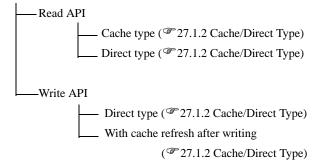
27 Designing Your Own Program

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

27.1 Using API Functions

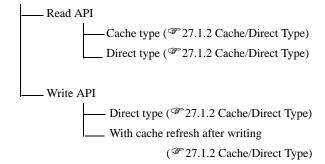
Reading and writing a Device/PLC

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



PLC communication with multiple devices

Multi-handle functions (^{CP}27.1.1 Single-/Multi-Handle Functions)



For effective communication

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

Other functions

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

27.1.1 Single-/Multi-Handle Functions

Single-Handle APIs

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

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

Multi-Handle APIs

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

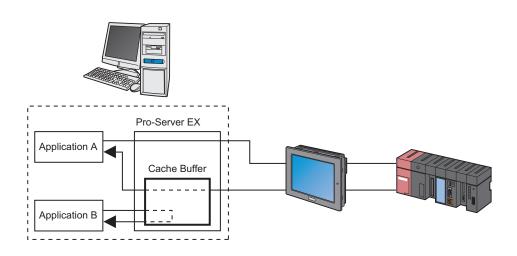
Multi-Handle APIs can be used for multi-thread applications, or for simultaneous access to multiple Devices/ PLCs.

27.1.2 Cache/Direct Type

Cache Read

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

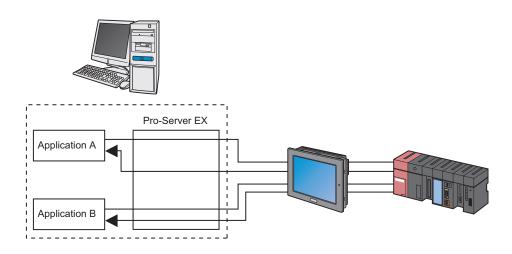
With the Cache Read feature, however, when two applications A and B send reading requests to the same Device/ PLC, 'Pro-Server EX' reads data from the Device/PLC according to the request of Application A first, stores the read data into the internal cache buffer, and sends the data to Application A in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



Direct Read

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

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



Direct Write

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

Write with Cache Refresh

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

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

27.1.3 Cache Buffer Control APIs

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

NOTE • 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:

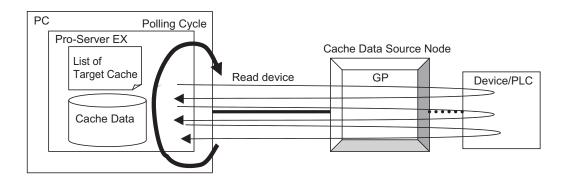
 Registration using 'Pro-Studio EX' (Create a cache buffer in "Device Cache" on the feature screen, and register it in a network project file.)
 Registration using API

Cache buffer updating procedure

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

The principle of polling method

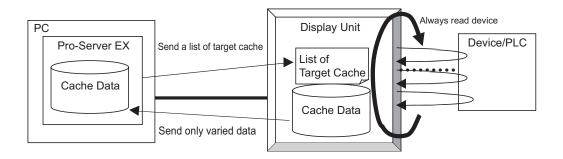
According to a list of target devices in the cache buffer, 'Pro-Server EX' reads device data to update the cache buffer when the cycle specified in cache buffer registration is reached.



◆ 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).

27.1.4 Group Access

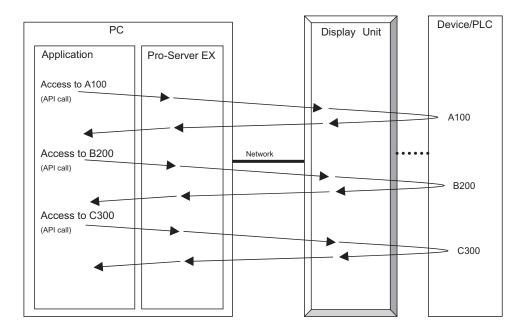
Some APIs use a group symbol to specify a device address.

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

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

♦ When calling API individually for each device:

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

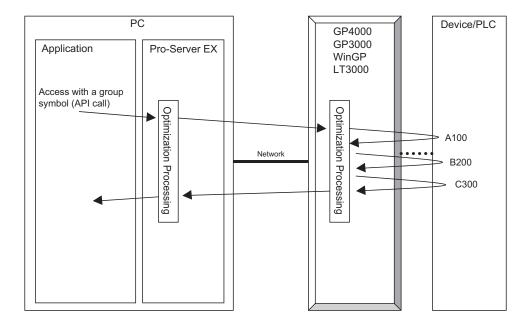


When accessing group symbols

Operation differs depending on the type of node.

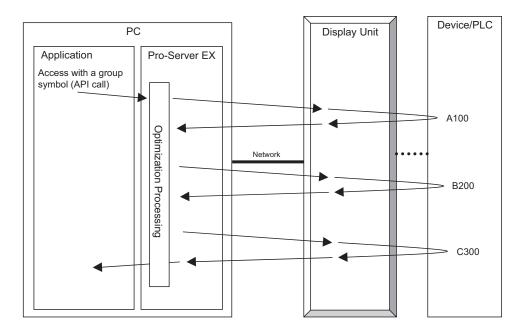
For GP4000/LT4000 Series node, GP3000 Series node, WinGP node or LT3000 node
 'Pro-Server EX' sends a request for each node only once. The node internally divides the request to access

each device separately. Thus, 'Pro-Server EX' can efficiently communicate with the devices on the network.



For GP Series node

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

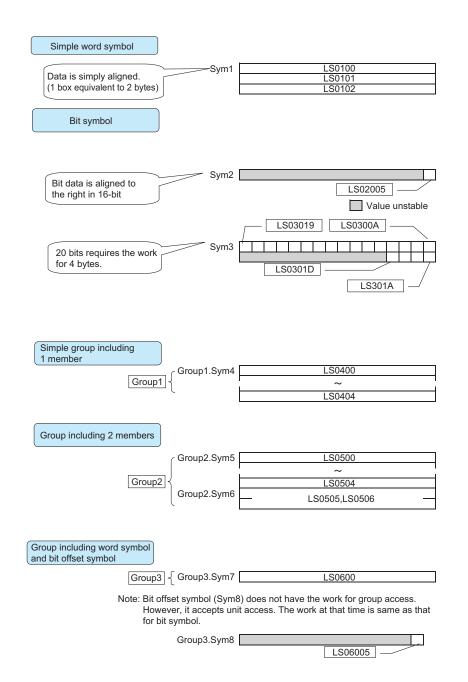


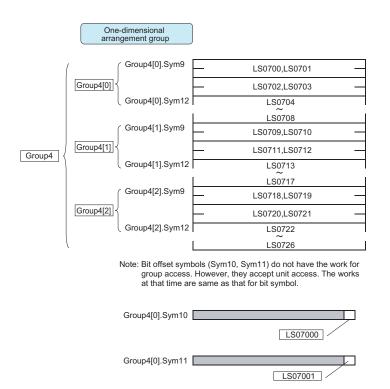
Data structure for group symbol access

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

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

Examples of the data buffer structures are shown below.





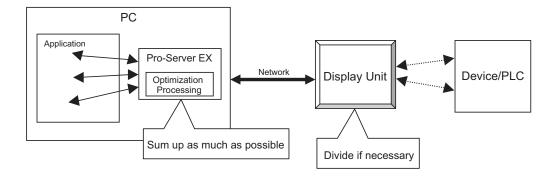
Device addresses for Group4[1].Sym10 and Group4[1].Sym11 are LS07090 and LS07091 respectively. Device addresses for Group4[2].Sym10 and Group4[2].Sym11 are LS0718 and LS07181 respectively.

Two-dimensional group (nest of groups)

	(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	LS0900 LS0901 LS0902 LS0903 LS0904 LS0905
		Group6[0].Sym15	LS0906,LS0907
		Group6[1].Group5[0] {Group6[1].Group5[0].Sym13	LS0908 LS0909
Group6	Group6[1] <	Group6[1].Group5[1] {Group6[1].Group5[1].Sym13	LS0910 LS0911
		Group6[1].Group5[2] {Group6[1].Group5[2].Sym13	LS0912 LS0913
		Group6[1].Sym15	LS0914,LS0915
		Group6[2].Group5[0] {Group6[2].Group5[0].Sym13	LS0916 LS0917
	Group6[2] <	Group6[2].Group5[1] {Group6[2].Group5[1].Sym13	LS0918 LS0919
		Group6[2].Group5[2] {Group6[2].Group5[2].Sym13	LS0920 LS0921
		Group6[2].Sym15	LS0922,LS0923

27.1.5 Queuing Access

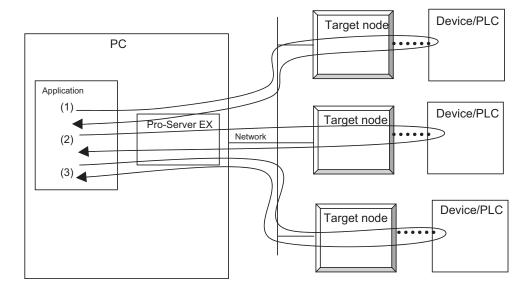
'Pro-Server EX' stores a device access request every time an API is called, and then optimizes the stored requests to access individual devices at once.



The principle of queuing access

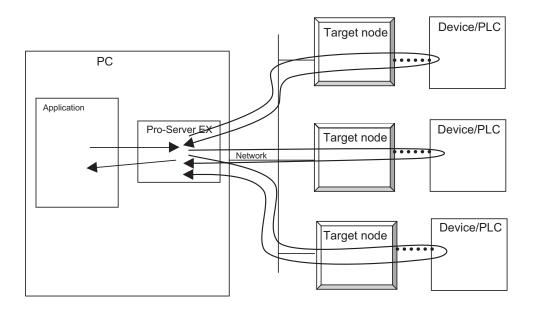
♦ Simple API access

'Pro-Server EX' executes sequential processing.



Queuing access

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



Procedures for use

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

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

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

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

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

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

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

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

27.1.6 Bit Data Access

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

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

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

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

(Example) Data buffer storing order for 20 bit devices

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

< Applicable API >

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

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

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

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

< Applicable API >

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

(3) Handling bit offset symbol for group symbol access

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

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

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

Refer to "27.1.4 Group Access" for more details.

27.1.7 System APIs

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

The system APIs are classified into the following categories:

Single-Handle APIs

You can use the 'Pro-Server EX' features without specifying a Pro-Server handle.

With this method, multiple APIs cannot be simultaneously used. (If you try to use multiple APIs simultaneously, the double-call error occurs.)

Multi-Handle APIs

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

27.1.8 SRAM Data Access APIs

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

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

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

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

27.1.9 CF Card and SD Card APIs

API for accessing data on CF and SD cards.

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

27.2 Device Access APIs

■ Single-Handle Cache Read APIs

Function	Bit data
INT WINAPI ReadDevi	iceBit(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount);
Function	8-bit data
INT WINAPI ReadDevi	ice8(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* obData,WORD wCount);
Function	16-bit data
INT WINAPI ReadDevi	ice16(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount);
Function	32-bit data
INT WINAPI ReadDevi	ice32(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* odwData,WORD wCount);
Function	8-bit BCD data
INT WINAPI ReadDevi	iceBCD8(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* obData,WORD wCount);
Function	16-bit BCD data
INT WINAPI ReadDevi	iceBCD16(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount);
Function	32-bit BCD data
INT WINAPI ReadDevi wCount);	iceBCD32(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* odwData,WORD
Function	Single-precision floating point data
INT WINAPI ReadDevi	iceFloat(LPCSTR sNodeName,LPCSTR sDeviceName,FLOAT* ofIData,WORD wCount);
Function	Double-precision floating point data
INT WINAPI ReadDevi wCount);	iceDouble(LPCSTR sNodeName,LPCSTR sDeviceName,DOUBLE* odbData,WORD
Function	Character string data
INT WINAPI ReadDevi	iceStr(LPCSTR sNodeName,LPCSTR sDeviceName,LPSTR psData,WORD wCount);
Function	General-use data
INT WINAPI ReadDevi wAppKind);	ice(LPCSTR sNodeName,LPCSTR sDeviceName,LPVOID pData,WORD wCount,WORD
Function	General-use data (Variant-type)
INT WINAPI ReadDevi wCount,WORD wAppk	iceVariant(LPCSTR sNodeName,LPCSTR sDeviceName,LPVARIANT pData,WORD Kind);
Function	Group symbol
INT WINAPI ReadSym	bol(LPCSTR sNodeName,LPCSTR sSymbolName,LPVOID oReadBufferData);
Function	Group symbol (Variant-type)
INT WINAPI ReadSym	bolVariant(LPCSTR sNodeName,LPCSTR sSymbolName,LPVARIANT pData);
Function	TIME data
INT WINAPI ReadDevi wCount);	iceTIME(LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* odwData, WORD

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

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

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

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

■ Single-Handle Direct Read APIs

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

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

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

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

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

■ Single-Handle Direct Write APIs

INT WINAPI WriteDevic Function INT WINAPI WriteDevic Function INT WINAPI WriteDevic Function INT WINAPI WriteDevic Function	Bit data ceBitD(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* pwData,WORD wCount); 8-bit data ce8D(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* pbData,WORD wCount); 16-bit data ce16D(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* pwData,WORD wCount); 32-bit data ce32D(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* pdwData,WORD wCount); 8-bit BCD data ceBCD8D(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* pbData,WORD wCount);
FunctionINT WINAPI WriteDevicFunctionINT WINAPI WriteDevicFunctionINT WINAPI WriteDevicFunction	8-bit data ce8D(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* pbData,WORD wCount); 16-bit data ce16D(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* pwData,WORD wCount); 32-bit data ce32D(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* pdwData,WORD wCount); 8-bit BCD data
INT WINAPI WriteDevic Function INT WINAPI WriteDevic Function INT WINAPI WriteDevic Function	ce8D(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* pbData,WORD wCount); 16-bit data ce16D(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* pwData,WORD wCount); 32-bit data ce32D(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* pdwData,WORD wCount); 8-bit BCD data
FunctionINT WINAPI WriteDevicFunctionINT WINAPI WriteDevicFunction	16-bit data ce16D(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* pwData,WORD wCount); 32-bit data ce32D(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* pdwData,WORD wCount); 8-bit BCD data
INT WINAPI WriteDevic Function INT WINAPI WriteDevic Function	ce16D(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* pwData,WORD wCount); 32-bit data ce32D(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* pdwData,WORD wCount); 8-bit BCD data
Function INT WINAPI WriteDevic Function	32-bit data ce32D(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* pdwData,WORD wCount); 8-bit BCD data
INT WINAPI WriteDevic	ce32D(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* pdwData,WORD wCount); 8-bit BCD data
Function	8-bit BCD data
INT WINAPI WriteDevic	ceBCD8D(LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* pbData,WORD wCount);
Function	16-bit BCD data
INT WINAPI WriteDevic wCount);	ceBCD16D(LPCSTR sNodeName,LPCSTR sDeviceName,WORD* pwData,WORD
Function	32-bit BCD data
INT WINAPI WriteDevic wCount);	ceBCD32D(LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* pdwData,WORD
Function	Single-precision floating point data
INT WINAPI WriteDevic	ceFloatD(LPCSTR sNodeName,LPCSTR sDeviceName,FLOAT* pflData,WORD wCount);
Function	Double-precision floating point data
INT WINAPI WriteDevic wCount);	ceDoubleD(LPCSTR sNodeName,LPCSTR sDeviceName,DOUBLE* pdbData,WORD
Function	Character string data
INT WINAPI WriteDevic	ceStrD(LPCSTR sNodeName,LPCSTR sDeviceName,LPCSTR psData,WORD wCount);
Function	General-use data
INT WINAPI WriteDevic wAppKind);	ceD(LPCSTR sNodeName,LPCSTR sDeviceName,LPVOID pData,WORD wCount,WORD
Function	General-use data (Variant-type)
INT WINAPI WriteDevic wCount,WORD wAppKi	ceVariantD(LPCSTR sNodeName,LPCSTR sDeviceName,LPVARIANT pData,WORD and);
Function	Group symbol
INT WINAPI WriteSymb	polD(LPCSTR sNodeName,LPCSTR sSymbolName,LPVOID pWriteBufferData);
Function	Group symbol (Variant-type)
INT WINAPI WriteSymb	polVariantD(LPCSTR sNodeName,LPCSTR sSymbolName,LPVARIANT pData);
Function	TIME data
INT WINAPI WriteDevic wCount);	ceTIMED(LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* pdwData, WORD
Function	DATE data
INT WINAPI WriteDevic wCount);	ceDATED(LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* pdwData, WORD

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

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

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

■ Single-Handle Write APIs with Cache Refresh after Writing

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

Pro-Server EX Reference Manual

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

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

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

■ Multi-Handle Cache Read APIs

Function	Bit data
INT WINAPI ReadDev owData,WORD wCoun	iceBitM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* t);
Function	8-bit data
INT WINAPI ReadDev obData,WORD wCount	ice8M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* ;);
Function	16-bit data
INT WINAPI ReadDev owData,WORD wCoun	ice16M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* t);
Function	32-bit data
INT WINAPI ReadDev odwData,WORD wCou	ice32M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* nt);
Function	8-bit BCD data
INT WINAPI ReadDev obData,WORD wCount	iceBCD8M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,BYTE*
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 ReadDev odwData,WORD wCou	iceBCD32M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* nt);
Function	Single-precision floating point data
INT WINAPI ReadDev oflData,WORD wCount	iceFloatM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,FLOAT* t);
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
INT WINAPI ReadDev pData,WORD wCount,	iceM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,LPVOID WORD wAppKind);
Function	General-use data (Variant-type)
	iceVariantM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR ANT pData,WORD wCount,WORD wAppKind);
Function	Group symbol
INT WINAPI ReadSym oReadBufferData);	bolM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sSymbolName,LPVOID

Function	Group symbol (Variant-type)									
INT WINAPI ReadSymbolVariantM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sSymbolName,LPVARIANT pData);										
Function	TIME data									
INT WINAPI ReadDevi odwData, WORD wCou	iceTIMEM(HANDLE hProServer, LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* int);									
Function	DATE data									
INT WINAPI ReadDevi odwData, WORD wCou	iceDATEM(HANDLE hProServer, LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* int);									
Function	TIME_OF_DAY data									
INT WINAPI ReadDeviceTIME_OF_DAYM(HANDLE hProServer, LPCSTR sNodeName, LPCSTR sDeviceName, DWORD* odwData, WORD wCount);										
Function	DATE_AND_TIME data									
INT WINAPI ReadDeviceDATE_AND_TIMEM(HANDLE hProServer, LPCSTR sNodeName, LPCSTR sDeviceName, QWORD* oqwData, WORD wCount);										
* For each parameter, please refer to "■ Parameters of read/write functions".										

