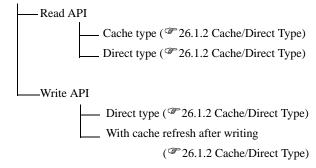
26 Designing Your Own Program

26.1	Using API Functions	26-2
26.2	Device Access APIs	26-19
26.3	Cache Buffer Control APIs	26-30
26.4	Queuing Access Control APIs	26-36
26.5	System APIs	.26-39
26.6	SRAM Data Access APIs	26-46
26.7	CF Card APIs	.26-52
26.8	Other APIs	26-61
26.9	Precautions for Using APIs	26-66
26.10	Using APIs (Examples)	26-77

26.1 Using API Functions

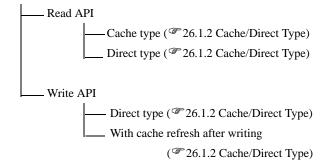
Reading and writing a Device/PLC

Single-handle functions (@ 26.1.1 Single-/Multi-Handle Functions)



PLC communication with multiple devices

Multi-handle functions (@26.1.1 Single-/Multi-Handle Functions)



For effective communication

- Group symbol access (@ 26.1.4 Group Access)
- Queuing access (@26.1.5 Queuing Access)

Other functions

- System APIs (→26.1.7 System APIs)
- SRAM Data Access APIs (→26.1.8 SRAM Data Access APIs)
- CF Card APIs (\rightarrow 26.1.9 CF Card APIs)
- Other APIs (\rightarrow 26.8 Other APIs)

26.1.1 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()".

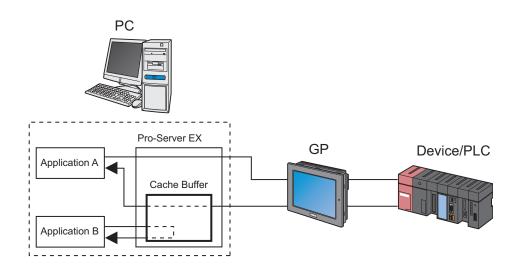
Multi-Handle APIs can be used for multi-thread applications, or for simultaneous access to multiple Devices/ PLCs.

26.1.2 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.

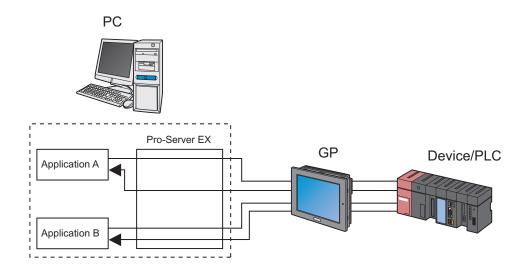
With the Cache Read feature, however, when two applications A and B send reading requests to the same Device/ PLC, 'Pro-Server EX' reads data from the Device/PLC according to the request of Application A first, stores the read data into the internal cache buffer, and sends the data to Application A in 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 "26.3 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.

26.1.3 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:

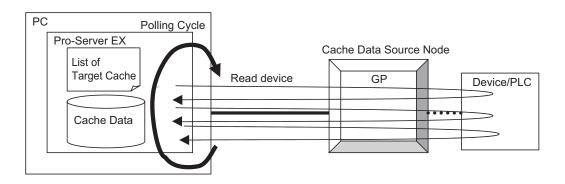
 Registration using 'Pro-Studio EX' (Create a cache buffer in "Device Cache" on the feature screen, and register it in a network project file.)
 Registration using API

Cache buffer updating procedure

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

The principle of polling method

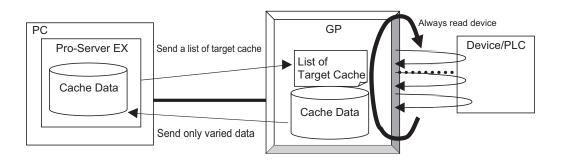
According to a list of target devices in the cache buffer, 'Pro-Server EX' reads device data to update the cache buffer when the cycle specified in cache buffer registration is reached.



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).

26.1.4 Group Access

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

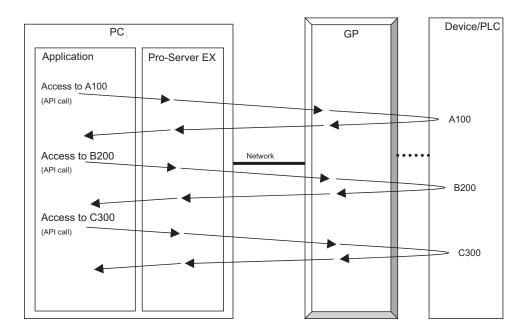
With a group symbol, 'Pro-Server EX' can efficiently access multiple devices with a single call of an API.

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 GP 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.

• 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.

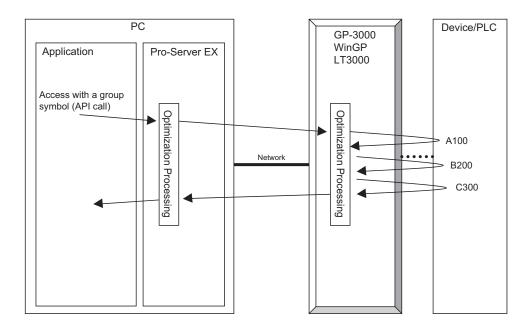


When accessing group symbols

The processing varies depending on whether the target node is in the GP3000 Series, WinGP, LT3000 or GP Series.

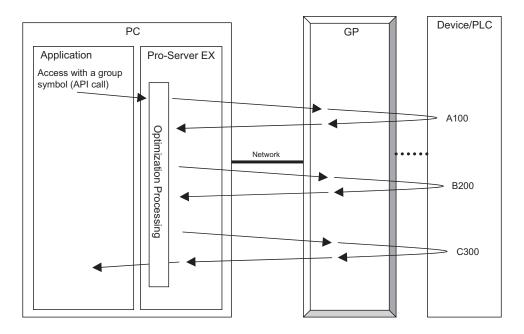
• For GP3000 Series node, WinGP node or LT3000 node

'Pro-Server EX' sends a request for the GP3000 Series node, WinGP node or LT3000 node only once. The GP3000 Series node, WinGP node or LT3000 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.



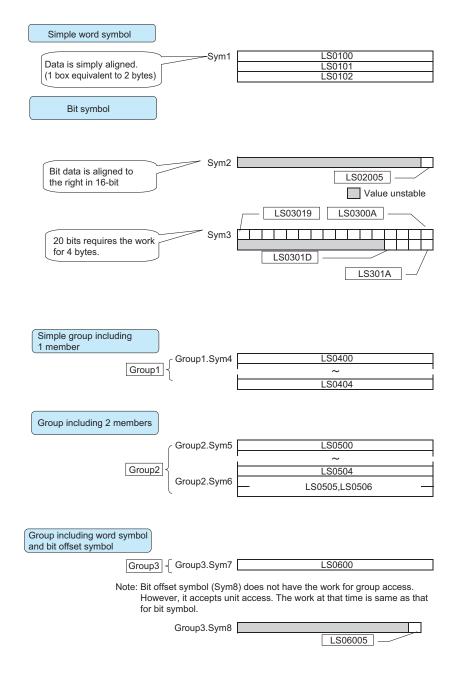
Data structure for group symbol access

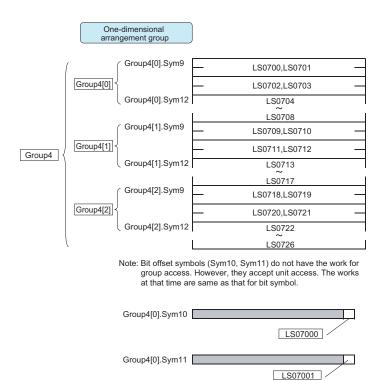
When 'Pro-Server EX' accesses devices via a group symbol, the data buffer structure varies depending on the symbol type or size of the group. The data buffer structure by group symbol type is as follows:

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.
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.

Group symbol data type	Secured data size
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.

Examples of the data buffer structures are shown below.





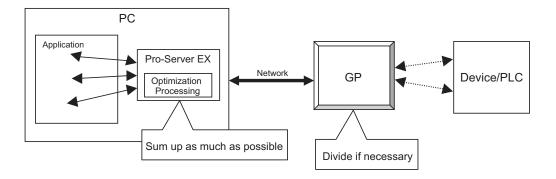
Device addresses for Group4[1].Sym10 and Group4[1].Sym11 are LS07090 and LS07091 respectively. Device addresses for Group4[2].Sym10 and Group4[2].Sym11 are LS0718 and LS07181 respectively.

Two-dimensional group (nest of groups)

	([Group6[0]] ≺	Group6[0].Group5[0] {Group6[0].Group5[0].Sym13 Group6[0].Group5[1] {Group6[0].Group5[1].Sym13 Group6[0].Group5[2] {Group6[0].Group5[2].Sym13 Group6[0].Sym15 {Group6[0].Sym15	LS0900 LS0901 LS0902 LS0903 LS0904 LS0905
Group6	Group6[1] ≺	Group6[1].Group5[0] {Group6[1].Group5[0].Sym13 Group6[1].Group5[1] {Group6[1].Group5[1].Sym13 Group6[1].Group5[2] {Group6[1].Group5[2].Sym13 Group6[1].Sym15 {Group6[1].Sym15	LS0908 LS0909 LS0910 LS0911 LS0912 LS0913 - LS0914,LS0915
	[Group6[2]] ≺	Group6[2].Group5[0] {Group6[2].Group5[0].Sym13 Group6[2].Group5[1] {Group6[2].Group5[1].Sym13 Group6[2].Group5[2] {Group6[2].Group5[2].Sym13 Group6[2].Sym15 {Group6[2].Sym15	LS0916 LS0917 LS0918 LS0919 LS0920 LS0921 - LS0922,LS0923 -

26.1.5 Queuing Access

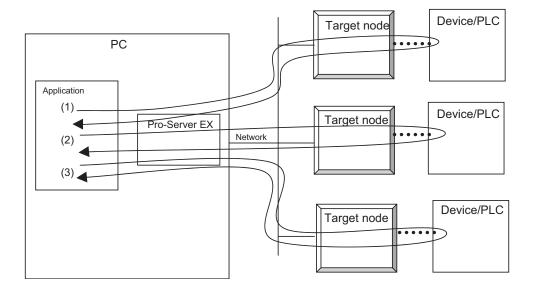
'Pro-Server EX' stores a device access request every time an API is called, and then optimizes the stored requests to access individual devices at once.



The principle of queuing access

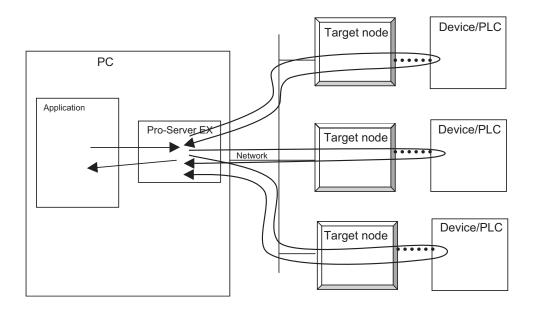
♦ Simple API access

'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.

NOTE

- 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. During "Access request registration", 'Pro-Server EX' stores the access data buffer address (address only, excluding data). Therefore, after "Access request registration", the data buffer must remain in the address specified in "Access request registration", until ExecuteQueuingAccess() returns a value after it is called.
 - Otherwise, 'Pro-Server EX' will access an invalid address, causing a fatal error.
 - Also, when queuing qccess is used again, the data buffer must remain in the address specified in "access request registration".

26.1.6 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 "26.1.4 Group Access" for more details.

26.1.7 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.

26.1.8 SRAM Data Access APIs

The SRAM incorporated in the GP Series stores various data depending on the GP 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.

26.1.9 CF Card APIs

The CF card incorporated in the GP Series stores various data depending on the GP setup and operating conditions, like SRAM.

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

26.2 Device Access APIs

■ Single-Handle Cache Read APIs

Function	Bit data
INT WINAPI ReadDev	viceBit(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount);
Function	16-bit data
INT WINAPI ReadDev	rice16(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount);
Function	32-bit data
INT WINAPI ReadDev	rice32(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* odwData,WORD wCount);
Function	16-bit BCD data
INT WINAPI ReadDev	riceBCD16(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount);
Function	32-bit BCD data
INT WINAPI ReadDev wCount);	iceBCD32(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* odwData,WORD
Function	Single-precision floating point data
INT WINAPI ReadDev	riceFloat(LPCSTR sNodeName,LPCSTR sDeviceName,FLOAT* ofIData,WORD wCount);
Function	Double-precision floating point data
INT WINAPI ReadDev wCount);	riceDouble(LPCSTR sNodeName,LPCSTR sDeviceName,DOUBLE* odbData,WORD
Function	Character string data
INT WINAPI ReadDev	riceStr(LPCSTR sNodeName,LPCSTR sDeviceName,LPSTR psData,WORD wCount);
Function	General-use data
INT WINAPI ReadDev wAppKind);	rice(LPCSTR sNodeName,LPCSTR sDeviceName,LPVOID pData,WORD wCount,WORD
Function	General-use data (Variant-type)
INT WINAPI ReadDev wCount,WORD wApp	riceVariant(LPCSTR sNodeName,LPCSTR sDeviceName,LPVARIANT pData,WORD Kind);
Function	Group symbol
INT WINAPI ReadSyn	bol(LPCSTR sNodeName,LPCSTR sSymbolName,LPVOID oReadBufferData);
Function	Group symbol (Variant-type)
INT WINAPI ReadSyn	bolVariant(LPCSTR sNodeName,LPCSTR sSymbolName,LPVARIANT pData);

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

■ Single-Handle Direct Read APIs

Function	Bit data				
INT WINAPI ReadDevi	INT WINAPI ReadDeviceBitD(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount);				
Function	16-bit data				
INT WINAPI ReadDevice16D(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount);					

Function	Bit data
Function	32-bit data
INT WINAPI ReadDevi	ce32D(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* odwData,WORD wCount);
Function	16-bit BCD data
INT WINAPI ReadDevi	ceBCD16D(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount);
Function	32-bit BCD data
INT WINAPI ReadDevi wCount);	ceBCD32D(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* odwData,WORD
Function	Single-precision floating point data
INT WINAPI ReadDevi	ceFloatD(LPCSTR sNodeName,LPCSTR sDeviceName,FLOAT* oflData,WORD wCount);
Function	Double-precision floating point data
INT WINAPI ReadDevi wCount);	ceDoubleD(LPCSTR sNodeName,LPCSTR sDeviceName,DOUBLE* odbData,WORD
Function	Character string data
INT WINAPI ReadDevi	ceStrD(LPCSTR sNodeName,LPCSTR sDeviceName,LPSTR psData,WORD wCount);
Function	General-use data
INT WINAPI ReadDevi wAppKind);	ceD(LPCSTR sNodeName,LPCSTR sDeviceName,LPVOID pData,WORD wCount,WORD
Function	General-use data (Variant-type)
INT WINAPI ReadDevi wCount,WORD wAppK	ceVariantD(LPCSTR sNodeName,LPCSTR sDeviceName,LPVARIANT pData,WORD (ind);
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);

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

■ Single-Handle Direct Write APIs

Function	Bit data				
INT WINAPI WriteDeviceBitD(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* pwData,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	16-bit BCD data				
INT WINAPI WriteDeviceBCD16D(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* pwData,WORD wCount);					
Function	32-bit BCD data				

Function	Bit 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);
* For each parameter, ple	ase refer to "■ Parameters of read/write functions".

■ 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	6-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	16-bit BCD data					
INT WINAPI WriteDev	iceBCD16(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* pwData,WORD wCount);					
Ennetion						
Function	32-bit BCD data					
	32-bit BCD data iceBCD32(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* pdwData,WORD					
INT WINAPI WriteDev						
INT WINAPI WriteDev wCount); Function	iceBCD32(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* pdwData,WORD					
INT WINAPI WriteDev wCount); Function	iceBCD32(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* pdwData,WORD Single-precision floating point data					
INT WINAPI WriteDev wCount); Function INT WINAPI WriteDev Function	iceBCD32(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* pdwData,WORD Single-precision floating point data iceFloat(LPCSTR sNodeName,LPCSTR sDeviceName,FLOAT* pflData,WORD wCount);					

Function	Bit data				
INT WINAPI WriteDevi	ceStr(LPCSTR sNodeName,LPCSTR sDeviceName,LPCSTR psData,WORD wCount);				
Function	General-use data				
INT WINAPI WriteDevi wAppKind);	ce(LPCSTR sNodeName,LPCSTR sDeviceName,LPVOID pData,WORD wCount,WORD				
Function	General-use data (Variant-type)				
INT WINAPI WriteDevi wCount,WORD wAppK	ceVariant(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);				

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

Multi-Handle Cache Read APIs

Function	Bit data				
INT WINAPI ReadDeviceBitM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount);					
Function	16-bit data				
	INT WINAPI ReadDevice16M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount);				
Function	32-bit data				
INT WINAPI ReadDev odwData,WORD wCou	ice32M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* nt);				
Function	16-bit BCD data				
INT WINAPI ReadDev owData,WORD wCoun	iceBCD16M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* t);				
Function	32-bit BCD data				
	INT WINAPI ReadDeviceBCD32M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* odwData,WORD wCount);				
Function	Single-precision floating point data				
INT WINAPI ReadDev oflData,WORD wCount	iceFloatM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,FLOAT*				
Function	Double-precision floating point data				
INT WINAPI ReadDev odbData,WORD wCour	iceDoubleM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,DOUBLE* nt);				
Function	Character string data				
INT WINAPI ReadDev psData,WORD wCount	iceStrM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,LPSTR);				
Function	General-use data				

Function	Bit data			
INT WINAPI ReadDeviceM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,LPVOID pData,WORD wCount,WORD wAppKind);				
Function	General-use data (Variant-type)			
INT WINAPI ReadDeviceVariantM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,LPVARIANT pData,WORD wCount,WORD wAppKind);				
Function	Group symbol			
INT WINAPI ReadSymbolM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sSymbolName,LPVOID oReadBufferData);				
Function	Group symbol (Variant-type)			
INT WINAPI ReadSymbolVariantM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sSymbolName,LPVARIANT pData);				

