

# Device/PLC Connection Manuals

---



## About the Device/PLC Connection Manuals

Prior to reading these manuals and setting up your device, be sure to read the "Important: Prior to reading the Device/PLC Connection manual" information. Also, be sure to download the "Preface for Trademark Rights, List of Units Supported, How to Read Manuals and Documentation Conventions" PDF file. Furthermore, be sure to keep all manual-related data in a safe, easy-to-find location.

# Chapter 1: Memory Link Method

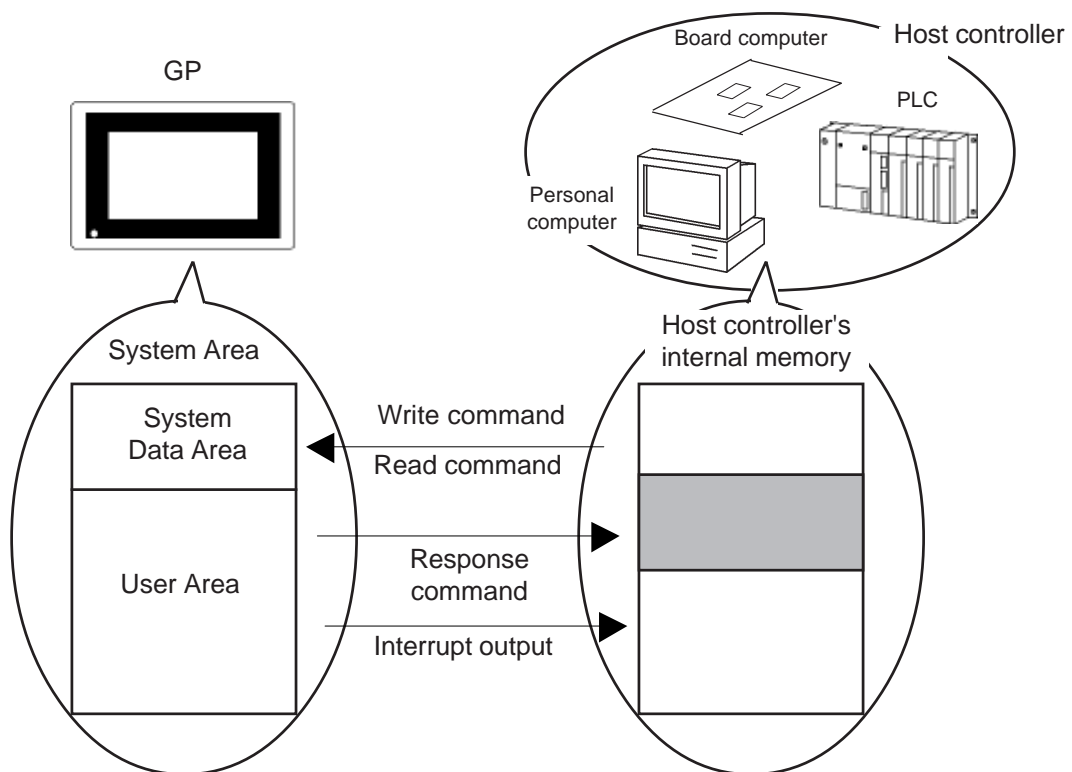
Please read this chapter when using the GP series' Memory Link Method.

This chapter describes 1:1 (one to one) Ethernet communication between a GP unit and a device with no specific protocol (e.g. a personal computer, or a one-board microcomputer).

## 1-1 Memory Link Method

With the Memory Link transmission method, all data transmission between the GP and the host controller is performed according to the host controller's ladder logic program. The GP displays screens according to the display data transferred by the host controller's Write commands. Also, the GP sends its stored data to the host controller according to the host controller's Read commands. Thus, the host controller controls all communications with the GP unit.

All data transferred between the GP and the host controller is control/stored via the GP's designated memory area (System Data Area).



## 1-1-1 System Area

Data transmission between the GP unit and a host controller is performed using a pre-defined System Area within the GP unit. The GP unit then displays screen images according to the data written in this System Area.

The System Area capacity is 8192 words <sup>\*1</sup> and consists of the following areas:

0	System Data Area
: 19	
20	Reading Area
: :	
2032	User Area
: :	
2047	Special Relays
2048	
: 2095	Reserved
2096	
: :	User Area
8191 <sup>*1</sup>	
8192	Reserved
: 8999	
9000	Extended System Data Area <sup>*3</sup>
: :	
: :	
9999	

### ◆ System Data Area

Data required for GP unit operations (e.g. GP panel screen control data, error information, etc.) is written into the System Area. The contents of the written data depends on the designation used for each address.

**Reference** *1-1-2 System Data Area Contents and Setting Range*

### ◆ User Area

The User Area is used to receive data from the host controller, and to send GP data to the host controller.

To perform data transmission, the host controller first specifies the address of the GP unit to which the data will be written and then creates the writing program. The GP unit sets up the Parts and Tags to display the data that has been written to the specified address. To read the data specified by the K-Tag (numeric key input) and T-Tag (touch panel input), the host computer must have a program to read data from the GP unit.

<sup>\*1</sup> Except for GP2000 Series units, 4096 words (LS0 to LS4095) can be used.

<sup>\*2</sup> Can be used for GP2000 Series only.

■ Special Relays

Special Relays are used to store various status data in the GP.



- When T-Tag data is written into address 13 of the System Data area, the interrupt output is activated. If the host controller receives a 1-byte interrupt output (e.g. by using the BASIC programming language's INPUT\$ command) and this interrupt output is used to execute a jump command to each sub-routine, the program can be simplified.



- To specify an address via a specific bit, assign a bit position (00 to 15) after the word address.  
 <Example> To specify bit "02" for address 20 of the User Area:

[2002]  
 Word address —┐└ Bit position

■ Extended System Area

This area is used for a specific feature. It can be used for GP2000 Series only. For details of addresses, see reference of each feature.

9000 : : 9099 9100 : 9199 9200 : 9210 9211 : : : : : 9299 9300 : : 9309 9310 : : : 9999	Trend Graph Previous Data Display function  Script-processing Area  CSV Data Transfer function  Reserved  Security Feature  Reserved	▼ <b>Reference</b> ▲ <i>Tag Reference Manual</i> 2.31.6 Principles of Historical Function  ▼ <b>Reference</b> ▲ <i>Tag Reference Manual</i> 3.1.3 Using D-Script, 3.2.3 Setting Parameters  ▼ <b>Reference</b> ▲ <i>Tag Reference Manual</i> 4.4.6 Automatic Transfer Operation on the GP  ▼ <b>Reference</b> ▲ <i>Operation Manual</i> 4.7 Security Feature
--	---	--

◆ Reserved

Please do not use this area. It is only for GP unit's internal use. If you use this area, the GP unit will not operate properly.

## 1-1-2

## System Data Area Contents and Setting Range

The following data is written in each address of the System Data area:



**When you wish to turn the GP unit's display OFF, use the Screen Display ON/OFF bit. Do not use the Control area's Backlight OFF bit. Be aware that this feature's system Data Area settings and range used during Memory Link Communication will differ from the settings used with Direct Access Communication.**

Address	Detail	Function	Bit	Particulars
1	Status *11		0, 1	Reserved
			2	Now Printing *1
			3	Writes a set value *2
			4 ~ 7	Reserved
			8	K-tag entry error *3
			9	Display 0:ON, 1:OFF *4
			10	Backlight burnout detection *5
			11	Touch-Panel Input Error *6
			12 ~ 15	Reserved
3	Error Status Each bit changes according to the GP error function. When an error occurs, the corresponding bit will turn on. * A bit that has turned on remains on until the power is turned off and back on, or until RUN mode is re-entered from OFFLINE mode or details and the handling process about the Error Status contents, refer to the Section 1-1-4.		0, 1	Unused
			2	System ROM/RAM
			3	Screen Memory Checksum
			4	SIO Framing
			5	SIO Parity
			6	SIO Overrun
			7, 8	Unused
			9	Initialization of Internal Memory Checksum Necessary
			10	Timer Clock Error
			11 ~ 15	Unused
4	Clock Data (Year)	"Year / Month / Day / Hour / Minute" Data is stored in BCD's 2digits. (E.g.) 98/02/01 17:15	0 ~ 7	Stores the last 2 digits of the Calendar year as 2 BCD digits
5	Clock Data (Month)		8 ~ 15	Unused
			0 ~ 7	Stores 01 to 12 (Month) as 2 BCD digits
6	Clock Data (Day)		8 ~ 15	Unused
			0 ~ 7	Stores 00 to 31 (Day) as 2 BCD digits
7	Clock Data (Hour)		8 ~ 15	Unused
			0 ~ 7	Stores 00 to 23 (Hour) as 2 BCD digits
8	Clock Data (Minute)		8 ~ 15	Unused
		0 ~ 7	Stores 00 to 59 (Minute) as 2 BCD digits	
10	Interrupt Output (Touch OFF)*16	If you Write in word data, the bottom 8 bits will be output as an interuppt code after touching OFF.However FFh will not be output.		
11	Control *12		0	Backlight *7
			1	Buzzer ON *8
			2	Starts Printing
			3	Reserved
			4	Buzzer *8 --- 0:enabled 1: disabled
			5	AUX Output *8 --- 0:enabled 1: disabled
			6	Interrupt Output when touching panel to turn the display ON. (Interrupt Code:FFh) 0: Disabled 1: Enabled *16
			7	Reserved
			8	VGA display *9 --- 0: Disabled 1: Enabled
			9, 10	Reserved
			11	Hard copy output *15 --- 0: Enabled 1: Disabled
			12 ~ 15	Reserved

Address	Detail	Function	Bit	Particulars
12	Screen Display <sup>**14</sup> ON/OFF	FFFFh : Screen clears almost immediately 0h: Screen turns ON		
13	Interrupt Output <sup>**15</sup>	Using a Touch Tag or other method to write absolute value data from GP causes an output of the interrupt code using the contents of the bottom 8 bits ( Will not output FFh)		
15	Screen Display No. <sup>**17</sup>	Write the Screen No. in binary to change the screen display	0 ~ 14	Screen change number, 1 to 8999.( 1 to 1999 when using BCD input)
			15	Forced Screen Change 0 : normal, 1: Forced Screen Change
16	Window Control <sup>**16</sup>		0	Display -- 0: OFF 1: ON
			1	Changing the order of window overlapping -- 0: Possible 1: Not Possible
			2 ~ 15	Reserved
17	Window Registration No. <sup>**16</sup>	Global Window registration number selected indirectly (BIN/BCD)		
18	Window Display Position (X-coordinate) <sup>**16</sup>	Global Window display position reached indirectly (BIN/BCD)		
19	Window Display Position (Y-coordinate) <sup>**16</sup>			

## \*1 &lt;Status&gt;

- Monitor, in bit units, only the necessary bits.
- Since reserved bits may be used for GP system maintenance, etc., their ON/OFF status is not defined.

## \*2 &lt;Status-Now Printing&gt;

This bit turns ON during printing. Changing to OFFLINE mode in the middle of printing can cause the print output to become disordered.

## \*3 &lt;Status-Write Setting Value&gt;

Every time a value is written with the K-tag or Keypad Input Display, the bit is reversed.

## \*4 &lt;Status-K-tag Input Error&gt;

If an (input value range) Error has been set for the K-tag data being entered, and a value outside the allowed range is entered, the bit turns ON. If, however, a value is entered that is within the Error range, or if the display screen is changed, this bit will turn OFF.

## \*5 &lt;Display ON/OFF status&gt;

The screen display ON/OFF can be detected from the PLC. This bit will change in the following cases:

- (1) "FFFF" is written to the system data area's screen display ON/OFF bit (LS9 when using link type), to turn the screen display OFF. (Bit 9 = 1)
- (2) After the stand-by time has elapsed, the screen display OFF bit is turned ON automatically. (Bit 9 = 1)
- (3) The screen display OFF status has been changed to the screen display ON status via screen switching, etc. (Bit 9 = 0)
- (4) The screen display ON/OFF status bit will not change via turning ON/OFF the system data area backlight OFF bit (Bit 0).

## \*6 &lt;Backlight Burnout Detection&gt;

The bit turns ON when backlight burnout is detected. This feature is available only on units equipped with a backlight.

## \*7 &lt;Touch-panel input error&gt;

The touch-panel input error bit is turned ON when input in the same position continues for longer than the specified time.

## \*8 &lt;Control&gt;

Be sure to turn all reserved bits OFF since they may be used for GP system maintenance, etc.

**\*9 <Control -Back light>**

With the GP series except GP-477R, GP-470, and GP-870 series units, the backlight turns OFF when this bit is ON(LCD display does not change) and turns ON when the bit is OFF.

When the Control area's Backlight OFF bit turns ON, only the backlight will turn OFF, however, the LCD display will remain ON and all touch switches set up on the display can still be used. Use the Screen display ON/OFF bit to actually turn the screen display OFF.

**\*10 <Control -Buzzer Sound>**

Control Bit 1 (Buzzer On) outputs as shown below.

Buzzer Sound ..... While Control Bit1 is on, the GP internal buzzer is activated.

AUX Output ..... While Control Bit 1 is on, the AUX buzzer output is activated.

**\*11 <Control -Interrupt output when touching the panel to turn the display OFF to ON>**

- Only when the display is turned ON by touching the panel, interrupt output will be operated.
- When using GP-H70, interrupt output will not be operated if the display is turned ON by the Operation Switch on the rear side.

**\*12 <Control -VGA Display>**

When using GP-570VM and GP-870VM, the entire screen becomes a VGA display when this bit is on. Pressing the screen options position during a VGA display turns this function off.

**\*13 <Hard Copy Output>**

Turning ON bit 11( Hard Copy Output ) in the Control Area will cancel the current printing of the display's hard copy.

- After printing is cancelled, bit 11, however, will not turn OFF automatically. Therefore, after checking the Status Area's Now Printing bit, turn off the Control Area's bit 11.
- While bit 11 in the Control area is turned ON, hard copy cannot be created. If you cancel printing before it is completed, printing will stop after the last line data on the panel's current display has been output. Data already input in the printer buffer's memory will not be deleted.

**\*14 <Screen ON/OFF>**

After the System Data Area's "Screen Display ON/OFF" bit is set to turn the display OFF, simply touching the screen will turn the display ON again.

**\*15 <Interrupt Output>**

Do not write control codes 00 to 1F to word addresses 10 and 13. It may terminate data communication.

**\*16 <Window Control/Window Registration/Window Display Position>**

**Reference** For more about windows, refer to GP-PRO/PBIII for Windows Tag Reference Manual. "2.26 U-tag (Window Display)"

**\*17 <Screen Number>**

When using "Screen changes" from the Host device and "Screen Changes" from the touch panel on the D-Script together, screens may not change as you wish. To avoid that, please integrate the both into the "Screen changes" from the Host device. Be sure to send signals to the Host device using [Interrupt Output] of Address 13 from the touch panel on the D-Script and change screens from the Host device.



- Addresses 0, 2, 9, 14 are reserved areas. Do not write data to this area.
- Since addresses 3, 12, 13, 15 are utilized for System Control, displays that depend on tags do not function.
- Since addresses 12, 13, 15 are used to control word units, bit write cannot be performed.
- Writing FFFFh to address 12 causes the screen display to erase within moments. When you wish to erase the screen using the STANDBY MODE TIME entered in GP unit's INITIALIZE setup, write 0000h in address 12.
- Do not write the control code 00-1F in addresses 10 and 13. Data transmission may become impossible.

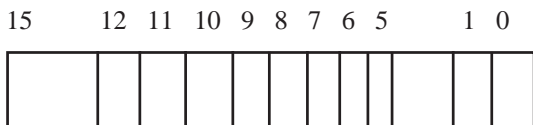
## 1-1-3 Special Relays

2032	Common Relay Information
2033	Base Screen Information
2034	Reserved
2035	1-second Binary Counter
2036	Tag Scan Time
2037	Reserved
2038	Tag Scan Counter
2039	Reserved
:	
:	
2047	

◆ Reserved

The value of the reserved address is not fixed. Do not use these areas.

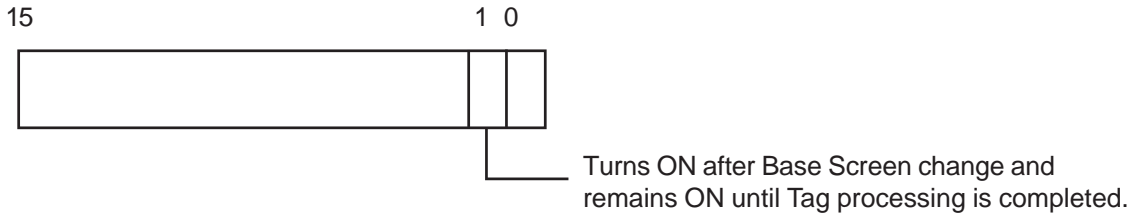
◆ Common Relay Information (2032)



0	Reserved
1	This bit is ON while from the time the screen (Base, Window) is changed until the Tag's processing is finished.
2	Reserved
3	This bit is ON from the time the power is turned ON until the initial screen appears.
4	Normally ON
5	Normally OFF
6	Turns ON when Backup SRAM is erased. (Only with GP units equipped with Backup SRAM.)
7	If a D-script is used, this relay is turned ON when a BCD error occurs. For D-Script information, see the Tag Reference Manual 3.1 D-Script
8	8 If a D-script is used, this relay turns ON when a zero division error occurs.
9	Turns ON when filing data cannot be sent to backup SRAM.
10	Turns ON when data transfer fails after a Control Word Address is used to transfer filing data from the PLC to SRAM. Also, when the Filing Item Display is used to send data from the PLC, only when a Transfer Completion Bit Address is used, and the transfer from PLC to the System Data Area, or PLC to SRAM fails, this bit will turn ON.
11	When filing data is being transferred from SRAM to the LS Area via the Filing Item Display, this bit is ON.
12	When using D-Script, if memcopy () is used to read out dat using the address offset designation and an error occurs, this bit turns ON. If the data read out is completed normally, this bit will turn OFF.
13 to 15	Reserved



◆ **Base Screen Information (2033)**



◆ **1-second Binary Counter (2035)**

This counter is incremented by seconds, immediately after the power switch is turned ON. The count is stored as binary data.

◆ **Tag Scan Time (2036)**

Indicates the time duration required for processing of all specified Tags, from the start of processing the first Tag to the completion of processing the last Tag specified on a screen.

The scan time is stored as binary data (unit: ms). The scan time is updated after completion of all Tag processing. The initial value is "0", and the margin of error is within +/-10 ms.

◆ **Tag Scan Counter (2038)**

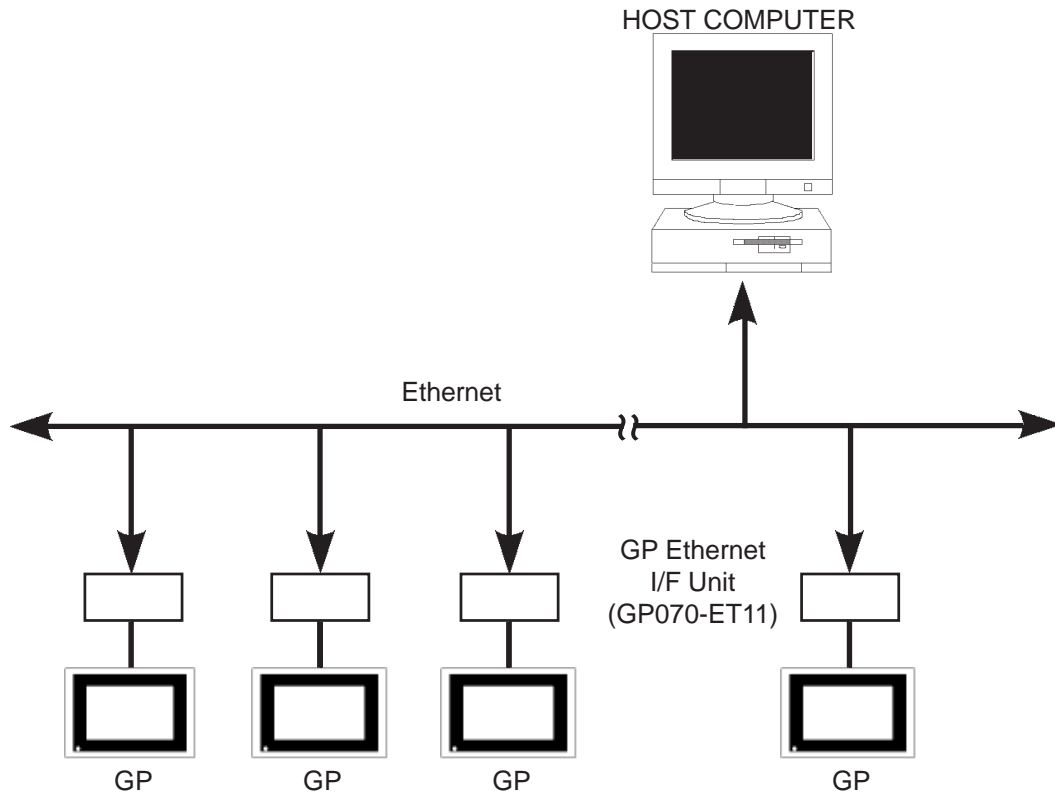
Each time the processing of all specified Tags is completed, the counter is incremented by one. The count is stored as binary data.



*These special relays are not write-protected. Do not turn the special relays ON/OFF using Tags.*

## 1-2 System Configuration

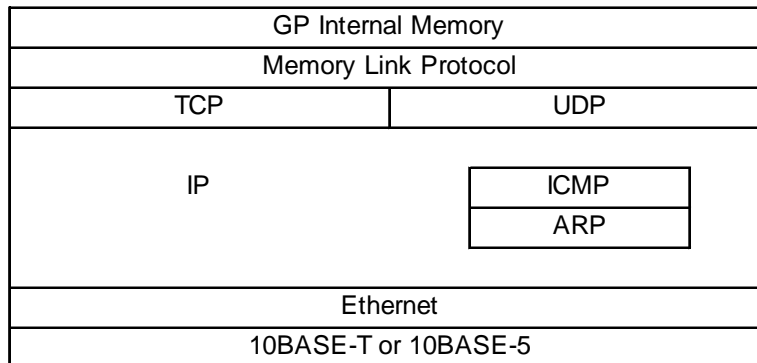
The GP Ethernet I/F Unit allows the GP to be easily connected to an Ethernet network.



- Note:**
- *The number of connectable GP units will vary depending on the type of host computer being used.*
  - *When using GP70/GP77R Series units, the optional Ethernet Expansion Unit is needed.*

## 1-3 Software Structure

The GP Ethernet I/F unit supports both TCP/IP and UDP/IP protocols.



### ■ TCP (Transmission Control Protocol)

The TCP protocol insures data reliability.

- The TCP protocol enables logical link control when a connection is established.
- Only one connection can be established.
- This protocol enhances data reliability through sequence numbers, data re-transmission, and checksum function controls.

### ■ UDP (User Datagram Protocol)

The UDP protocol will not insure data reliability.

If data is not received by the target node, data re-transmission is not performed.

- This protocol enables connection-less data transmission.
- This protocol provides a checksum function to ensure data reliability. However, for the highest level of data reliability, use the TCP protocol.

### ■ IP (Internet Protocol)

- Data is transmitted/received by datagram.

### ■ ARP (Address Resolution Protocol)

- Through broadcast, the ARP protocol uses the IP address to create an Ethernet address.

### ■ ICMP (Internet Control Message Protocol)

- This protocol supports a response to an echo request.