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

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

Multi-Handle Direct Read APIs

Function	Bit data							
	INT WINAPI ReadDeviceBitDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* owData,WORD wCount);							
Function	8-bit data							
INT WINAPI ReadDevice8DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,BYTE* obData,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 wCour	ce32DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* nt);							
Function	8-bit BCD data							
INT WINAPI ReadDevi obData,WORD wCount	ceBCD8DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,BYTE*);							
Function	16-bit BCD data							
INT WINAPI ReadDevi owData,WORD wCount	ceBCD16DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* t);							
Function	32-bit BCD data							
INT WINAPI ReadDevi odwData,WORD wCour	ceBCD32DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* nt);							
Function	Single-precision floating point data							
INT WINAPI ReadDevi oflData,WORD wCount	ceFloatDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,FLOAT*);							
Function	Double-precision floating point data							
	ceDoubleDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR * odbData,WORD wCount);							
Function	Character string data							
INT WINAPI ReadDevi psData,WORD wCount)	ceStrDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,LPSTR);							
Function	General-use data							
INT WINAPI ReadDevi pData,WORD wCount,V	ceDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,LPVOID WORD wAppKind);							
Function	General-use data (Variant-type)							
	ceVariantDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR ANT pData,WORD wCount,WORD wAppKind);							
Function	Group symbol							
INT WINAPI ReadSym oReadBufferData);	bolDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sSymbolName,LPVOID							
Function	Group symbol (Variant-type)							
INT WINAPI ReadSym sSymbolName,LPVARI	bolVariantDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR ANT pData);							
	07.00							

Pro-Server EX Reference Manual

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

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

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

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

Multi-Handle Direct Write APIs

Function	Bit data
INT WINAPI WriteDev pwData,WORD wCoun	iceBitDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* t);
Function	8-bit data
INT WINAPI WriteDev pbData,WORD wCount	ice8DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,BYTE*);
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	8-bit BCD data
INT WINAPI WriteDev pbData,WORD wCount	iceBCD8DM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,BYTE*);
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* t);
Function	Double-precision floating point data
	riceDoubleDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR E* 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,	iceDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,LPVOID WORD wAppKind);
Function	General-use data (Variant-type)
	riceVariantDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR ANT pData,WORD wCount,WORD wAppKind);
Function	Group symbol
INT WINAPI WriteSyn pWriteBufferData);	bolDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sSymbolName,LPVOID
Function	Group symbol (Variant-type)
INT WINAPI WriteSyn sSymbolName,LPVARI	nbolVariantDM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR ANT pData);

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

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

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

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

■ Multi-Handle Write APIs with Cache Refresh after Writing

Function	Bit data
	iceBitM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD*
pwData,WORD wCoun	
Function	8-bit data
INT WINAPI WriteDev pbData,WORD wCount	ice8M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,BYTE*);
Function	16-bit data
INT WINAPI WriteDev pwData,WORD wCoun	ice16M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* t);
Function	32-bit data
INT WINAPI WriteDev pdwData,WORD wCou	ice32M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* nt);
Function	8-bit BCD data
INT WINAPI WriteDev pbData,WORD wCount	iceBCD8M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,BYTE*);
Function	16-bit BCD data
INT WINAPI WriteDev pwData,WORD wCoun	iceBCD16M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,WORD* t);
Function	32-bit BCD data
INT WINAPI WriteDev pdwData,WORD wCou	iceBCD32M(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,DWORD* nt);
Function	Single-precision floating point data
INT WINAPI WriteDev pflData,WORD wCount	iceFloatM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,FLOAT*);
Function	Double-precision floating point data
INT WINAPI WriteDev pdbData,WORD wCour	iceDoubleM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,DOUBLE* nt);
Function	Character string data
INT WINAPI WriteDev psData,WORD wCount	iceStrM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,LPCSTR);
Function	General-use data
INT WINAPI WriteDev pData,WORD wCount,V	iceM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sDeviceName,LPVOID WORD wAppKind);
Function	General-use data (Variant-type)
	iceVariantM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR ANT pData,WORD wCount,WORD wAppKind);
Function	Group symbol
INT WINAPI WriteSyn pWriteBufferData);	bolM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR sSymbolName,LPVOID
Function	Group symbol (Variant-type)
INT WINAPI WriteSym sSymbolName,LPVARI	bolVariantM(HANDLE hProServer,LPCSTR sNodeName,LPCSTR ANT pData);
	07.94

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

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

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

Parameters of read/write functions

```
< Argument >
```

bsNodeName : Pointer to node name (character string)

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

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

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

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

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

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

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

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

pxxData

: Pointer to read/write target data

Accessible data types and corresponding argument types are listed below.

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

wCount

: Quantity of read/write target data

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

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

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

wAppKind : Data type specification

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

* Unable to use with VB functions.

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

< Return value >

Normal end: 0

Abnormal end: Error code

< Special Note >

When using the Read/WriteDeviceBit function:

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

Example: When wCount is "20"

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

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

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

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

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

When using the string data functions:

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

27.3 Cache Buffer Control APIs

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

Argumen				
CacheNa	me: (In) Cache buffer name			Return value
		ce into the	e cache buffer specified with this	Normal end: 0 Abnormal end: Error code
NodeNat	name. me: (In) Entry node name with ca	che soure	e Device/PLC name	Abhormar chu. Error couc
	In: (III) Entry node name with ca In) Cache source device	che sourc	e Device/PLC name	
		vou can d	irectly specify the device address, or	
	specify a symbol or group register			
	group, multiple symbols can be re	gistered a	at once.	
	d: (In) Source device data type			
	Available data types vary dependi method.	ng on the	cache source device designation	
	a) When device address of cache	source de	vice is directly specified:	
	Specify a data type (1 to 20) avail			
	specified.			
Value	Data type	Value	Data type	
1	Bit	11	Double-precision floating point	
2	16 bits, Signed decimal	12	Character string	
3	16 bits, Unsigned decimal	13	8 bit (Signed) data	
4	16 bits, Hexadecimal	14	8 bit (Unsigned) data	
5	16 bits, BCD	15	8 bit (HEX) data	
6	32 bits, Signed decimal	16	8 bit (BCD) data	
7	32 bits, Unsigned decimal	17	TIME data	
8	32 bits, Hexadecimal	18	TIME_OF_DAY data	
9	32 bits, BCD	19	DATE data	
10	Single-precision floating point	20	DATE_AND_TIME data	
vCount: (1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	the symbol type specified in symbol c) When group is specified for car Fixed to "0". The symbol type is registered for (In) Device data quantity subjecte Available values vary depending on method. a) When device address of cache Data quantity (1 to 2040) accordin maximum value varies depending b) When symbol is specified for c If you specify "0", the quantity sp If you specify any value other tha	able with bol definit che source all symbol d to cachi on the cac source de ng to the de ache sour ecified in n 0, data d	Pro-Server EX'. If you specify "0", tion is used. e device: ols in the specified group. ing the source device specification vice is directly specified: device type can be used. (The evice type.) the device:	

Function	Starting caching	
Starts caching.		
Multi	Cache(LPCSTR sCacheName); CacheM(HANDLE hProServer, LPCSTR sCacheName);	
Argument sCacheName: (In) Nam	e of cache buffer to start r 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 cach Caching stops, but defi To restart caching, call	nition of the cache buffer is retained.	
Multi	Cache(LPCSTR sCacheName); CacheM(HANDLE hProServer, LPCSTR sCacheName);	
	e of cache buffer to stop r 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.		
Multi	acheStatus(LPCSTR sCacheName);	
INT WINAPI PS_GetC	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.
· · ·	r name registered with 'Pro-Studio EX' can be also specified.	1: Caching in progress 2: Caching under suspension XX: Error code

Function	Discarding cache buffer					
Stops caching, and disca	Stops caching, and discards the cache buffer.					
Multi	byCache(LPCSTR sCacheName); byCacheM(HANDLE hProServer, LPCSTR sCacheName);					
Argument Return value sCacheName: (In) Name of cache buffer to be discarded Normal end: 0 A cache buffer name registered with 'Pro-Studio EX' can be also specified. Abnormal end: Error code						
Special Note		1				
Function	Setting cache update notification function					
Sets the function to notif	fy cache buffer update status to a specified window.					
device is frequently cach 'Pro-Server EX' can send has a change with the co If your system is built so be improved. This API allows you to s message" in 'Pro-Server After these settings are in Single INT WINAPI PS_SetNo LPARAM LParam, HAN Multi INT WINAPI PS_SetNo message, WPARAM WI	d a message to a specified window, when cache data is updated (workstant monitoring method, or when one polling cycle is complete as to execute cache-reading of a device after receiving this messages et "Target cache buffer name", "Window to receive the message"	when at least one target device and with the polling method). Age, the system efficiency can by and "Contents of the ently-set notification function. essage, WPARAM WParam,				
Argument Return value sCacheName: (In) Cache buffer name Normal end: 0 A cache buffer name registered with 'Pro-Studio EX' can be also specified. Normal end: 0 hWnd: (In) Handle for the window to receive the message Mormal end: Error cod message: (In) Message ID to be sent to the window WParam: (In) WPARAM value to be sent to the window together with message ID LParam: (In) LPARAM value to be sent to the window together with message ID Normal end: Error cod ohCacheNotifyID: (Out) Returns the ID that identifies the currently set notification Image: Colored also set in the image: Color						
After the cache buffer is wParam value (specified window (hWnd).	not necessary, discard it with PS_KillNotifyFromCache(). updated, call PostMessage() to send the message (specified with l with the third argument), and LParam value (specified with the f nge(), refer to the Windows API Manual.					

Function	Accepting next cache update notification				
Accepts the next cache upo	late notification.				
However, once this notifica again, even if the cache bu notification routine, a mult update message. (If the not call error occurs with the re To prevent this error, this A By calling this API at the e	he function to send a message to a specified window when ation function is executed, 'Pro-Server EX' will not send a ffer is updated next. This is because in case it has taken a iple-call error can occur with the relevant routine when 'P ification routine receives the next message before comple- putine.) API explicitly informs 'Pro-Server EX' that it can send the end of the processing of the notification routine, you can be ry time a cache buffer is updated.	a message until this API is called long time in processing with the Pro-Server EX' sends the next cach etion of the processing, a multiple e next message.			
Multi	lextNotifyFromCache(HANDLE hCacheNotifyID); lextNotifyFromCacheM(HANDLE hProServer, HANDL	E hCacheNotifyID);			
Argument hCacheNotifyID: (In) ID o ID obtained with PS_SetNo	f next message acceptance notification function otifyFromCache()	Return value Normal end: 0 Abnormal end: Error code			
Special Note					
Function	Canceling cache update notification				
Cancels the function for se	nding a cache buffer update message to a specified windo	OW.			
	rver EX' will not send a cache buffer update message to the CacheNotifyID is updated.	he relevant window, even if the			
Multi	fyFromCache(HANDLE hCacheNotifyID); fyFromCacheM(HANDLE hProServer, HANDLE hCach	neNotifyID);			
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	l discard a message sent from 'Pro-Server EX', even if the X' has sent a message to a window and the application ha called, the application can fetch the message from the wi	is not fetched the message from th			

window before this API is called, the application can fetch the message from the window even after this API is called. (Depending on the timing, the notification routine may be called even after this API is called.)

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

27.4 Queuing Access Control APIs

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

Function	Executing device read/write request in queue	
Accesses device data accor	ding to the device read/write request in queue.	
Single INT WINAPI ExecuteQuer Multi INT WINAPI ExecuteQuer	uingAccess(); uingAccessM(HANDLE hProServer);	
Argument		Return value Normal end: 0 Abnormal end: Error code
'Pro-Server EX' fails to a If you wish to know whe	ssfully accesses all specified devices, ExecuteQueuingAccess() access any device, on the other hand, ExecuteQueuingAccess() other each device access request has been successfully executed eded() to check the result. TONs in queuing access.	returns an access error code.
Function	Discarding device read/write request in queue	
Discards the device read/w	rite request in queue.	
Single INT WINAPI CancelQueu Multi INT WINAPI CancelQueu	ingAccess(); ingAccessM(HANDLE hProServer);	
Argument		Return value Normal end: 0 Abnormal end: Error code
ExecuteQueuingAccess() is) or BeginQueuingRead()is called, 'Pro-Server EX' queues devi s called. hes unnecessary for any reason, call this API. 'Pro-Server EX' d	•

Function	Checking the run result of device read/write request in queue	9
Checks whether or not each called.	h device access request has been successfully executed, after H	ExecuteQueuingAccess() is
Multi	ccessSucceeded(INT iIndex); ccessSucceededM(HANDLE hProServer,INT iIndex);	
Argument iIndex: (In) Number of req	uest to be checked	Return value XX: Error code 0: Indicates that the
called several times to que called. Note that you canno ExecuteQueuingAccess(). If you wish to know a resu) or BeginQueuingRead() is called, Device Access APIs are ue device access requests until ExecuteQueuingAccess() is of know an actual device access result until execution of It of each device access request, execute irst, and then specify the number (from 0) of the request for	device access request of the specified number has been successfully executed.
WriteDevice16("No	ode1","LS100",Data,10); ode1","LS200",Data,10); ode1","LS300",Data,10);	
	cess to "LS200" has been successfully executed, use IsQueuir is access has been successfully executed.	ngAccessSucceeded(1).

27.5 System APIs

Function	Creating Pro-Server handle	
Obtains a Pro-Server handl	e for use of a Multi-Handle function.	
HANDLE WINAPI Create	ProServerHandle();	
Argument		Return value Normal end: Other than 0 (Handle code) Abnormal end: 0
Special Note		
Function	Releasing Pro-Server handle	
Releases an obtained Pro-S	Server 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 = T etworkProjectM(HANDLE hProServer,LPCSTR sDBName,D'	
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	
Converts an error code retu EasyLoadErrorMessage() 1	returns a multibyte character string (ASCII) as a message. Easy ing (UNICODE) as a message.	LoadErrorMessageW()
	ErrorMessage(INT iErrorCode,LPSTR osErrorMessage); ErrorMessageW(INT iErrorCode,LPWSTR owsErrorMessage));
osErrorMessage: (Out) Poi string) storing area. (osErrorMessage: (Out) Poi	e returned by 'Pro-Server EX' function nter to the converted character string (multibyte character (To call this API, secure a storing area with at least 512 bytes.) nter to the converted character string (multibyte character (To call this API, secure a storing area with at least 1024	Return value Normal end: Other than 0 Failure in character string conversion (ex. Undefined code): 0
	ensure compatibility with older versions of 'Pro-Server'. essageEx() enables conversion into a more detailed error messag Ex().	ge. We recommend you to use
Function	Converting error code into character string (with status inform	nation)
possible. EasyLoadErrorMessage() a EasyLoadErrorMessageEx error occurrence place and return a different error mes EasyLoadErrorMessageEx message.	as the error message together with the error occurrence conditional ways returns the same error message relative to a specified error () returns more detailed error information including a name of c so on, depending on the error occurrence condition. Thus, Easy sage relative to the same error code, depending on the situation () and EasyLoadErrorMessageExM() return a multibyte charact W() and EasyLoadErrorMessageExWM() return a wide charact	for code. On the other hand, communication target device, LoadErrorMessageEx() may ter string (ASCII) as a
BOOL WINAPI EasyLoad Multi BOOL WINAPI EasyLoad	ErrorMessageEx(INT iErrorCode,LPSTR osErrorMessage); ErrorMessageExW(INT iErrorCode,LPWSTR owsErrorMessa ErrorMessageExM(HANDLE hProServer,INT iErrorCode,LPS ErrorMessageExWM(HANDLE hProServer,INT iErrorCode,L	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. Therefores an error and EasyLoadErrorMessage(),EasyLoadErrorMessage se stored error status information is rewritten. For this reason, we M(), you must specify the same Pro-Server handle as the handle	re, if you call another API sage() will not return error when using

Function	Initializing Pro-Server API	
	API, and declares use of the API internally. vithout starting 'Pro-Server EX', 'Pro-Server EX' will automatica	ally start.
INT WINAPI EasyInit();		
Argument		Return value Normal end: 0 Abnormal end: Error code
Special Note		
Function	Ending Pro-Server API	
INT WINAPI EasyTerm();		
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	
0	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		

Function	Pro-Server EX closing notice	
When 'Pro-Server EX' start using PostMessage() of Wi For details of PostMessage		
LPARAM LParam = 0); Multi INT WINAPI EasyNotifyF	romServerEnd(HWND hReceivedWnd,UINT uMessage,WPA) romServerEndM(HANDLE hProServer,HWND hReceivedWn ram = 0, LPARAM LParam = 0);	
Argument		Return value
uMessage: (In) Message ID This ID will be sent t EX is being closed.	by that receives a closing message. To be sent as a closing message. To the window specified with hReceivedWnd when Pro-Server	Normal end: 0 Abnormal end: Error code
PostMessage())	b be sent together with the message (Value of WPARAM in be sent together with the message (Value of LPARAM in	
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 E2 the application main window for hReceivedWnd, and WM_QU 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 cal In this case, however, the p and the processing of impo You can specify whether to	APIs (functions) process Windows messages during the process ong. This API can specify whether to execute or inhibit the Win rocessing is inhibited, the relevant Windows message is stored g execution of a function. Il a function over again by clicking the icon during execution of rocessing of all the Windows messages as well as an "icon click rtant messages for timer and window re-drawing is also disable execute or inhibit the processing of Windows messages for eac essage processing has been set to "Execute".	ndows message processing. in the message queue, and f the function. c'' message, will be inhibited, ed.
Single INT EasySetWaitType(DW Multi INT EasySetWaitTypeM(H	'ORD dwMode); ANDLE hProServer,DWORD dwMode);	
Argument hProServerHandle: (In) Pro dwMode: (In) To execute n	-Server handle subjected to processing mode change hessage processing, specify "1". rocessing, specify "2".	Return value Normal end: 0 Abnormal end: Error code
Special Note		

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

Function Ad	ding log into log v	iewer	
the event.		rror, etc.) occurs with internal processin	
		log viewer. (See "28.5 Monitoring Syste message by using this function. This AP	
INT WINAPI EasyOutputLog(BYTE bLevel,LPC	CSTR sPrompt,LPCSTR sMessage);	
'Pro-Server EX' provides	a filtering function at the current record	rmance deterioration. To prevent this, n for recording messages by event type. rding message belongs to.	Return value Normal end: 0 Abnormal end: Error code
	Hexad		
Definition	ecimal	Event type	
	value		
EASY_LogLevel_SysMess	sage 0x01	System message	
EASY_LogLevel_SysError	· 0x02	System error message	
EASY_LogLevel_AppError	r 0x04	User program error message	
EASY_LogLevel_AppStart	0x08	User program starting message	
EASY_LogLevel_AppEnd	0x10	User program closing message	
EASY_LogLevel_AppWarr	ning 0x20	User program warning message	
EASY_LogLevel_AppMess	sage1 0x40	User program detail message 1	
EASY_LogLevel_AppMess	sage2 0x80	User program detail message 2	
sMessage: (In) Character string	of the message to	currence position (NULL-terminated) be recorded (NULL-terminated) nation of two character stings (sPrompt	
Special Note			

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

27.6 SRAM Data Access APIs

Function

Argument

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 Normal end: 0

Abnormal end: Error code

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

iBackupDataType: (In) Type of data to be read

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

When the data source node is in the GP4000/LT4000 Series/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.
(1) Alarm history
(2) Alarm log
(3) Alarm active
If there is no valid data, an error occurs.
iSaveMode: (In) Saving mode

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

Others: Reserve

When reading Alarm or Sampling data, the date format is "yy/mm/dd".

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 GP4000/LT4000 Series, GP3000 Series, WinGP and LT3000.

