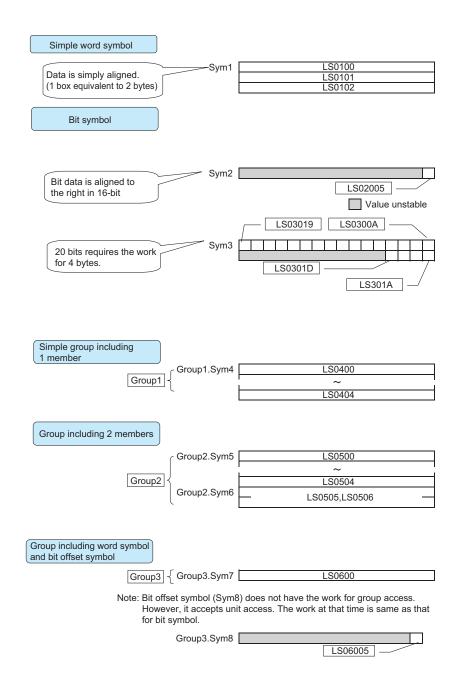


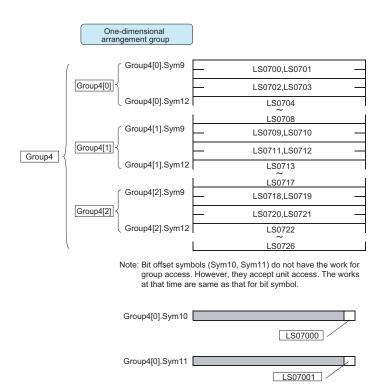
Data structure for group symbol access

When 'Pro-Server EX' accesses devices via a group symbol, the data buffer structure varies depending on the symbol type or size of the group. The data buffer structure by group symbol type is as follows:

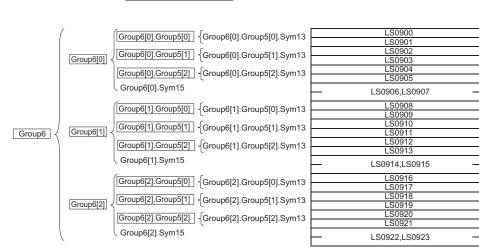
Group symbol data type	Secured data size
Bit Data	 For bit symbol Data buffer is secured in multiples of 16 bits. For bit offset symbol No data buffer is secured.
8-bit (Signed) Data	
8-bit (Unsigned) Data	Data buffer of 1 byte/device is secured. Binary value is used.
8-bit (HEX) Data	
8-bit (BCD) Data	Data buffer of 1 byte/device is secured. During access to a device, 'Pro-Server EX' executes BCD-Binary conversion.
16-bit (Signed) Data	
16-bit (Unsigned) Data	Data buffer of 2 bytes/device is secured. Binary value is used.
16-bit (HEX) Data	
16-bit (BCD) Data	Data buffer of 2 bytes/device is secured. During access to a device, 'Pro-Server EX' executes BCD-Binary conversion.
32-bit(Signed)Data	
32-bit(Unsigned)Data	Data buffer of 4 bytes/device is secured. Binary value is used.
32-bit(HEX)Data	
32-bit(BCD)Data	Data buffer of 4 bytes/device is secured. During access to a device, 'Pro-Server EX' executes BCD-Binary conversion.
Single-precision floating point	Data buffer of 4 bytes/device is secured. The value is handled as a single- precision floating point value.
Double-precision floating point	Data buffer of 8 bytes/device is secured. The value is handled as a single- precision floating point value.
Character string data	Data buffer of 1 byte/character is secured. The data is handled as a NULL-terminated character string.
TIME Data	
TIME_OF_DAY Data	Data buffer of 1 device/4 bytes is secured. When accessing actual device, binary value with internal format is converted to value with external device format.
DATE Data	
DATE_AND_TIME Data	Data buffer of 1 device/8 bytes is secured. When accessing actual device, binary value with internal format is converted to value with external device format.

Examples of the data buffer structures are shown below.





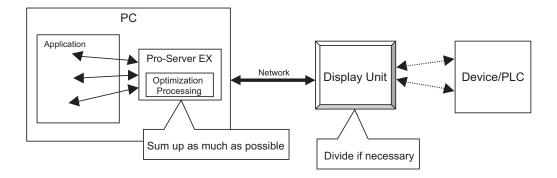
Device addresses for Group4[1].Sym10 and Group4[1].Sym11 are LS07090 and LS07091 respectively. Device addresses for Group4[2].Sym10 and Group4[2].Sym11 are LS0718 and LS07181 respectively.



Two-dimensional group (nest of groups)

27.1.5 Queuing Access

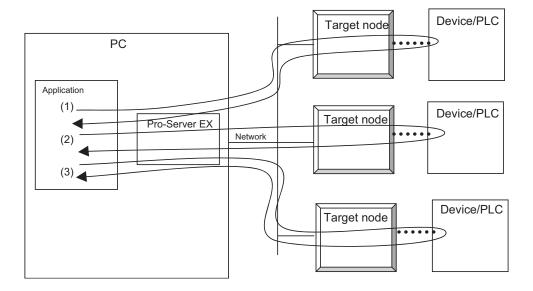
'Pro-Server EX' stores a device access request every time an API is called, and then optimizes the stored requests to access individual devices at once.



The principle of queuing access

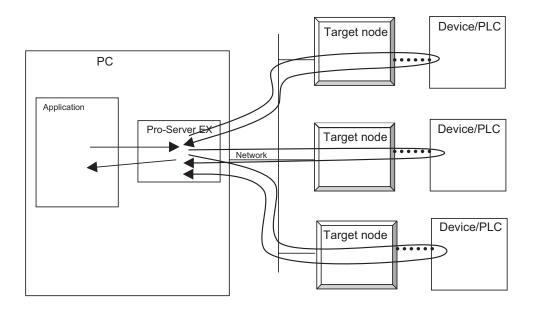
♦ Simple API access

'Pro-Server EX' executes sequential processing.



Queuing access

'Pro-Server EX' executes parallel processing for individual nodes.



Procedures for use

(1) Declare start of queuing access. (Call BeginQueuingRead() or BeginQueungWrite().)

(2) Call a Device Read or Device Write API.

(For example, call ReadDevice16() or WriteDevice16().)

If the argument is normal, the API is returned soon, and 'Pro-Server EX' stores the device access request only. This step is called "Access request registration".

(3) To execute the stored device access request actually, call ExecuteQueuingAccess(). In this step, 'Pro-Server EX' optimizes the device access request, and tries to communicate with the devices efficiently.

If 'Pro-Server EX' successfully accesses all specified devices, ExecuteQueuingAccess() returns a success code. If 'Pro-Server EX' fails to access any device, on the other hand, ExecuteQueuingAccess() returns an access error code.

If you wish to know whether each device access request has been successfully executed or not, call IsQueuingAcceessSucceeded() to check the result.

IMPORTANT	During "Access request registration", 'Pro-Server EX' stores the access data buffer address (address only, excluding data).
	Therefore, when running "Access request registration", the data buffer address passed to
	each API must continue to exist until ExecuteQueuingAccess() returns a value after it is called.
	Otherwise, 'Pro-Server EX' will access an invalid address and forcibly exit.
	Also, when queuing qccess is used again, the data buffer must remain in the address
	specified in "access request registration".
NOTE	When registering access requests, 'Pro-Server EX' remembers the data buffer's address that was
	used for access. (Remembers the address only, not the data.)
	As a result,
•	When using queuing access, you cannot register read access and write access simultaneously. For
	example, after declaration of start of queuing access for read access, write access cannot be
	registered. Also, after declaration of start of queuing access for write access, read access cannot be
	registered.
	However, since queuing access is registered for each Pro-Server handle, you can register write
	access and read access separately for different Pro-Server handles.
•	Once an access request is registered, you need not re-register it when you try to access the same
	device with the same method.
	Since 'Pro-Server EX' stores an access request per Pro-Server handle, it will be executed repeatedly
	based on the stored data, every time ExecuteQueuingAccess() is called.
	Access request registration memory will be cleared in the following cases:
	(1) When a stored Pro-Server handle is discarded.
	(2) When new queuing access registration is started.
	(3) When existing queuing access registration is cancelled (CancelQueuingAccess() is called). If a
	function other than Converting error code into character string(EasyLoadErrorMessage etc.) is
	executed after execution of ExecuteQueuingAccess(), 'Pro-Server EX' cancels existing queuing
	data, and starts new queuing access registration.

27.1.6 Bit Data Access

To access bit devices, 'Pro-Server EX' provides the following three types of bit data handling methods:

(1) Handling bit data in multiples of 16 bits: Bit devices are handled as bit strings in multiples of 16 bits.

A specified quantity of bit data are stored and used from bit D0 (right end).

Even if only one device is specified, a 16-bit data buffer is required. Data buffers are required in multiples of 16 bits, depending on the specified number of devices.

(Example) Data buffer storing order for 20 bit devices

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
16	15	14	13	12	10	11	10	9	8	7	6	5	3	2	1
*	*	*	*	*	*	*	*	*	*	*	*	20	19	18	17

< Applicable API >

When data type "1" (EASY_AppKind_Bit) is specified for ReadDeviceBit/WriteDeviceBit(), ReadDevice/ WriteDevice() or ReadDeviceVariant/WriteDeviceVariant();

When a bit symbol, or a group including a bit symbol is specified for ReadSymbol/WriteSymbol()

(2) Handling bit data as Variant BOOL data: One bit is handled as Variant BOOL data.

The data buffer handles one piece of Variant BOOL data for one bit. BOOL data alignments as many as the specified number of devices are provided.

< Applicable API >

When data type "0x201" (EASY_AppKind_BOOL) is specified for ReadDeviceVariant/WriteDeviceVariant(); When a bit symbol, or a group including a bit symbol is specified for ReadSymbolVariant/WriteSymbolVariant()

(3) Handling bit offset symbol for group symbol access

If you access a device by directly specifying a bit offset symbol, the data buffer handles "Strings in multiples of 16 bits", or "Variant BOOL data", as described in the above section.

However, when you access a device by using a group symbol that includes a bit offset symbol, a data area for the bit offset symbol is not secured in the data buffer.

A bit offset symbol cannot exist by itself without a word symbol, or a parent symbol. The data area is secured for this parent symbol, and you can use a part of that area for the bit offset symbol.

Refer to "27.1.4 Group Access" for more details.

27.1.7 System APIs

System APIs are intended for system control, such as starting or closing 'Pro-Server EX', loading network project files and so on.

The system APIs are classified into the following categories:

Single-Handle APIs

You can use the 'Pro-Server EX' features without specifying a Pro-Server handle. With this method, multiple APIs cannot be simultaneously used. (If you try to use multiple APIs simultaneously,

the double-call error occurs.)

Multi-Handle APIs

You can use the 'Pro-Server EX' features by specifying a Pro-Server handle. You can use multiple APIs simultaneously by specifying different Pro-Server handles.

27.1.8 SRAM Data Access APIs

The SRAM incorporated in the display unit Series stores various data depending on the display unit setup and operating conditions.

The following APIs are intended to access data stored in the SRAM.

All SRAM Data Access APIs support both Single-Handle and Multi-Handle functions.

This section describes Single-Handle APIs. Multi-Handle APIs are identified with "M" at the end of each API name, and a Pro-Server handle is added to the first argument.

27.1.9 CF Card and SD Card APIs

API for accessing data on CF and SD cards.

Like SRAM, stores various data depending on the display unit setup and operating conditions.

27.2 Device Access APIs

■ Single-Handle Cache Read APIs

INT WINAPI ReadDeviceBFunction8-bINT WINAPI ReadDevice8(Function16-INT WINAPI ReadDevice1(Function32-INT WINAPI ReadDevice32Function8-bINT WINAPI ReadDevice32Function8-bINT WINAPI ReadDeviceBFunction16-	it data Bit(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount); bit data B(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* obData,WORD wCount); 5-bit data 6(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount); 2-bit data B2(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* odwData,WORD wCount); bit BCD data BCD8(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* obData,WORD wCount); 5-bit BCD data BCD16(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount); 2-bit BCD data BCD16(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount); 2-bit BCD data BCD16(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount); 2-bit BCD data BCD32(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* odwData,WORD
Function8-bINT WINAPI ReadDevice8(Function16-INT WINAPI ReadDevice1(Function32-INT WINAPI ReadDevice32Function8-bINT WINAPI ReadDeviceBFunction16-	bit data B(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* obData,WORD wCount); 5-bit data 6(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount); 2-bit data B2(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* odwData,WORD wCount); bit BCD data BCD8(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* obData,WORD wCount); 5-bit BCD data BCD16(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount); 2-bit BCD data
INT WINAPI ReadDevice8(Function16-INT WINAPI ReadDevice1032-INT WINAPI ReadDevice3254-Function8-bINT WINAPI ReadDeviceB6-Function16-	B(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* obData,WORD wCount); 5-bit data 6(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount); 2-bit data B2(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* odwData,WORD wCount); bit BCD data BCD8(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* obData,WORD wCount); 5-bit BCD data BCD16(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount); 2-bit BCD data
Function16-INT WINAPI ReadDevice10Function32-INT WINAPI ReadDevice32Function8-bINT WINAPI ReadDeviceBFunction16-	5-bit data 6(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount); 2-bit data 32(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* odwData,WORD wCount); bit BCD data 3CD8(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* obData,WORD wCount); 5-bit BCD data 3CD16(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount); 2-bit BCD data
INT WINAPI ReadDevice10Function32-INT WINAPI ReadDevice32Function8-bINT WINAPI ReadDeviceBFunction16-	6(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount); 2-bit data 32(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* odwData,WORD wCount); bit BCD data 3CD8(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* obData,WORD wCount); 5-bit BCD data 3CD16(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount); 2-bit BCD data
Function32-INT WINAPI ReadDevice32Function8-bINT WINAPI ReadDeviceBFunction16-	2-bit data 32(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* odwData,WORD wCount); bit BCD data 3CD8(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* obData,WORD wCount); 5-bit BCD data 3CD16(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount); 2-bit BCD data
INT WINAPI ReadDevice32 Function 8-b INT WINAPI ReadDeviceB Function 16-	22(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* odwData,WORD wCount); bit BCD data 3CD8(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* obData,WORD wCount); 5-bit BCD data 3CD16(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount); 2-bit BCD data
Function8-bINT WINAPI ReadDeviceBFunction16-	bit BCD data BCD8(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* obData,WORD wCount); 5-bit BCD data BCD16(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount); 2-bit BCD data
INT WINAPI ReadDeviceB Function 16-	CD8(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* obData,WORD wCount); 5-bit BCD data 3CD16(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount); 2-bit BCD data
Function 16-	5-bit BCD data 3CD16(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount); 2-bit BCD data
	3CD16(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount); 2-bit BCD data
INT WINAPI PandDavicaB	2-bit BCD data
INT WINAIT ReadDeviceD	
Function 32-	3CD32(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* odwData,WORD
INT WINAPI ReadDeviceB wCount);	
Function Sin	ngle-precision floating point data
INT WINAPI ReadDeviceF	Float(LPCSTR sNodeName,LPCSTR sDeviceName,FLOAT* ofIData,WORD wCount);
Function Do	ouble-precision floating point data
INT WINAPI ReadDeviceD wCount);	Double(LPCSTR sNodeName,LPCSTR sDeviceName,DOUBLE* odbData,WORD
Function Ch	haracter string data
INT WINAPI ReadDeviceSt	Str(LPCSTR sNodeName,LPCSTR sDeviceName,LPSTR psData,WORD wCount);
Function Ge	eneral-use data
INT WINAPI ReadDevice(I wAppKind);	LPCSTR sNodeName,LPCSTR sDeviceName,LPVOID pData,WORD wCount,WORD
Function Ge	eneral-use data (Variant-type)
INT WINAPI ReadDeviceV wCount,WORD wAppKind)	/ariant(LPCSTR sNodeName,LPCSTR sDeviceName,LPVARIANT pData,WORD));
Function Gro	roup symbol
INT WINAPI ReadSymbol((LPCSTR sNodeName,LPCSTR sSymbolName,LPVOID oReadBufferData);
Function Gro	roup symbol (Variant-type)
INT WINAPI ReadSymbolV	Variant(LPCSTR sNodeName,LPCSTR sSymbolName,LPVARIANT pData);
Function TI	IME data
INT WINAPI ReadDeviceT wCount);	TIME(LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* odwData, WORD

Function	DATE data				
INT WINAPI ReadDeviceDATE(LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* odwData, WORD wCount);					
Function TIME_OF_DAY data					
INT WINAPI ReadDeviceTIME_OF_DAY(LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* odwData, WORD wCount);					
Function	DATE_AND_TIME data				
INT WINAPI ReadDeviceDATE_AND_TIME(LPCSTR sNodeName, LPCSTR sDeviceName, QWORD* oqwData, WORD wCount);					

* For each parameter, please refer to "■ Parameters of read/write functions".

* You can convert binary values read from TIME, DATE, TIME_OF_DAY, and DATE_AND_TIME data to text format.

For information about text conversion, refer to "27.8 Binary Date and Time / Text Display Conversion".

■ Single-Handle Direct Read APIs

Function	Bit data
INT WINAPI ReadDev	iceBitD(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount);
Function	8-bit data
INT WINAPI ReadDev	ice8D(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* obData,WORD wCount);
Function	16-bit data
INT WINAPI ReadDev	ice16D(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount);
Function	32-bit data
INT WINAPI ReadDev	ice32D(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* odwData,WORD wCount);
Function	8-bit BCD data
IINT WINAPI ReadDev	viceBCD8D(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* obData,WORD wCount);
Function	16-bit BCD data
INT WINAPI ReadDevi	iceBCD16D(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount);
Function	32-bit BCD data
INT WINAPI ReadDev wCount);	iceBCD32D(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* odwData,WORD
Function	Single-precision floating point data
INT WINAPI ReadDev	iceFloatD(LPCSTR sNodeName,LPCSTR sDeviceName,FLOAT* ofIData,WORD wCount);
Function	Double-precision floating point data
INT WINAPI ReadDev wCount);	iceDoubleD(LPCSTR sNodeName,LPCSTR sDeviceName,DOUBLE* odbData,WORD
Function	Character string data
INT WINAPI ReadDev	iceStrD(LPCSTR sNodeName,LPCSTR sDeviceName,LPSTR psData,WORD wCount);
Function	General-use data
INT WINAPI ReadDev wAppKind);	iceD(LPCSTR sNodeName,LPCSTR sDeviceName,LPVOID pData,WORD wCount,WORD
Function	General-use data (Variant-type)
INT WINAPI ReadDev wCount,WORD wAppk	iceVariantD(LPCSTR sNodeName,LPCSTR sDeviceName,LPVARIANT pData,WORD Kind);
Function	Group symbol
INT WINAPI ReadSym	bolD(LPCSTR sNodeName,LPCSTR sSymbolName,LPVOID oReadBufferData);
Function	Group symbol (Variant-type)
INT WINAPI ReadSym	bolVariantD(LPCSTR sNodeName,LPCSTR sSymbolName,LPVARIANT pData);
Function	TIME data
INT WINAPI ReadDev wCount);	iceTIMED(LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* odwData, WORD
Function	DATE data
INT WINAPI ReadDev wCount);	iceDATED(LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* odwData, WORD

Function	TIME_OF_DAY data					
INT WINAPI ReadDevi WORD wCount);	INT WINAPI ReadDeviceTIME_OF_DAYD(LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* odwData, WORD wCount);					
Function	DATE_AND_TIME data					
INT WINAPI ReadDeviceDATE_AND_TIMED(LPCSTR sNodeName, LPCSTR sDeviceName, QWORD* oqwData, WORD wCount);						

* For each parameter, please refer to "
Parameters of read/write functions".

* You can convert binary values read from TIME, DATE, TIME_OF_DAY, and DATE_AND_TIME data to text format.

For information about text conversion, refer to "27.8 Binary Date and Time / Text Display Conversion".

■ Single-Handle Direct Write APIs

Function	Bit data
INT WINAPI WriteDev	iceBitD(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* pwData,WORD wCount);
Function	8-bit data
INT WINAPI WriteDev	ice8D(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* pbData,WORD wCount);
Function	16-bit data
INT WINAPI WriteDev	ice16D(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* pwData,WORD wCount);
Function	32-bit data
INT WINAPI WriteDev	ice32D(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* pdwData,WORD wCount);
Function	8-bit BCD data
INT WINAPI WriteDev	iceBCD8D(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* pbData,WORD wCount);
Function	16-bit BCD data
INT WINAPI WriteDev wCount);	iceBCD16D(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* pwData,WORD
Function	32-bit BCD data
INT WINAPI WriteDev wCount);	iceBCD32D(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* pdwData,WORD
Function	Single-precision floating point data
INT WINAPI WriteDev	iceFloatD(LPCSTR sNodeName,LPCSTR sDeviceName,FLOAT* pflData,WORD wCount);
Function	Double-precision floating point data
INT WINAPI WriteDev wCount);	iceDoubleD(LPCSTR sNodeName,LPCSTR sDeviceName,DOUBLE* pdbData,WORD
Function	Character string data
INT WINAPI WriteDev	iceStrD(LPCSTR sNodeName,LPCSTR sDeviceName,LPCSTR psData,WORD wCount);
Function	General-use data
INT WINAPI WriteDev wAppKind);	iceD(LPCSTR sNodeName,LPCSTR sDeviceName,LPVOID pData,WORD wCount,WORD
Function	General-use data (Variant-type)
INT WINAPI WriteDev wCount,WORD wAppK	iceVariantD(LPCSTR sNodeName,LPCSTR sDeviceName,LPVARIANT pData,WORD [ind);
Function	Group symbol
INT WINAPI WriteSym	bolD(LPCSTR sNodeName,LPCSTR sSymbolName,LPVOID pWriteBufferData);
Function	Group symbol (Variant-type)
INT WINAPI WriteSym	bolVariantD(LPCSTR sNodeName,LPCSTR sSymbolName,LPVARIANT pData);
Function	TIME data
INT WINAPI WriteDev wCount);	iceTIMED(LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* pdwData, WORD
Function	DATE data
INT WINAPI WriteDev wCount);	iceDATED(LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* pdwData, WORD

Function	TIME_OF_DAY data				
INT WINAPI WriteDeviceTIME_OF_DAYD(LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* pdwData, WORD wCount);					
Function	DATE_AND_TIME data				
INT WINAPI WriteDeviceDATE_AND_TIMED(LPCSTR sNodeName, LPCSTR sDeviceName, QWORD* pqwData, WORD wCount);					

* For each parameter, please refer to "
Parameters of read/write functions".

* You can convert binary values written to TIME, DATE, TIME_OF_DAY, and DATE_AND_TIME data from text format. For information about text conversion, refer to "27.8 Binary Date and Time / Text Display Conversion".

■ Single-Handle Write APIs with Cache Refresh after Writing

Function	Bit data
INT WINAPI WriteDev	iceBit(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* pwData,WORD wCount);
Function	8-bit data
INT WINAPI WriteDev	ice8(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* pbData,WORD wCount);
Function	16-bit data
INT WINAPI WriteDev	ice16(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* pwData,WORD wCount);
Function	32-bit data
INT WINAPI WriteDev	ice32(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* pdwData,WORD wCount);
Function	8-bit BCD data
INT WINAPI WriteDev	iceBCD8(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* pbData,WORD wCount);
Function	16-bit BCD data
INT WINAPI WriteDev	iceBCD16(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* pwData,WORD wCount);
Function	32-bit BCD data
INT WINAPI WriteDev wCount);	iceBCD32(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* pdwData,WORD
Function	Single-precision floating point data
INT WINAPI WriteDev	iceFloat(LPCSTR sNodeName,LPCSTR sDeviceName,FLOAT* pflData,WORD wCount);
Function	Double-precision floating point data
INT WINAPI WriteDev wCount);	iceDouble(LPCSTR sNodeName,LPCSTR sDeviceName,DOUBLE* pdbData,WORD
Function	Character string data
INT WINAPI WriteDev	iceStr(LPCSTR sNodeName,LPCSTR sDeviceName,LPCSTR psData,WORD wCount);
Function	General-use data
INT WINAPI WriteDev wAppKind);	ice(LPCSTR sNodeName,LPCSTR sDeviceName,LPVOID pData,WORD wCount,WORD
Function	General-use data (Variant-type)
INT WINAPI WriteDev wCount,WORD wAppK	iceVariant(LPCSTR sNodeName,LPCSTR sDeviceName,LPVARIANT pData,WORD [ind);
Function	Group symbol
INT WINAPI WriteSym	bol(LPCSTR sNodeName,LPCSTR sSymbolName,LPVOID pWriteBufferData);
Function	Group symbol (Variant-type)
INT WINAPI WriteSym	bolVariant(LPCSTR sNodeName,LPCSTR sSymbolName,LPVARIANT pData);
Function	TIME data
INT WINAPI WriteDev wCount);	iceTIME(LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* pdwData, WORD
Function	DATE data
INT WINAPI WriteDev wCount);	iceDATE(LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* pdwData, WORD

Pro-Server EX Reference Manual

Function	TIME_OF_DAY data				
INT WINAPI WriteDev WORD wCount);	INT WINAPI WriteDeviceTIME_OF_DAY(LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* pdwData, WORD wCount);				
Function	DATE_AND_TIME data				
INT WINAPI WriteDeviceDATE_AND_TIME(LPCSTR sNodeName, LPCSTR sDeviceName, QWORD* pqwData, WORD wCount);					

* For each parameter, please refer to "
Parameters of read/write functions".

* You can convert binary values written to TIME, DATE, TIME_OF_DAY, and DATE_AND_TIME data from text format. For information about text conversion, refer to "27.8 Binary Date and Time / Text Display Conversion".

Multi-Handle Cache Read APIs

Function	Bit data
INT WINAPI ReadDevi owData,WORD wCount	ceBitM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD*
Function	8-bit data
INT WINAPI ReadDevi obData,WORD wCount	ce8M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,BYTE*);
Function	16-bit data
INT WINAPI ReadDevi owData,WORD wCount	ce16M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* t);
Function	32-bit data
INT WINAPI ReadDevi odwData,WORD wCou	ce32M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* nt);
Function	8-bit BCD data
INT WINAPI ReadDevi obData,WORD wCount	ceBCD8M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,BYTE*);
Function	16-bit BCD data
INT WINAPI ReadDevi owData,WORD wCount	ceBCD16M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* t);
Function	32-bit BCD data
INT WINAPI ReadDevi odwData,WORD wCou	ceBCD32M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* nt);
Function	Single-precision floating point data
INT WINAPI ReadDevi oflData,WORD wCount	ceFloatM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,FLOAT*);
Function	Double-precision floating point data
INT WINAPI ReadDevi odbData,WORD wCour	ceDoubleM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,DOUBLE* at);
Function	Character string data
INT WINAPI ReadDevi psData,WORD wCount	ceStrM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,LPSTR);
Function	General-use data
INT WINAPI ReadDevi pData,WORD wCount,V	ceM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,LPVOID WORD wAppKind);
Function	General-use data (Variant-type)
	ceVariantM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR ANT pData,WORD wCount,WORD wAppKind);
Function	Group symbol
INT WINAPI ReadSym oReadBufferData);	bolM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sSymbolName,LPVOID

Function	Group symbol (Variant-type)									
INT WINAPI ReadSyn sSymbolName,LPVAR	ibolVariantM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR IANT pData);									
Function	IME data									
INT WINAPI ReadDev odwData, WORD wCo	iceTIMEM(HANDLE hProServer, LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* unt);									
Function	DATE data									
INT WINAPI ReadDev odwData, WORD wCo	iceDATEM(HANDLE hProServer, LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* unt);									
Function	TIME_OF_DAY data									
INT WINAPI ReadDev DWORD* odwData, W	iceTIME_OF_DAYM(HANDLE hProServer, LPCSTR sNodeName, LPCSTR sDeviceName, ORD wCount);									
Function	DATE_AND_TIME data									
	iceDATE_AND_TIMEM(HANDLE hProServer, LPCSTR sNodeName, LPCSTR D* oqwData, WORD wCount);									
* For each parameter, pl	ease refer to "■ Parameters of read/write functions".									

* You can convert binary values read from TIME, DATE, TIME_OF_DAY, and DATE_AND_TIME data to text format.

For information about text conversion, refer to "27.8 Binary Date and Time / Text Display Conversion".

Multi-Handle Direct Read APIs

Function	Bit data
INT WINAPI ReadDevi owData,WORD wCoun	ceBitDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* t);
Function	8-bit data
INT WINAPI ReadDevi obData,WORD wCount	ce8DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,BYTE*);
Function	16-bit data
INT WINAPI ReadDevi owData,WORD wCount	ce16DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* t);
Function	32-bit data
INT WINAPI ReadDevi odwData,WORD wCour	ce32DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* nt);
Function	8-bit BCD data
INT WINAPI ReadDevi obData,WORD wCount	ceBCD8DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,BYTE*);
Function	16-bit BCD data
INT WINAPI ReadDevi owData,WORD wCount	ceBCD16DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD*
Function	32-bit BCD data
INT WINAPI ReadDevi odwData,WORD wCou	ceBCD32DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* nt);
Function	Single-precision floating point data
INT WINAPI ReadDevi oflData,WORD wCount	ceFloatDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,FLOAT*);
Function	Double-precision floating point data
	ceDoubleDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR 2* odbData,WORD wCount);
Function	Character string data
INT WINAPI ReadDevi psData,WORD wCount	ceStrDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,LPSTR);
Function	General-use data
INT WINAPI ReadDevi pData,WORD wCount,V	ceDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,LPVOID WORD wAppKind);
Function	General-use data (Variant-type)
	ceVariantDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR ANT pData,WORD wCount,WORD wAppKind);
Function	Group symbol
INT WINAPI ReadSym oReadBufferData);	bolDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sSymbolName,LPVOID
Function	Group symbol (Variant-type)
INT WINAPI ReadSym sSymbolName,LPVARI	bolVariantDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR ANT pData);
	27.20

Function	TIME data								
INT WINAPI ReadDeviceTIMEDM(HANDLE hProServer, LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* odwData, WORD wCount);									
Function	ATE data								
INT WINAPI ReadDeviceDATEDM(HANDLE hProServer, LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* odwData, WORD wCount);									
Function	TIME_OF_DAY								
INT WINAPI ReadDeviceTIME_OF_DAYDM(HANDLE hProServer, LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* odwData, WORD wCount);									
Function	Function DATE_AND_TIME data								
INT WINAPI ReadDeviceDATE_AND_TIMEDM(HANDLE hProServer, LPCSTR sNodeName, LPCSTR sDeviceName, QWORD* oqwData, WORD wCount);									

* For each parameter, please refer to "■ Parameters of read/write functions".