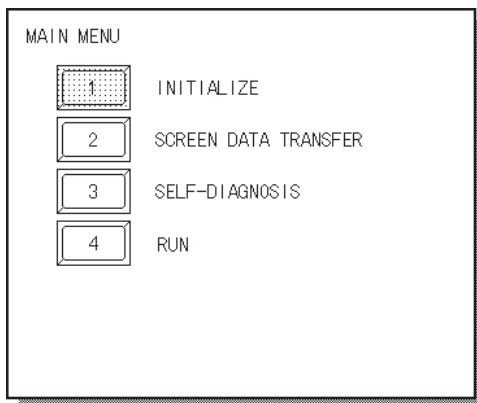
# Chapter 2: GP Setup and Data Communication

This chapter describes how to set up the GP for Ethernet communication, and the basic procedures used for data transmission between the GP and a host device.

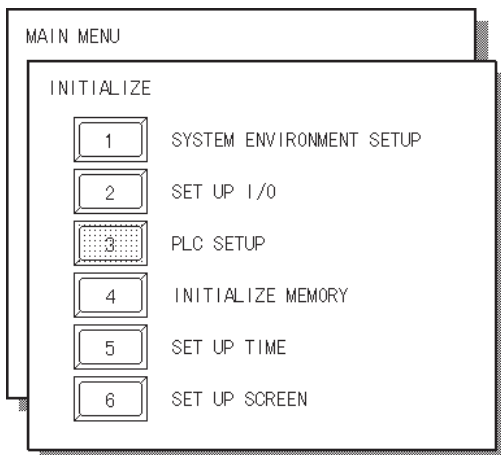
## 2-1 Setup Procedure

Before starting data transmission, you will need to set up the GP's Ethernet information (IP address and port number).

The procedures for setting up the communication protocol in OFFLINE mode are described below.

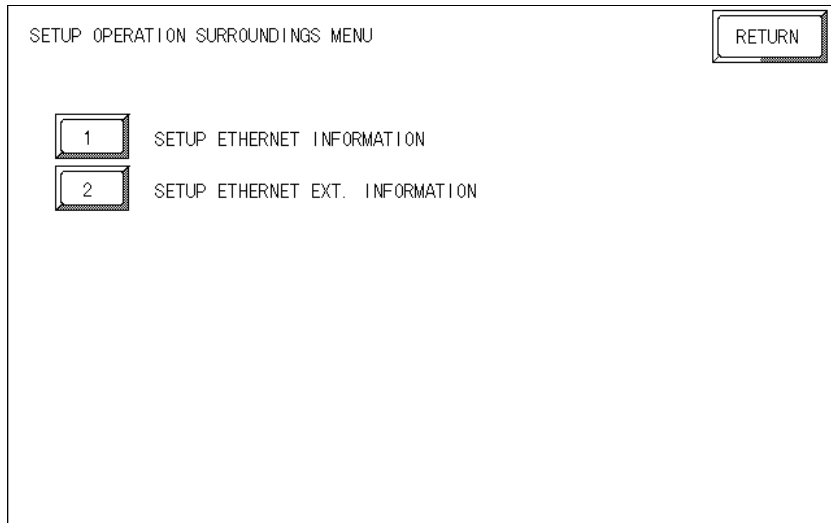


① Select [INITIALIZE].



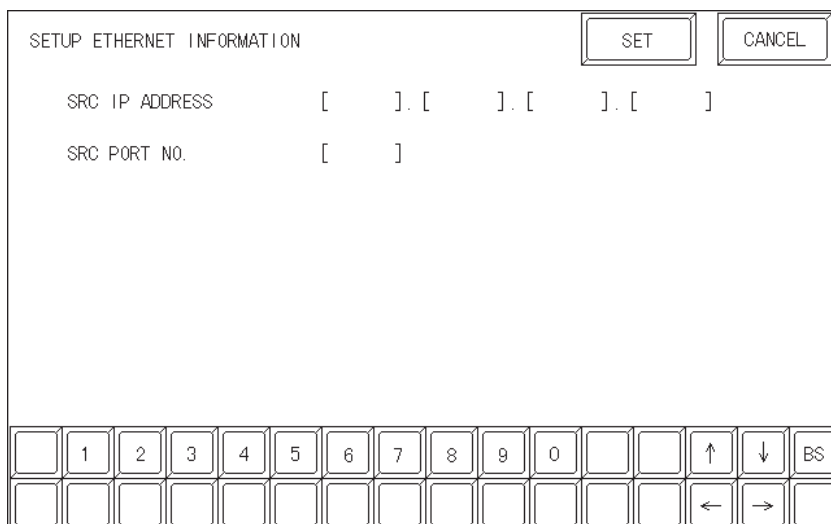
② Select [PLC SETUP].

③ The "SETUP OPERATION SURROUNDINGS MENU" screen will be displayed.



◆ **Ethernet Information Setup**

Select [SETUP ETHERNET INFORMATION], and specify the following parameters:



• **SRC IP Address**

The IP address (32 bits in total) is divided into four blocks (8 bits/block). Enter a decimal number in each group. (Each block is separated with a dot.)

• **SRC Port No.**

Specify the SRC (GP) port number.  
Enter a value between 1024 and 65535.



- *For the SRC IP address and SRC port number, ask your system's network administrator.*
- *Do not assign the same IP address to different GP. Otherwise, a communication error will occur.*
- *Do not use the same port as the 2-Way Driver.  
The 2-Way Driver's default Port numbers are 8000 to 8009.*

## ◆ Extended Ethernet Information Setup

Select [SETUP NETWORK EXT. INFORMATION], and specify the following parameters:

SETUP NETWORK EXT. INFORMATION												SET	CANCEL	
SEND WAIT TIME	[		]	(ms)										
TCP TIMEOUT	[		]	(x 2sec)										
IP ROUTE ADDRESS	[		]	.	[		]	.	[		]	.	[	
SUBNET MASK	[		]	.	[		]	.	[		]	.	[	

	1	2	3	4	5	6	7	8	9	0		↑	↓	BS
												←	→	

- **SEND WAIT TIME (0 to 255)**

Enter a time to wait for command transmission from the GP.  
You can use this function when the communication line is busy.  
If this function is not required, enter "0".

- **TCP TIMEOUT (0 to 65535)**

Enter a TCP timeout value. If the target node returns no response within the specified time, a timeout error occurs. If this parameter is "0", the timeout value is set to "15 seconds" (default setting).

- **IP ROUTE ADDRESS**

Specify a router IP address. (Only one router can be specified.)  
If the router is not used, enter "0" in each block of this parameter.

- **SUBNET MASK**

Specify a subnet mask.  
If the subnet mask is not used, enter "0" in each block of this parameter.



- *After the memory is initialized in the OFF-LINE mode, each parameter may be set to an indefinite value. Be sure to check the parameter settings.*
- *For the router IP address and subnet mask settings, ask your system's network administrator.*

## 2-2 Ethernet Connections

The Ethernet memory link protocol supports two types of connections: TCP/IP and UDP/IP.

These connections have the following characteristics:

- TCP/IP connection is intended for 1:1, not broadcast type communication
- UDP/IP connection will not recognize any response from the GP (e.g. Touch input < ESC I>).

	TCP/IP connection	UDP/IP connection
1:1 transmission	Enabled	Enabled
Broadcast transmission from host computer	Disabled	Enabled
GP Response	Enabled	Disabled

### 2-2-1 Establishing a Connection

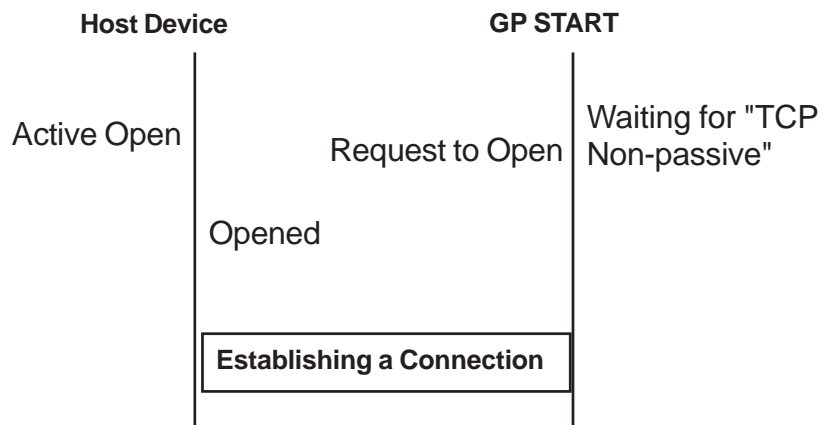
#### ■ TCP Connection

##### ◆ GP

After the GP enters RUN mode, it waits for a Connection Open Request command from the host device (Non-passive).

##### ◆ Host Device (Response)

The host device transmits the Connection Open Request command to the GP (Active Open). When the GP responds to this command, the connection is opened.

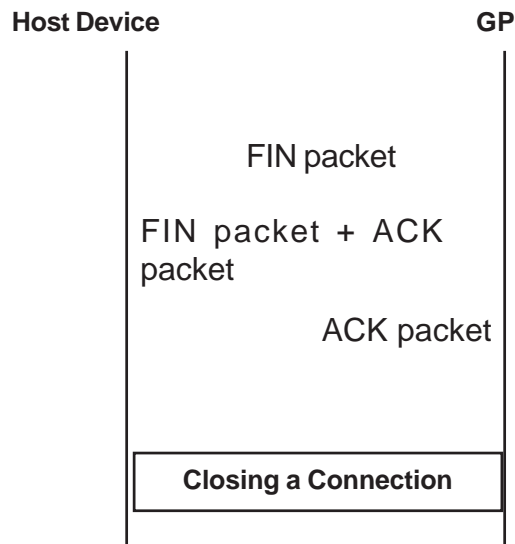


### ■ Closing a TCP Connection

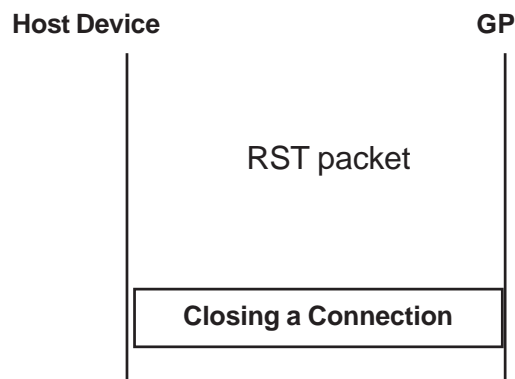
#### ◆ Host Device

When communication between the Host and the GP is completed, or a TCP/IP-level error occurs, the connection can be closed arbitrarily using the following method.

When sending a FIN packet:



When sending an RST packet:



**Note:**

After the command is sent, no matter what condition occurs afterwards, i.e. if the response is delayed or the command is still being sent, the connection will be closed.



---

## ◆ GP Unit

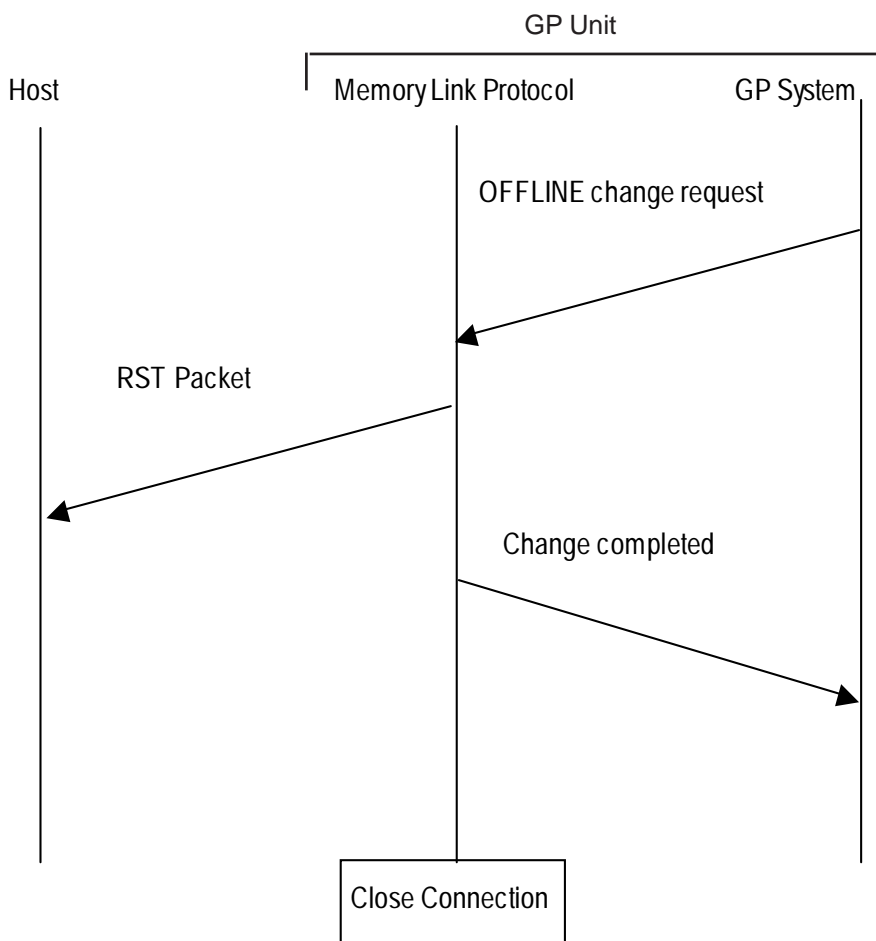
The following explanation describes the GP unit's closing of the connection.

When the GP closes the connection, it changes to OFFLINE mode and waits for reconnection.

- **When the GP has changed to OFFLINE mode**

After changing to OFFLINE mode, the GP can close the connection via sending an RST packet to the Host.

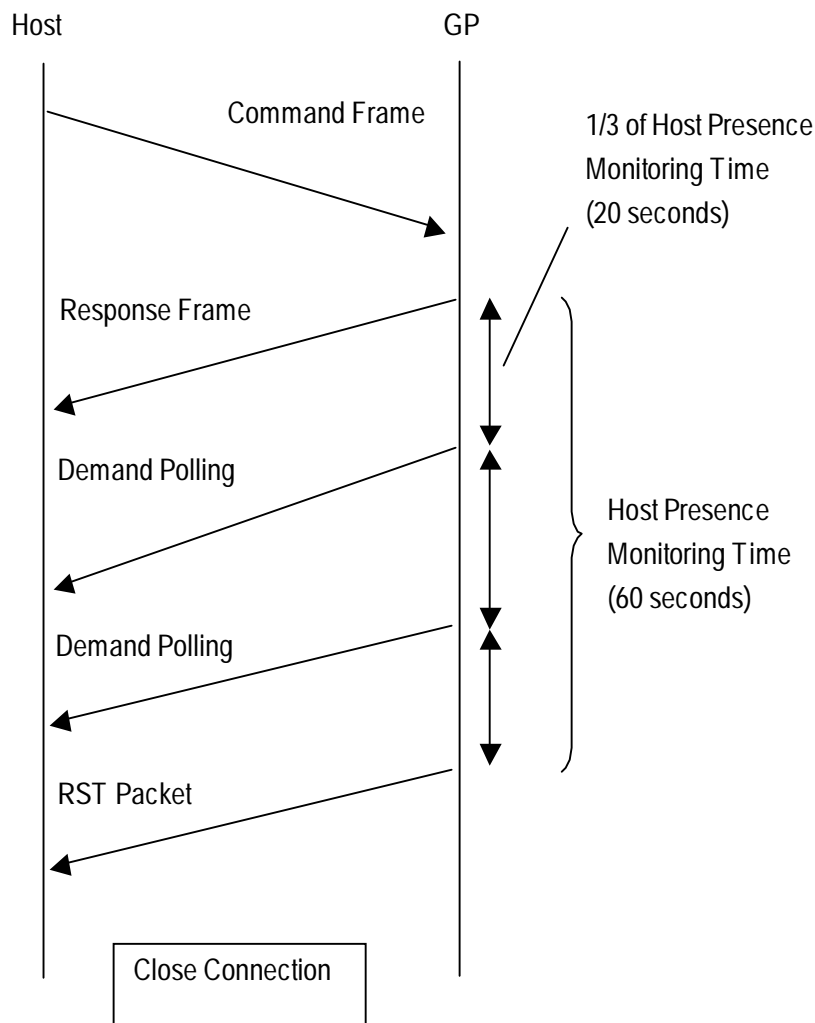
When sending an RST packet:



- **When no response is received from Host after Host presence monitoring time elapses**

When no response is received from the Host after the Host presence monitoring time elapses, an RST packet is sent to close the connection.

However, after one third of the monitoring time elapses, the GP unit will send a Demand Polling request to the Host. After the Host receives this request, be sure to respond by sending a Demand Polling request to the GP. This can be used to reset the Host presence monitoring time.



- **When a TCP/IP level error occurs**

When a TCP/IP level error occurs, the GP will send an RST packet and close the connection.

## ■ Polling Commands

When a periodic request is not received from the Host, the GP unit checks for the presence of a Host by performing Demand Polling. When the Host receives this request, be sure to send a similar Demand Polling request to the GP. After the GP receives this request, it confirms the existence of the Host.

If no response is received from the Host, the GP will close the connection.

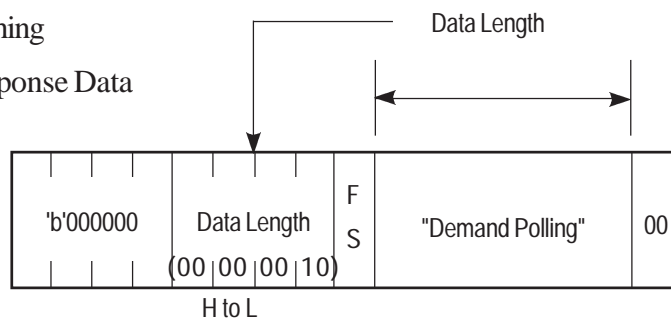
If you wish to use Digital Electronics Corporation APIs for the Host, the response processing of the Demand Polling request will be performed automatically by the API.

### 1 Demand Polling (FS Demand)

The data contents of the Demand Polling request sent from the GP to the Host are as follows:

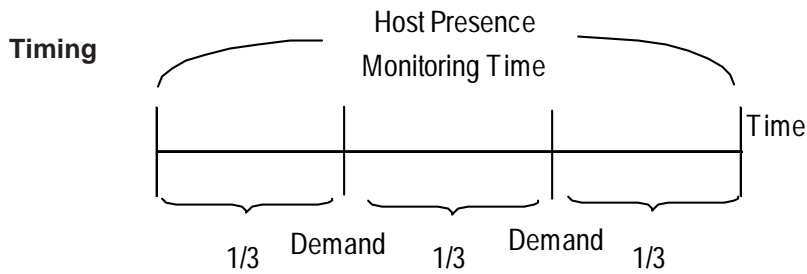
Host: Nothing

GP: Response Data



<Data Name:>

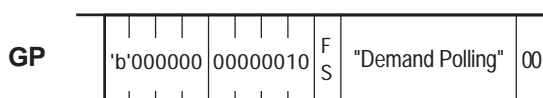
- Data: "DemandPolling"



When the 1/3 of the Host Presence Monitoring Time elapses, a Demand Polling request is sent. This request demands the Host send its own Polling command. This type of request allows the GP to not have to wait for a polling request.

Example:

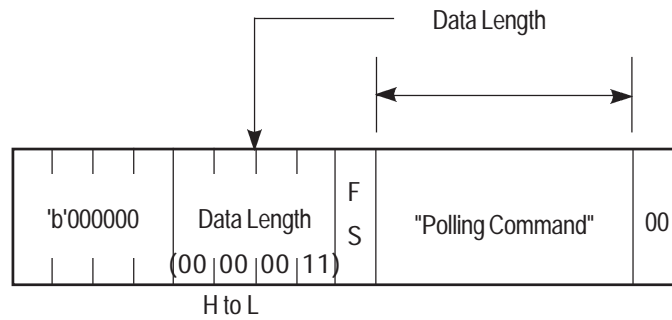
Host



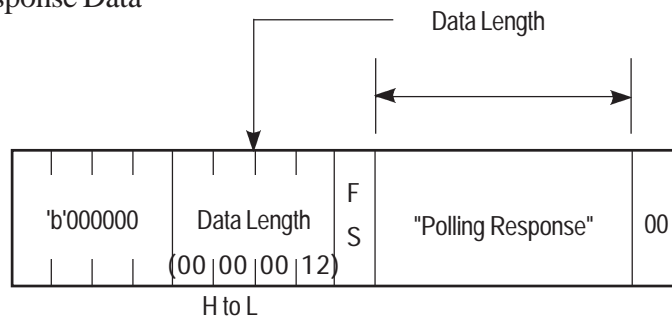
## 2 Polling Command (FS Polling)

The data contents of the Polling Command sent from the Host to the GP are as follows:

Host: Command Data



GP: Response Data

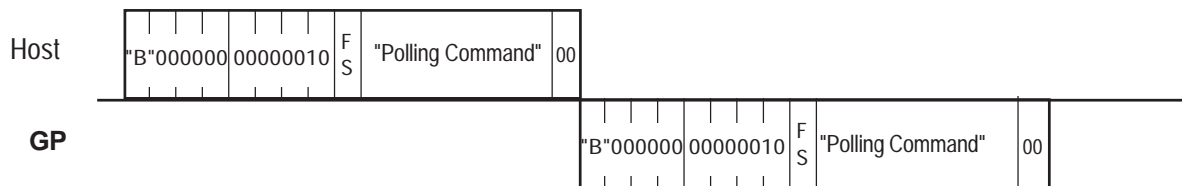


<Data Name:>

- Data: "Polling Command"

Example:

The Host sends this polling command to the GP to inform the GP that the Host is operating normally.

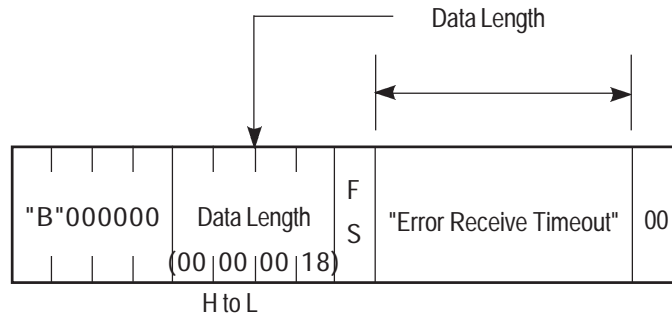


### 3 Error Detection (FS Error)

When a protocol error occurs, this command allows the GP or the Host to output an error notice about the other unit/device. This frame does not require a response.