INT WINAPI EasyBackupDataReadEx(LPCSTR sSaveFileName, LPCSTR sNodeName, INT iBackupDataType, INT iSaveMode, INT iNumber = 0, INT iStringTable = 0x0000);

Argument

sSaveFileName: (In) File path of the file to save read data. (String pointer) sNodeName: (In) Name of read data source node (String pointer) Pro-Server EX nodes cannot be specified. iBackupDataType: (In) Type of data to be read Return value Normal end: 0 Abnormal end: Error code

Value	Data source node	Data source node	
value	in GP Series	other than GP Series	
)x0001	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	
)x0005	Alarm block 1	Alarm block 1	
10000	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.		
0: Ne and o 1: Ad	verwrites it.)	already exists, 'Pro-Server EX' deletes the file, end of an existing file. If there is no file to save w file.))	

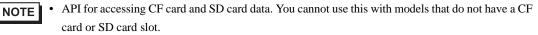
Others: Reserve

iNumber: (In) This argument is ignored when sSaveFileName specifies a GP Series file. In addition, the meaning of this argument varies depending on the value of iBackupDataType. Value of Description iBackupDataType Three types of alarm data (active, history and log) are available. Specify a target alarm type. Value of Description iNumber '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. 0 (1) Alarm history 0x0005 to 0x000C (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. Group number of sampling group to be read 0x8002 Any value from 1 to 64 Others (Reserve) iStringTable: (In) Reserve Always specify "0". **Special Note**

When reading Alarm or Sampling data, the date format is "yy/mm/dd".

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 WinGP nodes or LT3	e path of binary-formatted filing data to be written (String entry node to write data (String pointer) 5, GP4000/LT4000 Series nodes, GP3000 Series nodes, 6000 nodes cannot be specified. 6d to "1". ("1" indicates filing data.)	Return value Normal end: 0 Abnormal end: Error code	
Special Note		1	

27.7 CF Card / SD Card APIs



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

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

nction Reading CF card status				
Acquires connection status of the CF card in a connected the display unit.				
Single CF Card: INT WINAPI EasyIsCFCard(LPCSTR SD Card: INT WINAPI EasyIsSDCard(LPCSTR Multi CF Card: INT WINAPI EasyIsCFCardM(HAND SD Card: INT WINAPI EasyIsSDCardM(HAND	sNodeName); LE hProServer,LPC			
Argument hProServer: Pro-Server handle	Return value			
sNodeName: Name of node to read status (This	Function	For GP Series	Other than GP	
node name must be pre-registered in a network project.)	return value	node	Series node	
1 J	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	
	0x1000008	CF card not initialized		
	Others	Error without re	elation to CF card	
Special Note				

Function	Reading file list from CF card (Optional folder name)	
Outputs a list of files from the CF card inserted in a display unit node into a file specified with the parameter. You can specify an optional file to save the file list.		
sSaveFileName);	syGetListInCfCard(LPCSTR sNodeName, LPCSTR sDi	
sSaveFileName);	syderLisunisuCard(LI CS IK sivudeivaine, LI CS IK sD	needory, hvi * ofCount, Li CSTK
Argument		Return value
sNodeName: Name of node	e to output file list	Normal end: 0
-	to receive file list (All capitals)	Abnormal end: Error code
oiCount: Number of output	files	
	sSaveFileName: Name of file to save output directory information. The specified file	
stores binary data of the alignment type specified with stEasyDirInfo, in the		
quantity specified wi	th the return value of oiCount.	
struct stEasyDirInfo {		
BYTE bFileName[8+1]:// File name (Terminated with "0")		
BYTE bExt[3+1];// File extension (Terminated with "0")		
BYTE bDummy[3];// Dummy		
DWORD dwFileSize;// File size		
BYTE bFileTimeStamp[8+1];// File timestamp (Terminated with "0")		
BYTE bDummy2[3];// Dummy 2		
};		

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

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

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

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

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

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

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

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

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

Other than GP series node

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

GP series node

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

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

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

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

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

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

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

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

Function	Reading file list from CF card (Type specification)	
	he CF card inserted in a display unit into a file specified with ith "sDirectory" can be output.	the parameter. Only the file list
INT WINAPI EasyGetListl sSaveFileName);	inCard(LPCSTR sNodeName, LPCSTR sDirectory, INT* oi	Count, LPCSTR
following directories: LOG (Logging data) TREND (Trend data) ALARM (Alarm data CAPTURE (Capture FILE (Filing data) oiCount: Number of output sSaveFileName: Name of f stores binary data of quantity specified wi struct stEasyDirInfo BYTE bFileName[8- BYTE bFileName[8- BYTE bDummy[3];// DWORD dwFileSize	ry to output list (All capitals) This API supports only the (A) (A) (A) (A) (A) (A) (A) (A)	Return value Normal end: 0 Abnormal end: Error code

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

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

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

Other than GP series node

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

GP series node

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

	Function	Rea	ading file list from CF card (Including sub-folders in Type	specification)	
in the dir		d with "	CF card inserted in a display unit into a file specified with the sDirectory" can be output. Get the list of files to read by sory".		
INT WI sSaveFil	-	istRecu	rsivelyInCard(LPCSTR sNodeName, LPCSTR sDirectory	7, INT* oiCount, LPCSTR	
sDirector following LC TH AI CA	me: Name of n	ctory to uta) ata) data) ure data	output list (All capitals) This API supports only the	Return value Normal end: 0 Abnormal end: Error code	
oiCount: sSaveFil sto	Number of out Name: Name o res binary data	put file of file to of the a	s o save output directory information. The specified file alignment type specified with stEasyRecursiveDirInfo, in th the return value of oiCount.		
BY BY DV BY Stor BY };	TE bExt[3+1] TE bDummy[VORD dwFiles TE bFileTime TE bFolderNa red in remainir TE bDummy2	e[8+1];/ ;// File o 3];// Du Size;// F Stamp[8 .me[260 ng portio	File size 8+1];// File timestamp (Terminated with "0") 0+1];// Folder Name (Terminated with "0", "0" is also pons.)		
When rea	u set a GP Seri ading the file lis	st, file r	in this API, it will become an error. names shorter than 8 characters or file extensions shorter th 1] respectively, as shown below.	an 3 characters are displayed	
bFile	Name[8+1]		the file name is shorter than 8 characters, null (0x00) is st al file name, and undefined values are stored after null.	ored at the end of the	
bExt[3+1]When the file extension is shorter than 3 characters, null (0x00) is stored at the end of to original file extension, and undefined values are stored after null.					
(Exampl	e) When ABC.	D is the	file name and file extension		
1	bFileName	[8+1]	0x410x420x430x00********************************	icate an undefined value)	
	bExt[3+1]		0x440x00******* (**** indicate an undefined value)		

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

Function	Reading file from CF card (Type specification)	
Reads a specified file from the CF card. Only the file type specified with "pReadFileType" can be read.		
INT WINAPI EasyFileRea sWriteFileName, DWORD	dCard(LPCSTR sNodeName, LPCSTR pReadFileType, W0 * odwFileSize);	ORD wReadFileNo, LPCSTR
Argument		Return value
Argument sNodeName: Name of nod	le to output file list	Return value Normal end: 0
sNodeName: Name of nod	le to output file list urce file to be read from CF card (See <special note="">)</special>	
sNodeName: Name of noc pReadFileType: Type of so	1	Normal end: 0

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
Trend graph data	ZT	TREND
Sampling data	ZS	TREND
Alarm block 4 to 8	Z4 to Z8	ARAM
Logging data	ZL	LOG
Alarm Log	ZG	ALARM
Alarm History	ZH	ALARM
Alarm Active	ZA	ALARM
Screen data backup	ZC	MRM
GP Screen data (Jpeg)	СР	CAPTURE

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

■File types supported for GP4000/LT4000 series node, GP3000 Series node and WinGP node

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

Function	Writing file into CF card (Optional file name specification)		
Writes a specified file into	the CF card. You can specify an optional file to write.		
sFolderName, LPCSTR sF	syFileWriteInSdCard(LPCSTR sNodeName, LPCSTR pRea		
sFolderName: Name of fole characters)	source file to be written into CF card (Full path) der containing target file in CF card (Up to 32 single-byte	Return value Normal end: 0 Abnormal end: Error code	
Special Note	file in CF card (Up to 8.3 format character string)		

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

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

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

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

Special Note

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

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

■File types supported for GP Series node

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

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

■File types supported for GP4000/LT4000 Series node, GP3000 Series node and WinGP node

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

Function	Renaming file in CF card					
Renames a specified file in	the CF card.					
CF Card: INT WINAPI EasyFileRenameInCfCard(LPCSTR sNodeName, LPCSTR sFolderName, LPCSTR sFileName,LPCSTR sFileRename); SD Card: INT WINAPI EasyFileRenameInSdCard(LPCSTR sNodeName, LPCSTR sFolderName, LPCSTR sFileName,LPCSTR sFileRename);						
Argument		Return value				
sNodeName: Name of node		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	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		1				

Function	Acquiring information on CF card empty space				
Acquires information on empty space in the CF card connected to a specified entry node.					
	syGetCfFreeSpace(LPCSTR sNodeName,INT* o				
CF Card: INT WINAPI EasyGetCfFreeSpaceEx(LPCTSTR sNodeName,INT* pioUnallocatedL,INT* pioUnallocatedH);					
SD Card: INT WINAPI EasyGetSdFreeSpace(LPCSTR sNodeName,INT* oiUnallocated);					
SD Card: INT WINAPI EasyGetSdFreeSpaceEx(LPCTSTR sNodeName,INT* pioUnallocatedL,INT* pioUnallocatedH);					
Argument					
sNodeName: Name of nod	e to output file list	Return value			
oiUnallocated (*1): Empty space in CF card (number of bytes) Normal end: 0					
pioUnallocatedL: (Out) Empty space in bottom 4 bytes Abnormal end: Error c					
pioUnallocatedH: (Out) Empty space top 4 bytes					

Special Note

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

'Pro-Server EX' uses a special protocol to access the CF card in a GP Series node. However, to access a GP4000/ LT4000 Series node, GP3000 Series node and WinGP node, FTP protocol is used. 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);					
rgument		Return value			
iPassive: (In) 0: Normal mode Normal end: 0					
Other than 0: Passive mode Abnormal end: Error code					
At initialization of ProEasy, the FTP protocol is set to "Normal mode".					
pecial Note					

27.8 Binary Date and Time / Text Display Conversion

Convert from binary value to text API

Function	Binary va	lue text con-	version (Time	-type)	
Function to conve	ert binary value	e to TIME-ty	pe string.		
INT WINAPI Ea	syTIMEToStri	ng(DWORD	dwData, LPS'	TR osTime);	
Argument dwData: (In) Bin osTime: (Out) Co	ary value prior onverted text st	to conversion ring ^{*1}	on		Return value Normal end: 0 Abnormal end: Error code
Special Note Input Format					
31	24	16	8	0	
	Elapsed tim	e in millisecond	s (Signed)		
(1) 01d02h03m04 (2) -02d03h04m0 Function)5s006ms	lue text conv	version (TIME	E_OF_DAY-type	2)
Function Function to conve	-				
INT WINAPI Ea	syTIME_OF_I	DAYToString	g(DWORD dw	Data, LPSTR o	sTod);
Argument dwData: (In) Bin osTod: (Out) Cor			'n		Return value Normal end: 0 Abnormal end: Error code
Special Note Input Format					
Reserved 0	27 21 Hours Min GMT bit 0 or bit 0	15 utes Seco	9 nds Millisa	0 econds	
<i>Output Format</i> %02u:%02u:%02 <i>Output Example</i> 23:59:59.999	2u.%03u (hours	s, minutes, se	conds, millise	econds)	

Function	Binary value text conv	orgion (DATE t	vna)		
	binary value to DATE-ty		yhe)		
Function to convert	billary value to DATE-ty	pe sunig.			
INT WINAPI EasyI	DATEToString(DWORD	dwData, LPSTI	R osDate);		
Argument dwData: (In) Binary osDate: (Out) Conve	value prior to conversion erted text string ^{*1}	n		No	turn value rmal end: 0 normal end: Error code
Special Note Input Format				·	
31	24 21	8	4 0		
Reserved 0	Day 0 Year	Month	Date		
Error b	bit 0				
%04u-%02d-%02u (<i>Output Example</i> 2012-01-01	year, month, date)				
Function	Binary value text conv	ersion (DATE_	AND_TIM	IE-type)	
Function to convert	binary value to DATE_A	ND_TIME-type	e string.		
INT WINAPI EasyI	DATE_AND_TIMEToStr	ing(QWORD q	wData, LF	PSTR osDt);	
Argument dwData: (In) Binary osDt: (Out) Convert	value prior to conversion ed text string ^{*1}	n		No	turn value rmal end: 0 normal end: Error code
Special Note Input Format					
63	31 Date	Time	0		
Error bit		or bit 0]		
<i>Output Format</i> %04u-%02u-%02u-% <i>Output Example</i>	%02u:%02u:%02u.%03u	(year, month, d	late, hours	, minutes, seconds, mill	iseconds)
I					

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

 $^{\ast 2}$ For information about each device access API, refer to 27.2 Device Access APIs.

Convert from text to binary value API

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

Special Note

Input Format

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

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

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

Input Example

01d02h03m04s005ms

Function	INT WINAPI EasyStringToTIME_OF_DAY()

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

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

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

Special Note

Input Format

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

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

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

Input Example 23:59:59.999

Functio	on	INT WINAPI	EasyStringToDA	TE()				
Function to c	function to convert DATE-type string to a binary value.							
INT WINAP	PI EasySt	ringToDATE(L	PCSTR sDate, D	WORD *pdwD	ata);			
		g prior to conve verted binary va				Return value Normal end: 0 Abnormal end: Error code		
Special Note Input Forma %04u-%02d	t	rear, month, da	te)					
			Year	Month	Date			
	Setup ra	inge	19708191	112	131			
	Units (s	eparator)	-	-				
• I Input Examp 2012-01-01		the items in the	e setup range as p	er the input for	nat.			
Functio	on	INT WINAPI	EasyStringToDA	TE_AND_TIM	E()			
Function to c	convert E	DATE_AND_T	IME-type string to	o a binary value	·			
INT WINAP	PI EasySt	ringToDATE_/	AND_TIME(LPC	STR sDt, QWC	ORD *pqwData);			
ArgumentReturn valuesDt: (In) Text string prior to conversionNormal end: 0pdwData: (Out) Converted binary valueAbnormal end: Error cod								
Special Note	e t					1		

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

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

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

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

27.9 Other APIs

Function	Reading time from GP (DWORD-type)	
Acquires current tim 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 I	EasyGetGPTime(LPCSTR sNodeName, DWORD* odwTime);	
odwTime: Acquired	of target node (A Pro-Server EX node cannot be specified.) time (Time is acquired as a value of DWORD type, (substantially, fined by ANSI).)	Return value Normal end: 0 Abnormal end: Error code
Special Note		
Function	Reading time from GP (VARIANT-type)	
words from LS2048	e of a specified node as a Variant-type value. This function is valid or EasyGetGPTimeVariant(LPCSTR sNodeName, LPVARIANT ovTime	
ovTime: Acquired ti	of target node (A Pro-Server EX node cannot be specified.) me (Time is acquired as a value of VARIANT type. Internal mat is "Date".)	Return value Normal end: 0 Abnormal end: Error code
Special Note		
Function	Reading time from GP (STRING-type)	
saved in 6 words fro	e of a specified node as an LPTSTR-type character string. This functi m LS2048. EasyGetGPTimeString(LPCSTR sNodeName, LPCSTR sFormat, LPS	-
Argument sNodeName: Name pFormat: String to s specification of in <special n<br="">Other characte osTime: Time acquin + 1 (NULL) is this, you must</special>	of target node (A Pro-Server EX node cannot be specified.) pecify the format of time to be acquired as a string. The format codes subsequent to the percentage (%) symbol are changed as shown	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".)

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

Function	Reading time from GP (STRING VARIANT-type)						
-	Acquires current time of a specified node as a Variant-type character string. This function is valid only for the time saved in 6 words from LS2048.						
DWORD WINAPI E	asyGetGPTimeStringVariant(LPCSTR sNodeName, LPCSTR sForm	nat, LPVARIANT ovTime);					
Argument Return value sNodeName: Name of target node (A Pro-Server EX node cannot be specified.) Normal end: 0 pFormat: String to specify the format of time to be acquired as a string. The format specification codes subsequent to the percentage (%) symbol are changed to those listed below. Other characters are expressed without a change. (For details, refer to <special note=""> of "Reading time from GP (STRING-type)".) Normal end: 0 ovTime: Time acquired as a string (Time is acquired as VARIANT type. Internal possessing format is "BSTR".) Normal end: 0</special>							
Special Note							
Function	Reading entry node status						
osSystemVersion,LP Multi INT WINAPI GetNo	tus. deProperty(LPCSTR sNodeName,DWORD dwTimeLimit,LPSTR or STR osComVersion,LPSTR osECOMVersion); dePropertyM(HANDLE hProServer,LPCSTR sNodeName,DWORD SystemVersion,LPSTR osComVersion,LPSTR osECOMVersion);						
Argument hProServer: (In) Pro-Server handle sNodeName: (In) Name of node to read status dwTimeLimit: (In) Response time-out setting value (If "0" is specified, it is set to the default value of 3000 ms.) The setting range is from 1 to 2,147,483,647. (Unit: ms)Return value Normal end: 0 Abnormal end: Error complexity Abnormal end: Error complexityThe API returns status information on the target node to the following area. Secure an area of at least 32 bytes for each item. osGPType: (Out) Display unit model code osSystemVersion: (Out) Display unit system version osComVersion: (Out) PLC protocol driver version This item is blank except for GP Series nodes.This item is blank except for GP Series nodes.							
Special Note							

Function	Function Acquiring symbol/group byte size					
Acquires the total number of bytes required to access a device symbol or group symbol.						
INT WINAPI SizeOf	Symbol(LPCSTR sNodeName,LPCSTR sSymbolName,INT* oiByte	eSize);				
	me of entry node with Device/PLC name Name of target device or symbol name te size acquired	Return value Normal end: 0 Abnormal end: Error code				
Special Note For "sSymbolName", can be specified.	a device symbol, non-alignment group, whole alignment group, or ar	n element of alignment group				
Function	Acquiring number of group members					
Acquires the number	of members of a group or symbol sheet (total number of symbols and	d group members).				
INT WINAPI GetCo	untOfSymbolMember(LPCSTR sNodeName,LPCSTR sSymbolNam	e,INT* oiCountOfMember);				
Argument sNodeName: (In) Na sSymbolName: (In) N oiCountOfMember: (Return value Normal end: 0 Abnormal end: Error code					
	I exists in a specified group symbol, the number of members is count in the inner group symbol.	ed as one, even if multiple				
Function	Acquiring symbol/group/symbol sheet definition information					
Acquires definition in	nformation (data type, data quantity, etc.)					
	nbolInformation(LPCSTR sNodeName,LPCSTR sSymbolName,INT IMember,LPSTR osSymbolSheetName,SymbolInformation* oSymbolInformation* oSymbol					
sSymbolName: (In) N iMaxCountOfSymbo information + Specify the nur osSymbolSheetName symbol specifi oSymbolInformation structure. Prepare work f oiGotCountOfSymbo	me of entry node with Device/PLC name Name of symbol/group/symbol sheet IMember: (In) Specify a value of the maximum count of desired 1. mber of "oSymbolInformation" prepared. e: (Out) The API returns the name of symbol sheet that contains the ed with sSymbolName. Prepare 66 bytes or larger work. c: (Out) The API returns acquired detail information in the alignment for the number specified with iMaxCountOfSymbolMember. olMember: (Out) The API returns the information quantity that has mbolInformation.	Return value Normal end: 0 Abnormal end: Error code				

Special Note

• Structure of SymbolInformation

struct SymbolInformation

{

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

};

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

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

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

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

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

27.10 Precautions for Using APIs

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

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

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

(2) Data types available in special cases

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

About entry node name with Device/PLC name

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

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

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

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

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

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

About symbol searching precedence

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

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

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

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

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

Duplication of name

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

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

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

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

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

Duplication of global symbol name and local symbol name

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

Using Pro-Server EX API for multi-thread application

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

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

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

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

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

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

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

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

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

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

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

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

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

Improving cache buffer update efficiency

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

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

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

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

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

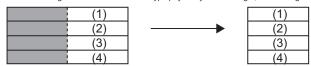
(5) When bit devices are registered in series, the processing speed becomes higher if they can be registered as word devices.

