OMRON

FinsGateway
FINS API

Programmer's Manual

Version 2 8/17/1998
OMRON Corporation

Contents

Revision History

Revision code	Date	Revised content
1.00	August 1998	Original production

.

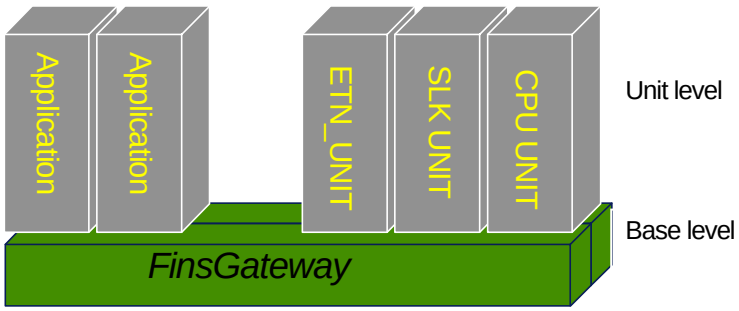
1 Introduction

1.1 FINS API

This manual describes the FINS API to use FINS, which is supported by OMRON FA networks on Windows95/WindowsNT. FINS API is provided as a component of FinsGateway.

For details of FINS command/responses, refer to the FINS Command Reference Manual (Cat. No. W227).

FinsGateway provides FINS API, which makes the network type almost transparent for applications to communicate using the FINS commands/responses. In FinsGateway, the Win32 applications are virtualized as high-function units of the PLC, so that the PCs as nodes on the network can be treated equally with the PLCs. The network-dependent portions are implemented as communication units (services), and the server function of FINS commands/responses is implemented as the CPU unit (service).



The major features of FinsGateway are the following:

A communication platform is provided in the personal computer to use OMRON FA networks, so that the user can communicate without concern for the specific network such, as SYSMAC LINK or Ethernet.

FinsGateway performs data transfer between units, recognizing the applications and network providers (including individual communication drivers) as equal units.

The communication functions (routing and gateway functions) of the same level as the PLC are provided.

- It is possible to perform direct communication with the PLCs which belong to other networks(i.e., realization of multi-layer network communication) by the FINS communication.
- It is possible to use a personal computer as a gateway between networks.

1.2 Operating Environment

The files needed for application development are as follows:

DLL	FinsGW32.dll, FinsMisc.dll
Import library	FinsGW32.lib
Include files	FinsGW.h, FinsHead.h, Fins.h, FinsLog.h, FgwError.h FgwResrc.h FinsHead.h, Fins.h, and FinsLog.h is included in FinsGW.h. FgwResrc.h is included in FgwError.h.

1.3 Structure

The basic processing requested of FINS API from applications is data send and receive. The data consists of the header section to control protocol and the application message section. The header consists of the portion from ICF to SID in the FINS protocol and a local-node-specific data structure used by FinsGateway.

1.3.1 Configuration

FINS API operates in cooperation with the following functions:

- FINS API library: Library that provides the API (Application Programming Interface) for FINS communication. The major functions are message sending and receiving.
- Communication unit: Exists for each network and actually performs sending and receiving on the network. Offers the same FINS services as PLC communication units offer.
- Communication driver: Exists for each network and directly controls the hardware of the network board.
- Main unit: The only system service existing in the system. Offers FINS services comparable to the main unit of PLC.

1.3.2 Unit Addresses

FinsGateway allows two or more applications to transmit/receive at the same time. To identify the applications, FINS API assigns unit addresses to them when they are registered.

Unit addresses from 0x01 to 0x0F can be explicitly assigned to user applications. These addresses are primarily used for applications having server functions. The Fins_new() function is used to register unit addresses from applications to FINS API.

If user applications have only client functions, the choice of unit addresses can be left to FINS API. The Fins_newReservedUnit function is used to register unit addresses from applications to FinsGateway.

Unit addresses from 0x10 to 0x2F can be assigned to communication units. In most cases, they are assigned to OMRON-offered communication units.

Own communication units can be operated with a unit address of 0xFE in addition to the unit addresses actually assigned.

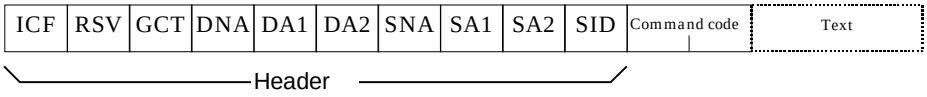
Unit address allocation

00	01 ... 0F	10 ... 2F	30 ... CF	D0 ... EF	F0 ... FF
Main units of a personal computer	User applications	Communication units	Not used	User applications (reserved)	Reserved by the system

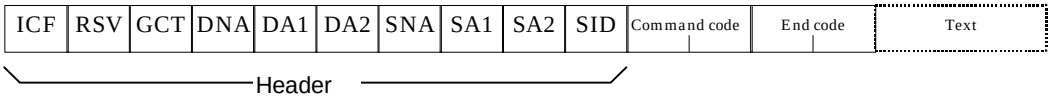
1.3.3 FINS Messages

The FINS command reference*describes the configuration of FINS command/response as follows:*

Configuration of send/receive data for command



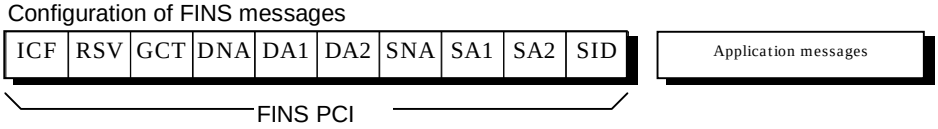
Configuration of send/receive data for response



FinsGateway is somewhat different from the FINS command reference in the description of interpretation and definition, although the data string is the same. FinsGateway handles both FINS commands and responses as FINS messages without differentiating between them.