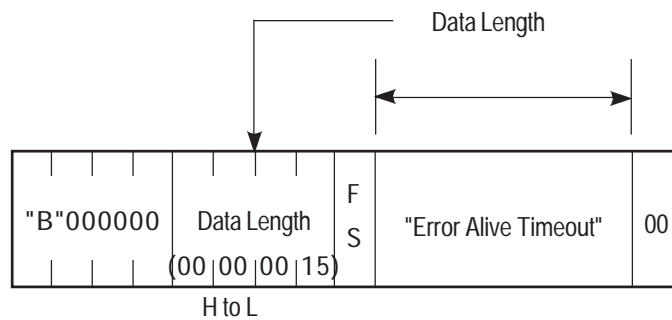
Host: Command Data

- Inter-character timeout error frame



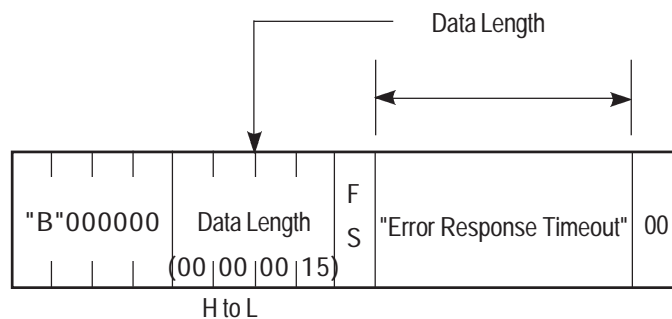
<Data Name:>

- Data: "Error Receive Timeout"
- GP presence monitoring timeout error frame



<Data Name:>

- Data: "Error Alive Timeout"
- Inter-protocol timeout error frame

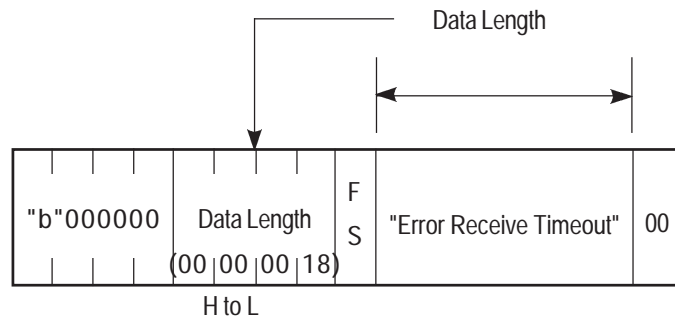


<Data Name:>

- Data: "Error Response Timeout"

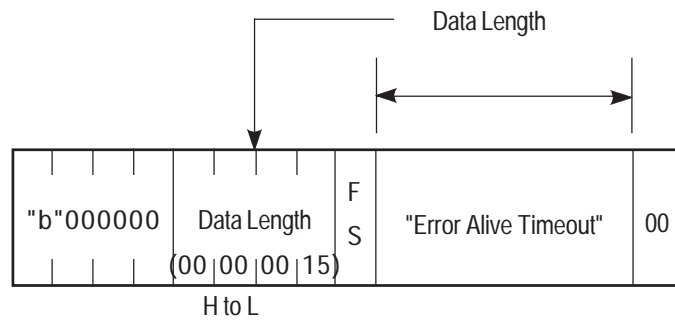
GP: Response Data

- Inter-character timeout error frame



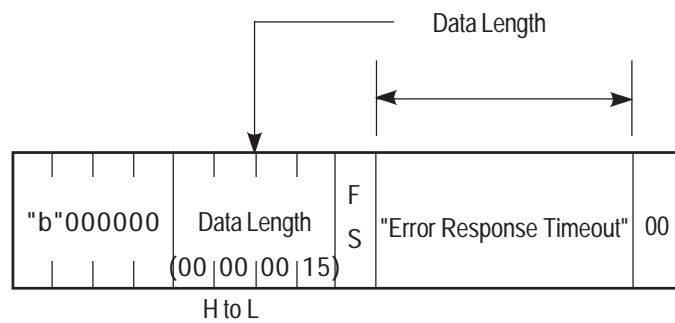
<Data Name:>

- Data: "Error Receive Timeout"
- Host presence monitoring timeout error frame



<Data Name:>

- Data: "Error Alive Timeout"
- Inter-protocol timeout error frame



<Data Name:>

- Data: "Error Response Timeout"

## 2-3 Communication Protocol Control

The commands for data transmission between the GP and host device are set up according to the procedure specified. This section describes the basic commands and the command setup procedure.

### ■ Commands Used in This Manual

The commands used in this manual are described below.

### ◆ Commands

The data transfer commands used are classified into the following groups:

Command Data : Instruction command from host device

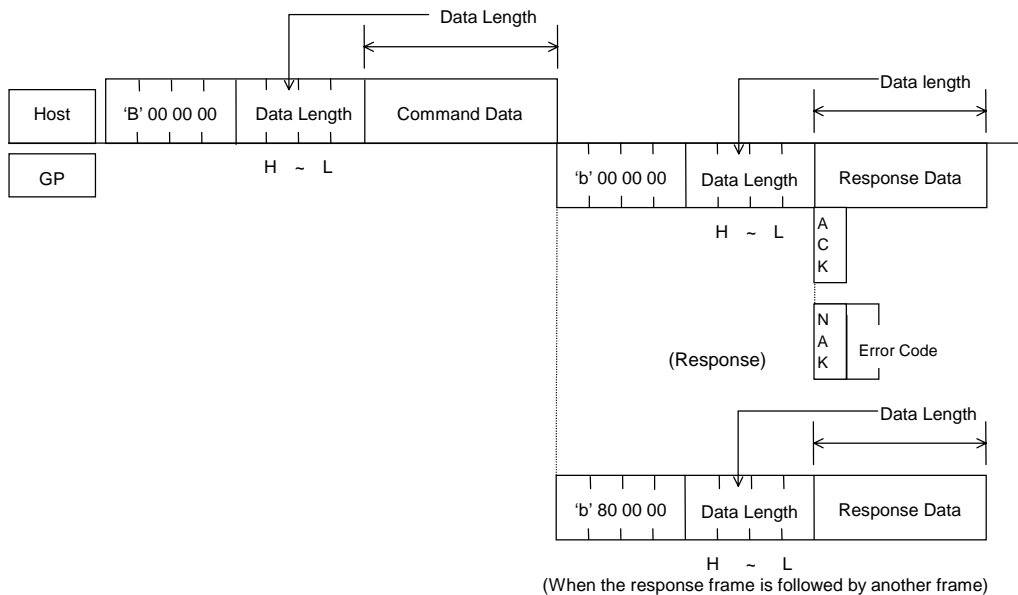
Response Data : GP's response command (to an instruction from host)

Interrupt Output Data : Response command for GP panel touch input

## 2-3-1 Basic Communication Protocol Control

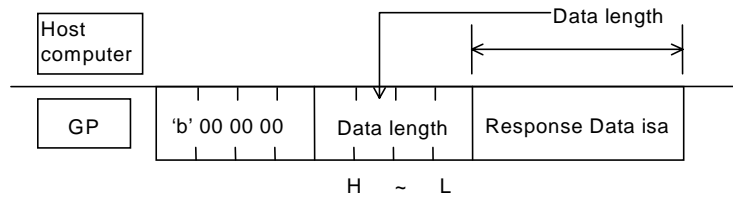
The basic procedure for controlling the communication protocol is shown below:

### ◆ Host to GP Data Transfer



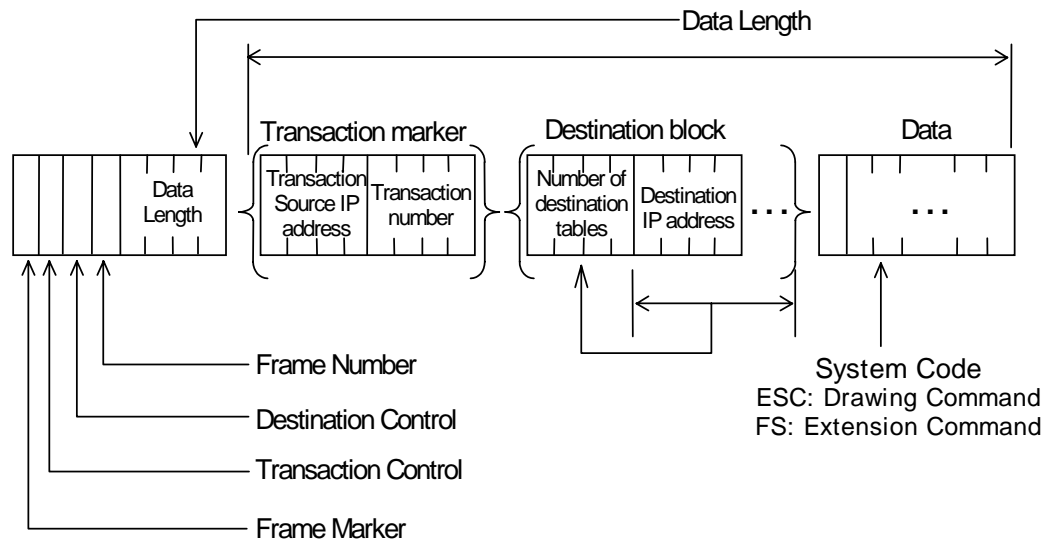
- ① The Command Data stores the data to be transmitted from the host device to the GP.
- ② After the GP analyzes the Command Data, the Response Data area stores the result of "ACK" or "NAK", or no response.

◆ GP to Host Data Transfer



**2-3-2** Frame Format

The memory link LAN frame is structured as follows:



The initial 8 bytes, from frame marker to data length, are provided in all memory link LAN frames.

Therefore, during a frame check, the system checks the initial 8 bytes first, and then checks the subsequent data based on the data length specified in the initial 8 bytes.

■ **Frame Marker (1 byte)**

The frame marker is used to identify the frame type.

‘B’: Binary command frame

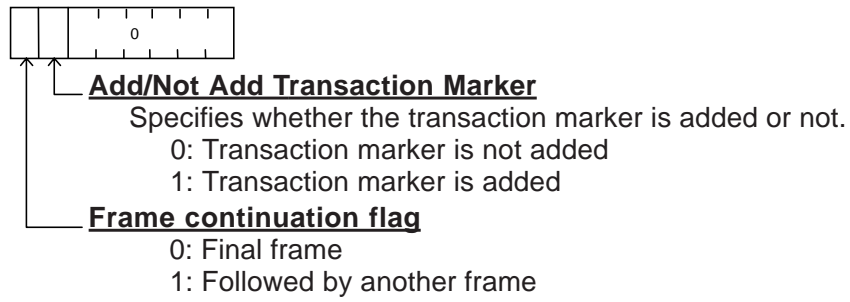
‘b’: Binary response frame



• Only binary frames are supported.

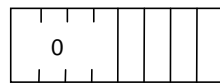


## ■ Transaction Control (1 byte)



During the transmission of large amounts of data, the data will be divided into several frames. The transaction control bit specifies whether it is a divided frame. To indicate the initial and subsequent frames, the control bit is set to "1". The final frame bit is "0".

## ■ Destination Control (1 byte)



### Add/Not Add Destination Block

Specifies whether the destination block is added to the frame or not.

0: Destination block is not added (Destination is not to be checked)

1: Destination block is added

### Use/Not Use Destination Block

Specifies how to use the destination block to determine the target node.

0: Only a node whose IP address is specified in the destination block is treated as the processing target.

1: Only a node whose IP address is not specified in the destination block is treated as the processing target.

This bit is enabled only when the "Add/Not Add Destination Block" control bit is "1".

### Responding Node

When a response is required, this bit specifies whether the target node returns a response or not.

0: All nodes return a response.

1: Only the node specified at the head of the destination block returns a response.

This bit is effective only when the "Add/Not Add Destination Block" and "Use/Not Use Destination Block" control bits are "1" and "0", respectively.

### Response Wait

This bit specifies whether the responding node waits before sending a response.

0: Sends a response immediately, without a wait period.

1: Waits

The GP sends back a response after waiting for a time duration of the "least-significant 7 bits of the SRC IP address x 1 ms".

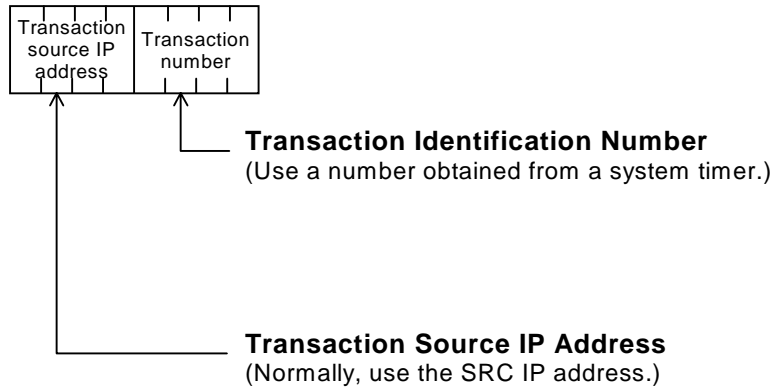
This function prevents several nodes from responding simultaneously.

## ◆ Destination Control Applications

To perform normal 1:1 communication, enter "00h" in the destination control bit.

For "1:n" (multi-link) communication, enter "05h" to request a response from only one target node among an unspecified number of nodes ("n" nodes). To request a response from all the nodes, enter "09h".

■ **Transaction Marker**



◆ **Application of transaction marker**

After receiving a command frame that includes a transaction marker, the GP executes the command (and sends back a response, if necessary). This process is the same as that for a command frame without a transaction marker. Next, the processing result is stored in the GP. When the GP receives the next transaction result request, the GP responds by sending the stored data.

The GP can store up to ten transaction results. If there are ten or more transactions, the existing transactions will be deleted, starting from the oldest one, and the new data will be registered.

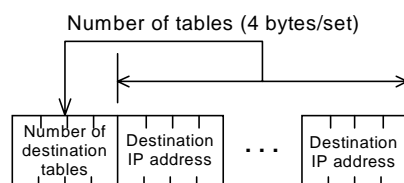
■ **Frame Number**

When a command or response is divided into several frames, serial numbers (0 to 255) are assigned as the frame numbers.

The maximum size of a divided frame is 1 Kbyte.

■ **Destination Block**

A destination block is added when the "Add/Not Add Destination Block" control bit is "1". A destination block is not added when this control bit is "0".



Example of two sets of destination IP addresses ("1.2.3.4" and "1.2.3.5")

0	0	0	2	1	2	3	4	1	2	3	5
---	---	---	---	---	---	---	---	---	---	---	---

# Chapter 3: Transferring Screens

This chapter describes how to transfer screen data, created with the GP-PRO/PB III for Windows program, to a GP connected to Ethernet. Also, how to receive screen data stored in the GP.

## 3-1 Screen Transfer Procedures

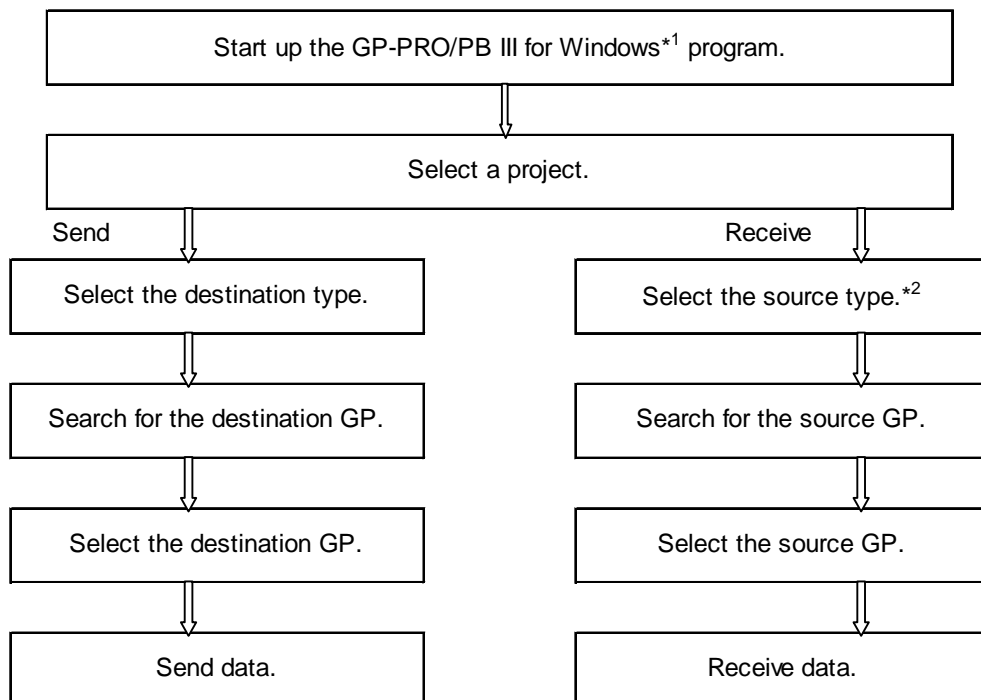
Before attempting to send screen data via an Ethernet network for the first time, the necessary Ethernet protocol must be transferred to each GP through the data transfer cable.

**Reference** To setup your GP, refer to "2-1 Setup Procedures",

**Reference** To setup an Ethernet system, refer to the "GP70 Series GP Ethernet I/F Unit User's Manual".

### 3-1-1 Transfer Operation Flow

The following flowchart shows the basic steps to use when sending "GP-PRO/PB III for Windows" data to a GP connected to Ethernet, and how to receive the data stored in the GP to the GP-PRO/PB III for Windows program.

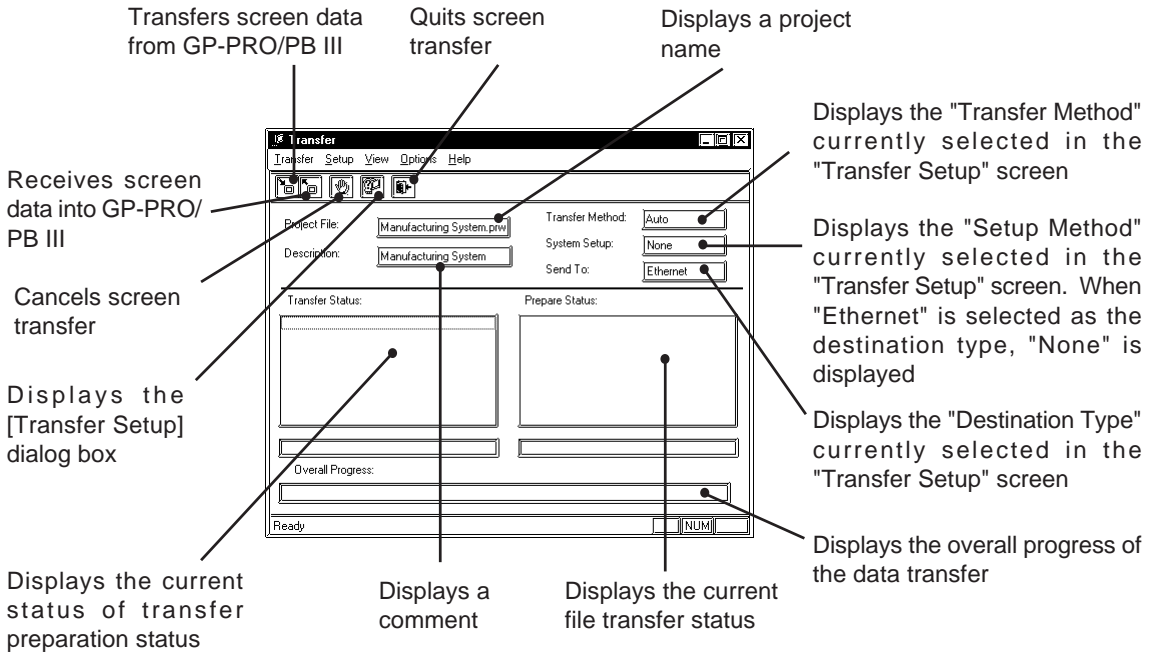


\*1 GP-PRO/PB III for Windows Ver. 2.0 or later version

\*2 During actual operation, select "Destination Type" in the "Transfer Setup" screen.

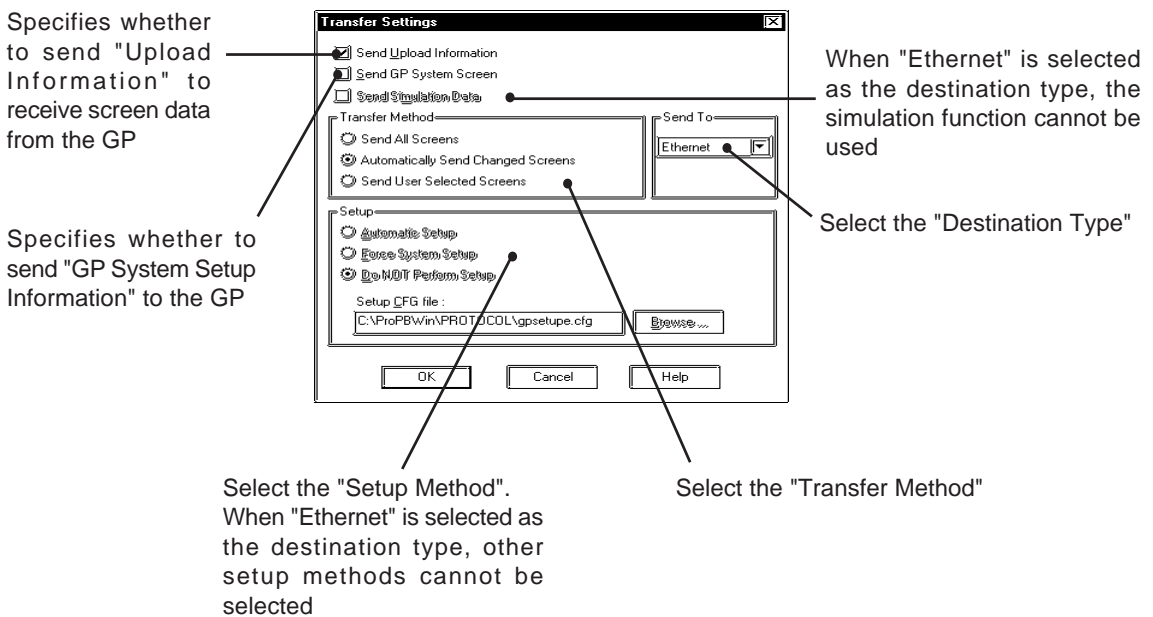
**Screen Transfer**

When you select the [Project] menu - [Transfer] command, the "Screen Transfer" screen will be displayed. The general description of this screen is shown below.





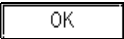
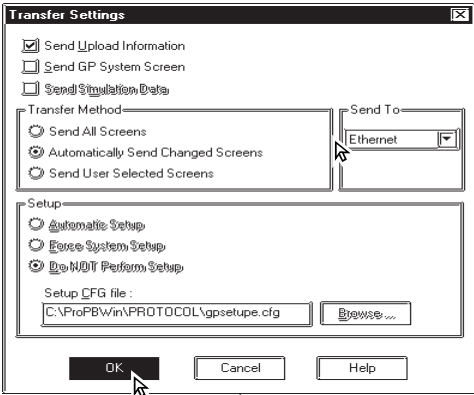



**Transfer Setup**

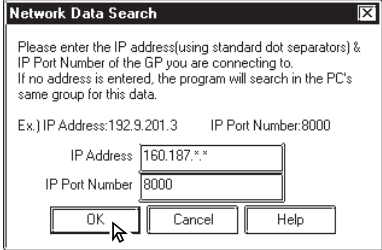
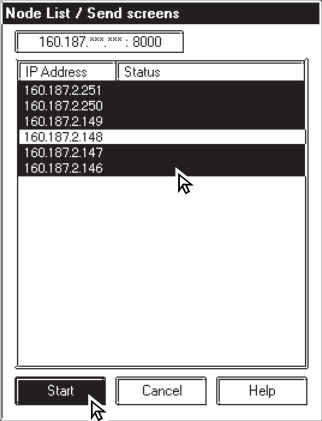
When you select the [Setup] menu - [Setup] command, the "Transfer Setup" screen will be displayed. The general description of this screen is shown below.



## 3-1-2 Screen Transfer Procedure

Send screen data to the GP connected to Ethernet. To receive data from the GP, refer to "NOTE".

