## Contents

1. **INTRODUCTION**
   1.1 FINS API ........................................................................................................... 1
   1.2 Operating Environment .................................................................................... 1
   1.3 Structure ............................................................................................................ 1

2. **USING FINS GATEWAY**
   2.1 Sending a FINS Command ............................................................................... 5
   2.2 Returning a FINS Response .............................................................................. 5
   2.3 Receiving FINS Commands/Responses .............................................................. 6
   2.4 Inter-process Communication ......................................................................... 7
   2.5 Ending Communication .................................................................................... 7

3. **ERROR EVENTS**
   3.1 Function Call Errors ....................................................................................... 9
   3.2 Cause and Action of Error Events .................................................................. 10

4. **FINS API REFERENCE**
   4.1 Summary .......................................................................................................... 11
   4.2 Fins_requestVersion Function ....................................................................... 12
   4.3 Fins_getVersion Function .............................................................................. 13
   4.4 Fins_new Function .......................................................................................... 14
   4.5 Fins_newReservedUnit Function .................................................................... 15
   4.6 Fins_delete Function ...................................................................................... 16
   4.7 Fins_sendData Function ................................................................................ 17
   4.8 Fins_receiveData Function ............................................................................ 18
   4.9 Fins_setMessageOnArrival Function .............................................................. 19
   4.10 Fins_setThreadMessageOnArrival Function ................................................. 20
   4.11 Fins_clearMessageOnArrival Function ......................................................... 21
   4.12 Fins_getMessageSize Function .................................................................... 22
   4.13 Fins_getTaddress Function .......................................................................... 23
   4.14 Fins_getEventHandle Function ................................................................... 24
   4.15 Fins_getNetworkInfo Function ..................................................................... 25
   4.16 Fins_getConnectUnits Function .................................................................. 26
   4.17 FinsHead-compose Function ....................................................................... 27
   4.18 FinsHead-composeResponse Function ......................................................... 28
   4.19 Fins_getLastErrorMessage Function ......................................................... 29

5. **DATA STRUCTURE**
   5.1 FINS Message .................................................................................................. 31
   5.2 Transmission Header ....................................................................................... 32

6. **SYSMAC NET FUNCTIONS**
   6.1 C-series and CV-series in One System ............................................................ 33

7. **SYSMAC NET C-MODE TRANSMISSION**
   7.1 FINS Messages (with Conversion) ................................................................. 35
   7.2 C-mode Messages (without Conversion) ........................................................ 36

©Copyright OMRON Corporation 1995-1998 All Rights Reserved.
<table>
<thead>
<tr>
<th>Revision code</th>
<th>Date</th>
<th>Revised content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>August 1998</td>
<td>Original production</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>DLL</th>
<th>FinsGW32.dll, FinsMisc.dll</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import library</td>
<td>FinsGW32.lib</td>
</tr>
<tr>
<td>Include files</td>
<td>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.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>00</th>
<th>01 ... 0F</th>
<th>10 ... 2F</th>
<th>30 ... CF</th>
<th>D0 ... EF</th>
<th>F0 ... FF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main units of a personal computer</td>
<td>User applications</td>
<td>Communication units</td>
<td>Not used</td>
<td>User applications (reserved)</td>
<td>Reserved by the system</td>
</tr>
</tbody>
</table>

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**

```
ICF  RSV  GCT  DNA  DA1  DA2  SNA  SA1  SA2  SID  Command code          Text
```

**Header**

**Configuration of send/receive data for response**

```
ICF  RSV  GCT  DNA  DA1  DA2  SNA  SA1  SA2  SID  Command code          End code  Text
```

**Header**

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).*  
©Copyright OMRON Corporation 1995-1998 All Rights Reserved.
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.

<table>
<thead>
<tr>
<th>ICF</th>
<th>RSV</th>
<th>GCT</th>
<th>DNA</th>
<th>DA1</th>
<th>DA2</th>
<th>SNA</th>
<th>SA1</th>
<th>SA2</th>
<th>SID</th>
</tr>
</thead>
</table>

Application messages
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:

<table>
<thead>
<tr>
<th>Receiving method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message driven method</td>
<td>Performs reception processing by an incoming notice of the previously set Windows message.</td>
</tr>
<tr>
<td></td>
<td>The application which received the Windows message executes the Fins_receiveData function.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>Polling method</td>
<td>Periodically checks if any incoming data is received with the timer or at the background (PeekMessage loop) using the Fins_receiveData function (reception standby time = 0).</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>Synchronization method</td>
<td>Blocks all other processing, waiting for reception with the Fins_receiveData function until incoming data is received.</td>
</tr>
<tr>
<td></td>
<td>It is possible to set a reception standby time. Stand by time = TimeLimit (0 to 4294967294 ms) or unlimited.</td>
</tr>
</tbody>
</table>