* Explanations of the meanings of ICF and RSV are omitted (see the FINS Command Reference Manual).

Configuration of FINS messages



The application messages refer to those in the application layer(*). FINS command codes and the end code are part of application messages, and FinsGateway, as a rule, is not involved with the content of application messages. FINS API designates as a transport address a set of network number, node number, and unit number, which represents a unit on FINS communication domain. Thus, a set of DNA, DA1, and DA2, and a set of SNA, SA1, and SA2 are interpreted as a transport address and a sender transport address.

2 Using FinsGateway

First, specify the operating version of the FINS API.

The operating version of FinsGateway must be specified before using FinsGateway. This implementation makes it possible to run applications without recompiling them, when an updated version of FinsGateway is used in the future.

Use the Fins_requestVersion function to specify the version. Note that if the function is not executed, all the functions provided by FinsGateway result in an error (unsupported version), and they can not be used.

Connection with FinsGateway

Second, it is necessary to register the applications as units which FinsGateway can recognize. This processing is to allocate a unit address to the application. For an application with the FINS server function, a unit address must be explicitly allocated using the Fins_new function. For other applications such as monitoring software which needs not recognize, use the Fins_newReservedUnit function to which FinsGateway automatically allocates unit addresses.

2.1 Sending a FINS Command

Command Header Preparation

The FINS header must be prepared using the FinsHead_compose function. The destination unit address includes the network, node and unit numbers. It is also necessary to specify the SID, so that the application can recognize the transmitted FINS command.

Message Preparation

Refer to the FINS Command Reference Manual (Cat. No. W227) to prepare a FINS message (command).

Transmission

Send the prepared FINS header and message using the Fins_sendData function.

2.2 Returning a FINS Response

Response Header Preparation

For transmission of FINS response, it is a prerequisite that a FINS command has been received from another unit. Prepare the response header needed to respond using the FinsHead_composeResponse function from the received FINS command header.

Message Preparation

Translate the received FINS message (command) and prepare the FINS response. Refer to the FINS Command Reference Manual for details of the FINS response format.

Transmission

Transmit the prepared FINS header and FINS message using the Fins_sendData function.

2.3 Receiving FINS Commands/Responses

There are three methods that an application can receive a FINS command/response; message driven, polling, and synchronization. Select the receiving method by the purpose as follows:

Receiving method	Description
Message driven method	✓ Performs reception processing by an incoming notice of the previously set Windows message. ✓ The application which received the Windows message executes the Fins_receiveData function ✓ The feature of this method is that other Windows messages are not blocked. Redrawing messages of the screen generated by the Windows system can be smoothly processed.
Polling method	✓ Periodically checks if any incoming data is received with the timer or at the background (PeekMesssage loop) using the Fins_receiveData function (reception standby time = 0). ✓ The feature of this method is that other Windows messages are not blocked. Redrawing messages of the screen generated by the Windows system can be smoothly processed.
Synchronization method	✓ Blocks all other processing, waiting for reception with the Fins_receiveData function until incoming data is received. ✓ It is possible to set a reception standby time. Stand by time = TimeLimit (0 to 4294967294 ms) or unlimited.

2.3.1 Message-driven Reception

Specify the notice message type using the Fins_setMessageOnArrival function or the Fins_setThreadMessageOnArrival function, in order to send a Windows message automatically to the window or thread, when the application (unit) receives incoming data. The specified Windows message will then be posted to the application (Windows or thread) at the arrival of a FINS message.

2.3.2 Waiting for Multiple Objects

Since it is possible to get an event handle to be used for the reception standby by the Fins_getEventHandle function, generation of multiple events can be waited for by the WaitForMultipleObjects function. An example is shown below. See the Win32 API Manual for the WaitForMultipleObjects function.

```
HANDLE lphObjects[2];
DWORD dwRet;

        . . . . .
lphObjects[0] = Fins_getEventHandle(ghNet);
lphObjects[1] = hEvent;

dwRet = WaitForMultipleObjects( 2,lphObjects,
                    WAIT_ANYONE,INFINITE);
switch(dwRet) {
case 0:
    nRecvSize = Fins_receiveData( hNet,
                                    &head,lpBuff,nSize,TIME_LIMIT);
    break;
default:
     . . . . .
    break;
}
```

2.4 Inter-process Communication

In FinsGateway, an application is handled as a virtual communication unit with a PLC unit number. Therefore, a FINS message can be transmitted specifying the FINS transport address in the same manner as the PLC. If the inter-process communication is to be explicitly indicated, transmit the message with the network and node numbers set to 0.

The destination unit address is the one with which the destination unit is registered using the Fins_new function. If the network, node, and unit numbers are 0, the message is sent to the main unit (service) residing in the computer.

2.5 Ending Communication

A unit address was assigned to the application by the Fins_new function to use FinsGateway. However, the assigned unit address must be released by the Fins_delete function when the communication process finishes, so that another application can use it.

If the application is finished without calling the Fins_delete, FinsGateway detects the detachment of the process and automatically performs the end processing (Fins_delete). Note, however, the unit address may not be released if the application was stopped or terminated using the debugger.

3 Error Events

If an error is detected in FINS routing, FinsGateway normally returns an error response. The error responses conform to the FINS command/response.

If it is not possible to return an error response to the message sent from an application, FinsGateway performs the following:

- Error return of the FINS API function
- Logging of error in the main and communication units

Incorrect use of the FINS API or argument result in function call errors.

3.1 Function Call Errors

If a call of the library functions provided by FinsGateway fails, one of the return values, NULL(0), C_RETURN_CODE_FAILED(-1), or FALSE(0) will be returned.

The application can identify the error cause by calling the GetLastError( ). The error code is managed by each thread, so that detailed error code is available about the function executed immediately before when errors occurred in multiple threads.