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

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

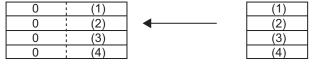
(1) When a 16-bit symbol is assigned to a device with a physically 32-bit width, and the device is accessed with the 16-bit symbol, or when 16-bit data type is directly specified to access a 32-bit device, 'Pro-Server EX' can handle the 32-bit device as a 16-bit device.

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



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

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



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

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

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

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

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

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

About Pro-Server auto start, forced closing and restart

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

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

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

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

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

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

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

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

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

About specification of symbol index

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

(Symbol name)[Value]

Example) Valve [2]

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

About queuing cache read and symbol cache read

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

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

About APIs that cannot be used for .NET

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

• Symbol access (Byte access)

ReadDevice(), ReadDeviceD(), WriteDevice(), WriteDeviceD()

ReadDeviceM(), ReadDeviceDM(), WriteDeviceM(), WriteDeviceDM()

ReadSymbol(), ReadSymbolD(), WriteSymbol(), WriteSymbolD()

ReadSymbolM(), ReadSymbolDM(), WriteSymbolDM()

Symbol size acquisition function

SizeOfSymbol()

About APIs that cannot be used in VB functions

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

ReadDeviceDATE_AND_TIME(), ReadDeviceDATE_AND_TIMEM(), ReadDeviceDATE_AND_TIMED(), ReadDeviceDATE_AND_TIMEDM(),

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

EasyStringToDATE_AND_TIME(), EasyDATE_AND_TIMEToString()

When using simple DLL in a multi-thread application

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

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

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

Message Process in Windows

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

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

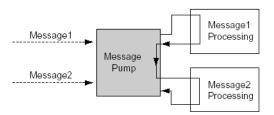
In this manual, the part which receives messages in order and branches into each process (corresponding to DoEvents for VB, or the part executing GetMessage() and DispatchMessage() for VC) is called the message pump. The message pump is not much recognized because it is hidden in the VC or VB framework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

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

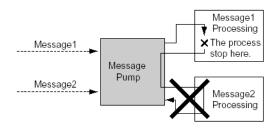
Example) Assume that messages are sent from Windows in the order of message 1 to message 2.

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

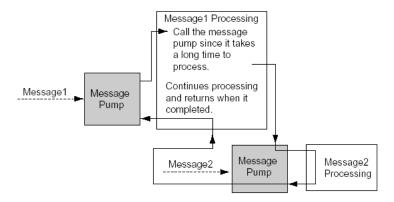
Then, when the message pump recovers from above, it takes out the following message (message 2) and calls the subroutine for message 2.



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



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



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

Prohibition of API Double Call

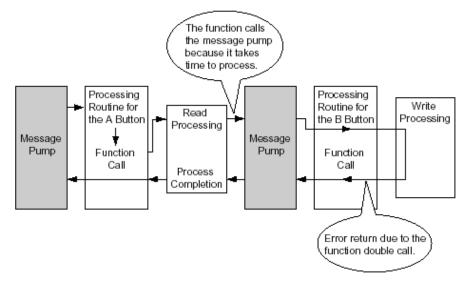
'Pro-Server EX API' prohibits another communication while communicating with a party (while calling a 'Pro-Server EX' function)(double-call). (Double-call is enabled if using the Multi-Handle. For details, refer to the section of Multi-Handle.) However, as 'Pro-Server EX API' runs the message pump inside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

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

Examples of double-call are shown below.

1. Double-call by pressing 2 buttons

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



2. Double-call by timer

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

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

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

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

Solutions to avoid API Double-Call

Solutions to avoid API double-call are shown below.

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

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

(3) Message should not be processed inside API

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

How to read character strings in VB

(1) Use ReadDeviceStr to read character strings in VB

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

```
Public Sub Sample1 ()
```

Dim strData As String * 10	' Correct designation method because it designates the size to read.
'Dim strData As String	' Incorrect designation method because it does not designate the character
	' string size.

Dim lErr As Long

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

If lErr ⇔ 0 Then

MsgBox "Read Error = " & lErr

Else

MsgBox "Read String = " & strData

End If

End Sub

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

```
Public Sub Smaple2 ()

Dim lErr As Long

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

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

lf lErr > 0 Then

MsgBox "Read Error = " & lErr

Else

MsgBox "Read String = " & vrData

End If

End Sub
```

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

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

```
Dim i As Integer

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

If 0 < i Then

TrimNull = Left (strData, i - 1)

Else

TrimNull = strData

End If

End Function
```

27.11 Using APIs (Examples)

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

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

Transformed Tenetion " "27.11.1 VB Support Function"

"27.11.2 VC Support Function"

"27.11.3 VB .NET Support Function"

"27.11.4 C# .NET Support Function"

27.11.1 VB Support Function

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

VB: Declaration statement

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

Show church ra	0						
饕 Pro-Studio E>	. Z.npx						<u> </u>
File Edit Tool	Programming Assist Set	ting Help					
Start .	VB & VBA 🕨 Decla	re Statement	Feature 🔉 📄		ио 🔪 🔥 т	ransfer	Monitor
Juan .	VC++ VC+ice	: Function		1 30	we 🚧 🙀 "		Status 😽
	EXCEL 🕨 Read	Function					
Symbol		Node Name AGP1		Device	Name PLC1		
Group	Ungroup	Sheet Name Sheet3		Set it as	a global symbol shee	et.	
Insert	Delete				5		
Сору	Cut Paste	Symbol	Data Type	Consec utive	Device Address	No. of Data	Comment
Symbol Sheet		_D0040_WORD	16Bit(Signed)		D0040	3	<u> </u>
Add	Delete					1	
						1	
Charle Durling	tion/List Used Addresses	1				1	
Check Duplica	IOUNTISCORED ADDI62262					1	
Global Con	Intent Calting Corport					1	

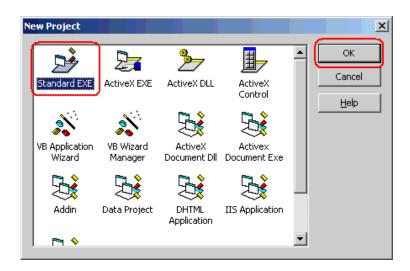
The VB declaration statement is coped to the clipboard.

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

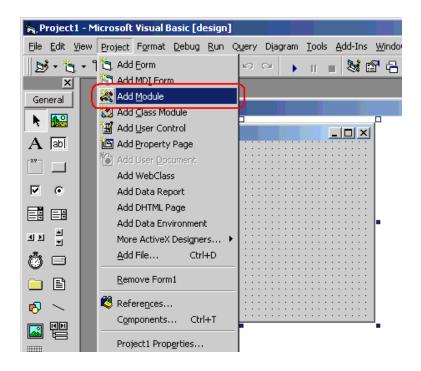
2 Start Microsoft Visual Basic, and select [New Project] from [File] on the menu.

licrosoft Visual	Basic					
e Edit View Pro	ject Forma	<u>D</u> ebug	<u>R</u> un	Q <u>u</u> ery	Djagram	<u>T</u> ools
<u>N</u> ew Project	Ctrl+N	Pa (2 /4	i 🗠	CH 🕨	П
Open Project	Ctrl+O					
A <u>d</u> d Project						
<u>R</u> emove Project						
Sa <u>v</u> e Project						
Sav <u>e</u> Project As.						
Save	Ctrl+S					
Save As						
Save Selection						
Save C <u>h</u> ange Scr	ript					
Brint	Ctrl+P					
Print Setyp						
Make,						
Make Project <u>G</u> ro	up					
E <u>x</u> it	Alt+Q					

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



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



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

Add Module	? ×
New Existing	1
	<u>O</u> pen 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	roject1 -	Microsoft	: Visual E	Basic [d	esign]				
Eile	<u>E</u> dit ⊻iev	v <u>P</u> roject	F <u>o</u> rmat	<u>D</u> ebug	<u>R</u> un	Query	Dįag	gram	<u>T</u> ools
	⊮⊃ Can't	Undo		Ctrl+Z	24	K)	Ω.	•	11 1
	™ <u>R</u> edo	Paste						_	_
Ger	X Cut			Ctrl+X	m	\			
k	E Copy			Ctrl+C		, (Code)	\ \		
_ (🛍 Paste	;		Ctrl+V		(COUE,		_	
	Paste	: Lin <u>k</u>							
	<u>R</u> emo)ve							
N	<u>D</u> elet	e		Del					
	Delet	e Table fro	m Data <u>b</u> a	ise					
	Selec	t <u>A</u> ll		Ctrl+A					
٩Þ	Selec	t <u>A</u> ll Colum	ns						
"R.	Table				•				

The deceleration statement is now pasted.

🚜 Project1 - Module1 (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-CARD 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

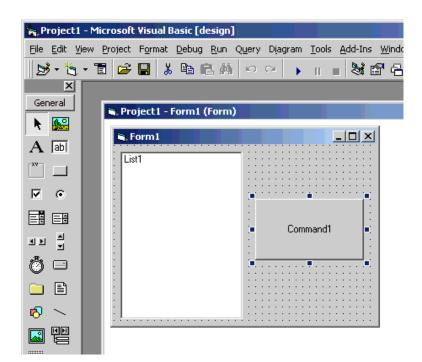
This section describes the procedure for creating an application that reads and displays data (16-bit signed data) for three points with a click on [Command1].

💐 Form1	_ 🗆 🗙
1111 2222 3333	Command1

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

<u>File E</u> dit <u>V</u> iew	Project Format Debug Run Query Diagram Tools Add-Ins Windo
🛛 😼 • 🏷 • 1	🗄 😅 🔒 🐍 🍋 📾 🛍 🗠 🗠 🕠 🔳 😹 🗃 名
×	
General	🗟 Project1 - Form1 (Form)
	🖼 Form1
A abl	List1
 ■ ■ ■	•
å 🗉	
N	· · · · · · · · · · · · · · · · · · ·

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	ig Help								
Start >> 🛐 Node >>	🍐 Symbol 🌺	Feature ン 📔	- 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	et.				
Copy Cut Paste	Symbol	Data Type	Consec	Device Address	No. of Data	Comment			
			utive	0.0040					
Symbol Sheet	_D0040_WORD	16Bit(Signed)		D0040	3	<u> </u>			
Add Delete				ļ	1				
			<u> </u>		1				
Check Duplication/List Used Addresses			<u> </u>		1				
			<u> </u>		1				
Global Constant Setting Screen			<u> </u>		1				
Ro-Server EX			<u> </u>		1				
🖻 🚚 PC1 (192.168.0.1)			<u> </u>	ļ	1				
#INTERNAL:Sheet1			<u> </u>		1				
GP3000 Series			<u> </u>		1				
È 💽 AGP1 (192.168.0.100)					1				
#INTERNAL:Sheet2		<u> </u>			1				
► WinGP					1				
► LT3000					1				
- • GP Series					1				
Global Symbol					1				
					1				
					1				
					1				
					1				

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

🎕 Pro-Studio EX	(2.прж				
File Edit Tool	Programming Assist	Setting	; Help		
Start .	VC++ •	Declare ! Write Fu Read Fu	nction	~	Feature ン [
Symbol			Node Nan	AGP1	
Group	Group Ungroup		Sheet Name Sheet3		
Insert	Insert Delete		01100(110)		
Сору	Cut Paste	,	Symbol		Data Type
Symbol Sheet			_D0040_W	ORD	16Bit(Signed)
Add	Delete		<u> </u>		
Check Duplical	tion/List Used Addres	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)						
Command			•	Click			•
Priva 1Err	te Sub Command1_ = ReadSymbolVari	Click() ant("AGP1.PLC1",	"_DOO4	0_WORD",Enter	a variant-typ	e variable	here)
End S	ιb						
							-
⋽ <u></u> ∃∎							• • · ·

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.

🖉 P	ject1 - Form1 (Code)	
Co	mand1 Click	•
	Private Sub Command1_Click() Dim wData As Variant	•
	<pre>LErr = ReadSymbolVariant("AGP1.PLC1","_D0040_WORD",Enter a variant-type variable here</pre>	
	Ind Sub	
		-
		▸

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

R P	Project1 - Form1 (Code)	
Co	Command1 Click	•
	Private Sub Command1_Click() Dim wData As Variant	_
	lErr = ReadSymbolVariant("AGP1.PLC1", "_D0040_WORD", wData)	
	End Sub	
3		▼ //

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				
<pre>lErr = ReadSymbolVariant("AGP1.PLC1",</pre>	"_DO)040_WORD", w)	Data)	
List1.AddItem CStr(wData(0)), 0 List1.AddItem CStr(wData(1)), 1 List1.AddItem CStr(wData(2)), 2				
End Sub				
				۲ ۲

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

🐂 Project 1	- Micro	soft Visual Basic [d	esign]				
<u>File E</u> dit <u>V</u> i	iew <u>P</u> roj	ject F <u>o</u> rmat <u>D</u> ebug	Run Query Diag	gram <u>Tools A</u> dd-Ios	Window <u>H</u> elp		
🛛 🔊 • 街	• 🗉	🖻 🖬 🕺 🖻 🖡	▶ <u>S</u> tart	F5	8 % 9 🔊	🖞 🛛 Ln 11, Col 1	
×			Start With <u>F</u> ul	Compile Ctrl+F5			
General				Ctrl+Break			
		Project1 - Form1 (🔲 End				
N		Form1	<u>R</u> estart	Shift+F5			
A abl			· · · · · · · · · · · · ·		ť		
		Project1 - Form1 ((Code)				
	E F	Command1					
		Designation of the	Common at C	1 1 - 7 5			
		Dim wData A	Command1_C S Variant	LICK()			
의 전 전 10 전 10 전 10 전 10 전 10 전 10 전 10 전		lErr = Read	SymbolVarian	nt("AGP1.PLC1",	"_DOO40_WORD"	, wData)	

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

🐂 Form1	
11111 2222 3333	Command1

Creating "Writing" application

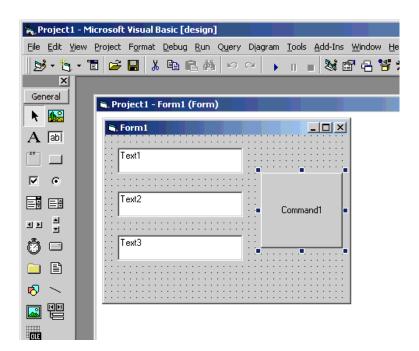
This section describes the procedure for creating an application that writes the data (16-bit signed data) entered for three points with a click on [Command1].

💐 Form1	
1010	[
2020	Command1
3030	

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

🐂 Project 1	l - Microsoft Visual Basic [design]
<u>File E</u> dit <u>V</u> i	<u>/</u> jew <u>P</u> roject F <u>o</u> rmat <u>D</u> ebug <u>R</u> un Q <u>u</u> ery Diagram <u>T</u> ools <u>A</u> dd-Ins <u>W</u> indow <u>H</u> el
🛛 🛃 • 街	・ 盲 😂 🖬 👗 陥 ඬ 桷 🗠 ⇔ 🕞 🗉 🗟 📽 🖯 臀 🌶
×	
General	🖲 Project1 - Form1 (Form)
	S. Form1
A abl	
	Text1
• •	
	:: Text2
ৰ চাৰ	
å 🗆	Text3
	·····

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



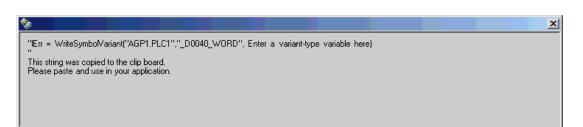
19 Select a target symbol name from those registered in 'Pro-Server EX'. (Select the symbol with first-address for writting.)

💱 Pro-Studio EX 2.npx						
File Edit Tool Programming Assist Settin	ng Help					
Start 🔉 🔪 Node 🔉	녿 Symbol ≫ 🍦	≷ Feature ン 📑	Sa	ive ᠉ 🄖 Ti	ransfer	Monitor Status
Symbol	Node Name AGP1		Device	Name PLC1		
Group Ungroup	Sheet Name Sheet3		Set it as	a global symbol shee	et.	
Insert Delete						
Copy Cut Paste	Symbol	Data Type	Consec utive	Device Address	No. of Data	Comment
Symbol Sheet	_D0040_WORD	16Bit(Signed)		D0040	3	
Add Delete					1	
					1	
Check Duplication/List Used Addresses					1	
					1	
Global Constant Setting Screen					1	
⊡ ▼ Pro-Server EX	l				1	
🖻 🐙 PC1 (192.168.0.1)					1	
INTERNAL:Sheet1			<u> </u>		1	
			<u> </u>		1	
			<u> </u>		1	
PLC1:Sheet3 A Series CPU I					1	
→ • WinGP					1	II
► LT3000			<u> </u>		1	i II
GP Series			<u> </u>		1	i
				,	1	
				,	1	
					1	
					1	
			-	-	-	

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

牧 P	ro-Stu	udio EX	(2.npx								
File	Edit	Tool	Programm	ing Assist	Settin	ig	Help				
	1	C11	VB & VB	A 🕨 🔟	Declare	St	tatement			.	ave ≫ 👔
	•	Start .	VC++	- (🗖	Write F	unc	ction	~ <	≷ Feature ン 🔓	1 30	we 🚧 🗓
			EXCEL		Read FI	unc	ction				
S	ymbol					Г	Node Name	AGP1		Device	Name PLC1
	G	iroup		Ungroup		L	Sheet Name	Sheet3		Sat it as	a global sym
	Insert Delete			L							
	Copy Cut Paste				Symbo	I	Data Type	Consec utive	Device Ac		
S	ymbol 9	Sheet -					_D0040_WOR	D	16Bit(Signed)		D0040
	,	Add		Delete							
C	Check (Duplica	tion/List Us	ed Addres:	ses	Ľ					
	Glo	bal Con	stant Settin	a Screen	1	lí					

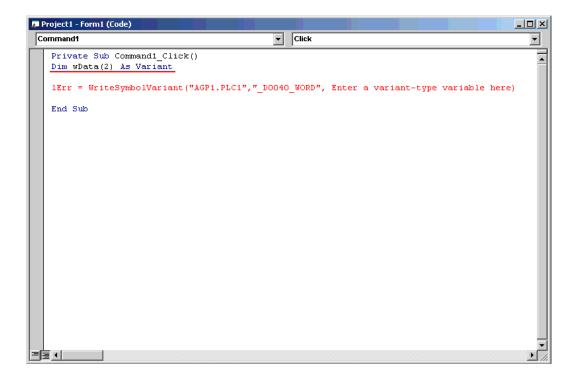
The write function is copied to the clipboard.