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

Multi-Handle Direct Read APIs

Function	Bit data				
INT WINAPI ReadDeviceBitDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount);					
Function	16-bit data				
	INT WINAPI ReadDevice16DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount);				
Function	32-bit data				
INT WINAPI ReadDevi odwData,WORD wCou	ce32DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* nt);				
Function	16-bit BCD data				
INT WINAPI ReadDevi owData,WORD wCount	ceBCD16DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* t);				
Function	32-bit BCD data				
INT WINAPI ReadDevi odwData,WORD wCou	ceBCD32DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* nt);				
Function	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 * 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,	ceDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,LPVOID WORD wAppKind);				
Function	General-use data (Variant-type)				

Function	Bit data			
INT WINAPI ReadDeviceVariantDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sNodeName,LPCSTR sDeviceName,LPVARIANT pData,WORD wCount,WORD wAppKind);				
Function	Group symbol			
INT WINAPI ReadSym oReadBufferData);	INT WINAPI ReadSymbolDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sSymbolName,LPVOID oReadBufferData);			
Function Group symbol (Variant-type)				
INT WINAPI ReadSymbolVariantDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sSymbolName,LPVARIANT pData);				

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

Multi-Handle Direct Write APIs

Function	Bit data					
INT WINAPI WriteDev pwData,WORD wCoun	iceBitDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* t);					
Function	Function 16-bit data					
INT WINAPI WriteDev pwData,WORD wCoun	ice16DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* t);					
Function	32-bit data					
INT WINAPI WriteDev pdwData,WORD wCou	ice32DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* nt);					
Function	16-bit BCD data					
INT WINAPI WriteDev pwData,WORD wCoun	iceBCD16DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* t);					
Function	32-bit BCD data					
	iceBCD32DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR * pdwData,WORD wCount);					
Function	Single-precision floating point data					
INT WINAPI WriteDev pflData,WORD wCount	iceFloatDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,FLOAT*);					
Function	Double-precision floating point data					
	iceDoubleDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR ^{2*} pdbData,WORD wCount);					
Function	Character string data					
INT WINAPI WriteDev psData,WORD wCount	iceStrDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,LPCSTR);					
Function	General-use data					
INT WINAPI WriteDev pData,WORD wCount,V	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	Bit data			
Function	Group symbol			
INT WINAPI WriteSym pWriteBufferData);	INT WINAPI WriteSymbolDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sSymbolName,LPVOID pWriteBufferData);			
Function Group symbol (Variant-type)				
INT WINAPI WriteSymbolVariantDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sSymbolName,LPVARIANT pData);				

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

■ Multi-Handle Write APIs with Cache Refresh after Writing

Function	Bit data					
	INT WINAPI WriteDeviceBitM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* pwData,WORD wCount);					
Function	Function 16-bit data					
	INT WINAPI WriteDevice16M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* pwData,WORD wCount);					
Function	32-bit data					
INT WINAPI WriteDev pdwData,WORD wCour	ice32M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* nt);					
Function	16-bit BCD data					
INT WINAPI WriteDev pwData,WORD wCount	iceBCD16M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD*					
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 WriteDeviceFloatM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,FLOAT* pflData,WORD wCount);					
Function	Double-precision floating point data					
INT WINAPI WriteDeviceDoubleM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,DOUBLE* pdbData,WORD wCount);						
Function	Character string data					
INT WINAPI WriteDeviceStrM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,LPCSTR psData,WORD wCount);						
Function	General-use data					
INT WINAPI WriteDeviceM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,LPVOID pData,WORD wCount,WORD wAppKind);						
Function	General-use data (Variant-type)					
	INT WINAPI WriteDeviceVariantM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,LPVARIANT pData,WORD wCount,WORD wAppKind);					
Function	Group symbol					

Function	Bit data			
INT WINAPI WriteSymbolM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sSymbolName,LPVOID pWriteBufferData);				
Function	Group symbol (Variant-type)			
INT WINAPI WriteSymbolVariantM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sSymbolName,LPVARIANT pData);				

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

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	16 bits		32 bits		Float	Double	String
	Dit	S/U/HEX	BCD	S/U/HEX	BCD	Tioat	Double	Othing
XXXDeviceBit	0	-	-	-	-	-	-	-
XXXDevice16	-	0	-	-	-	-	-	-
XXXDevice32	-	-	-	0	-	-	-	-
XXXDeviceBCD16	-	-	0	-	-	-	-	-
XXXDeviceBCD32	-	-	-	-	0	-	-	-
XXXDeviceFloat	-	-	-	-	-	0	-	-
XXXDeviceDouble	-	-	-	-	-	-	0	-
XXXDeviceStr	-	-	-	-	-	-	-	0
XXXDevice	0	0	0	0	0	0	0	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
16-bit data	WORD * pwData
32-bit data	DWORD * pdwData
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

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
16-bit data	1020	1020
32-bit data	510	510
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)

wAppKind : Data type specification

Value	Data type	Value	Data type
1	Bit	7	Unsigned 32 bits
2	Signed 16 bits	8	HEX 32 bits
3	Unsigned 16 bits	9	BCD 32 bits
4	HEX 16 bits	10	Float
5	BCD 16 bits	11	Double
6	Signed 32 bits	12	String

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 handing several consecutive bit strings, it is more effective to use the read/write functions that handle data in multiples of 16 bits or 32 bits (Read/WriteDevice16 or Read/WriteDevice32), rather than Read/WriteDeviceBit.

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

When using the 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/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.

26.3 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 INT WINAPI PS_CreateCacheM(HANDLE hProServer, LPCSTR sCacheName, DWORD dwPollingTime);						
Argument Return value sCacheName: (In) Cache buffer name Normal end: 0 dwPollingTime: (In) To select the constant monitoring method, specify "0". Abnormal end: Error code The cache buffer is updated as fast as possible. If you specify any value other than "0", the polling method is selected. Specify the polling cycle (cache updating cycle) by the millisecond. Abnormal end: Error code						
 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.						
Single INT WINAPI PS_EntryCacheRecord(LPCSTR sCacheName, LPCSTR sNodeName, LPCSTR sDevice, WORD wAppKind, WORD wCount); Multi INT WINAPI PS_EntryCacheRecordM(HANDLE hProServer, LPCSTR sCacheName, LPCSTR sNodeName, LPCSTR sDevice, WORD wAppKind, WORD wCount);						

0	rgument							
sCacheName: (In) Cache buffer name Register a cache source device into the cache buffer specified with this Normal end: 0								
	Re nai	Abnormal end: Error code						
sNodeN		Device/PLC name						
		he source device	ource					
			can dire	ectly specify the device address, or				
		a symbol or group registered w						
		nultiple symbols can be registe	red at	once.				
wAppK		Source device data type						
	Available data types vary depending on the cache source device designation method.							
		n device address of cache sourc	e devi	ce is directly specified.				
		a data type (1 to 12) available						
	specifie							
			Val					
	Value	Data type		Data type				
			ue					
	1	Bit	7	32 bits, Unsigned decimal				
	2	16 bits, Signed decimal	8	32 bits, Hexadecimal				
	3	16 bits, Unsigned decimal	9	32 bits, BCD				
	4	16 bits, Hexadecimal	10	Single-precision floating point				
	5	16 bits, BCD	11	Double-precision floating point				
	6	32 bits, Signed decimal	12	Character string				
	 6 32 bits, Signed decimal 12 Character string b) When symbol is specified for cache source device: Specify a data type (0 to 12) available with 'Pro-Server EX'. If you specify "0", the symbol type specified in symbol definition is used. c) When group is specified for cache source device: Fixed to "0". The symbol type is registered for all symbols in the specified group. wCoutt: (In) Device data quantity subjected to caching Available values vary depending on the cache source device specification method. a) When device address of cache source device is directly specified: Data quantity (1 to 1020) according to the device type can be used. (The maximum value varies depending on the device type.) b) When symbol is specified for cache source device: If you specify "0", the quantity specified in symbol definition is used. If you specify any value other than 0, data quantity (1 to 1020) according to the device type.) c) When group is specified for cache source device: Fixed to "0". c) When group is specified for cache source device: Fixed to "0". d) When group is specified for cache source device: Fixed to "0". d) When group is specified for cache source device: Fixed to "0". d) When group is specified for cache source device: Fixed to "0". 							
Special	Note							

Function	Starting caching	
Starts caching.		
Multi	Cache(LPCSTR sCacheName); CacheM(HANDLE hProServer, LPCSTR sCacheName);	
	e of cache buffer to start name registered with 'Pro-Studio EX' can be also specified.	Return value Normal end: 0 Abnormal end: Error code
Special Note		
Function	Stopping caching	
Temporarily stops cachi Caching stops, but defin To restart caching, call H	ition of the cache buffer is retained.	
Multi	Cache(LPCSTR sCacheName); CacheM(HANDLE hProServer, LPCSTR sCacheName);	
Argument sCacheName: (In) Name A cache buffer	e 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	acheStatus(LPCSTR sCacheName);	
	acheStatusM(HANDLE hProServer, LPCSTR sCacheName);	
	e of cache buffer to be checked	Return value 0 : The cache buffer has been created, but not started yet. 1: Caching in progress
	name registered with 'Pro-Studio EX' can be also specified.	2: Caching under suspension XX: Error code

Function	Discarding cache buffer	
Stops caching, and discar	rds the cache buffer.	
Multi	yCache(LPCSTR sCacheName); yCacheM(HANDLE hProServer, LPCSTR sCacheName);	
Argument		Return value
sCacheName: (In) Name	of cache buffer to be discarded istered with 'Pro-Studio EX' can be also specified.	Normal end: 0 Abnormal end: Error code
Function	Setting cache update notification function	1
Sets the function to notif	cache buffer update status to a specified window.	
device is frequently cach 'Pro-Server EX' can send has a change with the con If your system is built so be improved. This API allows you to s message" in 'Pro-Server' After these settings are no Single INT WINAPI PS_SetNo LPARAM LParam, HAN Multi INT WINAPI PS_SetNo message, WPARAM WP	a message to a specified window, when cache data is updated (w nstant monitoring method, or when one polling cycle is complete as to execute cache-reading of a device after receiving this messa et "Target cache buffer name", "Window to receive the message"	when at least one target device a with the polling method). age, the system efficiency ca a can a contents of the ently-set notification function essage, WPARAM WParam,
Argument sCacheName: (In) Cache A cache bu	buffer name affer name registered with 'Pro-Studio EX' can be also specified.	Return value Normal end: 0 Abnormal end: Error code
message: (In) Message II wParam: (In) WPARAM LParam: (In) LPARAM	the window to receive the message D to be sent to the window value to be sent to the window together with message ID value to be sent to the window together with message ID Returns the ID that identifies the currently set notification	
wParam: (In) WPARAM LParam: (In) LPARAM ohCacheNotifyID: (Out) function. Special Note If the returned handle is a After the cache buffer is	value to be sent to the window together with message ID value to be sent to the window together with message ID	

For details of PostMessage(), refer to the Windows API Manual.

	Γ				
Function	Accepting next cache update notification				
Accepts the next cache update notification.					
However, once this notifica again, even if the cache but notification routine, a multi update message. (If the not call error occurs with the ro To prevent this error, this A By calling this API at the e	the function to send a message to a specified window when a catation function is executed, 'Pro-Server EX' will not send a mess ffer is updated next. This is because in case it has taken a long iple-call error can occur with the relevant routine when 'Pro-Se ification routine receives the next message before completion putine.) API explicitly informs 'Pro-Server EX' that it can send the next nd of the processing of the notification routine, you can build by time a cache buffer is updated.	sage until this API is called time in processing with the erver EX' sends the next cache of the processing, a multiple- message.			
Multi	/extNotifyFromCache(HANDLE hCacheNotifyID); /extNotifyFromCacheM(HANDLE hProServer, HANDLE hC	acheNotifyID);			
Argument Return value hCacheNotifyID: (In) ID of next message acceptance notification function Normal end: 0 ID obtained with PS_SetNotifyFromCache() Abnormal end: Error code					
Special Note					
Function	Canceling cache update notification				
Cancels the function for set	nding a cache buffer update message to a specified window.				
cache buffer related with he Single INT WINAPI PS_KillNotit Multi	rver EX' will not send a cache buffer update message to the rel CacheNotifyID is updated. fyFromCache(HANDLE hCacheNotifyID); fyFromCacheM(HANDLE hProServer, HANDLE hCacheNot				
Argument Return value hCacheNotifyID: (In) ID of the notification function to be canceled Normal end: 0 ID obtained with PS_SetNotifyFromCache() Abnormal end: Error code					
Therefore, if 'Pro-Server E.' window before this API is	discard a message sent from 'Pro-Server EX', even if the mess X' has sent a message to a window and the application has not called, the application can fetch the message from the window the notification routine may be called even after this API is ca	fetched the message from the even after this API is called.			

Function	Acquiring cache buffer update count	
Returns a cache buffer upo	late count.	
Using this function, you c	count on the program, you can check if a cache buffer has be an omit unnecessary calls of device cache read APIs. (Even is o change, the value will not be changed.)	
Multi	ateCounter(LPCSTR sCacheName, DWORD* odwCount); ateCounterM(HANDLE hProServer, LPCSTR sCacheName,	DWORD* odwCount);
A cache buffer n odwCount: (Out) Cache b	of cache buffer to be monitored ame registered with 'Pro-Studio EX' can be also specified. uffer update count per of updates from 0 to 4294967295 endlessly.	Return value Normal end: 0 Abnormal end: Error code

26.4 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 BeginQueuingRead(); Multi					
INT WINAPI BeginQueun	ngReadM(HANDLE hProServer);	1			
Argument		Return value Normal end: 0 Abnormal end: Error code			
 Do not call a Device Wri BeginQueuingRead() is direct read requests cann To discard a request in q 	 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, 'Pro-Server EX' queues device write requests until ExecuteQueuingAccess() is called. Queuing is executed for each Pro-Server handle.				
Single INT WINAPI BeginQueuin Multi INT WINAPI BeginQueuin	ngWrite(); ngWriteM(HANDLE hProServer);				
		Return value			
Argument		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. 					

Function	Executing device read/write request in queue	
Accesses device data acco	rding to the device read/write request in queue.	
Single INT WINAPI ExecuteQue Multi INT WINAPI ExecuteQue	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	essfully accesses all specified devices, ExecuteQueuin access any device, on the other hand, ExecuteQueuin ether each device access request has been successfull eeded() to check the result. TIONs in queuing access.	ngAccess() returns an access error code.
Function	Discarding device read/write request in queue	
Discards the device read/v	vrite request in queue.	
Single INT WINAPI CancelQuer Multi INT WINAPI CancelQuer	iingAccess(); iingAccessM(HANDLE hProServer);	
Argument		Return value Normal end: 0 Abnormal end: Error code
ExecuteQueuingAccess()	() or BeginQueuingRead()is called, 'Pro-Server EX' c is called. mes unnecessary for any reason, call this API. 'Pro-Se	

Function	Checking the success or failure of device read/write request in queue		
called. Single INT WINAPI IsQueuingA Multi	h device access request has been successfully executed, after E ccessSucceeded(INT iIndex); ccessSucceededM(HANDLE hProServer,INT iIndex);	ExecuteQueuingAccess() is	
called several times to que called. Note that you canno ExecuteQueuingAccess(). If you wish to know a resu	uest to be checked) or BeginQueuingRead() is called, Device Access APIs are ue device access requests until ExecuteQueuingAccess() is ot know an actual device access result until execution of It of each device access request, execute ïrst, and then specify the number (from 0) of the request for	Return value XX: Error code 0: Indicates that the device access request of the specified number has been successfully executed.	
WriteDevice16("No WriteDevice16("No ExecuteQueuingAccess() To check if the "Node1" ac	ode1","LS100",Data,10); ode1","LS200",Data,10); ode1","LS300",Data,10); ccess to "LS200" has been successfully executed, use IsQueuin his access has been successfully executed.	ngAccessSucceeded(1).	

26.5 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 = TI etworkProjectM(HANDLE hProServer,LPCSTR sDBName,DW	
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()	Irned by each API of 'Pro-Server EX' into an error message. returns a multibyte character string (ASCII) as a message. Easy ring (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'. essageEx() 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)
EasyLoadErrorMessageEx error occurrence place and return a different error mess EasyLoadErrorMessageEx message.	always returns the same error message relative to a specified erro () 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	communication target device, LoadErrorMessageEx() may ter string (ASCII) as a
BOOL WINAPI EasyLoad 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	STR osErrorMessage);
osErrorMessage: (Out) Poi string) storing area.(owsErrorMessage: (Out) P	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.) ointer 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. Therefouses an error and EasyLoadErrorMessage(),EasyLoadErrorMessage se stored error status information is rewritten. For this reason, wM(), 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	lly start.
INT WINAPI EasyInit();		
Argument		Return value Normal end: 0 Abnormal end: Error code
Special Note		
Function	Ending Pro-Server API	
INT WINAPI EasyTerm();		
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.	
Argument		Return value Normal end: 0 Abnormal end: Error code
Special Note		

This ADI allows you to 1	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);	
uMessage: (In) Message IE This ID will be sent t EX is being closed. WParam: (In) WPARAM to PostMessage())	by 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	Return value Normal end: 0 Abnormal end: Error code
For example, if you specify API, 'Pro-Server EX' sends Generally, an application us	an application that closes at the same time when 'Pro-Server E. 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 le 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 procesong. This API can specify whether to execute or inhibit the Wirrocessing 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 clich rtant messages for timer and window re-drawing is also disable execute or inhibit the processing of Windows messages for eases 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);	
	o-Server handle subjected to processing mode change nessage processing, specify "1".	Return value Normal end: 0 Abnormal end: Error code

Function	Acquiring message processing mode				
1	ge processing mode during a call of a Pro-Server EX A rns the current message processing mode for each hand				
Single INT EasyGetWaitType(); Multi INT EasyGetWaitTypeM(H	IANDLE hProServerHandle);				
Argument HANDLE hProServerHand	lle: (In) Handle subjected to status acquisition	Return value1: Executes messageprocessing.2: Inhibits messageprocessing.			
Special Note					

Function A	Adding log into log viewer			
If a specific event ('Pro-Server the event.	EX' start/closin	ıg, ei	rror, etc.) occurs with internal processing	g, 'Pro-Server EX' can record
With this API, 'Pro-Server EX			log viewer. (See "27.5 Monitoring Systenessage by using this function. This AP	
debugging.				
INT WINAPI EasyOutputLog	(BYTE bLevel,	LPC	STR sPrompt,LPCSTR sMessage);	
				Return value
Argument				Normal end: 0 Abnormal end: Error code
bLevel: (In) Event type	more accult in m	- mf - m	monor deterioration. To provent this	
			mance deterioration. To prevent this, for recording messages by event type.	
Specify the event type the	hat the current r		ding message belongs to.	
The event types are liste	d below.			
	Hex	ad		
Definition	ecir		Event type	
	val			
EASY_LogLevel_SysMes	-)1	System message	
EASY_LogLevel_SysErro	or Ox(02	System error message	
EASY_LogLevel_AppErro	or Ox()4	User program error message	
EASY_LogLevel_AppStat	rt Ox()8	User program starting message	
EASY_LogLevel_AppEnd	0x1	10	User program closing message	
EASY_LogLevel_AppWa	rning 0x2	20	User program warning message	
EASY_LogLevel_AppMes	ssage1 0x4	40	User program detail message 1	
EASY_LogLevel_AppMes	ssage2 0x8	80	User program detail message 2	
	•			
			currence position (NULL-terminated) be recorded (NULL-terminated)	
	0			
	e is a simple cor	nbin	ation of two character stings (sPrompt	
and sMessage).				
Special Note				

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

26.6 SRAM Data Access APIs

Function

Reading SRAM backup data

Reads the following data stored in the SRAM of a GP Series node, 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		
gument	Normal end: 0				
	veFileName: (In) File path of the file to save read data. (String pointer)				
	(In) Name of read data source		code		
	rver EX nodes cannot be specif	fied.			
	(In) Saving mode	lready exists, 'Pro-Server EX' deletes the file,			
	erwrites it.)	ineady exists, 110-server EX deletes the file,			
	· ·	end of an existing file. If there is no file to			
save th	e data, 'Pro-Server EX' creates				
	: Reserve				
ackupData'	Type: (In) Type of data to be re	ead			
\/- ···	Data source node	Data source node			
Value	in GP Series	in GP3000 Series/WinGP/LT3000			
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			
0x0005 0x0006	Alarm block 1 Alarm history or Alarm block 2	Alarm block 1 Alarm block 2			
	Alarm history or Alarm				
0x0006	Alarm history or Alarm block 2	Alarm block 2			
0x0006 0x0007	Alarm history or Alarm block 2 Alarm log or Alarm block 3	Alarm block 2 Alarm block 3			
0x0006 0x0007 0x0008	Alarm history or Alarm block 2 Alarm log or Alarm block 3 Alarm block 4	Alarm block 2 Alarm block 3 Alarm block 4			

When the data source node is in the GP3000 Series/WinGP/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.