An error code consists of 32 bits (bit 31 is the most significant bit). An error code of FinsGateway is always set with the bit 29, as it is reserved by Microsoft Corporation for error code of the application definition. A list of error codes is shown in Table 1.

Table 1 Function error codes[†]

CODE	Definition(GW_ERROR_*)	Meaning
1	NOT_SUPPORTED	Service is not supported
2	INVALID_HANDLE	Incorrect network handle
3	NOT_ENOUGH_MEMORY	Impossible to execute because of memory shortage
4	ENTRY_ALREADY_EXISTS	Unit of the same unit address already exists
5	NETWORK_BUSY	Network busy
6	BUFFER_OVERFLOW	Receiving buffer overflow
7	DATA_SIZE_TOO_LONG	Data size is out of range to handle
8	INVALID_FINS_HEADER	Incorrect FINS header
9	RECEIVE_TIMEOUT	Receiving time out
10	ILLEGAL_SID	Invalid SID setting
11	NOT_SUPPORTED_VERSION	Version is not supported; impossible to execute
12	BAD_NETPATH	Wrong setting of routing table; cannot reach final network
13	ROUTE_ILLOGICAL	Routing table abnormal
14	CONSUME_GCT	Allowed number of gateways exceeded
15	VERSION_ALREADY_LOCKED	Version already in locked state and can't change
16	NO_RESPOND	Attempted to make header to response
17	NOT_NEED_ACK	Attempted to make response to command which requires no response
18	EVENT_CREATE_FAILED	Could not make event object to be used for send data notice
19	LIST_CREATE_FAILED	Could not make event object to be used for send data notice
20	WAIT_MUTEX_FAILED	Failed in standby process of exclusive control of shared memory

[†]Codes in the above table show values with the upper 2 bytes masked.

21	MEMORY_MAPPING	Failed in mapping of shared memory
22	NAME_TOO_LONG	Name of object is too long
23	DETACH_FREE_UNIT	Attempted to release unused unit address
24	UNEXPECTED_FAILED	Unexpected error occurred
25	MEMORY_INITIALIZE_FAILED	Failed to initialization of shared memory
26	MEMORY_ATTACH_FAILED	Failed in attachment to shared memory
27	NO_MORE_ENTRY_UNITS	Number of units (units) simultaneously usable exceeded
28	NO_PEER_UNIT	Send destination unit is no existence
29	SEND_INFORM_FAILED	Could not notify data transmission
30	WAIT_ABANDONED	Exit from receive standby due to abnormal end of other process
31	RECEIVE_WAIT_FAILED	Failed in receive standby
32	NO_MORE_HANDLE	All reserved area of network handle is fully used
33	NO_RECEIVE_DATA	Incoming data is not received yet
34	INVALID_MESSAGE	Invalid Windows message type
35	POST_MESSAGE_FAILED	Failed in message notice

3.2 Cause and Action of Error Events

See the FINS Command Reference Manual for those error events which can be detected by FINS error responses.

Table 2 shows error events which can be logged.

Table 2 Error log code

Log code	Description	Cause	Action
0x0108	Impossible to send as no relevant unit exists.	The unit is not participating in FinsGateway.	Start the application for the unit.
0x010D	Impossible to send as destination address is missing in routing table.	Setting error of routing table or invalid destination address.	Set the routing table again.
0x010E	Impossible to send as routing table is not registered yet.	The routing table is not registered.	Register the routing table.
0x010F	Impossible to send as routing table is abnormal.	Impossible to retrieve destination address.	Register the routing table.
0x0110	Impossible to send as number of relays exceeds.	Data is transmitted to networks of more than 3 layers.	Check the network configuration.
0x0117	Internal buffer full.	Load is concentrated to a single unit. There is some application which is not in reception operation.	Distribute the load in the system. Check the reception operation of applications.
0x0118	Invalid packet discarded.	Header was wrong if the error occurred at response distribution.	Check the send/receive messages of the applications.

4 FINS API Reference

4.1 Summary

The library consists of multiple functions of the following:

Version management	
Fins_requestVersion	Specification of version management
Fins_getVersion	Acquisition of DDL release version

Basic Operation	
Fins_new	Generation of network handle
Fins_newReservedUnit	Generation of network handle (automatic)
Fins_delete	Delete of network handle
Fins_sendData	FINS transmit processing
Fins_receiveData	FINS receive processing

Setting of message-driven type of reception	
Fins_setMessageOnArrival	Setting of Windows message (posting to window)
Fins_setThreadMessageOnArrival	Setting of Windows message (posting to thread)
Fins_clearMessageOnArrival	Clear of setting of Windows message

Acquisition operation of internal information	
Fins_getMessageSize	Acquisition of FINS message size
Fins_getEventHandle	Acquisition of FINS reception event handle
Fins_getTaddress	Acquisition of FINS transport address
Fins_getNetworkInfo	Acquisition of network information
Fins_getConnectUnits	Acquisition of FinsGateway registered unit information

Operation to manipulate send/receive data	
FinsHead_compose	Generation of FINS command header
FinsHead_composeResponse	Generation of FINS response header

Miscellaneous	
Fins_getLastErrorMessage	Gets an error message associated with FINS API error code.

4.2 Fins_requestVersion Function

Function

Defines the operation version of the F1NS-API.

```
BOOL Fins_requestVersion(
                BYTE byMajor,       //major version
                BYTE byMinor        //minor version
                )
```

Description

The Fins_requestVersion function requests FinsGateway to operate in the specified version. If the operation version is not specified by this function, all the functions of the FINS API will fail. The FGW_STARTUP( ) macro is defined by the Fins_requestVersion function as the argument of the release version.

Argument	Explanation
byMajor	The major version no. FinsGW.h(FinsHead.h) is defined with the CURRENT_MAJOR_VERSION as the major version at the release.
byMinor	The minor version no. FinsGW.h(FinsHead.h) is defined with the CURRENT_MINOR_VERSION as the minor version at the release.