* You can convert binary values read from TIME, DATE, TIME_OF_DAY, and DATE_AND_TIME data to text format.

For information about text conversion, refer to "27.8 Binary Date and Time / Text Display Conversion".

Multi-Handle Direct Write APIs

Function	Bit data									
INT WINAPI WriteDevi pwData,WORD wCount	iceBitDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD*);									
Function	8-bit data									
INT WINAPI WriteDevi pbData,WORD wCount)	ice8DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,BYTE*);									
Function	16-bit data									
INT WINAPI WriteDevi pwData,WORD wCount	ice16DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD*);									
Function	32-bit data									
INT WINAPI WriteDevi pdwData,WORD wCour	ice32DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* nt);									
Function	8-bit BCD data									
INT WINAPI WriteDevi pbData,WORD wCount)	iceBCD8DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,BYTE*);									
Function	16-bit BCD data									
INT WINAPI WriteDevi pwData,WORD wCount	iceBCD16DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD*);									
Function	32-bit BCD data									
	iceBCD32DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR * pdwData,WORD wCount);									
Function	Single-precision floating point data									
INT WINAPI WriteDevi pflData,WORD wCount	iceFloatDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,FLOAT*);									
Function	Double-precision floating point data									
	iceDoubleDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR * pdbData,WORD wCount);									
Function	Character string data									
INT WINAPI WriteDevi psData,WORD wCount)	iceStrDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,LPCSTR ;									
Function	General-use data									
INT WINAPI WriteDevi pData,WORD wCount,W	iceDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,LPVOID WORD wAppKind);									
Function	General-use data (Variant-type)									
	iceVariantDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR aNT pData,WORD wCount,WORD wAppKind);									
Function	Group symbol									
INT WINAPI WriteSym pWriteBufferData);	bolDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sSymbolName,LPVOID									
Function	Group symbol (Variant-type)									
INT WINAPI WriteSym sSymbolName,LPVARI	bolVariantDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR ANT pData);									
	22 Manual 27-32									

Pro-Server EX Reference Manual

Function	TIME data								
INT WINAPI WriteDeviceTIMEDM(HANDLE hProServer, LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* pdwData, WORD wCount);									
Function	ATE data								
INT WINAPI WriteDeviceDATEDM(HANDLE hProServer, LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* pdwData, WORD wCount);									
Function	TIME_OF_DAY data								
INT WINAPI WriteDeviceTIME_OF_DAYDM(HANDLE hProServer, LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* pdwData, WORD wCount);									
Function	Function DATE_AND_TIME data								
INT WINAPI WriteDeviceDATE_AND_TIMEDM(HANDLE hProServer, LPCSTR sNodeName, LPCSTR sDeviceName, QWORD* pqwData, WORD wCount);									

* For each parameter, please refer to "
Parameters of read/write functions".

* You can convert binary values written to TIME, DATE, TIME_OF_DAY, and DATE_AND_TIME data from text

format. For information about text conversion, refer to "27.8 Binary Date and Time / Text Display Conversion".

■ Multi-Handle Write APIs with Cache Refresh after Writing

Function	Bit data
INT WINAPI WriteDev pwData,WORD wCount	iceBitM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* t);
Function	8-bit data
INT WINAPI WriteDev pbData,WORD wCount	ice8M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,BYTE*);
Function	16-bit data
INT WINAPI WriteDev pwData,WORD wCount	ice16M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* t);
Function	32-bit data
INT WINAPI WriteDev pdwData,WORD wCour	ice32M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* nt);
Function	8-bit BCD data
INT WINAPI WriteDev pbData,WORD wCount	iceBCD8M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,BYTE*);
Function	16-bit BCD data
INT WINAPI WriteDev pwData,WORD wCount	iceBCD16M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* t);
Function	32-bit BCD data
INT WINAPI WriteDev pdwData,WORD wCour	iceBCD32M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* nt);
Function	Single-precision floating point data
INT WINAPI WriteDev pflData,WORD wCount	iceFloatM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,FLOAT*);
Function	Double-precision floating point data
INT WINAPI WriteDev pdbData,WORD wCour	iceDoubleM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,DOUBLE* at);
Function	Character string data
INT WINAPI WriteDev psData,WORD wCount	iceStrM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,LPCSTR);
Function	General-use data
INT WINAPI WriteDev pData,WORD wCount,W	iceM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,LPVOID WORD wAppKind);
Function	General-use data (Variant-type)
	iceVariantM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR ANT pData,WORD wCount,WORD wAppKind);
Function	Group symbol
INT WINAPI WriteSym pWriteBufferData);	bolM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sSymbolName,LPVOID
Function	Group symbol (Variant-type)
INT WINAPI WriteSym sSymbolName,LPVARI	bolVariantM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR ANT pData);
	27.24

Function	TIME data							
INT WINAPI WriteDeviceTIMEM(HANDLE hProServer, LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* pdwData, WORD wCount);								
Function	DATE data							
INT WINAPI WriteDeviceDATEM(HANDLE hProServer, LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* pdwData, WORD wCount);								
Function	TIME_OF_DAY data							
INT WINAPI WriteDeviceTIME_OF_DAYM(HANDLE hProServer, LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* pdwData, WORD wCount);								
Function	DATE_AND_TIME data							
INT WINAPI WriteDeviceDATE_AND_TIMEM(HANDLE hProServer, LPCSTR sNodeName, LPCSTR sDeviceName, QWORD* pqwData, WORD wCount);								
* For each parameter, please refer to " Parameters of read/write functions".								

* You can convert binary values written to TIME, DATE, TIME_OF_DAY, and DATE_AND_TIME data from text

format. For information about text conversion, refer to "27.8 Binary Date and Time / Text Display Conversion".

Parameters of read/write functions

< Argument >

bsNodeName : Pointer to node name (character string)

Specify the entry node name or the IP address registered in 'Pro-Studio EX' directly.

Ex. 1) When specifying node name: "AGP"

Ex. 2) When specifying IP address directly: "192.9.201.1"

bsDeviceName : Pointer to the symbol (character string) subjected to Read/Write function

Specify the symbol name or the device address registered in 'Pro-Studio EX' directly.

Ex. 1) When specifying symbol name: "SWITCH1"

Ex. 2) When specifying device address directly: "M100"

		Symbol data type												
Function	Bit	8 bits		16 bits		32 bits							TIME	DATE_
		S/U/ HEX	BCD	S/U/ HEX	BCD	S/U/ HEX	BCD	Float	Double	String	TIME	DATE	_OF_ DAY	AND_ TIME
XXXDeviceBit	0	-	-	-	-	-	-	-	-	-	-	-	-	-
XXXDevice8	-	0	-	-	-	-	-	-	-	-	-	-	-	-
XXXDevice16	-	-	-	0	-	-	-	-	-	-	-	-	-	-
XXXDevice32	-	-	-	-	-	0	-	-	-	-	-	-	-	-
XXXDeviceBCD8	-	-	0	-	-	-	-	-	-	-	-	-	-	-
XXXDeviceBCD16	-	-	-	-	0	-	-	-	-	-	-	-	-	-
XXXDeviceBCD32	-	-	-	-	-	-	0	-	-	-	-	-	-	-
XXXDeviceFloat	-	-	-	-	-	-	-	0	-	-	-	-	-	-
XXXDeviceDouble	-	-	-	-	-	-	-	-	0	-	-	-	-	-
XXXDeviceStr	-	-	-	-	-	-	-	-	-	0	-	-	-	-
XXXDevice	0	0	0	0	0	0	0	0	0	0	0	0	0	0
XXXDeviceTIME	-	-	-	-	-	-	-	-	-	-	0	-	-	-
XXXDeviceDATE	-	-	-	-	-	-	-	-	-	-	-	0	-	-
XXXDeviceTIME_OF _DAY	-	-	-	-	-	-	-	-	-	-	-	-	0	-
XXXDeviceDATE_A ND_TIME	-	-	-	-	-	-	-	-	-	-	-	-	-	0

pxxData

: Pointer to read/write target data

Accessible data types and corresponding argument types are listed below.

Accessible data type	Argument type
Bit data	WORD * pwData
8-bit data	BYTE * pbData
16-bit data	WORD * pwData
32-bit data	DWORD * pdwData
8-bit BCD data	BYTE * pbData
16-bit BCD data	WORD * pwData
32-bit BCD data	DWORD * pdwData
Single-precision floating point data	FLOAT * pflData
Double-precision floating point data	DOUBLE * pdbData
Character string data	LPTSTR psData
General-use data	LPVOID pData
General-use data (for VB)	LPVARIANT pData
TIME data	DWORD * pdwData
DATE data	DWORD * pdwData
TIME_OF_DAY data	DWORD * pdwData
DATE_AND_TIME data	QWORD * pdwData

wCount

: Quantity of read/write target data

With the Read/WriteDeviceStr function, character string data is counted as the number of bytes. For a device symbol with 16-bit width, specify multiples of two characters; for a device symbol with 32-bit width, specify multiples of four characters.

The maximum data quantities subjected to read/write functions are as follows:

Accessible data type	Read	Write
Bit data	255	255
8-bit data	1020	1020
16-bit data	1020	1020
32-bit data	510	510
8-bit BCD data	1020	1020
16-bit BCD data	1020	1020
32-bit BCD data	510	510
Single-precision floating point data	510	510
Double-precision floating point data	255	255
Character string data	2040 characters (single-byte)	2040 characters (single-byte)
TIME data	510	510
DATE data	510	510
TIME_OF_DAY data	510	510
DATE_AND_TIME data	255	255

wAppKind : Data type specification

Value	Data type	Value	Data type
1	Bit	11	Double
2	Signed 16 bits	12	String
3	Unsigned 16 bits	13	Signed 8 bit
4	HEX 16 bits	14	Unsigned 8 bit
5	BCD 16 bits	15	HEX 8 bit
6	Signed 32 bits	16	BCD 8 bit
7	Unsigned 32 bits	17	TIME
8	HEX 32 bits	18	DATE
9	BCD 32 bits	19	TIME_OF_DAY
10	Float	20	DATE_AND_TIME (*)

* Unable to use with VB functions.

With the Read/Write Device function, the data type is specified by parameter. Therefore, the data type can be dynamically changed.

< Return value >

Normal end: 0

Abnormal end: Error code

< Special Note >

When using the Read/WriteDeviceBit function:

pwData stores a quantity of data specified with wCount, consecutively from the D0 bit.

Example: When wCount is "20"

	F	Е	D	С	В	А	9	8	7	6	5	4	3	2	1	0
PwData	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
PwData+1	*	*	*	*	*	*	*	*	*	*	*	*	20	19	18	17

When reading/writing multiple consecutive bit data, it is more efficient to use Read/Write/Device 8, 16, and 32 functions than Read/WriteDeviceBit functions.

The bit indicated with "*" (asterisk) stores an undefined value. Mask these areas in your application program.

When using Read/WriteDeviceBCD8, Read/WriteDeviceBCD16 or Read/WriteDeviceBCD32 functions: If the target device/PLC handles BCD data, you can use these functions. However, the data passed with these functions (contents of pxxData) are handled as binary data, not BCD data. ('Pro-Server EX' internally executes BCD conversion.) A negative value cannot be handled.

Function	Decimal expression	Hexadecimal expression
Read/WriteDeviceBCD8	0 to 99	00 to 63
Read/WriteDeviceBCD16	0 to 9999	0000 to 270F
Read/WriteDeviceBCD32	0 to 99999999	00000000 to 05F5E0FF

When using the string data functions:

To receive character string data for variables, secure sufficient data storing area.

27.3 Cache Buffer Control APIs

Function	Creating cache buffer						
To increase the device read processing speed, 'Pro-Server EX' incorporates the device data caching function (with copy function). This API is used to create a cache buffer. This API only defines a cache buffer. To define which device to cache, use PS_EntryCacheRecord(). Single INT WINAPI PS_CreateCache(LPCSTR sCacheName, DWORD dwPollingTime); Multi							
	acheM(HANDLE hProServer, LPCSTR sCacheName, DWOR	C					
ArgumentReturn valuesCacheName: (In) Cache buffer nameNormal end: 0dwPollingTime: (In) To select the constant monitoring method, specify "0".Abnormal end: Error constantThe cache buffer is updated as fast as possible.Abnormal end: Error constantIf you specify any value other than "0", the polling method is selected.Specify the polling cycle (cache updating cycle) by the millisecond.							
 Special Note Up to 1000 cache buffers can be created for a single 'Pro-Server EX' program. You can directly use the cache buffer which has been registered when creating a network project file with 'Pro-Studio EX'. It is unnecessary to re-create it with this API. 							
Function	Registering record into cache buffer						
Registers a caching device (cache source device) into the cache buffer created with PS_CreateCache(). For a GP Series node or Pro-Server EX node, 'Pro-Server EX' does not support the constant monitoring method to update a cache buffer. Therefore, if you specify a GP Series node or Pro-Server EX node with this API for a cache buffer subjected to the constant monitoring method (if dwPollingTime is set to "0" when a cache buffer is created with PS_CreateCache()), an error occurs.							
wAppKind, WORD wCoun Multi INT WINAPI PS_EntryCa	cheRecord(LPCSTR sCacheName, LPCSTR sNodeName, LPC nt); cheRecordM(HANDLE hProServer, LPCSTR sCacheName, L wAppKind, WORD wCount);						

rgumen CacheNa	me: (In) Cache buffer name			Return value
	Register a cache source devie	Normal end: 0 Abnormal end: Error code		
NodeNar	name. me: (In) Entry node name with ca	che sourc	e Device/PLC name	Abhormar chu. Error cou
	In) Cache source device	ene soure		
			irectly specify the device address, or	
	specify a symbol or group registe group, multiple symbols can be re			
	d: (In) Source device data type	gistered a	a once.	
	Available data types vary dependi	ng on the	cache source device designation	
	method. a) When device address of cache	source de	vice is directly specified.	
	Specify a data type (1 to 20) avail			
5	specified.			
Value	Data type	Value	Data type	
1	Bit	11	Double-precision floating point	
2	16 bits, Signed decimal	12	Character string	
3	16 bits, Unsigned decimal	13	8 bit (Signed) data	
4	16 bits, Hexadecimal	14	8 bit (Unsigned) data	
5	16 bits, BCD	15	8 bit (HEX) data	
6	32 bits, Signed decimal	16	8 bit (BCD) data	
7	32 bits, Unsigned decimal	17	TIME data	
8	32 bits, Hexadecimal	18	TIME_OF_DAY data	
9	32 bits, BCD	19	DATE data	
10	Single-precision floating point	20	DATE_AND_TIME data	
t vCount: (1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	the symbol type specified in symbol c) When group is specified for car Fixed to "0". The symbol type is registered for (In) Device data quantity subjecte Available values vary depending on method. a) When device address of cache Data quantity (1 to 2040) accordin maximum value varies depending b) When symbol is specified for c If you specify "0", the quantity sp If you specify any value other tha	able with bol definit che source all symbol d to cachi on the cac source de ng to the de ache sour ecified in n 0, data d	Pro-Server EX'. If you specify "0", ion is used. e device: als in the specified group. ng the source device specification vice is directly specified: levice type can be used. (The evice type.) re device:	

Function	Starting caching	
Starts caching.		
Multi	ache(LPCSTR sCacheName); acheM(HANDLE hProServer, LPCSTR sCacheName);	
Argument sCacheName: (In) Name		Return value Normal end: 0 Abnormal end: Error code
Special Note		
Function	Stopping caching	
To restart caching, call P	tion of the cache buffer is retained.	
Multi	ache(LPCSTR sCacheName); acheM(HANDLE hProServer, LPCSTR sCacheName);	
Argument sCacheName: (In) Name A cache buffer	of cache buffer to stop name registered with 'Pro-Studio EX' can be also specified.	Return value Normal end: 0 Abnormal end: Error code
Special Note		
Function	Checking caching status	
Checks caching status.		
Single INT WINAPI PS_GetCa Multi	cheStatus(LPCSTR sCacheName);	
INT WINAPI PS_GetCa	cheStatusM(HANDLE hProServer, LPCSTR sCacheName);	Γ
Argument	of cache buffer to be checked name registered with 'Pro-Studio EX' can be also specified.	 Return value 0 : The cache buffer has been created, but not started yet. 1: Caching in progress
	name registered with 110-studio EX can be also specified.	2: Caching under suspension XX: Error code

Function	Discarding cache buffer					
Stops caching, and disca	Stops caching, and discards the cache buffer.					
Multi	oyCache(LPCSTR sCacheName); oyCacheM(HANDLE hProServer, LPCSTR sCacheName);					
Argument Return value sCacheName: (In) Name of cache buffer to be discarded Normal end: 0 A cache buffer name registered with 'Pro-Studio EX' can be also specified. Abnormal end: Error cod						
Special Note		<u> </u>				
Function	Setting cache update notification function					
Sets the function to noti	fy cache buffer update status to a specified window.					
device is frequently cach 'Pro-Server EX' can send has a change with the co- lf your system is built so be improved. This API allows you to a message" in 'Pro-Server After these settings are r Single INT WINAPI PS_SetNo LPARAM LParam, HAI Multi INT WINAPI PS_SetNo message, WPARAM WI	d a message to a specified window, when cache data is updated (workstant monitoring method, or when one polling cycle is complete to as to execute cache-reading of a device after receiving this message set "Target cache buffer name", "Window to receive the message"	when at least one target device d with the polling method). age, the system efficiency can , and "Contents of the ently-set notification function. essage, WPARAM WParam,				
Argument sCacheName: (In) Cache buffer name Return value Normal end: 0 A cache buffer name registered with 'Pro-Studio EX' can be also specified. Normal end: 0 Abnormal end: Error cod hWnd: (In) Handle for the window to receive the message message: (In) Message ID to be sent to the window WParam: (In) WPARAM value to be sent to the window together with message ID Image: Comparison of the window together Image: Comparison of the window toget						
After the cache buffer is wParam value (specified window (hWnd).	not necessary, discard it with PS_KillNotifyFromCache(). updated, call PostMessage() to send the message (specified with d with the third argument), and LParam value (specified with the f age(), refer to the Windows API Manual.					

Function	Accepting next cache update notification				
Accepts the next cache up	date notification.				
However, once this notific again, even if the cache bu notification routine, a mult update message. (If the no call error occurs with the r To prevent this error, this a By calling this API at the o	he function to send a message to a specified window when ation function is executed, 'Pro-Server EX' will not send a ffer is updated next. This is because in case it has taken a tiple-call error can occur with the relevant routine when 'Pr tification routine receives the next message before complet outine.) API explicitly informs 'Pro-Server EX' that it can send the end of the processing of the notification routine, you can b ry time a cache buffer is updated.	a message until this API is called long time in processing with the ro-Server EX' sends the next cache etion of the processing, a multiple- next message.			
Multi	NextNotifyFromCache(HANDLE hCacheNotifyID); NextNotifyFromCacheM(HANDLE hProServer, HANDLI	E hCacheNotifyID);			
Argument Return value hCacheNotifyID: (In) ID of next message acceptance notification function Normal end: 0 ID obtained with PS_SetNotifyFromCache() Abnormal end: Error co					
Special Note					
Function	Canceling cache update notification				
Cancels the function for se	ending a cache buffer update message to a specified windo	DW.			
	rver EX' will not send a cache buffer update message to th CacheNotifyID is updated.	ne relevant window, even if the			
Multi	ifyFromCache(HANDLE hCacheNotifyID); ifyFromCacheM(HANDLE hProServer, HANDLE hCach	eNotifyID);			
ArgumentReturn valuehCacheNotifyID: (In) ID of the notification function to be canceledNormal end: 0ID obtained with PS_SetNotifyFromCache()Abnormal end: Error code					
Therefore, if 'Pro-Server E window before this API is	d discard a message sent from 'Pro-Server EX', even if the X' has sent a message to a window and the application ha called, the application can fetch the message from the window and the application can fetch the message from the window and the application can fetch the message for the window application can fetch the window application can fetch the message for the window application can fetch the message for the window application can fetch the message for the window application can fetch the window application can fetch	s not fetched the message from the ndow even after this API is called.			

(Depending on the timing, the notification routine may be called even after this API is called.)

Function	Acquiring cache buffer update count				
Returns a cache buffer upo	late count.				
By monitoring the update count on the program, you can check if a cache buffer has been updated or not. Using this function, you can omit unnecessary calls of device cache read APIs. (Even if a device cache read API is called for a device with no change, the value will not be changed.)					
Single INT WINAPI PS_GetUpdateCounter(LPCSTR sCacheName, DWORD* odwCount); Multi INT WINAPI PS_GetUpdateCounterM(HANDLE hProServer, LPCSTR sCacheName, DWORD* odwCount);					
Argument Return value sCacheName: (In) Name of cache buffer to be monitored Normal end: 0 A cache buffer name registered with 'Pro-Studio EX' can be also specified. Abnormal end: Error cod odwCount: (Out) Cache buffer update count Abnormal end: Error cod Counts the number of updates from 0 to 4294967295 endlessly. After the count reaches 4294967295, it returns to"0".)					

27.4 Queuing Access Control APIs

Function	Starting the queuing of device read request				
After this API is called, 'Pro-Server EX' queues device read requests until ExecuteQueuingAccess() is called. Queuing is executed for each Pro-Server handle.					
Single INT WINAPI BeginQueuin Multi	ngRead();				
	ngReadM(HANDLE hProServer);				
Argument		Return value Normal end: 0 Abnormal end: Error code			
 Special Note Do not call a Device Write API until you call ExecuteQueuingAccess() after BeginQueuingRead(). After BeginQueuingRead() is called, 'Pro-Server EX' queues cache read or direct read requests. However, cache read and direct read requests cannot be queued together. To discard a request in queue, call CancelQueuingAccess(). Queuing is available up to 1500 requests and a data size of 1 Mbyte. 					
Function	Starting the queuing of device write request				
After this API is called, 'Pr Queuing is executed for each	o-Server EX' queues device write requests until ExecuteQueuir ch Pro-Server handle.	ngAccess() is called.			
Single INT WINAPI BeginQueuin Multi	-				
	ngWriteM(HANDLE hProServer);	1			
Argument		Return value Normal end: 0 Abnormal end: Error code			
 Special Note Do not call a Device Read API until you call ExecuteQueuingAccess() after BeginQueuingWrite(). After BeginQueuingWrite() is called, 'Pro-Server EX' queues cache write or direct write requests. However, cache write and direct write requests cannot be queued together. To discard a request in queue, call CancelQueuingAccess(). Queuing is available up to 1500 requests and a data size of 1 Mbyte. 					
Special Note					

Function	Executing device read/write request in queue	
Accesses device data acc	ording to the device read/write request in queue.	
Single INT WINAPI ExecuteQu Multi INT WINAPI ExecuteQu	euingAccess(); euingAccessM(HANDLE hProServer);	
Argument		Return value Normal end: 0 Abnormal end: Error code
'Pro-Server EX' fails to If you wish to know wh IsQueuingAcceessSucc	essfully accesses all specified devices, ExecuteQueu access any device, on the other hand, ExecuteQueu hether each device access request has been successfu ceeded() to check the result. CTIONs in queuing access.	ingAccess() returns an access error code.
Function	Discarding device read/write request in queue	
Discards the device read/	write request in queue.	
Single INT WINAPI CancelQue Multi INT WINAPI CancelQue	uingAccess(); uingAccessM(HANDLE hProServer);	
Argument		Return value Normal end: 0 Abnormal end: Error code
ExecuteQueuingAccess()	e() or BeginQueuingRead()is called, 'Pro-Server EX is called. omes unnecessary for any reason, call this API. 'Pro-	

Function	Checking the run result of device read/write request in queue			
called. Single INT WINAPI IsQueuingAd	h device access request has been successfully executed, after F	ExecuteQueuingAccess() is		
Multi INT WINAPI IsQueuingAo	ccessSucceededM(HANDLE hProServer,INT iIndex);			
Argument iIndex: (In) Number of req	uest to be checked	Return value XX: Error code 0: Indicates that the		
called several times to quet called. Note that you canno ExecuteQueuingAccess(). If you wish to know a result) or BeginQueuingRead() is called, Device Access APIs are the device access requests until ExecuteQueuingAccess() is of know an actual device access result until execution of at of each device access request, execute irst, and then specify the number (from 0) of the request for	device access request of the specified number has been successfully executed.		
WriteDevice16("No	ode1","LS100",Data,10); ode1","LS200",Data,10); ode1","LS300",Data,10);			
	cess to "LS200" has been successfully executed, use IsQueuir is access has been successfully executed.	ngAccessSucceeded(1).		

27.5 System APIs

Function	Creating Pro-Server handle			
Obtains a Pro-Server handl	e for use of a Multi-Handle function.			
HANDLE WINAPI Create	ProServerHandle();			
Argument		Return value Normal end: Other than 0 (Handle code) Abnormal end: 0		
Special Note				
Function	Releasing Pro-Server handle			
Releases an obtained Pro-S	erver handle.			
INT WINAPI DeleteProSe	rverHandle(HANDLE hProServer);			
Argument hProServer: (In) Pro-Serve	r handle to be released	Return value Normal end: 0 Abnormal end: Error code		
Special Note				
Function	Loading network project file			
Loads the network project	file specified with the argument.			
Multi	etworkProject(LPCSTR sDBName,DWORD dwSetOrAdd = Tl etworkProjectM(HANDLE hProServer,LPCSTR sDBName,DV			
Argument sDBName: Specify the full dwSetOrAdd: Reserve (Fix hProServer: Pro-Server har	,	Return value Normal end: 0 Abnormal end: Error code		
Special Note				

Function	Converting error code into character string	
EasyLoadErrorMessage() r	rned by each API of 'Pro-Server EX' into an error message. eturns a multibyte character string (ASCII) as a message. Easy ing (UNICODE) as a message.	LoadErrorMessageW()
	ErrorMessage(INT iErrorCode,LPSTR osErrorMessage); ErrorMessageW(INT iErrorCode,LPWSTR owsErrorMessage)	;;
osErrorMessage: (Out) Poi string) storing area. (osErrorMessage: (Out) Poi	e returned by 'Pro-Server EX' function nter to the converted character string (multibyte character To call this API, secure a storing area with at least 512 bytes.) nter to the converted character string (multibyte character To call this API, secure a storing area with at least 1024	Return value Normal end: Other than 0 Failure in character string conversion (ex. Undefined code): 0
	ensure compatibility with older versions of 'Pro-Server'. sssageEx() enables conversion into a more detailed error messag Ex().	e. We recommend you to use
Function	Converting error code into character string (with status inform	nation)
possible. EasyLoadErrorMessage() a EasyLoadErrorMessageExt error occurrence place and return a different error mes EasyLoadErrorMessageExt message.	s the error message together with the error occurrence condition always returns the same error message relative to a specified error () returns more detailed error information including a name of c so on, depending on the error occurrence condition. Thus, Easy sage relative to the same error code, depending on the situation () and EasyLoadErrorMessageExM() return a multibyte charact W() and EasyLoadErrorMessageExWM() return a wide charact	for code. On the other hand, communication target device, LoadErrorMessageEx() may ter string (ASCII) as a
BOOL WINAPI EasyLoad Multi BOOL WINAPI EasyLoad	ErrorMessageEx(INT iErrorCode,LPSTR osErrorMessage); ErrorMessageExW(INT iErrorCode,LPWSTR owsErrorMessa ErrorMessageExM(HANDLE hProServer,INT iErrorCode,LPS ErrorMessageExWM(HANDLE hProServer,INT iErrorCode,L	TR osErrorMessage);
osErrorMessage: (Out) Poi string) storing area.(owsErrorMessage: (Out) Po	e returned by 'Pro-Server EX' function nter to the converted character string (multibyte character To call this API, secure a storing area with at least 1024 bytes.) pointer to the converted character string (wide character string) this API, secure a storing area with at least 2048 bytes.)	Return value Normal end: Other than 0 Failure in character string conversion (ex. Undefined code): 0
 Server EX' is called and 'Pro-Server EX' can store between the API that can status information becau) is used to convert an error code into a message, assuming a cathen the API returns an error code. e only one piece of error status information per handle. Therefores an error and EasyLoadErrorMessage(),EasyLoadErrorMessage stored error status information is rewritten. For this reason, we M(), you must specify the same Pro-Server handle as the handle	re, if you call another API sage() will not return error when using

Function	Initializing Pro-Server API	
	API, and declares use of the API internally. ithout starting 'Pro-Server EX', 'Pro-Server EX' will automatica	ally start.
INT WINAPI EasyInit();		
Argument		Return value Normal end: 0 Abnormal end: Error code
Special Note		
Function	Ending Pro-Server API	
INT WINAPI EasyTerm();	L	
Argument		Return value
	ure compatibility with older versions of 'Pro-Server'. need not call this API. (Even if you call this API, it will not be	executed.)
Function	Closing Pro-Server EX	
	ot call any API of 'Pro-Server EX'. sure to discard Pro-Server handles etc. rver();	
Argument		Return value Normal end: 0 Abnormal end: Error code
Special Note		1

Function	Pro-Server EX closing notice	
When 'Pro-Server EX' start using PostMessage() of Wi For details of PostMessage		
LPARAM LParam = 0); Multi INT WINAPI EasyNotifyF	romServerEnd(HWND hReceivedWnd,UINT uMessage,WPA romServerEndM(HANDLE hProServer,HWND hReceivedWn ram = 0, LPARAM LParam = 0);	
Argument		Return value
uMessage: (In) Message IE This ID will be sent t EX is being closed. WParam: (In) WPARAM to PostMessage())	we that receives a closing message. to be sent as a closing message. to the window specified with hReceivedWnd when Pro-Server to be sent together with the message (Value of WPARAM in the sent together with the message (Value of LPARAM in	Normal end: 0 Abnormal end: Error code
For example, if you specify API, 'Pro-Server EX' sends Generally, an application us closes at the same time who	an application that closes at the same time when 'Pro-Server E's the application main window for hReceivedWnd, and WM_Q' WM_QUIT to the application main window when 'Pro-Server ses WM_QUIT as an application closing signal. Therefore, you en 'Pro-Server EX' is closed.	UIT for uMessage to call this EX' is being closed.
Function	Inhibiting message processing	
processing time would be I When Windows message p will not be processed durin As a result, you will not ca In this case, however, the p and the processing of impo You can specify whether to	APIs (functions) process Windows messages during the process ong. This API can specify whether to execute or inhibit the Wi rocessing is inhibited, the relevant Windows message is stored g execution of a function. Il a function over again by clicking the icon during execution of rocessing of all the Windows messages as well as an "icon click rtant messages for timer and window re-drawing is also disable execute or inhibit the processing of Windows messages for ea- essage processing has been set to "Execute".	ndows message processing. in the message queue, and f the function. c'' message, will be inhibited, ed.
Single INT EasySetWaitType(DW Multi INT EasySetWaitTypeM(H	'ORD dwMode); ANDLE hProServer,DWORD dwMode);	
dwMode: (In) To execute n	-Server handle subjected to processing mode change nessage processing, specify "1". rocessing, specify "2".	Return value Normal end: 0 Abnormal end: Error code
Special Note		

Function	Acquiring message processing mode				
	ge processing mode during a call of a Pro-Server EX A urns the current message processing mode for each hand				
Single INT EasyGetWaitType(); Multi INT EasyGetWaitTypeM(F	IANDLE hProServerHandle);				
Argument HANDLE hProServerHand	lle: (In) Handle subjected to status acquisition	Return value 1: Executes message			
		processing. 2: Inhibits message processing.			