Procedure	Remarks
<p>① Select the project to be sent from the GP-PRO/PB III for Windows program.</p> <p>② Open the [Transfer Screen] menu.</p> <p>③ Select the [Setup] menu - [Transfer Settings] command, or click on the  icon.</p>  <p>④ Select "Ethernet" as the destination type in the [Transfer Settings] dialog box, and click on the  button.</p>  <p>⑤ Select the [Transfer] menu - [Send] command, or click on the  icon.</p> 	<p>To receive data from the GP through Ethernet, select a project that specifies the "Memory Link Ethernet Type" as the PLC type, so that "Ethernet" can be selected as the destination type.</p> <p>When you select "Send GP System Screen", the IP addresses of all the GPs connected to the network will be changed to the settings specified in the "GP System Setup" screen. When you select "Send GP System Screen", pay attention to this point.</p> <p>To receive data from the GP, select [Transfer] menu - [Send] command, or click on the  icon. <u>To receive data from the GP, the required "Upload Information" must have been previously transferred to the GP together with the screen data.</u></p>

Procedure	Remarks
<p>⑥ Enter the IP address and port number of the destination GP in the [Network Data Search] dialog box, and click on the <input type="button" value="OK"/> button to start searching for the destination GP.</p>	<p>When you attempt to send screen data for the second time, the system automatically starts searching for the destination GP based on the preset search conditions.</p>
	<p>Asterisks (*) can be entered for the search conditions.</p>
<p>⑦ The search result will be displayed in the [Node List/Send Screen] dialog box. Select the IP address of the destination GP, and click on the <input type="button" value="Start"/> button to start screen transfer.</p>	<p>Example) 102.9.*.*</p>
	<p>When you select the destination GP while pressing the [Ctrl] key, several IP addresses can be simultaneously selected. When you select the destination GP while pressing the [Shift] key, several IP addresses can be continuously selected.</p>
	<p>When you attempt to receive data from the GP, you cannot select more than one IP address.</p>
	<p>After the host computer starts receiving data from the GP, a dialog box appears, inquiring the storing location. Specify the folder and file names to store the received data.</p>

# Chapter 4: Command Data

This chapter describes the Command Data isa used to send a command from the host computer to the GP, and the Response Data isa used to send a response from the GP to the host computer. This chapter provides an example of each command.

## 4-1 Display Command Data

The commands used to write data into the System Area, to read data from the System Area, and to enter graphic data is listed below.

### ■ Command List

Command	Contents
ESC W	Writes data into the System Area.
ESC R	Reads data from the System Area.
ESC T	Displays a character string.
ESC L	Displays a line.
ESC B	Displays a rectangle.
ESC S	Displays a filled rectangle.
ESC C	Displays a circle.
ESC A	Displays an arc.
ESC G	Displays a sector.
ESC P	Paint
ESC I	Interrupt Output Inquiry
ESC I	Touch Input

### ■ Extended Command List

Command	Contents	Extended function
ESC t	Extended function for displaying a character string	Turn, Direction, Highlighting
ESC l	Extended function for displaying a line	Arrow
ESC b	Extended function for displaying a rectangle	Chamfering
ESC s	Extended function for displaying a filled rectangle	Chamfering
ESC c	Extended function for displaying a circle	Tiling pattern
ESC g	Extended function for displaying a sector	Line type
ESC #	Brightness/Contrast Adjustment	
ESC \$	Brightness/Contrast Current Value	



#### Note:

- The maximum X and Y coordinates of the drawing commands vary depending on the model being used.
- **Reference** For the control codes used in the commands, refer to "*2-3 Communication Protocol Control Procedure*".

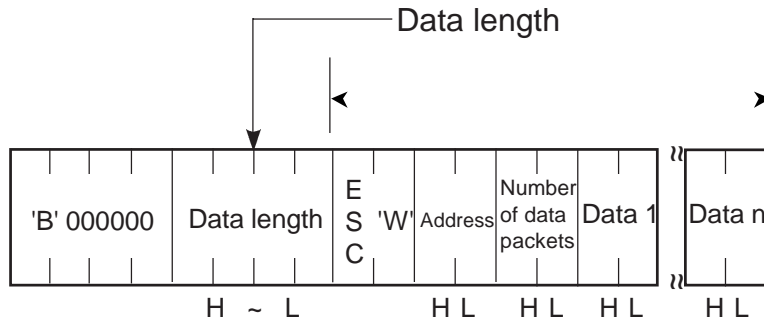


# 4-2 Writing Data To the System Area

This Write command allows the host computer to write data into the specified address of the System Area.

The contents of the Write command are as follows:

### Host Computer Command Data

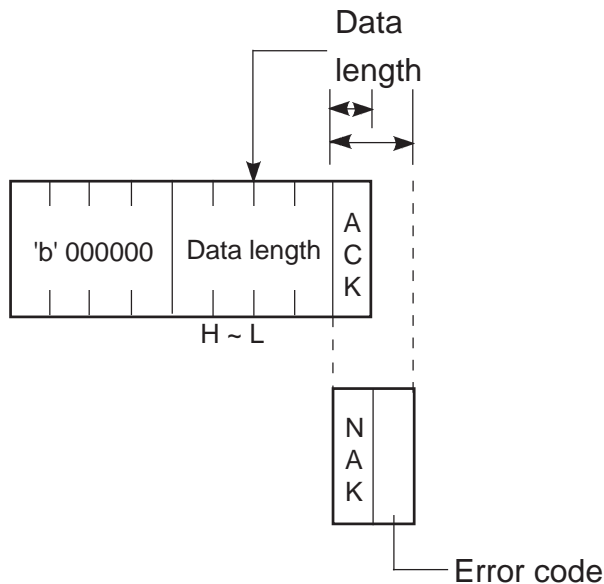


### <Setting Range>

- Address: 0000h to 1FFFh (0 to 8191)
- Number of data packets: 0001h to 0040h (1 to 64)
- Data: 0000h to FFFFh

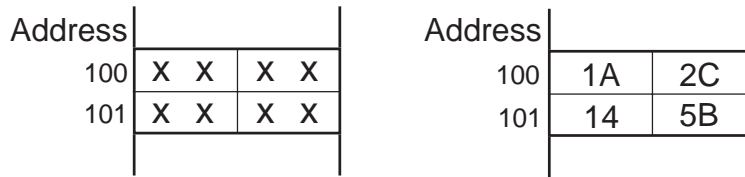
### GP Response Data is a

The response is "ACK" or "NAK".

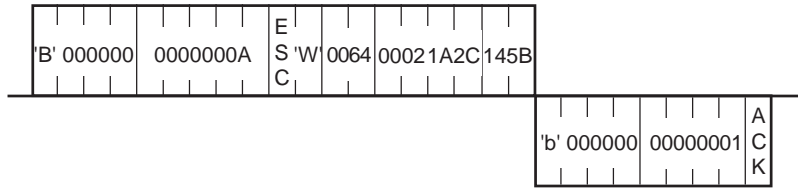


<Example>

Write hexadecimal data "1A2C" and "145B" to address 100 of the System Area.



**Host  
computer**

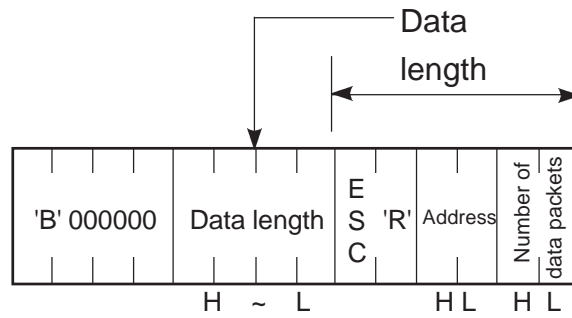


# 4-3 Reading System Area Data

This Read command allows the host computer to read data from the designated System Area address.

The contents of the Read command are as follows:

### Host Computer Command Data



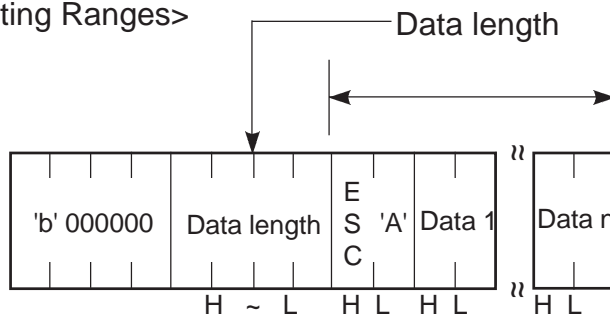
#### <Setting Range>

- Address : 0000h to 1FFFh (0 to 8191)
- Number of data packets : 0001h to 0040h (1 to 64)

### GP Response Data is a

- ◆ Normal response

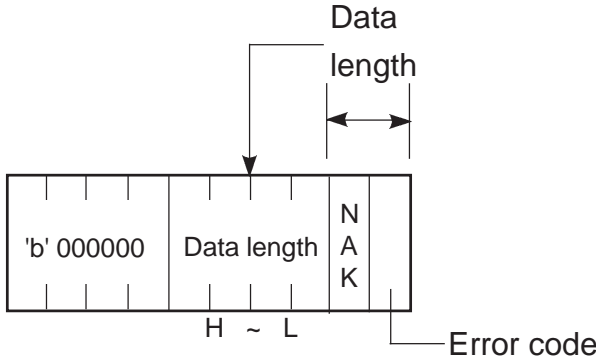
#### <Setting Ranges>



- Data: 0000h to FFFFh

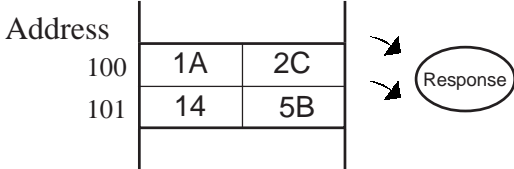
◆ Abnormal Response

The response is "NAK".

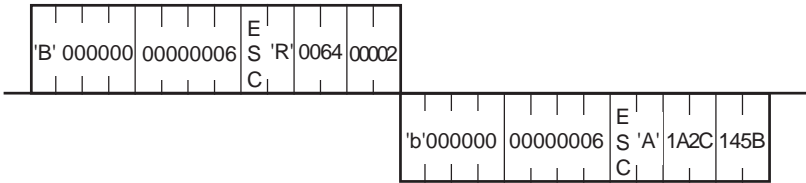


<Example>

Read two words of hexadecimal data from address 100 of the System Area.



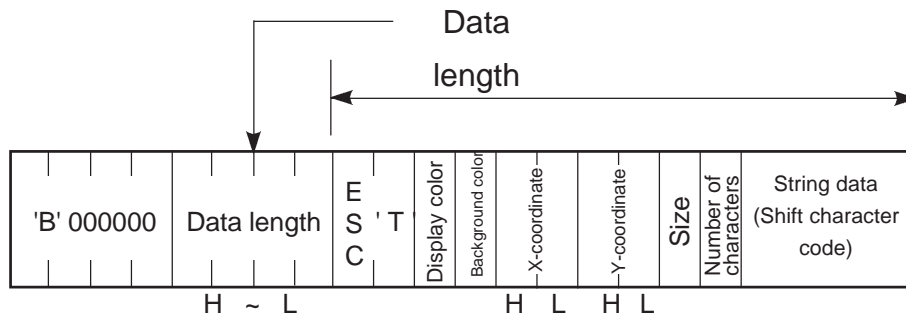
Host computer



# 4-4 Displaying a Character String

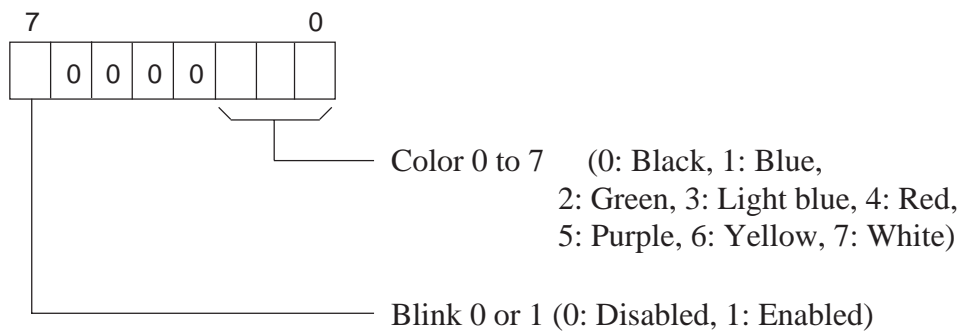
The contents of the command data isa for displaying a character string are as follows:

**Host Computer** Command Data

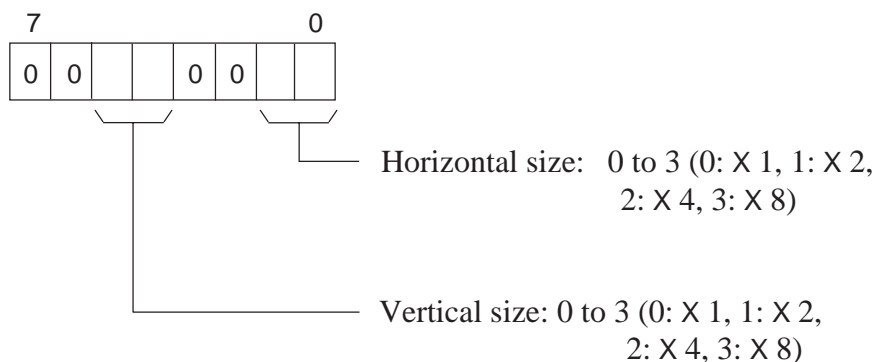


### <Setting Ranges>

- Display color/Background color



- X-coordinate: 0000h ~ 031Fh (0 ~799)
- Y-coordinate: 0000h ~ 0257h (0 ~599)
- Size



- Character size (bytes) : 01h to 50h (1 to 80)
- Character string data : An ASCII character uses one byte.  
A double-sized character uses 2 bytes.

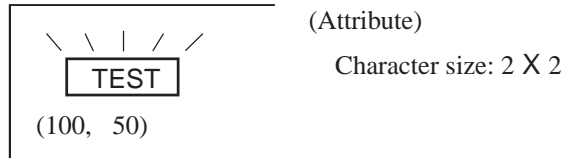
**GP** Response Data

The response is "ACK" or "NAK".

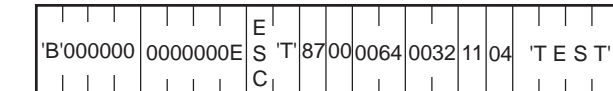
**Reference** 4-2 Writing Data To System Area

<Example>

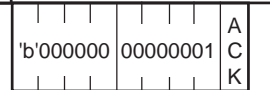
Display "TEST" in the blink mode at the coordinates of (100, 50).



**Host computer**



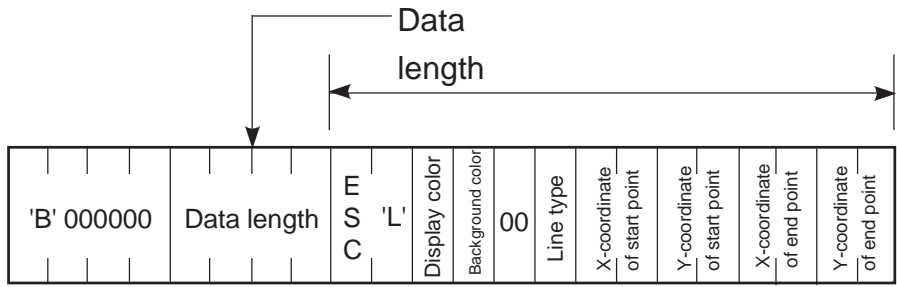
**GP**



# 4-5 Displaying a Line

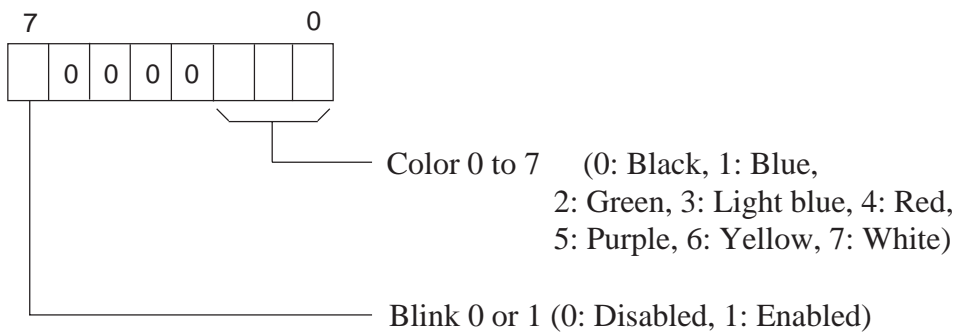
The contents of the command data for drawing a line are as follows:

**Host Computer**    **Command Data**



<Setting Range>

- Display color/Background color



- Line type :    00h to 07h (00:\_\_\_\_\_, 01:\_ \_ \_ \_ , 02:\_\_\_\_\_, 03:\_\_\_\_\_, 04:\_\_\_\_\_, 05:\_ \_ \_ \_ , 06:\_\_\_\_\_, 07:\_\_\_\_\_,)
- X-coordinate : 0000h ~ 031Fh (0 ~799)
- Y-coordinate : 0000h ~ 0257h (0 ~599)

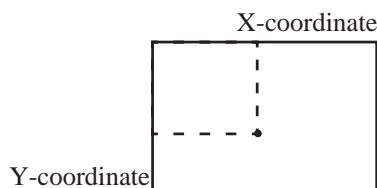
**GP**                      **Response Data**

The response is "ACK" or "NAK".

**Reference** 4-2 Writing Data Into the System Area

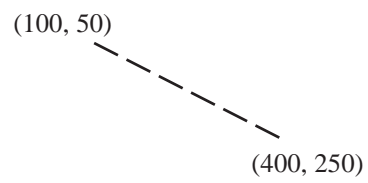


- Line types 0 to 3 are 1-dot lines, and line types 4 to 7 are 2-dot lines.
- To draw a dot, enter the same values for the X coordinate start and end points, and for the Y-coordinate start and end points.



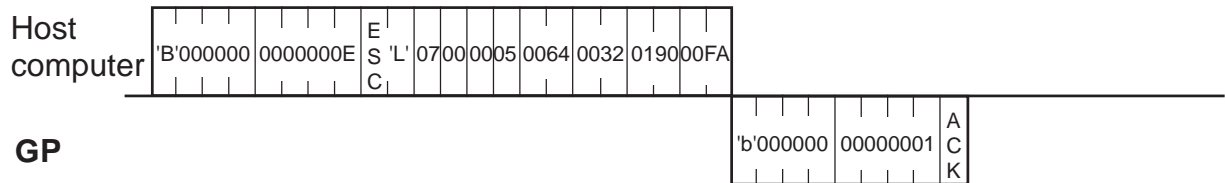
<Example>

Draw a 2-dot dotted line between (100, 50) and (400, 250).



(Attributes)

Display color : White  
 Background color : Black  
 Line type : 2-dot dotted line

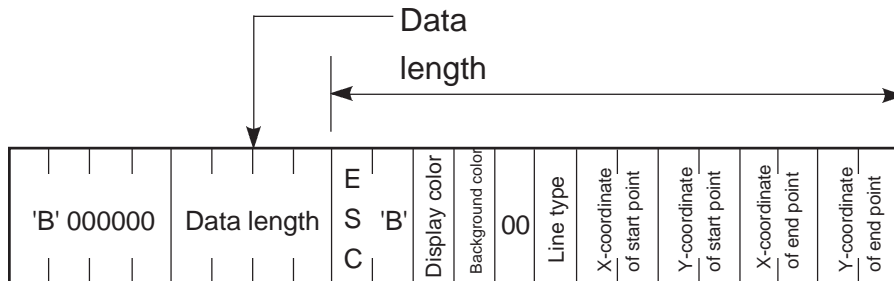




# 4-6 Displaying a Rectangle

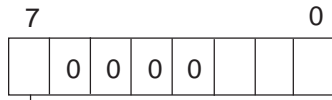
The contents of the command data for drawing a rectangle are as follows:

### Host Computer Command Data



### <Setting Range>

- Display color/Background color



Color 0 to 7 (0: Black, 1: Blue, 2: Green, 3: Light blue, 4: Red, 5: Purple, 6: Yellow, 7: White)

Blink 0 or 1 (0: Disabled, 1: Enabled)

- Line type : 00h to 07h (00: ———, 01: - - - -, 02: — — —, 03: — — — —, 04: ———, 05: - - - -, 06: — — —, 07: — — — —,)
- X-coordinate : 0000h ~ 031Fh (0 ~799)
- Y-coordinate : 0000h ~ 0257h (0 ~599)

### GP Response Data

The response is "ACK" or "NAK".

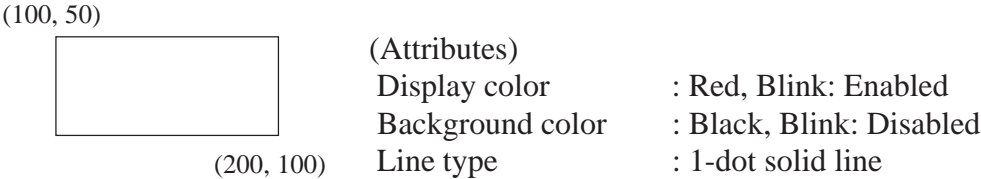
**Reference** 4-2 Writing Data Into the System Area



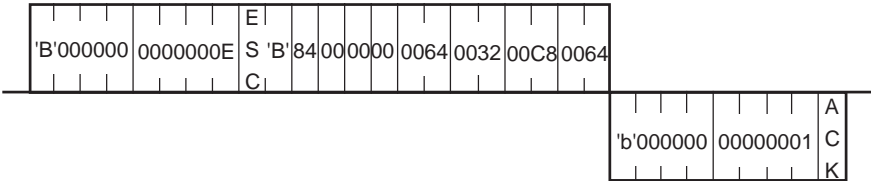
**Note:** • Line types 0 to 3 are 1-dot lines, and line types 4 to 7 are 2-dot lines.

<Example>

Draw a rectangle with its two diagonal points placed at (100, 50) and (200, 100).



Host  
computer



GP

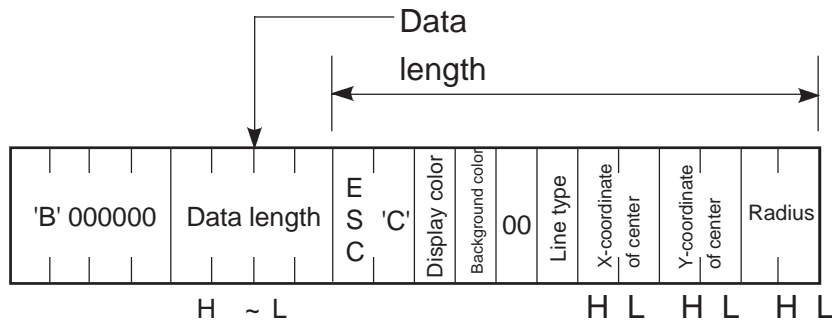




# 4-8 Displaying a Circle

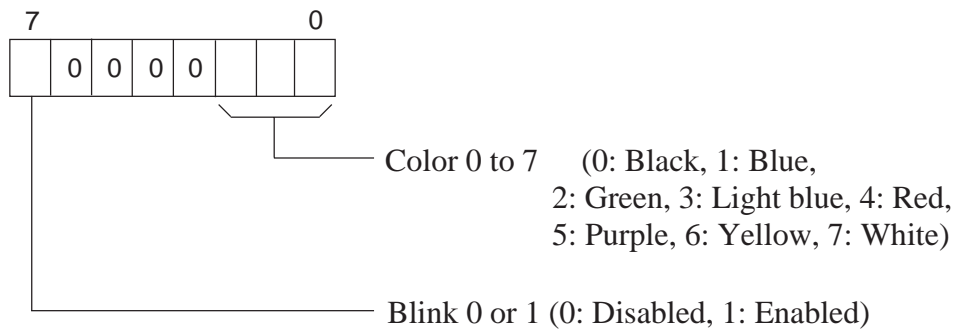
The contents of the command data for drawing a circle are as follows:

**Host Computer**    Command Data



<Setting Range>

- Display color/Background color



- Line type :        00h to 03h (00: \_\_\_\_\_, 01: - - - -, 02: \_ \_ \_ -, 03: \_ \_ \_ \_)
- X-coordinate :    0000h ~ 031Fh (0 ~799)
- Y-coordinate :    0000h ~ 0257h (0 ~599)
- Radius :            0000h ~ 031Fh (0 ~799)

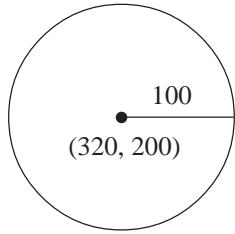
**GP**                    Response Data is a

The response is "ACK" or "NAK".

**Reference** *4-2 Writing Data To the System Area*

<Example>

Draw a circle with its center placed at (320, 200) and radius of "100".