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

🛃 Pro	t1 - Form1 (Code)	<u> </u>
Соп	and1 Click	•
	ivate Sub Command1_Click() rr = WriteSymbolVariant("AGP1.PLC1","_DOO40_WORD", Enter a variant-type variable here)	^
	al Sub	
==		

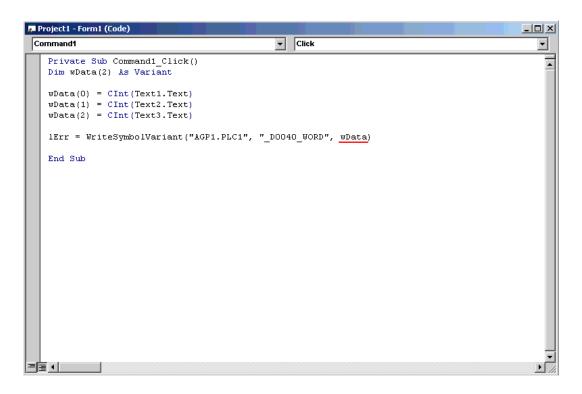
22 Declare the area (alignment) to store the written data. Ensure that the alignment type (in this example, Variant-type) is matched with the data type of the symbol being used.



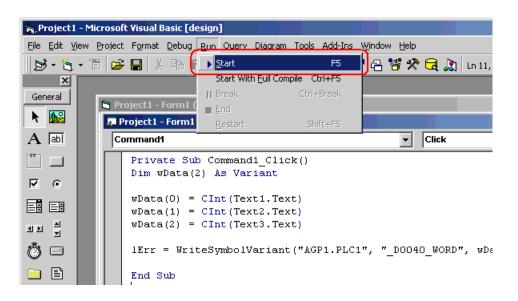
 $23\,$ Set the data entered in [TextBox] into the alignment.

/= Project1 - Form1 (Code)	_ 🗆 🗡
Command1 Click	-
Private Sub Command1_Click() Dim wData(2) &s Variant	-
<pre>wData(0) = CInt(Text1.Text) wData(1) = CInt(Text2.Text) wData(2) = CInt(Text3.Text)</pre>	
lErr = WriteSymbolVariant("AGP1.PLC1","_D0040_WORD", Enter a variant-type variab	le here)
End Sub	
	▼ ▶

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



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".

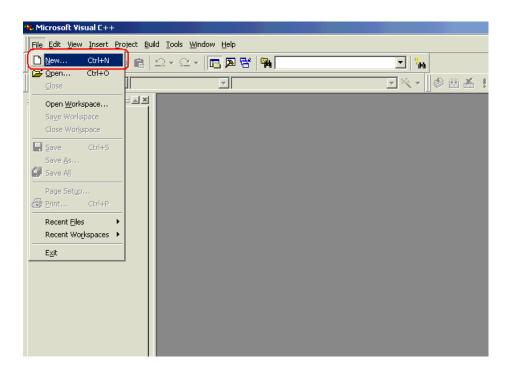
🖷 Form1	
1010 2020 3030	Command1

27.11.2 VC Support Function

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

VC: Declaration statement

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



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

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

New	<u>? ×</u>
Files Projects Workspaces Other Documents	
 ATL COM AppWizard Cluster Resource Type Wizard Custom AppWizard Database Project DevStudio Add-in Wizard Extended Stored Proc Wizard ISAPI Extension Wizard MAkefile MFC ActiveX ControlWizard MFC AppWizard (dll) MFC AppWizard (cxe) Utility Project Win32 Application Win32 Dynamic-Link Library Win32 Static Library 	Project <u>n</u> ame: Sample Logation: C:\PRDGRAM FILES\PRO-FAC Cigeate new workspace Add to current workspace Dependency of: Platforms: Win32
	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? Single document Multiple documents Dialog based Document/View architecture support?
	What Janguage would you like your resources in? English [United States] (APPWZENU.DLL 💌
< <u>B</u> ack	<u>N</u> ext >EinishCancel

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

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

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

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

%)	Sa	mple	- Mic	rosoft V	'isual C+	++				
Ei	le	<u>E</u> dit	⊻iew	Insert	<u>P</u> roject	<u>B</u> uild	<u>L</u> ayout	<u>T</u> ools	<u>W</u> indow	Help
				7 %		Acti <u>v</u> e To Pr	Project		*	
J	зa	mpleD	'Ig		Dep	ender	icies			SampleDlg
		_		(<u>S</u> ett	ings		A	lt+F7	PLE_DIALOG (Dialog)
E		1 0	ample	e resou	Exp	ort <u>M</u> a	kefile			
L		+	lcor	-	Inse	ert Pro	ject into '	Worksp	ace	······
			Strir	ng Table					то	DD: Place dialog controls here.

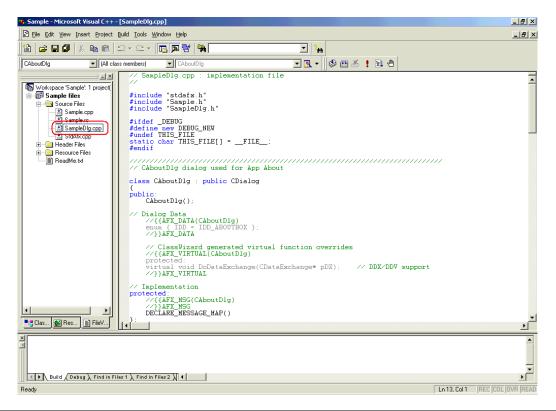
6 Specify a LIB file for [Object/library modules] in the [Link] tab. Then, click the [OK] button.

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.

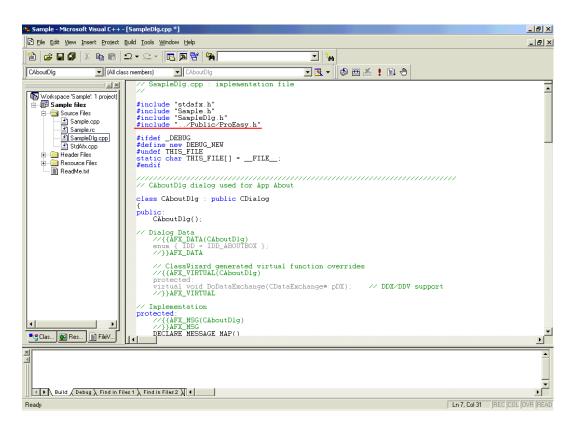
Project Settings	<u>?</u> ×
Settings For: Win32 Debug	General Debug C/C++ Link Resourc?
	OK Cancel

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 11111 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 V	/isual C++ - [Sa	mple.rc - IDD_SAMP	E_DIALOG (Dialog)]				_	₽×
Eile Edit View Inse	ert <u>P</u> roject <u>B</u> uild	Layout Tools Winde	w <u>H</u> elp				_	8×
12 🚅 🖬 🕼 👗	h 🔒 🕰	🖪 🗖 🗟	* 4	- *				
CSampleDlg	(All class me	embers) 💌 💊 I	SampleDlg	• 🗟 •	🕸 🖽 者 🚦 🗄 🕚			
					· · · · · · · · · · · · · · · · · · ·			
Sample resou	rces	Sample			×			
E IDD_SA	MPLE_DIA				OK			
🗄 🧰 String Table					Cancel	Con 🗵		
🗄 – 🧰 Version								
						$A\alpha$ ab		
						• •		
		TOD	D: Place dialog controls here.					
						(I)		
						♦ <u>■</u>		
						0- 🖙 		
						<u>ab</u>		
						S		
	FileV							
바뀌루	H H	ĭ = ¤ ⊕ #						
× -								-
								_
Build Debug	λ Find in Files 1	λ Find in Files 2 λ ◀						<u>۲</u>
Ready						50, 90	I [™] 200×8	READ

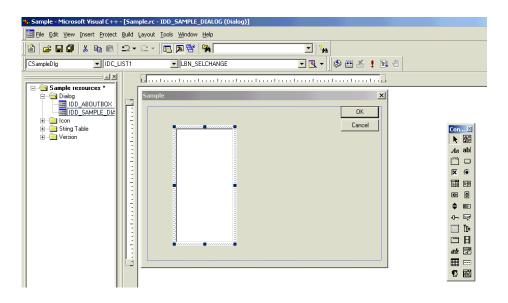
10 Select [Customize] from [Tools] on the Microsoft Visual C++ menu.

Sample - Microsoft Visual C++ - [Sample.	rc - IDD_SAMPLE_DIALOG (Dialog)]
🗐 File Edit Yiew Insert Project Build Layo	ut Tools Window Help
11 12 12 12 12 12 12 12 12 12 12 12 12 1	Close Source Browser File
CSampleDIg (All class members) 💦 Register Control
	📖 🥕 Error Lookup
Sample resources	ActiveX Control Test Container
E Dialog	Vi OLE/COM Object viewer
IDD_SAMPLE_DIA	Х SEX++ ОК
i⊞ icon	MFC Iracer Cancel
⊕ String Table	<u>Customize</u>
	Options
	Staro
	Record Quick Macro Ctrl+Shift+R
	Play Quick Macro Ctrl+Shift+P

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

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

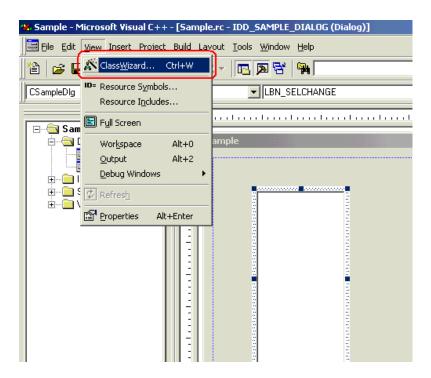
12 Select [ListBox], and paste it to the dialog box.



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

List Box P	roperties						×
-¤ ?	General	Styles	Extended Styles				
Sele <u>c</u> tio	n:						
Single		<u>B</u> order	Horizontal scroll		Want <u>k</u> eyi	input	
0wner d	traw: 🧲	Sort	✓ Vertical scroll		<u>D</u> isable no	scroll	
No	N	<u>N</u> otify	🔲 No <u>r</u> edraw	$\overline{}$	No integral	l <u>h</u> eighl	ŧ۴
∏ H <u>a</u> s	strings 🗖	<u>M</u> ulti-colu	mn 🔲 <u>U</u> se tabstops				

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



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

MFC ClassWizard	<u>? ×</u>
Message Maps Member Variables Automation ActiveX Events Class Info	
Project: Class <u>n</u> ame: Sample CS ampleDig	Add Class ▼
C:\\VC\Sample\SampleDig.h, C:\\VC\SampleDig.cpp <u>Control IDs:</u> <u>Type</u> <u>Member</u>	<u>D</u> elete Variable
IDC_LIST1 IDCANCEL IDOK	Update <u>C</u> olumns
	<u>B</u> ind All
Description:	
	Cancel

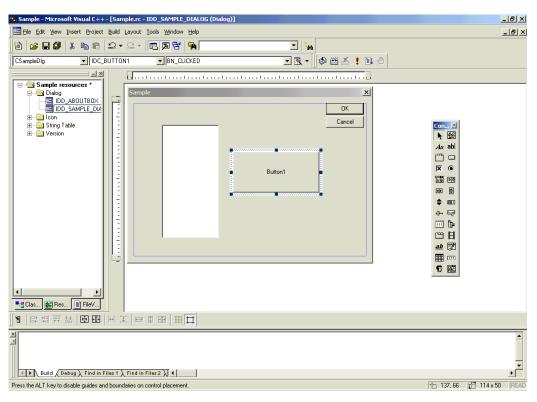
16 Click [Add Variable], and enter "m_List" for [Member variable name]. After selecting "Control" for [Category], click the [OK] button.

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

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

MFC ClassWizard					<u>? ×</u>
Message Maps	Member Variables	Automation	ActiveX Events	Class Info	
Project: Sample C:\VC\Sample\ Control_Ds: IDC_LIST1 IDCANCEL IDDK	⊾ SampleDlg.h, C:\WC	Class <u>p</u> am CSamplel Sample\Samp Type CListBox	Dlg		Add Class Add Variable Delete Variable Update Columns Bind All
Description: m	ap to CListBox member				
				OK	Cancel

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



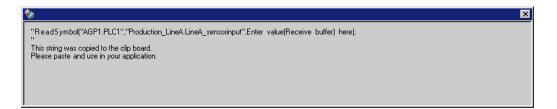
19 Select a target symbol name from those registered in 'Pro-Studio EX'. (Select the symbol with first-adress for reading.)

Pro-Studio EX 2.npx e Edit Tool Programming Assist Settin	g Help					
🔰 Start ン 🟹 Node ン	🌔 Symbol 🌺	≷ Feature ≫ [🚽 Sa	ive ᠉ 🄖 T	ransfer	Moni Stal
Symbol	Node Name AGP1		Device	Name PLC1		
Group Ungroup	Sheet Name Sheet	3	Set it as	a global symbol shee	et.	
Copy Cut Paste	Symbol	Data Type	Consec utive	Device Address	No. of Data	Commen
Symbol Sheet	_D0040_WORD	16Bit(Signed)		D0040	3	
Add Delete					1	
					1	
Check Duplication/List Used Addresses					1	
					1	
Global Constant Setting Screen					1	
⊡					1	
È-₩ PC1 (192.168.0.1)					1	
IIII #INTERNAL:Sheet1					1	
GP3000 Series			<u> </u>		1	
Ġ 🔂 AGP1 (192.168.0.100) 💢 #INTERNAL:Sheet2		_ <u></u>	<u> </u>		1	
PLC1:Sheet3 A Series CPU [<u> </u>		1	
→ WinGP			<u> </u>		1	<u> </u>
> LT3000		<u> </u>	<u> </u>		1	
F GP Series		<u> </u>	<u> </u>		1	
Global Symbol			<u> </u>		1	
					1	
					1	
		<u> </u>	<u> </u>		1	<u> </u>
					1	

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

_											
2	Pro-St	udio E	X 1	23.np:	ĸ						
Fi	le Edit	Tool	Pro	gramm	ing As:	sist	Set	ing	Help		
		C 1-1	1	VB & VE	A 🕨	١.]	1			No.
		Start	· - 1	/C++	•	C)ecla	re S	tatement	٢	
			E	EXCEL	•	6	Froup) str	ucturization	L	
Г	Symbol		-			\	Vrite	Fur	ction	AGI	P1
	Ē	quoi			Ungro	F	lead	Fun	ction 📐		_
		noup			Sugle	-	-		onset ivame	he	eet3
		nsert			Deleti	e					
	Сор	y I	C	Cut	P	aste			Symbol		Data 1
									0.10000	10	Dir(Cianad

The read function is copied to the clipboard.



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

Sample - Microsoft Visual C++ -	[Sample.rc - IDD_SAMPLE_DIALOG (Dialo	9)]		
Eile Edit View Insert Project E	<u>B</u> uild Layout <u>T</u> ools <u>W</u> indow <u>H</u> elp			
12 😂 🖬 🕼 🗼 🖻 🗄	요 • 요 • 🖪 🖪 🗟 🐂	✓ ³ 44		
CSampleDIg 💽 (All clas	ss members) 💽 💊 CS ampleD Ig		🖻 🛗 🛃 🚦 🕙	
AX		μ		
Sample resources *	Sample	Button1	OK Cancel	Con

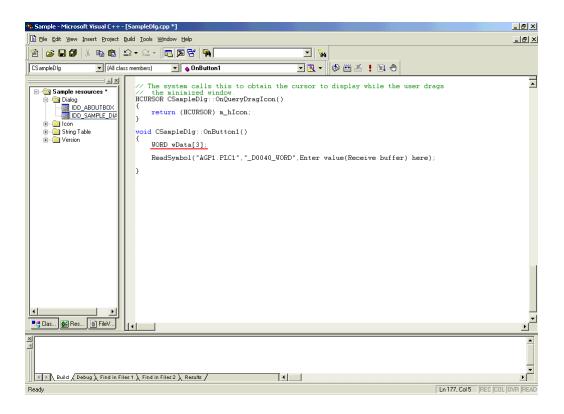
22 Click the [OK] button.

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

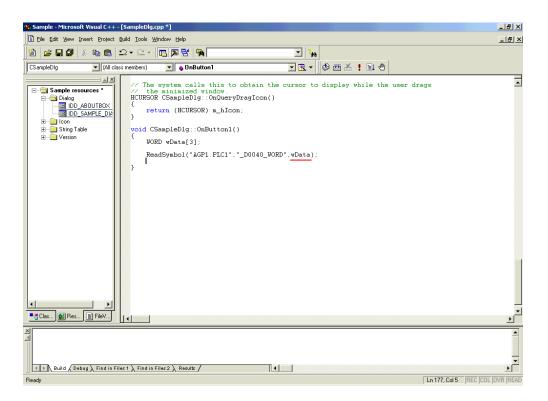
23 Paste the data on the clipboard (read function) into the OnButton1 member function.

2. Sample - Microsoft Visual C++ - [SampleDlg.cpp *]	8 ×
Ele Edit View Insert Project Build Iools Window Help	8×
CSampleDig 🔄 (All class members) 🔄 💊 OnButton1 💽 💘 🗸 🕼 🖄 🛃 🛃 🖑	
<pre>Sample resources Sample resources Dolog ABOUTBOX Dolog ABOUTBOX Constrained window. BOU ABOUTBOX Constrained window. HCURSOR CSampleDig::OnQueryDragIcon() Constrained window. HCURSOR CSampleDig::OnButton1() ReadSymbol("AGP1.FIC1"."_D0040_WORD".Enter value(Receive buffer) here); } </pre>	1
	•
Build (Debug) Find in Files 1 Find in Files 2 Results / Ready	•

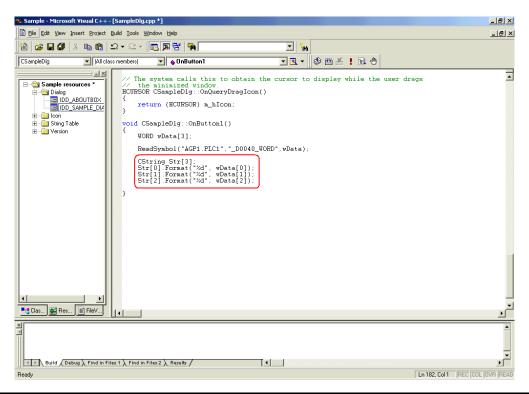
 $24 \ \ {\rm Declare \ the \ area \ (Array) \ to \ store \ the \ read \ data.}$



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.



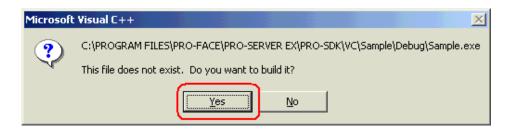
27 The list box (m_List) displays the read data (that has been converted into string data) in sequence.