Alarm block 8

(Reserve)

- (1) Alarm history
- (2) Alarm log

0x000C

Others

(3) Alarm active

If there is no valid data, an error occurs.

Alarm block 8

(Reserve)

Special Note

Function	Reading extended SRAM backup data
Reads the following data stored in the SRAM of a GP Series node, 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. Unlike EasyBackupDataRead(), this API enables access to extended data for the GP3000 Series, WinGP and LT3000	
<i>v</i> 1	ttaReadEx(LPCSTR sSaveFileName, LPCSTR sNodeName, INT iBackupDataType, INT), INT iStringTable = 0x0000);

sNod Pro-S SaveMode 0: Ne and o 1: Ad the da Other	eName: (In) Name of read data erver EX nodes cannot be spec :: (In) Saving mode w (If a file with the same name verwrites it.)	ified. already exists, 'Pro-Server EX' deletes the file, e end of an existing file. If there is no file to save ew file.))	Return value Normal end: 0 Abnormal end: Erro code
Value	Data source node	Data source node	
value	in GP Series	in GP3000 Series/WinGP/LT3000	
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	
00005	Alarm block 1	Specify iNumber for alarm type.	
0x0006	Alarm history or Alarm block 2	Alarm block 2 Specify iNumber for alarm type.	
0x0007	Alarm log or Alarm 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.	
0x8002	(Reserve)	Sampling group of a specific group number Specify iNumber for group number.	

Series, WinGP and LT For a file of the GP Se	T3000. eries, this argun	sSaveFileName specifies a file of the GP300 nent is ignored. nent varies depending on the value of	0		
Value of iBackupDataType		Description			
	Three types of alarm data (active, history and log) are available. Specify a target alarm type.				
	Value of iNumber	Description			
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				
Others	(Reserve)				
iStringTable: (In) Reserve Always specify "0".					

Function	Writing SRAM backup data		
Writes specified filing 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 Pro-Server EX nodes cannot be specified.	e path of binary-formatted filing data to be written (String entry node to write data (String pointer) s, GP3000 Series nodes, WinGP nodes or LT3000 nodes ed to "1". ("1" indicates filing data.)	Return value Normal end: 0 Abnormal end: Error code	
Special Note		I	

26.7 CF Card APIs

Acquires connection status of the CF card in a o Single INT WINAPI EasyIsCFCard(LPCSTR sNodeN Multi INT WINAPI EasyIsCFCardM(HANDLE hPro	Jame);	deName) ·	
Argument ProServer: Pro-Server handle SNodeName: Name of GP node to read status	Return value		
(This node name must be pre-registered	Function	For GP Series	For GP3000 Series
in a network project.)	return value	node	node/WinGP node
	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 relation to CF card	

Function	n Reading file list from CF card (Optional folder name)	
Outputs a list of files from the CF card inserted in a GP node into a file specified with the parameter. You can specify an optional file to save the file list.		
INT WINAPI EasyGetList sSaveFileName) ;	InCfCard(LPCSTR sNodeName, LPCSTR sDirectory, INT* oi	Count, LPCSTR
oiCount: Number of output sSaveFileName: Name of f stores binary data of quantity specified wi and extension are sav struct stEasyDirInfo BYTE bFileName[8- BYTE bExt[3+1];// I BYTE bDummy[3];/ DWORD dwFileSize	<pre>to receive file list (All capitals) files files file to save output directory information. The specified file the alignment type specified with stEasyDirInfo, in the th the return value of pioCount. All characters of the file name wed in capitals. {</pre>	Return value Normal end: 0 Abnormal end: Error code

As a supplement of "bFileTimeStamp" (8 bytes), high-order four bytes indicate time in the MS-DOS format, and loworder 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)

Function	Reading file list from CF card (Type specification)	
	n the CF card inserted in a GP node into a file specified with the p th "sDirectory" can be output.	parameter. Only the file list in
INT WINAPI EasyGetLi sSaveFileName);	stInCard(LPCSTR sNodeName, LPCSTR sDirectory, INT* oiCo	ount, LPCSTR
following directories: LOG (Logging dat TREND (Trend da ALARM (Alarm d CAPTURE (Captu FILE (Filing data) oiCount: Number of outp sSaveFileName: Name or stores binary data of quantity specified v and extension are s struct stEasyDirInf BYTE bFileName[BYTE bFileName] BYTE bExt[3+1];/ BYTE bExt[3+1];/ BYTE bExt[3+1];/ BYTE bFileTimeS BYTE bFileTimeS BYTE bDummy2[};	tory to output list (All capitals) This API supports only the a) ta) ta) ata) ata) re data) ut files f file to save output directory information. The specified file of the alignment type specified with stEasyDirInfo, in the with the return value of pioCount. All characters of the file name aved in capitals. to { 8+1];// File name (Terminated with "0") / File extension (Terminated with "0")];// Dummy ze;// File size tamp[8+1];// File timestamp (Terminated with "0")	Return value Normal end: 0 Abnormal end: Error code
Special Note		
Function	Reading file from CF card (Optional file name specification)	
-	m the CF card. You can specify an optional file to read. eadInCfCard(LPCSTR sNodeName, LPCSTR sFolderName, LP D* odwFileSize);	CSTR sFileName, LPCSTR
single-byte charact	older containing source file to be read from CF card (Up to 32 ers) rce file to be read from CF card (Up to 8.3 format character	Return value Normal end: 0 Abnormal end: Error code
Special Note		1

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 EasyFileReadCard(LPCSTR sNodeName, LPCSTR pReadFileType, WORD wReadFileNo, LPCSTR sWriteFileName, DWORD* odwFileSize);		
1 21 21	node to output file list urce file to be read from CF card (See <special note="">) r of source file to be read from CF card</special>	Return value Normal end: 0 Abnormal end: Error code
odwFileSize: Size of read	CF file	

Special Note

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
Line graph data	ZT	TREND
Sampling	ZS	TREND
Alarm 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
Screen capture	СР	CAPTURE

Data type	File type	Target folder
Filing data	ZF or F	FILE
CSV data	ZR	FILE
Image screen	ZI or I	DATA
Sound data	ZO or O	DATA
Line graph data dedicated to 'GP-Pro EX' (for compatibility)	ZT	TREND
Sampling data dedicated to 'GP-Pro EX' (for compatibility)	ZS	TREND
Alarm 1	Z1 or ZA	ALARM
Alarm 2	Z2 or ZH	ALARM
Alarm 3	Z3 or ZG	ALARM
Alarm 4 to 8	Z4 to Z8	ALARM
Logging data dedicated to 'GP-Pro EX' (for compatibility)	ZL	LOG
Capture data	СР	CAPTURE
Sampling 1 to 64	ZS1 to ZS64	SAMP01 to SAMP64

■File types supported for GP3000 Series node and WinGP node

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.

INT WINAPI EasyFileWriteInCfCard(LPCSTR sNodeName, LPCSTR pReadFileName, LPCSTR sFolderName, LPCSTR sFileName);

Argument	Return value
sNodeName: Name of GP node 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)	
sFileName: Name of target file in CF card (Up to 8.3 format character string)	
Special Note	

Function	Writing file into CF card (Type specification)			
Writes a specified file into	Writes a specified file into the CF card. Only the file type specified with "pWriteFileType" can be written.			
INT WINAPI EasyFileWri wWriteFileNo);	teCard(LPCSTR sNodeName, LPCSTR pReadFileName, LPCS	STR sWriteFileType, WORD		
Argument Return value sNodeName: Name of GP node to write file Normal end: 0 pReadFileName: Name of source file to be written into CF card (Full path) Normal end: 0 sWriteFileType: Type of target file in CF card Abnormal end: Error code (See <special note=""> of the function for "Reading file into CF card (Type specification)") WriteFileNo: File number of target file in CF card</special>				
Special Note				
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.			
INT WINAPI EasyFileDeleteInCfCard(LPCSTR sNodeName, LPCSTR sFolderName, LPCSTR sFileName) ;				
Argument sNodeName: Name of GP 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 care	Deleting file from CF card (Type specification)			
Deletes a specified file f	from the CF card. Only the file	e type specified with "pDe	eleteFileType" can be deleted.		
INT WINAPI EasyFileI	DeleteCard(LPCSTR sNodeNa	nme, LPCSTR pDeleteFile	eType, WORD wDeleteFileNo);		
pDeleteFileType: Type	GP node containing file to be d of file to be deleted from CF c mber to be deleted from CF ca	ard (See <special note="">)</special>	Return value Normal end: 0 Abnormal end: Error code		
ends normally.	bllowing file types. Only the fi		udged as an error, and the processing F card folder can be read.		
Data type	File type	Target folder			
Filing data	ZF	FILE	1		
CSV data	ZR	FILE			
Image screen	ZI	DATA			
			1		

ZF ZR ZI ZO ZT	FILE FILE DATA DATA
ZI ZO	DATA
ZO	2
	DATA
7T	
<i>L</i> 1	TREND
ZS	TREND
Z4 to Z8	ARAM
ZL	LOG
ZG	ALARM
ZH	ALARM
ZA	ALARM
ZC	MRM
СР	CAPTURE
	Z4 to Z8 ZL ZG ZH ZA ZA ZC

Data type	File type	Target folder
Filing data	ZF or F	FILE
CSV data	ZR	FILE
Image screen	ZI or I	DATA
Sound data	ZO or O	DATA
Line graph data dedicated to 'GP-Pro EX' (for compatibility)	ZT	TREND
Sampling data dedicated to 'GP-Pro EX' (for compatibility)	ZS	TREND
Alarm 1	Z1 or ZA	ALARM
Alarm 2	Z2 or ZH	ALARM
Alarm 3	Z3 or ZG	ALARM
Alarm 4 to 8	Z4 to Z8	ALARM
Logging data dedicated to 'GP-Pro EX' (for compatibility)	ZL	LOG
Capture data	СР	CAPTURE
Sampling 1 to 64	ZS1 to ZS64	SAMP01 to SAMP64

Function

Renaming file in CF card

■File types supported for GP3000 Series node and WinGP node

Renames a specified file in the CF card.

INT WINAPI EasyFileRenameInCfCard(LPCSTR sNodeName, LPCSTR sFolderName, LPCSTR sFileRename);

Argument Return value sNodeName: Name of GP node to write file Normal end: 0 sFolderName: Name of folder containing file to be renamed in CF card (Up to 32 single-byte characters) Abnormal end: Error code sFileName: Name to file to be renamed in CF card (Up to 8.3 format character string) SfileRename: New file name (Up to 8.3 format character string) Special Note Special Note

Function	Acquiring information on CF card empty space			
Acquires information on empty space in the CF card connected to a specified entry node.				
INT WINAPI EasyGetCfFreeSpace(LPCSTR sNodeName, INT* oiUnallocated);				
Argument	Return value			
sNodeName: Name of GP	1	Normal end: 0		
oiUnallocated: Empty space	e in CF card (number of bytes)	Abnormal end: Error code		
Special Note				
Function	Function FTP passive mode setup			
'Pro-Server EX' uses a spec	ial protocol to access the CF card in a GP Series node. However	er, to access a GP3000 Series		
node and WinGP node, FTP protocol is used.				
For FTP protocol, 'Pro-Serv	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		Return value		
iPassive: (In) 0: Normal mode		Normal end: 0		
Other than 0: Passive mode Abnormal end: Error				
At initialization of DroEasy, the ETD protocol is set to "Normal mode"				
At initialization of ProEasy, the FTP protocol is set to "Normal mode".				
Special Note				

26.8 Other APIs

Function	Reading time from GP (DWORD-type)			
Acquires current time words from LS2048.	e of a specified node as a DWORD-type value. This function is valid	only for the time saved in 6		
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	Reading time from GP (VARIANT-type)			
Acquires current time words from LS2048.	e of a specified node as a Variant-type value. This function is valid on	ly for the time saved in 6		
DWORD WINAPI E	asyGetGPTimeVariant(LPCSTR sNodeName, LPVARIANT ovTime);		
Argument sNodeName: Name of target node (A Pro-Server EX node cannot be specified.) ovTime: Acquired time (Time is acquired as a value of VARIANT type. Internal possessing format is "Date".) Return value				
Special Note				
Function	Reading time from GP (STRING-type)			
saved in 6 words from	e of a specified node as an LPTSTR-type character string. This function 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.

Acquires current tim	Reading time from GP (STRING VARIANT-type)	
saved in 6 words from	e of a specified node as a Variant-type character string. This function n LS2048.	is valid only for the time
DWORD WINAPI E	asyGetGPTimeStringVariant(LPCSTR sNodeName, LPCSTR sForm	nat, LPVARIANT ovTime);
pFormat: String to sp specification c listed below. C <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 sta Single INT WINAPI GetNo osSystemVersion,LP Multi INT WINAPI GetNo osGPType,LPSTR os	GP node status. Since the response time-out value can be changed, th tus. deProperty(LPCSTR sNodeName,DWORD dwTimeLimit,LPSTR o STR osComVersion,LPSTR osECOMVersion); dePropertyM(HANDLE hProServer,LPCSTR sNodeName,DWORD SystemVersion,LPSTR osComVersion,LPSTR osECOMVersion);	sGPType,LPSTR
dwTimeLimit: (In) R (If "0" is speci	Server handle me of GP 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)	Return value Normal end: 0 Abnormal end: Error code
	is information on the target node to the following area. east 32 bytes for each item.	

Function	Acquiring symbol/group byte size				
Acquires the total number of bytes required to access a device symbol or group symbol.					
INT WINAPI SizeOfSymbol(LPCSTR sNodeName,LPCSTR sSymbolName,INT* oiByteSize);					
Argument sNodeName: (In) Na sSymbolName: (In) N oiByteSize: (Out) By	Return value Normal end: 0 Abnormal end: Error code				
Special Note For "sSymbolName", a device symbol, non-alignment group, whole alignment group, or an element of alignment group can be specified.					
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);			
Argument sNodeName: (In) Name of entry node with Device/PLC name sSymbolName: (In) Name of target group symbol or symbol sheet oiCountOfMember: (Out) Number of members acquiredReturn value Normal end: 0 Abnormal end: Error co Error co Normal end: Error co Normal end: Error co Error co Error co					
	Special Note When a group symbol exists in a specified group symbol, the number of members is counted as one, even if multiple device symbols exist in the inner group symbol.				
Function	Function Acquiring symbol/group/symbol sheet definition information				
Acquires definition in	nformation (data type, data quantity, etc.)				
	mbolInformation(LPCSTR sNodeName,LPCSTR sSymbolName,INT IMember,LPSTR osSymbolSheetName,SymbolInformation* oSymbo IMember);				
Argument sNodeName: (In) Na sSymbolName: (In) N iMaxCountOfSymbo information + Specify the nur osSymbolSheetName symbol Specifi oSymbolInformation structure. Prepare work f oiGotCountOfSymbo returned to oSy	Return value Normal end: 0 Abnormal end: Error code				

Special Note

Structure of SymbolInformation
struct SymbolInformation

WORDm_wAppKind;// Data type, Symbol: 1 to 12, 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.

26.9 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

(2) Data types available in special cases

Definition name	Decimal value	Hexadecim al 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) GP3000 Series nodes, WinGP nodes and LT3000 nodes can be connected to multiple devices/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 a device incorporated in a GP3000 Series node/WinGP node/LT3000 node or Pro-Server EX node, specify "#INTERNAL" as the Device/PLC name. (It can be omitted.)

(4) To access the memory of a memory link driver in a GP3000 Series node/WinGP node/LT3000 node, 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) To access a Device/PLC assigned to an internal device or "system area device" in a GP3000 Series node/ WinGP node/LT3000 node, you can omit specification of the Device/PLC name by specifying an entry node name with 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.

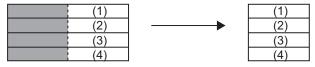
(Ex.) When devices for 20 bits are registered in series from LS123401, the processing speed becomes higher if they are registered in two words from LS1234.

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

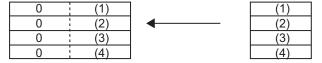
(1) When a 16-bit symbol is assigned to a device with a physically 32-bit width, and the device is accessed with the 16-bit symbol, or when 16-bit data type is directly specified to access a 32-bit device, 'Pro-Server EX' can handle the 32-bit device as a 16-bit device.