(Attributes)

- Display color : White
- Background color : Black
- Line type : 1-dot solid line

Host  
computer



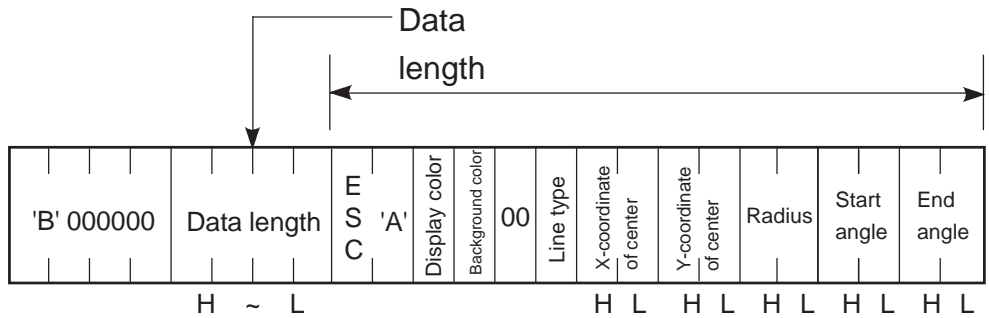
GP



# 4-9 Displaying an Arc

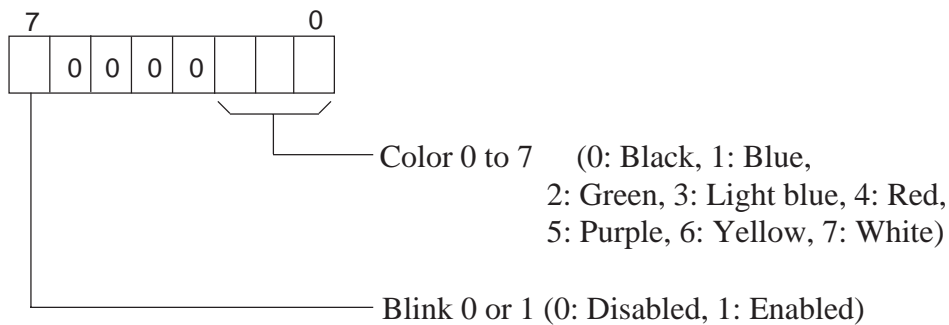
The contents of the command data for drawing an arc are as follows:

## Host Computer Command Data



### <Setting Range>

- Display color/Background color



- Line type : 00h to 03h (00: \_\_\_\_\_, 01: \_ \_ \_ \_ \_, 02: \_ \_ \_ \_ , 03: \_ \_ \_ \_ \_)
- X-coordinate : 0000h ~ 031Fh (0 ~799)
- Y-coordinate : 0000h ~ 0257h (0 ~599)
- Radius : 0000h ~ 031Fh (0 ~799)
- Angle : 0000h ~ 0168h (0 to 360)

## GP Response Data

The response is "ACK" or "NAK".

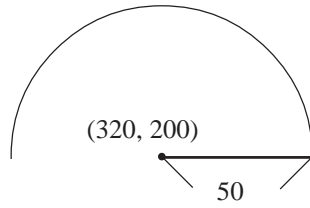
**Reference** 4-2 Writing Data To the System Area



- The drawing direction is counter-clockwise.
- Do not enter the same value for the start angle and end angle.

<Example>

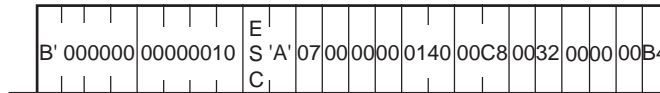
Draw an arc with its center placed at (320, 200) with a radius of "50".



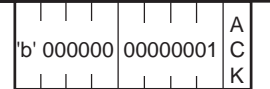
(Attributes)

- Display color : While
- Background color : Black
- Line type : 1-dot solid line
- Start angle : 0°
- End angle : 180°

Host  
computer



GP

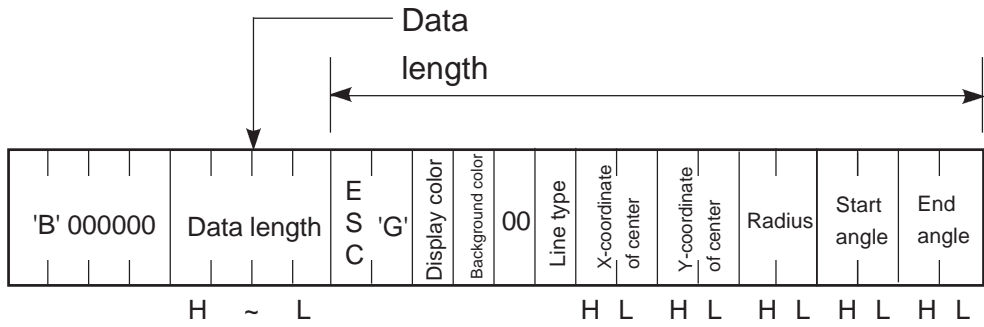




# 4-10 Displaying a Sector

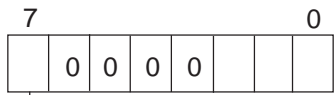
The contents of the command data for drawing a sector are as follows:

**Host Computer**    Command Data



<Setting Range>

- Display color/Background color



Color 0 to 7 (0: Black, 1: Blue, 2: Green, 3: Light blue, 4: Red, 5: Purple, 6: Yellow, 7: White)

Blink 0 or 1 (0: Disabled, 1: Enabled)

- Line type : 00h to 03h (00: ———, 01: - - - -, 02: — - —, 03: — - - -)
- X-coordinate : 0000h ~ 031Fh (0 ~799)
- Y-coordinate : 0000h ~ 0257h (0 ~599)
- Radius: 0000h ~ 031Fh (0 ~799)
- Start angle : 0000h to 0168h (0 to 360)
- End angle : 0000h to 0168h (0 to 360)

**GP**                      Response Data

The response is "ACK" or "NAK".

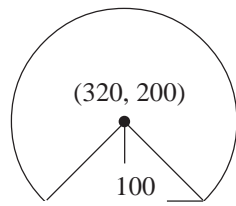
**Reference** 4-2 Writing Data To the System Area



- The drawing direction is counter-clockwise.
- Do not enter the same value for the start angle and end angle.

<Example>

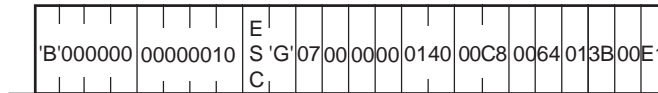
Draw a sector with its center placed at (320, 200) and a radius of "100".



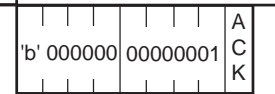
(Attributes)

- Display color : White, Start angle: 315°
- Background color : Black, End angle: 225°
- Line type : 1-dot solid line

Host  
computer



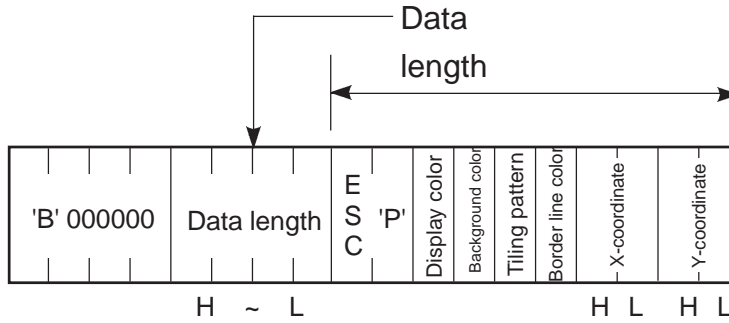
GP



# 4-11 Filling an Object

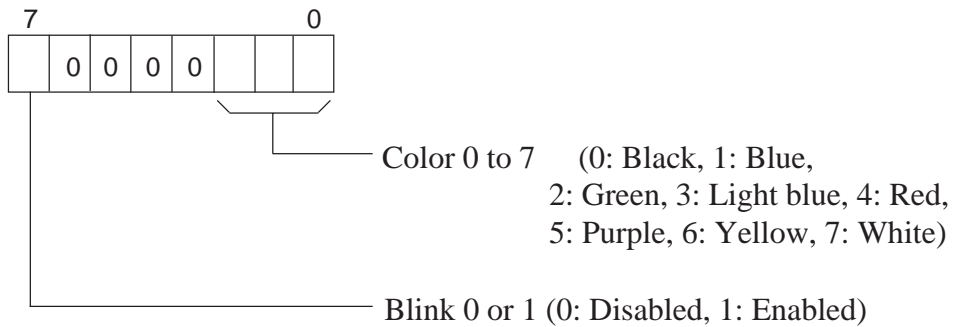
The contents of the command data for filling an object are as follows:

**Host Computer**    **Command Data**



<Setting Range>

- Display color/Background color



- Tiling pattern : 00h to 08h  
▼Reference▲ *4-7 Displaying a Filled "Rectangle TilingPatterns"*
- X-coordinate : 0000h ~ 031Fh (0 ~799)
- Y-coordinate : 0000h ~ 0257h (0 ~599)

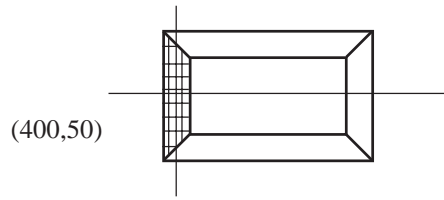
**Careful!** • *Be sure to set the border line blink function to "Disabled".*

**GP**    Response Data  
 The response is "ACK" or "NAK"

▼Reference▲ *4-2 Writing Data To the System Area*

<Example>

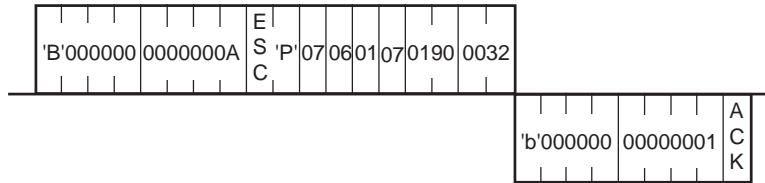
Fill the object placed at (400, 50).



(Attributes)

- Display color : White
- Background color : Yellow
- Border line color : White
- Tiling pattern : 1

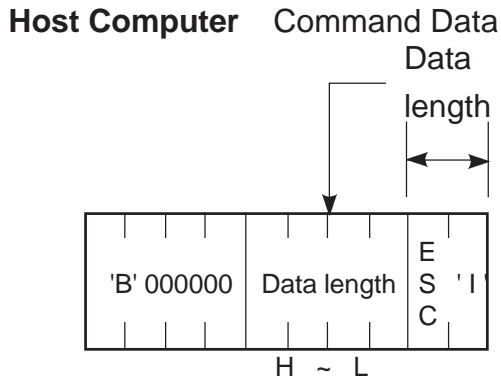
Host  
computer



GP

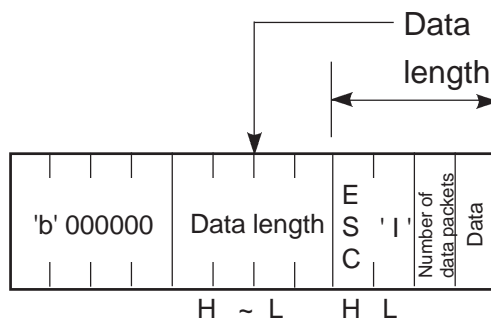
# 4-12 Interrupt Output Requests

The contents of the command for requesting touch panel input from the GP are as follows:



**GP**      Response Data

◆ Normal response



◆ Abnormal response

The response is "NAK".

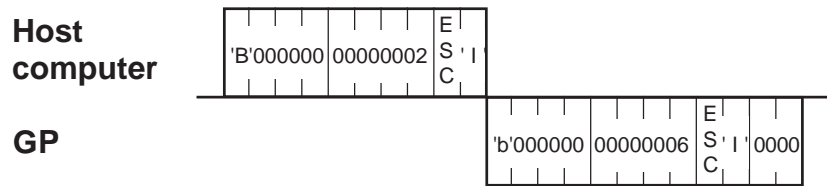
**Reference** 4-3 Reading Data From the System Area

<Settings>

- Number of data packets : Includes the data to be currently sent and data of the previous interrupt output. If there is no interrupt output data, "00h" is entered.
- Data : "00h" to "FEh" will be output. If there is no interrupt output data, "00h" is entered.

<Example>

Request to the GP whether the touch panel input has been activated or not.

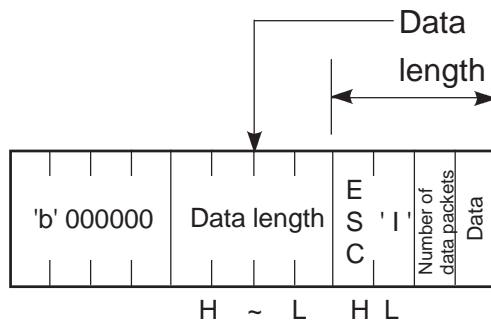


# 4-13 Touch Input

The contents of the command for sending touch input data from the GP to the host computer are as follows:

**Host computer** None

**GP** Response Data



<Settings>

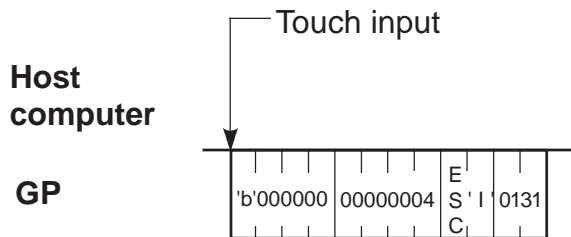
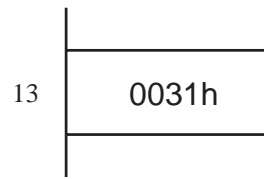
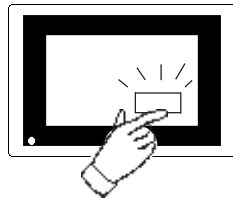
- Number of data packets : 01h
- Data : 00h to FFh

- ◆ If T-Tag data or an absolute value is written into address 13 of the System Area, the lower 8-bit data will be output as an interrupt code. ("13H", "11h" or "FFh" will not be output. "00" will be output.)
- ◆ When any data is written into address 13, the GP automatically outputs response data.

This function is intended for TCP connection only. During UDP connection, use the interrupt output request command.

## &lt;Example&gt;

Use a T-tag to enter "0031h" in address 13 of the System Area.

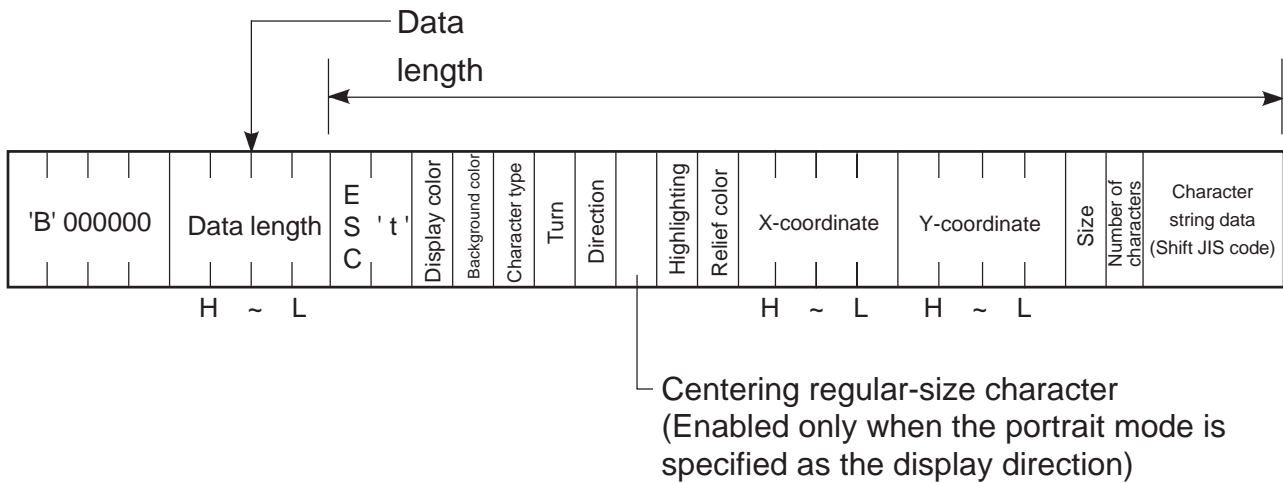




# 4-14 Additional Character String Features

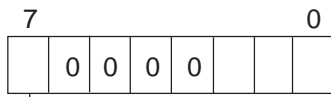
The contents of the command data for displaying a character string using the extended functions are as follows:

## Host Computer Command Data



### <Setting Range>

- Display color/Background color/Relief color

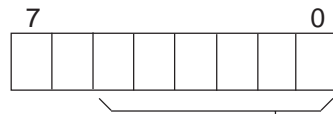


Color 0 to 7 (0: Black, 1: Blue, 2: Green, 3: Light blue, 4: Red, 5: Purple, 6: Yellow, 7: White)

Blink 0 or 1 (0: Disabled, 1: Enabled)

\* The relief color is fixed to the blink mode.

- Character type:



Character types 0 to 9

0: Quarter size (Shift code)

1: Regular size

2: Double size (Shift code), regular-size display

3: 5 X 7 font

4: 7 X 9 font

5: 11 X 16 font

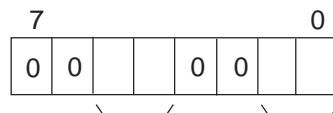
6: 24 X 32 font

7: 7 X 9F font

8: 11 X 16F font

9: Regular size (Shift code)

- Turn : 00f to 03h (0: 0°, 1: 90°, 2: 180°, 3: 270°)
- Direction : 00h to 01h (0: Landscape, 1: Portrait)
- Centering regular-size character : 00h to 01h (0: Disabled, 1: Enabled)
- Highlighting : 00h to 02h (0: Normal, 1: Highlighted 2: Relief)
- Relief : 00h to 07h (0: Black, 1: Blue, 2: Green, 3: Light blue, 4: Red, 5: Purple, 6: Yellow, 7: White)
- X-coordinates : 0000h ~ 031Fh (0 ~799)
- Y-coordinates : 0000h ~ 0257h (0 ~599)
- Size:



Lateral size: 0 to 3 (0: X 1, 1: X 2,  
2: X 4, 3: X 8)

Vertical size: 0 to 3 (0: X 1, 1: X 2,  
2: X 4, 3: X 8)

- Character size (bytes): 01h to 50h (1 to 80)
- Character string data: An ASCII character uses one byte. A "Double-sized" character uses 2 bytes.

## GP

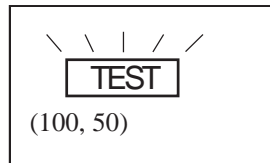
### Response Data

The response is "ACK" or "NAK".

**Reference** 4-2 Writing Data To System Area

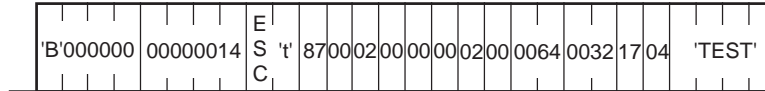
<Example>

Display "TEST" using double-size characters in the blink mode at the coordinates of (100, 50).



(Attributes)  
 Character size : 2 X 2  
 Relief characters  
 Relief color : Black

**Host computer**



**GP**

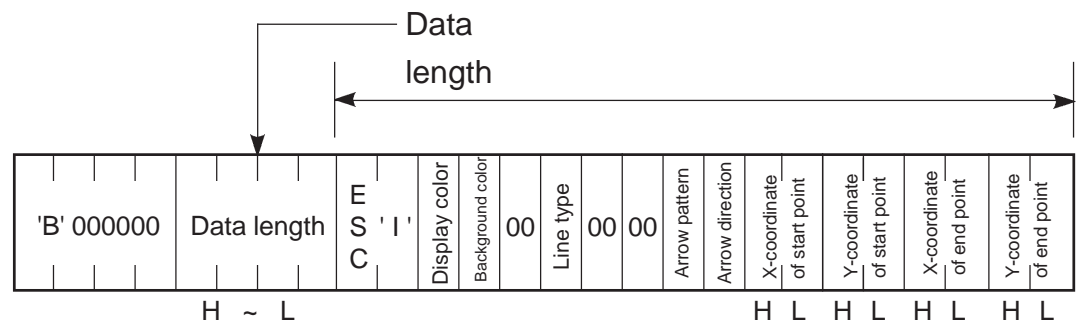


## 4-15 Additional Line Features

The contents of the command data for drawing a line using the extended function are as follows:

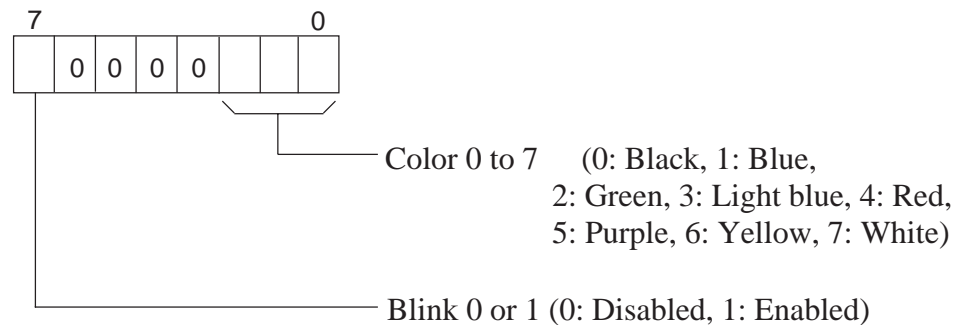
The arrow display function is added as the extended function.

### Host Computer Command Data



### <Setting Range>

- Display color/Background color



- Line type: 00h to 07h (00: ———, 01: - - - -, 02: — — —, 03: — — — —, ——— — — — —, — — — — — — — —, — — — — — — — —, — — — — — — — —, 04: ——— ———, 05: ——— ———, 06: ——— ———, 07: ——— ———, )
- Arrow pattern: 00h to 03h (00: None, 01: Arrow, 02: Reserved, 03: Reserved)
- Arrow direction: 00h to 01h (0: Both ends, 1: End point)
- X-coordinate : 0000h ~ 031Fh (0 ~ 799)
- Y-coordinate : 0000h ~ 0257h (0 ~ 599)

The response is "ACK" or "NAK".

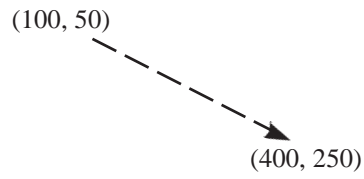
### Reference 4-2 Writing Data To System Area



- Line types 0 to 3 are 1-dot lines, and line types 4 to 7 are 2-dot lines.

<Example>

Draw a 2-dot dotted line between (100, 50) and (400, 250).