. Sample - Microsoft Visual C++ - [SampleDig.cpp]	_ 8 ×
🗅 Elle Edit View Insert Project Build Iools Window Help	_ 8 ×
1) 🕼 🖬 🕼 🕲 💵 · · · · · · · · · · · · · · · · · ·	
CSampleDig 🔽 (Ali class members) 🔄 💊 OnButton 1 🔄 👿 🗸 🕴 🥸 📇 🚣 🚦 🕲	
<pre>Sample resources Didg Didg DidgABOUTBOX HCURSOR CsampleDig::OnDueryDragIcon() { return (HCURSOR) m_hIcon; } void CsampleDig::OnButton1() { WORD vData[3]; ReadSymbol("AGP1.FLC1", "_D0040_WORD", vData); CString Str[3]; Str[1].Format("Xd", vData[0]); Str[1].Format("Xd", vData[2]); m_List InsertString(0.Str[0]); m_List InsertString(2.Str[2]); } // 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 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 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 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 system calls this to obtain the cursor to display while the user drags // The system calls the cursor to display while the user drags // The system calls the cursor to display while the user drags // The system calls the cursor to display while the user drags // The system calls the cursor to display while the user drags // The system calls the cursor to display while the user drags // The system calls the cursor to display while the user drags // The system calls the cursor to display while the user drags // The system calls the cursor to display while the user drags // The system calls the cursor to display while the user drags // The system calls the cursor to display while the user drags // The system calls the cursor to display while the user drags // The system calls the cursor to display while the user drags // The system calls the cursor to display while the user drags // The system calls the cursor to display while the user drags // The system call the cursor to display while the user drags // The sys</pre>	4
	•
	<u> </u>
	1
$ \mathbb{A} \rightarrow \mathbb{A} $ Build $\langle Debug \rangle$ Find in Files 1 \rangle Find in Files 2 \rangle Results /	
Ready In 186, Col 1 REC COL	OVR READ

 $28 \hspace{0.1 cm} \text{Select [Execute Sample.exe] from [Build] on the Microsoft Visual C++ menu.}$

🦇 Sample - Microsoft Visual C++ ·	- [SampleDlg.cpp]
Eile Edit View Insert Project	Build Tools Window Help
CsampleDlg All cla Sample resources Dialog DD_ABOUTBOX DD_SAMPLE_DIA DO String Table D Version	<pre> Compile SampleDig.cpp Ctrl+F7 Build Sample.exe F7 Batch Build Clean Start Debug Debugger Remote Connection Execute Sample.exe Ctrl+F5 Cn1() Set Active Configuration Configurations Profile C1", "_D0040_WORI </pre>
	CString Str[3]; Str[0].Format("%d", wData[0]); Str[1].Format("%d", wData[1]); Str[2].Format("%d", wData[2]); m_List.InsertString(0,Str[0]); m_List.InsertString(1,Str[1]); m_List.InsertString(2,Str[2]); }

29 Click the [Yes] button.



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

Sample 11111 2222 3333	Button1	OK Cancel

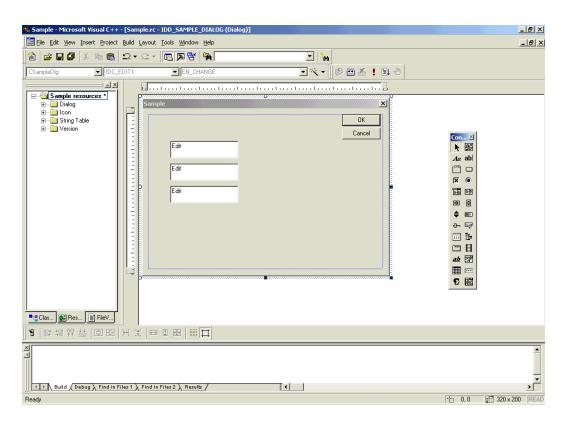
Creating "Writing" application

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

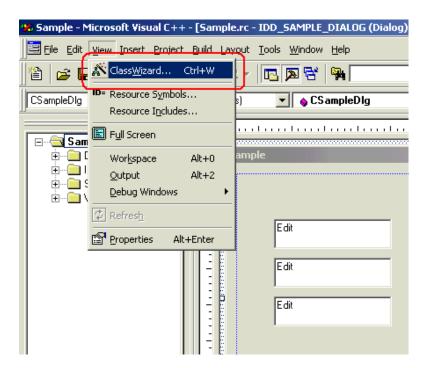
Steps 9 to 11 are the same as those for creating "Reading" application.

🛃 Sample		×
		OK
		Cancel
1010		
2020	Button1	
3030		

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



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



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

MFC ClassWiz	ard				? ×
Message M	aps Member Variables	Automation	ActiveX Events	Class Info	
Project:		Class <u>n</u> am	e:		Add Class 👻
Sample		CSamplel	-	<u> </u>	Add Variable
	mple\SampleDlg.h, C:\\V(1	
Control IDs:		Туре	Member		<u>D</u> elete Variable
IDC EDIT1					Update <u>C</u> olumns
IDC_EDIT3					Bind All
IDOK					
, Description:					
e ocompaiera					
				ОК	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	? ×
Member variable <u>n</u> ame:	ОК
m_Edit1	Cancel
Category:	
Value	
Variable type:	
short	
Description:	
short with range validation	

35 Click the [OK] button.

MFC ClassWizard					<u>? ×</u>
Message Maps	Member Variables	Automation	ActiveX Events	Class Info	
Project: Sample C:\\VC\Sample\ Control [Ds: IDC_EDIT1 IDC_EDIT2 IDC_EDIT3 IDCANCEL IDOK	SampleDlg.h, C:\WC	Class <u>n</u> am CSamplet Sample\Samp Type short short short	Dig	_	Add Qass Add Variable Delete Variable Update Columns Bind All
Description: sl <u>M</u> inimum Value: Ma <u>x</u> imum Value:	nort with range validatio	n -		Сок	Cancel

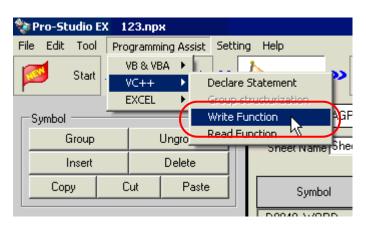
36 Select [Button], and paste it to [Dialog].

🎋 Sample - Microsoft Visual C++ - [Sample.rc - IDD_SAMPLE_DIALOG (Dialog)]		_ 8 ×
Elle Edit View Insert Project Build Layout Iools Window Help		_ 8 ×
) 🖆 😅 🖬 🕼 😂 × ♀ × 📧 🗖 😤 🦬 💽 💽 🐂		
CSampleDig 💽 IDC_BUTTON1 💽 BN_CLICKED 💽 🗟 🔻 🕼 🖑		
Sample resources Dis DD_ABOUTBOX DD SaMPLE DIE String Table B Version B Uton1 B Uton1 B Uton1 B Uton1 B Uton1	Con	
」3 ☞ 垌 趈 ট ᡂ 涎 実 ━ ㅍ 凾 ⅲ 茸		
Build (Debug) Find in Files 1) Find in Files 2		- - -
Press the ALT key to disable guides and boundaries on control placement.	+ 137,66 🗗 1	14 x 50 READ

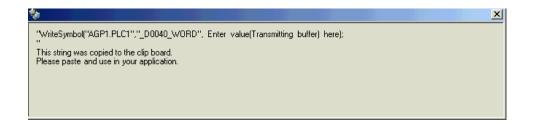
37 Select a target symbol name from those registered in 'Pro-Studio EX'. (Select the symbol with first-adress for writting.)

🂱 Pro-Studio EX 2.npx						_ 🗆 ×
File Edit Tool Programming Assist Sett	ing Help					
Start 😕 🟹 Node ン	🌔 Symbol 🌺 🏹	Feature ン 📑	Save 2	יד 💕 <	ansfer	Monitor Status
Symbol	Node Name AGP1		Device Name	e PLC1		
Group Ungroup	Sheet Name Sheet3		Set it as a glob	oal symbol shee	t.	
Insert Delete						
Copy Cut Paste	Symbol	Data Type	Consec Der utive Der	vice Address	No. of Data	Comment
Symbol Sheet	_D0040_WORD	16Bit(Signed)	D004	40	3	 _
Add Delete					1	
					1	
Check Duplication/List Used Addresses					1	
			<u> </u>		1	
Global Constant Setting Screen			<u> </u>		1	
			<u> </u>		1	
🖻 🛒 PC1 (192.168.0.1)					1	i
					1	
AGP1 (192.168.0.100)					1	
#INTERNAL:Sheet2					1	
PLC1:Sheet3 A Series CPU I					1	
LT3000					1	
GP Series					1	
Global Symbol					1	
	I				1	I
	I				1	
			<u> </u>		1	
	•	1	<u> </u>		<u> </u>	

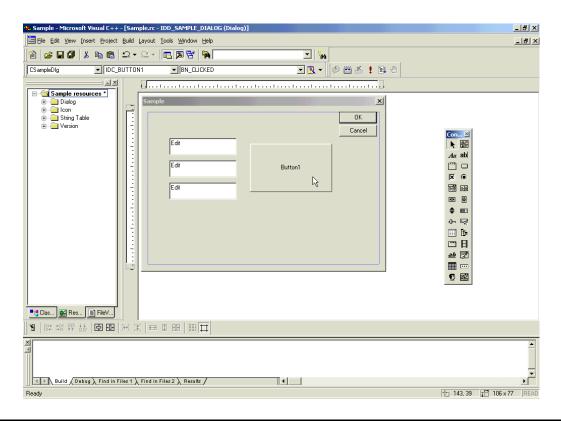
38 Select [Programming Assist] - [VC++] - [Write Function] on the menu.



The write function is copied to the clipboard.



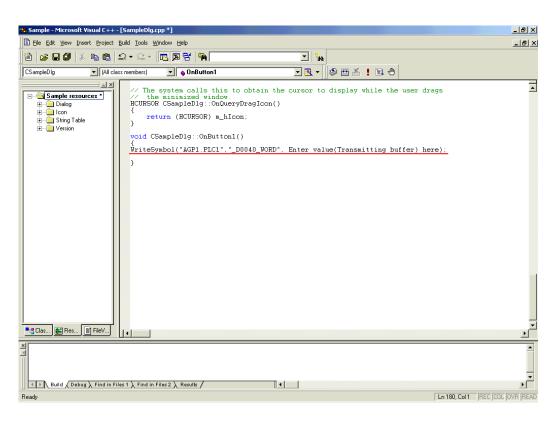
39 Double-click [Button1] that has been pasted to [Dialog] in Microsoft Visual C++.



40 Click the [OK] button.

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

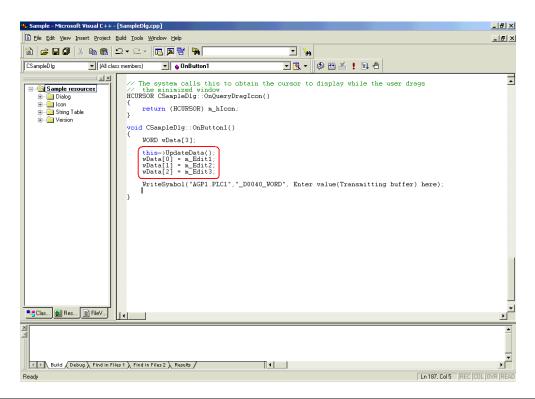
41 Paste the data on the clipboard (write function) into the OnButton1 member function.



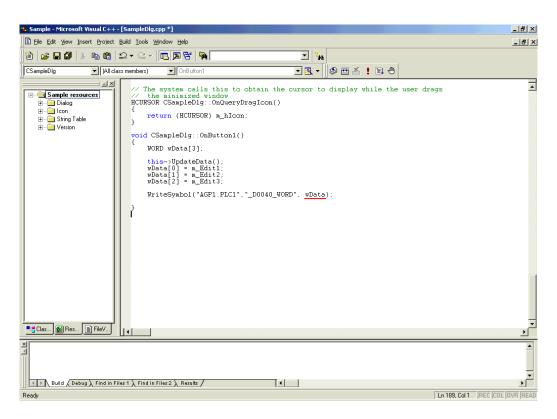
42 Declare the area (Array) to store the write data. For three or more writing points, specify three or more array elements.

🚧 Sample - Microsoft	¥isual C++ - [S	ampleDlg.cpp *]			_ 8 ×
Eile Edit View Ins	sert <u>P</u> roject <u>B</u> u	d <u>T</u> ools <u>W</u> indow <u>H</u> elp			X
12 😂 🖬 🕼 🕅	6 📭 🖻 🖆	• · · · [] 🖪 🕅 😽 🙀	- 1		
CSampleDig	 (All class) 	nembers) 👤 💊 OnButton1	- 🗟 - 🔇) 🖽 🛃 🖠 🕛	
Clas Res	e	}	con()	splay while the user drags	۲. ۲. ۲.
× •					<u> </u>
Build Debu	1g λ Find in Files	1 λ Find in Files 2 λ Results /			v F
Ready				Ln 182,	Col 1 REC COL OVR READ

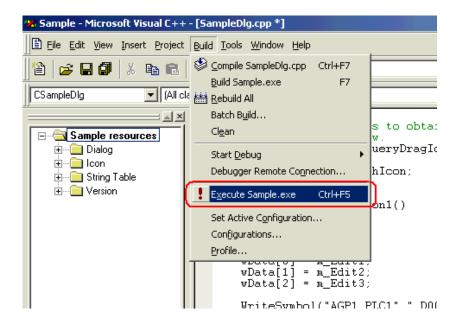
 $43 \ \, \text{Set the data entered in [Edit Box] (for three points) into the array.}$



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 Click the [Yes] button.

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

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

💑 Sample		×
		OK Cancel
1010		
2020	Button1	
3030		

27.11.3 VB .NET Support Function

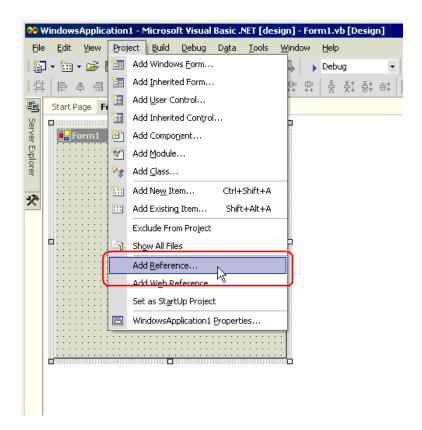
1 Start Microsoft Visual Studio .NET 2003 (or later version), and select [New] - [Project] from the [File] menu.

	<u>N</u> ew			۲	Projec	ct Ctrl+Sł	nift+N		- 🍻
	Open			ъ	Eile	W (Thrl+N	/	
	⊆lose				🖏 <u>B</u> lank	Solution	-		
	A <u>d</u> d Project			×	esources	My Pr	ofile	-	
3	Op <u>e</u> n Solution.								
à	Close Solu <u>t</u> ion								
	Save Selected :	Items	Ctrl+S						
	Save Selected	Items <u>A</u> s						Modified	
đ	Save Aļļ	Ctrl+S	hift+S						
	Sou <u>r</u> ce Control			۲					
D	Page Set <u>u</u> p								
5	<u>P</u> rint		Ctrl+P						
	E <u>x</u> it								

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	
Visual Basic Pro		VB.	11 VB	HVB -	
		Windows Application	Class Library	Windows Control Library	
visual studio s	Diutions	<u>r</u> ys			
		Smart Device Application	ASP.NET Web Application	ASP.NET Web Service	-
A project for creating a	an application with a Window	s user interface			
Name:	WindowsApplication1				
Location:	C:\Documents and Setting	s\Administrator\M	y Documen 💌	Browse	1
Project will be created a	at C:\\My Documents\Visu	al Studio Projects)	WindowsApplica	ation1.	
▼ 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		
7.0.3300.0			
	C:\WINNT\Microsoft.NET\Fra		
7.0.3300.0	C:\Program Files\Microsoft Vi		
1.0.5000.0			
1.0.5000.0			
		-	
7.0.5000.0	C:)Program Eiles)Microsoft Vi	<u> </u>	
vne	Source		Remove
//**	200.00		T Solito F S
	.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 0.0.5000.0 C:\WINNT\Microsoft.NET\Fra 0.0.5000.0 C:\WINNT\Microsoft.NET\Fra 0.0.5000.0 C:\WINNT\Microsoft.NET\Fra 0.0.00 C:\Program Files\Microsoft VI 0.0.00 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 C:\WINNT\Microsoft.NET\Fra 1.0.5000.0 C:\WINNT\Microsoft.NET\Fra 0.5000.0 C:\WINNT\Microsoft.NET\Fra 0.5000.0 C:\WINNT\Microsoft.NET\Fra 0.0.5000.0 C:\WINNT\Microsoft.NET\Fra 0.0.00 C:\WINNT\Microsoft.NET\Fra 0.0.00 C:\Program Files\Microsoft V

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

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

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

Select Compone	nt						×
Look in:	🛅 bin		•	🗢 🕶 🔁 🛛 🖉) 🗙 👛 🗉	🗄 👻 Tools 👻	
History	ProEasyDot	Net.dl					
My Projects							
Desktop							
* Favorites							
My Network Places	 File <u>n</u> ame: Files of <u>t</u> ype:	Component Files	(*.dll;*.tlb;*.	olb;*.ocx;*.ex	=)	; (──	Open Cancel

6 Click the [OK] button.

		Path ▲	
Component Name	Version		Select
Accessibility.dll	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra	Select
adodb	7.0.3300.0	C:\Program Files\Microsoft.N	
CustomMarshalers	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra	
envdte	7.0.3300.0	C:\WINNT\Microsoft.NET\Fra	
extensibility	7.0.3300.0	C:\Program Files\Microsoft Vi	
IEExecRemote	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra	
IEHost	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra	
IIEHost	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra	
ISymWrapper	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra	
Managed C# Compiler	7.0.5000.0	C:\WINNT\Microsoft.NET\Fra	
MCppCodeDomParser	0.0.0.0	C:\Program Files\Microsoft Vi	
MCnnCodeDomProvider	7.0.5000.0	C:\Program Files\Microsoft Vi	
ected Components:			7
omponent Name	Туре	Source	Remove
oEasyDotNet.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.

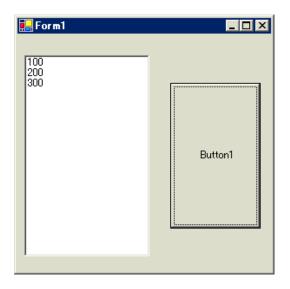
The following procedure varies depending on whether the application is intended for reading or writing, and so is explained individually.

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].