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()

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.

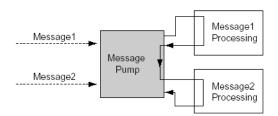
In this manual, the part which receives messages in order and branches into each process (corresponding to DoEvents for VB, or the part executing GetMessage() and DispatchMessage() for VC) is called the message pump. The message pump is not much recognized because it is hidden in the VC or VB framework 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.

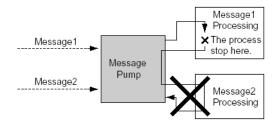
Example) Assume that messages are sent from Windows in the order of message 1 to message 2.

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

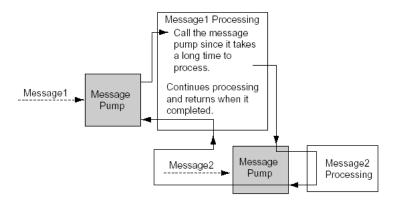
Then, when the message pump recovers from above, it takes out the following message (message 2) and calls the subroutine for message 2.



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

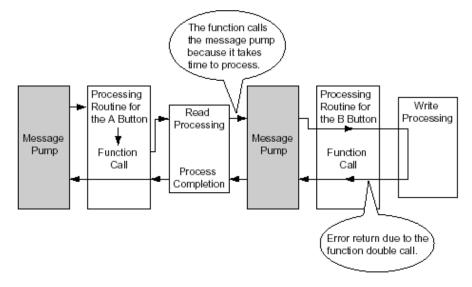
'Pro-Server EX API' prohibits another communication while communicating with a party (while calling a 'Pro-Server EX' function)(double-call). (Double-call is enabled if using the Multi-Handle. For details, refer to the section of Multi-Handle.) However, as 'Pro-Server EX API' runs the message pump inside 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 Sample 1 ()
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 IErr As Long
IErr = ReadDeviceStr ("GP1", "LS100", strData, 10)
If IErr <> 0 Then
MsgBox "Read Error = " & IErr
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 GP 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
```

26.10 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.

- To "26.10.1 VB Support Function"
- "26.10.2 VC Support Function"
- "26.10.3 VB .NET Support Function"
- "26.10.4 C# .NET Support Function"

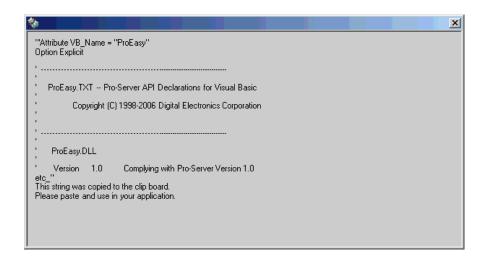
26.10.1 VB Support Function

VB: Declaration statement

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

🎕 Pro-Studio EX	2.прх						_ 🗆 🗙
File Edit Tool	Programming Assist Settin	ng Help					
Start .	VC++ Write F	Statement	Feature 🔉 📔	- Sa	we > 🆄 Ti	ransfer	Monitor Status
Symbol	EXCEL Read F	Node Name AGP1		Device	Name PLC1		
Group	Ungroup	Sheet Name Sheet3		Set it as	a global symbol shee	et.	
Insert	Delete						
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	I
Check Duplicat	ion/List Used Addresses					1	
Global Con	stant Setting Screen					1	

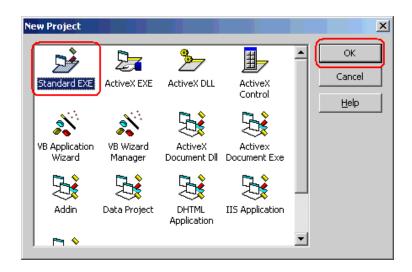
The VB declaration statement is coped to the clipboard.



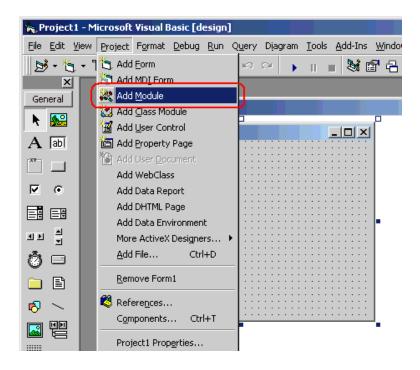
2 Start Microsoft Visual Basic, and select [New Project] from [File] on the menu.

1	icrosoft \	/isua	Bas	ic					
	Edit Viev	v Pro	oject	Format	<u>D</u> ebug	<u>R</u> un	Query	Djagran	
	<u>N</u> ew Projec	t	Ctrl	+N	Pa (2 /4		ο j	
7	Open Proje	ect	Ctrl	+0					
	A <u>d</u> d Projec	t							
	<u>R</u> emove Pr	oject							
	Sa <u>v</u> e Proje	ct							
	Sav <u>e</u> Proje	ct As,							
	Save		Ctr	I+S					
	Save As								
	Save Selec	tion							
	Save C <u>h</u> an	ge Sa	ript						
5	Print		Ctr	+P					
D	Print Setyp								
	Make								
	Make Proje	ct <u>G</u> ro	up.,						
	E <u>×</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.

🐂 Pi	oje	t1 - M	licrosoft	Visual I	Basic [d	esign]			
Eile	<u>E</u> dit	<u>V</u> iew	Project	F <u>o</u> rmat	<u>D</u> ebug	<u>R</u> un	Query	Dįag	gram	<u>T</u> ools
	K)	Can't U	ndo		Ctrl+Z	鸿	E)	$\mathbf{\alpha}$	•	П
	C ^a	<u>R</u> edo P	aste						_	_
Ger	X	Cu <u>t</u>			Ctrl+X	m	<u>, </u>			
k	Ē.	- Сору			Ctrl+C		′ (Code)	· · ·		
	æ	<u>P</u> aste			Ctrl+V		(code,	, 	_	
		Paste L	in <u>k</u>							
		<u>R</u> emov	e							
N		<u>D</u> elete			Del					
		Delete	Table from	n Data <u>b</u> a	ise					
		Select <u>(</u>	<u>A</u> ll		Ctrl+A					
٩Þ		Select (<u>A</u> ll Column	15						
A.R.		Tabla								

The deceleration statement is now pasted.

🖧 Project 1 - M	lodule1 (Code)	
(General)		▼ (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-CI	ARD access	control API
Declare	Function	EasyFileSetPassiveMode Lib "ProEasy.DLL"
		•

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

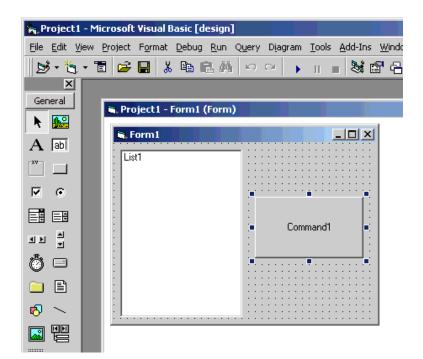
This section describes the procedure for creating an application that reads and displays data (16-bit signed data) for three points with a click on [Command1].

🖹 Form1	
1111 2222 3333	Command1

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

File Edit View	Project Format Debug Run Query Diagram Tools Add-Ins Windo
	E 😕 🖬 📽 🛤 🗠 ↔ → 🗉 = 😻 🗳 🚭
General	🖷 Project1 - Form1 (Form)
▲ 🔛	S. Form1
	List1
• •	
Ö 🗆	

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 ≫ 葇	Feature ⋗ 📑	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.			
Copy Cut Paste	Symbol	Data Type	Consec utive	Device Address	No. of Data	Comment		
	_D0040_WORD	16Bit(Signed)	uuve	D0040	3			
Symbol Sheet		,,	<u> </u>		1			
Add Delete					1			
Check Duplication/List Used Addresses					1			
Check Dupication/List Osed Addresses					1			
Global Constant Setting Screen					1			
					1			
E - ₩ PC1 (192.168.0.1)					1			
#INTERNAL:Sheet1			<u> </u>		1			
E ← GP3000 Series			<u> </u>		1			
			<u> </u>		1			
PLC1:Sheet3 A Series CPU [<u> </u>		1			
→ WinGP			<u> </u>		1			
► LT3000					1	II		
GP Series Global Symbol			<u> </u>		1	II		
			<u> </u>		1	i II		
		, 	<u> </u>	,	1	II		
				,	1	·		
					1			
	•	•		•				

 $10~{\rm Select}~{\rm [Programming~Assist]}$ - [VB & VBA] - [Read Function] on the menu.

🎕 Pro-Studio EX	(2.npx					
File Edit Tool	Programming Assist	Setting	g Help			
Start .	VC++ •	Declare Write Fu Read Fu	nction		Feature ≫ 📑	
Symbol			Node Na	AGP1		
Group	Group Ungroup			me Sheet3		
Insert	Insert Delete			,		
Сору	Copy Cut Paste		Symbol		Data Type	
Symbol Sheet			_D0040_W	/ORD	16Bit(Signed)	
Add	Delete		<u> </u>		<u> </u>	
Check Duplicat	ses					
Global Con	istant Setting Screen					

X

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.

٥.

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

🖉 Project1 - Form1 (Code)		_ 🗆 ×
Command1	Click	•
Private Sub Command1_Click() lErr = ReadSymbolVariant("A) GP1.PLC1","_DOO40_WORD",Enter a variant-typ	pe 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.

📕 Proj	ject1 - Form1 (Code)	
Com	mand1 Click	•
	rivate Sub Command1_Click() im wData As Variant	1
1	Err = ReadSymbolVariant("AGP1.PLC1","_D0040_WORD",Enter a variant-type variable her	:e)
E	nd Sub	
		• • //

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

🖉 Project1 - Form1 (Code)			
Command1		Click		•
Private Sub Cor Dim wData As Va				-
lErr = ReadSymk	oolVariant("AGP1.PL	.C1", "_DOO40_WORI)", <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		<u> </u>
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	- Microsoft Visual Basic [design]
<u>File E</u> dit <u>Vi</u> e	ew Project Format Debug Run Query Diagram Tools Add-Ins Window Help
🛃 • 🏷	🝷 🛅 😅 🔚 🐰 🗈 👔 🗩 Start 👘 F5 🔤 🖶 😵 🛠 🔂 🐊 🛛 Ln 11, Col 1
×	Start With Full Compile Ctrl+F5
General	II Break Ctrl+Break
	Project1 - Form1 (
	Form1 Restart Shift+F5
A abl	
	Project1 - Form1 (Code)
	Command1 Click
	Private Sub Command1 Click()
	Dim wData As Variant
∢► ₹	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

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		Comn	nand1
3030			

 $17\ {\rm Select}\ [{\rm TextBox}]\ {\rm and}\ {\rm paste}\ {\rm it}\ {\rm to}\ [{\rm Form1}].$ Paste [Text Box] for three items.

🙀 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 Query Diagram <u>T</u> ools <u>A</u> dd-Ins <u>W</u> indow <u>H</u> elp
🛛 😼 • 🍓	・ 盲 😂 🖬 👗 陥 ඬ 桷 🗠 ⇔ 🕞 🔳 😻 🖆 名 😽 🛠
×	
General	🖷 Project1 - Form1 (Form)
N 🔛	
A abl	Sector Se
	· · · · · · · · · · · · · · · · · · ·
	Text1
• •	·····
	Text2
의 N	
å 🗆	Text3
90	
👩 🔨	
् ।	
010	

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

🐂 Project1 -	Microsoft Visual Basic [design]
<u>File E</u> dit <u>Vier</u>	w <u>P</u> roject F <u>o</u> rmat <u>D</u> ebug <u>R</u> un Query Diagram <u>T</u> ools <u>A</u> dd-Ins <u>W</u> indow <u>H</u> e
ج 🛃 - 🛃	1월 😅 🔒 👗 🛍 💼 🏘 🕬 🔿 🕠 👔 🔳 😻 🖆 🖴 🦉 :
×	
General	🖷 Project1 - Form1 (Form)
N 🔛	S. Form1
A abl	
XV-	Text1
	Text2
	Command1
4 D 🖉	
Ö 🗆	Text3
õ 🗈 🛛	
R >	
OLE	

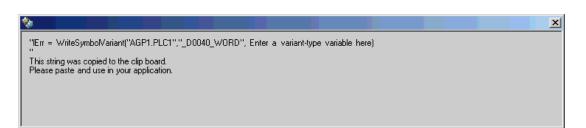
19 Select a target symbol name from those registered in 'Pro-Server EX'. (Select the symbol with first-address for writting.)

💱 Pro-Studio EX 2.npx						
File Edit Tool Programming Assist Setting	ı Help					
📁 Start ン 🐚 Node ン	🍐 Symbol ≫ 🍦	Feature ⋗ 📑	Sa	ave >> 🔖 T	ransfer	Monitor Status
Symbol	Node Name AGP1		Device	Name PLC1		
Group Ungroup	Sheet Name Sheet3		Set it as	a global symbol shee	ət.	
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	
					1	I I
Global Constant Setting Screen					1	
⊡▼ Pro-Server EX			<u> </u>		1	
È₩ PC1 (192.168.0.1)			<u> </u>		1	
INTERNAL:Sheet1			<u> </u>		1	
È ▼ GP3000 Series È[] AGP1 (192.168.0.100)					1	
			<u> </u>		1	
PLC1:Sheet3 A Series CPU [<u> </u>		1	
• WinGP			<u> </u>		1	
• LT3000			<u> </u>		1	
→ GP Series → Global Symbol			<u> </u>		1	
····· 🖡 Global Symbol			<u> </u>		1	
	[<u> </u>		1	III
					1	
			<u> </u>		1	
	•				P.	

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

۹	Pro-Stu	udio EX	(2.npx									
File	Edit	Tool	Programm	ing Assist	Settir	ng	Help					
2	3	Start	VB & VB	3A 🔸 🔜	Declare	e St	atement		Eastura		- c-	ive ≫ 👔
	-	start .	VC++	<u> </u>	Write F				≷ Feature ン		30	
			EXCEL	· •	Read F	uno	ction			_		
5	ymbol					Г	Node Nam	e AGP1			Device	Name PLC1
	G	iroup		Ungroup		L	Sheet Nam	- Sheet3			Satitae :	a global sym
	h	nsert		Delete		L	Sheet Nam	eleneere				a giobai syin
	Cop	y	Cut	Paste			Symt	ool	Data Type		Consec utive	Device Ac
-9	ymbol S	Sheet -					_D0040_WC	IRD	16Bit(Signed)			D0040
		Add		Delete		L						
							<u> </u>					
	Check I	Duplica	tion/List Us	ed Addres:	ses	L				_		
	Glo	hal Cor	stant Settin	a Screen								

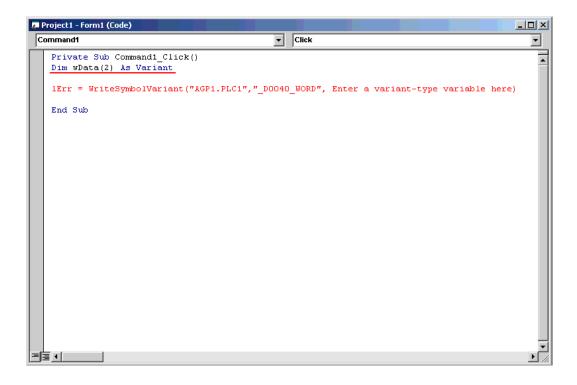
The write function is copied to the clipboard.



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

Project1 - Form1 (Code)	_ 🗆 ×
Command1 Click	•
Private Sub Command1_Click() lErr = WriteSymbolVariant("AGP1.PLC1","_D0040_WORD", Enter a variant-type variable }	here)
End Sub	
	_
	Þ

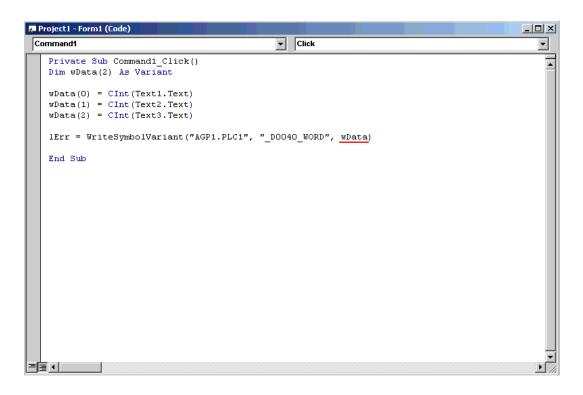
22 Declare the area (alignment) to store the written data. Ensure that the alignment type (in this example, Variant-type) is matched with the data type of the symbol being used.



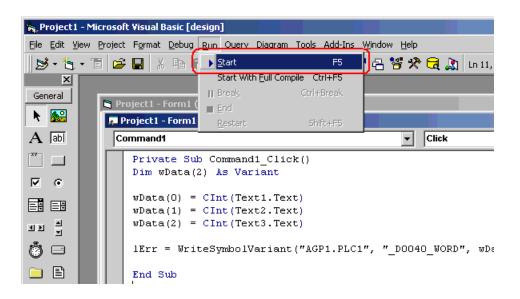
 $23\,$ Set the data entered in [TextBox] into the alignment.

Project1 - Form1 (Code)			
ommand1	•	Click	-
Private Sub Command1_Click() Dim wData(2) As Variant			
<pre>wData(0) = CInt(Text1.Text) wData(1) = CInt(Text2.Text)</pre>			
wData(2) = CInt(Text3.Text)			
lErr = WriteSymbolVariant("AGP1.P	LC1","_D0040	_WORD", Enter a variant-t	ype variable here)
End Sub			
Life Seb			

 $24 \ \ \text{Specify the first area (wData) where the written data has been set.}$



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



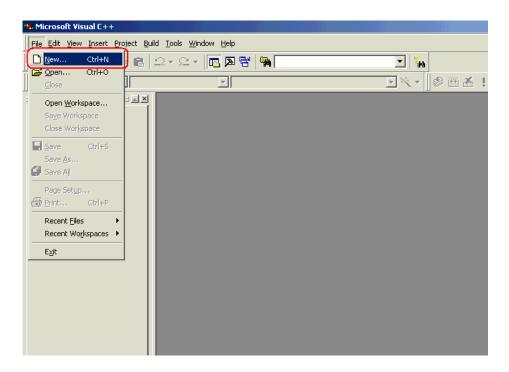
26 After entering values (for three points) in [TextBox], click [Command1]. Then, 'Pro-Server EX' executes the writing of the data for three points from the symbol "_D0040_WORD".

🛢 Form1			- D X
1010		ſ	
2020		Comma	and1
3030			

26.10.2 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" is entered for [Location].

New	<u>?</u> ×
Files Projects Workspaces Other Documents	
 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 Dynamic-Link Library Win32 Static Library 	Project <u>name:</u> Sample Logation: C:\PROGRAM FILES\PRO-FAC Create new workspace Add to current workspace Dependency of: Platforms: Wwin32
	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/⊻jew architecture support? What Janguage would you like your resources in? English [United States] (APPwZENU.DLL ▼
< <u>B</u> ack	<u>N</u> ext > <u>Finish</u> Cancel

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

New Project Information
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: CSampleDlg in SampleDlg.h and SampleDlg.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.

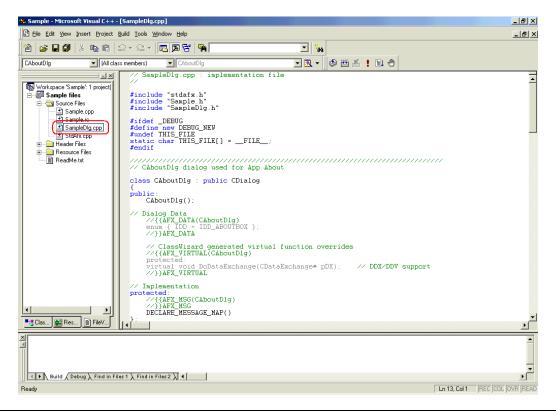
4	o Sa	mple	- Mici	rosoft V	'isual C+	+-				
I	<u>F</u> ile	<u>E</u> dit	<u>V</u> iew	Insert	<u>P</u> roject	<u>B</u> uild	Layout	<u>T</u> ools	<u>W</u> indow	Help
Ī	1			7 %		Acti <u>v</u> e To Pr	Project		+ +	
	JUSa	mpleD	Ig		Dep	enden	cies			SampleDlg
1	_	_		(ings		ρ	lt+F7	PLE_DIALOG (Dialog)
				e resou	Exp	ort <u>M</u> a	kefile			
		÷	📄 Dial 📄 Icor	-	Inse	ert Pro	ject into ^v	Worksp	ace	······
			Strir	ng Table					το	DD: 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.

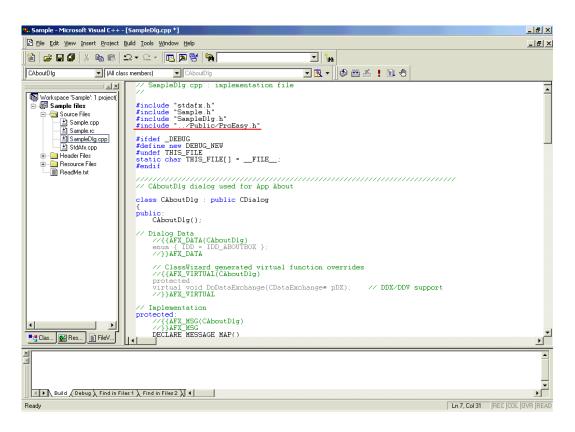
Project Settings	<u>?</u> ×
Settings For: Win32 Debug	General Debug C/C++ Link Resource (
e-f ≝ Sample	Category: General
	✓ Link incrementally
	Enable profiling
	Project Options:
	/nologo /subsystem:windows /incremental.yes /pdb:"Debug/Sample.pdb" /debug /machine:1386 /out:"Debug/Sample.exe" /pdbtype:sept
	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		×
1111 2222 3333	Button1	OK Cancel

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.

% Sample - Microsoft Visual C++ - [Sample.rc - IDD_SAMPLE_DIALOG (Dialog)]			_ 🖪 ×
🔄 Elle Edit View Insert Broject Build Layout Iools Window Help		l	_ 8 ×
1월 1			
CSampleDlg 💽 (Al class members) 🔽 💊 CSampleDlg 🔽 🕄 🗟 🖬 🔏 🚦 🕤			
Sample resurces in Ino. ABRILITROX TOD. SAMPLE DIE String Table TOD.: Place dialog controls here. TOD:: Place dialog controls here. TOD:: Place dialog controls here.			
9 評 詞 砰 趈 ❶ 凾 洲 苫 ━ □ 凾 🛱			
X J Build (Debug) Find in Files 1) Find in Files 2] 4			• •
Ready	50, 90	1 [™] 200×8	READ

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

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

Customize		<u>?</u> ×
Commands Toolbars Tools Loolbars: Very Menu bar Standard Build Build Build MiniBar ATL Resource Dialog Controls	Keyboard Add-ins and I ✓ Show ToolTips ✓ With shortcut <u>k</u> eys ✓ Large buttons	
Toolbar name:	1	Close

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

🦝 Sample - Microsoft ¥isual C++ - [San		
Ele Edit View Insert Project Build		
11		
12 🖌 🖬 🕼 👗 🖻 🖻 🗠 ד		
CSampleDig IDC_LIST1	🔽 LBN_SELCHANGE 💽 💽 🗸 🚽 🖄	
	In the standard matrix to the standard matrix to the	
Sample resources *	Sample	Con× ▲ ▲ ▲ ▲ ● ●

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

Image: Selection: Selection: Single Image: Sort Owner draw: Sort Image: Sort Image: Sort No Image: No integral height	List Box	Properties						×
Single Border Horizontal scroll Want key input Owner draw: Sort Vertical scroll Disable no scroll No No redraw No integral height 	-¤ ?	General	Styles	Extended Styles				
☐ Has strings ☐ Multi-column ☐ Use tabstops	Singl	e 🔽 🔽 r draw:	Sort Notify	✓ Vertical scroll No redraw	Γ	<u>D</u> isable no	o scroll	ıt

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

🐝 Sample - N	1icrosoft Visual C++ - [Sample	e.rc - IDD_SAMPLE_DIALOG (Dialog)]
Eile Edit	View Insert Project Build Lay	out <u>T</u> ools <u>W</u> indow <u>H</u> elp
12 🖙 🖬	Class <u>W</u> izard Ctrl+W	- 🖪 🗖 🗟 🦬
CSampleDIg	ID= Resource Symbols Resource I <u>n</u> cludes	LBN_SELCHANGE
🖃 🚖 Sar	🖹 Full Screen	
	Wor <u>k</u> space Alt+0	ample

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

MFC ClassWizard					? ×
Message Maps	Member Variables	Automation	ActiveX Events	Class Info	
Project:		Class <u>n</u> am	e:		Add Class 👻
Sample		CSample[-	-	Add Variable
Control IDs:	.SampleDlg.h, C:\\VC	\Sample\Samp Tune	leDig.cpp Member		Delete Variable
IDC LIST1					Update <u>C</u> olumns
IDOK					Bind All
Description:					
				OK	Cancel

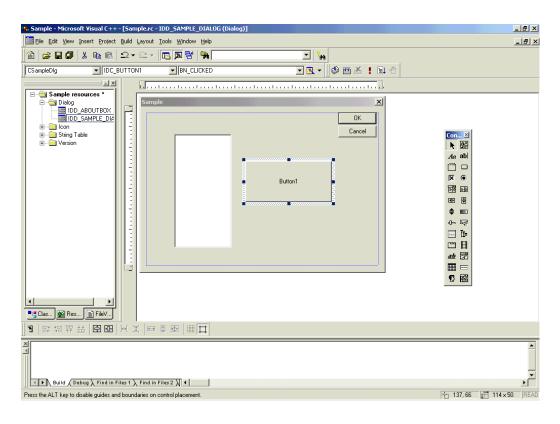
16 Click [Add Variable], and enter "m_List" for [Member variable name]. After selecting "Control" for [Category], click the [OK] button.

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.

FC ClassWizard					?
Message Maps	Member Variables	Automation	ActiveX Events	Class Info	
Project:	Class <u>n</u> ame:				Add Class 🔻
Sample	-	CSample	Dig	•	Add Variable
C:\\VC\Sample\	SampleDlg.h, C:\\VC	:\Sample\Samp	leDlg.cpp		
Control <u>I</u> Ds:		Туре	Member		<u>D</u> elete Variable
IDC LIST1 IDCANCEL IDOK		CListBox	m List		Update <u>C</u> olumns <u>B</u> ind All
Description: m	ap to CListBox member				

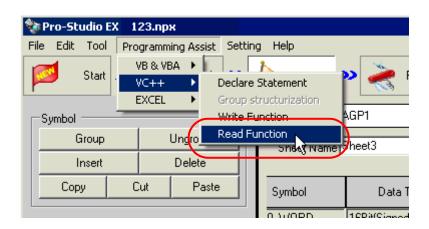
18 Select [Button], and paste it to the dialog box.



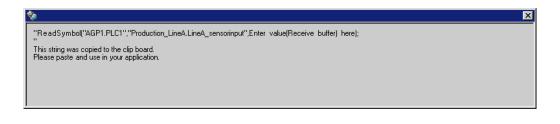
19 Select a target symbol name from those registered in 'Pro-Studio EX'. (Select the symbol with first-adress for reading.)

Pro-Studio EX 2.1 Edit Tool Prog	n px ramming Assist Sett	tina Help					
🔰 Start ≫	Node >		≷ Feature ນ [n Sa	ave 渊 太 T	ransfer	Mon Sta
Symbol		Node Name AGP1		Device	Name PLC1		
Group	Ungroup	Sheet Name Sheet	З Г	Set it as	a global symbol shee	et.	
Insert	Delete		_	-			
Copy Cu	it Paste	Symbol	Data Type	Consec utive	Device Address	No. of Data	Commer
Symbol Sheet		_D0040_WORD	16Bit(Signed)		D0040	3	
Add	Delete					1	
						1	
Check Duplication/L	ist Used Addresses					1	
						1	
Global Constant	Setting Screen					1	
Pro-Server EX		1		<u> </u>	<u> </u>	1	
🖻 🐙 PC1 (192.1	68.0.1)			<u> </u>		1	
🔤 📰 🗰	RNAL:Sheet1		_	<u> </u>		1	
 GP3000 Series 				<u> </u>		1	
🚊 🔄 AGP1 (192	:168.0.100) RNAL:Sheet2					1	
	Sheet3 A Series CPU I	{					
► WinGP		11				1	
··· ► LT3000			_			1	
► GP Series			_			1	
> Global Symbol			_			1	
			_			1	
					 	1	
						1	
1			1	1	1	P.	

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



The read function is copied to the clipboard.



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

😘 Sample - Microsoft Visual C++ - [Sample.rc -]	IDD_SAMPLE_DIALOG (Dia	log)]		
Eile Edit View Insert Project Build Layout	<u>T</u> ools <u>W</u> indow <u>H</u> elp			
🎦 😅 🖬 🕼 🗼 🖻 💼 🗠 + 🗠 +	🖪 🗖 😤 🀐	~ 349		
CSampleDIg (All class members)	💌 💊 CSampleDIg	- 🗷 -	🕸 🏙 👗 🖠 🕲 🖉	
Sample resources *		Butten1	X OK Cancel	X X X X Aa Abl Image: Constraint of the state o

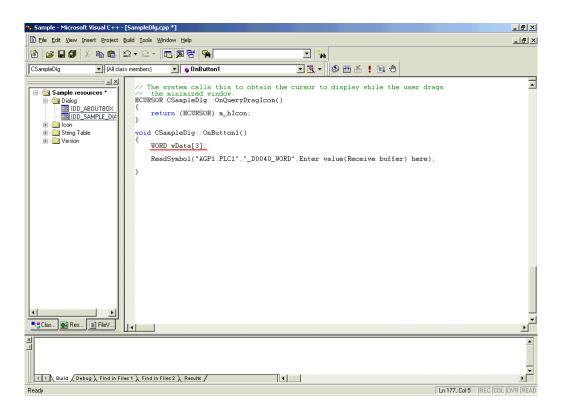
22 Click the [OK] button.

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

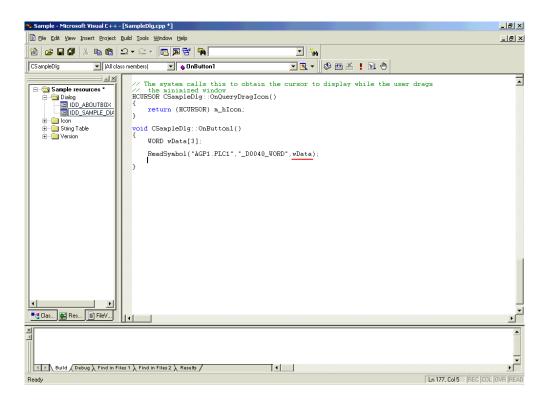
23 Paste the data on the clipboard (read function) into the OnButton1 member function.

9. Sample – Microsoft Visual C++ - [SampleDig.cpp *]	_ 8 ×
Ele Edit View Insert Broject Build Iools Window Help	_ 8 ×
CSampleDig 🔄 (Al class members) 🔄 💊 OnButton1 🔄 🔍 🗸 🕼 🖑	
The system calls this to obtain the cursor to display while the user drags The system calls this to obtain the cursor to display while the user drags the minimized yindow Blob ABOUTBOX return (HCURSOR)m_hIcon; Constraint (HCURSOR) m_hIcon; void CSampleDlg::OnButton1() ReadSymbol("AGP1.PIC1", "_D0040_WORD", Enter value(Receive buffer) here); }	
	-
Build / Debug & Find in Files 1 & Find in Files 2 & Results /	-
Control C	

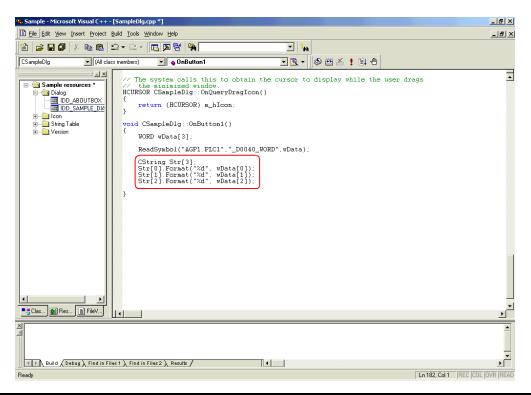
 $24 \ \ \text{Declare the area} \ (\text{Array}) \ \text{to store the read data}.$



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 The list box (m_List) displays the read data (that has been converted into string data) in sequence.

🗱 Sample - Microsoft ¥isual C++	- [SampleDlg.cpp]	_ 8 ×
Eile Edit View Insert Project	<u>Build T</u> ools <u>Window</u> Help	_ 8 ×
12 😅 🖬 🕼 🐇 🖻 🔞		
CSampleDIg (All cli	ass members) 🔽 💊 OnButton1 🔽 👿 🗸 🕼 🍪 🖽 🔏 🗜 🗒 🖑	
Sample resources Sample resources Dialog DD_ABOUTBOX DD_ABUTBOX B-DI Loon B-DI String Table B Version	<pre>// The system calls this to obtain the cursor to display while the user drags // the minimized window. HCURSOR CSampleDlg::OnQueryDragIcon() { return (HCURSOR) m_hIcon; } void CSampleDlg::OnButton1() { WORD vData[3]; ReadSymbol("AGP1.PIC1", "_D0040_WORD", vData); CString Str[3]; Str[0].Format("Xd", vData[0]); Str[1].Format("Xd", vData[2]); Str[1].Format("Xd", vData[2]); m_List_InsertString(0.Str[0]); m_List_InsertString(2.Str[2]); } </pre>	-
Clas Kes FileV		
×		•
	iles 1 ∑ Find in Files 2 ∑ Results /	Ŀ
Ready	Ln 186, Col 1 REC COL	OVR READ

🐝 Sample - Microsoft Visual C++ ·	- [SampleDlg.cpp]	
Eile Edit View Insert Project	Build Tools Window Help	
`````   🖨   X 🗈 🖻	Compile SampleDig.cpp Ctrl+F7 Build Sample.exe F7 Rebuild All Batch Build Clean Start Debug Debugger Remote Cognection E Execute Sample.exe Ctrl+F5 Set Active Cgnfiguration Configurations Profile	<pre>s to obtain the w. ueryDragIcon() hIcon; n1() C1","_D0040_WORI</pre>
	CString Str[3]; Str[0].Format("%d", Str[1].Format("%d", Str[2].Format("%d", m_List.InsertString m_List.InsertString m_List.InsertString }	wData[1]); wData[2]); g(0,Str[0]); g(1,Str[1]);