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.

```c
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

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

†Codes in the above table show values with the upper 2 bytes masked.
### 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>
<thead>
<tr>
<th>Log code</th>
<th>Description</th>
<th>Cause</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0108</td>
<td>Impossible to send as no relevant unit exists.</td>
<td>The unit is not participating in FinsGateway.</td>
<td>Start the application for the unit.</td>
</tr>
<tr>
<td>0x010D</td>
<td>Impossible to send as destination address is missing in routing table.</td>
<td>Setting error of routing table or invalid destination address.</td>
<td>Set the routing table again.</td>
</tr>
<tr>
<td>0x010E</td>
<td>Impossible to send as routing table is not registered yet.</td>
<td>The routing table is not registered.</td>
<td>Register the routing table.</td>
</tr>
<tr>
<td>0x010F</td>
<td>Impossible to send as routing table is abnormal.</td>
<td>Impossible to retrieve destination address.</td>
<td>Register the routing table.</td>
</tr>
<tr>
<td>0x0110</td>
<td>Impossible to send as number of relays exceeds.</td>
<td>Data is transmitted to networks of more than 3 layers.</td>
<td>Check the network configuration.</td>
</tr>
<tr>
<td>0x0117</td>
<td>Internal buffer full.</td>
<td>Load is concentrated to a single unit. There is some application which is not in reception operation.</td>
<td>Distribute the load in the system. Check the reception operation of applications.</td>
</tr>
<tr>
<td>0x0118</td>
<td>Invalid packet discarded.</td>
<td>Header was wrong if the error occurred at response distribution.</td>
<td>Check the send/receive messages of the applications.</td>
</tr>
</tbody>
</table>
4 FINS API Reference

4.1 Summary

The library consists of multiple functions of the following:

<table>
<thead>
<tr>
<th>Version management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fins_requestVersion</td>
</tr>
<tr>
<td>Fins_getVersion</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Basic Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fins_new</td>
</tr>
<tr>
<td>Fins_newReservedUnit</td>
</tr>
<tr>
<td>Fins_delete</td>
</tr>
<tr>
<td>Fins_sendData</td>
</tr>
<tr>
<td>Fins_receiveData</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Setting of message-driven type of reception</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fins_setMessageOnArrival</td>
</tr>
<tr>
<td>Fins_setThreadMessageOnArrival</td>
</tr>
<tr>
<td>Fins_clearMessageOnArrival</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Acquisition operation of internal information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fins_getMessageSize</td>
</tr>
<tr>
<td>Fins_getEventHandle</td>
</tr>
<tr>
<td>Fins_getTaddress</td>
</tr>
<tr>
<td>Fins_getNetworkInfo</td>
</tr>
<tr>
<td>Fins_getConnectUnits</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operation to manipulate send/receive data</th>
</tr>
</thead>
<tbody>
<tr>
<td>FinsHead_compose</td>
</tr>
<tr>
<td>FinsHead_composeResponse</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Miscellaneous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fins_getLastErrorMessage</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>byMajor</td>
<td>The major version no. FinsGW.h(FinsHead.h) is defined with the CURRENT_MAJOR_VERSION as the major version at the release.</td>
</tr>
<tr>
<td>byMinor</td>
<td>The minor version no. FinsGW.h(FinsHead.h) is defined with the CURRENT_MINOR_VERSION as the minor version at the release.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>byUnitAddr</td>
<td>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.</td>
</tr>
<tr>
<td>lpszUnitName</td>
<td>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.</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpszUnitName</td>
<td>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.</td>
</tr>
</tbody>
</table>

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.

```c
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.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>hNet</td>
<td>The network handle generated by the Fins_new or Fins_newReservedUnit function.</td>
</tr>
</tbody>
</table>

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

```c
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.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>hNet</td>
<td>Specifies the network handle generated by the Fins_new or Fins_newReservedUnit function.</td>
</tr>
<tr>
<td>lpHead</td>
<td>Header section (protocol control information)</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>lpMessage</td>
<td>The pointer of the user buffer which stores the FINS message (from MRC to the end).</td>
</tr>
<tr>
<td>dwSize</td>
<td>The size (in bytes) of the FINS message (from MRC to the end).</td>
</tr>
</tbody>
</table>

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

```c
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.

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

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.

```c
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.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>hNet</td>
<td>The network handle generated by the Fins_new or Fins_newReservedUnit function.</td>
</tr>
<tr>
<td>hWnd</td>
<td>The window handle of the application to notify arrival of FINS message.</td>
</tr>
<tr>
<td>uMsg</td>
<td>Specifies the message type. Use a value between WM_USER to 0x7FFF.</td>
</tr>
</tbody>
</table>

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.

```c
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.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>hNet</td>
<td>The network handle generated by the Fins_new or Fins_newReservedUnit function.</td>
</tr>
<tr>
<td>dwThreadId</td>
<td>The ID of the thread to notify arrival of FINS message.</td>
</tr>
<tr>
<td>uMsg</td>
<td>Specifies the message type. Use a value between WM_USER to 0x7FFF.</td>
</tr>
</tbody>
</table>

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.

```c
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`.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>hNet</code></td>
<td>The network handle generated by the <code>Fins_new</code> or <code>Fins_newReservedUnit</code> function.</td>
</tr>
</tbody>
</table>

**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.

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

Description

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>hNet</td>
<td>The network handle generated by the Fins_new or Fins_newReservedUnit function.</td>
</tr>
</tbody>
</table>

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.