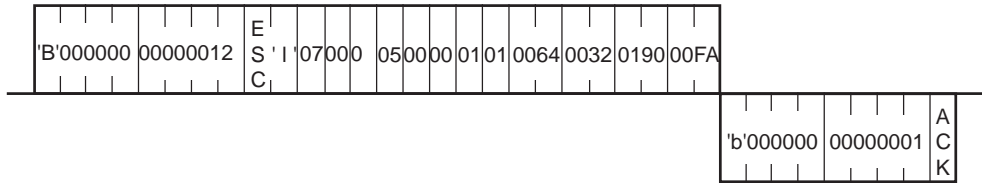


(Attributes)

Display color : White

Background color: Black

**Host  
computer**

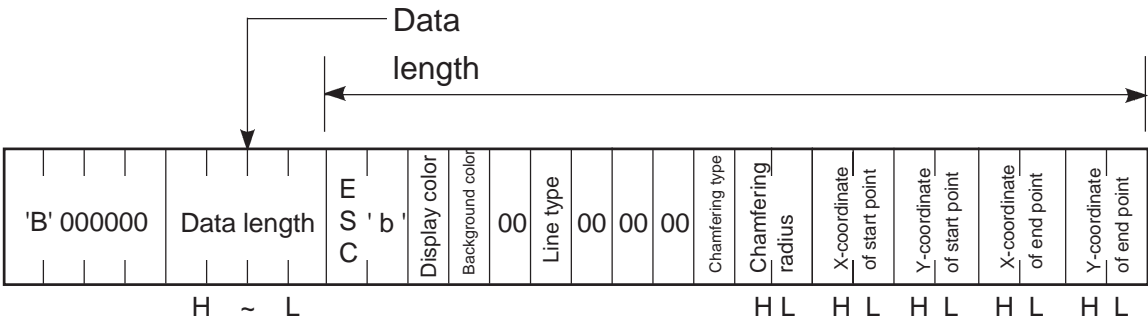


**GP**

# 4-16 Additional Rectangle Features

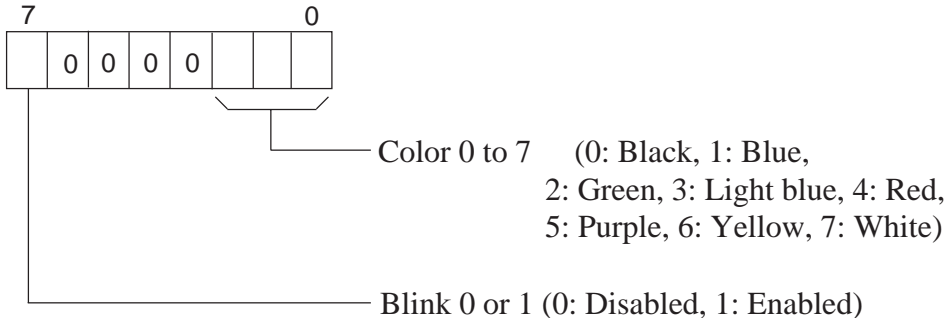
The contents of the command data for drawing a rectangle using the extended function are as follows:  
 The chamfering function is added as the extended function.

**Host Computer** Command Data



<Setting Range>

- Display color/Background color



- Chamfering type : 00h to 02h (00: None, 01: Curve, 02: Line)
- Chamfering radius : 00h to 20h (0 to 32)
- Line type : 00h to 03h, 08h, 09h (00: \_\_\_\_\_, 01: - - - , 02: - \_ \_ , 03: - - - - -, 08: \_\_\_\_\_, 09: \_\_\_\_\_)
- X-coordinate : 0000h ~ 031Fh (0 ~799)
- Y-coordinate : 0000h ~ 0257h (0 ~599)

**GP** Response Data

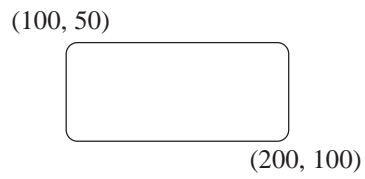
The response is "ACK" or "NAK".

**Reference** 4-2 Writing Data To System Area

**Note:** • Line types 0 to 3, 8, and 9 are 1-dot, 8-dot, and 5-dot lines, respectively.

<Example>

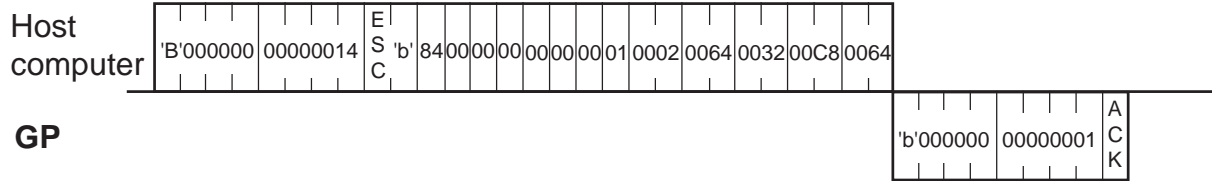
Draw a rectangle using a 1-dot solid line with its two diagonal points placed at (100, 50) and (200, 100).



(Attributes)

Display color : Red, Blink: Enabled  
 Background color : Black, Blink: Disabled  
 Line type : 1-dot solid line,  
 Chamfering type:

Curve

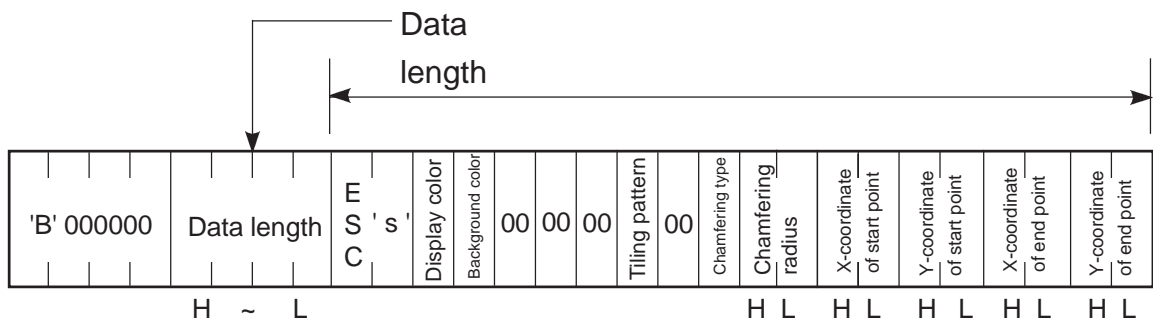


## 4-17 Additional Filled Rectangle Features

The contents of the command data for drawing a painted rectangle using the extended function are as follows:

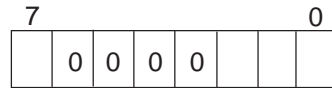
The chamfering function is added as the extended function.

### Host Computer Command Data



### <Setting Range>

- Display color/Background color



Color 0 to 7 (0: Black, 1: Blue, 2: Green, 3: Light blue, 4: Red, 5: Purple, 6: Yellow, 7: White)

Blink 0 or 1 (0: Disabled, 1: Enabled)

- Chamfering type : 00h to 02h (00: None, 01: Curve, 02: Line)
- Chamfering radius : 00h to 20h (0 to 32)
- Tiling pattern : 00h to 08h

**Reference** *"Tiling Patterns" in "4-7 Displaying Painted Rectangle"*

- X-coordinate : 0000h ~ 031Fh (0 ~799)
- Y-coordinate : 0000h ~ 0257h (0 ~599)

### GP Response Data

The response is "ACK" or "NAK".

**Reference** *4-2 Writing Data To System Area*

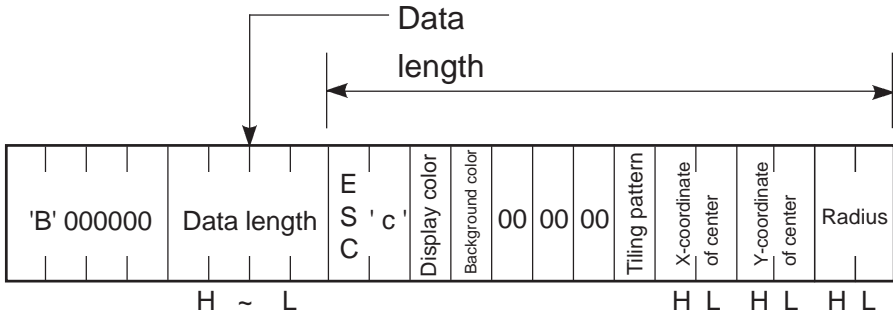




# 4-18 Additional Painted Circle Features

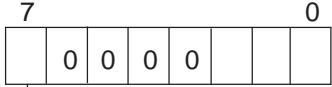
The contents of the command data for drawing a painted circle using the extended function are as follows:  
 The tiling pattern is added as the additional function.

**Host Computer** Command Data



<Setting Range>

- Display color/Background color



Color 0 to 7 (0: Black, 1: Blue, 2: Green, 3: Light blue, 4: Red, 5: Purple, 6: Yellow, 7: White)  
 Blink 0 or 1 (0: Disabled, 1: Enabled)

- Tiling pattern : 00h to 08h  
▼Reference▲ 4-7 Displaying Painted Rectangle "Tiling Patterns"
- X-coordinate : 0000h ~ 031Fh (0 ~799)
- Y-coordinate : 0000h ~ 0257h (0 ~599)
- Radius: 0001h ~ 031Fh (0 ~799)

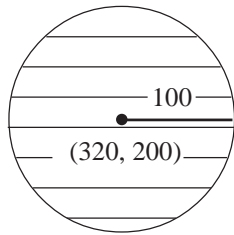
**GP** Response Data

The response is "ACK" or "NAK".

▼Reference▲ 4-2 Writing Data To System Area

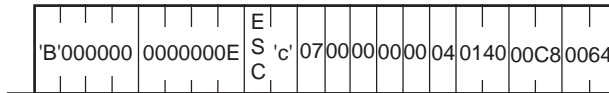
<Example>

Draw a circle of tiling pattern 4 with its center placed at (320, 200) and radius of "100".

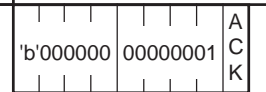


(Attributes)  
 Display color : White  
 Background color : Black  
 Tiling pattern : 4

Host  
computer



GP

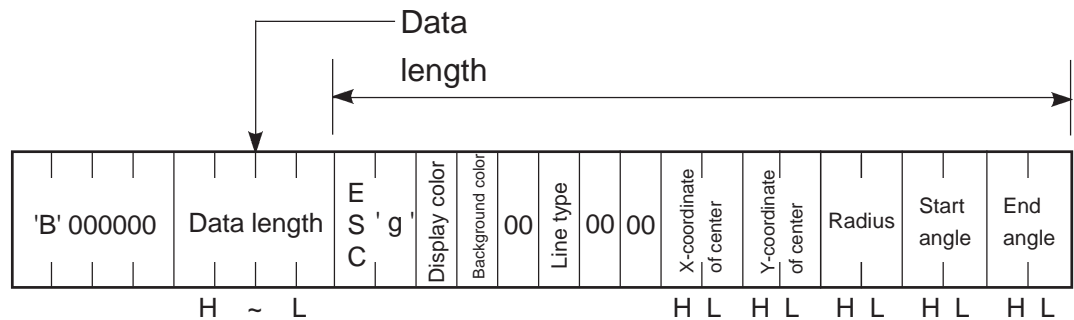


## 4-19 Additional Pie Shape Features

The contents of the command data for drawing a pie shape using the extended function are as follows:

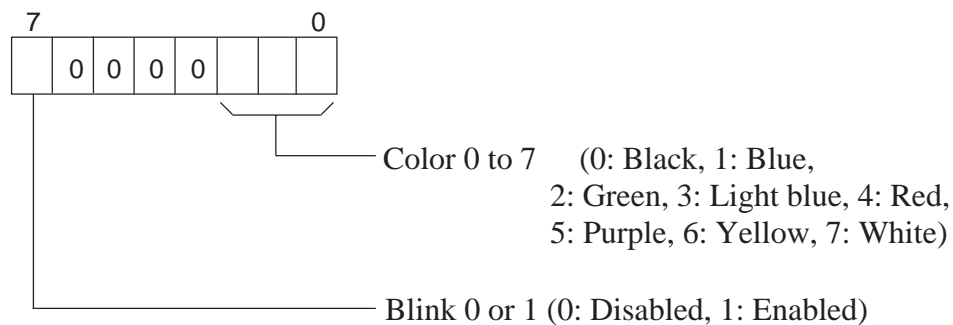
The line type is added as the extended function.

### Host Computer Command Data



### <Setting Range>

- Display color/Background color



- Line type: 00h to 03h, 08h, 09h (00: ———, 01: - - -, 02: - — —, 03: — - - —, 08: ———, 09: ———)



- **Note:** Line types 0 to 3, 8, and 9 are 1-dot lines, 8-dot line, 5-dot line, respectively.

- X-coordinate: 0000h ~ 031Fh (0 ~ 799)
- Y-coordinate: 0000h ~ 0257h (0 ~ 599)
- Radius: 0001h ~ 031Fh (0 ~ 799)
- Angle: 0000h ~ 0168h (0 ~ 360)



- *The drawing direction is counter-clockwise.*
- *Do not enter the same value for the start angle and end angle.*

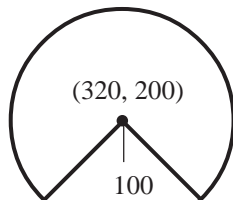
### GP Response Data

The response is "ACK" or "NAK".

**Reference** 4-2 Writing Data To System Area

<Example>

Draw a pie shaped using a 3-dot solid line with its center placed at (320, 200) and radius of "100".



(Attributes)

Display color : White, Start angle: 315°

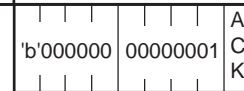
Background color: Black, End angle: 225°

Line type : 3-dot solid line

Host  
computer



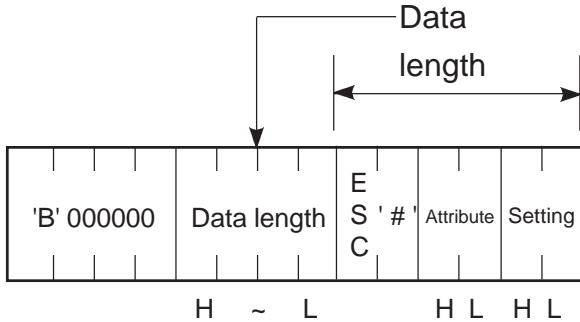
GP



# 4-20 Brightness/Contrast Adjustments

The contents of the command data for brightness/contrast adjustment are as follows:  
(With some models, the brightness/contrast cannot be adjusted. Refer to page 4-41 "Brightness and Contrast Setting Table".)

### Host Computer Command Data



### <Setting Range>

- Attribute: 00h to 01h (0: Contrast adjustment, 1: Brightness adjustment)
- Setting: Refer to page 4-41 "Brightness and Contrast Table".

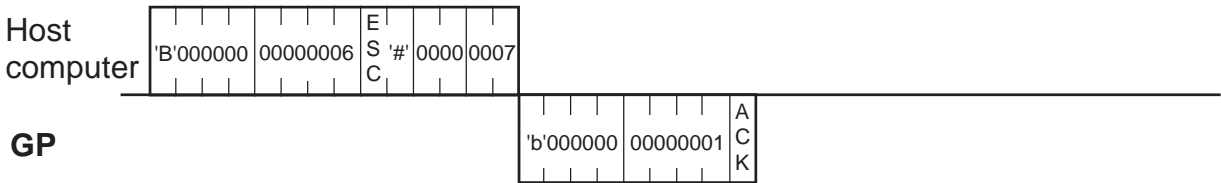
### GP Response Data

The response is "ACK" or "NAK".

**Reference** 4-2 Writing Data To System Area

### <Example>

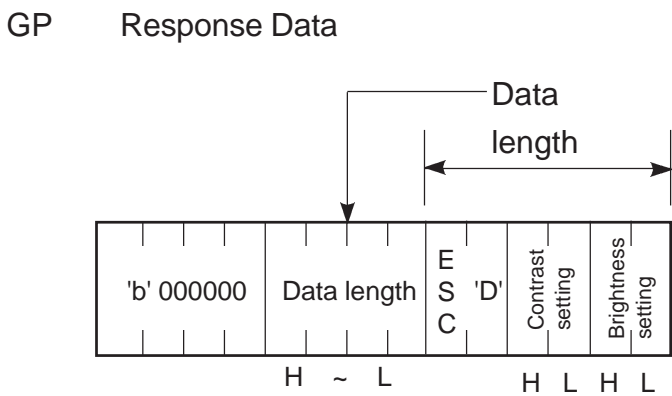
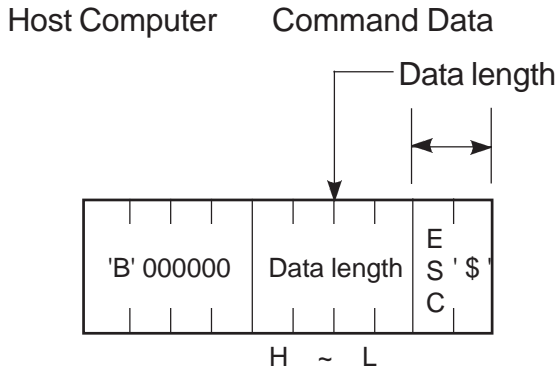
Enter "7" for contrast adjustment.



# 4-21 Brightness/Contrast Settings

The contents of the command data is a that allows you to acquire the current brightness/contrast settings are as follows:

(With some models, the brightness/contrast cannot be adjusted. Refer to page 4-41 "Brightness and Contrast Setting Table".)



<Setting Range>

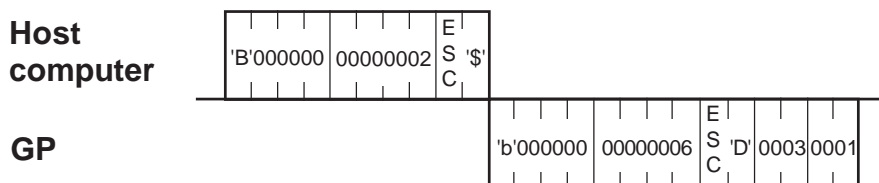
- Contrast setting :      00h to 07h (0: Bright ~ 7: Dark)
- Brightness setting :      Refer to page 4-41 "Brightness and Contrast Setting Table".



• *With the models that disable both contrast adjustment and brightness adjustment, the contrast/brightness settings are "FFFF".*

<Example>

Obtain contrast setting "3" and brightness setting "1".



### ■ Brightness and Contrast Setting Table

	Brightness	Contrast
GP Units	Setting Range	Setting Range
GP470E	0(Dark) to 1(Light)	
GP570S		0(Light) to 7(Dark)
GP570T		
GP570VM		
GP571T		
GP57JS		0(Light) to 7(Dark)
GP675S		0(Light) to 7(Dark)
GP675T		
GP870VM		

 Cannot be set

	Brightness	Contrast
GP/GLC Units	Setting Range	Setting Range
GP477RE	0 (Dark) to 1(Light)	
GP577RS	0 (Light) to 3(Dark)	0(Light) to 7(Dark)
GP577RT	0 (Light) to 3(Dark)	
GP377RT	0 (Light) to 3(Dark)	
GP2600T	0 (Light) to 3(Dark)	
GP2601T	0 (Light) to 3(Dark)	
GP2500T	0 (Light) to 3(Dark)	
GP2501T	0 (Light) to 3(Dark)	
GP2501S	0 (Light) to 3(Dark)	0(Light) to 7(Dark)
GP2501L	0 (Light) to 3(Dark)	0(Light) to 7(Dark)
GP2500S	0 (Light) to 3(Dark)	0(Light) to 7(Dark)
GP2500L	0 (Light) to 3(Dark)	0(Light) to 7(Dark)
GP2400T	0 (Light) to 3(Dark)	
GP2300T	0 (Light) to 3(Dark)	
GP2300S	0 (Light) to 3(Dark)	0(Light) to 7(Dark)
GP2300L	0 (Light) to 3(Dark)	0(Light) to 7(Dark)
GLC2600T	0 (Light) to 3(Dark)	
GLC2500T	0 (Light) to 3(Dark)	
GLC2400T	0 (Light) to 3(Dark)	
GLC2300T	0 (Light) to 3(Dark)	
GLC2300L	0 (Light) to 3(Dark)	0(Light) to 7(Dark)



# *Memo*

# Chapter 5: Memory Link API

The Memory Link API is a 32-bit API for Windows that enables you to easily access the GP from the application using the memory link protocol, without understanding details of the memory link.

## 5-1 How to Use Memory Link API

Memory Link API users need to first create a communication channel to the GP. (Creating a communication channel is referred to as "opening a connection". )

After necessary communication with the GP is completed, close the communication channel (connection). If you do not intend to use the same socket for the next communication with the GP, cancel the socket. To perform the next communication with the GP, open the connection again. (The socket can be re-used)

### ■ Development Environments

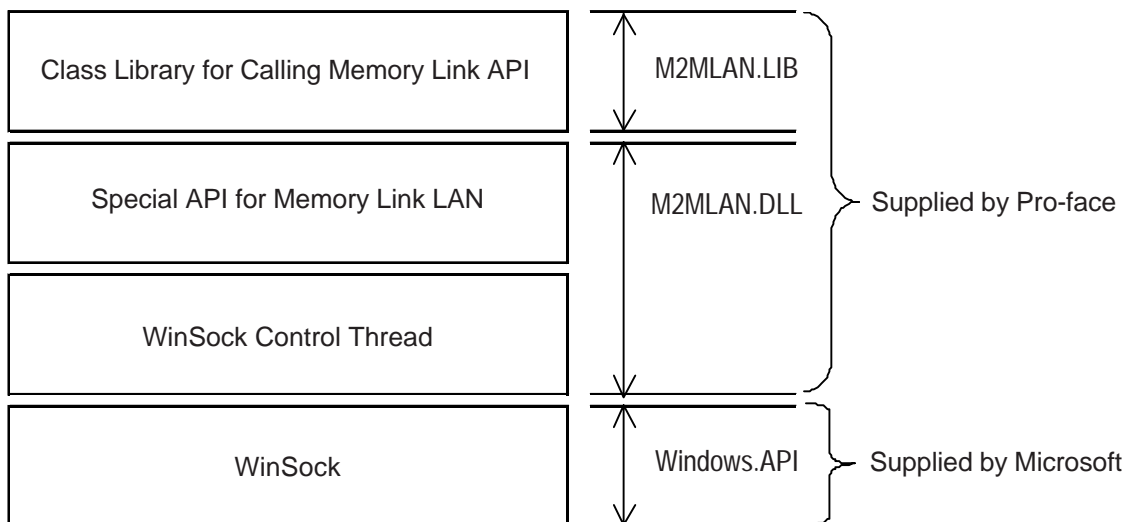
Compiler: Microsoft Visual C++™ Ver 4.1

OS : Microsoft Windows® 95 or higher

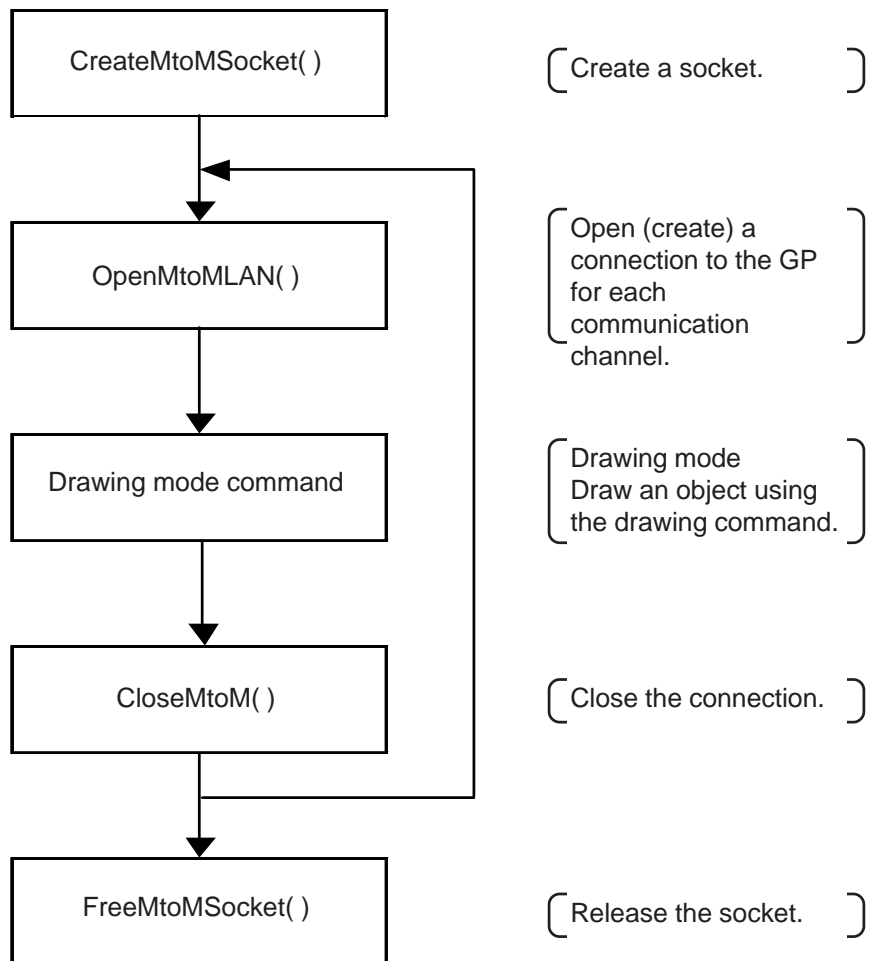
Others : The following files are contained on the GP-PRO/PBIII for Windows CD-ROM. To view them, open the CD-ROM's [MTOMLAN] folder and double-click on the [MTOMLAN.EXE] file. (Self-extracting file)

- MTOMAPI.H
- MTOMLAN.LIB
- MTOMLAN.DLL

### ■ Memory Link API Software Structure Diagram



### ■ General Operation of Memory Link API



## 5-1-1

**Synchronous and Asynchronous Transmission**

Synchronous transmission" is a transmission method with which system functionality does not return until the API's processing normally or abnormally ends after an API command is called.