29 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?

**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

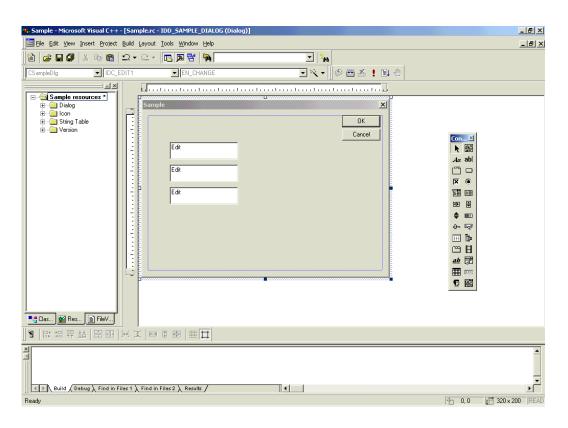
#### Creating "Writing" application

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

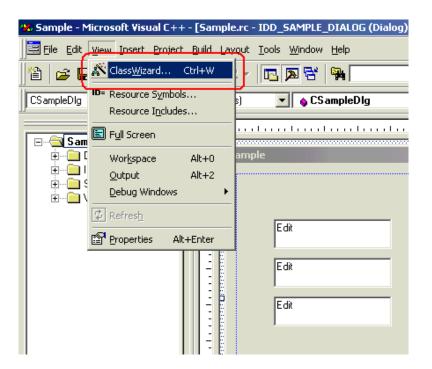
Steps 9 to 11 are the same as those for creating "Reading" application.

🛃 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** Select "IDC_EDIT1" for [Control IDs] in the [Member Variables] tab, and click the [Add Variable] button.