Function Ac	lding log into log v	iewer	
f a specific event ('Pro-Server he event.	EX' start/closing, e	error, etc.) occurs with internal processin	g, 'Pro-Server EX' can record
	mation through the	log viewer. (See "28.5 Monitoring Syste	em Event Logs")
	records a specific	message by using this function. This AP	I is available for application
ebugging.			
NT WINAPI EasyOutputLog	BYTE bLevel,LPC	CSTR sPrompt,LPCSTR sMessage);	
Argument			Return value
Level: (In) Event type			Normal end: 0
		rmance deterioration. To prevent this, n for recording messages by event type.	Abnormal end: Error code
		rding message belongs to.	
The event types are liste	d below.		
	Hexad		
Definition	ecimal	Event type	
	value		
EASY_LogLevel_SysMes	sage 0x01	System message	
EASY_LogLevel_SysErro	r 0x02	System error message	
EASY_LogLevel_AppErro	or 0x04	User program error message	
EASY_LogLevel_AppStar	t 0x08	User program starting message	
EASY_LogLevel_AppEnd	0x10	User program closing message	
EASY_LogLevel_AppWar	ming 0x20	User program warning message	
EASY_LogLevel_AppMes	sage1 0x40	User program detail message 1	
EASY_LogLevel_AppMes	sage2 0x80	User program detail message 2	
		currence position (NULL-terminated) be recorded (NULL-terminated)	
The actually recorded message nd sMessage).	is a simple combine	nation of two character stings (sPrompt	
Special Note			1
-			

Function	Clearing log from log viewer			
Clears the information reco This API is available for ap INT WINAPI EasyOutputL	plication debugging.			
Argument HANDLE hProServerHand	lle: (In) Handle subjected to status acquisition	Return value Normal end: 0 Abnormal end: Error code		
Special Note				

27.6 SRAM Data Access APIs

Function

Argument

Reading SRAM backup data

Reads the following data stored in the SRAM of display unit, and saves the data into a file on the PC. Filing data are saved in binary format, and other types of data are saved in CSV format.

INT WINAPI EasyBackupDataRead(LPCSTR sSaveFileName,LPCSTR sNodeName,INT iBackupDataType,INT iSaveMode);

Return value Normal end: 0

Abnormal end: Error code

sSaveFileName: (In) File path of the file to save read data. (String pointer) sNodeName: (In) Name of read data source node (String pointer) Pro-Server EX nodes cannot be specified.

iBackupDataType: (In) Type of data to be read

Value	Data source node	Data source node	
	in GP Series	other than GP Series	
0x0001	Filing data	Filing data	
0x0002	Logging data	Sampling data of sampling group No. 1	
0x0003	Line graph data	Data of all sampling groups other than	
0x0004	Sampling data	sampling group No. 1	
0x0005	Alarm block 1	Alarm block 1	
0x0006	Alarm history or Alarm block 2	Alarm block 2	
0x0007	Alarm log or Alarm block 3	Alarm block 3	
0x0008	Alarm block 4	Alarm block 4	
0x0009	Alarm block 5	Alarm block 5	
0x000A	Alarm block 6	Alarm block 6	
0x000B	Alarm block 7	Alarm block 7	
0x000C	Alarm block 8	Alarm block 8	
Others	(Reserve)	(Reserve)	

When the data source node is in the SP-5B40/WinGP, SP-5B10, GP4000/LT4000 Series, GP3000 Series, LT3000, and the data type is Alarm block 1 to 8, one alarm block stores up to three types of data (active data, history data and log data) depending on the settings of 'GP-Pro EX'. However, this API checks if the alarm block contains valid data or not according to the following order of precedence, and reads valid data if any.
(1) Alarm history
(2) Alarm log
(3) Alarm active
If there is no valid data, an error occurs.
iSaveMode: (In) Saving mode

0: New (If a file with the same name already exists, 'Pro-Server EX' deletes the file, and overwrites it.)
1: Add (The read data is added to the end of an existing file. If there is no file to

1: Add (The read data is added to the end of an existing file. If there is no file to save the data, 'Pro-Server EX' creates a new file.) Others: Reserve

- When reading Alarm or Sampling data, the date format is "yy/mm/dd".
- If [Multiple Line Message Output (Save Alarm to CSV)] is enabled in GP-Pro EX alarm settings, messages with line breaks are output to a single cell. If [Multiple Line Message Output (Save Alarm to CSV)] is disabled, the message up to the line break only is saved.

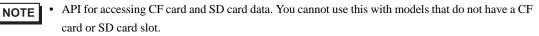
Fu	inction	Reading exte	ended SRAM backup data	
ïling data a Inlike Easy	are saved in binar	y format, and (), this API er	M of display unit, and saves the data into a file of other types of data are saved in CSV format. hables access to extended data for the SP-5B40/V	
			CSTR sSaveFileName, LPCSTR sNodeName, I gTable = 0x0000);	NT iBackupDataType, IN
sNode Pro-S		e of read data annot be speci		Return value Normal end: 0 Abnormal end: Error code
Value	Data source	ce node	Data source node	
Value	in GP S	eries	other than GP Series	
0x0001	Filing data		Filing data]
0x0002	Logging data		Sampling data of sampling group No. 1	
0x0003	Line graph data		- Data of an sampling groups other than	
0x0004	Sampling data			
0x0005 Alarm block 1			Alarm block 1	
			Specify iNumber for alarm type.	
0x0006	Alarm history of block 2	r Alarm	Alarm block 2 Specify iNumber for alarm type.	
0x0007	Alarm log or Al	arm block 3	Alarm block 3 Specify iNumber for alarm type.	
0x0008	Alarm block 4		Alarm block 4 Specify iNumber for alarm type.	
0x0009	Alarm block 5		Alarm block 5 Specify iNumber for alarm type.	
0x000A	Alarm block 6		Alarm block 6 Specify iNumber for alarm type.	
0x000B	Alarm block 7		Alarm block 7 Specify iNumber for alarm type.]
0x000C	Alarm block 8		Alarm block 8 Specify iNumber for alarm type.	
			Sampling group of a specific group number	1

and overwrites it.) 1: Add (The read data save the data, 'Pro-Se Others: Reserve nber: (In) This argumen	a is rve nt is	added to the r EX' creates s ignored wh	already exists, 'Pro-Server EX' deletes the file e end of an existing file. If there is no file to s a new file.)) nen sSaveFileName specifies a GP Series file ment varies depending on the value of	
Value of iBackupDataType			Description]
			of alarm data (active, history and log) are ecify a target alarm type.	
		Value of	Description	
		iNumber	Decomption	
0x0005 to 0x000C		0	 'Pro-Server EX' checks if the alarm block contains valid data or not according to the following order of precedence, and reads valid data if any. (1) Alarm history (2) Alarm log (3) Alarm active If there is no valid data, an error occurs. 	
		1	Reads alarm active data.	
		2	Reads alarm history data.	
		3	Reads alarm log data.	
		If the target data type does not exist in the alarm block specified withiBackupDataType, an error occurs.		
0x8002		Group numbe Any value fro	er of sampling group to be read om 1 to 64	
Others	(Reserve)	11	

- When reading Alarm or Sampling data, the date format is "yy/mm/dd".
- If [Multiple Line Message Output (Save Alarm to CSV)] is enabled in GP-Pro EX alarm settings, messages with line breaks are output to a single cell. If [Multiple Line Message Output (Save Alarm to CSV)] is disabled, the message up to the line break only is saved.

Function	Writing SRAM backup data	
Writes specified filing data	in binary format into the SRAM of a GP Series node.	
INT WINAPI EasyBackup	DataWrite(LPCSTR sSourceFileName,LPCSTR sNodeNam	e,INT iBackupDataType);
pointer) sNodeName: (In) Name of You can specify GP S	path of binary-formatted filing data to be written (String entry node to write data (String pointer) Series nodes only. d to "1". ("1" indicates filing data.)	Return value Normal end: 0 Abnormal end: Error code
Special Note		

27.7 CF Card / SD Card APIs



- When using a model with a SD card slot, please read "CF" and "CF card" as "SD" and "SD card".
- You can use the CF card API functions to read from and write to a SD card.

Similarly, you can use the SD card API functions to read from and write to a CF card.

Function Reading CF card status	Reading CF card status		
Acquires connection status of the CF card in a connected the display unit.			
Single CF Card: INT WINAPI EasyIsCFCard(LPCSTR SD Card: INT WINAPI EasyIsSDCard(LPCSTR Multi CF Card: INT WINAPI EasyIsCFCardM(HAND SD Card: INT WINAPI EasyIsSDCardM(HAND	sNodeName); LE hProServer,LPC		
Argument	Return value		
hProServer: Pro-Server handle sNodeName: Name of node to read status (This node name must be pre-registered in a network project.)	Function return value	For GP Series node	Other than GP Series node
1 5 /	0x00000000	Normal	Normal
	0x10000001	No CF card	No CF card, or CF card slot cover is opened (regardless of presence/absence of CF card)
	0x10000002	Detection of device incompatible with CF card driver	
	0x10000004	Detection of CF card error	Detection of CF card error
	0x10000008	CF card not initialized	
	Others	Error without re	elation to CF card
	L	1	
Special Note			

Function	Reading file list from CF card (Optional folder name)	
Outputs a list of files from specify an optional file to s	the CF card inserted in a display unit node into a file specie ave the file list.	fied with the parameter. You can
sSaveFileName);	syGetListInCfCard(LPCSTR sNodeName, LPCSTR sDire	-
SD Card: INT WINAPI Ea sSaveFileName);	syGetListInSdCard(LPCSTR sNodeName, LPCSTR sDire	ctory, INT* oiCount, LPCSTR
Argument		Return value
sNodeName: Name of node	e to output file list	Normal end: 0
sDirectory: Name of folder to receive file list (All capitals)		Abnormal end: Error code
-	oiCount: Number of output files	
	ile to save output directory information. The specified file	
•	the alignment type specified with stEasyDirInfo, in the	
quantity specified wi	th the return value of oiCount.	
struct stEasyDirInfo	{	
BYTE bFileName[8+1];// File name (Terminated with "0")		
BYTE bExt[3+1];// File extension (Terminated with "0")		
BYTE bDummy[3];// Dummy		
DWORD dwFileSize;// File size		
BYTE bFileTimeStamp[8+1];// File timestamp (Terminated with "0")		
BYTE bDummy2[3];// Dummy 2		
};		

As a supplement of "bFileTimeStamp" (8 bytes), high-order four bytes indicate time in the MS-DOS format, and low-order four bytes indicate date in the MS-DOS format (hexadecimal string).

The MS-DOS time/date format is as follows:

(Example: 20C42C22 is expressed as 2002/1/2 4:6:8. "2C22" is hexadecimal notation of date, and "20C4" is hexadecimal notation of time.)

Bit	Description
0 to 4	Day (1 to 31)
5 to 8	Month (1 = January, 2 = February, 12 = December)
9 to 15	Year: Expressed with the number of elapsed years from 1980. The actual year is the sum of 1980 and a value of these bits.

Specify time in the MS-DOS format. Time is packed in 16 bits in the following format:

Bit	Description
0 to 4	Number of seconds divided by two (0 to 29)
5 to 10	Minute (0 to 59)
11 to 15	Hour (0 to 23, on 24-hour basis)

When reading the file list, file names shorter than 8 characters or file extensions shorter than 3 characters are displayed as bFileName[8+1] or bExt[3+1] respectively, as shown below.

Read Source Node	Other than GP series node	GP series node
bFileName[8+1]	When the file name is shorter than 8 characters, null $(0x00)$ is stored at the end of the original file name, and undefined values are stored after null.	When the file name is shorter than 8 characters, single-byte spaces (0x20) are stored after the original file name, with null (0x00) as the final character.
bExt[3+1]	When the file extension is shorter than 3 characters, null $(0x00)$ is stored at the end of the original file extension, and undefined values are stored after null.	When the file extension is shorter than 3 characters, single-byte spaces (0x20) are stored after the original file extension, with null (0x00) as the final character.

(Example) When ABC.D is the file name and file extension

Other than GP series node

bFileName[8+1]	0x410x420x430x00********************************	
bExt[3+1]	0x440x00******* (**** indicate an undefined value)	

GP series node

bFileName[8+1]	0x410x420x430x200x200x200x200x200x00
bExt[3+1]	0x440x200x200x00

Function	Reading file list from CF card (including the sub-folders or b name)	elow in an optional folder
specify an optional file to s	the CF card inserted in a display unit node into a file specified ave the file list. Optionally, you can define the folder with the ed by searching the folder passed by the parameter, including s	list of files you want to get.
LPCSTR sSaveFileName);	syGetListRecursivelyInSdCard(LPCSTR sNodeName, LPCST	•
oiCount: Number of output sSaveFileName: Name of f stores binary data of	to receive file list (All capitals)	Return value Normal end: 0 Abnormal end: Error code
BYTE bExt[3+1];// I BYTE bDummy[3];/ DWORD dwFileSize BYTE bFileTimeSta	 +1];// File name (Terminated with "0") File extension (Terminated with "0") / Dummy e;// File size mp[8+1];// File timestamp (Terminated with "0") [260+1];// Folder name (Terminated with "0", "0" is also portions. 	

If you select a GP Series node with this API, an error is generated.

As a supplement of "bFileTimeStamp" (8 bytes), high-order four bytes indicate time in the MS-DOS format, and low-order four bytes indicate date in the MS-DOS format (hexadecimal string).

The MS-DOS time/date format is as follows:

(Example: 20C42C22 is expressed as 2002/1/2 4:6:8. "2C22" is hexadecimal notation of date, and "20C4" is hexadecimal notation of time.)

Bit	Description
0 to 4	Day (1 to 31)
5 to 8	Month (1 = January, 2 = February, 12 = December)
9 to 15	Year: Expressed with the number of elapsed years from 1980. The actual year is the sum of 1980 and a value of these bits.

Specify time in the MS-DOS format. Time is packed in 16 bits in the following format:

Bit	Description
0 to 4	Number of seconds divided by two (0 to 29)
5 to 10	Minute (0 to 59)
11 to 15	Hour (0 to 23, on 24-hour basis)

When reading the file list, file names shorter than 8 characters or file extensions shorter than 3 characters are displayed as bFileName[8+1] or bExt[3+1] respectively, as shown below.

bFileName[8+1]	When the file name is shorter than 8 characters, null (0x00) is stored at the end of the original file name, and undefined values are stored after null.
bExt[3+1]	When the file extension is shorter than 3 characters, null (0x00) is stored at the end of the original file extension, and undefined values are stored after null.

(Example) When ABC.D is the file name and file extension

bExt[3+1] $0x440x00******** (**** indicate an undefined value)$	bFileName[8+1]	0x410x420x430x00********************************
	bExt[3+1] 0x440x00******* (**** indicate an undefined value)	

Function Reading file list from CF card (Type specification)			
	he CF card inserted in a display unit into a file specified with ith "sDirectory" can be output.	the parameter. Only the file list	
INT WINAPI EasyGetListl sSaveFileName);	nCard(LPCSTR sNodeName, LPCSTR sDirectory, INT* oi	Count, LPCSTR	
following directories: LOG (Logging data) TREND (Trend data) ALARM (Alarm data CAPTURE (Capture FILE (Filing data) oiCount: Number of output sSaveFileName: Name of f stores binary data of quantity specified wi struct stEasyDirInfo BYTE bFileName[8- BYTE bFileName[8- BYTE bDummy[3];// DWORD dwFileSize	y to output list (All capitals) This API supports only the a) data) files lie to save output directory information. The specified file the alignment type specified with stEasyDirInfo, in the th the return value of oiCount. { -1];// File name (Terminated with "0") Pile extension (Terminated with "0") / Dummy ;// File size mp[8+1];// File timestamp (Terminated with "0")	Return value Normal end: 0 Abnormal end: Error code	

When reading the file list, file names shorter than 8 characters or file extensions shorter than 3 characters are displayed as bFileName[8+1] or bExt[3+1] respectively, as shown below.

Read Source Node	Other than GP series node	GP series node
bFileName[8+1]	When the file name is shorter than 8 characters, null $(0x00)$ is stored at the end of the original file name, and undefined values are stored after null.	When the file name is shorter than 8 characters, single-byte spaces $(0x20)$ are stored after the original file name, with null $(0x00)$ as the final character.
bExt[3+1]	When the file extension is shorter than 3 characters, null $(0x00)$ is stored at the end of the original file extension, and undefined values are stored after null.	When the file extension is shorter than 3 characters, single-byte spaces (0x20) are stored after the original file extension, with null (0x00) as the final character.

(Example) When ABC.D is the file name and file extension

Other than GP series node

bFileName[8+1]	0x410x420x430x00********************************	
bExt[3+1]	1] 0x440x00******* (**** indicate an undefined value)	

GP series node

bFileName[8+1]	0x410x420x430x200x200x200x200x200x00
bExt[3+1]	0x440x200x200x00

1	Function	Rea	ading file list from CF card (Including sub-folders in Type	specification)	
in the dir		d with "	CF card inserted in a display unit into a file specified with th sDirectory" can be output. Get the list of files to read by sory".		
INT WIN sSaveFile	-	istRecu	rsivelyInCard(LPCSTR sNodeName, LPCSTR sDirectory	7, INT* oiCount, LPCSTR	
sDirector following LC TF AI CA FII oiCount:	ame: Name of n r: Name of dire g directories: DG (Logging da END (Trend d ARM (Alarm APTURE (Capt LE (Filing data Number of out	ctory to ata) data) ure data) put file	output list (All capitals) This API supports only the	Return value Normal end: 0 Abnormal end: Error code	
the	e quantity speci uct stEasyDirIr	fied wit	alignment type specified with stEasyRecursiveDirInfo, in the return value of oiCount.		
BY BY BY BY sto BY };	TE bExt[3+1] TE bDummy[YORD dwFiles TE bFileTime TE bFolderNa red in remainir TE bDummy2	;// File 6 3];// Du Size;// F Stamp[8 me[260 ng portio	File size 8+1];// File timestamp (Terminated with "0") 0+1];// Folder Name (Terminated with "0", "0" is also pons.)		
When rea	u set a GP Seri ading the file lis	st, file r	in this API, it will become an error. names shorter than 8 characters or file extensions shorter th 1] respectively, as shown below.	an 3 characters are displayed	
bFile	Name[8+1]		en the file name is shorter than 8 characters, null (0x00) is stored at the end of the inal file name, and undefined values are stored after null.		
b	bExt[3+1] When the file extension is shorter than 3 characters, null (0x00) is stored at the end of original file extension, and undefined values are stored after null.				
(Example	e) When ABC.	D is the	file name and file extension		
	bFileName	[8+1]	0x410x420x430x00********************************	icate an undefined value)	
	bExt[3+1]		0x440x00******** (**** indicate an undefined value)		

FunctionReading file from CF card (Optional file name specification)		
Reads a specified file from	the CF card. You can specify an optional file to read.	
LPCSTR pWriteFileName,	syFileReadInSdCard(LPCSTR sNodeName, LPCSTR sFolder	
Argument		Return value
sNodeName: Name of node	e to output file list	Normal end: 0
sFolderName: Name of fol single-byte character	der containing source file to be read from CF card (Up to 32 s)	Abnormal end: Error code
sFileName: Name of source string)	e file to be read from CF card (Up to 8.3 format character	
pWriteFileName : File name of read CF file (Full path)		
odwFileSize: Size of read C		
Special Note		

Function	Reading file from CF card (Type specification)
Reads a specified file from	the CF card. Only the file type specified with "pReadFileType" can be read.
INT WINAPI EasyFileRea sWriteFileName, DWORD	dCard(LPCSTR sNodeName, LPCSTR pReadFileType, WORD wReadFileNo, LPCSTR * odwFileSize);

Argument	Return value
sNodeName: Name of node to output file list	Normal end: 0
pReadFileType: Type of source file to be read from CF card (See <special note="">)</special>	Abnormal end: Error code
wReadFileNo: File number of source file to be read from CF card	
sWriteFileName : File name of read CF file (Full path)	
odwFileSize: Size of read CF file	

This API supports the following file types. Only the files saved in a specified CF card folder can be read.

■File types supported for GP Series node

Data type	File type	Target folder
Filing data	ZF	FILE
CSV data	ZR	FILE
Image screen	ZI	DATA
Sound data	ZO	DATA
Trend graph data	ZT	TREND
Sampling data	ZS	TREND
Alarm block 4 to 8	Z4 to Z8	ARAM
Logging data	ZL	LOG
Alarm Log	ZG	ALARM
Alarm History	ZH	ALARM
Alarm Active	ZA	ALARM
Screen data backup	ZC	MRM
GP Screen data (Jpeg)	СР	CAPTURE

Data type	File type	Target folder
Filing data	ZF or F	FILE
CSV data	ZR	FILE
mage screen	ZI or I	DATA
Sound data	ZO or O	DATA
Alarm block 1	Z1 or ZA	ALARM *1
Alarm block 2	Z2 or ZH	ALARM *1
Alarm block 3	Z3 or ZG	ALARM *1
Alarm block 4 to 8	Z4 to Z8	ALARM *1
Sampling group 1 to 64	ZS1 to ZS64	SAMP01 to SAMP64 *1
GP Screen data (Jpeg)	СР	CAPTURE
GP-PRO/PB Trend graph data (compatible)	ZT	TREND
GP-PRO/PB Sampling data (compatible)	ZS	TREND
GP-PRO/PB Logging data (compatible)	ZL	LOG

*1) When using GP-Pro EX's [Set number of files in destination folder on external storage] feature, reads the files in sub-folders (for example: "ALARM\00000"). However, if you are using a version of GP-Pro EX before V3.12, or a version of Pro-server EX before V1.32, reads only the files in the [ALARM] or [SAMP**] folder, regardless of this setting.

Function	Writing file into CF card (Optional file name specification)		
Writes a specified file into	the CF card. You can specify an optional file to write.		
sFolderName, LPCSTR sF	syFileWriteInSdCard(LPCSTR sNodeName, LPCSTR pRea		
Argument		Return value	
sNodeName: Name of nod	e to write file	Normal end: 0	
pReadFileName: Name of source file to be written into CF card (Full path)		Abnormal end: Error code	
sFolderName: Name of folder containing target file in CF card (Up to 32 single-byte characters)			
characters)			

Function	Writing file into CF card (Type specification)		
Writes a specified file into the CF card. Only the file type specified with "pWriteFileType" can be written.			
INT WINAPI EasyFileWriteCard(LPCSTR sNodeName, LPCSTR pReadFileName, LPCSTR sWriteFileType, WORE wWriteFileNo);			
Argument		Return value	
sNodeName: Name of node to write file		Normal end: 0	
pReadFileName: Name of source file to be written into CF card (Full path)		Abnormal end: Error code	
sWriteFileType: Type of target file in CF card			
(See <special note=""> of the function for "Reading file into CF card (Type specification)")</special>			
~r · · · · · · / /			

When using GP-Pro EX's [Set number of files in destination folder on external storage] feature, writes the files in subfolders (for example: "ALARM\00000"). However, if you are using a version of GP-Pro EX before V3.12, or a version of Pro-server EX before V1.32, writes only the files in the [ALARM] or [SAMP**] folder, regardless of this setting.

Function	Deleting file from CF card (Optional file)			
Deletes a specified file from	n the CF card. You can specify an optional file to delete.			
CF Card: INT WINAPI EasyFileDeleteInCfCard(LPCSTR sNodeName, LPCSTR sFolderName, LPCSTR sFileName); SD Card: INT WINAPI EasyFileDeleteInSdCard(LPCSTR sNodeName, LPCSTR sFolderName, LPCSTR sFileName);				
Argument sNodeName: Name of node containing file to be deleted sFolderName: Name of folder containing file to be deleted from CF card (Up to 32 single-byte characters) sFileName: Name of file to be deleted from CF card (Up to 8.3 format character string)		Return value Normal end: 0 Abnormal end: Error code		
Special Note				

Function	Deleting file from CF card (Type specification)				
Deletes a specified file from the CF card. Only the file type specified with "pDeleteFileType" can be deleted.					
INT WINAPI EasyFileDel	INT WINAPI EasyFileDeleteCard(LPCSTR sNodeName, LPCSTR pDeleteFileType, WORD wDeleteFileNo);				
Argument Return value					
sNodeName: Name of node containing file to be deleted Normal end: 0					
pDeleteFileType: Type of f wDeleteFileNo: File numb	Abnormal end: Error code				

Special Note

If this function is executed for a file that does not exist in the CF card, it is not judged as an error, and the processing ends normally.

This API supports the following file types. Only the files saved in a specified CF card folder can be delete.

■File types supported for GP Series node

Data type	File type	Target folder
Filing data	ZF	FILE
CSV data	ZR	FILE
Image screen	ZI	DATA
Sound data	ZO	DATA
Trend graph data	ZT	TREND
Sampling data	ZS	TREND
Alarm block 4 to 8	Z4 to Z8	ARAM
Logging data	ZL	LOG
Alarm Log	ZG	ALARM
Alarm History	ZH	ALARM
Alarm Active	ZA	ALARM
Screen data backup	ZC	MRM
GP screen data (Jpeg)	СР	CAPTURE

Data type	File type	Target folder
Filing data	ZF or F	FILE
SV data	ZR	FILE
mage screen	ZI or I	DATA
Sound data	ZO or O	DATA
Alarm block 1	Z1 or ZA	ALARM *1
Alarm block 2	Z2 or ZH	ALARM *1
Alarm block 3	Z3 or ZG	ALARM *1
Alarm block 4 to 8	Z4 to Z8	ALARM *1
Sampling group 1 to 64	ZS1 to ZS64	SAMP01 to SAMP64 *1
GP Screen data (Jpeg)	СР	CAPTURE
GP-PRO/PB Trend graph data (compatible)	ZT	TREND
GP-PRO/PB Sampling data (compatible)	ZS	TREND
GP-PRO/PB Logging data (compatible)	ZL	LOG

*1) When using GP-Pro EX's [Set number of files in destination folder on external storage] feature, delets the files in sub-folders (for example: "ALARM\00000"). However, if you are using a version of GP-Pro EX before V3.12, or a version of Pro-server EX before V1.32, delets only the files in the [ALARM] or [SAMP**] folder, regardless of this setting.

Function	Renaming file in CF card					
Renames a specified file in	Renames a specified file in the CF card.					
CF Card: INT WINAPI EasyFileRenameInCfCard(LPCSTR sNodeName, LPCSTR sFolderName, LPCSTR sFileName,LPCSTR sFileRename); SD Card: INT WINAPI EasyFileRenameInSdCard(LPCSTR sNodeName, LPCSTR sFolderName, LPCSTR sFileName,LPCSTR sFileRename);						
Argument		Defense and a				
Argument		Return value				
sNodeName: Name of node	e to write file	Normal end: 0				
sNodeName: Name of node	e to write file der containing file to be renamed in CF card (Up to 32 single-					
sNodeName: Name of node sFolderName: Name of fold byte characters)		Normal end: 0				
sNodeName: Name of node sFolderName: Name of fold byte characters) sFileName: Name to file to	der containing file to be renamed in CF card (Up to 32 single-	Normal end: 0				

Function	Acquiring information on CF card empty space				
Acquires information on empty space in the CF card connected to a specified entry node.					
CF Card: INT WINAPI Ea pioUnallocatedH); SD Card: INT WINAPI Ea	asyGetCfFreeSpace(LPCSTR sNodeName,INT* o asyGetCfFreeSpaceEx(LPCTSTR sNodeName,IN asyGetSdFreeSpace(LPCSTR sNodeName,INT* o asyGetSdFreeSpaceEx(LPCTSTR sNodeName,IN	T* pioUnallocatedL,INT* viUnallocated);			
	y space in CF card (number of bytes) npty space in bottom 4 bytes	Return value Normal end: 0 Abnormal end: Error code			

Special Note

*1 When the free space exceeds the range for INT, use the CF card (expansion) or SD card (expansion) function.