Return Value

If the function terminates normally, the TRUE is returned. Otherwise, the FALSE is returned. Use the GetLastError function to get detailed error information.

Specification of a version which is not in the release history results in an error.

See Also

Fins_*, FinsHead_*

4.3 Fins_getVersion Function

Function

Obtains the release version of the FinsGW32.dll.

```
FGW_VERSION Fins_getVersion(VOID)
```

Description

The Fins_getVersion function obtains the release version of FinsGW32.dll. It has no relation with the operation version requested by the Fins_requestVersion function.

Argument	Explanation
None	

Return Value

If the function terminates normally, the VERSION structure body is returned. There is no situation in which this function will fail.

```
typedef struct _Version_Struc {
                BYTE   Major;               //major version
                BYTE   Minor;               //minor version
                BYTE   Revision;     //revision
                BYTE   Reserved;     //reserved area
} FGW_VERSION;
```

See Also

Fins_requestVersion

4.4 Fins_new Function

Function

Generates the network handle which the FINS API requires.

```
HNET Fins_new(
                BYTE   byUnitAddr,   //unit address
                LPSTR lpszUnitName //unit name
                )
```

Description

The Fins_new function connects the application to FinsGateway. The application which uses FinsGateway has to execute this Fins_new function first, then the network handle as the return value. After connection is established between the application and FinsGateway by the Fins_new function, the send/receive units are identified by the network handle.

Argument	Explanation
byUnitAddr	The address of the unit to connect to FinsGateway. After executing this function, the application is identified on the network node by the unit address specified by the byUnitAddr.
lpszUnitName	The name of the unit to connect to FinsGateway. It is registered to FinsGateway and can be referred from other applications. The maximum number of characters of the lpszUnitName is 15, and if a character string longer than that length is specified, only the first 15 characters are registered.

Return Value

If the function terminates normally, the network handle is returned. Otherwise, the NULL is returned. Use the GetLastError function to get detailed error information.

If another application is already using the same unit address, it results in an error.

See Also

Fins_requestVersion, Fins_newReservedUnit, Fins_delete

4.5 Fins_newReservedUnit Function

Function

Automatically allocates and generates the network handle required by the FINS API to the unit address.

```
HNET Fins_newReservedUnit(
                LPSTR lpszUnitName // unit name
                )
```

Description

The network allocates an unused address from the unit addresses reserved by the system and generates the network handle. If the application needs to recognize its own unit address, the Fins_newReservedUnit function is used instead of the Fins_new function. A reserved unit address has the same meaning as the Reserved Port of the TCP/IP, BSD Socket.

Argument	Explanation
lpszUnitName	The name of the unit to be connected to FinsGateway. It is registered to FinsGateway and can be referred from other applications. The maximum number of characters of the lpszUnitName is 15, and if a character string longer than that length is specified, only the first 15 characters are registered.

Return Value

If the function terminates normally, the network handle is returned. Otherwise, the NULL is returned. Use the GetLastError function to get detailed error information.

See Also

Fins_requestVersion, Fins_new, Fins_delete

4.6 Fins_delete Function

Function

Releases the network handle.

```
int Fins_delete(
                HNET hNet    //network handle
                )
```

Description

The Fins_delete function deletes the contact point with FinsGateway connected by the Fins_new or Fins_newReservedUnit function. The application which terminated communication processing has to call this function and release the unit address in use.

If the application is terminated without calling this function, FinsGateway detects the detachment of the process and automatically performs the end processing (Fins_delete). However, if the application is stopped or terminated using the debugger or any other means, the unit address may not be released.

Argument	Explanation
hNet	The network handle generated by the Fins_new or Fins_newReservedUnit function.

Return Value

If the function terminates normally, the C_RETURN_CODE_SUCCESS(=0) is returned. Otherwise, the C_RETURN_CODE_FAILED(=-1) is returned. Use the GetLastError function to get detailed error information.

See Also

Fins_new, Fins_newReservedUnit

4.7 Fins_sendData Function

Function

Transmits FINS messages

```
int Fins_sendData(
                HNET        hNet,          //network handle
                LPFINSHEAD lpHead,         //FINS header
                LPVOID      lpMessage,     //FINS message
                DWORD        dwSize              //FINS message size
                )
```

Description

The Fins_sendData function transmits data using FinsGateway. The send destination is determined by the FINS transport address when the FINS protocols are used.

If other protocols are used, the data is transferred to a local unit specified by the GWPCI in the header.

Argument	Explanation
hNet	Specifies the network handle generated by the Fins_new or Fins_newReservedUnit function.
lpHead	Header section (protocol control information) This header includes a code to control FinsGateway, in addition to the header of 10 bytes of ICF to SID which depend on the FINS command/response. Therefore, it must be generated using the FinsHead_compose/FinsHead_composeResponse function.
lpMessage	The pointer of the user buffer which stores the FINS message (from MRC to the end).
dwSize	The size (in bytes) of the FINS message (from MRC to the end).

Return Value

If the function terminates normally, the size of transmitted message (use data) is returned. Otherwise, the C_RETURN_CODE_FAILED(=-1) is returned. Use the GetLastError function to get detailed error information.

See Also

Fins_new, Fins_newReservedUnit, Fins_receiveData, FinsHead_compose, FinsHead_composeResponse

4.8 Fins_receiveData Function

Function

Receives FINS messages

```
int Fins_receiveData(
                HNET        hNet,          //network handle
                LPFINSHEAD lpHead,         //FINS header
                LPVOID      lpMessage,     //FINS message buffer
                DWORD       dwSize,              //FINS message buffer size
                DWORD       dwTimeLimit  //receive standby time
                )
```

Description

The Fins_receiveData function receives data using FinsGateway.