Μ	IFC ClassWizard					? ×
	Message Maps	Member Variables	Automation	ActiveX Events	Class Info	
	<u>P</u> roject:		Class <u>n</u> am	e:		Add Class 👻
	Sample		CSample	-	•	Add Variable
		SampleDlg.h, C:\\V(				ليتسب
	Control IDs: IDC EDIT1		Туре	Member		<u>D</u> elete Variable
	HDC_EDIT2				J	Update <u>C</u> olumns
	IDC_EDIT3					Bind All
	IDOK					
	Description:					
	D coonpliant.					
					04	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 Variable	<u>? ×</u>
Member variable <u>n</u> ame:	ОК
m_Edit1	Cancel
Category:	
Value	
Variable <u>t</u> ype:	
short	
Description:	
short with range validation	

35 Click the [OK] button.

MFC ClassWizard					? ×
Message Maps	Member Variables	Automation	ActiveX Events	Class Info	
<u>P</u> roject:		Class <u>n</u> am			Add Class 🔻
Sample	SamalaDia h. C.V. VVC	CSample[	-	<b>_</b>	Add Variable
Control IDs:	SampleDlg.h, C:\\VC	, vsampievsamp Type	Member		Delete Variable
IDC_EDIT1		short short	m_Edit1 m_Edit2		Update <u>C</u> olumns
IDC EDIT3 IDCANCEL		short	m Edit3		<u>B</u> ind All
IDOK					
Description: sł	nort with range validatio	n 			
<u>M</u> inimum Value:					
Ma <u>x</u> imum Value:					
				ОК	Cancel

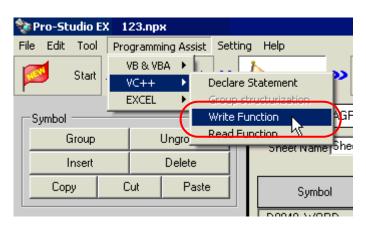
**36** Select [Button], and paste it to [Dialog].

% Sample - Microsoft Visual C++ - [Sample.rc - IDD_SAMPLE_DIALOG (Dialog)]		L	. 8 ×
Ele Edit View Insert Project Build Layout Iools Window Help		5	- 8 ×
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1			
CSampleDig 🔽 IDC_BUTTON1 🔽 BN_CLICKED 🔽 🔯 🖉 🔏 ! 🖬 🖑			
Sample resources *	Con		
)¶  ☞ 눼 쮸 쳘  ▣ ▣  땐 봄  ፡፡፡ ▣ ▣			
Build (Debug), Find in Files 2)			•
Press the ALT key to disable guides and boundaries on control placement.	137,66	114 x 50	READ

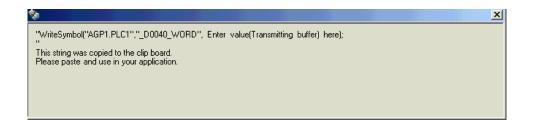
37 Select a target symbol name from those registered in 'Pro-Studio EX'. (Select the symbol with first-adress for writting.)

🂱 Pro-Studio EX 2.npx						_ 🗆 🗙
File Edit Tool Programming Assist Settin	ng Help					
Start >> 🔪 Node >>	녿 Symbol ⋗ 🄾	Feature ン 📑	- 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	
· · · · · · · · · · · · · · · · · · ·			<u> </u>		1	
Check Duplication/List Used Addresses			<u> </u>		1	
Global Constant Setting Screen			<u> </u>		1	
Giobal Constant Setting Screen			<u> </u>		1	
Pro-Server EX		,	<u> </u>		1	II
E E PC1 (192.168.0.1)					1	
Er ▼ GP3000 Series					1	
🖻 🖃 AGP1 (192.168.0.100)					1	
#INTERNAL:Sheet2					1	
PLC1:Sheet3 A Series CPU I					1	
• LT3000		<u> </u>			1	
• GP Series		<u> </u>	<u> </u>		1	
Global Symbol			<u> </u>		1	
			<u> </u>		1	
			<u> </u>		1	
		 	<u> </u>		1	
	•	1	1		P	

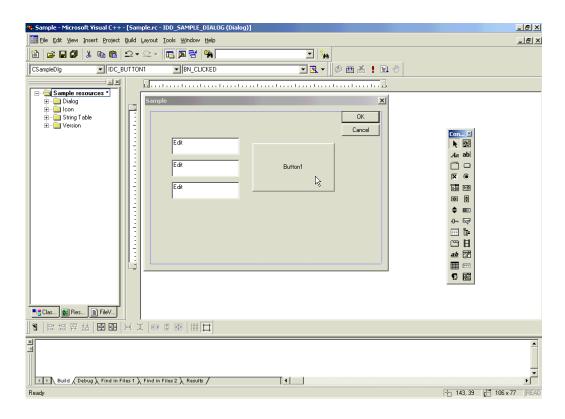
38 Select [Programming Assist] - [VC++] - [Write Function] on the menu.



The write function is copied to the clipboard.



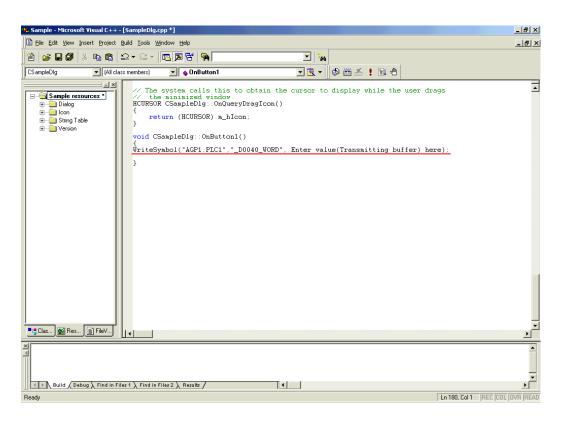
**39** Double-click [Button1] that has been pasted to [Dialog] in Microsoft Visual C++.



40 Click the [OK] 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	

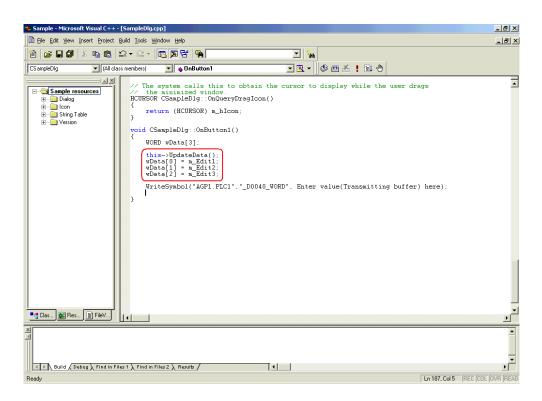
**41** Paste the data on the clipboard (write function) into the OnButton1 member function.



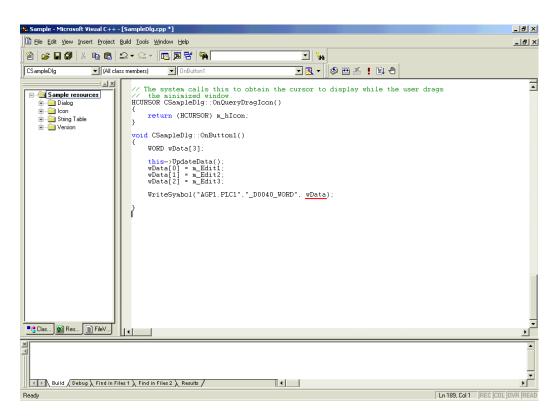
42 Declare the area (Array) to store the write data. For three or more writing points, specify three or more array elements.

Sample - Microsoft Visual C++ - [SampleDig.cpp *]	Ð×
Ele Edit View Insert Project Build Icols Window Help	8 ×
CSampleDig 🔽 (All class members) 🔽 💊 OnButton1 🔽 🕱 🗸 🕼 🖄 🖽 🚣 🚦 🖏 🖑	
<pre>Sample resource: Sample resource: Sample resource: SimpTable P load P load</pre>	
	•
Build / Debug > Find in Files 1 > Find in Files 2 > Results /	Ī
Ready Ln182, Col1 REC [COL OVR	READ

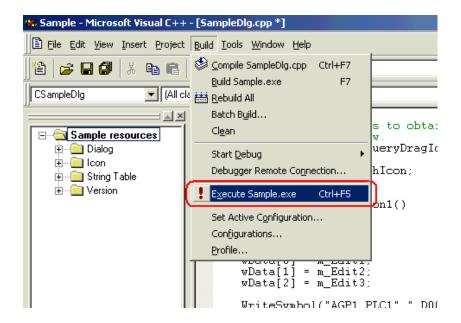
 $43 \ \, \text{Set the data entered in [Edit Box] (for three points) into the array.}$ 



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.



 $46 \ {\rm Click} \ {\rm the} \ [{\rm Yes}] \ {\rm button}.$ 

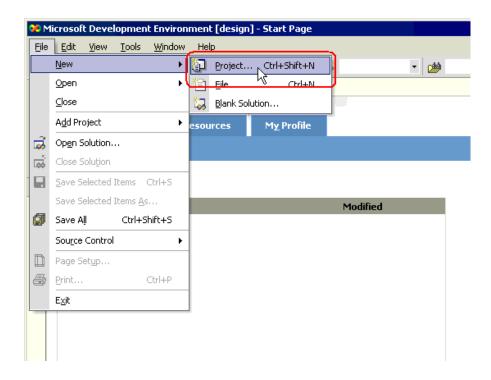
Microsoft	Visual C++
?	C:\PROGRAM FILES\PRO-FACE\PRO-SERVER EX\PRO-SDK\VC\Sample\Debug\Sample.exe
	This file does not exist. Do you want to build it?
	<u>Yes</u> <u>N</u> o

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		X
		ОК
		Cancel
1010		
2020	Button1	
3030		

# 26.10.3 VB .NET Support Function

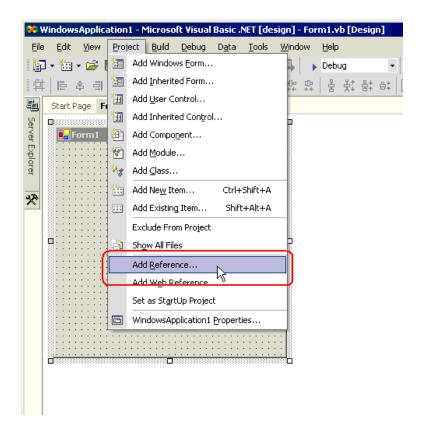
1 Start Microsoft Visual Studio .NET 2003 (or later version), and select [New] - [Project] from the [File] menu.



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

New Project					×
Project Types:		Templates:		000	8-8- 8-8- 8-8-
Visual Basic Proje	·	VB	1 MB	H VB	-
Setup and Dep     Other Projects     Visual Studio St	Windows Application	Class Library	Windows Control Library		
	olucions			R	
		Smart Device Application	ASP.NET Web Application	ASP.NET Web Service	•
A project for creating a	an application with a Windows	user interface			
Name:	WindowsApplication1				
Location:	C:\Documents and Settings'	Browse			
Project will be created a	at C:\\My Documents\Visual	Studio Projects\	WindowsApplica	ation1.	
<b>▼</b> Mor <u>e</u>		ок	Cancel	Help	

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



4 Click the [Browse] button.

1.0.5000.0 7.0.3300.0 1.0.5000.0 7.0.3300.0	C:\WINNT\Microsoft.NET\Fra C:\Program Files\Microsoft.N C:\WINNT\Microsoft.NET\Fra		Select
1.0.5000.0			
	C:\WINNT\Microsoft.NET\Fra		
7 0 2200 0			
/.0.3300.0	C:\WINNT\Microsoft.NET\Fra		
7.0.3300.0	C:\Program Files\Microsoft Vi		
1.0.5000.0	C:\WINNT\Microsoft.NET\Fra		
1.0.5000.0	C:\WINNT\Microsoft.NET\Fra		
1.0.5000.0			
1.0.5000.0	C:\WINNT\Microsoft.NET\Fra		
7.0.5000.0			
0.0.0.0		-1	
7.0.5000.0	C:\Program Eiles\Microsoft Vi	-	
			1
уре	Source		Remove
	.0.5000.0 .0.5000.0 .0.5000.0 .0.5000.0 ?.0.5000.0 ?.0.5000.0 ?.0.5000.0		0.5000.0         C:\WINNT\Microsoft.NET\Fra          0.5000.0         C:\WINNT\Microsoft.NET\Fra          0.5000.0         C:\WINNT\Microsoft.NET\Fra           .0.5000.0         C:\WINNT\Microsoft.NET\Fra           .0.5000.0         C:\WINNT\Microsoft.NET\Fra           .0.5000.0         C:\WINNT\Microsoft.NET\Fra           .0.5000.0         C:\WINNT\Microsoft.NET\Fra           .0.5000.0         C:\WINNT\Microsoft.NET\Fra           .0.5000.0         C:\Program Files\Microsoft Vi           .0.5000.0         C:\Program Files\Microsoft Vi

5 Specify the directory for ProEazyDotNet.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".)

Select Compone	ent service ser	×
Look in:	💼 bin 💽 😓 🗲 🔁 🍭 🗙 🖄 📰 - Tools -	
History	ProEasyDotNet.dll	
My Projects		
Desktop		
Favorites		
My Network Places	File name:     Image: Ima	<u>گ</u>

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	
MCnnCodeDomProvider	7.0.5000.0	C:\Program Eiles\Microsoft Vi	
ected Components:			
iomponent Name	Туре	Source	Remove
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.

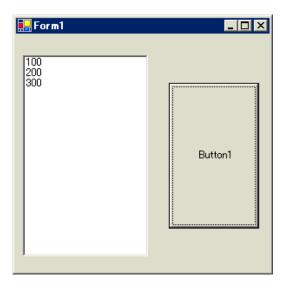
The following procedure varies depending on whether the application is intended for reading or writing, and so is explained individually.

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].

<u></u>	Windou	usānnli	cation1 -	Microsol	it Visua	Basic	NET [de	sian] -	For	m1 vh	[Desi	an]*			
File			Project	Build	Debug	Data	Tools	Wind		Help	201231	9···1			
	_	- <u>n</u> on		<u>к</u>		ງ ⊬ (:4	-	- 🖪		Debug					
ő	. <u> </u>	] • 🗁							1			•	1	_	
1	뤸	후 릐	10- <u>10</u> 0			#10 E#3	아아 뜻	)⊂ D(0 + ++	₽ ≁+	움	\$\$ 음:	* 合*	0[-	<u>\$</u>	•
5	Toolbo	x		<b>4</b> ×	Start	Page I	Form1.v	/b [De	sign]	*					
ß	Data														
Server Explorer	Compo					Form1						_			
Ψ Ψ	Windo	ws Forms	5	<b></b>			aba <b>m</b> bababa								
ore	R P	ointer				ListBox1	<u></u> 99999		U				::::		
~	A۷	abel													
	🔥 Li	nkLabel							•••				::::		
	ab B	utton							•••						
	abl T	extBox													
	🖹 М	lainMenu							 				::::		
		heckBox													
	⊙ R	adioButt	on						•••				::::		
	[ ^{xv} ] G	roupBox							•••						
	R P	ictureBox	<						•••				::::		
	P	anel							• •						
		ataGrid				*1*1*1*1*1*1*1*1*1*1			<u>.</u>						
	<u> </u>	stBox					<b>.</b>		<u>.</u>				::::		
		heckedLi	stBox											-	
		omboBo>													
		stView													
		reeView													
		abContro	-												
		abcontro	1												

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

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

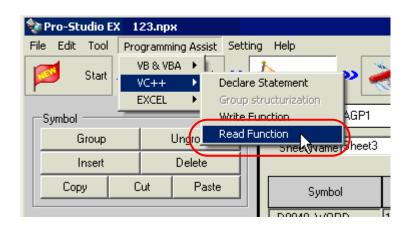
20	Window	sApplio	ation1 -	Microsol	ît Visua	Basic .	NET [desi	ign] - Foi	m1.vb [I	Design]*			
Eil		⊻iew	Project	Build	<u>D</u> ebug	D <u>a</u> ta		<u>T</u> ools	<u>W</u> indow	Help			
ł	- 🛅	- 🚅	80	<u>ж</u> 🗈		<del>ງ -</del> ເພ	- Ja -	<b>B</b> ,   ,	Debug	-	2		
Į 📫		후 릐	-00- t00			10 EB	000 H0c	D[o []o →+ →+	응 붉	: 음‡ 음‡	아 후	<u>г</u> , с	а.
5	Toolbox			<b>4</b> ×	Start	Page	orm1.vb	[Design	]*				
ŝ	Data												
rver	Compon					Form1				_			
¥	Window	s Forms	:	<b></b>									
Server Explorer	🕨 Poi	nter			1	ListBox1			:::::		::::		
	A Lat	bel											
	A Lini	kLabel			1.1			::)					
	ab Bu	itton											
	abl Te:	xtBox			1.1								
	🖹 Ma	inMenu							л в	utton1			
	🔽 Ch	eckBox								actorn			
	💽 Ra	dioButte	on										
	[ ^{XV} ] Gro	oupBox			1 ::								
	🔜 Pid	tureBo×	c					::1		, 🗖 i i i i i i i i			
	🗌 Par	nel						: : :					
	🕤 Da	taGrid											
	≣∰ List	:Box											
	🔡 Ch	eckedLi	stBox										
	📑 Co	mboBo×											
	eee List	View											
	🧱 Tre	eView											
	🛅 Tal	Contro	bl										

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

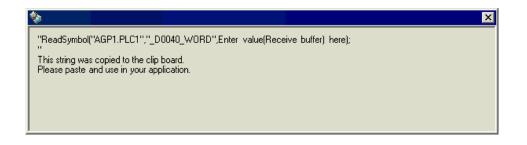
💱 Pro-Studio EX 2.npx						
File Edit Tool Programming Assist Settin	g Help					
Start >> 🔪 Node >>	녿 Symbol ⋗ 葔	Feature ≫ 📑	Sa Sa	ave 😕 🆄 T	ransfer	Monitor Status
Symbol	Node Name AGP1		Device	Name PLC1		
Group Ungroup	Sheet Name Sheet3		Set it as	a global symbol shee	ət.	
Insert Delete						
Copy Cut 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	
					1	
Global Constant Setting Screen				<u> </u>	1	
Pro-Server EX					1	
E ₽ PC1 (192.168.0.1)					1	
#INTERNAL:Sheet1					1	
□					1	
AGP1 (192.168.0.100)					1	
#INTERNAL:Sheet2					1	
→ WinGP					1	
• LT3000					1	
GP Series					1	
Global Symbol						
			<u> </u>		1	
	l				1	
			<u> </u>		1	
•		1	1	1	P	

* 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*			4 ⊳
🖍 (General)		•	(Declarations)	[
Public Clas	s Form1			
Inherit	s System.Windows.Forms.Form			
Uindows F	orm Designer generated code			
Private	Sub Button1 Click(ByVal sender As Sy	stem.Object.	, ByVal e As System.EventArgs) Handles Button1.Click	
	dSymbol ("AGP1.PLC1", "_DOO40_WORD", Ent			
- End Sul				
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.vb [Design]* Form1.vb*		4 ⊳
🖍 (General)	▼ I\ (Declarations)	
Imports Publi () ATL	1	
I A Microsoft	ws.Forms.Form	
Uin O System	erated code	
{} WindowsApplication2		
	Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click	
ReadSymbol ("AGP1.	.PLC1","_D0040_WORD",Enter value(Receive buffer) here);	
- End Sub		
L End 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.v	Design]* Form1.vb*	1 Þ
🖍 (Gen	al) 🔽 🚺 (Declarations)	
Im	prts ProEasyDotNet	_
🖃 Pu	Lic Class Form1	
	Inherits System.Windows.Forms.Form	
t U	ndows Form Designer generated code	
þ	Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click	
	Dim wData(3) As Short	
	ReadSymbol("AGP1.PLC1","_D0040_WORD",Enter value(Receive buffer) here);	
-	End Sub	
- En	Class	

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

Form1.vb [Design]* Form1.vb*	4 Þ
Button1 S Click	
Imports ProEasyDotNet	
Public Class Form1	
Inherits System.Windows.Forms.Form	
<ul> <li>Windows Form Designer generated code</li> <li>Private Sub Button1 Click(ByVal sender As System.Object, ByVal e As System</li> </ul>	- Frentwrge) Hendles Buttoni Click
Dim wData(3) As Short	. Eventarys) handles buttoni.click
ProEasy.ReadSymbol("AGP1.PLC1","_D0040_WORD",Enter value(Receive buffe	r) here):
End Sub End Class QueuingStat QueuingStat QueuingStat ReadDevice160 ReadDevice160 ReadDevice16M ReadDevice232 ReadDevice32	

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

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

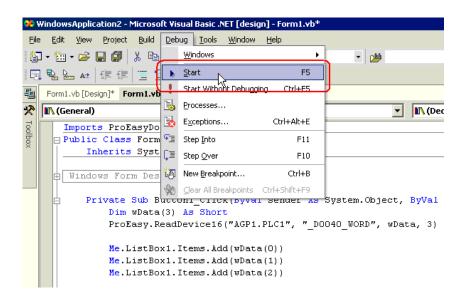
16 Specify a data storing area "wData" as the third argument. Enter ", " (comma) at the end of the third argument, and then enter "3" to specify the length of the target symbol as the fourth argument. After that, delete ";" (semicolon) at the end of the line.