WinGP node, SP-5B10 node, GP4000/LT4000 Series node and GP3000 Series node FTP protoc For FTP protocol, 'Pro-Server EX' supports two modes: normal mode and passive mode. This API specifies the mode of FTP protocol. INT WINAPI EasyFileSetPassiveMode(INT iPassive); Argument iPassive: (In) 0: Normal mode						
iPassive: (In) 0: Normal mode Nor Other than 0: Passive mode Abn	This API specifies the mode of FTP protocol.					
At initialization of ProEasy, the FTP protocol is set to "Normal mode".	Return value Normal end: 0 Abnormal end: Error code					
	At initialization of ProEasy, the FTP protocol is set to "Normal mode".					
Special Note						

27.8 Binary Date and Time / Text Display Conversion

Convert from binary value to text API

Function	Binary value tex	t conversio	n (Time-type)			
Function to convert binary value to TIME-type string.						
INT WINAPI EasyT	IMEToString(DW	ORD dwDa	ata, LPSTR osTi	ime);		
	ArgumentReturn valuedwData: (In) Binary value prior to conversion osTime: (Out) Converted text string*1Normal end: 0 Abnormal end: Error code					
Special Note Input Format						
31	24 16	5	8	0		
	Elapsed time in mill	iseconds (Signe	ed)]		
<i>Output Format</i> %s%02ud%02uh%02 <i>Output Example</i> (1) 01d02h03m04s00 (2) -02d03h04m05s0)5ms	ns (sign, da <u>v</u>	y, hours, minute:	s, seconds, milliseconds))	
Function	FunctionBinary value text conversion (TIME_OF_DAY-type)					
Function to convert b	inary value to TI	ME_OF_DA	AY-type string.			
INT WINAPI EasyT	IME_OF_DAYTo	String(DW	ORD dwData, L	PSTR osTod);		
Argument dwData: (In) Binary osTod: (Out) Conver		version			Return value Normal end: 0 Abnormal end: Error code	
Special Note Input Format						
31 27 Reserved 0 Error bit Output Format %02u:%02u:%02u.% Output Example 23:59:59.999	Hours Minutes – GMT bit 0 0	15 9 Seconds	Milliseconds s, milliseconds)	0		

Function	Binary value text conv	ersion (DATE-ty	/pe)		
	binary value to DATE-ty		1 */		
	ATETa String (DWORD	durData I DSTD	a (Data)		
Argument	DATEToString(DWORD value prior to conversio erted text string ^{*1}		(osDate)	;	Return value Normal end: 0 Abnormal end: Error code
Special Note Input Format					
31 Reserved	24 21 Day 0 Year	8 Month	4 0 Date		
L O Error			Date		
Output Format %04u-%02d-%02u Output Example 2012-01-01	(year, month, date)				
Function	Binary value text conv	ersion (DATE_A	ND_TIM	ME-type)	
Function to convert	binary value to DATE_A	ND_TIME-type	string.		
INT WINAPI EasyI	DATE_AND_TIMEToSt	ing(QWORD qv	vData, Ll	PSTR osDt);	
Argument dwData: (In) Binary osDt: (Out) Convert	value prior to conversio red text string ^{*1}	n			Return value Normal end: 0 Abnormal end: Error code
Special Note Input Format					
63	31 Date	Time	0		
Error bit Output Format %04u-%02u-%02u- Output Example 2012-01-02-03:04:0	%02u:%02u:%02u.%03u	or bit 0 (year, month, d	ate, hours	s, minutes, seconds,	milliseconds)

 *1 Make sure the area is 32 bytes or greater.

 *2 For information about each device access API, refer to 27.2 Device Access APIs.

Convert from text to binary value API

Function	INT WINAPI EasyStringToTIME()			
Function to convert TIME-type string to a binary value.				
INT WINAPI EasyStringToTIME(LPCSTR sTime, DWORD *pdwData);				
Argument		Return value		
sTime: (In) Text string prior to conversion Normal end: 0				
pdwData: (Out) Converted binary value Abnormal end: Error code				
Su anial Mada				

Special Note

Input Format

%s%02ud%02uh%02um%02us%03ums (sign, day, hours, minutes, seconds, milliseconds)

	Day	Hours	Minutes	Seconds	Milliseconds
Setup range	-2424	023	059	059	0999
Units (separator)	d	h	m	S	ms

- Inputs all the items in the setup range as per the input format.
- Setup each item so that when converted to milliseconds, the total results in a value between -2,147,483,648 and 2,147,483,647.

Input Example

01d02h03m04s005ms

Function	INT WINAPI EasyStringToTIME_OF_DAY()
1 unetion	Inter the international provided in the provid

Function to convert TIME_OF_DAY-type string to a binary value.

INT WINAPI EasyStringToTIME_OF_DAY(LPCSTR sTod, DWORD *pdwData);

Argument	Return value
sTod: (In) Text string prior to conversion	Normal end: 0
pdwData: (Out) Converted binary value	Abnormal end: Error code

Special Note

Input Format

%02u:%02u:%02u.%03u (hours, minutes, seconds, milliseconds)

	Hours	Minutes	Seconds	Milliseconds
Setup range	023	059	059	0999
Units (separator)	:	:	•	

• Inputs all the items in the setup range as per the input format.

Input Example 23:59:59.999

Functi	on	INT WINAPI	EasyStringToDA	TE()		
Function to	convert I	DATE-type strii	ng to a binary valu	ıe.		
INT WINAI	PI EasyS	tringToDATE(I	PCSTR sDate, D	WORD *pdwD	Pata);	
		g prior to conve verted binary va				Return value Normal end: 0 Abnormal end: Error code
Special Not Input Forma %04u-%02c	ıt	year, month, da	te)			
			Year	Month	Date	
	Setup r	ange	19708191	112	131	
	Units (separator)	-	-		
• Input Exam 2012-01-01		l the items in th	e setup range as p	er the input for	mat.	
Functi	on	INT WINAPI	EasyStringToDA	TE_AND_TIM	E()	
Function to	convert I	DATE_AND_T	IME-type string to	o a binary value	2.	
INT WINAI	PI EasyS	tringToDATE_A	AND_TIME(LPC	STR sDt, QWO	ORD *pqwData);	
		prior to convers				Return value Normal end: 0 Abnormal end: Error code
Special Not	е					1

	Year	Month	Date	Hours	Minutes	Seconds	Milliseconds
Setup range	19708191	112	-2424	023	059	059	0999
Units (separator)	-	-	-	:	:	•	

• Inputs all the items in the setup range as per the input format.

Input Example 2012-03-21-01:02:03.004

*1 For information about each device access API, refer to 27.2 Device Access APIs.

27.9 Other APIs

Function	Read Time as DWORD	
Acquires a specified of time saved in 6 words	display unit's current time as a numeric value (DWORD-type). This f s from LS2048.	function is valid only for the
DWORD WINAPI E	asyGetGPTime(LPCSTR sNodeName, DWORD* odwTime);	
odwTime: Acquired t	of target node (A Pro-Server EX node cannot be specified.) ime (Time is acquired as a value of DWORD type, (substantially, ined by ANSI).)	Return value Normal end: 0 Abnormal end: Error code
Special Note		
Function	Read Time as VARIANT	
Acquires a specified of time saved in 6 words	display unit's current time as a numeric value (Variant-type). This fur s from LS2048.	nction is valid only for the
DWORD WINAPI E	asyGetGPTimeVariant(LPCSTR sNodeName, LPVARIANT ovTime);
	f target node (A Pro-Server EX node cannot be specified.) ne (Time is acquired as a value of VARIANT type. Internal nat is "Date".)	Return value Normal end: 0 Abnormal end: Error code
Special Note		
Function	Read Time as STRING	
time saved in 6 words	display unit's current time as a character string (LPTSTR-type). This s from LS2048. asyGetGPTimeString(LPCSTR sNodeName, LPCSTR sFormat, LPS	
pFormat: String to sp specification cc in <special no<br="">Other character osTime: Time acquire + 1 (NULL) is this, you must s</special>	of target node (A Pro-Server EX node cannot be specified.) ecify the format of time to be acquired as a string. The format odes subsequent to the percentage (%) symbol are changed as shown te>. rs are expressed without a change. ed as a string (If a memory area larger than the acquired string length not secured, unexpected memory destruction occurs. To prevent secure a memory area larger than the expected string length + 1 rwise, the operation cannot be guaranteed.)	Return value Normal end: 0 Abnormal end: Error code

Special Note

The format specification codes subsequent to the percentage (%) symbol are changed to those listed in the table below. Other characters are expressed without a change. For example, if "%Y_%M %S" is specified, an actual time of "2006/ 1/2 12:34:56" is expressed as a string of "2006_34 56".

Format specification code	Folder
%a	Abbreviated name of day of week (*2)
%A	Formal name of day of week (*2)
%b	Abbreviated name of month (*2)
%B	Formal name of month (*2)
%c	Expression of date and time depending on locale
%#c	Longer expression of date and time depending on locale
%d	Decimal expression of day of month (01 to 31) (*1)
%H	Time expression on 24-hour basis (00 to 23) (*1)
%I	Time expression on 12-hour basis (01 to 12) (*1)
%j	Decimal expression of day of year (001 to 366) (*1)
%m	Decimal expression of month (01 to 12) (*1)
%M	Decimal expression of minute (00 to 59) (*1)
%p	AM/PM division for current locale (*2)
%S	Decimal expression of second (00 to 59) (*1)
%U	Decimal expression of serial week number. Sunday is regarded as the first day of the week. (00 to 53) (*1)
% w	Decimal expression of day of week. Sunday is regarded as "0 ". (0 to 6) (*1)
%W	Decimal expression of serial week number. Monday is regarded as the first day of the week. (00 to 53) (*1)
% X	Expression of date for current locale
%#x	Longer expression of date for current locale
%X	Expression of time for current local (*2)
% y	Decimal expression of low-order 2 digits of the dominical year (00 to 99) (*1)
% Y	Decimal expression of 4 digits of the dominical year (*1)
%z, %Z	Name or abbreviated name of time zone. If time zone is unknown, leave it blank. (*2)
%%	Percentage symbol (*2)

* 1: If "#" is added before d, H, I, j, m, M, S, U, w, W, y or Y (ex. %#d), leading "0" will be deleted. (ex. "05" is expressed as "5".)

* 2: If "#" is added before a, A, b, B, p, X, z, Z or % (ex. %#a), "#" will be ignored.

Function	Read Time as STRING VARIANT	
Acquires a specified time saved in 6 word	display unit's current time as a character string (Variant-type). This for s from LS2048.	unction is valid only for the
DWORD WINAPI E	asyGetGPTimeStringVariant(LPCSTR sNodeName, LPCSTR sForm	at, LPVARIANT ovTime);
pFormat: String to sp specification co listed below. O <special note:<br="">ovTime: Time acquir</special>	of target node (A Pro-Server EX node cannot be specified.) ecify the format of time to be acquired as a string. The format odes subsequent to the percentage (%) symbol are changed to those ther characters are expressed without a change. (For details, refer to > of "Reading time from GP (STRING-type)".) ed as a string (Time is acquired as VARIANT type. Internal nat is "BSTR".)	Return value Normal end: 0 Abnormal end: Error code
Special Note		
Function	Reading entry node status	
check connection stat Single INT WINAPI GetNo osSystemVersion,LP Multi INT WINAPI GetNo	lisplay unit status. Since the response time-out value can be changed, tus. deProperty(LPCSTR sNodeName,DWORD dwTimeLimit,LPSTR os STR osComVersion,LPSTR osECOMVersion); dePropertyM(HANDLE hProServer,LPCSTR sNodeName,DWORD SystemVersion,LPSTR osComVersion,LPSTR osECOMVersion);	sGPType,LPSTR 9 dwTimeLimit,LPSTR
dwTimeLimit: (In) R (If "0" is specif The setting ran The API returns statu Secure an area of at 1 osGPType: (Out) Dis osSystemVersion: (Out) This item is bla osECOMVersion: (O This item is bla	me of node to read status esponse time-out setting value fied, it is set to the default value of 3000 ms.) ge is from 1 to 2,147,483,647. (Unit: ms) as information on the target node to the following area. east 32 bytes for each item.	Return value Normal end: 0 Abnormal end: Error code
Special Note		

Function	Acquiring symbol/group byte size	
Acquires the total nu	mber of bytes required to access a device symbol or group symbol.	
-		- C :).
	fSymbol(LPCSTR sNodeName,LPCSTR sSymbolName,INT* oiByte	
Argument sNodeName: (In) Na	me of entry node with Device/PLC name	Return value Normal end: 0
	Name of target device or symbol name	Abnormal end: Error code
Special Note	, a device symbol, non-alignment group, whole alignment group, or ar	n element of alignment group
Function	Acquiring number of group members	
Acquires the number	of members of a group or symbol sheet (total number of symbols and	d group members).
INT WINAPI GetCo	untOfSymbolMember(LPCSTR sNodeName,LPCSTR sSymbolNam	e,INT* oiCountOfMember);
sSymbolName: (In) I	me of entry node with Device/PLC name Name of target group symbol or symbol sheet (Out) Number of members acquired	Return value Normal end: 0 Abnormal end: Error code
	ol exists in a specified group symbol, the number of members is count in the inner group symbol.	ed as one, even if multiple
Function	Acquiring symbol/group/symbol sheet definition information	
Acquires definition is	nformation (data type, data quantity, etc.)	
	mbolInformation(LPCSTR sNodeName,LPCSTR sSymbolName,INT IMember,LPSTR osSymbolSheetName,SymbolInformation* oSymbo IMember);	
sSymbolName: (In) I	me of entry node with Device/PLC name Name of symbol/group/symbol sheet lMember: (In) Specify a value of the maximum count of desired	Return value Normal end: 0 Abnormal end: Error code
Specify the nu osSymbolSheetName symbol specifi oSymbolInformation structure.	mber of "oSymbolInformation" prepared. e: (Out) The API returns the name of symbol sheet that contains the ed with sSymbolName. Prepare 66 bytes or larger work. : (Out) The API returns acquired detail information in the alignment	
oiGotCountOfSymbo	for the number specified with iMaxCountOfSymbolMember. olMember: (Out) The API returns the information quantity that has ymbolInformation.	

Special Note

• Structure of SymbolInformation

struct SymbolInformation

{

WORDm_wAppKind;// Data type, Symbol: 1 to 20, Group: 0x8000 WORDm_wDataCount; // Data quantity DWORDm_dwSizeOf; // Number of bytes in buffer required for access char m_sSymbolName[64+1];// Name of symbol or group charm_bDummy1[3];// Reserve charm_sDeviceAddress[256+1]; // Device address (For group, leave it blank.) charm_bDummy2[3];// Reserve

};

Acquired information is returned to oSymbolInformation in the alignment structure specified with SymbolInformation. Information on the symbol, group or sheet specified with sSymbolName is set in the first element. Group member information is set in the second and subsequent elements, when sSymbolName indicates a group. When sSymbolName indicates a sheet, information on the whole sheet is set in these elements. When sSymbolName indicates a symbol, there is no information in the second or subsequent elements.

If the target symbol is a bit offset symbol, pay attention to the following points:

(1) When a bit offset symbol is directly specified as an information source symbol (a bit offset symbol is directly specified for sSymbolName), "2" is set to m_dwSizeOf of SymbolInformation, or the first element of oSymbolInformation, as the number of bytes required to access the bit symbol. In this case, since the information source is one symbol, oSymbolInformation does not have second or subsequent element.

(2) When a group symbol is specified as an information source symbol and the specified group contains a bit offset symbol, "0" is set to m_dwSizeOf, or the second or subsequent element of oSymbolInformation, because it indicates the access size required for a group access member.

• If the number of members is unknown, call GetCountOfSymbolMember() to acquire it. To call this function, prepare SymbolInformation as the number of work of the specified count + 1.

27.10 Precautions for Using APIs

■ About data types available with 'Pro-Server EX'

(1) Principal data types that can be specified with APIs, or received in response to APIs

Definition name	Decimal value	Hexadecim al value	Meaning of data
EASY_AppKind_Bit	1	0x0001	Bit Data
EASY_AppKind_SignedWord	2	0x0002	16-bit (Signed) Data
EASY_AppKind_UnsignedWord	3	0x0003	16-bit (Unsigned) Data
EASY_AppKind_HexWord	4	0x0004	16-bit (HEX) Data
EASY_AppKind_BCDWord	5	0x0005	16-bit (BCD) Data
EASY_AppKind_SignedDWord	6	0x0006	32-bit (Signed) Data
EASY_AppKind_UnsignedDWord	7	0x0007	32-bit (Unsigned) Data
EASY_AppKind_HexDWord	8	0x0008	32-bit (HEX) Data
EASY_AppKind_BCDDWord	9	0x0009	32-bit (BCD) Data
EASY_AppKind_Float	10	0xA	Single-precision floating point data
EASY_AppKind_Real	11	0xB	Double-precision floating point data
EASY_AppKind_Str	12	0xC	Character string data
EASY_AppKind_SignedByte	13	0x0013	8 Bit (Signed) Data
EASY_AppKind_UnsignedByte	14	0x0014	8 Bit (Unsigned) Data
EASY_AppKind_HexByte	15	0x0015	8 Bit (HEX) Data
EASY_AppKind_BCDByte	16	0x0016	8 Bit (BCD) Data
EASY_AppKind_TIME	17	0x0017	TIME Data
EASY_AppKind_TIME_OF_DAY	18	0x0018	TIME_OF_DAY Data
EASY_AppKind_DATE	19	0x0019	DATE Data
EASY_AppKind_DATE_AND_TIME	20	0x0020	DATE_AND_TIME Data

(2) Data types available in special cases

Definition name	Decimal value	Hexadecimal value	Meaning of data
EASY_AppKind_NULL	0	0x0000	Indicates that the data type defined for a symbol is used with the API that can use the symbol as the device address.
EASY_AppKind_BOOL	513	0x0201	Handles bit data as Variant BOOL data per bit.
EASY_AppKind_Group	-32768	0x8000	Group symbol
EASY_AppKind_SymbolSheet	-28672	0x9000	Symbol sheet

About entry node name with Device/PLC name

(1) Except for GP Series nodes, you can connect display units to multiple device/PLCs. To access these Device/ PLCs, you must specify the names of the entry node and Device/PLCs.

(2) For some arguments of the Pro-Server EX APIs, you may specify an entry node name only. For other arguments, you must specify a Device/PLC name as well as the entry node name.
<How to specify a Device/PLC name>
To specify a D"evice/PLC name, add "." (dot) after the entry node name.
Example)
AGPNode.PLC1

(3) To access the memory link driver of display units (except those set up as GP Series nodes), specify "#INTERNAL" as the Device/PLC name. (It can be omitted.)

(4) To access the memory link driver of display units (except those set up as GP Series nodes), specify"#MEMLINK" as the Device/PLC name. (It cannot be omitted.)

(5) To access a GP Series node or Pro-Server EX node, you need not specify a Device/PLC name. ("." (dot) is not necessary.)

(6) For internal devices of display units (except those set up as GP Series nodes) and device/PLCs mapped to "system area devices", you can omit the device/PLC name by defining the node with the device/PLC name. In this case, however, 'Pro-Server EX' searches the target device for an internal device first, and then searches for a Device/PLC assigned to the "system area device".

About symbol searching precedence

For the Device Access APIs of 'Pro-Server EX', you must specify the entry node name with Device/PLC name, and the device address or device symbol as a character string. 'Pro-Server EX' judges according to the following order of precedence whether the specified character string directly specifies the device address or a device symbol.

(1) 'Pro-Server EX' searches the symbol sheet for a matching name. If the specified string exists in the symbol sheet, it is regarded as a sheet.

(2) 'Pro-Server EX' regards the specified string as a group name or symbol, and searches a local symbol sheet. If the specified string exists in the local symbol sheet, it is regarded as a local symbol.

(3) If the specified string does not exist in the local symbol sheet, 'Pro-Server EX' searches a global symbol sheet. (In this case, the target global symbol sheet is that for the Device/PLC that has been specified with "entry node name with Device/PLC name". Global symbol sheets for different Device/PLCs are not searched.)

(4) If the specified string does not exist in the global symbol sheet, it is regarded as a device address.

Duplication of name

'Pro-Server EX' provides the following name categories:

- (1) Node Name
- (2) Device/PLC Name
- (3) Trigger Condition Name
- (4) Symbol Sheet Name
- (5) Group/Symbol Name
- (6) ACTION Name

In principle, 'Pro-Server EX' must not have a duplicated name, excepting the following cases:

(1) Duplication of a Device/PLC name causes no problem, if they belong to different entry nodes.

(2) Duplication of a group/symbol name causes no problem, if they belong to different entry nodes or different Device/PLCs.

Duplication of global symbol name and local symbol name

When a Pro-Server EX API uses a symbol to specify a device address and the same symbol name exists for both local symbol and global symbol, it is regarded as a local symbol.

Using Pro-Server EX API for multi-thread application

All functions of Pro-Server EX APIs are synchronous type. (Once a function is called, it will not be returned until processing is completed.)

Therefore, when 'Pro-Server EX' accesses multiple entry nodes by using a single-thread application, processing is executed for individual nodes in sequence.

On the other hand, with a multi-thread application, 'Pro-Server EX' can access another entry node through another thread, even when one thread is used for access to one entry node.

Pro-Server EX APIs can be used for the multi-thread application.

To create a multi-thread application, pay attention to the following points:

(1) In principle, to execute a multi-thread application, use Multi-Handle functions.

(2) To use Multi-Handle functions, you must create Pro-Server EX handles. Use separate Pro-Server EX handles for individual threads.

Even if multiple Pro-Server EX handles are created for one thread, there is no problem. However, you must not use a Pro-Server EX handle that has been created for another thread.

To release a Pro-Server EX handle, use the same thread where the handle has been created.

(3) To use a Pro-Server EX API, you must call EasyInit() first.

However, most Pro-Server EX APIs automatically call EasyInit() when each API is called before EasyInit(). Therefore, when using a single-thread application, you need not consider EasyInit() in your program.

(4) The thread where EasyInit() is called must exist until the end of application. If the thread where EasyInit() is called is closed in the middle of application, the operation cannot be guaranteed.

(5) For general applications, the thread used to start an application will exist until the end of application. (Normally, this applies to applications created by VB or VC.) Therefore, to create a multi-thread application, we recommend you to call EasyInit() at the start of application.

■ Improving cache buffer update efficiency

(1) To use the cache function, you must register a device in the cache buffer. (Register a device on the Pro-Studio EX cache registration screen, or by using the cache buffer control APIs.)Performance of the whole system varies depending on the registration method.

(2) To select a device to be registered, use the device access log function to identify the device that 'Pro-Server EX' accesses.

(3) In principle, you should cache-register a device that has been frequently read.

(4) When multiple devices are registered, the processing speed becomes higher if these devices can be registered in series.

(Ex.1) When LS100 and LS101 are registered in a cache buffer, the processing speed becomes higher if two devices are registered in series from LS100, rather than separately registered. Also, if the interval between two devices is only several words, the processing speed may be increased if these devices are registered in series. (Ex.2) When LS100 and LS103 are registered in a cache buffer, the processing speed becomes higher if four devices are registered in series from LS100, rather than separately registered.

(5) When bit devices are registered in series, the processing speed becomes higher if they can be registered as word devices.

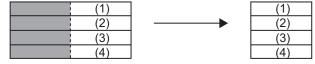
(Ex.) When devices for 20 bits are registered in series from LS123401, the processing speed becomes higher if they are registered in two words from LS1234.

■ 16-bit access operation for device with physically 32-bit width

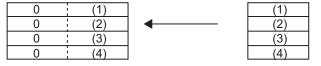
(1) When a 16-bit symbol is assigned to a device with a physically 32-bit width, and the device is accessed with the 16-bit symbol, or when 16-bit data type is directly specified to access a 32-bit device, 'Pro-Server EX' can handle the 32-bit device as a 16-bit device.

In this case, 'Pro-Server EX' executes the following conversion for READ and WRITE APIs.

When defining a 32-bit device as 16-bit type physically and reading it, data on High side is ignored.



When defining a 32-bit device as 16-bit type physically and writing it, 0 is always set on High side.



(2) The above conversion is executed during access using a data transfer function or API.

(3) When data is transferred between GP Series nodes, an error occures.

(4) With older versions of 'Pro-Server', if 16-bit access is executed for a device with physically 32-bit width, an error occures.

■ 16-bit access operation for device with physically 32-bit width

When a 32-bit symbol is assigned to a device with a physically 16-bit width, and the device is accessed with the 32-bit symbol, or when 32-bit data type is directly specified to access a 16-bit device, 'Pro-Server EX' can handle the 16-bit device as a 32-bit device.

In this case, 'Pro-Server EX' handles a series of two devices with a 16-bit width as one device.

About Pro-Server auto start, forced closing and restart

(1) If 'Pro-Server EX' has not been started yet, calling a Pro-Server EX API automatically starts 'Pro-Server EX' (excepting some APIs).

If 'Pro-Server EX' cannot start, the API always returns an error code.

(2) After 'Pro-Server EX' normally starts, calling the second or subsequent API will not start 'Pro-Server EX' again, because 'Pro-Server EX' has already been started.

(3) If 'Pro-Server EX' is closed in the middle of application processing, and then an API is called ('Pro-Server EX' has been closed when the second or subsequent API is called), the API will not start 'Pro-Server EX'. It returns an error code.

(4) Do not close 'Pro-Server EX' in the middle of application processing.

Before closing 'Pro-Server EX', be sure to close the application first. (Do not call an API after closing 'Pro-Server EX'.)

However, if 'Pro-Server EX' is manually restarted from the Windows START menu, the API executes Pro-Server EX recovery processing, and tries to continue processing. If 'Pro-Server EX' can be recovered, it continues processing. However, 'Pro-Server EX' may fail in recovery processing, depending on the previous closing method. For example, recovery processing failures may occur in the following cases:

- When 'Pro-Server EX' is forcibly closed from Task Manager

- When 'Pro-Server EX' is closed during a call of an API

About specification of symbol index

Specification of symbol index is enabled only by a device name for an API. Specification of symbol index is to specify a value in [] after a symbol name, as shown below. The symbol index indicates the device located ahead from the device specified with the symbol name, by the number of devices specified by the "value" of the symbol data type.

(Symbol name)[Value]

Example) Valve [2]

When valve symbol "D100" is specified as "16-bit signed", Valve [2] indicates D102. When "D100" is specified as "32-bit unsigned", it indicates D104.

About queuing cache read and symbol cache read

When queuing cache read (queuing registration using a ReadDevice function (without "D") after BeginQueuingRead) or symbol cache read (ReadSymbol (without "D")) is used, the operation varies depending on which part of target devices has been cache-registered.

- When all target devices have been cache-registered: cache read is executed.
- When all target devices have not been cache-registered: direct read is executed.
- When only some of target devices have been cache-registered: Some of target devices are subjected to cache read, and remaining devices are subjected to direct read. However, cache read is not applied to all of the cache-registered devices. direct read may be applied to some of the cache-registered devices. If you have a trouble in identifying the devices subjected to cache read, you should cache-register all target devices, or use a Direct Read API instead of a Cache Read API.

About APIs that cannot be used for .NET

The following APIs cannot be used for .NET. If these APIs are used, operations cannot be guaranteed.

Symbol access (Byte access)
ReadDevice(), ReadDeviceD(), WriteDevice(), WriteDeviceD()
ReadDeviceM(), ReadDeviceDM(), WriteDeviceM(), WriteDeviceDM()
ReadSymbol(), ReadSymbolD(), WriteSymbol(), WriteSymbolD()
ReadSymbolM(), ReadSymbolDM(), WriteSymbolM(), WriteSymbolDM()
Symbol size acquisition function

SizeOfSymbol()

About APIs that cannot be used in VB functions

You cannot use the following APIs in Visual Basic functions. If these APIs are used, we are unable to verify that the functions will work.

WriteDeviceDATE_AND_TIME(), WriteDeviceDATE_AND_TIMEM(), WriteDeviceDATE_AND_TIMED(), WriteDeviceDATE_AND_TIMEDM(),

EasyStringToDATE_AND_TIME(), EasyDATE_AND_TIMEToString()

When using simple DLL in a multi-thread application

All functions of Pro-Easy APIs are synchronous type. (Once a function is called, it will not be returned until processing is completed.) Therefore, when accessing multiple entry nodes by using a single-thread application, processing is executed for individual nodes in sequence. On the other hand, with a multi-thread application, you can access another entry node through another thread, even when one thread is used for access to one entry node. Pro-Easy APIs can be used for the multi-thread application.

To create a multi-thread application, pay attention to the following points:

- 1. In principle, to execute a multi-thread application, use Multi-Handle functions.
- 2. To use Multi-Handle functions, you must create 'Pro-Server EX' handles. Use separate 'Pro-Server EX' handles for individual threads. Even if multiple 'Pro-Server EX' handles are created for one thread, there is no problem. However, you must not use a 'Pro-Server EX' handle that has been created for another thread. To release a 'Pro-Server EX' handle, use the same thread where the handle has been created.
- 3. To use 'Pro-Server EX API', you must call EasyInit() first. As most Pro-Server EX APIs automatically call EasyInit() when each API is called before EasyInit(), you need not to consider EasyInit() call in your program.
- 4. In the multi-thread program, the program must call EasyInit() first from the thread (main thread) which was started first. When you call a Pro-Server EX API except from the main thread, call EasyInit() from the main thread in advance.

Message Process in Windows

Most of the Windows programs are event-driven, i.e. displaying the dialog box or playing the sounds according to various events including "an icon is clicked", "a mouse is moved", or "a key is pressed".

When an event occurs, Windows will send the message showing the event type to the application. The application confirms that the event occurs by receiving the message and executes each process.

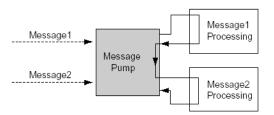
In this manual, the part which receives messages in order and branches into each process (corresponding to DoEvents for VB, or the part executing GetMessage() and DispatchMessage() for VC) is called the message pump. The message pump is not much recognized because it is hidden in the VC or VB framework when programming with VC or VB normally. However, unless this message pump operates properly, Windows applications will cause unintended operation.

For example, when it takes long time for a routine to process a message and recover, the application fails to process the event because it cannot receive an event which occurs in the meantime from Windows.

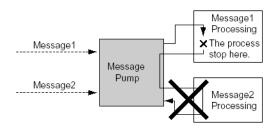
Example) Assume that messages are sent from Windows in the order of message 1 to message 2.

The message pump takes out the message 1 and calls the subroutine for message 1.

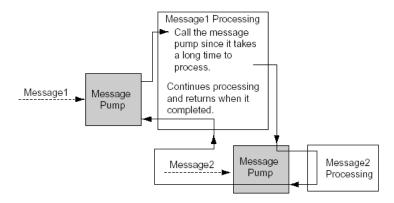
Then, when the message pump recovers from above, it takes out the following message (message 2) and calls the subroutine for message 2.



In this case, assume that it takes long time for processing message 1. Then the message pump cannot process message 2 without recovering.



In such case, force the message pump to run. (calling DoEvents,VC for VB, or GetMessage() and DispatchMessage() for VC)



Windows applications are created assuming an application should run the message pump properly. "Pro-Server EX API" runs the message pump using function for time-consuming process so as to avoid the case shown in (Example).