	Windou	usānnlie	ation1 - I	Microso	ft Visua	Basic	NET [de	sian] -	For	m1 v	h [De	sian]*				
File			Project	Build	Debug	Data	Tools	Wind		Help	0 [04	.sign]				
	-	-		-		_	_	_								
1	μ - <u>hi</u>] 🕶 📂		X 🖻		C) + C)	• • • •	•	•	Debu			- 12	9		
ĮΨ	中 臣	후 릐	<u>미</u> -아			10 日	에야 산	+c D]0 → →+	₽ **	붐	볶	응* 6;		후	.	
5	Toolbo	x		4 ×	Star	Page I	Form1.v	b [Des	sign)	*						
ŝ	Data															
Server Explorer	Compo					Form1								<u><</u>		
¥	Windo	ws Forms	:													
lore	🕨 P	ointer				ListBox1		88888888	.	::::	:::		::::	1		
	\mathbf{A} L	abel												-		
	🔥 🚹 Li	nkLabel							•••	::::	:::	:::::	::::	:		
	ab B	utton							•••					-		
	abl T	extBox							•••							
	📑 M	lainMenu												:		
		heckBox														
	⊙ R	adioButto	on						•••	::::	:::	:::::	::::	:		
	[^{xv}] G	roupBox												-		
	R P	ictureBox	(•••	::::	:::	:::::	::::	:		
	P	anel							•••					:		
	 	ataGrid				1+										
	E E L	stBox			: [_]				ш. 		:::		::::	:		
		heckedLi:	stBox		<u> </u>											
	(Frank)	omboBo×														
		stView														
		reeView														
		abContro	4													
	'	aucontro	л													

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

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

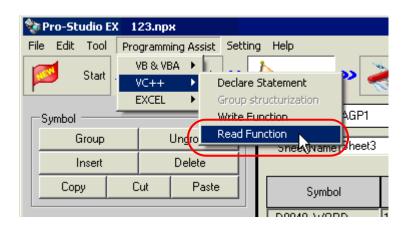
	Wedeweterliest Missee	ft Visual Basic .NET [design] - Form1.vb [Design]*
Eil		Debug Data Format Tools Window Help
1	🛛 • 🛅 • 🚅 🖬 🕼 👗 🖻	💼 🗠 🕶 🖉 🗉 🖳 🕨 Debug 🔹 🍅
Ē	후 특 후 릐 ㅠ ~ 프 5	글 尊 前 원 ┉ ಭ 밖 밖 움 찾 탉 타 팬 혼 ┗ ┖ ┖ .
5	Toolbox P ×	Start Page Form1.vb [Design]*
	Data	
rver	Components	Form1
¥	Windows Forms	
Server Explorer	Pointer	ListBox1
~	${f A}$ Label	
	A LinkLabel	
	ab Button	
	abl TextBox	
	🛓 MainMenu	Button1
	CheckBox	
	RadioButton	
	GroupBox	
	😹 PictureBox	
	Panel	
	🚰 DataGrid	
	∎∉ ListBox	
	📰 CheckedListBox	
	E ComboBox	
	222 ListView	
	्न_ इन्_ TreeView	
	TabControl	

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

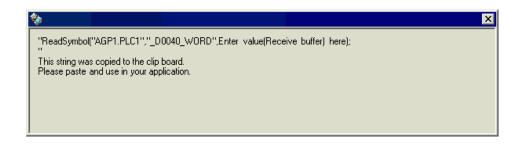
💱 Pro-Studio EX 2.npx							
File Edit Tool Programming Assist Settin	g Help						
Start >> 🟹 Node >>	🔶 Symbol ⋗ 🚄	Feature ン 📑	Sa Sa	ave > 🆄 T	ransfer	Monitor Status	
Symbol	Node Name AGP1		Device	Name PLC1			
Group Ungroup	Sheet Name Sheet3		Set it as	a global symbol shee	ət.		
Insert Delete							
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					1		
⊡▼ Pro-Server EX	I				1		
🖻 🐙 PC1 (192.168.0.1)			<u> </u>		1		
					1		
E▼ GP3000 Series E GP AGP1 (192.168.0.100)			<u> </u>		1	II	
#INTERNAL:Sheet2				, 	1		
PLC1:Sheet3 A Series CPU [1		
→ WinGP LT3000					1		
GP Series					1		
Global Symbol					1		
					1		
					1		
					1		
					1	L . 🔳	

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

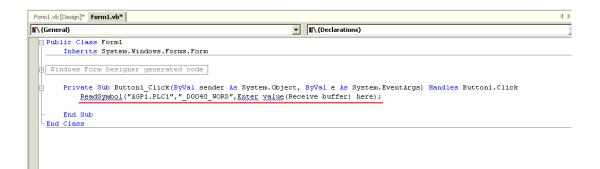
10 Select [VC++] - [Read Function] from the [Programming Assist] menu.



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.



$12 \ {\rm Import \ the \ ProEasyDotNet \ library}.$

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

ws.Forms.Form
verated code
tion2
ttoni Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
("AGP1.PLC1"," D0040 WORD",Enter yalue(Receive buffer) here);
itt

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.vb [Design]* Form1.vb*	4 Þ					
🖍 (General)	(Declarations)					
Imports ProEasyDotNet						
Public Class Form1						
Inherits System.Windows.Forms.Form						
🕀 Windows Form Designer generated code						
Private Sub Button1_Click(ByVal sender & System.Object, ByVal e & System.Event&rgs) Handles Button1.Click						
Dim wData(3) As Short						
ReadSymbol("AGP1.PLC1","_D0040_WORD",Enter value(Receive buffer) here);					
- End Sub						
- End Class						

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

Form1.vb [Design]* Form1.vb [Design]*	form1.vb*	1 Þ
🛷 Button1	Ø Click	
Imports ProE	EasyDotNet	_
🖃 Public Class		
Inherits	s System. Windows. Forms. Form	
Uindows For	rm Designer generated code	
Private	Sub Button1 Click(ByVal sender & System.Object, ByVal e & System.Event&rgs) Handles Button1.Click	
	wData(3) As Short	
ProE	Easy.ReadSymbol("AGP1.PLC1","_D0040_WORD",Enter value(Receive buffer) here);	
	QueuingExec	
- End Sub	- Voodangstand	
End Class	2 QueuingStatusCard	
	OueuingTblCell	
	We ReadDevice16	
	ReadDevice32	
	ReadDevice32D	

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

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

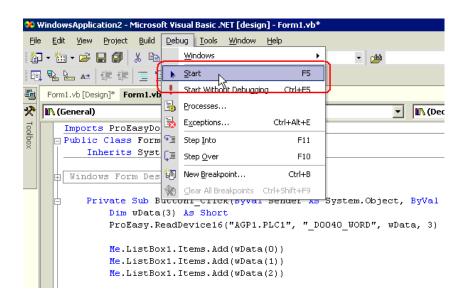
16 Specify a data storing area "wData" as the third argument. Enter ", " (comma) at the end of the third argument, and then enter "3" to specify the length of the target symbol as the fourth argument. After that, delete ";" (semicolon) at the end of the line.

Form1.vb [Design]* Form1.vb*		4 1
🖍 (General)	•	In (Declarations)
Imports ProEasyDotNet		
Public Class Form1		
Inherits System.Windows.Forms.Form		
😰 Windows Form Designer generated code		
 Private Sub Button1_Click(ByVal sender As System.Object Dim wData(3) As Short ProEasy.ReadDevice16("AGP1.PLC1", "_D0040_WORD", wData End Sub 		
-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*		1 b
🕼 (General)	-	N (Declarations)
Imports ProEasyDotNet		
Public Class Form1		
Inherits System.Windows.Forms.Form		
B Windows Form Designer generated code		
Private Sub Button1_Click(ByVal sender As Syste	em.Object,	ByVal e As System.EventArgs) Handles Button1.Click
Dim wData(3) As Short		
ProEasy.ReadDevice16("AGP1.PLC1", "_D0040_W	JORD", wDat	a, 3)
<pre>Me.ListBox1.Items.Add(uData(0)) Me.ListBox1.Items.Add(uData(1)) Me.ListBox1.Items.Add(uData(2))</pre>		
- 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	
1000	<u>.</u>

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 멷 훅 릐 ㅠ ~ 典	🛱 寺 印 昭 ┉ 짣 약 약 부 응 찾 왕 학 凾 色 📲 🖷 🖕	
Toolbox 7 ×	Start Page Form1.vb [Design]*	
Data		
Components	🖳 Form1 📃 🗖 🗙	
Windows Forms	· · · · · · · · · · · · · · · · · · ·	
Pointer		
A Label	TextBox1	
A LinkLabel	······································	
ab Button	TextBox2	
abl TextBox	· · · · · · · · · · · · · · · · · · ·	
🛓 MainMenu	TextBox3	
CheckBox		
RadioButton		
GroupBox		
RictureBox		
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].

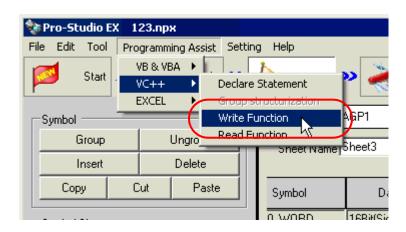
🥺 WindowsApplication3 - Micro	osoft ¥isual Basic .NET [design] - Form1.vb [Design]*	
<u>File E</u> dit <u>V</u> iew <u>P</u> roject <u>B</u> u	ld <u>D</u> ebug D <u>a</u> ta <u>T</u> ools <u>W</u> indow <u>H</u> elp	
🎦 • 🛅 • 🛩 🖬 🕼 🐰	🛍 💼 🗠 - 🗠 - 🚚 - 🌉 - Debug - 🍅	•
世 윤 후 릐 〒 숀 些	🛱 拱 印 昭 ┉ 깍 약 약 명 울 찾 삵 랴 凾 욘 ங 🐃 🗸	
Toolbox 🛛 🕈 🗙	Start Page Form1.vb [Design]*	
Data		
Components	Form1	
Windows Forms	······································	
Pointer	· · · · · · · · · · · · · · · · · · ·	
${f A}$ Label	TextBox1	
$\underline{\mathbf{A}}$ LinkLabel		
ab Button	TextBox2	
abl TextBox		
🛓 MainMenu	TextBox3	
🔽 CheckBox		
RadioButton		
GroupBox		
💦 PictureBox		
Panel		
词 DataGrid		
E# ListBox		

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

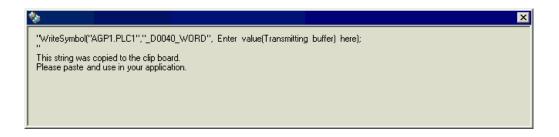
💱 Pro-Studio EX 2.npx					_ 🗆 🗵
File Edit Tool Programming Assist Setti	ng Help				
Start 🔉 🔪 Node 🔉	녿 Symbol ≫ 🍦	Feature ≫ [📔 Save ≫ 🆄	Transfer	Monitor Status
Symbol	Node Name AGP1		Device Name PLC1		
Group Ungroup	Sheet Name Sheet3		Set it as a global symbol sh	eet.	
Copy Cut Paste	Symbol	Data Type	Consec Device Address	No. of Data	Comment
Control Character	D0040 WORD	16Bit(Signed)	D0040	3	
Symbol Sheet				1	
				1	
Check Duplication/List Used Addresses				1	
				1	
Global Constant Setting Screen				1	<u> </u>
Pro-Server EX				1	<u> </u>
📄 🐙 PC1 (192.168.0.1)				1	
#INTERNAL:Sheet1	I	.		1	
				1	
HINTERNAL:Sheet2				1	
PLC1:Sheet3 A Series CPU [1	II
• WinGP				1	II
→ LT3000 → GP Series		·	<u> </u>	1	I
Global Symbol				1	i II
				1	
				1	
				1	
				1	

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

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



The write function is copied to the clipboard.



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

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

25 Import the ProEasyDotNet library.

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

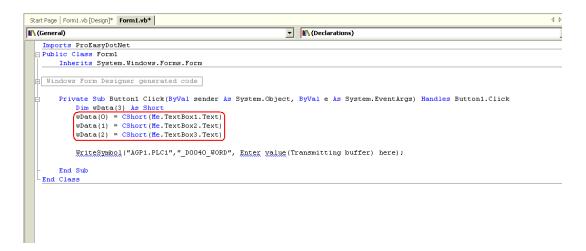
Start Page Form1.vb [Design]* Form1.vt	4 b
🖍 (General)	IN (Declarations)
U Processy OotNet U III U System U WindowsApplication3 Private Sub Button1_C UriteSymbol ("AGP1	ws.Forms.Form perated code lick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click .PLC1", "_D0040_WORD", <u>Enter value</u> (Transmitting buffer) here);
- End Sub End Class	

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

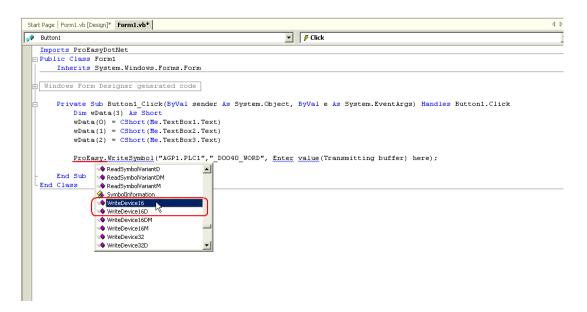
The array type ("Short" in this example) must conform to the data type of the target symbol. Specify the same data length as the target symbol ("3" in this example).

Start Page Form1.vb [Design]*	Form1.vb*	٩ ۵
🖍 (General)	•	🕼 (Declarations)
Imports ProEasyDo	tNet	
🖃 Public Class Form	1	
Inherits Syst	em.Windows.Forms.Form	
Private Sub B Dim wData	<pre>signer generated code] Button1_Click(ByVal sender As System.Object, a(3) As Short col("AGP1.PLC1","_D0040_WORD", Enter value(T</pre>	ByVal e As System.EventArgs) Handles Button1.Click
- End Sub		
End Class		

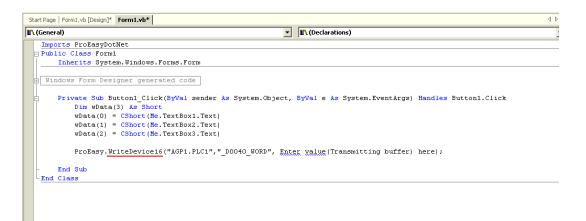
27 Set the data to be entered in [TextBox1] to [TextBox3] in the array.



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



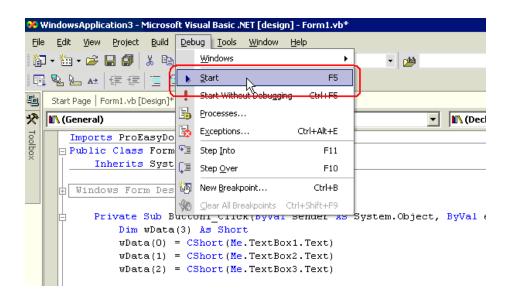
29 Delete "WriteSymbol" from the character string (write function) that has been pasted from the clipboard.



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

(General)	In (Declarations)	
Imports ProEasyDotNet		
- Public Class Form1		
Inherits System.Windows.Forms.Form		
⊕ Windows Form Designer generated code		
Private Sub Button1_Click(ByVal sender A Dim wData(3) As Short	As System.Object, ByVal e As System.EventArgs) Handles Button1.Click	
wData(0) = CShort(Me.TextBox1.Text)		
wData(1) = CShort(Me.TextBox2.Text)		
<pre>wData(2) = CShort(Me.TextBox3.Text)</pre>		
ProEasy.WriteDevice16("AGP1.PLC1", "	"_D0040_WORD", wData, 3)	
- End Sub		
- End Class		

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	

27.11.4 C# .NET Support Function

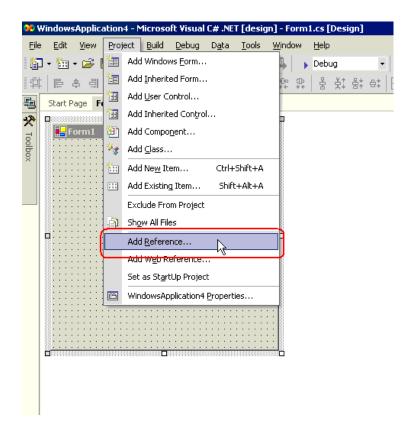
1 Start Microsoft Visual Studio .NET 2003 (or later version), and select [New] - [Project] from the [File] menu.

	New				Ctrl+Shift+N		- 🍅
	Open		N	Eile	r% Ctrl+N	-	
	⊆lose			😺 🛚 <u>B</u> lank So	olution		
	A <u>d</u> d Project		۲	esources	M <u>y</u> Profile		
5	Op <u>e</u> n Solution.						
È.	Close Solution						
1	Save Selected)	items Ctrl+S					
	Save Selected)	(tems <u>A</u> s				Modified	
1	Save Aļ	Ctrl+Shift+S				, iounco	
	Sou <u>r</u> ce Control		۲				
]	Page Setyp						
þ	<u>P</u> rint	Ctrl+P					
	E <u>x</u> it						

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

New Project					×
Project Types:		Templates:		00	5-5- 5-5- 5-5-
Visual Basic Proje	7		11 #	*	-
Setup and Dep Other Projects Visual Studio S		Windows Application	Class Library	Windows Control Library	
	olucions	1		ø	
		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'	\Administrator\M	y Documen 💌	Browse	
Project will be created at C:\\My Documents\Visual Studio Projects\WindowsApplication1.					
▼ Mor <u>e</u>		ок	Cancel	Help	

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



4 Click the [Browse] button.

Component Name	Version	Path		
Accessibility.dll	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra		Select
adodb	7.0.3300.0	C:\Program Files\Microsoft.N		
CustomMarshalers	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra		
envdte	7.0.3300.0	C:\WINNT\Microsoft.NET\Fra		
extensibility	7.0.3300.0	C:\Program Files\Microsoft Vi		
IEExecRemote	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra		
IEHost	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra		
IIEHost	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra		
ISymWrapper	1.0.5000.0	C:\WINNT\Microsoft.NET\Fra		
Managed C# Compiler	7.0.5000.0	C:\WINNT\Microsoft.NET\Fra		
MCppCodeDomParser	0.0.0.0	C:\Program Files\Microsoft Vi		
MCnnCodeDomProvider	7.0.5000.0	C:\Program Files\Microsoft Vi	<u> </u>	
ected Components:				
omponent Name	Туре	Source		Remove
	1.764			

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

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

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

Sele	ect Compone	nt		×
	Look in:	🛅 bin	💽 🗢 • 🗈 🔍 🔍	🌱 🎫 🕶 Too <u>l</u> s 🕶
	I istory	ProEasyDotNet.dll		
ſ	My Projects			
	Desktop			
	* Favorites			
r	My Network Places	File name: Files of type: Component Files (*	.dll;*.tlb;*.olb;*.ocx;*.exe)	Open Cancel

6 Click the [OK] button.

Component Name	Version	Path 🔺	
Accessibility.dll	1.0.5000.0		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	
	7.0.5000.0	C:)Program Eiles)Microsoft Vi	
ected Components:			
Component Name	Туре	Source R	emove
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.

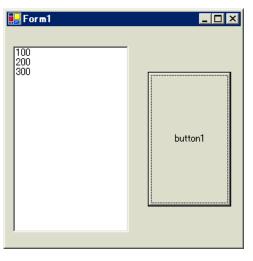
The following procedure varies depending on whether the application is intended for reading or writing, and so is explained individually.

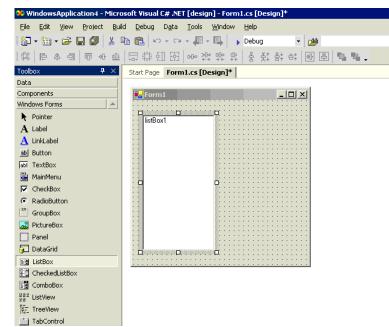
To create a "Reading" application, refer to steps 7 to 19.