Argument	Explanation
hNet	Specifies the network handle generated by the Fins_new or Fins_newReservedUnit function.
lpHead	The header section (protocol control information) of the received message.
lpMessage	The pointer of receiving buffer to store the FINS message (from MRC to the end).
dwSize	The size (in bytes) of reserved receiving buffer.
dwTimeLimit	Specifies the receiving standby time in ms. Immediately returns if the standby time is 0. Normally terminates only if receive data exists. Specify the INFINITE to wait infinitely.

Return Value

If the function terminates normally, the size of received message (user data) is returned. Otherwise, the C_RETURN_CODE_FAILED(=-1) is returned. Use the GetLastError function to get detailed error information.

If the reserved receiving buffer size is smaller than the received message, it results in an error, but a portion of the received message equivalent to the buffer size is stored. The remaining portion is discarded.

If this function return as failure cause user buffer is too small, the function GetLastError will return the value GW_ERROR_BUFFER_OVERFLOW.

See Also

Fins_new, Fins_newReservedUnit, Fins_sendData

4.9 Fins_setMessageOnArrival Function

Function

Makes setting to notify the Windows by a Windows message at arrival of FINS message.

```
BOOL   Fins_setMessageOnArrival(
                HNET hNet,   //network handle
                HWND hWnd, //window handle
                UINT uMsg,   //message type
                )
```

Description

The Fins_setMessageOnArrival function specifies to notify the Windows specified by a Windows message when FINS message is received. The application is able to monitor reception in parallel with processing of other Windows messages without being blocked by at arrival processing. It is not possible to use this together with the Fins_setThreadMessageOnArrival function.

Argument	Explanation
hNet	The network handle generated by the Fins_new or Fins_newReservedUnit function.
hWnd	The window handle of the application to notify arrival of FINS message.
uMsg	Specifies the message type. Use a value between WM_USER to 0x7FFF.

Return Value

If the function terminates normally, the TRUE is returned. Otherwise, the FALSE is returned. Use the GetLastError function to get detailed error information.

See Also

Fins_setThreadMessageOnArrival, Fins_clearMessageOnArrival, Fins_receiveData

4.10 Fins_setThreadMessageOnArrival Function

Function

Makes setting to notify the thread by a Windows message at arrival of FINS message.

```
BOOL   Fins_setThreadMessageOnArrival(
                HNET   hNet,        //network handle
                DWORD dwThreadId, //Thread ID
                UINT   uMsg,        //message type
                )
```

Description

The Fins_setThreadMessageOnArrival function specifies to notify the thread specified by a Windows message when FINS message is received. The application is able to monitor reception in parallel with processing of other Windows messages without being blocked by at arrival processing. It is not possible to use this together with the Fins_setThreadMessageOnArrival function.

Argument	Explanation
hNet	The network handle generated by the Fins_new or Fins_newReservedUnit function.
dwThread	The ID of the thread to notify arrival of FINS message.
uMsg	Specifies the message type. Use a value between WM_USER to 0x7FFF.

Return Value

If the function terminates normally, the TRUE is returned. Otherwise, the FALSE is returned. Use the GetLastError function to get detailed error information.

See Also

Fins_setMessageOnArrival, Fins_clearMessageOnArrival, Fins_receiveData

4.11 Fins_clearMessageOnArrival Function

Function

Clears the setting to notify by a Windows message at arrival of FINS message.

```
BOOL   Fins_clearMessageOnArrival(
                HNET   hNet, //network handle
                )
```

Description

The Fins_clearMessageOnArrival function clears the setting of Windows message notification specified by the Fins_setMessageOnArrival or Fins_setThreadMessageOnArrival function. The Windows message notification will then not be performed when a FINS message arrives to the unit specified by hNet.

Argument	Explanation
hNet	The network handle generated by the Fins_new or Fins_newReservedUnit function.

Return Value

If the function terminates normally, the TRUE is returned. Otherwise, the FALSE is returned. Use the GetLastError function to get detailed error information.

See Also

Fins_setMessageOnArrival, Fins_setThreadMessageOnArrival, Fins_receiveData

4.12 Fins_getMessageSize Function

Function

Returns the maximum size (excluding the header) of FINS message.

```
int Fins_getMessageSize(
                HNET hNet    //network handle
                );
```

Description

Returns the maximum size (excluding the header) of FINS message that which FinsGateway can handle.

Argument	Explanation
hNet	The network handle generated by the Fins_new or Fins_newReservedUnit function.

Return Value

If the function terminates normally, the buffer size internally reserved by FinsGateway is returned. Otherwise, the C_RETURN_CODE_FAILED(=-1) is returned. Use the GetLastError function to get detailed error information.

See Also

Fins_new, Fins_newReservedUnit, Fins_sendData, Fins_receiveData

4.13 Fins_getTaddress Function

Function

Returns the FINS transport address.

```
int Fins_getTaddress(
                HNET    hNet,                   //network handle
                LPFINSADDR lpFinsTaddress //FINS transport address
                );
```

Description

The Fins_getTaddress function returns the FINS transport address. Normally, the network and node numbers are 0s as the application can not decide them. Use this function to obtain the unit address from the network handle.

Argument	Explanation
hNet	The network handle generated by the Fins_new or Fins_newReservedUnit function.
lpFinsTAddress	The pointer of the structure body to store the FINS transport address.

Return Value

If the function terminates normally, the C_RETURN_CODE_SUCCESS is returned. Otherwise, the C_RETURN_CODE_FAILED is returned. USE the GetLastError function to get detailed error information.

See Also

Fins_new, Fins_newReservedUnit

4.14 Fins_getEventHandle Function

Function

Obtains the handle of the event object which FinsGateway uses for the reception standby processing.

```
HANDLE Fins_getEventHandle(
                HNET hNet    //network handle
                );
```

Description

The Fins_getEventHandle function obtains the handle of the Win32 event object which FinsGateway uses for the reception standby processing.