Prohibition of API Double Call

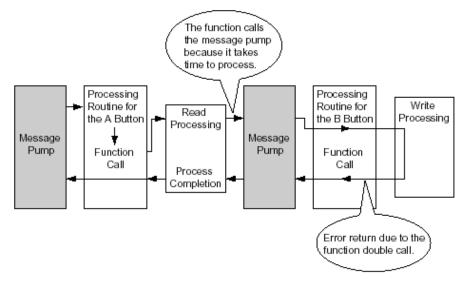
'Pro-Server EX API' prohibits another communication while communicating with a party (while calling a 'Pro-Server EX' function)(double-call). (Double-call is enabled if using the Multi-Handle. For details, refer to the section of Multi-Handle.) However, as 'Pro-Server EX API' runs the message pump inside API, a user program will start to run when an event occurs.

When API is called in the message process routine, double-call may occur.

Examples of double-call are shown below.

1. Double-call by pressing 2 buttons

Assume that there are 2 buttons, A and B. Device read API is called when A is pressed; device write API is called when B is pressed. In this case, press the button B to cause the device write API to be called while calling the device read API when pressing the button A, which leads API double-call and error occurs.



2. Double-call by timer

When periodical process is executed in the Windows program, timer events are often used. However, API doublecall may happen in the program using timer events due to careless programming.

- (1) Call the device read API periodically per second, read the device and display it.
- (2) Such programs as call the device write API when a button is pressed and write the value in the device causes an error in the following cases.

When pressing the button (2) while reading a timer event (1), and the process (2) starts to run

When a timer event occurs while writing (2) and read (1)

Solutions to avoid API Double-Call

Solutions to avoid API double-call are shown below.

- (1) Improve the algorithm not to execute API double-call in a user program.For example,
 - 1. Timer should be always cancelled at the head of timer process routine and button process routine.
 - 2. While a process is running by pressing a button, the button or another button should be ignored even if pressed.
- (2) API double-call does not occur if the 'Pro-Server EX' handle using multi-handle is different.

Use API in Multi-Handle type to set the handle of the program in the area which is possible to cause doublecall to different handle.

(3) Message should not be processed inside API

Call EasySetWaitType() by argument 2. However, in this case, other problems such as an application causes unintended operation may occur, because other messages except the one which causes double-call will not be processed.

How to read character strings in VB

(1) Use ReadDeviceStr to read character strings in VB

In this case, you need to specify (fix) the size of storing destination of character strings read in advance.'

```
Public Sub Sample1 ()
```

 Dim strData As String * 10
 'Correct designation method because it designates the size to read.

 'Dim strData As String
 'Incorrect designation method because it does not designate the character 'string size.

Dim lErr As Long

lErr = ReadDeviceStr ("GP1", "LS100", strData, 10)

If lErr <> 0 Then

MsgBox "Read Error = " & lErr

Else

MsgBox "Read String = " & strData

End If

End Sub

(2) Use Variant type if you use ReadDeviceVariant to read character strings in VB, but not specify the size of storing destination of character strings read in advance.

```
Public Sub Smaple2 ()

Dim lErr As Long

Dim vrData As Variant 'Designate the Variant type to the area to save data read.

lErr = ReadDeviceVariant ("GP1", "LS100", vrData, 10, EASY_AppKind_Str)

lf lErr > 0 Then

MsgBox "Read Error = " & lErr

Else

MsgBox "Read String = " & vrData

End If

End Sub
```

Note that display unit uses NULL for the completion of character strings. For that reason, you need to shorten the character string if the character string obtained in the above method includes NULL as the completion of character strings.

Sample functions to shorten character strings to NULL are shown below.

```
Dim i As Integer

i = InStr (1, strData, Chr$(0), vbBinaryCompare)

If 0 < i Then

TrimNull = Left (strData, i - 1)

Else

TrimNull = strData

End If

End Function
```

27.11 Using APIs (Examples)

By using the read/write functions provided by 'Pro-Server EX', you can read/write data from/into a VB or VC application.

This section describes the procedure for reading/writing a specified symbol with the APIs.

"27.11.1 VB Support Function"

"27.11.2 VC Support Function"

"27.11.3 VB .NET Support Function"

"27.11.4 C# Support Function"

27.11.1 VB Support Function

NOTE • You cannot use the DATE_AND_TIME data type or API functions in VB functions.

VB: Declaration statement

1 Select [Programming Assist] - [VB & VBA] - [Declare Statement].

饕 Pro-Studio E	Х 2.прх						_ 🗆 ×
File Edit Tool	Programming Assist Settin	a Help					
Start		Statement	≷ Feature 🔉 📄		ave 🔉 🏡 Ti	ransfer	Monitor
	VC++ • write F	unction					Status
	EXCEL Read Fi						
Symbol		Node Name AGP1		Device	Name PLC1		
Group	Ungroup	Sheet Name Sheet3		Cohitan	a global symbol shee	-t	
Insert	Delete	Sheet Name Shoeto	I	Secilias	a giobai symbol sriee	st.	
Сору	Cut Paste	Symbol	Data Type	Consec utive	Device Address	No. of Data	Comment
Symbol Sheet		_D0040_WORD	16Bit(Signed)		D0040	3	
Add	Delete					1	
						1	
Check Duplica	tion/List Used Addresses					1	
	Non Electro cour Addicestes					1	
Global Cor	natant Catting Corean					1	

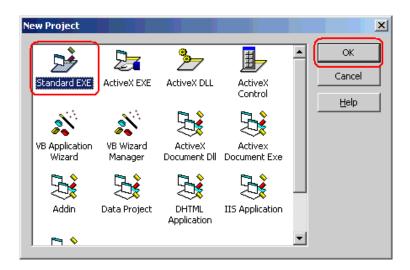
The VB declaration statement is coped to the clipboard.

	X
"Attribute VB_Name = "ProEasy" Option Explicit	
ProEasy.TXT Pro-Server API Declarations for Visual Basic	
Copyright (C) 1998-2006 Digital Electronics Corporation	
ProEasy.DLL Version 1.0 Complying with Pro-Server Version 1.0 etc." This string was copied to the clip board. Please paste and use in your application.	

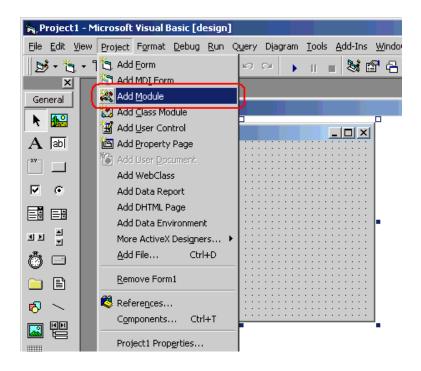
2 Start Microsoft Visual Basic, and select [New Project] from [File] on the menu.

crosoft ¥isua	Bas	ic					
Edit View Pro	oject	Format	<u>D</u> ebug	<u>R</u> un	Q <u>u</u> ery	Djagram	<u>T</u> ools
<u>N</u> ew Project	Ctrl	+N	Ba (3 44		C4 🕨	
🖻 Open Project	Ctrl-	+0					
Add Project							
<u>R</u> emove Project							
Save Project							
Sav <u>e</u> Project As,							
Save	Ctrl	+5					
Save As							
Save Selection							
Save C <u>h</u> ange Scr	ript						
9 Print	Ctrl	+P					
Print Setyp							
Make							
Make Project <u>G</u> ro	oup,,,						
E <u>x</u> it	Alt	+Q					

3 Select [Standard EXE], and click the [OK] button.



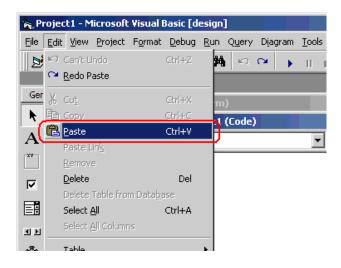
4 Select [Add Module] from [Project] on the Microsoft Visual Basic menu.



5 Select [Module] in the [New] tab, and click the [Open] button.

Add Module	? ×
New Existing	
Modula	
	<u>pen</u> Cancel <u>H</u> elp
☐ Don't show this dialog in the f <u>u</u> ture	

6 Select [Paste] from [Edit] on the Microsoft Visual Basic menu, and paste the declaration statement (data on the clipboard) to the added standard module.



The deceleration statement is now pasted.

🦂 P	roject1 - Ma	dule1 (Code)	
(G	eneral)		(Declarations)
	Declare	Function	ReadSymbolD Lib "ProEasy.DLL" (ByVal sNod
	Declare	Function	ReadSymbolVariant Lib "ProEasy.DLL" (ByVa
	Declare	Function	ReadSymbolVariantD Lib "ProEasy.DLL" (ByV
	Declare	Function	ReadSymbolM Lib "ProEasy.DLL" (ByVal hPro
	Declare	Function	ReadSymbolDM Lib "ProEasy.DLL" (ByVal hPr
	Declare	Function	ReadSymbolVariantM Lib "ProEasy.DLL" (ByV
	Declare	Function	ReadSymbolVariantDM Lib "ProEasy.DLL" (By
	Declare	Function	WriteSymbol Lib "ProEasy.DLL" (ByVal sNod
	Declare	Function	WriteSymbolD Lib "ProEasy.DLL" (ByVal sNo
	Declare	Function	WriteSymbolVariant Lib "ProEasy.DLL" (ByV
	Declare	Function	WriteSymbolVariantD Lib "ProEasy.DLL" (By
	Declare	Function	WriteSymbolM Lib "ProEasy.DLL" (ByVal hPr
	Declare	Function	WriteSymbolDM Lib "ProEasy.DLL" (ByVal hP
	Declare	Function	WriteSymbolVariantM Lib "ProEasy.DLL" (By
	Declare	Function	WriteSymbolVariantDM Lib "ProEasy.DLL" (B
	' CF-CAH	RD access	control API
	Declare	Function	EasyFileSetPassiveMode Lib "ProEasy.DLL"
E			

This is the end of the function (read/write function) declaration procedure.

The above 1 to 6 steps apply to both reading and writing applications.

The following procedure varies depending on whether the application is intended for reading or writing, and so is explained individually.

To create a "Reading" application, refer to steps 7 to 16.

To create a "Writing" application, refer to steps 17 to 26.

Creating "Reading" application

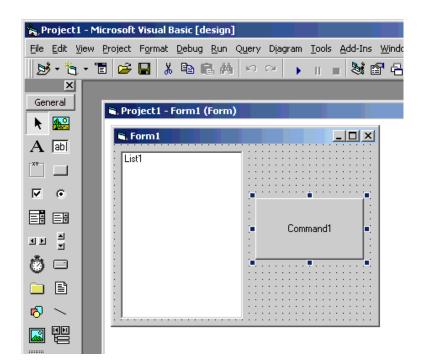
This section describes the procedure for creating an application that reads and displays data (16-bit signed data) for three points with a click on [Command1].

💐 Form1	
1111 2222 3333	Command1

7 Select [ListBox] and paste it to [Form1].

<u>File E</u> dit <u>V</u> iew	Project F <u>o</u> rmat <u>D</u> ebug <u>R</u> un Q <u>u</u> ery D <u>i</u> agram <u>T</u> ools <u>A</u> dd-Ins <u>W</u> indo
😼 • 🍖 • '	Т 😅 🖬 🐍 純 🗠 ↔ → 🗊 🗉 😹 🖆 名
×	
General	🚔 Project1 - Form1 (Form)
N 🔛	S. Form1
A abi	List1
• •	
1 N 1	· · · · · · · · · · · · · · · · · · ·
.#.	
R 🔨	
	· · · · · · · · · · · · · · · · · · ·

8 Select [CommandButton] and paste it [Form1].



9 Select a target symbol name from those registered in 'Pro-Server EX'. (Select the symbol with first-address for reading.)

💱 Pro-Studio EX 2.npx 💶 🗖 🗙						
File Edit Tool Programming Assist Settin	g Help					
Start >> 🟹 Node >>	🍐 Symbol ≫ 🄾	≷ Feature ン 📑	- Sa	ave ≫ 🄖 T	ransfer	Monitor Status
Symbol	Node Name AGP1		Device	Name PLC1		
Group Ungroup Insert Delete	Sheet Name Sheet3		Set it as	a global symbol shee	et.	
Copy Cut Paste	Symbol	Data Type	Consec utive	Device Address	No. of Data	Comment
Symbol Sheet	_D0040_WORD	16Bit(Signed)		D0040	3	<u> </u>
Add Delete					1	
					1	
Check Duplication/List Used Addresses					1	
					1	
Global Constant Setting Screen					1	
Ru ▼ Pro-Server EX					1	
E ■ PC1 (192.168.0.1)					1	
#INTERNAL:Sheet1		<u> </u>			1	
GP3000 Series					1	
🖻 - 💽 AGP1 (192.168.0.100)		<u> </u>			1	
#INTERNAL:Sheet2					1	
→ WinGP					1	
► LT3000		<u> </u>			1	
• GP Series		<u> </u>			1	
Global Symbol					1	
					1	
					1	
					1	
		I		l	1	

÷.,

10 Select [Programming Assist] - [VB & VBA] - [Read Function] on the menu.

💱 Pro-Studio E	Х 2.прх					
File Edit Tool	Programming Assist	Setting	; Help			
Start	VB & VBA > VC++ > EXCEL >	Declare : Write Fu Read Fu	nction	» 🦂	Feature	» [:
Symbol			Node Name /	AGP1		
Group	Ungroup		Sheet Name	Sheet3		
Insert	Insert Delete Copy Cut Paste					
Сору			Symbol		Data Typ	е
Symbol Sheet			_D0040_WORD)	16Bit(Signed)	
Add	Delete					
Check Duplic	Check Duplication/List Used Addresses					
Global Constant Setting Screen						

х

The read function is copied to the clipboard.

"ReadSymbol("AGP1.PLC1","Production_LineA.LineA_sensorinput",Enter value(Receive buffer) here); " This string was copied to the clip board. Please paste and use in your application.

Pro-Server EX Reference Manual

11 Double-click [Command1] on [Form1], and paste the data on the clipboard (read function) between 'private sub Command1_Click()' and 'End Sub'.

🖉 Pr	roject1 - Form1 (Code)					
Co	mmand1	•	Click			•
	Private Sub Command1_Click() lErr = ReadSymbolVariant("AGP1.PL0	C1","_DOO4	O_WORD",Enter	a variant-type	variable	here)
	End Sub					
						_
						► //.

12 Declare the area (Array) to store the read data. Ensure that the array type (in this example, Variant-type) is matched with the data type of the symbol being used.

Je P	Project1 - Form1 (Code)	
Co	iommand1 Click	•
	Private Sub Command1_Click() Dim wData As Variant	•
	lErr = ReadSymbolVariant("AGP1.PLC1","_D0040_WORD",Enter a variant-type variable here)
	End Sub	
		▶

13 Specify the first area (wData) to store the read data.

🖉 Project1 - Form1 (Code)		
Command1	Click	•
Private Sub Command1_Click() Dim wData As Variant		
lErr = ReadSymbolVariant("AGP1	.PLC1", "_DOO40_WORD", <u>wData</u>)	
End Sub		

14 The List Box displays the read data for three points (wData(0), wData(1) and wData(2)) in sequence.

🐖 Project1 - Form1 (Code)		
Command1	- Click	•
Private Sub Command1_Click() Dim wData As Variant		-
lErr = ReadSymbolVariant("AGP1.PLC1",	"_DOO40_WORD", wData)	
List1.AddItem CStr(wData(0)), 0 List1.AddItem CStr(wData(1)), 1 List1.AddItem CStr(wData(2)), 2		
End Sub		
		-

15 Select [Start] from [Run] on the Microsoft Visual Basic menu.

🙀 Project1 - N	Microsoft Visual Basic [design]
<u>Eile E</u> dit <u>V</u> iew	Project Format Debug Run Query Diagram Iools Add-Ins Window Help
😼 - 🏷 -	👕 🖙 🔚 🕺 🗈 🖡 🔁 Start 🛛 👘 🕫 🛠 🔁 🐊 Ln 11, Col 1
×	Start With Full Compile Ctrl+F5
General	II Break Ctrl+Break
	Project1 - Form1 (
	Shift+F5
A abl	
XV-	📮 Project1 - Form1 (Code)
	Command1 Click
• •	Private Sub Command1 Click()
	Dim wData As Variant
4 N N N	lErr = ReadSymbolVariant("AGP1.PLC1", "_D0040_WORD", wData)

16 Click [Command1]. Then, the List Box displays the data for three points from the symbol "_D0040_WORD".

💐 Form1	
1111 2222 3333	Command1
1	

Creating "Writing" application

This section describes the procedure for creating an application that writes the data (16-bit signed data) entered for three points with a click on [Command1].

💐 Form1		_ 🗆 ×
1010		[:
2020		Command1
3030		

17 Select [TextBox] and paste it to [Form1]. Paste [Text Box] for three items.

🐂 Project 1	- Microsoft Visual Basic [design]
<u>File E</u> dit <u>V</u> i	jew <u>P</u> roject F <u>o</u> rmat <u>D</u> ebug <u>R</u> un Q <u>u</u> ery Diagram <u>T</u> ools <u>A</u> dd-Ins <u>W</u> indow <u>H</u> elp
🛛 🔊 • 街	・ 盲 😂 🖬 🐍 純 🗠 🗠 🔸 🗉 🖷 😽 📽 🕾 😤 🛠
×	
General	🗟 Project1 - Form1 (Form)
	S. Form1
A abl	
	Text1
•	
	Text2
▲ ■ ►	
s A 🗆	Text3
<u> </u>	
🔊 🔨	

18 Select [CommandButton] and paste it [Form1].

🐂 Project 1	- Microsoft Visual Basic [design]
<u>Eile E</u> dit <u>V</u> i	ew <u>P</u> roject F <u>o</u> rmat <u>D</u> ebug <u>R</u> un Q <u>u</u> ery Diagram <u>T</u> ools <u>A</u> dd-Ins <u>W</u> indow <u>H</u> e
🛃 - 🏂	・ 🏽 😅 🖶 👗 🛍 🖻 🎮 🖙 📯 🕟 🔳 🔡 😤 👹 🕾 👹 😤
×	
General	🖷, Project1 - Form1 (Form)
N 🔛	
A abi	S. Form1
	Text1
	Command1
ৰ চাৰ	
	Text3
9 🗆	· · · · · · · · · · · · · · · · · · ·
8 🔨	
OLE	

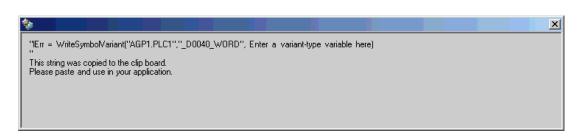
19 Select a target symbol name from those registered in 'Pro-Server EX'. (Select the symbol with first-address for writting.)

💱 Pro-Studio EX 2.npx						
File Edit Tool Programming Assist Sett	ing Help					
Start >> 🛐 Node >>	🍐 Symbol ≫ 🧳	Feature ⋗ 📑	Sa	ive ᠉ 🆄 Ti	ransfer	Monitor Status
Symbol	Node Name AGP1		Device	Name PLC1		
Group Ungroup	Sheet Name Sheet3		Set it as a	a global symbol shee	et.	
Insert Delete						
Copy Cut Paste	Symbol	Data Type	Consec utive	Device Address	No. of Data	Comment
Symbol Sheet	_D0040_WORD	16Bit(Signed)		D0040	3	
Add Delete					1	
					1	
Check Duplication/List Used Addresses					1 -	
			<u> </u>		1	
Global Constant Setting Screen			<u> </u>		1	
		·	<u> </u>		1	
🖻 🐙 PC1 (192.168.0.1)			<u> </u>		1	III
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII			<u> </u>		1	II
⊟			<u> </u>		1	i
#INTERNAL:Sheet2			<u> </u>		1	·
PLC1:Sheet3 A Series CPU [<u> </u>	·	1	
→ WinGP					1	
→ LT3000 → GP Series	li				1	
Global Symbol					1	
					1	
					1	
					1	
					1	
	•					

20 Select [Programming Assist] - [VB & VBA] - [Write Function] on the menu.

铃 P	ro-St	udio E>	(2.прх									
File	Edit	Tool	Programm	ing Assist	Settin	ng	Help					
	1	Start	VB & VE				atement	- >>>	Feature	» [=	Sa Sa	ave ≫ 👔
			VC++		Write F				3			
⊢s	ymbol		EXCEL	-	Read F	una		ame AGP1			Device	Name PLC1
	Group Ungroup Insert Delete Copy Cut Paste					L	Sheet Name Sheet3					
						Sy	mbol	Data T	ype	Consec utive	Device Ac	
S	ymbol :	Sheet -				Ľ	_D0040_\	WORD	16Bit(Signed)			D0040
		Add		Delete		L			<u> </u>			
						L			<u> </u>		<u> </u>	
	Check	Duplica	tion/List Us	ed Addres	ses	Ľ						
	Glo	bal Cor	nstant Settin	a Screen								

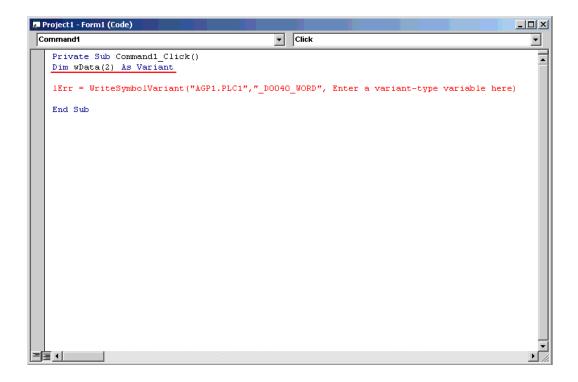
The write function is copied to the clipboard.



21 Double-click [Command1] on [Form1], and paste the data on the clipboard (write function) between the Sub statement and the End Sub statement.

📕 Pr	ject1 - Form1 (Code)	
Cor	mand1 Click	•
	Private Sub Command1_Click() Err = WriteSymbolVariant("AGP1.PLC1","_DOO40_WORD", Enter a variant-type variable here	
	Ind Sub	
33	4	▼

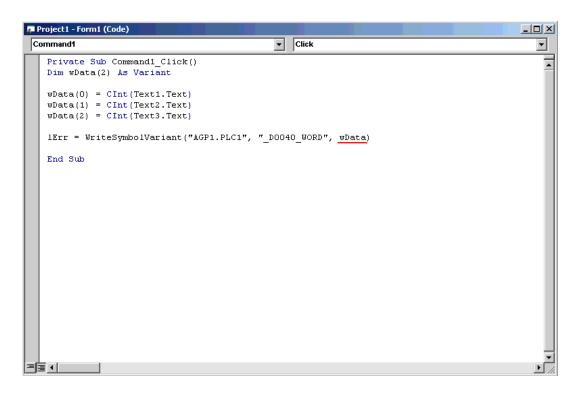
22 Declare the area (alignment) to store the written data. Ensure that the alignment type (in this example, Variant-type) is matched with the data type of the symbol being used.



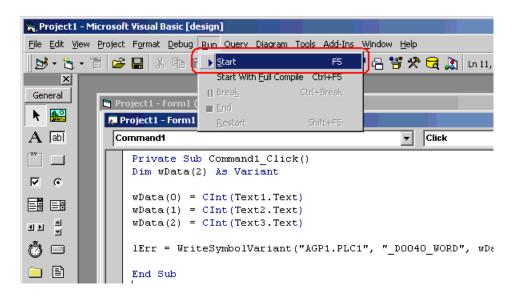
23 Set the data entered in [TextBox] into the alignment.

🖉 Project1 - Form1 (Code)		
Command1	Click	-
Private Sub Command1_Click() Dim wData(2) As Variant		^
wData(0) = CInt(Text1.Text) wData(1) = CInt(Text2.Text) wData(2) = CInt(Text3.Text)		
lErr = WriteSymbolVariant("AGP1.	PLC1","_DOO40_WORD", Enter a varia	ant-type variable here)
End Sub		
		▼ ▼

24 Specify the first area (wData) where the written data has been set.



25 Select [Start] from [Run] on the Microsoft Visual Basic menu.



26 After entering values (for three points) in [TextBox], click [Command1]. Then, 'Pro-Server EX' executes the writing of the data for three points from the symbol "_D0040_WORD".

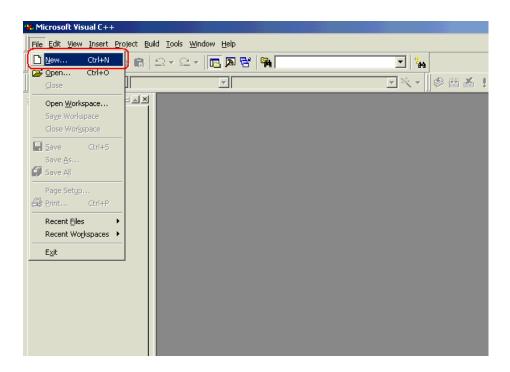
Command1

27.11.2 VC Support Function

For example, this section describes the procedure for creating a dialog-based application by using MFC (Microsoft Foundation Class).

VC: Declaration statement

1 Start Microsoft Visual C++, and select [New] from [File].



2 After selecting [MFC AppWizard(exe)] in the [Projects] tab, enter [Project name] and [Location], and click the [OK] button.

In this example, "Sample" is entered for [Project name], and "C:\Program Files\Pro-face\Pro-Server EX\PRO-SDK\VC" (Windows Vista or later: "C:\Pro-face\Pro-Server EX\PRO-SDK\VC") is entered for [Location].

New	<u>? ×</u>
Files Projects Workspaces Other Documents	1
ATL COM AppWizard Cluster Resource Type Wizard Custom AppWizard Database Project DevStudio Add-in Wizard Extended Stored Proc Wizard ISAPI Extension Wizard ISAPI Extension Wizard MFC ActiveX ControlWizard MFC AppWizard (dll) MFC AppWizard (exe) Utility Project Win32 Application Win32 Console Application Win32 Static Library Win32 Static Library	Project <u>mame</u> : Sample Logation: C:\PROGRAM FILES\PRO-FAC
	OK Cancel

3 Select [Dialog Based] for "What type of application would you like to create?", and click the [Finish] button.

MFC AppWizard - Step 1	<u>? ×</u>
Application OK Cancel	What type of application would you like to create? <u>Single document</u> <u>Multiple documents</u> <u>Dialog based</u>
	☑ Document/yjew architecture support? What language would you like your resources in? English [United States] (APPwZENU.DLL ▼
< <u>B</u> ack	Next > Einish Cancel

4 Click the [OK] button to complete the project.

New Project Information	1
AppWizard will create a new skeleton project with the following specifications:	
Application type of Sample: Dialog-Based Application targeting: Win32	
Classes to be created: Application: CSampleApp in Sample.h and Sample.cpp Dialog: CSampleDIg in SampleDIg.h and SampleDIg.cpp	
Features: + About box on system menu + 3D Controls + Uses shared DLL implementation (MFC42.DLL) + ActiveX Controls support enabled + Localizable text in: English [United States]	
Project Directory: C:\PROGRAM FILES\PRO-FACE\PRO-SERVER EX\PRO-SDK\VC\Sample	
OK Cancel	

The read/write functions provided by 'Pro-Server EX' are available as DLL. To use DLL, you must specify a LIB file.

5 Select [Settings] from [Project] on the Microsoft Visual C++ menu.

*	sa	mple	- Mici	rosoft V	'isual C+	+-				
	<u>F</u> ile	<u>E</u> dit	<u>V</u> iew	Insert	<u>P</u> roject	<u>B</u> uild	Layout	<u>T</u> ools	<u>W</u> indow	Help
						Acti <u>v</u> e To Pr	Project		+ +	
]]	LSa	mpleD	Ig		Dep	ender	cies			SampleDlg
-		_		(<u>S</u> ett	ings		A	lt+F7	PLE_DIALOG (Dialog)
	-	1 0		e resou	Exp	ort <u>M</u> a	kefile			
		÷	📄 Dial 📄 Icor	-	Inse	ert Pro	ject into '	Worksp	ace	······
			Strir	ng Table					то	D0: Place dialog controls here.

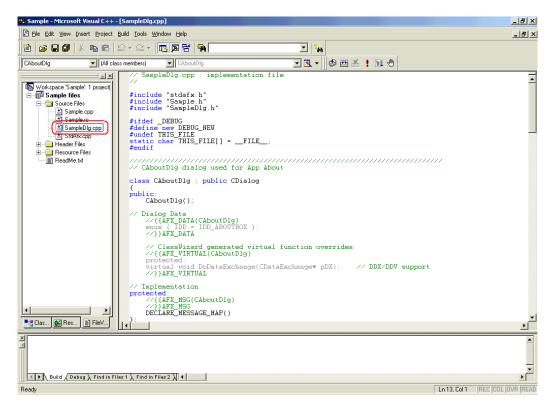
6 Specify a LIB file for [Object/library modules] in the [Link] tab. Then, click the [OK] button.

The LIB file (ProEasy.lib) exists in "PRO-SDK\Vc\Public" in the folder where 'Pro-Server EX' has been installed. In this example, "..\Public\ProEasy.lib" is specified.

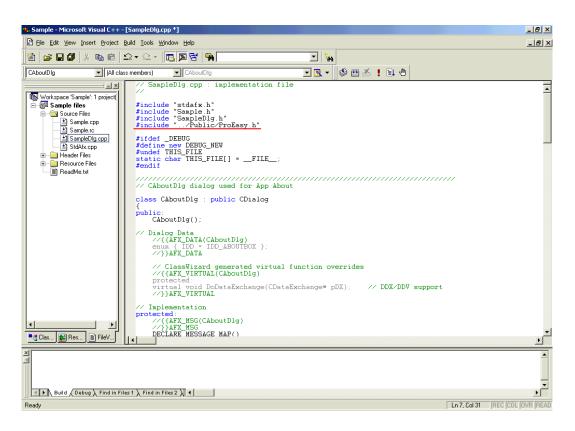
Project Settings	<u>? ×</u>
Settings For: Win32 Debug	General Debug C/C++ Link Resourc
⊕- 🛱 Sample	Category: General
	Output file <u>n</u> ame:
	Debug/Sample.exe
	Object/library modules:
	/Public/ProEasy.lib
	☑ <u>G</u> enerate debug info Ignore all default li <u>b</u> raries
	✓ Link incrementally
	Enable profiling
	Project Options:
	/nologo /subsystem:windows /incremental:yes /pdb:"Debug/Sample.pdb" /debug /machine:1386 /out:"Debug/Sample.exe" /pdbtype:sept
	OK Cancel

7 To use read/write functions provided by 'Pro-Server EX', you must include a header file (ProEasy.h). After clicking the [FileView] tab in the [Work Space] window of Microsoft Visual C++, double-click the "SampleDig.cpp" file.