Fo	orm1.vb[Design]* Form1.vb*	4 ۵
	(General) 🔽 🕅 (Declarations)	
	Imports ProEasyDotNet	
	E Public Class Form1	
	Inherits System.Windows.Forms.Form	
	Windows Form Designer generated code Private Sub Button1_Click(ByVal sender & System.Object, ByVal e & System.Event&rgs) Handles Button1.Click Dim wData(3) & Short ProEasy.ReadDevice16("AGP1.PLC1", "_D0040_WORD", wData, 3)	
	- End Sub	
	L End Class	

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

Form1.vb [Design]* Form1.vb*	٩
🖍 (General)	▼ I∿ (Declarations)
Imports ProEasyDotNet	
Public Class Form1	
Inherits System.Windows.Forms.Form	
<ul> <li>Windows Form Designer generated code</li> <li>Private Sub Button1_Click(ByVal sender &amp; System.Obj</li> <li>Dim wData(3) &amp; Short</li> </ul>	ect, ByVal e Ås System.EventÅrgs) Handles Button1.Click
ProEasy.ReadDevice16("AGP1.PLC1", "_D0040_WORD",	wData, 3)
Me.ListBox1.Items.Add(wData(0)) Me.ListBox1.Items.Add(wData(1)) Me.ListBox1.Items.Add(wData(2))	
- End Sub End Class	

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].

🔜 Form1	_ 🗆 ×
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.vb [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  ի 후 릐   ᅲ 애 쓰	昂 蒋 印 昭   咖 跸 跸 牌   울 幹 왕 왕 岡 图   🍓 🖷 🗸	
Toolbox 🕈 X	Start Page Form1.vb [Design]*	
Data		
Components	Form1	
Windows Forms		
Reinter		
${f A}$ Label	TextBox1	
A LinkLabel		
ab Button	TextBox2	
abl TextBox		
🛓 MainMenu	TextBox3	
CheckBox	· · · · · · · · · · · · · · · · · · ·	
RadioButton		
GroupBox		
RictureBox		
Panel		
🛐 DataGrid		
특별 ListBox		

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

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

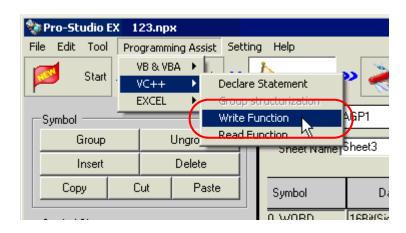
😻 WindowsApplication3 - Micr	osoft Visual Basic .NET [design] - Form1.vb [Design]*	
<u>File E</u> dit <u>V</u> iew <u>P</u> roject <u>B</u> u	id <u>D</u> ebug D <u>a</u> ta <u>T</u> ools <u>W</u> indow <u>H</u> elp	
🖥 • 🖮 • 🛩 🖬 🕼	🗈 💼 🗠 - 🗠 - 🚚 - 🖳 🕨 Debug 🔹 🍻	-
[][] [] [] [] [] [] [] [] [] [] [] [] []	品 寺 印 昭  ┉ 챠 챠 챠 片 옷 찾 왕 탁  凾 宮  💼 률 🕈	
Toolbox 🛛 🕂 🗙	Start Page Form1.vb [Design]*	
Data		
Components	Form1	
Windows Forms		
Pointer		
${f A}$ Label	TextBox1	
A LinkLabel		
ab Button	TextBox2	
abl 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 Setti	ng Help					
Start 💙 🐚 Node ン	🍐 Symbol ≫ 🄾	Feature ≫ 🗜	Sav	/e 渊 🆄 Tr	ansfer	Monitor Status
Symbol	Node Name AGP1		Device N	lame PLC1		
Group Ungroup	Sheet Name Sheet3		Set it as a	global symbol shee	t.	
Insert Delete						
Copy Cut Paste	Symbol	Data Type	Consec utive	Device Address	No. of Data	Comment
Symbol Sheet	_D0040_WORD	16Bit(Signed)		00040	3	<b>▲</b>
Add Delete					1	
· · · · · · · · · · · · · · · · · · ·			<u> </u>		1	
Check Duplication/List Used Addresses					1	
Clabel Constant Colling Conserve			-		1	
Global Constant Setting Screen			-		1	
Pro-Server EX			<u> </u>  -		1	
E			ii		1	
GP3000 Series			i — i		1	
🖻 - 🔙 AGP1 (192.168.0.100)					1	
#INTERNAL:Sheet2					1	
PLC1:Sheet3 A Series CPU I	I				1	
• LT3000			┝──┝		1	
+ GP Series	II		-		1	II
📖 🕨 Global Symbol			<u> </u>  -		1	
			-		1	
			<u> </u>  -		1	
		ľ	i—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.

1	🌜 💫 🕹
	"WriteSymbol("AGP1.PLC1","_D0040_WORD", Enter value(Transmitting buffer) here); "
	This string was copied to the clip board. Please paste and use in your application.
I	
I	

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*	
🖪 (General)	▼ V (Declarations)
- Public Class Form1	
Inherits System.Windows.Form	
B Windows Form Designer generated	]
	nder Ås System.Object, ByVal e Ås System.Eventårgs) Handles Button1.Click 0_WORD", <u>Enter value</u> (Transmitting buffer) here);
End Sub	
- Ind Grass	

### 25 Import the ProEasyDotNet library.

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

Start Page   F	Form1.vb [Design]* Form1.vt	b*	4 ۵
🚺 (General)	1	💌 🚺 (Declarations)	
⊕ Vin	ATL     Microsoft     OročasyDotNet     System     WindowsApplication3     ivate Sub Button1_C	vs.Forms.Form herated code Click(ByVal sender & System.Object, ByVal e & System.Event&rg 1.PLC1","_D0040_WORD", <u>Enter yalue</u> (Transmitting buffer) here);	s) Handles Button1.Click
End C	nd Sub lass		

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

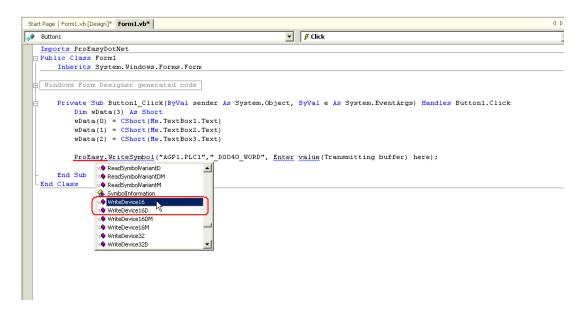
The array type ("Short" in this example) must conform to the data type of the target symbol. Specify the same data length as the target symbol ("3" in this example).

<b>I</b> \ (G	eneral)	•	(Declarations)	
	Imports ProEasyDotNet			
🗆 F	Public Class Form1			
	Inherits System.Windows.Forms.Form			
⊡[	Windows Form Designer generated code			
	Private Sub Button1 Click(ByVal sender As Syst	em.Object,	ByVal e As System.EventArgs) Handles Button1.Click	
	Dim wData(3) As Short			
	WriteSymbol("AGP1.PLC1"," D0040 WORD", Ent	er value(Tr	ansmitting buffer) here);	
	End Sub			
LF	Ind Class			

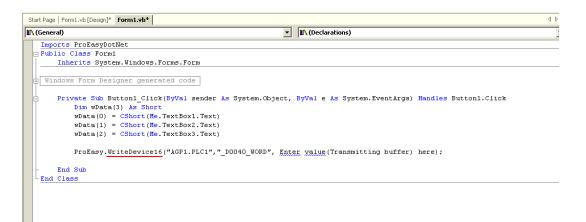
27 Set the data to be entered in [TextBox1] to [TextBox3] in the array.

Start	Start Page   Form1.vb [Design]* Form1.vb*	4 Þ
<b>I</b> (G	IV (General)	
	Imports ProEasyDotNet	
E 1	Public Class Form1	
	Inherits System. Windows. Forms. Form	
æ[	0 Windows Form Designer generated code	
þ	Private Sub Buttoni Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Hand	les Button1.Click
	Dim wData(3) As Short	
	WData(0) = CShort(Me.TextBox1.Text)	
	<pre>wData(1) = CShort(Me.TextBox2.Text) </pre>	
	<pre>wData(2) = CShort(Me.TextBox3.Text)</pre>	
	WriteSymbol("AGP1.PLC1"," D0040 WORD", Enter value(Transmitting buffer) here);	
	- End Sub	
LL	Lend Class	

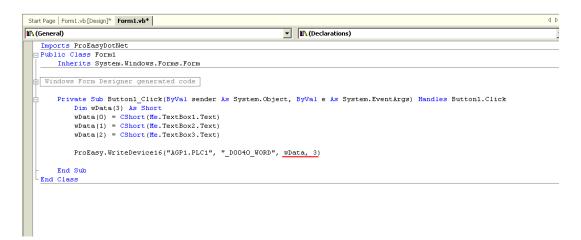
28 Enter "ProEasy." before "WriteSymbol", and select [WriteDevice16] from the displayed list box.



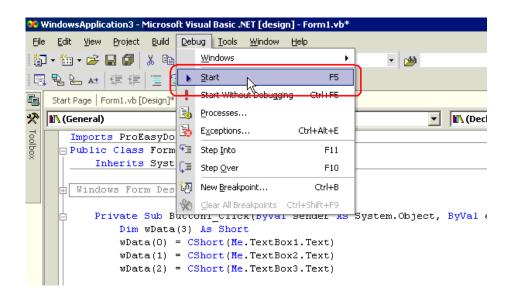
**29** Delete "WriteSymbol" from the character string (write function) that has been pasted from the clipboard.



**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].

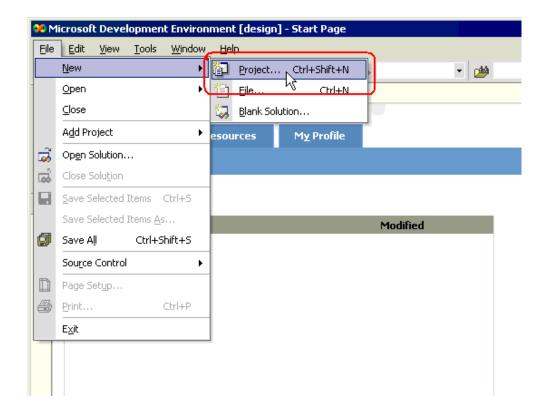
🖳 Form1	
TextBox1	
TextBox2	Button1
TextBox3	

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

🖶 Form 1	
200	Button1
300	

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

New Project					×			
Project Types:		Templates:			5-5- 5-5- 5-5-			
Visual Basic Proje			<b>11</b> #	<b>*</b>	-			
Setup and Dep     Other Projects     Visual Studio S		Windows Application	Class Library	Windows Control Library				
	olations	<b>G</b> #		ø				
		Smart Device Application	ASP.NET Web Application	ASP.NET Web Service	•			
A project for creating	an application with a Windows	user interface						
Name:	WindowsApplication1							
Location:	C:\Documents and Settings	\Administrator\M	y Documen 💌	Browse				
Project will be created at C:\\My Documents\Visual Studio Projects\WindowsApplication1.								
<b>▼</b> Mor <u>e</u>		ок	Cancel	Help				

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

96 V	Vindow:	sApplica	ation	4 - N	licros	oft Visual	C# .NE	T [desig	in] - F	orm	1.cs [	Desi	ign]		
Eile	<u>E</u> dit	⊻iew	Proj	iect	Build	<u>D</u> ebug	D <u>a</u> ta	<u>T</u> ools	Wind	wob	Help	I			
皆	- 🏪	• 🚔	1	Ado	d Windo	ws <u>F</u> orm			₽.		Deb	ug		-	
I II		후 릐	1	Ado	l <u>I</u> nherit	ed Form				₽ ++	움	북‡	움‡	合*	
5	Start P	age F	1	Ado	<u>l U</u> ser C	ontrol									
			1	Ado	l Inherit	ed Contro:	l								
7	F	orm1	1	Ado	d Compo	0 <u>n</u> ent									
🛠 Toolbox	••••		***	Ado	d <u>⊂</u> lass.										
	••••		****	Ado	l Ne <u>w</u> It	:em	Ctrl+	Shift+A							
	••••			Ado	d Existin	g Item	Shift	+Alt+A							
				Exc	lude Fro	om Project									
			þ	Shg	w All Fi	es									
				Ado	l <u>R</u> efere	ence	2								
				Add	i w <u>e</u> b R	eference.			_						
				Set	as St <u>a</u> r	tUp Projec	t								
			<b></b>	Win	idowsAp	oplication4	Propert	ies							

4 Click the [Browse] button.

1.0.5000.0 7.0.3300.0 1.0.5000.0 7.0.3300.0	C:\WINNT\Microsoft.NET\Fra C:\Program Files\Microsoft.N C:\WINNT\Microsoft.NET\Fra	
1.0.5000.0		-
	C:\WINNT\Microsoft.NET\Fra	
7 0 2200 0		
7.0.3300.0	C:\WINNT\Microsoft.NET\Fra	
7.0.3300.0	C:\Program Files\Microsoft Vi	
1.0.5000.0		
1.0.5000.0	C:\WINNT\Microsoft.NET\Fra	
1.0.5000.0	C:\WINNT\Microsoft.NET\Fra	
1.0.5000.0	C:\WINNT\Microsoft.NET\Fra	
7.0.5000.0	C:\WINNT\Microsoft.NET\Fra	
0.0.0.0	C:\Program Files\Microsoft Vi	=1
7.0.5000.0	C:)Program Eiles:)Microsoft Vi	-
Туре	Source	Remov
	1.0.5000.0 1.0.5000.0 1.0.5000.0 1.0.5000.0 7.0.5000.0 0.0.0.0 7.0.5000.0	1.0.5000.0         C:\WINNT\Microsoft.NET\Fra           1.0.5000.0         C:\WINNT\Microsoft.NET\Fra           1.0.5000.0         C:\WINNT\Microsoft.NET\Fra           1.0.5000.0         C:\WINNT\Microsoft.NET\Fra           7.0.5000.0         C:\WINNT\Microsoft.NET\Fra           0.0.00         C:\WINNT\Microsoft.NET\Fra           0.0.00         C:\WINNT\Microsoft.NET\Fra           0.0.00         C:\Program Files\Microsoft Vi           7.0.5000.0         C:\Program Files\Microsoft Vi

5 Specify the directory for ProEazyDotNet.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".)

Select Compone	ent enter en	×
Look in:	💼 bin 💽 🖕 🕈 🔁 🥘 🗙 📑 🗉 - Tools -	
History	ProEasyDotNet.dll	
My Projects		
Desktop		
* Favorites		
My Network Places	File name:     Image: Open interview       Files of type:     Component Files (*.dll;*.tlb;*.olb;*.ocx;*.exe)         Cancel	Ĵ

6 Click the [OK] button.

Component Name	Version	Path	
Accessibility.dll	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra	Sele
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	-
ected Components:			
omponent Name	Туре	Source	Remo
roEasyDotNet.dll	File	C:\Program Files\Pro-face\Pro	

"ProEasyDotNet.dll" will be registered.

This completes the C# .NET operating environment setup.

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

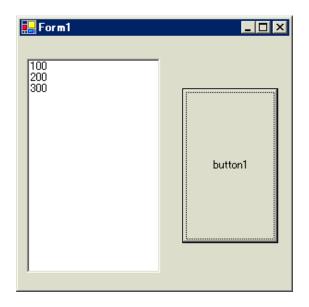
The following procedure varies depending on whether the application is intended for reading or writing, and so is explained individually.

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].