Use the handle of the event object gained by this function as the argument of the WaitForMultipleObjects function, in order to wait for multiple objects turn into the signaling state together with the FINS standby.

Argument	Explanation
hNet	The network handle generated by the Fins_new or Fins_newReservedUnit function.

Return Value

If the function terminates normally, the handle of event object is returned. Otherwise, the NULL is returned. Use the GetLastError function to get detailed error information.

See Also

Fins_new, Fins_newReservedUnit, Fins_receiveData

4.15 Fins_getNetworkInfo Function

Function

Obtains the information of the network which can be currently used.

```
int Fins_getNetworkInfo(
                HNET        hNet,    //network handle
                PNETINFO    pNetInfo, //array of structure body of network
                DWORD       cbNetworks //number of networks
                );
```

Description

The Fins_getNetworkInfo function obtains the information of the network which can be currently used.

Argument	Explanation
hNet	The network handle generated by the Fins_new or Fins_newReservedUnit function.
pNetInfo	The pointer of the structure body to store the network information.

Structure for the information a network

```
typedef struct _Network_Info_Struc {
                char
szName[MAX_COMMENT_SIZE];
                FINSADDR transport;
                DWORD dwMessageSize;
} NETINFO, *PNETINFO;
```

Structure member	Explanation
szName	Network name（max 15 characters + NULL terminated）
transport	FINS transport address
dwMessageSize	Maximum message size（for MRC b to the end of message）

cbNetworks	Number of networks

Return Value

If the function terminates normally, number of networks which can be currently used is returned. Otherwise, the C_RETURN_CODE_FAILRD(=-1)is returned. Use the GetLastError function to get detailed error information.

If the argument pNetInfo=0, or if the argument cbNetworks=0, this function set the maximum number of networks and return as success.

See Also

Fins_new, Fins_newReservedUnit

4.16 Fins_getConnectUnits Function

Function

Obtains the information of the units currently connected.

```
int Fins_getConnectUnits(
                HNET           hNet,              //network handle
                PCONNECTINFO pConnectInfo,
                                   //array of structure body of
                                   //connected unit information
                DWORD          cbUnits //number of storable units
                )
```

Description

The Fins_getConnectUnits function obtains the information of the units currently connected FinsGateway.

Argument	Explanation
hNet	The network handle generated by the Fins_new or Fins_newReservedUnit function.
pConnectInfo	The pointer of the structure body to store the unit information.

The structure for the information of a connected unit

```
typedef struct _Connect_Info_Struc {
                    BYTE byUnitAddr;
                    Char
zUnitName[MAX_COMMENT_SIZE];
} CONNECTINFO, *PCONNECTINFO;
```

Structure member	Explanation
byUnitAddr	Unit number
szUnitName	Unit name (max 15 characters + NULL teminated)

cbUnits	Number of units which can be stored.

Return Value

If the function terminates normally, number of networks which can be currently used is returned. Otherwise, the C_RETURN_CODE_FAILRD(=-l) is returned. Use the GetLastError function to get detailed error information.

If the argument pConnectInfo=0, or if the argument cbUnits=0, this function returns the current number of units connected to FinsGateway.

See Also

Fins_new, Fins_newReservedUnit

4.17 FinsHead_compose Function

Function

Prepares the header of the FINS command.

```
int FinsHead_compose(
                LPFINSHEAD lpHead,//FINS header
                BYTE       icfBits, //least significant ICF bit
                LPFINSADDR lpAddr,
                        //FINS transport address of send destination
                int        nSid     //service ID
                );
```

Description

The FinsHead_compose function is a utility function to prepare the header section to be passed to FinsGateway by the Finds_sendData function. The header section prepared by this function specifies the FINS protocol header.

Argument	Explanation
lpHead	Specifies the pointer of the FINS header structure body.
icfBits	Specifies if the response is required or not for the FINS command. C_NEED_RESPONSE(=0): response required C_NO_ACK_RESPONSE(=1): response not required
lpAddr	Specifies the FINS transport address of the send destination.
nSid	Specifies the service ID attached to the FINS. If SID = -1, the SID store area is automatically incremented every time the FinsHead_compose function is called. ‡

Return Value

If the function terminates normally, the SID attached to the prepared header section is returned. Otherwise, the NULL is returned. Use the GetLastError function to get detailed error information.

See Also

Fins_sendData

‡ Note:The SID store area which is automatically incremented in the function is assured only one for the process. Note that it is not by the unit of network handle nor thread.

4.18 FinsHead_composeResponse Function

Function

Prepares the response header from the FINS command header.

```
int FinsHead_composeResponse(
                LPFINSHEAD   lpHeadR,          //response header
                LPFINSHEAD   lpHeadC,          //command header
                );
```

Description

The FinsHead_composeResponse function prepares the response header to the command if data received by the application is a FINS command. The function automatically set the SA, DA and SID of the command as the SA, DA and SID of the response.

Argument	Explanation
lpHeadR	The header of response protocol.
lpHeadC	The header of command protocol.

Return Value

If the function normally terminates, the C_RETURN_CODE_SUCCESS(=0) is returned. Otherwise, the C_RETURN_CODE_FAILED(=-1) is returned. Use the GetLastError function to get detailed error information.

If the response header is passed as the input, an error is returned.

See Also

Fins_sendData, Fins_receiveData

4.19 Fins_getLastErrorMessage Function

Function

Gets an error message associated with FINS API error code.

```
BOOL Fins_getLastErrorMessage (
                DWORD   dwCode,                 //error code
                LPSTR   lpszErrorMessage,  //pointer for string
                DWORD   dwSize
                );
```

Description

The Fins_getLastErrorMessage function gets the error message associated with the error code of the FINS API detailed error information.