In this example, the read/write functions are used in the "SampleDig.cpp" file.



8 Add #include "..\Public\ProEasy.h" to the "SampleDig.cpp" file. This completes the function (read/write function) decleration procedure.



The above 1 to 8 steps apply to both reading and writing applications.

The following procedure varies depending on whether the application is intended for reading or writing, and so is explained individually.

To create a "Reading" application, refer to steps 9 to 30.

To create a "Writing" application, refer to steps 31 to 47.

Creating "Reading" application

This section describes the procedure for creating an application that reads and displays data (16-bit signed data) for three points with a click on [Button1].

💑 Sample		×
		ОК
1111		Cancel
1111 2222 3333		
	Button1	
,		

9 After clicking the [ResourceView] tab in the [Work Space] window of Microsoft Visual C++, double-click [IDD_SAMPLE_DIALOG].

Select [Static Text] at the center of the dialog box, and delete it.

33 Sample - Microsoft Visual C++ - [Sample.rc - IDD_SAMPLE_DIALOG (Dialog)]		_	- 8 ×
📴 Eile Edit View Insert Project Build Layout Iools Window Help		I	_ 8 ×
12 2 2 3 1 1 1 1 2 2 2 2 7 1 1 1 1 1 1 1 1 1 1 1 1			
CSampleDig 🔽 (All class members) 🔽 💊 CSampleDig 🔽 🔯 🗸 🖠 🗄			
Sample resources			
OK Cancel Cancel Version	Con 🗵		
	Aa ab		
TODO: Place dialog controls here.	x •		
	● ■		
	0- 🖙		
	山田		
	S 💽		
X			
Build & Debug & Find in Files 1 & Find in Files 2			Ē
Ready	50, 90	1 [™] 200×8	READ

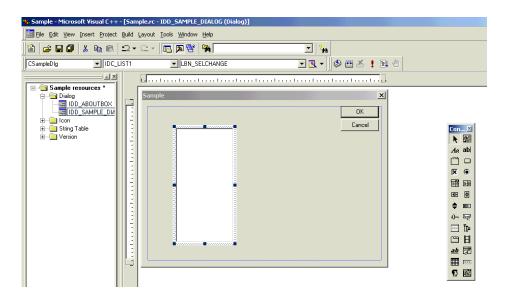
10 Select [Customize] from [Tools] on the Microsoft Visual C++ menu.

🤲 Sample - Microsoft Visual C++ - [Sample.rc	- IDD_SAMPLE_DIALOG (Dialog)]
Eile Edit View Insert Project Build Layout	Tools Window Help
) 🎦 😅 🖬 🕼 👗 🖬 💼 그 - 그 -	Source Browser Alt+F12
CSampleDig (All class members)	A Register Control
Sample resources	• Error Lookup • Inner Lookup • Active2X Control Test Container • Inner Lookup • OLE/COM Object Viewer • Inner Lookup
BIDD_ABOUTBOX	→ Spy++
⊕ Gring Table ⊕ Qring Table ⊕ Qring Table	Customize
	Macro Record Quick Macro Ctrl+Shift+R
	Play Quick Macro Ctrl+Shift+P

11 Check the [Controls] checkbox in the [Toolbars] tab, and click the [Close] button.

Customize	? ×
Commands Toolbars Tools Keyboard Add-ins and M	tacro Files
⊥oolbars: ✓ Menu bar ✓ Standard Build ✓ Build MiniBar △ATL Resource ✓ Dialog ✓ Controls Edit Debua Toolbar name:	<u>N</u> ew Delete Reset Reset <u>A</u> ll
	Close

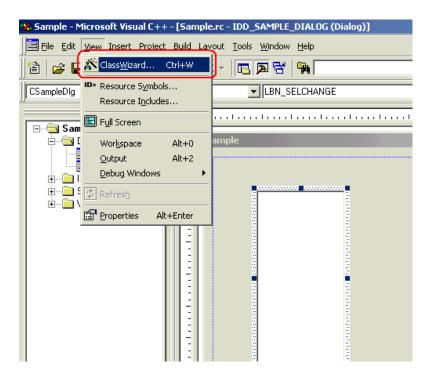
12 Select [ListBox], and paste it to the dialog box.



13 Right-click the pasted [ListBox], and select [Property]. The [List Box Propertis] dialog box appears. Then, uncheck the [Sort] checkbox.

List	Вох Р	Propertie	:5					×
-ja	ę	General	Styles	Extended Styles				
S	ele <u>c</u> tio	in:						
F	Single	-	Border	🔲 Horizontal scro	oll 🗖	Want <u>k</u> ey	input	
	<u>)</u> wner d	draw: (Sort	✓ Vertical scroll		<u>D</u> isable no	scroll	
	No	-	✓ <u>N</u> otify	🔲 No <u>r</u> edraw	☑	No integra	l <u>h</u> eighi	τ
Г	H <u>a</u> s	strings	<u>M</u> ulti-colu	ımn 🔲 <u>U</u> se tabstops				

14 Select [ClassWizard] from [View] on the Microsoft Visual C++ menu.



15 Select the [Member Variables] tab, and select "IDC_LIST1" for [Control IDs].

MFC ClassWizard					<u>? ×</u>
Message Maps	Member Variables	Automation	ActiveX Event	s Class Info	
Project:		Class na			Add Class 🔻
Sample C:\\VC\Sample\	SampleDlg.h, C:\\V0	CSample CSample	-	•	Add Variable
Control IDs:		Туре	Member		<u>D</u> elete Variable
IDC_LIST1)	Update <u>C</u> olumns
IDOK					<u>B</u> ind All
, Description:					
				04	Cancel

16 Click [Add Variable], and enter "m_List" for [Member variable name]. After selecting "Control" for [Category], click the [OK] button.

Add Member Variable	? ×
Member variable <u>n</u> ame: m_List Category: Control Variable type: CListBox	OK Cancel
Description: map to CListBox member	

17 After confirming that the member variable has been added, click the [OK] button.

MFC ClassWizard					? ×
Message Maps	Member Variables	Automation	ActiveX Events	Class Info	
Project: Sample C:\\VC\Sample\ Control_IDs: IDC_LIST1 IDCANCEL IDOK	SampleDlg.h, C:\WC	Class <u>n</u> am CSample Sample Sample Sample Sample CList Box)lg	•	Add Class Add Variable Delete Variable Update Columns Bind All
Description: m	ap to CListBox member				
				ОК	. Cancel

18 Select [Button], and paste it to the dialog box.

🌮 Sample - Microsoft Visual C++ - [Sample.rc - IDD_SAMPLE_DIALOG (Dialog)]		8 ×
Ele Edit View Insert Project Build Layout Tools Window Help	_ 14	a ×
CSampleDig 🔄 IDC_BUTTON1 💽 BN_CLICKED 🔄 🖻 🖉 📩 🚦 🕤		
Sample resources		
Build (Debug) Find in Files 1) Find in Files 2]		×
Press the ALT key to disable guides and boundaries on control placement.	🕂 137,66 📑 114 x 50 F	READ

19 Select a target symbol name from those registered in 'Pro-Studio EX'. (Select the symbol with first-adress for reading.)

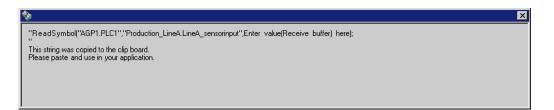
Statt Node Symbol Image: Feature Save Image: Transfer Symbol Image: Feature	EX 2.npx Programming Assist Setting Help					
Group Ungroup Insert Delete Copy Cut Symbol Sheet Symbol Add Delete Check Duplication/List Used Addresses Image: Sheet Signed Sign	» Node »	Symbol ⋗ ≷ Fea	ture 渊 📑 S	ave ᠉ 🆄 Ti	ransfer	Mon Sta
Insert Delete Copy Cut Paste Symbol Data Type Consec utive Device Address No. of Data Consec utive Symbol Data Type Consec utive Device Address No. of Data Consec utive Symbol Data Type Consec utive Device Address No. of Data Consec utive Symbol Data Type Consec utive Device Address No. of Data Consec utive Add Delete D0040_wORD 16Bit(Signed) D0040 3 Check Duplication/List Used Addresses Image: Consection of the time of time of the time of time of the time of time o	Noc	le Name AGP1	Device	Name PLC1		
Symbol Data type utive Device Address No. or Data Col Symbol Dodd_WORD 16Bit(Signed) D0040 3 1	She	et Name Sheet3	🗖 Set it as	a global symbol shee	et.	
Add Delete 1 Add Delete 1 Check Duplication/List Used Addresses 1 1 Global Constant Setting Screen 1 1	Cut Paste	Symbol Da	ata Type Consec utive	Device Address	No. of Data	Commen
Add Delete 1 Check Duplication/List Used Addresses 1 1 Global Constant Setting Screen 1 1	D004	40_WORD 16Bit(Sig	ned)	D0040	3	
Check Duplication/List Used Addresses 1 Global Constant Setting Screen 1 - ▼ Pro-Server EX 1					1	
Global Constant Setting Screen 1 Global Constant Setting Screen 1 → Pro-Server EX 1 → Pro-Server EX 1 → Pro-Server EX 1 → GP3000 Series 1 → GP3000 Series 1 → GP3000 Series 1 → INTERNAL:Sheet1 1 → GP3000 Series 1 → INTERNAL:Sheet2 1 → WinGP 1 → VinGP 1 → LT3000 1					1	
Global Constant Setting Screen 1 Global Constant Setting Screen 1 → Pro-Server EX 1 → Pro-Server EX 1 → Pro-Server EX 1 → GP3000 Series 1 → GP3000 Series 1 → GP3000 Series 1 → INTERNAL:Sheet1 1 → GP3000 Series 1 → INTERNAL:Sheet2 1 → WinGP 1 → VinGP 1 → LT3000 1	ation/List Used Addresses				1	
→ Pro-Server EX 1 → PC1 (192.168.0.1) 1 → INTERNAL:Sheet1 1 → GP3000 Series 1 → GP3000 Series 1 → INTERNAL:Sheet1 1 → INTERNAL:Sheet2 1 → INTERNAL:Sheet2 1 → VinGP 1 → LT3000 1 → GP Series 1					1	
Image: PC1 (192.168.0.1) Image: PC1 (192.168.0.1) Image: Image: PC1 (192.168.0.10) Image: Image: PC1 (192.168.0.100) Image: Image: PC1 (192.168.0.100) Image: Image: PC1 (192.168.0.100) Image: Image: Image: PC1 (192.168.0.100) Image: Image: Image: PC1 (192.168.0.100) Image: Imag	onstant Setting Screen			<u> </u>	1	
Image: PC1 (192.168.0.1) Image: PC1 (192.168.0.1) Image: Image: PC1 (192.168.0.10) Image: Image: PC1 (192.168.0.100) Image: Image: PC1 (192.168.0.100) Image: Image: PC1 (192.168.0.100) Image: Image: Image: PC1 (192.168.0.100) Image: Image: Image: PC1 (192.168.0.100) Image: Imag	ver FX			<u> </u>	1	
→ GP3000 Series 1 → AGP1 (192.168.0.100) 1 → WinGP 1 → UT3000 1 → GP Series 1				<u> </u>	1	
AGP1 (192.168.0.100) HINTERNAL:Sheet2 I HINTERNAL:Sheet2 I FLCT:Sheet3 A Series CPU I				<u> </u>	1	
Image: Hinternal:Sheet2 1 Image: FLC1:Sheet3A Series CPU i 1 Image: WinGP 1				<u> </u>	1	
Image: PLC1:Sheet3 A Series CPU 1 Image: Image				<u> </u>		
WinGP 1 > LT3000 1 > GP Series 1				<u> </u>	1	
GP Series				<u> </u>	1	
				<u> </u>	1	
					1	
	Symbol				1	
					1	
					1	
					1	

 $20 \ \ Select \ [VC++] \ - \ [Read Function] \ from \ "Programming \ Assist" \ on the \ menu.$

_											
2	p P	ro-St	udio E	X 1	23.np:	к					
F	ile	Edit	Tool	Pro	gramm	ing Ass	sist S	Setting	g Help		
		1	C1	,	VB & VE	A 🕨			1		No.
	2	•	Start	1	VC++	×	De	clare	Statement	٢	
				' I	EXCEL	•	Gr	oup st	ructurization		
Г	-S	ymbol		-	_		W	rite Fu	Inction	AGF	P1
			Group			Ungro	Re	ad Fu	nction 📐		
			uroup				_		Snew Name	he	eet3
			Insert			Delete	e				
		Сор	y v	C	Cut Paste		it Paste		Symbol	Τ	Data 1
									0.17080	10	Direinand

Pro-Server EX Reference Manual

The read function is copied to the clipboard.



 $21 \ \ \text{Double-click} \ [\text{Button1}] \ \text{that has been pasted to} \ [\text{Dialog}] \ \text{in Microsoft Visual C++}.$

🐝 Sample - Microsoft Visual C++ - [Samp	erc - IDD SAMPLE DIALOG (Dial	lan		
Eile Edit View Insert Project Build L		~9/1		
¹ 2 2 0 % B 2 - :		**	[
CSampleDIg (All class memb	oers) 💌 💊 CS ampleD Ig	<u>- ■ R</u> +	🕸 🎬 👗 🖠 🕲 🖉	
Sample resources Dialog DiD_ABOUTBOX DiD_SAMPLE_DIA DiD_SAMPLE_DIA Did Di	Sample	Button1	OK Cancel	Con × ▶ ※ Ax abl

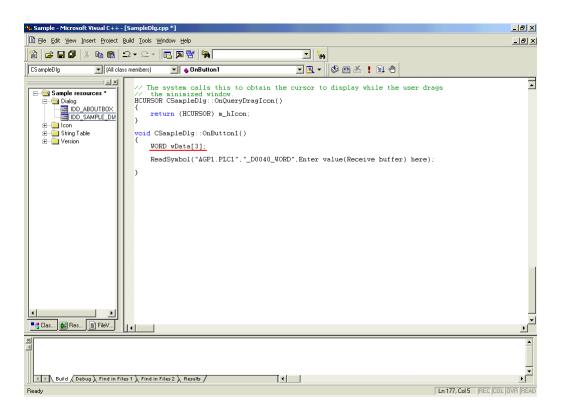
22 Click the [OK] button.

Add Member Function	? ×
Member function <u>n</u> ame:	ОК
On <mark>Button1</mark>	Cancel
Message: BN_CLICKED Object ID: IDC_BUTTON1	

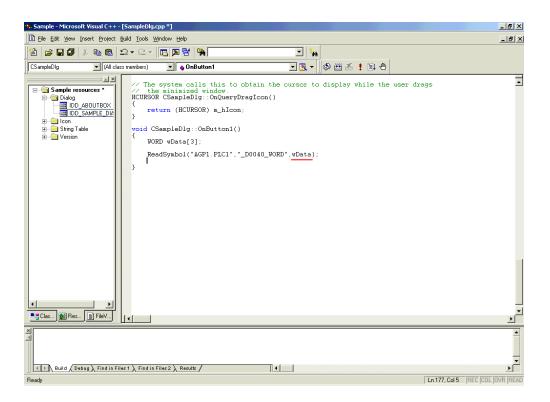
23 Paste the data on the clipboard (read function) into the OnButton1 member function.

Sample - Microsoft Visual C++ - [SampleDig.cpp *]	_ 8 ×
Ele Edit View Insert Broject Build Iools Window Help	_ 8 ×
🏦 😅 🖬 🕼 💼 오 · 오 · 🖪 風 담 🐃 💽 🖕	
CampleDig 💽 (All class members) 🔄 💊 OnButton1 💽 🔍 🗸 🖉 🕮 📥 🗜 🗒 🖑	
<pre>Sample recources' Sample recources' Somple recources' Dialog Dialog Dialog Dialog Dialog Dialog Dialog Dialog Current (HCURSOR) m_hIcon; void CSampleDlg::OnButton1() { ReadSymbol("AGP1.FLC1", "_D0040_WORD",Enter value(Receive buffer) here); }</pre>	1
Clas @ Res] FileV	
Build / Debug & Find in Files 1 & Find in Files 2 & Results /	* •
Ready	

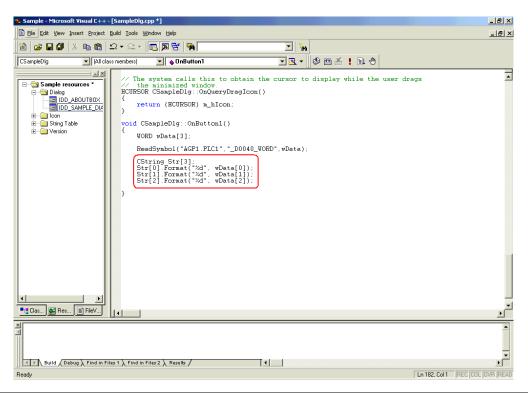
 $24 \ \ \text{Declare the area} \ (\text{Array}) \ \text{to store the read data}.$



25 Specify the first area (wData) to store the read data.



26 To display the read data for three points (wData(0), wData(1) and wData(2)) in the list box, convert the data into Cstring-type string data.



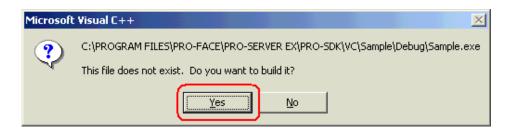
 $27 \ \ \text{The list box (m_List) displays the read data (that has been converted into string data) in sequence.}$

3. Sample - Microsoft Visual C++ - [SampleDlg.cpp]	_ 8 ×
Eie Edit View Insert Project Build Iools Window Help	_ 8 ×
12 ☞ ■ 🖉 🖇 📾 🕲 그 - 그 - 🕞 🗖 🖼 🦄 💽 🔽 🐪	
CSampleDig 🔽 (All class members) 🔽 💊 OnButton1 🔽 🔃 🗸 🖉 🖽 📩 🗜 🔛 🖑	
<pre>% Sample resources</pre>	4
Clas Here Res	▼ ▶
Build (Debug) Find in Files 1) Find in Files 2) Results /	• •
Ready Ln 186, Col 1	REC COL OVR READ

28 Select [Execute Sample.exe] from [Build] on the Microsoft Visual C++ menu.

🤲 Sample - Microsoft Visual C++ ·	- [SampleDlg.cpp]
Eile Edit View Insert Project	Build Iools Window Help
CSampleDlg All ck CSampleDlg All ck CSample resources Dialog Dial	<pre></pre>
	CString Str[3]; Str[0].Format("%d", wData[0]); Str[1].Format("%d", wData[1]); Str[2].Format("%d", wData[2]); m_List.InsertString(0.Str[0]); m_List.InsertString(1.Str[1]); m_List.InsertString(2.Str[2]); }

29 Click the [Yes] button.



30 Click [Button1]. Then, the list box displays the data for three points from the symbol "_D0040_WORD".

Sample 11111 2222 3333	Button1	OK Cancel

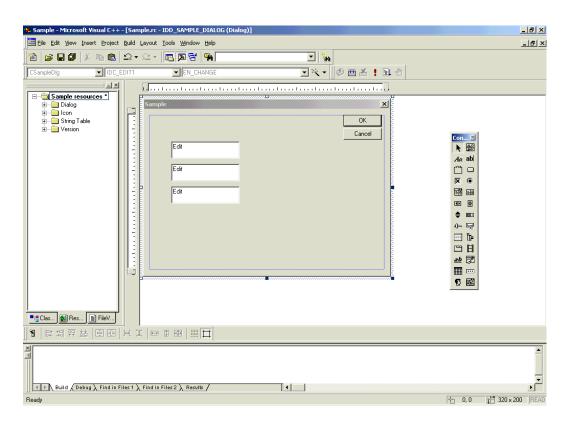
Creating "Writing" application

This section describes the procedure for creating an application that writes the data entered for three points with a click on [Button1].

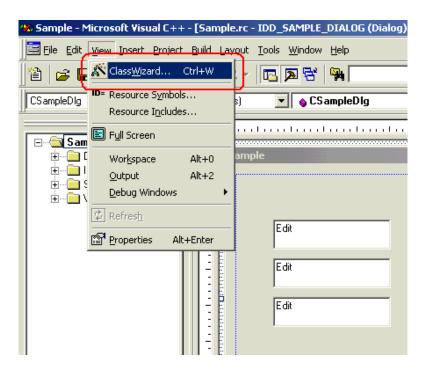
Steps 9 to 11 are the same as those for creating "Reading" application.

🛃 Sample		×
		ОК
		Cancel
1010		
2020	Button1	
3030		

31 Select [EditBox], and paste it to [Dialog]. Paste [Edit Box] for three items.



32 Select [ClassWizard] from [View] on the Microsoft Visual C++ menu.



 $33 \hspace{0.1 cm} \text{Select "IDC_EDIT1" for [Control IDs] in the [Member Variables] tab, and click the [Add Variable] button.}$

MFC ClassWizard	?	<
Message Maps Member Variables Automation ActiveX Events	s Class Info	
Project: Class <u>n</u> ame:	Add Class 🔻	
Sample CSampleDlg	Add Variable	۱
C:\\VC\Sample\SampleDig.h, C:\\VC\Sample\SampleDig.cpp Control/Ds: Tupe Member		1
	Update Columns	l
HDC_EDIT2 IDC_EDIT3 IDCANCEL IDOK	Bind All	
, Description:		
	OK Cancel	

34 Enter "m_Edit1" for [Member Variable], and select "short" for [Variable type]. Then, click the [OK] button. For remaining two [Edit Box], repeat steps 33 and 34. Specify "m_Edit2" and "m_Edit3" for member variables, respectively.

Add Member ¥ariable	? ×
Member variable <u>n</u> ame:	ОК
m_Edit1	Cancel
Category:	
Value	
Variable <u>type:</u>	
short	
Description:	
short with range validation	

35 Click the [OK] button.

MFC ClassWizard					<u>? ×</u>
Message Maps	Member Variables	Automation	ActiveX Events	Class Info	
Project: Sample C:\\VC\Sample\	SampleDig.h, C:\\VC	Class <u>n</u> ame CSampleD	lg	•	Add Class Add Variable
Control <u>I</u> Ds:		Туре	Member		<u>D</u> elete Variable
IDC_EDIT1 IDC_EDIT2 IDC_EDIT3 IDCANCEL IDOK		short short short	m_Edit1 m_Edit2 m,Edit3		Update <u>C</u> olumns <u>B</u> ind All
, Description: sł <u>M</u> inimum Value: Ma <u>x</u> imum Value:	nort with range validatic	n 			
				ОК	Cancel

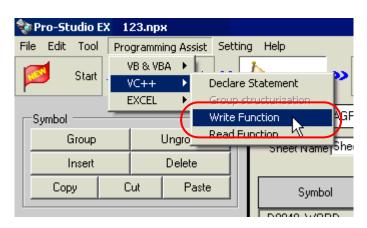
36 Select [Button], and paste it to [Dialog].

* Sample - Microsoft Visual C++ - [Sample.rc - IDD_SAMPLE_DIALOG (Dialog)]	<u>_8</u>
Ele Edit View Insert Broject Build Layout Iools Window Help	_ 5
12 2 3 4 1 1 2 • 2 • 1 3 5 1 1	
CSampleDig 🔽 IDC_BUTTON1 🖳 BN_CLICKED 🔽 🔣 🖉 👗 🚦 🖑	
Sample resources *	Con
X Image: State of the sta	
Press the ALT key to disable guides and boundaries on control placement.	🕂 137, 66 📑 114 x 50 RE

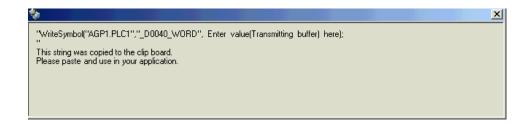
37 Select a target symbol name from those registered in 'Pro-Studio EX'. (Select the symbol with first-adress for writting.)

🂱 Pro-Studio EX 2.npx						
File Edit Tool Programming Assist Settin	ng Help					
Start 🔉 🔪 Node >	녿 Symbol ᆇ 🄾	Feature 🔉 📑	Sa Sa	ive 🔉 🆄 T	ransfer	Monitor Status
Symbol	Node Name AGP1		Device	Name PLC1		
Group Ungroup	Sheet Name Sheet3		Set it as	a global symbol shee	et.	
Insert Delete						
Copy Cut Paste	Symbol	Data Type	Consec utive	Device Address	No. of Data	Comment
Symbol Sheet	_D0040_WORD	16Bit(Signed)		D0040	3	
Add Delete					1	
					1	
Check Duplication/List Used Addresses			<u> </u>		1	<u> </u>
			<u> </u>		1	I
Global Constant Setting Screen	II		<u> </u>		1	
			<u> </u>		1	
🖻 🐺 PC1 (192.168.0.1)		,	<u> </u>		1	I
International #INTERNAL:Sheet1		,	<u> </u>	,	1	I
È - GA AGP1 (192.168.0.100)					1	
#INTERNAL:Sheet2					1	
PLC1:Sheet3 A Series CPU I					1	
► LT3000					1	
> GP Series					1	
Global Symbol			<u> </u>		1	<u> </u>
	I		<u> </u>		1	
					1	
					1	
	•	1	1	1	P.	╵

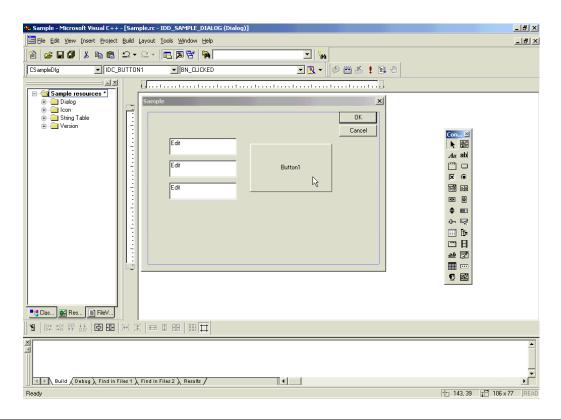
38 Select [Programming Assist] - [VC++] - [Write Function] on the menu.



The write function is copied to the clipboard.



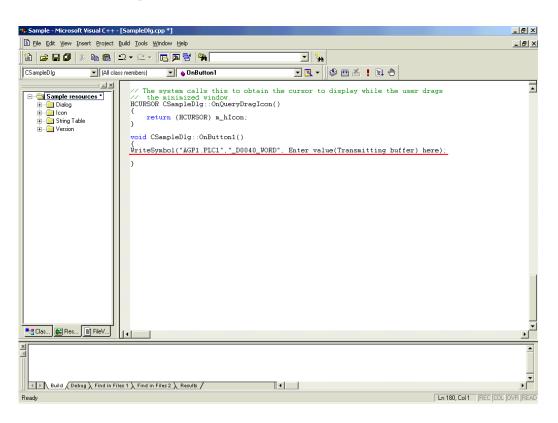
39 Double-click [Button1] that has been pasted to [Dialog] in Microsoft Visual C++.



 $40 \ {\rm Click} \ {\rm the} \ [{\rm OK}] \ {\rm button}.$

Add Member Function	<u>? ×</u>
Member function <u>n</u> ame:	ОК
On <mark>Button1</mark>	Cancel
Message: BN_CLICKED	
Object ID: IDC_BUTTON1	

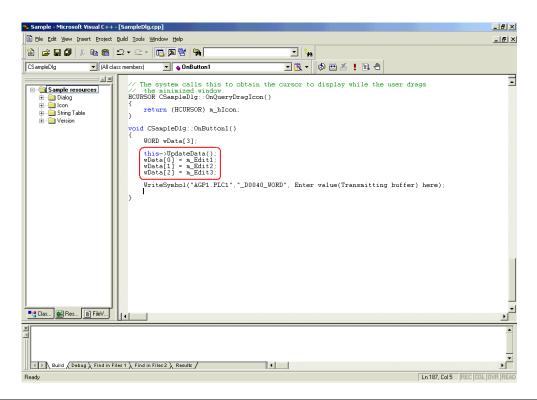
41 Paste the data on the clipboard (write function) into the OnButton1 member function.



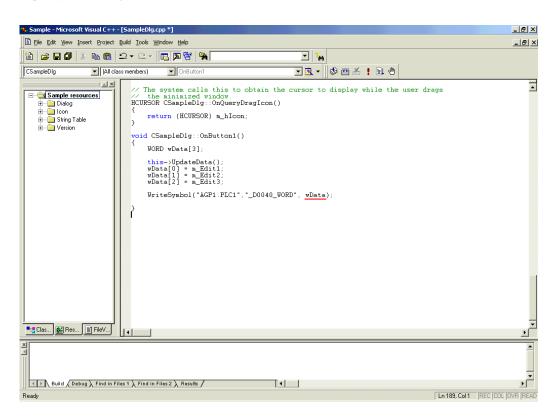
42 Declare the area (Array) to store the write data. For three or more writing points, specify three or more array elements.

🐝 Sample - Microsoft V	risual C++ - [SampleDlg.cpp *]	_ 8 ×
Eile Edit View Inse	rt Project <u>B</u> uild Iools <u>Wi</u> ndow <u>H</u> elp	_ 8 ×
12 😂 🖬 🕼 🐰		
CSampleDig	💌 (All class members) 💽 💊 OnButton1 💽 🐼 💌 🖄 😫 🖄	
	<pre>HCURSOR CSampleD1g::OnQueryDragIcon() { return (HCURSOR) m_hlcon; } void CSampleD1g::OnButton1() { WORD vData[3]; VriteSymbol("AGP1.FLC1", "_D0040_WORD", Enter value(Transmitting buffer) here); } </pre>	
X		
Build Debug	λ Find in Files 1 λ Find in Files 2 λ Results /	•
Ready	Ln 182, Col 1 REC CO	L OVR READ

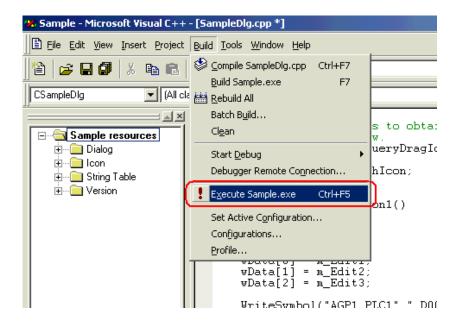
 $43 \ \, \text{Set the data entered in [Edit Box] (for three points) into the array.}$



44 Specify the first alignment (wData) where the written data has been set.



45 Select [Execute Sample.exe] from [Build] on the Microsoft Visual C++ menu.



46 Click the [Yes] button.

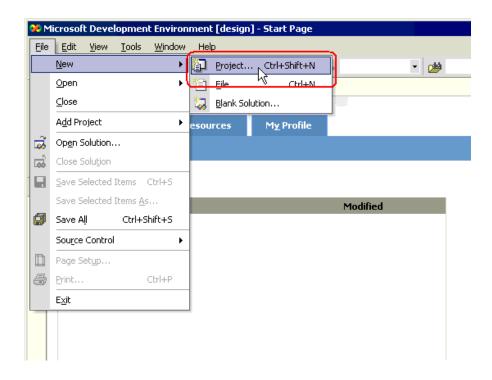
Microsoft	Visual C++
?	C:\PROGRAM FILES\PRO-FACE\PRO-SERVER EX\PRO-SDK\VC\Sample\Debug\Sample.exe
	This file does not exist. Do you want to build it?