```c
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.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>hNet</td>
<td>The network handle generated by the Fins_new or Fins_newReservedUnit function.</td>
</tr>
<tr>
<td>lpFinsTAddress</td>
<td>The pointer of the structure body to store the FINS transport address.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>hNet</td>
<td>The network handle generated by the Fins_new or Fins_newReservedUnit function.</td>
</tr>
</tbody>
</table>

**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.

```c
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.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>hNet</td>
<td>The network handle generated by the Fins_new or Fins_newReservedUnit function.</td>
</tr>
<tr>
<td>pNetInfo</td>
<td>The pointer of the structure body to store the network information.</td>
</tr>
<tr>
<td>cbNetworks</td>
<td>Number of networks</td>
</tr>
</tbody>
</table>

Structure member Explanation

<table>
<thead>
<tr>
<th>Member</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>szName</td>
<td>Network name (max 15 characters + NULL terminated)</td>
</tr>
<tr>
<td>transport</td>
<td>FINS transport address</td>
</tr>
<tr>
<td>dwMessageSize</td>
<td>Maximum message size (for MRC b to the end of message)</td>
</tr>
</tbody>
</table>

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.

```c
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.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>hNet</td>
<td>The network handle generated by the Fins_new or Fins_newReservedUnit function.</td>
</tr>
<tr>
<td>pConnectInfo</td>
<td>The pointer of the structure body to store the unit information.</td>
</tr>
</tbody>
</table>

The structure for the information of a connected unit

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

<table>
<thead>
<tr>
<th>Structure member</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>byUnitAddr</td>
<td>Unit number</td>
</tr>
<tr>
<td>szUnitName</td>
<td>Unit name (max 15 characters + NULL terminated)</td>
</tr>
</tbody>
</table>

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

©Copyright OMRON Corporation 1995-1998 All Rights Reserved.
4.17 FinsHead_compose Function

Function
Prepares the header of the FINS command.

```c
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.

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

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.

```c
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.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpHeadR</td>
<td>The header of response protocol.</td>
</tr>
<tr>
<td>lpHeadC</td>
<td>The header of command protocol.</td>
</tr>
</tbody>
</table>

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_getLastErrorErrorMessage Function

Function

Gets an error message associated with FINS API error code.

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

Description

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>dwCode</td>
<td>Error code of detailed error information gotten by the GetLastError function</td>
</tr>
<tr>
<td>lpszErrorMessage</td>
<td>Pointer to the start of a character-string array to store error messages.</td>
</tr>
<tr>
<td>dwSize</td>
<td>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.</td>
</tr>
</tbody>
</table>

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;

<table>
<thead>
<tr>
<th>Structure member</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>byNetAddr</td>
<td>Network number</td>
</tr>
<tr>
<td>byNodeAddr</td>
<td>Node number</td>
</tr>
<tr>
<td>byUnitAddr</td>
<td>Unit number</td>
</tr>
</tbody>
</table>

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;

<table>
<thead>
<tr>
<th>Structure member</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICF</td>
<td>Information control field</td>
</tr>
<tr>
<td>RSV</td>
<td>System reserved area</td>
</tr>
<tr>
<td>GCT</td>
<td>Number of allowed gateway (Gateway Count)</td>
</tr>
<tr>
<td>DNA</td>
<td>Destination network number</td>
</tr>
<tr>
<td>DA1</td>
<td>Destination node number</td>
</tr>
<tr>
<td>DA2</td>
<td>Destination unit number</td>
</tr>
<tr>
<td>SNA</td>
<td>Source network number</td>
</tr>
<tr>
<td>SA1</td>
<td>Source node number</td>
</tr>
<tr>
<td>SA2</td>
<td>Source unit number</td>
</tr>
<tr>
<td>SID</td>
<td>Service ID</td>
</tr>
</tbody>
</table>
5.2 Transmission Header

Header Definition (GATEWAY PCI + FINS PCI)

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

<table>
<thead>
<tr>
<th>Structure member</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>gwPci</td>
<td>GATEWAY-PCI: Internal information for FinsGateway</td>
</tr>
<tr>
<td>finsPci</td>
<td>FINS-PCI: FINS protocol header</td>
</tr>
</tbody>
</table>
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.

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

©Copyright OMRON Corporation 1995-1998 All Rights Reserved.
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.

---

©Copyright OMRON Corporation 1995-1998 All Rights Reserved.
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_getAddress 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.

```c
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
    BYTE input_nSid)  // SID
{
    int sid;            // SID
    FINSADDR srcAddr;   // source address
    CONNECTINFO connectInfo[16]; // array of structure body of connected unit information
    int cbUnits;        // number units which can be stored
    int cbConnectUnits; // number 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_getAddress(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;
}
```

©Copyright OMRON Corporation 1995-1998 All Rights Reserved.
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.

```c
HNET handle;
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