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].

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

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

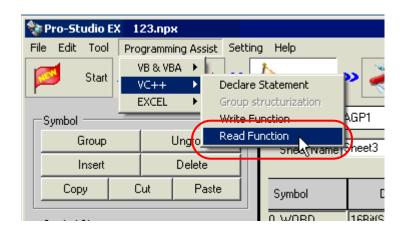
💥 WindowsApplication4 - Micro	osoft Visual C# .NET [design] - Form1.cs [Design]*
<u>File Edit View Project Bui</u>	
	🛍 🛍 🗠 - 🖓 - 🜉 🖡 Debug 🔹 🍅
박 [은 후 의] 때 아 쓰	品蒜舀図 ∞ 拌拌 st st st st ━ ■ ■ ■ ■
Toolbox $\Psi \times$	Start Page Form1.cs [Design]*
Data	
Components	Form1
Windows Forms	
k Pointer	listBox1
A Label	
<u>A</u> LinkLabel	
ab Button	
abl TextBox	
🛓 MainMenu	··· button1
CheckBox	
RadioButton	
GroupBox	
RictureBox	
Panel	
🚰 DataGrid	
≡∉ ListBox	
🚉 CheckedListBox	
ComboBox	
222 ListView	
🚛 TreeView	
📺 TabControl	

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

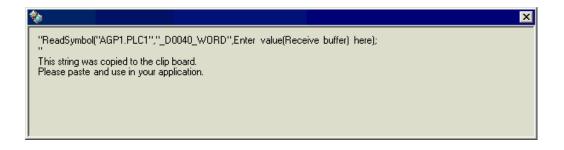
🂱 Pro-Studio EX 2.npx						_ 🗆 ×
File Edit Tool Programming Assist Settin	ng Help					
Start 🌺 🔪 🔪 Node ≫	녿 Symbol ≫ 🍦	Feature ⋗ 📑	Save 🔉	Tr	ansfer	Monitor Status
Symbol	Node Name AGP1		Device Name F	PLC1		
Group Ungroup	Sheet Name Sheet3		Set it as a global	symbol shee	t.	
Copy Cut Paste	Symbol	Data Type	Consec Devic	e Address	No. of Data	Comment
Symbol Sheet		16Bit(Signed)	D0040		3	
Add Delete					1	
					1	
Check Duplication/List Used Addresses					1	
· · · · · · · · · · · · · · · · · · ·					1	
Global Constant Setting Screen					1	
⊡ ▼ Pro-Server EX	l				1	
🖻 🚚 PC1 (192.168.0.1)			<u> </u>		1	
#INTERNAL:Sheet1					1	·
					1	
#INTERNAL:Sheet2					1	
PLC1:Sheet3 A Series CPU [·			1	
→ WinGP LT3000					1	
GP Series					1	
Global Symbol					1	
					1	
					1	
					1	
					1	

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

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



The read function is copied to the clipboard.

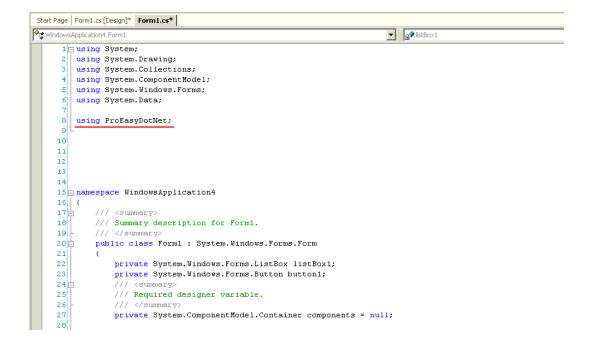


11 Double-click [button1] in [Form1], and paste the clipboard data (read function) below the [button1_Click] method ("private void button1_Click..." character string).

```
Start Page | Form1.cs [Design]* Form1.cs*
WindowsApplication4.Form1
                                                                          System.EventArgs e)
            public class Form1 : System.Windows.Forms.Form
     14
             ł
     15
                 private System.Windows.Forms.ListBox listBox1;
     16
                private System.Windows.Forms.Button button1;
     17 🛓
                 /// <summarv>
                /// Required designer variable.
     18
                /// </summary>
     19
     20
                private System.ComponentModel.Container components = null;
    21
    22 🕂
                public Form1()...
    34
                /// <summary>
                /// Clean up any resources being used.
    35
     36
                /// </summary>
     37 🗄
                 protected override void Dispose( bool disposing )...
     49 🕁
                Windows Form Designer generated code
    88
    89白
                /// <summary>
     90
                /// The main entry point for the application.
     91
                /// </summary>
    92
                [STAThread]
    93
                static void Main()
    94
                {
     95
                     Application.Run(new Form1());
     96
                }
     97
    98년
                 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
             3
    104 }
    105
    106
    107
    108
    109
```

12 Describe the ProEasyDotNet directive.

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

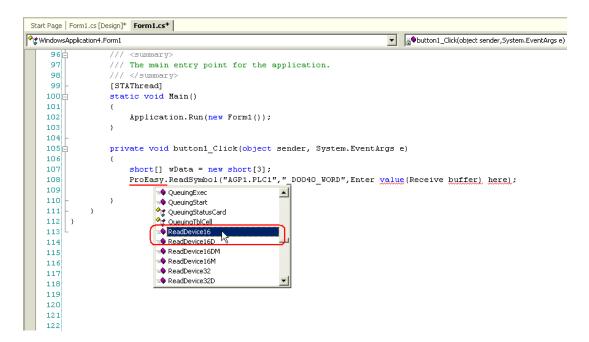


13 For the read data storing area, declare a variable "wData".

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

```
Start Page | Form1.cs [Design]* Form1.cs*
Ag WindowsApplication4.Form1
                                                                                ▼ ∰$button1_Click(object sender,System.EventArgs e)
     96位
                  /// <summary>
     97
                 /// The main entry point for the application.
     98
                  /// </summary>
     99
                 [STAThread]
    100白
                  static void Main()
    101
                  {
    102
                       Application.Run(new Form1());
    103
                  }
    104
    105
                  private void button1_Click(object sender, System.EventArgs e)
    106
                  {
    107
                      short[] wData = new short[3];
    108
                       ReadSymbol("AGP1.PLC1","_D0040_WORD",Enter value(Receive buffer) here);
    109
    110
                  }
    111
              }
    112 }
    113
    114
    115
    116
    117
    118
```

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



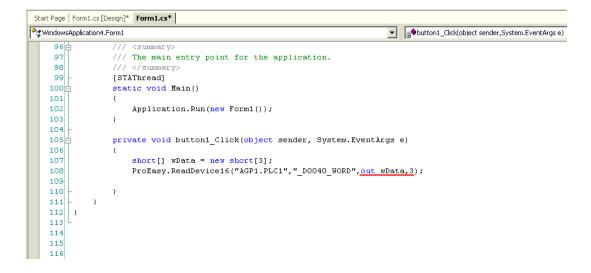
15 Delete "ReadSymbol" from the character string (read function) that has been pasted from the clipboard.

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

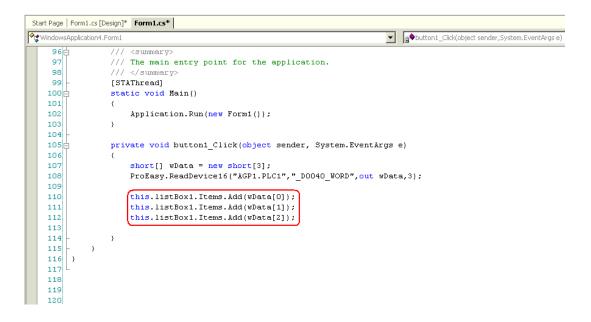
    System.EventArgs e)

       96時
                                <summary>
                          ///% \left( {{\mathcal T}} \right) = {{\mathbb T}} \left( {{\mathbb T}} \right) = {{\mathbb T}} \left( {{\mathbb T}} \right) = {{\mathbb T}} \left( {{\mathbb T}} \right) = {{\mathbb T}} \left( {{{\mathbb T}} \right) = {{\mathbb T}} \left( {{{\mathbb T}} \right) = {{\mathbb T}} } \right) .
       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内
      106
                           {
      107
                                 short[] wData = new short[3];
      108
                                 ProEasy.ReadDevice16("AGP1.PLC1","_D0040_WORD",Enter value(Receive buffer) here);
      109
      110
                          3
      111
                    }
      112
            }
      113
      114
      115
      116
```

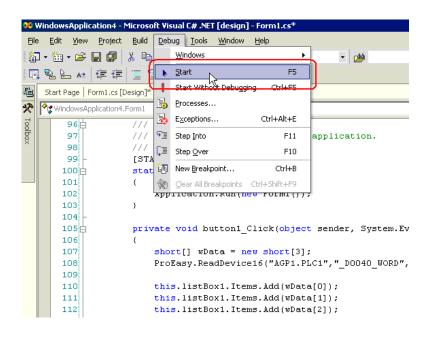
16 Specify a data storing area "wData" with the reference modifier (out), as the third argument. Enter "," (comma) at the end of the third argument, and then enter "3" to specify the length of the target symbol as the fourth argument.



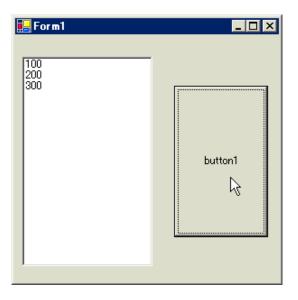
17 Add the read data on three items (wData[0], wData[1], wData[2]) into [listBox1] in this order.



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



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



Creating "Writing" application

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

🖳 Form 1	
100	
200	button1
300	

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

🥺 WindowsApplication5 - Micro	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	
🎦 • 🛅 • 🚔 🖬 🕼 🛛	🛍 🛍 🗠 - 🗠 - 🚚 - 📕 🕞 Debug 🔹 🍻	•
[井 타 후 릐 ㅠ ~ 관	류 寺 前 段 마 캬 캬 캬 봄 찾 밝 타 匝 원 📲 🖷 🕇	
Toolbox 4 ×	Start Page Form1.cs [Design]*	
Data		
Components	Form1	
Windows Forms		
📐 Pointer		
A Label	textBox1	
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 ¥isual C# .NET [design] - Form1.cs [Design]*
<u>File Edit View Project Bu</u>	ild <u>D</u> ebug D <u>a</u> ta <u>T</u> ools <u>W</u> indow <u>H</u> elp
🛅 • 🛅 • 😅 🖬 🕼 🐰	🖹 🛍 🗠 - 🗠 - 🚚 - 🖳 Debug 🔹 🍎
井 臣 후 릐 两 애 맨	🖙 葉 印 路 👓 跸 말 및 불 찾 황 탁 凾 图 🐚 🖏 🖡
Foolbox 🛛 🕈 🗙	Start Page Form1.cs [Design]*
Data	
Iomponents	Form1
Windows Forms 📃 🔺	·····
Pointer	
A Label	textBox1
A LinkLabel	· · · · · · · · · · · · · · · · · · ·
ab Button	textBox2
abl TextBox	
🛓 MainMenu	··· textBox3
CheckBox	
RadioButton	
[^{xv}] GroupBox	·····
RictureBox	
Panel	
📶 DataGrid	
E ListBox	

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

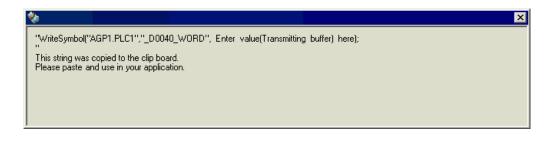
💱 Pro-Studio EX 2.npx					_ 🗆 ×
File Edit Tool Programming Assist Setti	ng Help				
Start 🔉 🟹 Node 🔉	녿 Symbol ⋗ 葇	Feature ≫ 📑	👔 Save ン 🔖	Transfer	Monitor Status
Symbol	Node Name AGP1		Device Name PLC1		
Group Ungroup	Sheet Name Sheet3		Set it as a global symbol sh	eet.	
Insert Delete	,				
Copy Cut Paste	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				1	
Pro-Server EX				1	
i III PC1 (192.168.0.1)				1	
IIII #INTERNAL:Sheet1				1	
E GP3000 Series				1	
GP1 (192.168.0.100)				1	
PLC1:Sheet3 A Series CPU [II			1	
→ WinGP				1	
> LT3000				- 1	
GP Series				1	
Global Symbol				- 1	·
				1	·
				- 1	
				1	· · · · · ·
	•	1			

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

 $23 \hspace{0.1 cm} \text{Select [VC++] - [Write Function] from the [Programming Assist] menu.}$

📚 Pro-Studio E>	(123.np	ж			
File Edit Tool	Programm	ning Assist	Setting	Help	
	VB & VI	BA 🕨 📜		<u> </u>	
Start Start	VC++	• •	eclare S	tatement	" 룫
	EXCEL	•	iroup str	ucturization	
Symbol		- V	Vrite Fur	nction	GP1
Group		UngroF	lead Fun	iction k	No 10
		_		Sneet Name	sheets
Insert		Delete			
Сору	Cut	Paste		Symbol	D
-Contrationary				0 WORD	16Bit(Si

The write function is copied to the clipboard.

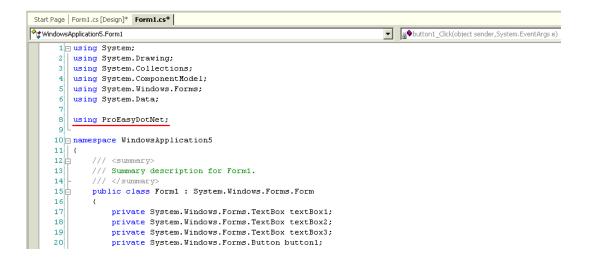


24 Double-click [button1] in [Form1], and paste the clipboard data (write function) below the [button1_Click] method ("private void button1_Click..." character string).

```
Start Page Form1.cs [Design]* Form1.cs*
                                                                            ▼ average button1_Click(object sender,System.EventArgs e)
✤ WindowsApplication5.Form1
    12
             /// </summary>
    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;
    18
                 private System.Windows.Forms.Button button1;
    19 白
                 /// <summary>
    20
                /// Required designer variable.
    21
                /// </summary>
    22
                private System.ComponentModel.Container components = null;
    23
    24
                 public Form1()...
    36
                 /// <summary>
    37
                /// Clean up any resources being used.
    38
                /// </summary>
    39 🗄
                 protected override void Dispose( bool disposing )...
    51
                Windows Form Designer generated code
    111
    112 白
                 /// <summary>
    113
                 /// The main entry point for the application.
    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","_DOO40_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*
♥☆ WindowsApplication5.Form1
                                                                              ▼ 😽 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由
    53 🖨
                 Windows Form Designer generated code
    113
    114
                 /// <summary>
    115
                 /// The main entry point for the application.
                 /// </summary>
    116
    117
                 [STAThread]
    118
                 static void Main()
    119
                 {
    120
                      Application.Run(new Form1());
    121
                 }
    122
    123
                 private void button1 Click(object sender, System.EventArgs e)
    124
                  - {
    125
                      short[] wData = new short[3];
    126
    127
                      WriteSymbol("AGP1.PLC1","_D0040_WORD", Enter value(Transmitting buffer) here);
    128
    129
                 }
    130
             }
    131 }
    132
    133
    134
    135
    136
```

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

```
Start Page | Form1.cs [Design]* Form1.cs*
✤ WindowsApplication5.Form1
                                                                              ▼ System.EventArgs e)
    26
38 🗗
                 public Form1()...
                 /// <summary>
     39
                 /// Clean up any resources being used.
     40
                 /// </summary>
     41 🕁
                 protected override void Dispose( bool disposing )...
    53 🖨
                 Windows Form Designer generated code
    113
   114
                 /// <summary>
                 /// The main entry point for the application.
   115
   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];
wData[0] = short.Parse(this.textBox1.Text);
   126
                     wData[1] = short.Parse(this.textBox2.Text);
   127
   128
                     wData[2] = short.Parse(this.textBox3.Text);
   129
                      WriteSymbol("AGP1.PLC1","_DOO40_WORD", Enter value(Transmitting buffer) here);
   130
   131
   132
                 }
   133
             }
   134 }
   135
   136
   137
   138
```

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

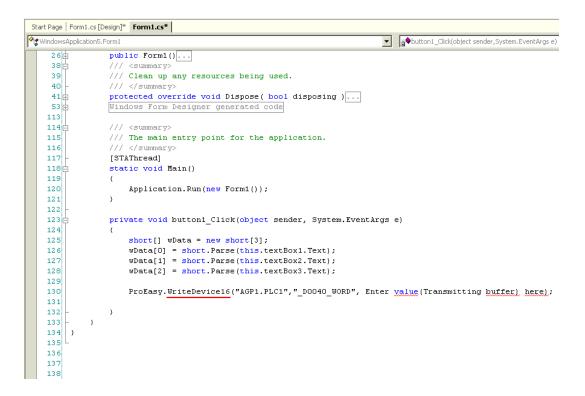
```
Start Page | Form1.cs [Design]* Form1.cs*
VindowsApplication5.Form1

    Boutton1_Click(object sender,System.EventArgs e)

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

SymbolInformation
    134
                             VriteDevice16
    135 L
    136
                             VriteDevice16D
    137
                             VriteDevice16DM
    138
                             WriteDevice16M
                             🔷 WriteDevice32
    139
                             🕸 WriteDevice32D
    140
    141
    142
    143
    144
    145
    146
```

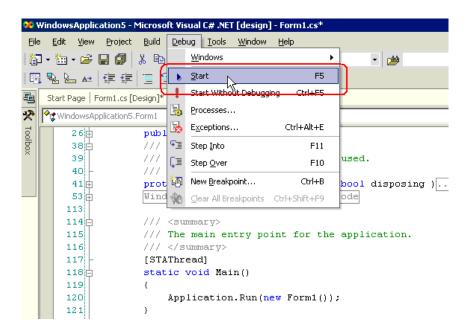
29 Delete "WriteSymbol" from the character string (write function) that has been pasted from the clipboard.



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

```
Start Page | Form1.cs [Design]* Form1.cs*
WindowsApplication5.Form1
                                                                              ▼ 😽 button1_Click(object sender,System.EventArgs e)
     26曲
                 public Form1()...
     38
                 /// <summary>
     39
                 /// Clean up any resources being used.
     40
                 /// </summary>
     41由
                 protected override void Dispose( bool disposing )...
     53 🗄
                 Windows Form Designer generated code
    113
                 /// <summary>
    114世
    115
                 /// The main entry point for the application.
                 /// </summary>
    116
    117
                 [STAThread]
    118
                 static void Main()
    119
                 {
    120
                      Application.Run(new Form1());
    121
                 -}
    122
    123
                 private void button1_Click(object sender, System.EventArgs e)
    124
                 -{
    125
                      short[] wData = new short[3];
                      wData[0] = short.Parse(this.textBox1.Text);
    126
                      wData[1] = short.Parse(this.textBox2.Text);
    127
                      wData[2] = short.Parse(this.textBox3.Text);
    128
    129
    130
                      ProEasy.WriteDevice16("AGP1.PLC1","_D0040_WORD",wData,3);
    131
    132
                 }
    133
             }
    134 }
    135
    136
    137
    138
```

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



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

E Form1	
textBox1	
textBox2	button1
textBox3	
]

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

🛃 Form1	
100	
200	button1
300	k