47 After entering the values for three points in each [Edit Box], click [Button1]. Then, 'Pro-Server EX' executes the writing of the data for three points from the symbol "_D0040_WORD".

💑 Sample		×
		OK Cancel
1010		
2020	Button1	
3030		

27.11.3 VB .NET Support Function

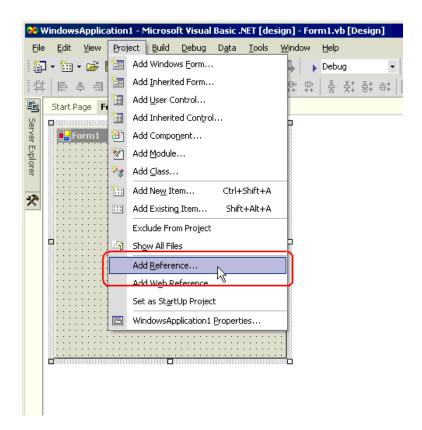
1 Start Microsoft Visual Studio .NET 2003 (or later version), and select [New] - [Project] from the [File] menu.



2 After selecting [Visual Basic Projects] in [Project Types:], select [Windows Application] in [Templates:], and click the [OK] button.

New Project					×
Project Types:		Templates:		0 0 0 0 0 0 0 0 0 0	:
Visual Basic Pro	·	VB	11 VB	<mark>∎V</mark> B ▲	
🕀 🧰 Other Projects	loyment Projects	Windows Application	Class Library	Windows Control Library	
Visual Studio So	Diutions	C.VB			
		Smart Device Application	ASP.NET Web Application	ASP.NET Web Service	·I
A project for creating a	an application with a Windows	user interface			-
Name:	WindowsApplication1				
Location:	C:\Documents and Settings\	Administrator\M	y Documen 💌	Browse	L
Project will be created a	t C:\\My Documents\Visual	Studio Projects\	WindowsApplica	ation1.	_
▼ Mor <u>e</u>		ок	Cancel	Help	

3 Select [Add Reference] from the [Project] menu.



4 Click the [Browse] button.

Component Name	Version	Path		
Accessibility.dll	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra		Select
adodb	7.0.3300.0	C:\Program Files\Microsoft.N	- -	
CustomMarshalers	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra		
envdte	7.0.3300.0	C:\WINNT\Microsoft.NET\Fra		
extensibility	7.0.3300.0	C:\Program Files\Microsoft Vi		
IEExecRemote	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra		
IEHost	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra		
IIEHost	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra		
ISymWrapper	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra		
Managed C# Compiler	7.0.5000.0	C:\WINNT\Microsoft.NET\Fra		
MCppCodeDomParser	0.0.0.0	C:\Program Files\Microsoft Vi	-1	
MCnnCodeDomProvider	7.0.5000.0	C:\Program Files\Microsoft Vi		
cted Components:				
mponent Name	Turce	Source		Remove
importent Name	Туре	Dource		Remove

5 Specify the directory for ProEasyDotNet.dll to be installed, and click the [Open] button. (When installed as standard, the directory is "C:\Program Files\Pro-face\Pro-Server EX\PRO-SDK\DotNet\bin\ProEazyDotNet.dll".)

NOTE	Microsoft .NET Framework 1.1 support for ProEasyDotNet
	Windows Vista or later
	C:\Pro-face\Pro-Server EX\PRO-SDK\DotNet\bin\ProEasyDotNet.dll
	• Windows 2000 / XP / Server 2003
	C:\Program Files\Pro-face\Pro-Server EX\PRO-SDK\DotNet\bin\ProEasyDotNet.dll
•	Microsoft .NET Framework 2.0 support for ProEasyDotNet
	Windows Vista or later
	C:\Pro-face\Pro-Server EX\PRO-SDK\DotNet20\bin\ProEasyDotNet.dll

- Windows 2000 / XP / Server 2003
 - $C:\Program Files\Pro-face\Pro-Server EX\PRO-SDK\DotNet20\bin\ProEasyDotNet.dll$

Select Compone	nt							×
Look in:	🗀 bin			• 🗢 •	E	× 👛 🗉	🛛 🕶 Tools 🕶	
History	ProEasyDo	tNet.dl						
My Projects								
Desktop								
* Favorites								
My Network Places	 File <u>n</u> ame: Files of <u>t</u> ype:	Component	Files (*.dll;*	.tlb;*.olb;*.	ocx;*.exe)	•	; [—	Open Cancel

6 Click the [OK] button.

Component Name	Version	Path 🖌	
Accessibility.dll	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra	Select
adodb	7.0.3300.0	C:\Program Files\Microsoft.N	」 ────
CustomMarshalers	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra	
envdte	7.0.3300.0	C:\WINNT\Microsoft.NET\Fra	
extensibility	7.0.3300.0	C:\Program Files\Microsoft Vi	
IEExecRemote	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra	
IEHost	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra	
IIEHost	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra	
ISymWrapper	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra	
Managed C# Compiler	7.0.5000.0	C:\WINNT\Microsoft.NET\Fra	
MCppCodeDomParser	0.0.0.0	C:\Program Files\Microsoft Vi	-1
MCnnCodeDomProvider	7.0.5000.0	C:\Program Eiles\Microsoft Vi	
lected Components:			
Component Name	Туре	Source	Remov
roEasyDotNet.dll	File	C:\Program Files\Pro-face\Pro	

"ProEasyDotNet.dll" will be registered.

This completes the VB.NET operating environment setup.

The above 1 to 6 steps apply to both reading and writing applications.

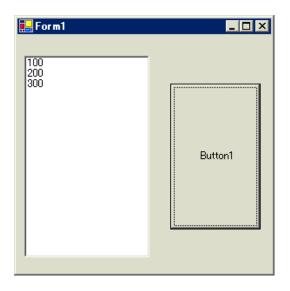
The following procedure varies depending on whether the application is intended for reading or writing, and so is explained individually.

To create a "Reading" application, refer to steps 7 to 19.

To create a "Writing" application, refer to steps 20 to 32.

Creating "Reading" application

This section describes the application that reads and displays data (signed 16 bits) on three items when you click [Button1].



7 After selecting [ListBox] in [Toolbox], clip and paste it onto [Form1].

					o	1	A REAL PROPERTY AND				LER							
_			ation1 -								_	esignj						
Ei	e <u>E</u> dil	: <u>V</u> iew	Project	<u>B</u> uild	Debug	D <u>a</u> ta	<u>T</u> ools	<u>W</u> ind	ow	Help								
1	p - 🏠	। 🕶 🖻	80	X 🖻	B •	0 + CI	- J	- 💷	•	Deb	Jg		•	2				
Ē	t B	홈 레	00 TTO		3	前 昭	에야 것	He 000	œ	4	Ăţ,	음‡ 음	t I	아 목	<u></u> 到一			
	Toolbo		1 8. 8	. Ф ×			1				Δ¥.	UT -	. .		-	-	•	
a	Data	~			Scar	Page	Form1.v	o [ne:	sign _.	r.								
berv	Compo	nents				Form1									1			
er E		ws Forms	;						:::					<u> </u>				
Server Explorer	► P	ointer							0::			::::	:::	:::				
ēř	AL					ListBox1												
_		inkLabel							•••		: : : :	::::	:::	:::				
	ab B													:::				
	_	extBox							•••					: : :				
		1ainMenu																
		heckBox			· C				□: :					:::				
		.adioButto	-						•••				:::	:::				
		iroupBox							•••					:::				
		ictureBox							•••			::::	:::	:::				
		anel							•••									
	·i	anei ataGrid							::			::::	:::	:::				
					Ē		ф О , ф											
		istBox	10		<u></u>													
		heckedLi:																
		omboBox																
		istView																
		reeView																
	Т	abContro	d i															

* If [Toolbox] is not displayed, select [Toolbox] from the [View] menu.

8 After selecting [Button] in [Toolbox], clip and paste it onto [Form1].

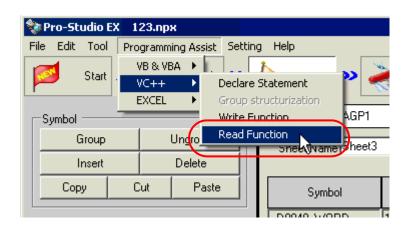
	111- J 8 11 11 4 641	
		ft Visual Basic .NET [design] - Form1.vb [Design]*
Eile		Debug Data Format Tools Window Help
6] • 🛅 • 📂 🔛 🕼 👗 🖻	💼 🗠 🕶 🖓 📲 🖷 🌓 Debug 💽 💌
1	티 튜 후 릐 ㅠ ~ 프 태	금 ‡ 前 昭 ┉ 辉 맞 응 붉 탉 탉 타 •• 욛 •• •• •
5	Toolbox Ψ ×	Start Page Form1.vb [Design]*
_	Data	
rver	Components	Form1
Å	Windows Forms	
Server Explorer	Pointer	ListBox1
	${f A}$ Label	
	A LinkLabel	
	ab Button	
	abl TextBox	
	🛓 MainMenu	Button1
	🔽 CheckBox	
	RadioButton	
	GroupBox	
	💦 PictureBox	
	Panel	
	🚰 DataGrid	
	≡∰ ListBox	
	🚉 CheckedListBox	
	E ComboBox	
	222 ListView	
	हुन् TreeView	
	📩 TabControl	

9 Select a desired read symbol name from the symbols that have been registered in 'Pro-Studio EX'.

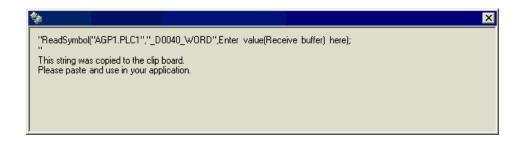
💱 Pro-Studio EX 2.npx						
File Edit Tool Programming Assist Settin	ng Help					
Start >> 🛐 Node >>	🌔 Symbol 🌺 🍝	Feature ン 📑	Sa	ave > 🆄 T	ansfer	Monitor Status
Symbol	Node Name AGP1		Device	Name PLC1		
Group Ungroup	Sheet Name Sheet3		Set it as	a global symbol shee	et.	
Copy Cut Paste	Symbol	Data Type	Consec utive	Device Address	No. of Data	Comment
Symbol Sheet	_D0040_W0RD	16Bit(Signed)		D0040	3	
Add Delete					1	
					1	
Check Duplication/List Used Addresses					1	
			<u> </u>		1	
Global Constant Setting Screen			<u> </u>		1	
Pro-Server EX	II		<u> </u>		1	
🗄 🚚 PC1 (192.168.0.1)	I		<u> </u>		1	III
#INTERNAL:Sheet1			<u> </u>		1	
			<u> </u>		1	
#INTERNAL:Sheet2			<u> </u>		1	
PLC1:Sheet3 A Series CPU I			<u> </u>		1	II
→ WinGP			<u> </u>		1	
 LT3000 GP Series 			<u> </u>	, 	1	
Global Symbol					1	
allocal cylinder			<u> </u>		1	
	li		<u> </u>		1	
					1	
					1	

* The above example shows the symbol for the data type of [16Bit (Signed)] and the data quantity of "3".

10 Select [VC++] - [Read Function] from the [Programming Assist] menu.



The read function is copied to the clipboard.



11 Double-click [Button1] in [Form1], and paste the clipboard data (read function) between the Sub statement and the End Sub statement.

Form1.vb [Design]* Form1.vb*	٩ ٩
🕼 (General)	(Declarations)
□ Public Class Form1	
Inherits System.Windows.Forms.Form	
	nder Ås System.Object, ByVal e Ås System.Eventårgs) Handles Button1.Click WORD", <u>Enter value</u> (Receive buffer) here);
- End Sub	
L End Class	

12 Import the ProEasyDotNet library.

Enter "Imports" at the head of the source code, and select [ProEasyDotNet] from the displayed list box.

Form1.vt	b [Design]* Form1.vb*	4 ⊳
🚺 (Gene	eral) 🔽 🚺 (Declarations)	
□ Puk	ports Di () ATL I () Microsoft WindowsApplexion2 Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click ReadSymbol("ACP1.PLC1"," D0040 WORD",Enter yalue(Receive buffer) here);	
End	End Sub d Class	

13 For the read data storing area, declare a variable "wData".

The array type ("Short" in this example) must conform to the data type of the target symbol. Specify the same data length as the target symbol ("3" in this example).

Form1.vt	[Design]* Form1.vb*			4 ⊳
🖍 (Gen	ral)	•	(Declarations)	
Im	orts ProEasyDotNet			
🖃 Pul	lic Class Form1			
	Inherits System.Windows.Forms.Form			
🕀 W:	ndows Form Designer generated code			
þ	Private Sub Button1_Click(ByVal sender As System.Object	ct,	, ByVal e As System.EventArgs) Handles Button1.Click	
	Dim wData(3) As Short			
	ReadSymbol("AGP1.PLC1","_D0040_WORD",Enter value()	Rece	ceive buffer) here);	
	End Sub			
End	Class			

14 Enter "ProEasy." before "ReadSymbol", and select [ReadDevice16] from the displayed list box.

Form1.vb [Design]* Fo	arm1.vb* ↓
🛷 Button1	▼ Ø Click
Imports ProE	asyDotNet
🖃 Public Class	
Inherits	: System.Windows.Forms.Form
	m Designer generated code
	Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click wData(3) As Short
	wadca(3) XS maar (% ACP1.PLC1"," D0040 WORD", <u>Enter value</u> (Receive buffer) here);
- End Sub	QuedanglacStart
End Class	A GueuingStatusCard
	<u>A</u> g QueuingTblCell
	ReadDevice16
	New ReadDevice16D W →
	Se ReadDevice16M
	** ReadDevice32D

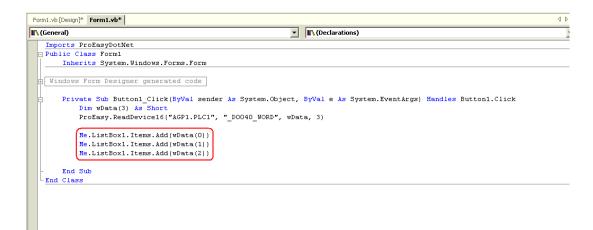
15 Delete "ReadSymbol" from the character string (read function) that has been pasted from the clipboard.

Form1.vb [Design]* Form1.vb*	4 Þ
🕼 (General) 💌	(Declarations)
Imports ProEasyDotNet	
□ Public Class Form1	
Inherits System.Windows.Forms.Form	
 Windows Form Designer generated code Private Sub Button1_Click(ByVal sender As System.Object, Dim wData(3) As Short ProEasy.ReadDevice16("AGP1.PLC1", "_D0040_WORD", Enter 	
- End Sub	
End Class	

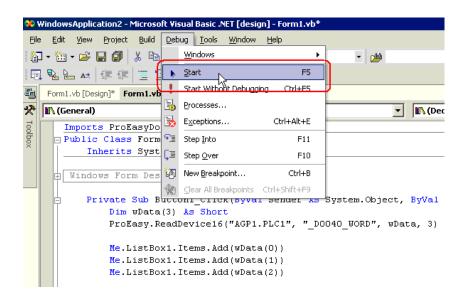
16 Specify a data storing area "wData" as the third argument. Enter ", " (comma) at the end of the third argument, and then enter "3" to specify the length of the target symbol as the fourth argument. After that, delete ";" (semicolon) at the end of the line.

Form1.v	b [Design]* Form1.vb*	4 ⊳
🖍 (Gen	eral) The Construction of	
Im	ports ProEasyDotNet	
🖃 Pul	blic Class Form1	
	Inherits System.Windows.Forms.Form	
🕀 😈	indows Form Designer generated code	
ļ ļ	Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click	
	Dim wData(3) As Short	
	ProEasy.ReadDevice16("AGP1.PLC1","_D0040_WORD", <u>wData,3</u>)	
-	End Sub	
En	d Class	

17 Add the read data on three items (wData(0), wData(1), wData(2)) into [ListBox1] in this order.



18 Select [Start] from the [Debug] menu.



19 If you click [Button1], the target symbol data (three items) are displayed in [ListBox].

🖶 Form 1	
100 200 300	Button1

Creating "Writing" application

This section describes the application that writes data (signed 16 bits) on three items when you click [Button1].

🖳 Form 1	_ 🗆 ×
100	
200	Button1
300	

20 After selecting [TextBox] in [Toolbox], clip and paste three text boxes onto [Form1].

😵 WindowsApplication3 - Micro	osoft Visual Basic .NET [design] - Form1.yb [Design]*	
<u>File E</u> dit <u>V</u> iew <u>P</u> roject <u>B</u> ui	ld <u>D</u> ebug D <u>a</u> ta F <u>o</u> rmat <u>T</u> ools <u>W</u> indow <u>H</u> elp	
🛛 🕶 🕶 🕶 🖼 🕼 🐰	🛍 🛍 🗠 🕶 🖉 - 🚇 - 🖳 Debug 🛛 🔹 🍻	•
[井 타 후 릐 ㅠ 애 쁘	ः 井 印 昭 ┉ 챠 약 후 용 찾 왕 타 凾 區 🖷 🖷 🕇	
Toolbox 4 ×	Start Page Form1.vb [Design]*	
Data		
Components	🛃 Form1 📃 🗖 🗙	
Windows Forms		
🕨 Pointer		
A Label	TextBox1	
A LinkLabel		
<u>ab</u> Button	TextBox2	
abl TextBox		
🛓 MainMenu	TextBox3	
CheckBox		
 RadioButton 		
[^{xv}] GroupBox		
🔊 PictureBox		
Panel		
🚰 DataGrid		
토령 ListBox		

* If [Toolbox] is not displayed, select [Toolbox] from the [View] menu.

21 After selecting [Button] in [Toolbox], clip and paste it onto [Form1].

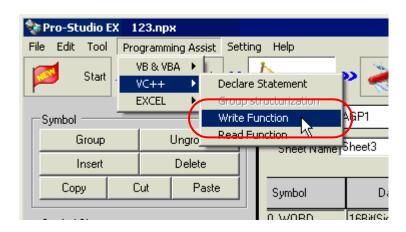
🚧 WindowsApplication3 - Micr	osoft ¥isual Basic .NET [design] - Form1.yb [Design]*	
<u>File E</u> dit <u>V</u> iew <u>P</u> roject <u>B</u> u	ld <u>D</u> ebug D <u>a</u> ta <u>T</u> ools <u>W</u> indow <u>H</u> elp	
🛅 • 🛅 • 🛩 🖬 🕼 🐰	🛍 💼 🗠 र 🖓 र 🜉 🖡 Debug 🛛 🗸 🍅	-
ᄨᆥᆙᆥᆃᆁᆝᅘᄵᄜᆖ	昂 寺 印 昭 마 ಭ 맞 많 울 찾 왕 학 팬 图 🖷 📲 🖡	
Toolbox 🛛 🕈 🗙	Start Page Form1.vb [Design]*	
Data		
Components	Form1	
Windows Forms		
🕨 Pointer	······································	
${f A}$ Label	:: TextBox1	
A LinkLabel		
ab Button	TextBox2	
abi TextBox		
🛓 MainMenu	TextBox3	
CheckBox		
RadioButton		
GroupBox		
😹 PictureBox		
Panel		
🛐 DataGrid		
≡∰ ListBox		

22 Select a desired write symbol name from the symbols that have been registered in 'Pro-Studio EX'. (Select the first writing area.)

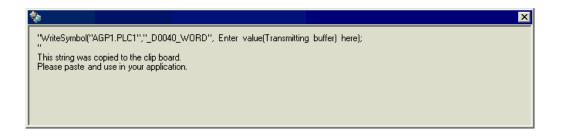
💱 Pro-Studio EX 2.npx						
File Edit Tool Programming Assist Sett	ing Help					
Start >> 🛐 Node >>	🍐 Symbol ≫ 🍦	Feature ⋗ 📑	Save	>> 🔖 Tr	ansfer	Monitor Status
Symbol	Node Name AGP1		Device Nar	me PLC1		
Group Ungroup	Sheet Name Sheet3		Set it as a gli	lobal symbol shee	t.	
Copy Cut Paste	Symbol	Data Type	Consec D	evice Address)	No. of Data	Comment
Symbol Sheet	_D0040_WORD	16Bit(Signed)	DO	040	3	
Add Delete					1	
					1	
Check Duplication/List Used Addresses					1	
					1	
Global Constant Setting Screen	I		<u> </u> _		1	
E▼ Pro-Server EX					1	
Ė-🐙 PC1 (192.168.0.1)					1	
			<u> </u> -		1	
⊟▼ GP3000 Series			<u> </u>		1	
#INTERNAL:Sheet2		·			1	
PLC1:Sheet3 A Series CPU I					1	
					1	
→ GP Series					1	
• Global Symbol					1	
					1	
	I	ļ			1	
		ļ			1	
		I			1	

* The above example shows the symbol for the data type of [16Bit (Signed)] and the data quantity of "3".

23 Select [VC++] - [Write Function] from the [Programming Assist] menu.



The write function is copied to the clipboard.



24 Double-click [Button1] in [Form1], and paste the clipboard data (write function) below the [Button1_Click] method ("Private Sub Button1_Click..." character string).

Start Page Form1.vb [Design]* Form1.vb*		4
🕼 (General)	💌 🖍 (Declara	ations)
- Public Class Form1		
Inherits System.Windows.Forms	Drm	
⊕ Windows Form Designer generated	de	
WriteSymbol ("AGP1.PLC1","	sender As System.Object, ByVal e 3 0040_WORD", Enter value(Transmittir	<pre>As System.EventArgs) Handles Button1.Click ng buffer) here);</pre>
- End Sub		
LEnd Class		

25 Import the ProEasyDotNet library.

Enter "Imports" at the head of the source code, and select [ProEasyDotNet] from the displayed list box.

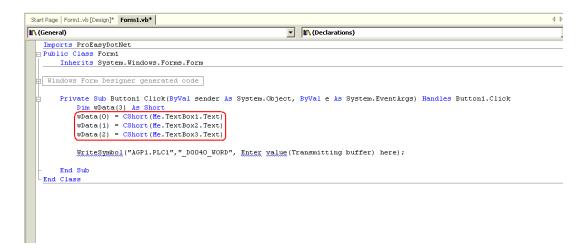
Start Page Form1.vb [Design]* Form1.vb	4 b
🖍 (General)	✓ IN (Declarations)
Wint Image: Construction of the construction o	vs.Forms.Form perated code Slick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click .PLC1", "_DO040_WORD", Enter value(Transmitting buffer) here);
- End Sub End Class	

26 For the write data storing area, declare a variable "wData".

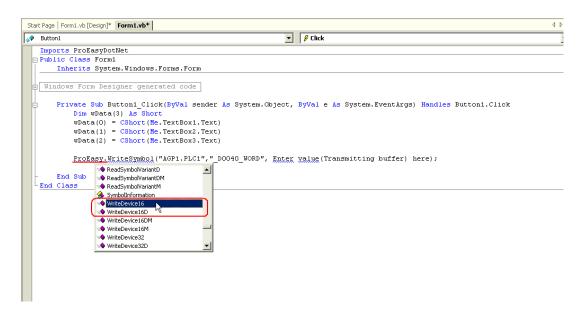
The array type ("Short" in this example) must conform to the data type of the target symbol. Specify the same data length as the target symbol ("3" in this example).

Start Page Form1	vb [Design]*	Form1.vb*					4 ⊳
🖍 (General)				-	•	🕼 (Declarations)	
Imports P	roEasyDotl	let					
🖃 Public Cl	ass Form1						
Inher	its System	.Windows.For	ms.Form				
- Priva	te Sub Bu im wData(:) As Short	ByVal sender			ByVal e Ås System.Eventårgs) Handles Buttonl.Click ansmitting buffer) here);	
- End S	ub						
End Class							

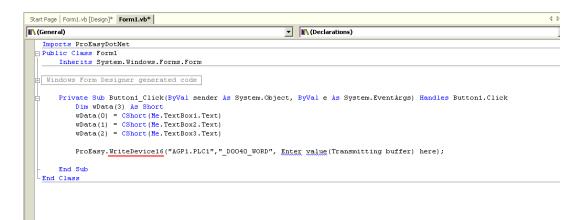
27 Set the data to be entered in [TextBox1] to [TextBox3] in the array.



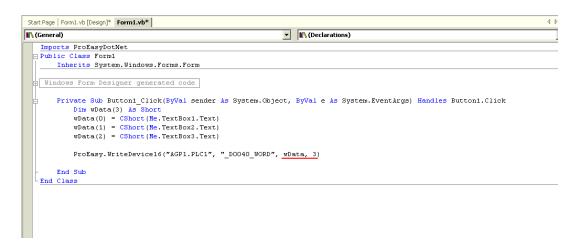
28 Enter "ProEasy." before "WriteSymbol", and select [WriteDevice16] from the displayed list box.



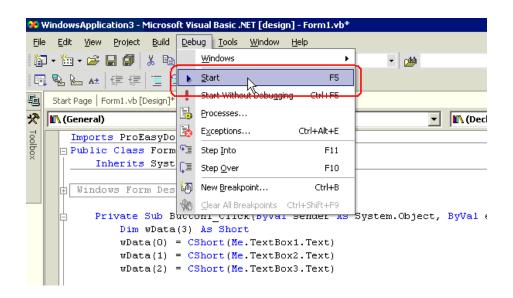
29 Delete "WriteSymbol" from the character string (write function) that has been pasted from the clipboard.



30 Specify a data storing area "wData" as the third argument. Enter "," (comma) at the end of the third argument, and then enter "3" to specify the length of the target symbol as the fourth argument. After that, delete ";" (semicolon) at the end of the line.



31 Select [Start] from the [Debug] menu.



32 Immediately after startup, a character string "TextBox*" is displayed in [TextBox].

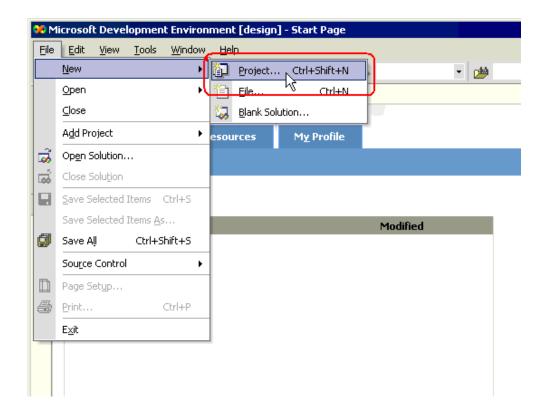
🛃 Form1	
TextBox1	
TextBox2	Button1
TextBox3	

After entering the write data (three items) in [TextBox], click [Button1]. Then, the data will be written into the area specified with the symbol.

🖳 Form1	_ 🗆 ×
100	
200	Button1
300	
	<u>,</u>

27.11.4 C# Support Function

1 Start Microsoft Visual Studio .NET 2003 (or later version), and select [New] - [Project] from the [File] menu.



2 After selecting [Visual C# Projects] in [Project Types:], select [Windows Application] in [Templates:], and click the [OK] button.

New Project					×
Project Types:	ojects	Templates:			8-8- 8-8- 8-8-
Visual C# Proj		<mark>ہے</mark> Windows	Class Library	علیہ Windows	
⊕ ☐ Other Projects ☐ Visual Studio S		Application		Control Library	
		Smart Device			
	b is at in t	Application	Application	Service	•
A project for creating	an application with a Window	s user interface			
Name:	WindowsApplication1				
Location:	C:\Documents and Settings	s\Administrator\M	y Documen 💌	Browse	
Project will be created	at C:\\My Documents\Visua	al Studio Projects\	WindowsApplica	ation1.	
▼ Mor <u>e</u>		ок	Cancel	Help	

3 Select [Add Reference] from the [Project] menu.

<mark>98</mark> V	Vindow	sApplica	ation	14 - M	Aicrosa	ft Visual	C# .NET	「 [desig	n] - F	orm	l.cs [Desi	gn]	
Eile	<u>E</u> dit	<u>V</u> iew	Proj	ject	<u>B</u> uild	<u>D</u> ebug	D <u>a</u> ta	<u>T</u> ools	Wind	low	Help			
8	- 🏪	- 🗃 [1	Add	d Windov	ws <u>F</u> orm			8	•	Deb	Jg		-
1		후 릐	1	Ado	l <u>I</u> nherit	ed Form			ille		움	š ‡	믕;	et
5	Start F	Page Fr	8 <u>99</u>	Add	<u>l U</u> ser ⊂	ontrol			-					. •
*			8 9 3	Add	d Inherit	ed Con <u>t</u> ro	l							
		orm1	8	Add	d Compo	nent								
Toolbox			***	Ado	d <u>⊂</u> lass…									
×	• • •		****	Ado	d Ne <u>w</u> It	em	Ctrl+:	Shift+A						
	· · · · · · ·		:::	Ado	d Existing	g Item	Shift	+Alt+A						
	· · · · · · ·			Exc	lude Fro	m Project								
	••••		a	Shg	w All Fil	es								
		: : : : : (Ado	d Refere	nce	Ν		-					
	•••	:::: !		Ado	- J Web R	eference.	- k }							
	•••					tUp Projec								
	••••		c	Win	- ndowsAr	plication4	Properti	es						
	· · · · · · ·													
	••••													
									Ö					

4 Click the [Browse] button.

Component Name	Version	Path		
Accessibility.dll	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra		Select
adodb	7.0.3300.0	C:\Program Files\Microsoft.N		
CustomMarshalers	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra		
envdte	7.0.3300.0	C:\WINNT\Microsoft.NET\Fra		
extensibility	7.0.3300.0	C:\Program Files\Microsoft Vi		
IEExecRemote	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra		
IEHost	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra		
IIEHost	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra		
ISymWrapper	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra		
Managed C# Compiler	7.0.5000.0	C:\WINNT\Microsoft.NET\Fra		
MCppCodeDomParser	0.0.0.0	C:\Program Files\Microsoft Vi		
MCnnCodeDomProvider	7.0.5000.0	C:\Program Files\Microsoft Vi	<u> </u>	
ected Components:				
omponent Name	Туре	Source		Remove

5 Specify the directory for ProEasyDotNet.dll to be installed, and click the [Open] button. (When installed as standard, the directory is "C:\Program Files\Pro-face\Pro-Server EX\PRO-SDK\DotNet\bin\ProEazyDotNet.dll".)