Argument	Explanation
dwCode	Error code of detailed error information gotten by the GetLastError function
lpszErrorMessage	Pointer to the start of a character-string array to store error messages.
dwSize	Number of character-string arrays to store error messages. If the number of character strings for error messages exceeds dwSize, this function sets dwSize as the number of character strings for error messages obtained.

Return Value

The function returns TRUE if it terminates normally. In other cases, it returns FALSE. To get additional error information, use the GetLastError function.

See Also

GetLastError

5 Data Structure

5.1 FINS Message

FINS Transport Address

```
typedef struct _FinsTAddress_Struc {
                BYTE    byNetAddr;
                BYTE    byNodeAddr;
                BYTE    byUnitAddr;
}   FINSADDR,   *LPFINSADDR, *PFINSADDR;
```

Structure member	Explanation
byNetAddr	Network number
byNodeAddr	Node number
byUnitAddr	Unit number

FINS PCI (Protocol Control Information)

```
/*see. "FINS Command Reference Manual" */
typedef struct _FinsPci_Struc {
                BYTE    ICF;
                BYTE    RSV;
                BYTE    GCT;
                BYTE    DNA;
                BYTE    DA1;
                BYTE    DA2;
                BYTE    SNA;
                BYTE    SA1;
                BYTE    SA2;
                BYTE    SID;
}               FINSPCI, *PFINSPCI;
```

Structure member	Explanation
ICF	Information control field
RSV	System reserved area
GCT	Number of allowed gateway(Gateway Count)
DNA	Destination network number
DA1	Destination node number
DA2	Destination unit number
SNA	Source network number
SA1	Source node number
SA2	Source unit number
SID	Service ID

5.2 Transmission Header

Header Definition (GATEWAY PCI + FINS PCI)

```
typedef struct _FinsHeader {
                GWPCI       gwPci;          //Internal Info. for FinsGateway
                FINSPCI     finsPci;        //FINS Protocol Header
} FINSHEAD, *LPFINSHEAD, *PFINSHEAD;
```

Structure member	Explanation
gwPci	GATEWAY-PCI: Internal information for FinsGateway
finsPci	FINS-PCI: FINS protocol header

6 SYSMAC NET Functions

FinsGateway + SYSMAC NET provides functions for SYSMAC NET communications with control devices such as SYSMAC series PLC to applications on a personal computer.

Applications that perform communications using FinsGateway can communicate with control devices via standardized communications protocols, via FINS (Factory Interface Network Service), which is a communications service, and via a general API (Application Programming Interface).

FinsGateway + SYSMAC NET supports transmission with PLCs that do not support FINS, such as the SYSMAC C series. FinsGateway + SYSMAC NET also supports ASCII code.

6.1 C-series and CV-series in One System

6.1.1 Command Modes

A SYSMAC NET network can include both C-series SYSMAC NET Lint Units (mounted to C-series PLCs), CV-series SYSMAC NET Lint Units (mounted to CV-series PLCs) so that these PLCs and personal computers can exchange data.

The SYSMAC NET unit on FinsGateway can be used in either of the following two modes (similar to CV-series SYSMAC NET Link Units). C-mode enables it to communicate with C-series SYSMAC NET Link Units on the same network.

CV-mode: Used by networks that include only CV-series PLCs. These PLCs and personal computers can exchange data using only FINS message in this mode. Communications are not possible with C-series PLCs in this mode.

C-mode: Used by networks that include both CV-series and C-series PLCs. The CV-series PLCs and the C-series PLCs can exchange data in the C-mode using Send and Receive instructions. The Units must be set to C-mode in the communications unit settings.

In C-mode transmission, the application software on FinsGateway can use both message types that are FINS commands and C-mode commands.

6.1.2 Settings

SYSMAC NET Unit on FinsGateway

To select the mode in the datagram message specifications of the communications unit(C-mode or CV-mode), use the SNT Network Config utility. Restart the unit or the personal computer after changing the mode.

CV-series PLCs

To select the mode in the datagram message specifications of the communications unit(C-mode or CV-mode), use the PLC programming tools on personal computer. Restart the Unit or the PLC after changing the mode.

7 SYSMAC NET C-mode Transmission

In C-mode, a FinsGateway application can use both FINS messages and C-mode messages.

7.1 FINS Messages (with Conversion)

There is an unspoken agreement between the application software and the SYSMAC NET unit on FinsGateway. If the application software sends a FINS message to another node in C-mode, the unit translates into a C-mode message automatically.

The SYSMAC NET unit on FinsGateway converts FINS commands transmitted to the destination node to C-mode internally upon sending the command if the Datagram Message Specifications are set to C-mode. If the unit receives a C-mode response from the destination node, the unit once again converts it internally to a FINS response and sends it to the application software on WindowsNT.

C-mode transmission is possible only if the target node is one of the following:

- C-series PLC: CPU, SYSMAC NET Link Unit
- CV-series PLC (in C-mode): CPU, SYSMAC NET Link Unit
- FinsGateway (SYSMAC NET Unit in C-mode): CPU_UNIT

The following is a table of the FINS commands for which the FinsGateway SYSMAC NET unit provides conversion support.