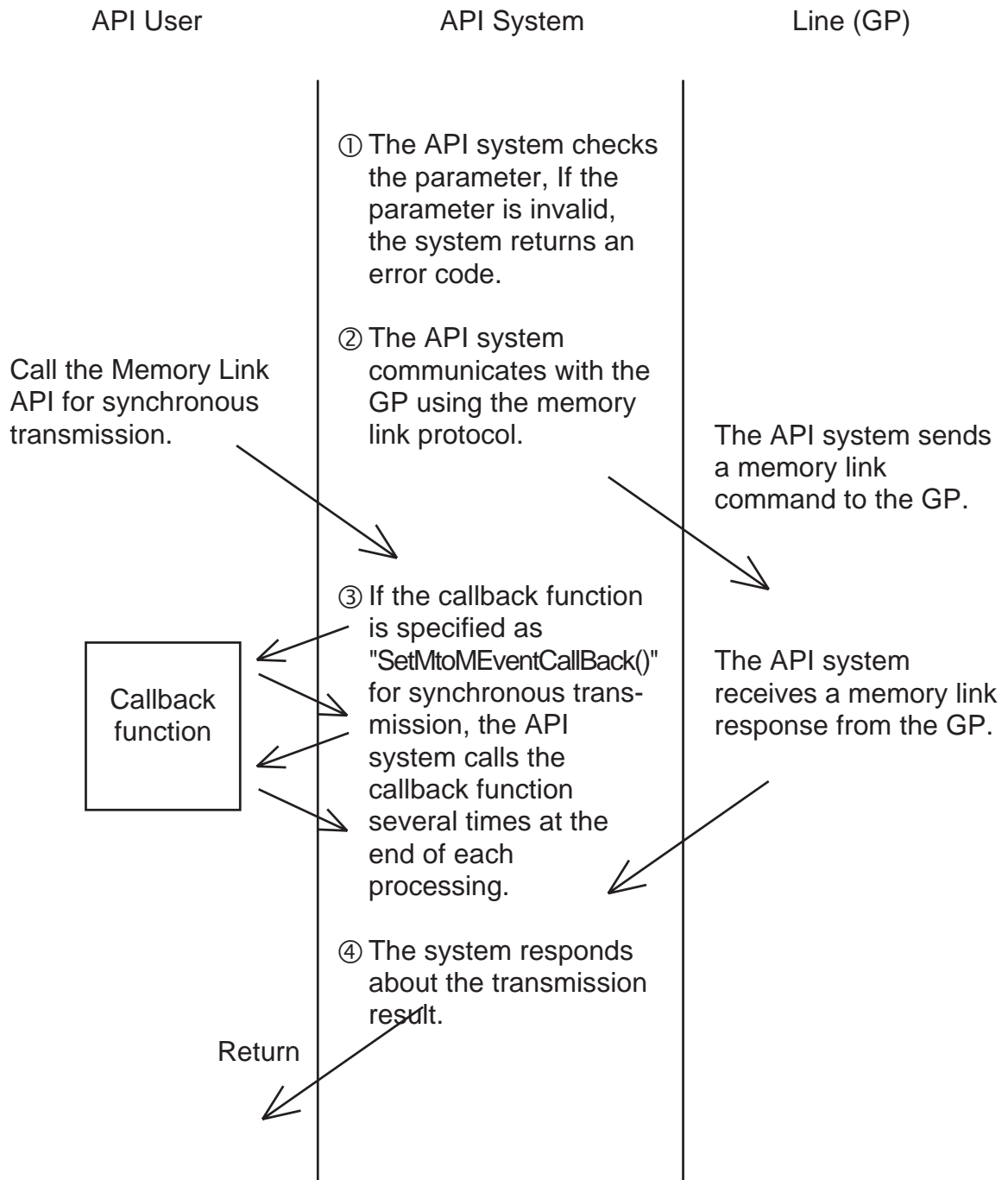
"Asynchronous Transmission" is a transmission method with which the system will return and become ready for further processing before the API's current processing is completed.

Memory Link API supports both synchronous and asynchronous transmission methods. The second parameter specifies which transmission method is to be used: synchronous or asynchronous.

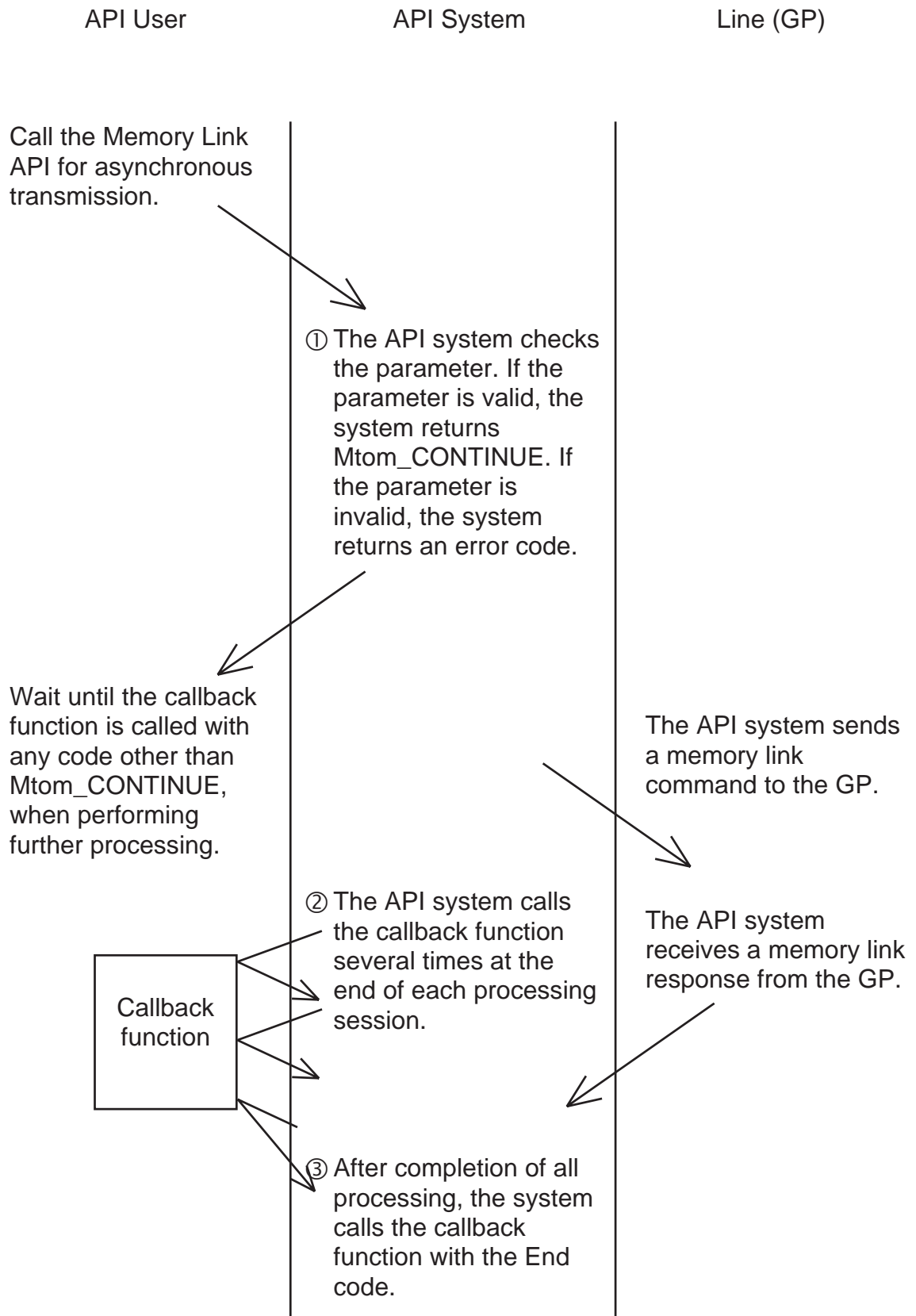


- *When the second parameter is any value other than "MTOMCALLBACK", the API system is automatically set to synchronous transmission mode.*
- *When "NULL" is specified for the MTOMCALLBACK-type argument of the second parameter, API is set to synchronous transmission mode.*
- *When any value other than "NULL" is specified for the MTOMCALLBACK-type argument of the second parameter, the API system is set to the asynchronous transmission mode, and it is judged as the callback function to perform the processing.*

■ Procedure for Synchronous Transmission



## ■ Procedure for Asynchronous Transmission



## ■ Canceling Asynchronous Transmission

To cancel the API's processing during asynchronous transmission, the following two methods are available:

### ◆ Return "FALSE"

The Memory Link API calls the callback function at the end of the current processing session. If the callback function returns FALSE in this status, the Memory Link API cancels subsequent processing safely.

### ◆ Call "CancelMtoM()"

After canceling subsequent processing, the Memory Link API calls the callback function with the "MtoM\_CANCEL" code. In this status, the socket is unstable, and the API user must then call the FreeMtoMSocket() function to free the socket. To continue communication, use another socket.

This procedure is used for forced-termination of the communication application.

## ■ Callback Function for Asynchronous Transmission

To perform asynchronous transmission, API users must prepare the callback function to learn that the processing of asynchronous transmission has been completed.

The type of the callback function is shown below.

### ◆ Syntax

```
MTOMCALLBACK FinisheMtoM(LPMtoMSOCK pMSock,int iMtoMCode)
```

### ◆ Argument

LPMtoMSOCK pMSock	Socket handle used for processing
int iMtoMCode	Processing result MTOM_OK : Processing has been normally completed. MTOM_CONTINUE : Processing is in progress. Other : Processing has been canceled due to an error.



- *The API specifies the MTOM\_CONTINUE code for the iMtoMcode parameter at the end of the current processing, and calls the callback function.*

**5-1-2****Socket Members dwUser1 and dwUser2**

The API system does not re-write "dwUser1" or "dwUser2", but the API users can freely use them.

Normally, an identifier is used for each socket.

**Example**

If you design the "C++" class that supports the memory link socket, this class can be used with a callback function when the API user calls the CreateMtoMSocket() function with the constructor of this class, creates a socket, and sets up the pointer of this class to "dwUser1" of the socket.

**Example of Operation**

- ① Set up "this" pointer of the class to "dwUser1" using the constructor of this class.
- ② Register the function (global and static function) to be called back first when the API system informs of any event using the SetMtoMEvent CallBack() function.
- ③ If any event occurs, the function registered by the SetMtoMEventCallBack() function (EventFuncJump() in this example) will be called back.
- ④ The class pointer is extracted from "dwUser 1" of the EventFuncJump() function, as if the API system had called back the OnEventFunc() function.
- ⑤ Normally, you can declare the OnEventFunc() function as a virtual function and override it for convenient use.

```

class CMtoMSock {
public:
    LPMtoMSOCK m_pMSock ;

    CMtoMSock();
    ~CMtoMSock();
    //If you need event information from the API system, override this member.
    virtual void OnEventFunc(int iCode,DWORD dwParam1,DWORD dwParam2){};
    /⑤
};

//Function to be called back when an event occurs
③
void CALLBACK EventFuncJump
(LPMtoMSOCK pMSock,int iCode,DWORD dwParam1,DWORD dwpara)
{
    CMSock* pCMSock ;

    pCMSock = (CMSock*)pMSock->swUser1 ;

    pCMSock->OnEventFunc(iCode,dwParam1,dwParam2) ;//④
}

```



```
CMSock::CMSock(DWORD dwProtocolType)
{
if( m_pMSock = ::CreateMtoMSocket(dwProtocolType) ){
    m_pMSock->dwUser1 = (DWORD )this ; //①
    ::SetMtoMEventCallBack(m_pMSock,EventFuncJump) ;//②
}
}
```

**5-1-3****Transmission Methods (Transaction Types)**

This Ethernet protocol supports the following four transmission methods (transaction types):

**■ 1:1 Transmission**

The API system communicates with one GP, ensuring the reliability of the communication result. The internal TCP/IP protocol is used.

The basic procedure for using this transaction type is as follows:

1. Create a socket using the `CreateMtoMSocket()` function.  
(When a socket is created, this transaction type is selected as the default setting.)
2. Open a connection using the `OpenMtoMLAN()` function.
3. Perform transmission using the `MtoMESC_*` function.
4. Close the connection using the `CloseMtoMLAN()` function.
5. Free the socket using the `FreeMtoMSocket()` function.

**■ Transmission to Unspecified Number of Nodes**

The API system communicates with an unspecified number of nodes without checking the response. Therefore, the reliability of the communication results cannot be ensured. Since this transmission method does not consider the processing speed of the destination nodes, transmission data may overflow during continuous transmission.

The UDP/IP broadcast protocol is used. The desired broadcast Net ID (`dwNetID`), specified in the network information area, is used as the broadcast destination Net ID.

The basic procedure for using this transaction type is as follows:

1. Create a socket using the `CreateMtoMSocket()` function.
2. Set up the transaction type by specifying "Transmission to Unspecified Number of Nodes" (`B_dwTransactionType_BroadCast`) for the `SetTransactionType()` function.
3. Open a connection using the `OpenMtoMLAN()` function. Specify `NULL` for the destination node IP address.
4. Perform transmission using the `MtoMESC_*` function.
5. Close the connection using the `CloseMtoMLAN()` function.
6. Free the socket using the `FreeMtoMSocket()` function.

## ■ Transmission to Specified Node

“The API system communicates with a specified node (that has been selected as the processing target in the network information area).

Only a response from the node that has been specified as the processing target at the head of the network information area is treated as effective. In other words, the first node is used as the representative of all nodes in the network. This transmission method is used to send a drawing command to several nodes.

If only one node has been specified in the network information area, the normal UDP/IP protocol (not for broadcast) is used. If several nodes have been specified, the UDP/IP broadcast protocol is used.

The basic procedure for using this transaction type is as follows:

1. Create a socket using the CreateMtoMSocket() function.
2. Set up the transaction type by specifying "Transmission to Unspecified Number of Nodes" (B\_dwTransactionType\_Specific) for the SetTransactionType() function.
3. Specify the target network Net ID as the broadcast target Net ID (dwNetID) in the broadcast network information (pGPNetWORKData) area.
4. Specify the destination node in the network information area.  
If the destination node is clearly known, call the MtoM\_ResizeGPNetWORKData() function to change the network information size, and specify the destination node IP address and enter B\_dwNodeStatus\_Find as the dwNodeStatus parameter for the node record of the network information area so that the node record becomes effective. If the destination node is not clearly known, call the MtoMFS\_FindNode function to search for the nodes participating in the network automatically. The search result will be added to the network information



- *During this transmission mode, the node record specified at the head of the network information area indicates the node that represents all nodes in this network.*

5. Perform transmission using the MtoMESC\_\*() function.
6. Close the connection using the CloseMtoMLAN() function.
7. Free the socket using the FreeMtoMSocket() function.

## ■ Transmission to Specified Node (checking the processing status of each node)

The API system communicates with a specified node (that has been selected as the processing target in the network information area).

After a processing command is transmitted, only a response from the node that has been specified as a processing target at the head of the network information area is treated as effective. This transmission method is different from that described in the previous page, since the processing status of each node is checked.

If only one node has been specified in the network information area, the normal UDP/IP protocol (not for broadcast) is used. If several nodes have been specified, the UDP/IP broadcast protocol is used.

1. Create a socket using the `CreateMtoMSocket()` function.
2. Set up the transaction type by specifying "Transmission to Unspecified Number of Nodes" (`B_dwTransactionType_Specific`) for the `SetTransactionType()` function.
3. Open a connection using the `OpenMtoMLAN()` function. Specify `NULL` for the destination node IP address.
4. Specify the destination node in the network information area.  
If the destination node is clearly known, call the `MtoM_ResizeGPNetWORKData()` function to change the network information size, and specify the destination node IP address and enter `B_dwNodeStatus_Find` as the `dwNodeStatus` parameter for the node record of the network information area so that the node record becomes effective. If the destination node is not clearly known, call the `MtoMFS_FindNode` function to search for the nodes participating in the network automatically. The search result will be added to the network information area.



- *During this transmission mode, the node record specified at the head of the network information area indicates the node that represents all nodes in this network.*

5. Specify `TRUE` for the (`dwCheckButton`) parameter so that this node becomes effective as the processing target in the node record of the network information area.
6. Perform transmission using the `MtoMESC_*` function.
7. Check the node status of each node record to verify that the processing of each node has been normally completed.  
When the node status is specified as `B_dwNodeStatus_Nothing`, this node record can be ignored since it is empty.

<code>B_dwNodeStatus_Find</code>	: Processing has been normally completed.
<code>B_dwNodeStatus_NotFind</code>	: Processing has been abnormally completed.
<code>B_dwNodeStatus_NonAction</code>	: This node is not the processing target. This means that <code>TRUE</code> has not been specified for <code>dwCheckButton</code> in step 5.

8. If you attempt to retry after checking the `dwNodeStatus` value, enter `TRUE` in the `dwCheckButton` parameter of the retry nodes only. With other nodes, enter `FALSE` in this parameter, and perform step 5 and the subsequent steps again.
9. Close the connection using the `CloseMtoMLAN()` function.
10. Free the socket using the `FreeMtoMSocket()` function.

## 5-1-4 Coordinate System

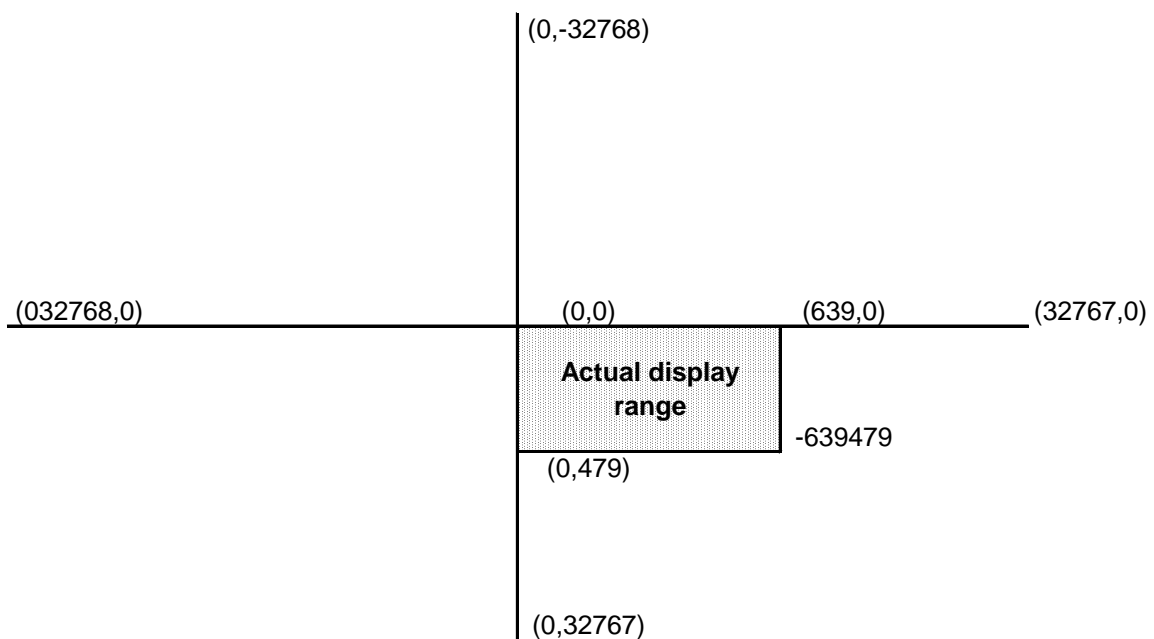
The Memory Link API's coordinate system is represented as the third quadrant with its origin (0,0) placed at the upper left corner of the GP panel screen. One pixel corresponds to one dot.

The maximum range of this coordinate system is -32768 to 32767 (16-bit, expressed by two's complement). However, the actual display range depends on the resolution of the GP being used. (If you specify any value exceeding the resolution of the GP, it will be automatically clipped.)



- *The API system's parameters are 32-bit data. However, when data are transferred from the API system to the GP, the 32-bit data are rounded off to 16-bit data.*

Example) When resolution is 640 X 480



## 5-1-5 Port Number Settings

To change the port number with the Memory Link API, use the program and change it as shown below. When the port number is not changed, 8000 is used as default.

Reference files

- MTOMAPI.H
- MSOCK.CPP

Each description is as follows;

```
***** MTOMAPI.H *****

// Definition of a socket for M to M
struct tagMtoMSOCK {
    DWORD dwStockSize ;           // The socket's effective data size (sizeof(MtoMSOCK))

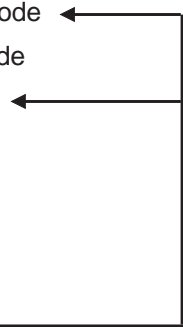
    DWORD dwProtocolType ;       // Protocol Type
#define B_ProtocolType_SIO      1    // The protocol type is MtoM SIO
#define B_ProtocolType_LAN      2    // The protocol type is MtoM LAN

    DWORD dwMtoM_Mode ;         // Operation Mode of the current socket
#define B_MtoM_Mode_ESC        0    // Drawing Mode (Default at the time of socket creation)
#define B_MtoM_Mode_DLE        1    // Data Transfer Mode (MtoMDLE_Communication())

    :

    DWORD dwIPAddress ;         // IP Address of the destination node
    DWORD dwPortNo ;           // Port No. of the destination node
    DWORD dwLocalIPAddress ;    // IP Address of the source node
    DWORD dwLocalPortNo ;      // Prot No. of the source node

    void *pDLLWork;
    DWORD dwErrorCode;         // Details of error
};
```



After calling the CreateMtoMSocket() function (refer to the sample source, MSOCK.CPP), enter arbitrary port numbers for use. (Set the port number setup in the GP.)

The default (when the port number is not changed) is 8000 for both the destination node and the source node.

```
CMSock::CMSock(DWORD dwProtocolType)
{
    if( m_pMSock = ::CreateMtoMSocket(dwProtocolType) ){
        m_pMSock->dwUser1 = (DWORD )this ;
        ::SetMtoMEventCallBack(m_pMSock,EventFuncJump) ;

        // Default is Synchronous Mode.
        SyncModeEnable();
    }
}
```

The return value of m\_pMSock is the pointer of the socket.

## 5-2 Basic Commands

This section describes the basic commands used for the Memory Link API system.