😵 WindowsApplication4 - Micro	osoft Visual C# .NET [design] - Form1.cs [Design]*
Eile Edit Yiew Project Bui	
🗿 • 🛅 • 🚔 🖬 🗿 🐰 I	🖹 🛍 හ - ශ - ළ - 🖳 🌗 Debug 🔹 🍎
[井] [] 후 릐   雨 굓 프	등 禁 했 路 @
Toolbox 🛛 🕈 🗙	Start Page Form1.cs [Design]*
Data	
Components	Form1
Windows Forms	
Pointer	L L L L L L L L L L L L L L L L L L L
${f A}$ Label	
A LinkLabel	
ab Button	
abl TextBox	
喜 MainMenu	
CheckBox	
RadioButton	
GroupBox	
🔏 PictureBox	
Panel	
🛐 DataGrid	
E ListBox	
📰 CheckedListBox	
E ComboBox	
222 ListView	
treeView	
TabControl	

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

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

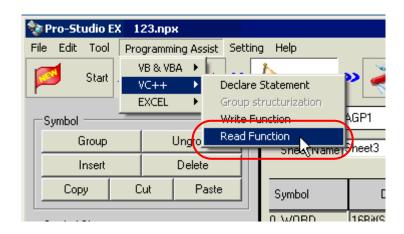
😢 Wind	lowsAppl	ication4 -	Microso	oft Visual	C# .NE	T [desig	n] - Form1	l.cs [Desig	jn]*		
Eile E	dit <u>V</u> iew	Project	<u>B</u> uild	Debug	D <u>a</u> ta	Tools	<u>W</u> indow	Help			
- 📬	🋅 🕶 🚅	8	χ 🗈		) + Ci	- J	· 🖳 🕨	Debug	-	2	
I ∰   I		]0- <u>†0</u> 0	) <u>oDi</u> (			000 H	< 0]a (]a → → ← → ↔	물 붓	응; 응;	아 후	<b>B B</b> .
Toolbox		9	×	Start Page	Form	n1.cs [D	esign]*				
Data											
Compon				E Forn	n1					<u> </u>	
Windows											
🕨 Poi				listBo	x1					-	
A Lab	el										
A Lini	Label			::			:: <b>p</b>	<u>, , , , , , , , , , , , , , , , , , , </u>		:	
ab But	ton										
abl Tex	tBox			::						:	
📑 Mai	nMenu						· · 4	button1		-	
🔽 Che	eckBox			::				battorn	T:	1	
💽 Ra	lioButton			::						:	
Gro	upBox			::							
🔜 Pict	ureBox			::			:: <b>D</b>	i i i i i i i i i i i i i i i i i i i	<u>iiii</u>	:	
Par	el										
😼 Dal	aGrid			:: L					:::::	:	
=∉ List	Box									-	
🔡 Ch	eckedListBo	x									
Cor	nboBox										
BB2 List	View										
Tre											
Tat	Control										

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

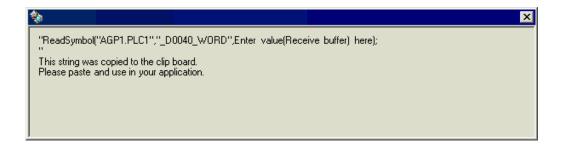
🂱 Pro-Studio EX 2.npx						
File Edit Tool Programming Assist Setti	ng Help					
Start 🔉 🔪 Node ン	🌔 Symbol ≫ 🥃	Feature ⋗ 📑	Save	<b>&gt;&gt;</b> 🆄 Tr	ansfer	Monitor Status
Symbol	Node Name AGP1		Device Nar	me PLC1		
Group Ungroup Insert Delete	Sheet Name Sheet3		Set it as a gl	lobal symbol shee	t	
Copy Cut Paste	Symbol	Data Type	Consec D utive	)evice Address	No. of Data	Comment
Symbol Sheet	_D0040_WORD	16Bit(Signed)	DO	0040	3	<b>_</b> _
Add Delete					1	
· · · · · · · · · · · · · · · · · · ·					1	
Check Duplication/List Used Addresses			<u> </u> _		1	II
			-		1	II
Global Constant Setting Screen			-		1	
Pro-Server EX			<u> </u>  -		1	
PC1 (192.168.0.1)					1	
GP3000 Series					1	
🖻 - 🔄 AGP1 (192.168.0.100)					1	
#INTERNAL:Sheet2					1	
PLC1:Sheet3 A Series CPU I					1	
• LT3000			$\vdash$		1	
GP Series			<u> </u>		1	II
Global Symbol			<u> </u>		1	II
			-		1	
			<u> </u>		1	
					1	<b>•</b>
<b>▲</b>					-	

* 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.

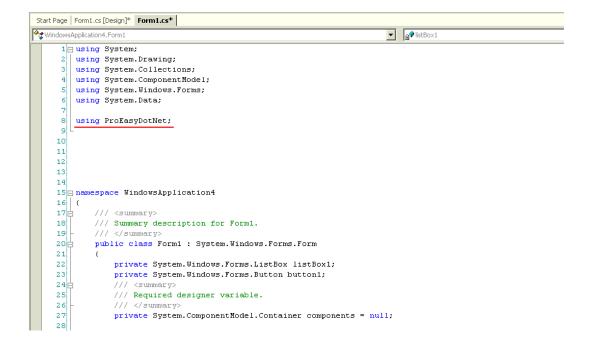


11 Double-click [button1] in [Form1], and paste the clipboard data (read function) below the [button1_Click] method ("private void button1_Click..." character string).

```
Start Page | Form1.cs [Design]* Form1.cs*
✤ WindowsApplication4.Form1
 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
 /// </summary>
 19
 20
 private System.ComponentModel.Container components = null;
 21
 public Form1()...
 22 🕁
 34
 /// <summary>
 /// Clean up any resources being used.
 3.5
 36
 /// </summary>
 37
 protected override void Dispose(bool disposing)...
 49 🖨
 Windows Form Designer generated code
 88
 89
 /// <summary>
 90
 /// The main entry point for the application.
 91
 /// </summary>
 92
 [STAThread]
 93 E
 static void Main()
 94
 {
 95
 Application.Run(new Form1());
 96
 }
 97
 98
 private void button1_Click(object sender, System.EventArgs e)
 99
 ł
 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.

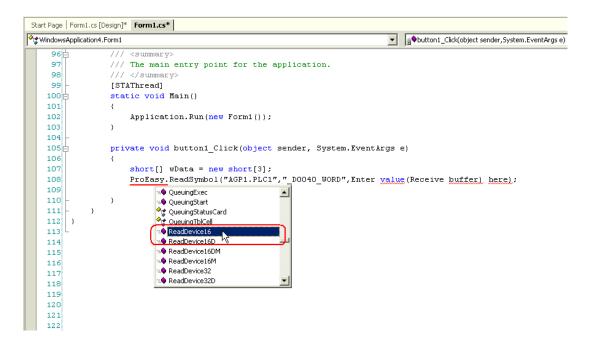


13 For the read data storing area, declare a variable "wData".

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 [D	esign]* Form1.cs*	
WindowsApplication4.	Form1	button1_Click(object sender,System.EventArgs e)
96	/// <summary></summary>	
97	/// The main entry point for the application.	
98	///	
99 -	[STAThread]	
100向	static void Main()	
101	(	
102	Application.Run(new Form1());	
103	}	
104 -		
105白	<pre>private void button1_Click(object sender, System.EventArgs</pre>	e)
106	(	
107	<pre>short[] wData = new short[3];</pre>	
108	ReadSymbol("AGP1.PLC1","_D0040_WORD",Enter value(Received)	ve <u>buffer) here)</u> ;
109		
110 -	}	
111 - }		
112 }		
113		
114		
115		
116		
117		
118		

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



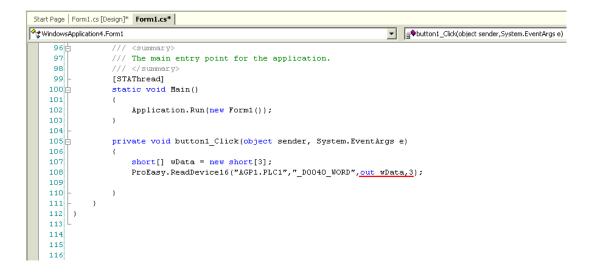
15 Delete "ReadSymbol" from the character string (read function) that has been pasted from the clipboard.

```
Start Page | Form1.cs [Design]* Form1.cs*
VindowsApplication4.Form1

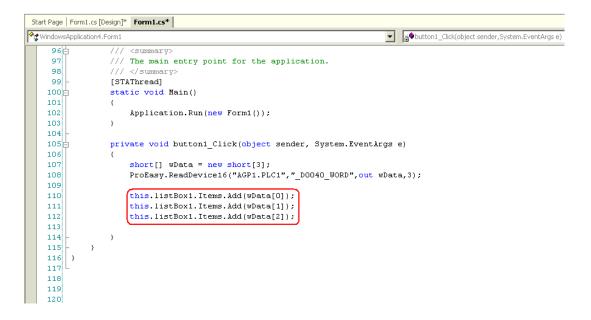
 System.EventArgs e)

 96H
 <summary>
 /// 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
 ł
 107
 short[] wData = new short[3];
 108
 ProEasy.ReadDevice16("AGP1.PLC1"," D0040 WORD",Enter value(Receive buffer) here);
 109
 110
 }
 111
 }
 112
 }
 113
 114
 115
 116
```

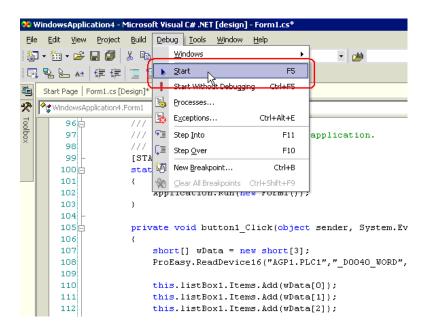
16 Specify a data storing area "wData" with the reference modifier (out), as the third argument. Enter "," (comma) at the end of the third argument, and then enter "3" to specify the length of the target symbol as the fourth argument.



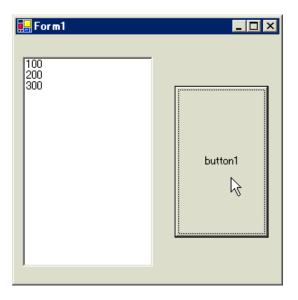
17 Add the read data on three items (wData[0], wData[1], wData[2]) into [listBox1] in this order.



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



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



## 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	L

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

😢 WindowsApplication5 - 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	d <u>D</u> ebug D <u>a</u> ta <u>T</u> ools <u>W</u> indow <u>H</u> elp	
🔯 • 🛅 • 🚔 🖬 🕼 🛛	🛍 💼 म्म् न् 🖉 न 📕 🕞 Debug 💿 🚽 🍅	•
弉   臣 혹 릐   ㅠ ~ 典	昆 拱 玑 函   四 改 架 架 🖇 於 計 타   匝 图   ங 🖷 🗸	
Toolbox <b>4</b> ×	Start Page Form1.cs [Design]*	
Data		
Components	Form1	
Windows Forms		
Revinter Revinter		
A Label	textBox1	
$\underline{\mathbf{A}}$ LinkLabel		
ab Button	textBox2	
abl TextBox		
🛓 MainMenu	textBox3	
CheckBox		
RadioButton		
GroupBox		
RectureBox		
Panel		
DataGrid		
E ListBox		

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

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

WindowsApplication5 - Micr	osoft Visual C# .NET [design] - Form1.cs [Design]*
<u>File Edit V</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
🎦 • 🛅 • 🚔 🔲 🞒 🐰	🖺 💼 🗠 - 🗠 - 🚚 - 🖳 Debug 🛛 - 🍎
神 [] 후 릐   ㅠ ᅲ @	🖙 拱 卯 段   咖 玲 映 史   울 於 왕 타   凾 凰 👒 🖷 🖡
oolbox 🛛 🕈 🗙	Start Page Form1.cs [Design]*
Data	
Components	Form1
Vindows Forms	
Rointer	
A Label	textBox1
A LinkLabel	
ab Button	textBox2
abi TextBox	
🛓 MainMenu	···· textBox3
CheckBox	····
RadioButton	
^{xv} GroupBox	
RictureBox	
Panel	
🚰 DataGrid	
音賞 ListBox	

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

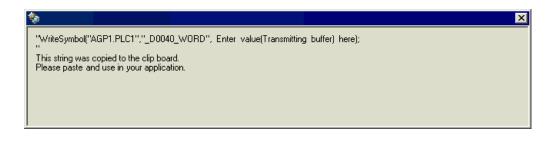
💱 Pro-Studio EX 2.npx					_ 🗆 🗵
File Edit Tool Programming Assist Settin	ng Help				
Start 🔉 🟹 Node 🔉	🍐 Symbol 🌺	Feature ≫ 📔	📔 Save ≫ 🆄	Transfer	Monitor Status
Symbol	Node Name AGP1		Device Name PLC1		
Group Ungroup Insert Delete	Sheet Name Sheet3		Set it as a global symbol she	et.	
Copy Cut Paste	Symbol	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				1	
				1	
Global Constant Setting Screen				1	
Pro-Server EX				1	
E		<u></u>		1	
#INTERNAL:Sheet1		<u> </u>	<u> </u>	1	
E → GP3000 Series		<u></u>		1	
E - GP1 (192.168.0.100)		<u> </u>		1	I
#INTERNAL:Sheet2		<u> </u>		1	
→ WinGP		<u> </u>		1	
> LT3000		<u> </u>		1	
► GP Series		<u> </u>		1	
Global Symbol		<u> </u>		1	
	I	<u> </u>		1	
	I	<u> </u>		1	
	l	<u> </u>		1	
4 F		1	1 1	P.	

* 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.

🎨 F	🂱 Pro-Studio EX 🛛 123.npx									
File	Edit	Tool	Pro	Programming Assist Setting Help						
			١	/B & VE	5A 🕨 📗			K.	Ξ,	
	<b>/</b>	Start	<u>ا ا</u>	/C++	•	Decl	are S	Statement	ľ	1 🦿
			E	EXCEL	1	Grou	<del>ip sti</del>	ucturization	$\rightarrow$	
Symbol Write			e Function GP1							
Group				Ungro	Rea	d Fur	nction 内		haat?	
	Insert			Delete			Sheet war	nejo	neeto	
	Сору		0	Cut Paste				Symbol		D
							0 WORD	-	16Bit/Si	

The write function is copied to the clipboard.

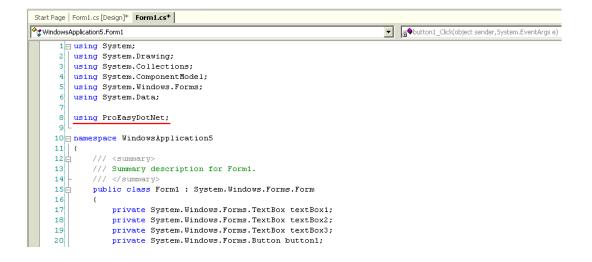


24 Double-click [button1] in [Form1], and paste the clipboard data (write function) below the [button1_Click] method ("private void button1_Click..." character string).

```
Start Page | Form1.cs [Design]* Form1.cs*
VindowsApplication5.Form:
 ▼ Š♥button1_Click(object sender,System.EventArgs e)
 12
 /// </summarv>
 13
 public class Form1 : System.Windows.Forms.Form
 14
 {
 15
 private System.Windows.Forms.TextBox textBox1;
 16
 private System.Windows.Forms.TextBox textBox2;
 17
 private System.Windows.Forms.TextBox textBox3;
 private System.Windows.Forms.Button button1;
 18
 19
 /// <summary>
 20
 /// Required designer variable.
 21
 /// </summary>
 22
 private System.ComponentModel.Container components = null;
 23
 24
36
 public Form1()...
 /// <summarv>
 /// Clean up any resources being used.
 37
 38
 /// </summary>
 39 🖨
 protected override void Dispose(bool disposing)...
 51 🗄
 Windows Form Designer generated code
 111
 112
 /// <summary>
 /// The main entry point for the application.
 113
 114
 /// </summary>
 115
 [STAThread]
 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","_D0040_WORD", Enter value(Transmitting buffer) here);
 123
 124
 125
 }
 126
 }
 127 }
 128 L
 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*
▼ 😽 button1_Click(object sender,System.EventArgs e)
 25
 26
 public Form1()...
 386
 /// <summary>
 /// Clean up any resources being used.
 39
 40
 /// </summarv>
 protected override void Dispose(bool disposing)...
 41曲
 53 🖨
 Windows Form Designer generated code
 113
 114
 /// <summary>
 /// The main entry point for the application.
 115
 /// </summary>
 116
 117
 [STAThread]
 118内
 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*
VindowsApplication5.Form1
 ▼ Š♥button1_Click(object sender,System.EventArgs e)
 26
 public Form1()...
 380
 /// <summary>
 /// Clean up any resources being used.
 39
 40
 /// </summary>
 protected override void Dispose(bool disposing)...
 41 🕁
 53 🖨
 Windows Form Designer generated code
 113
 114
 /// <summary>
 115
 /// The main entry point for the application.
 /// </summary>
 116
 [STAThread]
 117
 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
 WriteSymbol("AGP1.PLC1","_D0040_WORD", Enter value(Transmitting buffer) here);
 131
 132
 }
 133
 }
 134 }
 135
 13.6
 137
 138
```

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

```
Start Page | Form1.cs [Design]* Form1.cs*
VindowsApplication5.Form1
 26
 public Form1()...
 380
 /// <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>
 ///% \label{eq:linear} The main entry point for the application.
 115
 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
 {
 12.5
 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
 VriteDevice16
 🔶 WriteDevice16D
 136
 WriteDevice16DM
 137
 138
 🕸 WriteDevice16M
 ViteDevice32
 139
 WriteDevice32D
 140
 141
 142
 143
 144
 145
 146
```

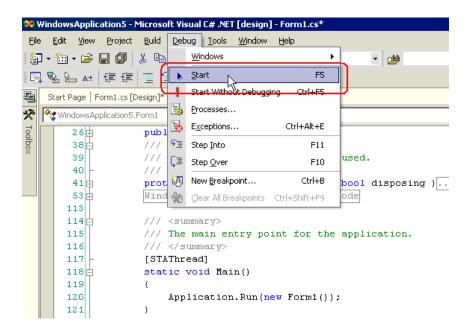
**29** Delete "WriteSymbol" from the character string (write function) that has been pasted from the clipboard.



**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
 ▼ 8♥button1_Click(object sender,System.EventArgs e)
 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
 /// <summarv>
 114 内
 115
 /// The main entry point for the application.
 116
 /// </summary
 117
 [STAThread]
 118
 static void Main()
 119
 {
 120
 Application.Run(new Form1()):
 121
 3
 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
 128
 wData[2] = short.Parse(this.textBox3.Text);
 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].

🖳 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	R