NOTE • Microsoft .NET Framework 1.1 support for ProEasyDotNet
Windows Vista or later
C:\Pro-face\Pro-Server EX\PRO-SDK\DotNet\bin\ProEasyDotNet.dll
• Windows 2000 / XP / Server 2003
C:\Program Files\Pro-face\Pro-Server EX\PRO-SDK\DotNet\bin\ProEasyDotNet.dll
 Microsoft .NET Framework 2.0 support for ProEasyDotNet
Windows Vista or later

- $C:\Pro-face\Pro-Server\ EX\PRO-SDK\DotNet20\bin\ProEasyDotNet.dll$
- Windows 2000 / XP / Server 2003
 - $C:\Program Files\Pro-face\Pro-Server \ EX\PRO-SDK\DotNet20\bin\ProEasyDotNet.dll$

Se	elect Compone	nt 📉	3
	Look in:	💼 bin 💽 🤄 🖛 💼 🧔 🗙 🖄 🏢 - Tools -	
		ProEasyDotNet.dll	
	My Projects		
	Desktop		
	* Favorites		
	My Network Places	File name: Image: Open Files of type: Component Files (*.dll;*.tlb;*.olb;*.ocx;*.exe)	

6 Click the [OK] button.

Component Name	Version	Path 🔺	Browse
	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra	Select
Accessibility.dll adodb	7.0.3300.0	C:\WINN (MICrosoft.NET(Fra	
CustomMarshalers	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra	
envdte	7.0.3300.0	C:\WINNT\Microsoft.NET\Fra	
extensibility	7.0.3300.0	C:\Program Files\Microsoft Vi	
IEExecRemote	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra	
IEHost		C:\WINNT\Microsoft.NET\Fra	
IEHost	1.0.5000.0		
	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra	
ISymWrapper Managed C# Counting	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra	
Managed C# Compiler	7.0.5000.0	C:\WINNT\Microsoft.NET\Fra	
MCppCodeDomParser	0.0.0.0	C:\Program Files\Microsoft Vi	
	7.0.5000.0	C:)Program Eiles)Microsoft VI	
leated Companyates			
lected Components:	[[5	D
Component Name	Туре	Source	Remove
roEasyDotNet.dll	File	C:\Program Files\Pro-face\Pro	

"ProEasyDotNet.dll" will be registered.

This completes the C# operating environment setup.

The above 1 to 6 steps apply to both reading and writing applications.

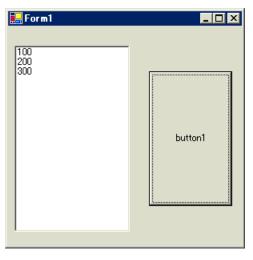
The following procedure varies depending on whether the application is intended for reading or writing, and so is explained individually.

To create a "Reading" application, refer to steps 7 to 19.

To create a "Writing" application, refer to steps 20 to 32.

Creating "Reading" application

This section describes the application that reads and displays data (signed 16 bits) on three items when you click [button1].



💔 WindowsApplication4 - Microsoft Visual C# .NET [design] - Form1.cs [Design]* Eile Edit View Project Build Debug Data Tools Window Help 🎦 • 🛅 • 🚔 🔒 🕼 👗 🛍 🋍 い • ロ • 🚚 • 🖳 🕟 Debug - 🍅 | 拝 | 悒 卒 引 | 雨 st 四 昂 拱 朝 昭 | ゆ 珠 맞 忠 | そ 封 計 む 國 图 | 9 | 9 | 1 Ψ× Toolbox Start Page Form1.cs [Design]* Data Components <u>- 🗆 ×</u> 🖳 Form1 . Windows Forms listBox1 Reinter ${f A}$ Label A LinkLabel ab Button sbl TextBox 🛓 MainMenu 🔽 CheckBox RadioButton GroupBox 😹 PictureBox 🗌 Panel 🛐 DataGrid Ц 트럼 ListBox CheckedListBox ComboBox 222 ListView TreeView 📋 TabControl

7 After selecting [ListBox] in [Toolbox], clip and paste it onto [Form1].

* If [Toolbox] is not displayed, select [Toolbox] from the [View] menu.

8 After selecting [Button] in [Toolbox], clip and paste it onto [Form1].

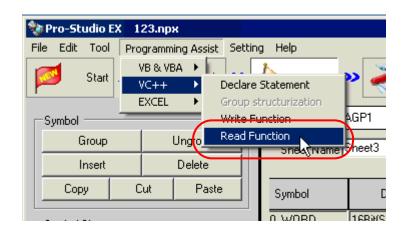
😵 WindowsApplication4 - Micro	soft Visual C# .NET [design] - Form1.cs [Design]*
<u>File E</u> dit <u>V</u> iew <u>P</u> roject <u>B</u> ui	
🛛 🖓 • 🛅 • 😂 🖬 🕼 🕺	🛍 💼 ၊ ၊ ၊ ၊ ၊ ၊ 🕮 - 🖳 🖡 🖡 Debug 🔹 💌
[井] [후 릐] ㅠ ~ 쁘	등 禁 때 뗣 · ··· 갸 꺄 꺄 울 찾 왉 라 팬 톤 嘴 🖷 .
Toolbox 4 ×	Start Page Form1.cs [Design]*
Data	
Components	E Form1
Windows Forms	
🕨 Pointer	listBox1
A Label	
A LinkLabel	
ab Button	
abl TextBox	
🛓 MainMenu	button1
CheckBox	
RadioButton	
GroupBox	
RictureBox	
Panel	
🚰 DataGrid	
≡∉ ListBox	
🔛 CheckedListBox	
ComboBox	
222 ListView	
ing TreeView	
TabControl	

9 Select a desired read symbol name from the symbols that have been registered in 'Pro-Studio EX'.

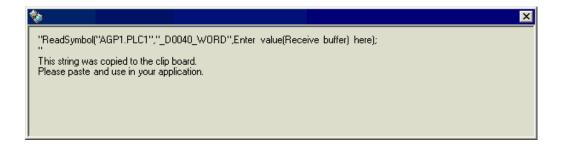
💱 Pro-Studio EX 2.npx						
File Edit Tool Programming Assist Settin	g Help					
Start 🍛 🟹 Node ン	녿 Symbol ⋗ 葇	Feature ⋗ 📑	Sa Sa	ave 渊 🆄 Ti	ransfer	Monitor Status
Symbol	Node Name AGP1		Device	Name PLC1		
Group Ungroup	Sheet Name Sheet3		Set it as	a global symbol shee	et.	
			-			
Copy Cut Paste	Symbol	Data Type	Consec utive	Device Address	No. of Data	Comment
Symbol Sheet	_D0040_WORD	16Bit(Signed)		D0040	3	
Add Delete					1	
					1	
Check Duplication/List Used Addresses		<u> </u>			1	
		<u> </u>			1	
Global Constant Setting Screen					1	
Pro-Server EX					1	
				ļ	1	
INTERNAL:Sheet1					1	
GP3000 Series				<u> </u>	1	
GRAND AGP1 (192.168.0.100)				ļ	1	
PLC1:Sheet3 A Series CPU [1	
→ WinGP					1	
• LT3000					1	
• GP Series			<u> </u>		1	
Global Symbol					1	
					1	
					1	
			<u> </u>		1	
	1	1	1	1	11	

* The above example shows the symbol for the data type of [16Bit (Signed)] and the data quantity of "3".

10 Select [VC++] - [Read Function] from the [Programming Assist] menu.



The read function is copied to the clipboard.

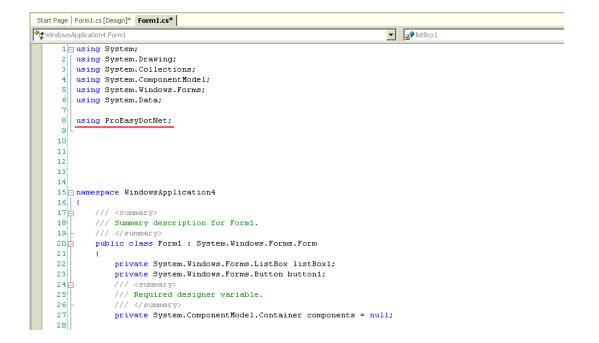


11 Double-click [button1] in [Form1], and paste the clipboard data (read function) below the [button1_Click] method ("private void button1_Click..." character string).

```
Start Page | Form1.cs [Design]* Form1.cs*
WindowsApplication4.Form1
                                                                              ▼ ∰$button1_Click(object sender,System.EventArgs e)
             public class Form1 : System.Windows.Forms.Form
     14
             {
     15
                 private System.Windows.Forms.ListBox listBox1;
     16
                 private System.Windows.Forms.Button button1;
     17
                 /// <summary>
                 /// Required designer variable.
     18
     19
                 /// </summary>
     20
                 private System.ComponentModel.Container components = null;
     21
     22 🕁
                 public Form1()...
     34
                 /// <summarv>
                 /// Clean up any resources being used.
     35
     36
                 /// </summary>
     37 🖨
                 protected override void Dispose( bool disposing )...
     49 🕁
                 Windows Form Designer generated code
     88
     89¢
                 /// <summarv>
                 ///% \label{eq:linear} The main entry point for the application.
     90
     91
                 /// </summary>
     92
                 [STAThread]
     93 þ
                 static void Main()
     94
                 {
     95
                      Application.Run(new Form1());
     96
                 }
     97
     98
                 private void button1_Click(object sender, System.EventArgs e)
    99
                 - {
                      ReadSymbol("AGP1.PLC1","_D0040_WORD",Enter value(Receive buffer) here);
    100
    101
    102
                 }
    103
              }
    104 }
    105
    106
    107
    108
    109
```

12 Describe the ProEasyDotNet directive.

Enter "using ProEasyDotNet;" at the bottom of the lines that state "using..." at the head of the source code.

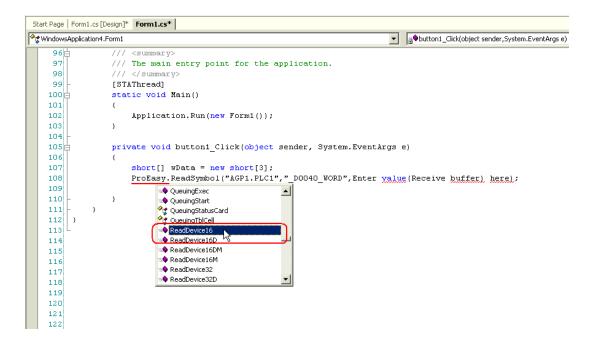


13 For the read data storing area, declare a variable "wData".

The array type ("Short" in this example) must conform to the data type of the target symbol. Specify the same data length as the target symbol ("3" in this example).

```
Start Page | Form1.cs [Design]* Form1.cs*
Vir WindowsApplication4.Form1
                                                                                ▼ ∰$button1_Click(object sender,System.EventArgs e)
     96
                  /// <summary>
     97
                 /// The main entry point for the application.
     98
                 /// </summary>
     99
                 [STAThread]
    1000
                 static void Main()
    101
                 {
    102
                       Application.Run(new Form1());
    103
                 }
    104
    105
                  private void button1_Click(object sender, System.EventArgs e)
                  {
    106
    107
                       short[] wData = new short[3];
                       ReadSymbol("AGP1.PLC1","_D0040_WORD",Enter value(Receive buffer) here);
    108
    109
    110
                  }
    111
              }
    112 }
    113
    114
    115
    116
    117
    118
```

14 Enter "ProEasy." before "ReadSymbol", and select [ReadDevice16] from the displayed list box.



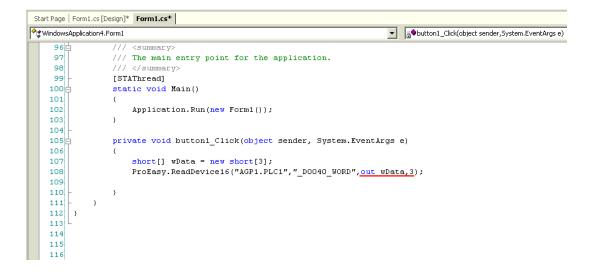
15 Delete "ReadSymbol" from the character string (read function) that has been pasted from the clipboard.

```
Start Page | Form1.cs [Design]* Form1.cs*
✤ WindowsApplication4.Form1

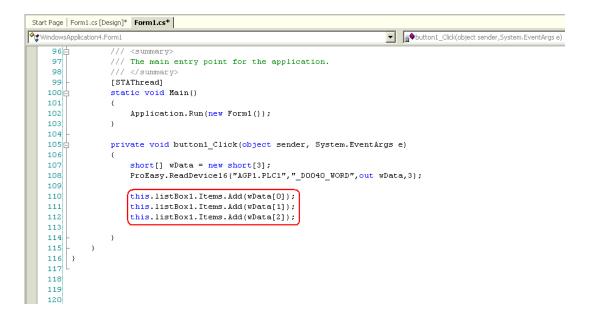
    System.EventArgs e)

     96
                      <summary>
                  ///% \label{eq:linear} The main entry point for the application.
     97
     98
                  /// </summary:
     99
                  [STAThread]
    100
                  static void Main()
    101
                  {
    102
                      Application.Run(new Form1());
    103
                  }
    104
                  private void button1_Click(object sender, System.EventArgs e)
    105
    106
                  {
    107
                      short[] wData = new short[3];
    108
                      ProEasy.ReadDevice16("AGP1.PLC1","_D0040_WORD",Enter value(Receive buffer) here);
    109
    110
                  3
    111
             }
    112
         }
    113
    114
    115
    116
```

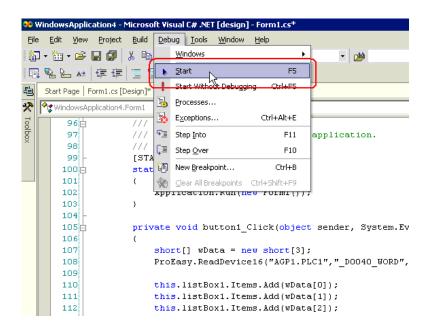
16 Specify a data storing area "wData" with the reference modifier (out), as the third argument. Enter "," (comma) at the end of the third argument, and then enter "3" to specify the length of the target symbol as the fourth argument.



17 Add the read data on three items (wData[0], wData[1], wData[2]) into [listBox1] in this order.



18 Select [Start] from the [Debug] menu.



19 If you click [button1], the target symbol data (three items) are displayed in [ListBox].

🖳 Form 1	
100 200 300	button1

Creating "Writing" application

This section describes the application that writes data (signed 16 bits) on three items when you click [button1].

🖳 Form 1	
100	
200	button1
300	

20 After selecting [TextBox] in [Toolbox], clip and paste three text boxes onto [Form1].

😵 WindowsApplication5 - Micro	osoft Visual C# .NET [design] - Form1.cs [Design]*	
<u>File E</u> dit <u>V</u> iew <u>P</u> roject <u>B</u> ui	ld <u>D</u> ebug D <u>a</u> ta <u>T</u> ools <u>W</u> indow <u>H</u> elp	
🎦 • 🛅 • 🖻 🔒 🦚 🐰	🛍 💼 🗠 - 🖓 - 🚚 - 📕 🕞 Debug 🛛 - 🍅	•
[井] [말 후 릐] (편 애 프	昂 寺 印 密 ┉ 琌 약 약 유 봉 황 황 ● ● 图 📲 🖷 🕇	
Toolbox 🛛 🕈 🗙	Start Page Form1.cs [Design]*	
Data		
Components	Form1	
Windows Forms		
Rointer		
A Label	textBox1	
A LinkLabel	· · · · · · · · · · · · · · · · · · ·	
ab Button	LextBox2	
abl TextBox		
🛓 MainMenu	textBox3	
🔽 CheckBox		
RadioButton		
GroupBox		
🚜 PictureBox		
Panel		
词 DataGrid		
=봄 ListBox		

* If [Toolbox] is not displayed, select [Toolbox] from the [View] menu.

21 After selecting [Button] in [Toolbox], clip and paste it onto [Form1].

Image: Start Page Image: Start Page <th>WindowsApplication5 - Mici</th> <th>osoft Visual C# .NET [design] - Form1.cs [Design]*</th>	WindowsApplication5 - Mici	osoft Visual C# .NET [design] - Form1.cs [Design]*
다 나 나 나 나 나 나 나 나 나 나 나 나 나 나 나 나 나 나 나	<u>File Edit Y</u> iew Project <u>B</u> u	ild <u>D</u> ebug D <u>a</u> ta <u>T</u> ools <u>W</u> indow <u>H</u> elp
ioolbox ♥ × aata ionponents Windows Forms Image: Pointer A Label A LinkLabel ionitication ionitication <t< th=""><th>🛐 • 🛅 • 🚔 🔛 🕼 👗</th><th>🛍 💼 🗠 - 여 - 🚚 - 📕 🕞 Debug 🛛 - 🍅</th></t<>	🛐 • 🛅 • 🚔 🔛 🕼 👗	🛍 💼 🗠 - 여 - 🚚 - 📕 🕞 Debug 🛛 - 🍅
ata components Windows Forms ▲ Pointer A Label A LinkLabel ab Button ab TextBox MainMenu ✓ CheckBox ④ RadioButton ④ Pointer ■ DataGrid	# 타 후 릐 ㅠ ~ 므	🗟 井 前 昭 ┉ ಭ 밖 밖 울 찾 밝 타 팬 區 ங 🖷 .
Somponents Windows Forms ▶ Pointer ▲ Label ▲ LinkLabel >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>	Toolbox 🛛 🕈 🗙	Start Page Form1.cs [Design]*
Vindows Forms ▶ Pointer A Label A LinkLabel Button Butt	Data	
Pointer A Label A LinkLabel abl tinkLabel button1 button1 c CheckBox c CheckBox c CheckBox Pointer Pointer add TextBox button1 c CheckBox c CheckBox Pointer panel p DataGrid	Iomponents	Form1
A Label A LinkLabel a) Button a) TextBox button1 ↓ textBox2 button1 ↓ textBox3 ↓ CheckBox ↓ CheckBox	Windows Forms	
A Label A LinkLabel a) Button a) TextBox button1 ↓ textBox2 button1 ↓ textBox3 ↓ checkBox ↓	Pointer	
A LinkLabel ■ abl LinkLabel ■ abl Button ■ abl TextBox ■ MainMenu ■ ✓ CheckBox ■ ⑥ RadioButton ■ ③ PictureBox ■ Panel ■ ⑤ DataGrid ■	•	
Button abi TextBox MainMenu textBox3 textBox3 CheckBox GroupBox PictureBox Panel DataGrid		
ati TextBox MainMenu CheckBox RadioButton GroupBox Panel DataGrid		· · · · · · · · · · · · · · · · · · ·
MainMenu CheckBox RadioButton GroupBox PictureBox Panel DataGrid		
CheckBox		
RadioButton Image: Constraint of the second of the secon		· · · · ·
Image: Second	CheckBox	
PictureBox Panel JoataGrid	RadioButton	
Panel DataGrid	GroupBox	
J DataGrid	RictureBox	
	Panel	
	Tunia Tunia International In International International Internation International International International International International International International International Int	
	≡∰ ListBox	

22 Select a desired write symbol name from the symbols that have been registered in 'Pro-Studio EX'. (Select the first writing area.)

💱 Pro-Studio EX 2.npx					- 🗆 ×
File Edit Tool Programming Assist Settir	ng Help				
Start 🔉 🟹 Node 🔉	🍐 Symbol 🌺	≷ Feature ン 📑	📔 Save ≫ 🆄	Transfer	Monitor Status
Symbol	Node Name AGP1		Device Name PLC1		
Group Ungroup	Sheet Name Sheet3		Set it as a global symbol sł	neet.	
Insert Delete	,				
Copy Cut Paste	Symbol	Data Type	Consec utive Device Addres	s No. of Data	Comment
Symbol Sheet	_D0040_WORD	16Bit(Signed)	D0040	3	
Add Delete				1	
				1	
Check Duplication/List Used Addresses				1	
				1	
Global Constant Setting Screen				1	
		<u> </u>		1	
E ₽ PC1 (192.168.0.1)		<u></u>		1	
#INTERNAL:Sheet1		<u> </u>	<u> </u>	1	
□ - ▼ GP3000 Series		<u> </u>		1	
E - G AGP1 (192.168.0.100)				1	
#INTERNAL:Sheet2		<u> </u>		1	
→ WinGP		<u> </u>		1	
• LT3000		<u></u>		1	
• GP Series		<u></u>		1	
Global Symbol		<u></u>		1	
				1	
				1	
				1	
				1	L

* The above example shows the symbol for the data type of [16Bit (Signed)] and the data quantity of "3".

23 Select [VC++] - [Write Function] from the [Programming Assist] menu.

A					
💱 Pro-Studio E>	< 123.np	×			
File Edit Tool	Programm	ing Assist S	etting	Help	
1	VB & VE	3A 🕨 📜 🔒	;	ĸ,	. · · · ·
Start Start	VC++	🕨 🕨 Dec	lare S	itatement	~
	EXCEL	Gro	up str	ucturization	
Symbol		Wri	te Fur	nction	GP1
Group		Ungro <u>Rea</u>	ad Fur	nction k	1 10
				Sheet Name	sheets
Insert		Delete			
Сору	Cut	Paste		Symbol	D
			-		
-Currhal Charak				0 WORD	16Bit(Si

The write function is copied to the clipboard.

٠	×
	WriteSymbol("AGP1.PLC1","_D0040_W0RD", Enter value(Transmitting buffer) here);
F	his string was copied to the clip board. lease paste and use in your application.

24 Double-click [button1] in [Form1], and paste the clipboard data (write function) below the [button1_Click] method ("private void button1_Click..." character string).

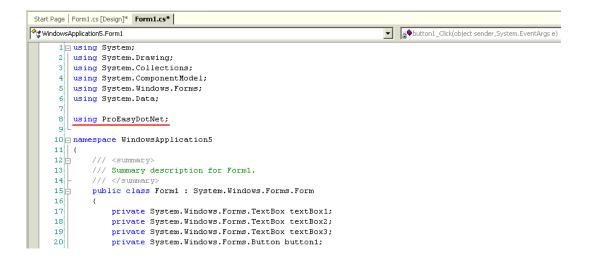
```
Start Page | Form1.cs [Design]* Form1.cs*
WindowsApplication5.Form1

    System.EventArgs e)

     12
            /// </summary>
    13
             public class Form1 : System.Windows.Forms.Form
    14
             {
    15
                private System.Windows.Forms.TextBox textBox1;
                private System.Windows.Forms.TextBox textBox2;
     16
    17
                private System.Windows.Forms.TextBox textBox3;
    18
                private System.Windows.Forms.Button button1;
    19
                /// <summary>
    20
                /// Required designer variable.
    21
                /// </summary>
    22
                private System.ComponentModel.Container components = null;
    23
    24
                public Form1()...
    36
                /// <summary>
    37
                /// Clean up any resources being used.
    38
                /// </summary>
    39 🕁
                protected override void Dispose( bool disposing )...
    51
                Windows Form Designer generated code
    111
   112
                /// <summary>
    113
                /// The main entry point for the application.
    114
                /// </summary>
                [STAThread]
    115
    116
                static void Main()
    117
                {
    118
                     Application.Run(new Form1());
    119
                }
    120
    121
                private void button1 Click(object sender, System.EventArgs e)
    122
                {
                     WriteSymbol("AGP1.PLC1","_DOO40_WORD", Enter value(Transmitting buffer) here);
    123
    124
    125
                }
    126
             }
   127 }
128 }
    129
    130
    131
    132
```

25 Describe the ProEasyDotNet directive.

Enter "using ProEasyDotNet;" at the bottom of the lines that state "using..." at the head of the source code.



26 For the write data storing area, declare a variable "wData".

The array type ("Short" in this example) must conform to the data type of the target symbol. Specify the same data length as the target symbol ("3" in this example).

```
Start Page | Form1.cs [Design]* Form1.cs*
♥☆ WindowsApplication5.Form1
                                                                              ▼ 중♦button1_Click(object sender,System.EventArgs e)
     25
     26
                 public Form1()...
     38
                 /// <summary>
     39
                 /// Clean up any resources being used.
     40
                 /// </summary>
                 protected override void Dispose( bool disposing )...
     41 🛱
    53 🖨
                 Windows Form Designer generated code
    113
    114
                 /// <summary>
                 /// The main entry point for the application.
    115
                 /// </summary>
    116
    117
                 [STAThread]
    118
                 static void Main()
    119
                 {
    120
                      Application.Run(new Form1());
    121
                 }
    122
    123
                 private void button1_Click(object sender, System.EventArgs e)
    124
                  {
    125
                      short[] wData = new short[3];
    126
    127
                      WriteSymbol("AGP1.PLC1","_D0040_WORD", Enter value(Transmitting buffer) here);
    128
    129
                 }
    130
             }
    131 }
    132
    133
    134
    135
    136
```

27 Set the data to be entered in [textBox1] to [textBox3] in the array.

Start Page | Form1.cs [Design]* Form1.cs* WindowsApplication5.Form1 Boutton1_Click(object sender,System.EventArgs e) 26由 public Form1()... 38ф /// <summary> /// Clean up any resources being used. 39 40 /// </summary> 41 protected override void Dispose(bool disposing)... 53 🖨 Windows Form Designer generated code 113 114 /// <summary> 115 /// The main entry point for the application. 116 /// </summary> 117 [STAThread] 118 static void Main() 119 { 120 Application.Run(new Form1()); 121 } 122 123 private void button1_Click(object sender, System.EventArgs e) 124 { 125 short[] wData = new short[3]; wData[0] = short.Parse(this.textBox1.Text); 126 wData[1] = short.Parse(this.textBox2.Text); 127 wData[2] = short.Parse(this.textBox3.Text); 128 129 WriteSymbol("AGP1.PLC1","_D0040_WORD", Enter value(Transmitting buffer) here); 130 131 132 } 133 } 134 } 135 136 137 138

28 Enter "ProEasy." before "WriteSymbol", and select [WriteDevice16] from the displayed list box.

```
Start Page | Form1.cs [Design]* Form1.cs*
✤ WindowsApplication5.Form1

    System.EventArgs e)

     26
                  public Form1()...
     38
                  /// <summary>
                  /// Clean up any resources being used.
     39
     40
                 /// </summary>
     41
                  protected override void Dispose( bool disposing )...
     53 🗄
                  Windows Form Designer generated code
    113
    114
                  /// <summary>
    115
                 ///% \label{eq:linear} The main entry point for the application.
    116
                  /// </summary>
    117
                 [STAThread]
    118
                  static void Main()
    119
                  {
    120
                      Application.Run(new Form1());
    121
                  }
    122
    123
                  private void button1_Click(object sender, System.EventArgs e)
    124
                  {
    125
                      short[] wData = new short[3];
    126
                      wData[0] = short.Parse(this.textBox1.Text);
    127
                      wData[1] = short.Parse(this.textBox2.Text);
    128
                      wData[2] = short.Parse(this.textBox3.Text);
    129
                      ProEasy. WriteSymbol("AGP1.PLC1","_D0040_WORD", Enter value(Transmitting buffer) here);
    130
    131
                             🕸 ReadSymbolVariantDM
                                                        132
                  }
                             🕸 ReadSymbolVariantM
    133
              }
                             🔷 ReferenceEquals
    134
        🚸 SymbolInformation
    135
                             WriteDevice16
                             WriteDevice16D
    136
                             🔷 WriteDevice16DM
    137
                             🚸 WriteDevice16M
    138
                             WriteDevice32
    139
                             📣 WriteDevice32D
    140
    141
    142
    143
    144
    145
    146
```

29 Delete "WriteSymbol" from the character string (write function) that has been pasted from the clipboard.



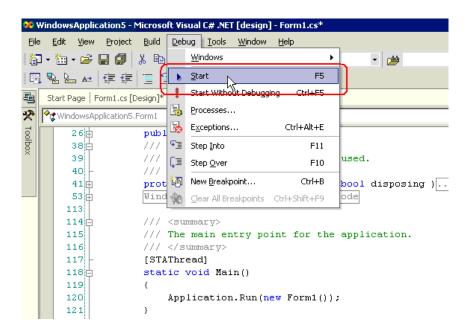
30 Specify a data storing area "wData" as the third argument. Enter "," (comma) at the end of the third argument, and then enter "3" to specify the length of the target symbol as the fourth argument.

```
Start Page | Form1.cs [Design]* Form1.cs*
♥☆ WindowsApplication5.Form1

    System.EventArgs e)

                 public Form1()...
    38
                 /// <summarv>
    39
                /// Clean up any resources being used.
    40
                 /// </summary>
     41 🗄
                 protected override void Dispose( bool disposing )...
    53 🖨
                 Windows Form Designer generated code
    113
    114
                 /// <summarv>
    115
                 /// The main entry point for the application.
                 /// </summary>
    116
    117
                 [STAThread]
    118
                 static void Main()
    119
                 {
    120
                     Application.Run(new Form1());
    121
                 }
    122
    123
                 private void button1_Click(object sender, System.EventArgs e)
    124
                 {
    125
                     short[] wData = new short[3];
    126
                     wData[0] = short.Parse(this.textBox1.Text);
                     wData[1] = short.Parse(this.textBox2.Text);
    127
                     wData[2] = short.Parse(this.textBox3.Text);
    128
    129
    130
                     ProEasy.WriteDevice16("AGP1.PLC1","_D0040_WORD",wData,3);
    131
    132
                 }
    133
             }
    134 }
    135
    136
    137
    138
```

31 Select [Start] from the [Debug] menu.



32 Immediately after startup, a character string "textBox*" is displayed in [TextBox].

E Form1	
textBox1	
textBox2	button1
textBox3	

After entering the write data (three items) in [TextBox], click [button1]. Then, the data will be written into the area specified with the symbol.

🖳 Form1	_ 🗆 🗡
100	
200	button1
300	ß