FINS Commands for CPU (PLC)

Command Code		Name	Limitations
01	01	Memory Area Read	The following variables are not supported: ● Transition: 03 ● Step: 04 ● Manual On/Off area: 05, 85 ● Expand DM area: 90~97 ● Expand DM current bank: 98 ● Action: 1B
	02	Memory Area Write	The following variables are not supported: ● Step: 84 ● Manual On/Off area: 05, 85 ● Expand DM area: 90~97 ● Expand DM current bank: 98 ● Register: 9C ● Insert: DD
	03	Memory Area Fill	The following variables are not supported: ● Expand DM area: 90~97 ● Expand DM current bank: 98 ● Register: 9C ● Insert: DD
04	01	Run(start operation)	
	02	Stop(stop operation)	
05	01	Controller Data Read	Only terminal information can be obtained
07	02	Clock Write	
08	01	Internode Echo Test	
21	01	Error Clear	Cannot select error
23	01	Forced Set/Reset	Manual operation can only be specified per connection
	02	Forced Set/Reset Cancel	

FINS Commands for SYSMAC NET Link Unit (PLC)

Command Code		Name
04	01	Run (start data link)
	02	Stop (stop data link)
05	01	Controller Data Read
08	01	Internode Echo Test
26	01	Name Set
	02	Name Delete
	03	Name Read

FINS Commands for CPU_UNIT (FinsGateway)

Command Code		Name	Limitations
01	01	Memory Area Read	
	02	Memory Area Write	
	03	Memory Area Fill	
08	01	Internode Echo Test	

7.2 C-mode Messages (without Conversion)

The application software can also send C-mode messages directly to another node in C-mode . The application software must specify FinsGateway to send C-mode messages. For details on how to send in other protocols (not FINS), refer to the following.

7.2.1 Programming

Sending raw C-mode commands and FINS only differs in the process of creating the header part of the message. The rest of the send procedure is the same. (The API to send a message is the same). These operations use the same FINS API as described previously in this manual.

C-mode Message Headers

You can make a C-mode header by adding the following information to original FINS header.

- Protocol type
- Local source address
- Local destination address

Local Source Address specifies the unit address within the application program. The application program sends C-mode messages. Local Destination Address, the unit of SYSMAC NET unit. The SYSMAC NET unit sends these application C-mode messages to destination.

The procedure to add this information is as follows: The procedure to make a C-mode header in is sown in the following figure.

(1) Make Original FINS header using **FinsHead_compose** function.

(2) Setting of GWPCI header in FINSHEAD structure body.

(2-1) Setting of protocol type to send message. SYSMAC NET C-mode protocol number is defined by C_PROTOCOL_SYSNET_CMODE.

(2-2) Setting of local source unit address (LocalSU)

Get local source unit address using **Fins_getTaddress** function.

(2-3) Setting of local destination unit address(LocalDU)

(2-3-1) Obtain active units list from FinsGateway using **Fins_getConnectUnits** function.

(2-3-2) Search the units list for SYSMAC NET unit using pre-definition unit name ,

' FG_UNITNAME_SYSNET'

(3) Send C-mode commands to SYSMAC series PLCs.

```
int   SysnetCmodeHead_compose(
            HNET            handle,     // network handle
            LPFINSHEAD      lpHead,     // FINS header
            BYTE            icfBits,    // the least significant bit of ICF
            LPFINSADDR      lpAddr,     // FINS transport address of   send destination
            int             input_nSid)// SID
{
            int             sid;                // SID
            FINSADDR        srcAddr;            // source address
            CONNECTINFO     connectInfo[16];    // array of structure body of
                                                //         connected unit information
            int             cbUnits;            // number of units which can be stored
            int             cbConnectUnits;     // number of units which
                                                //         can be currently used
            int             ret;                // return value of FINS API

            /*
             * (1) Make Original FINS header
             */
            sid = FinsHead_compose(lpHead, 0, lpAddr, -1);
            if (sid == -1)
                        return -1;

            /*
             * (2) Setting of GWPCI header
             */
            // (2-1)Setting of protocol type
            lpHead->gwPci.usProtoType = C_PROTOCOL_SYSNET_CMODE;

            // (2-2) Setting of local source unit address (LocalSU)
            ret = Fins_getTaddress(handle, &srcAddr);
            if (ret == C_RETURN_CODE_FAILED)
                        return -1;
            lpHead->gwPci.byLocalSU = srcAddr.byUnitAddr;

            // (2-3) Setting of local destination unit address(LocalDU)
            // (2-3-1)
            cbUnits = sizeof(connectInfo)/sizeof(CONNECTINFO);
            cbConnectUnits = Fins_getConnectUnits(handle, connectInfo, cbUnits);
            if (cbConnectUnits == C_RETURN_CODE_FAILED)
                        return -1;
```

```
        // (2-3-2) Looking for SYSMAC NET unit using "FG_UNITNAME_SYSNET"
        while (cbConnectUnits--) {
                if (strcmp(connectInfo[cbConnectUnits-1].szUnitName,
                                FG_UNITNAME_SYSNET) == 0) {
                        lpHead->gwPci.byLocalDU =
                                connectInfo[cbConnectUnits-1].byUnitAddr;
                        return sid;
                }
        }

        //Error occurred: Not found SYSMAC NET unit
        Fins_setLastError(GW_ERROR_INVALID_FINS_HEADER);
        return -1;
}
```

Figure-1: SysnetCmodeHead_compose function

Using the SysnetCmodeHead_compose API

Next, this section explains how to use the procedure SysnetCmodeHead_compose that makes C-mode header. The usage is completely same as FinsHead_compose function. The procedure to send C-mode commands is as follows:

(1) Make C-mode header for SYSMAC NET using above new procedure.
(2) Make C-mode command for SYSMAC C-series.
(3) Send C-mode message to PLC on SYSMAC NET.

Finally, the following figure shows an example of code using the API.

```
HNEThandle;
FINSHEAD finsHead;
FINSADDR finsAddr;
BYTE buffer[100];
DWORD           dwSize;

// (a) make C-mode header for SYSMAC NET.
sid = SysnetCmodeHead_compose(handle, &finsHead, 0, &finsAddr, -1);

// (b) make C-mode command for SYSMAC C series.
buffer[0] =  …
buffer[1] =  …

// (c) send C-mode message to PLC on SYSMAC NET.
sendByte = Fins_sendData( handleFins, &finsHead, buffer, size );
```

Figure-2: Example code of calling side