### ■ Basic Command List

Command	Action
CreateMtoMSocket	Creates a memory link socket of a specified protocol type.
OpenMtoMSocket	Opens a connection with the node specified in the memory link LAN.
CloseMtoM	Closes the connection with the destination node.
FreeMtoMSocket	Frees the socket.
SetMtoMEventCallBack	Registers the function to accept an event when any event occurs in the Memory link API system.
CancelMtoM	Cancels the currently processed asynchronous transmission.
MtoMFS_FindNode	Searches for the nodes participating in the network.
MtoM_ResizeGPNetWorkData	Changes the size of the network information of the socket.
SetTransitionType	Specifies the transmission method (transaction type).
GetTransitionType	Acquires the currently specified transmission method (transaction type).
MtoMGetLastError	Acquires the details of an error, when any error occurs.



**5-2-1****Creating Specified Protocol's Memory Link Socket**

To create a memory link socket of a specified protocol type, use the following command:  
This API system secures the resources of the socket.

**■ Syntax**

LPMtoMSOCK WINAPI CreateMtoMSocket(DWORD dwProtocolType)

**■ Return Value**

NULL : Normal termination  
Other : Failed to create a socket

**■ Argument**

DWORD dwProtocolType	Protocol type to be used	
	B_ProtocolType_SIO	: Memory link SIO
	B_ProtocolType_LAN	: Memory link LAN

## 5-2-2 Opening Connection with Node Specified in Memory Link LAN

To open a connection with the node specified in the memory link LAN, use the following command:

**■ Syntax**

int WINAPI OpenMtoMLAN(LPMtoMSOCK pMSock, MTOMCALLBACK pfFinish, LPCSTR szIPAddress)

**■ Return Value**

When pfFinish is NULL  
00 : Normal termination  
Other : Error code

When pfFinish is any value other than NULL  
MTOM\_CONTINUE : The system is normally informed of the processing request. Completion of the processing is informed when pfFinish is called back  
Other : Error code

**■ Argument**

LPMtoMSOCK pMSock                    Handle of memory link socket  
MTOMCALLBACK pfFinishNULL : This API system will not be completed until the specified processing is completed. (Synchronous Transmission)  
After the specified processing is completed, the API system will return with the processing result code.  
Other than NULL : Pointer to the function will be called back after completion of processing. When this parameter is specified, this API system will return with MTOM\_CONTINUE immediately after the processing is requested. (Asynchronous Transmission)  
After the system completes the processing, the specified callback function is called with the corresponding socket handle and processing result code.  
LPCSTR szIPAddress                    IP address of destination node (GP)  
For 1:n communication, specify NULL.



- The IP address can be specified with the following two methods:
  - ① Separating an IP address with dots:  
Example) `szipaddress="11.22.33.44";`
  - ② Specifying a node name for an IP address  
Example) `szipaddress="GP1";`

To use this method, you must prepare the HOSTS file that describes the IP addresses corresponding to the node names specified in the Windows directory.

Example) Contents of C: ¥ Windows ¥ HOSTSC

11.22.33.44	GP1
-------------	-----

## 5-2-3 Closing TCP Connection With Destination Node

To close the TCP connection with the destination node, use the following command:

### ■ Syntax

```
int WINAPI CloseMtoM(LPMtoMSOCK pMSock, MTOMCALLBACK pfFinish)
```

### ■ Return Value

Other : Handle of the created socket  
 NULL : Failed to create a socket

### ■ Argument

LPMtoMSOCK pMSock Handle of memory link socket  
 MTOMCALLBACK pfFinish NULL: This API system will not be completed until the specified processing is completed. (Synchronous Transmission)  
 After the specified processing is completed, the API system will return with the processing result code.  
 Other than NULL: Pointer to the function will be called back after the completion of processing. When this parameter is specified, this API system will return with MTOM\_CONTINUE immediately after the processing is requested. (Asynchronous Transmission)

After the system completes the processing, the specified callback function is called with the corresponding socket handle and processing result code.

## 5-2-4 Freeing a Socket

To free a socket, use the following command:

### ■ Syntax

```
int WINAPI FreeMtoMSocket(LPMtoMSOCK pMSock)
```

### ■ Return Value

Other : Handle of the created socket

NULL : Failed to create a socket

### ■ Argument

LPMtoMSOCK pMSock      Handle of memory link socket

## 5-2-5 Registering a Function (Memory Link API System Event)

To register the function to accept an event when any event occurs in the Memory Link API system, use the following command:

### ■ Syntax

```
int WINAPI SetMtoMEventCallBack(LPMtoMSOCK pMSock,MTOMEVENTBACK
pfEventFunc)
```

### ■ Return Value

Other : Handle of the created socket  
 NULL : Failed to create a socket

### ■ Argument

LPMtoMSOCK pMSock Handle of memory link socket  
 MTOMEVENTBACK pfEventFunc Function to be called back when an event occurs.  
 If NULL is specified, it will not be called back.

When any event occurs, the API system calls back the specified pfEventFunc with the socket, event code and information (2 packets of 32-bit data, max.). The function to be called back must have the following format:

During synchronous transmission, the system calls the registered callback function, at the end of the processing. During asynchronous transmission, however, the system calls the callback function specified when the processing is requested, instead of the callback function registered by this command.

```
MTOMEVENTBACK EventFunc
(LPMtoMSOCK pMtoMSOCK,int iMtoMCode,DWORD dwParam1,DWORD
dwParam2);
```

LPMtoMSOCK pMtoMSOCK	Socket handle
int iMtoMCode	Even code
DWORD dwParam1	First information
DWORD dwParam2	Second information

The following events can be called back:

Event code	First information	Second information	Contents of event
MTOM_EVENT_TOUCH	T-Tag code	Meaningless	Touch panel has been pressed.
MTOM_EVENT_CLOSED	Meaningless	Meaningless	Connection has been closed.
MTOM_CONTINUE	Meaningless	Meaningless	Synchronous transmission

**5-2-6****Canceling an Asynchronous Transmission (Current)**

To cancel the currently-processed asynchronous transmission, use the following command:

**■ Syntax**

int WINAPI CancelMtoM(LPMtoMSOCK pMSock)

**■ Return Value**

Other : Handle of the created socket

NULL : Failed to create a socket

**■ Argument**

LPMtoMSOCK pMSock          Handle of memory link socket



- *After calling this API command, the socket becomes unstable. Be sure to call the FreeMtoMSocket() function to free the socket.*

## 5-2-7 Searching For Active Network Nodes

To search for nodes participating in the network, use the following command:

The searched network information will be registered in the LPMtoMSOCK pMSock network information.

### ■ Syntax

```
int WINAPI MtoMFS_FindNode(LPMtoMSOCK pMSock, MTOMCALLBACK
pfFinish, LPCTSTR szNetID)
```

### ■ Return Value

Other : Handle of the created socket

NULL : Failed to create a socket

### ■ Argument

LPMtoMSOCK pMSock : Handle of memory link socket

MTOMCALLBACK pfFinish NULL : This API system will not be completed until the specified processing is completed. (Synchronous Transmission)

Other than NULL : After the specified processing is completed, the API system will return with the processing result code.

Pointer to the function will be called back after completion of processing. When this parameter is specified, this API system will return with MTOM\_CONTINUE immediately after the processing is requested. (Asynchronous Transmission)

After the system completes the processing, the specified callback function is called with the corresponding socket handle and processing result code.

LPCTSTR szNetID: Net ID of the network



## 5-2-8 Changing Network Information Size

To change the size of the network information, use the following command:

If there are a small number of node records to be managed, this API command enables the number of node records to be increased or reduced. Calling this API command changes the value of `pGPNetworkData` that indicates the `pMSock` network information area.

### ■ Syntax

```
int WINAPI MtoM_ResizeGPNetworkData(LPMtoMSOCK pMSock,DWORD dwNodeCounter)
```

### ■ Return Value

0 : The network information size has been normally changed.  
Other : The network information size cannot be changed due to insufficient memory capacity.

### ■ Argument

<code>LPMtoMSOCK pMSock</code>	Handle of memory link socket
<code>DWORD dwNodeCounter</code>	Desired number of node records

## 5-2-9 Transmission Method Setup (Transaction Type)

To set up the transmission method (transaction type), use the following commands:

### ■ Syntax

```
DWORD WINAPI SetTransctionType(LPMtoMSOCK pMSock,DWORD
dwTranscitionType)
```

### ■ Return Value

Setting of the transaction type yet to be changed

### ■ Argument

LPMtoMSOCK pMSock	Handle of memory link socket
DWORD dwTranscitionType	Transaction type to be changed
B_dwTransctionType_Only1:	Transmission to only one specified node for which connection has been opened. (Default setting) (The TCP/IP protocol is used.)
B_dwTransctionType_BroadCast:	Transmission to unspecified number of nodes (without response check) Since this transmission method does not consider the processing speed of the destination nodes, transmission data may overflow during continuous transmission. (The UDP/IP broadcast protocol is used.)
B_dwTransctionType_Specific:	Transmission to the specified node (that has been selected as the processing target in the network information area) Only a response from the node that has been specified as a processing target at the head of the network information area is treated as effective. In other words, the first node is used as the representative of all nodes in the network. This transmission method is used to send a drawing command to several nodes. (The UDP/IP broadcast protocol is used.)
B_dwTransctionType_SpecificCheck:	Transmission to the specified node (that has been selected as the processing target in the network information area) Only the response from the node that has been specified as a processing target at the head of the network information area is treated as effective. This transmission method is different from the method specified by B_dwTransctionType_Specific, since the processing result of each node is checked This transmission method is used to closely check the processing results of several nodes (e.g. for file transfer processing). (The UDP/IP broadcast protocol is used.)

## 5-2-10 Acquiring the Current Transmission Method

To acquire the currently-specified transmission method (transaction type), use the following commands:

### ■ Syntax

DWORD WINAPI GetTransctionType(LPMtoMSOCK pMSock)

### ■ Return Value

Setting of the currently-specified transmission method (transaction type)

For details, refer to the dwTransctionType parameter of the SetTransctionType() function.

### ■ Argument

LPMtoMSOCK pMSock

Handle of memory link socket

## 5-2-11 Acquiring Current Error Details

When an error occurs, use the following commands to acquire details of the error.

### ■ Syntax

DWORD WINAPI MtoMGetLastError(LPMtoMSOCK pMSock)

### ■ Return Value

If an error occurs when the Memory Link API system is used, details of the error are returned.

### ■ Argument

LPMtoMSOCK pMSock          Handle of memory link socket

### ■ Description

Generally, details of the error are classified into two types: error response from the GP, and error due to line trouble.

If the former type of error occurs, a value of 9999 or less will be returned. However, "0" indicates that the processing has been normally completed.

If the latter type of error occurs, a value of 10000 or more will be returned. Specifically, since the Memory Link API internally uses Winsock of Microsoft Visual C++, the error code becomes the return value.

The error codes are listed on the next page.

◆ GP-related error codes  
 ▼ **Reference** ▲ *Chapter 7: Error Messages*

## ◆ Winsock-related error codes

Code	Error	Code	Error
10004	WSAEINTR	10053	WSAECONNABORTED
10009	WSAEBADF	10054	WSAECONNRESET
10013	WSAEACCES	10055	WSAENOBUFS
10014	WSAEFAULT	10056	WSAEISCONN
10022	WSAEINVAL	10057	WSAENOTCONN
10024	WSAEMFILE	10058	WSAESHUTDOWN
10035	WSAEWOULDBLOCK	10059	WSAETOOMANYREFS
10036	WSAEINPROGRESS	10060	WSAETIMEDOUT
10037	WSAEALREADY	10061	WSAECONNREFUSED
10038	WSAENOTSOCK	10062	WSAELOOP
10039	WSAEDESTADDRREQ	10063	WSAENAMETOOLONG
10040	WSAEMSGSIZE	10064	WSAEHOSTDOWN
10041	WSAEPROTOTYPE	10065	WSAEHOSTUNREACH
10042	WSAENOPROTYPE	10066	WSAENOTEMPTY
10043	WSAEPROTONOSUPPORT	10067	WSAEPROCLIM
10044	WSAESOCKTNOSUPPORT	10068	WSAEUSERS
10045	WSAEOPNOTSUPP	10069	WSAEDQUOT
10046	WSAEPFNOSUPPORT	10070	WSAESTALE
10047	WSAEAFNOSUPPORT	10071	WSAEREMOTE
10048	WSAEADDRINUSE	10091	WSASYSNOTREADY
10049	WSAEADDRNOTAVAIL	10092	WSAVERNOTSUPPROTED
10050	WSAENETDOWN	10093	WSANOTINITIALISED
10051	WSAENETUNREACH	10101	WSAEDISCON
10052	WSAENETRESET		

## 5-3 Drawing Mode Commands

This section describes the drawing commands used for the Memory Link API system.

### ■ Drawing Command List

Command	Action
MtoMESC_W	Writes data into the System Area.
MtoMESC_R	Reads data from the System Area.
MtoMESC_T	Displays a character string.
MtoMESC_L	Draws a line.
MtoMESC_B	Draws a rectangle.
MtoMESC_S	Draws a filled rectangle.
MtoMESC_C	Draws a circle.
MtoMESC_A	Draws an arc.
MtoMESC_G	Draws a sector.
MtoMESC_P	Paints an object.
MtoMESC_I	Inquires if the touch panel has been pressed.
MtoMESC_t	Displays a character string using the extended function.
MtoMESC_l	Draws a line using the extended function.
MtoMESC_b	Draws a rectangle using the extended function.
MtoMESC_s	Draws a filled rectangle using the extended function.
MtoMESC_c	Draws a circle using the extended function.
MtoMESC_g	Draws a sector using the extended function.
MtoMESC_SetContrast	Sets brightness/contrast.
MtoMESC_GetContrast	Acquires the brightness/contrast setting.

## 5-3-1 Writing Data To System Area

To write data to the System Area, use the following command:

### ■ Syntax

```
int WINAPI MtoMESC_W
(LPMtoMSOCK pMSock, MTOMCALLBACK pfFinish, WORD wAddress, INT
iDataCount, WORD* pData)
```

### ■ Return Value

When pfFinish is NULL

00 : Normal termination  
Other : Error code

When pfFinish is any value other than NULL

MTOM\_CONTINUE: The system is normally informed of the processing request. Completion of the processing is communicated when pfFinish is called back.

Other: Error code

### ■ Argument

LPMtoMSOCK pMSock	Handle of memory link socket
MTOMCALLBACK pfFinish	<p>NULL: This API system will not be completed until the specified processing is completed. (Synchronous Transmission) After the specified processing is completed, the API system will return with the processing result code.</p> <p>Other than NULL: Pointer to the function will be called back after completion of processing. When this parameter is specified, this API system will return with MTOM_CONTINUE immediately after the processing is requested. (Asynchronous Transmission) After the system completes the processing, the specified callback function is called with the corresponding socket handle and processing result code.</p>
WORD wAddress	Specifies the address of the System Area to write data. 0000h ~ 1FFFh
INT iDataCount	Specifies the number of data packets to be written. 0001h ~ 0040h(1 ~ 64)
WORD* pData	Data to be written

## 5-3-2 Reading Data From System Area

To read data from the System Area, use the following command:

### ■ Syntax

```
int WINAPI MtoMESC_R
(LPMtoMSOCK pMSock, MTOMCALLBACK pfFinish, WORD wAddress, INT
iDataCount, WORD pwoData)
```

### ■ Return Value

When pfFinish is NULL

00 : Normal termination  
Other : Error code

When pfFinish is any value other than NULL

MTOM\_CONTINUE : The system is informed of the processing request.  
Completion of the processing is communicated when  
pfFinish is called back.  
Other : Error code

### ■ Argument

LPMtoMSOCK pMSock	Handle of memory link socket
MTOMCALLBACK pfFinish	NULL : This API system will not be completed until the specified processing is completed. (Synchronous Transmission) After the specified processing is completed, the API system will return with the processing result code. Other than NULL : Pointer to the function will be called back after completion of processing. When this parameter is specified, this API system will return with MTOM_CONTINUE immediately after the processing is requested. (Asynchronous Transmission) After the system completes the processing, the specified callback function is called with the corresponding socket handle and processing result code.
WORD wAddress	Specifies the address of the System Area to read the data. 0000h ~ 1FFFh
INT iDataCount	Specifies the number of data packets to be read. 0001h ~ 0040h (1 ~ 64)
WORD pwoData	Location to store read data.



• *This API system does not check the buffer size specified by pwoData. The API users must prepare a sufficient buffer size.*



### 5-3-3 Displaying a Character String

To display a character string, use the following commands:

#### ■ Syntax

```
int WINAPI MtoMESC_T
(LPMtoMSOCK pMSock, MTOMCALLBACK pfFinish, GPCOLOR
cDisplayColor, GPCOLOR cBackColor,
INT iX1, INT iY1, GPFONTSIZE cFontSize, CHAR* szString)
```

#### ■ Return Value

When pfFinish is NULL

00 : Normal termination  
Other : Error code

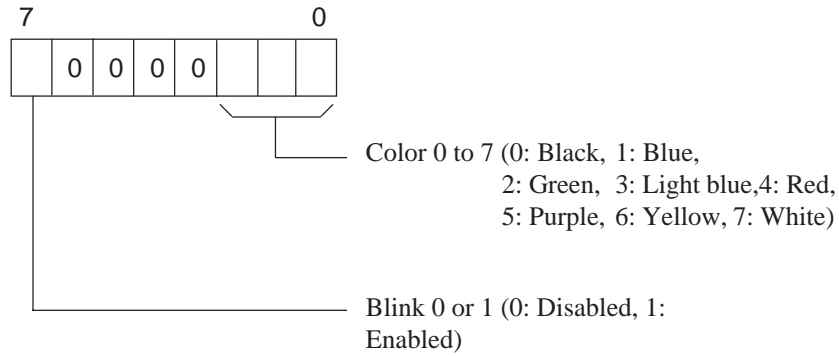
When pfFinish is any value other than NULL

MTOM\_CONTINUE : The system is normally informed of the processing request. Completion of the processing is communicated when pfFinish is called back.  
Other : Error code

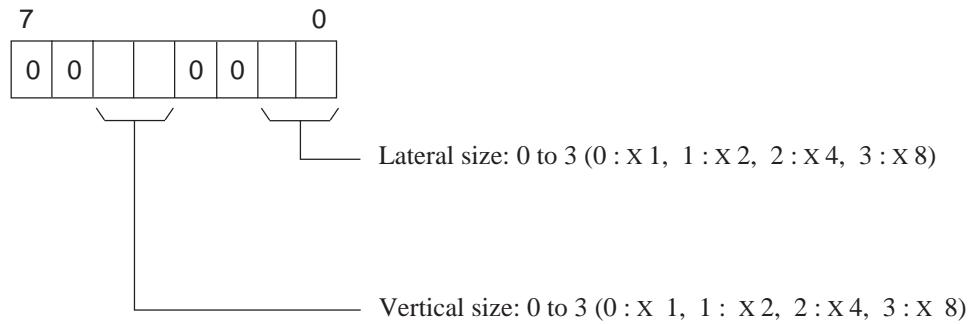
#### ■ Argument

LPMtoMSOCK pMSock Handle of memory link socket  
MTOMCALLBACK pfFinishNULL : This API system will not be completed until the specified processing is completed. (Synchronous Transmission) After the specified processing is completed, the API system will return with the processing result code.  
Other than NULL : Pointer to the function will be called back after completion of processing. When this parameter is specified, this API system will return with MTOM\_CONTINUE immediately after the processing is requested. (Asynchronous Transmission) After the system completes the processing, the specified callback function is called with the corresponding socket handle and processing result code.

GPCOLOR cDisplayColor Specifies the display color in the following range.  
 GPCOLOR cBackColor Specifies the background color in the following range.



INT iX1 Specifies the X-coordinate in the following range:  
 INT iY1 Specifies the Y-coordinate in the following range:  
 X-coordinate: 0000h ~ (0 ~)  
 Y-coordinate: 0000h ~ (0 ~)



GPFONTSIZE cFontSize Specifies the font size in the following range:  
 CHAR\* szString Specifies Shift codes.  
 An ASCII character uses one byte. A double-sized character uses 2 bytes.

## 5-3-4 Drawing a Line

To draw a line, use the following command:

### ■ Syntax

```
int WINAPI MtoMESC_L
(LPMtoMSOCK pMSock, MTOMCALLBACK pfFinish, GPCOLOR
cDisplayColor, GPCOLOR cBackColor, GPLINESTYLE cLineStyle, INT iX1, INT
iY1, INT iX2, INT iY2)
```

### ■ Return Value

When pfFinish is NULL

00 : Normal termination

Other : Error code

When pfFinish is any value other than NULL

MTOM\_CONTINUE : The system is normally informed of the processing request.

Completion of the processing is communicated when pfFinish is called back.

Other : Error code

### ■ Argument

LPMtoMSOCK pMSock

Handle of memory link socket

MTOMCALLBACK pfFinishNULL: This API system will not be completed until the specified processing is completed. (Synchronous Transmission)

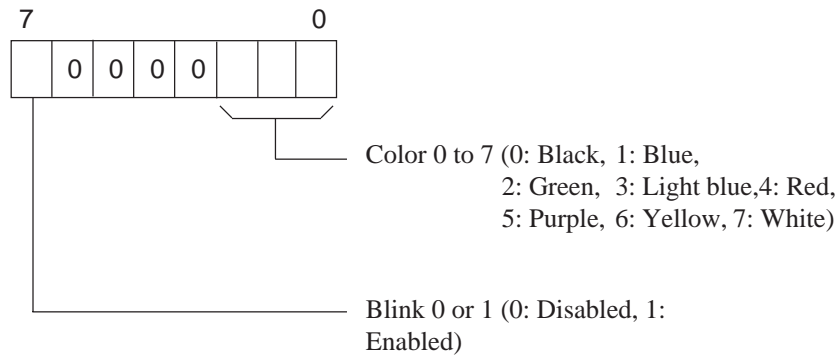
After the specified processing is completed, the API system will return with the processing result code.

Other than NULL: Pointer to the function will be called back after completion of processing. When this parameter is specified, this API system will return with MTOM\_CONTINUE immediately after the processing is requested.

(Asynchronous Transmission)

After the system completes the processing, the specified callback function is called with the corresponding socket handle and processing result code.

GPCOLOR cDisplayColor Specifies the display color in the following range:  
 GPCOLOR cBackColor Specifies the background color in the following range:



GPLINESTYLE cLineStyle Specifies the line type in the following range:  
 (00: \_\_\_\_\_01: - - - 02: \_\_\_ \_03: \_ \_ \_ \_  
 04: \_\_\_\_\_05: - - - 06: \_\_\_ \_07: \_ \_ \_ \_)

INT iX1 Specifies the X-coordinate of the start point in the following range:

INT iY1 Specifies the Y-coordinate of the start point in the following range:

INT iX2 Specifies the X-coordinate of the end point in the following range:

INT iY2 Specifies the Y-coordinate of the end point in the following range:  
 X-coordinate: 0000h ~ (0 ~)  
 Y-coordinate: 0000h ~ (0 ~)

## 5-3-5 Drawing a Rectangle

To draw a rectangle, use the following commands:

### ■ Syntax

```
int WINAPI MtoMESC_B
(LPMtoMSOCK pMSock, MTOMCALLBACK pfFinish, GPCOLOR
cDisplayColor, GPCOLOR cBackColor, GPLINESTYLE cLineStyle, INT iX1, INT
iY1, INT iX2, INT iY2)
```

### ■ Return Value

When pfFinish is NULL

00 : Normal termination

Other : Error code

When pfFinish is any value other than NULL

MTOM\_CONTINUE : The system is informed of the processing request.  
Completion of the processing is communicated when  
pfFinish is called back.

Other : Error code

### ■ Argument

LPMtoMSOCK pMSock

Handle of memory link socket

MTOMCALLBACK pfFinishNULL: This API system will not be completed until

the specified processing is completed.

(Synchronous Transmission)

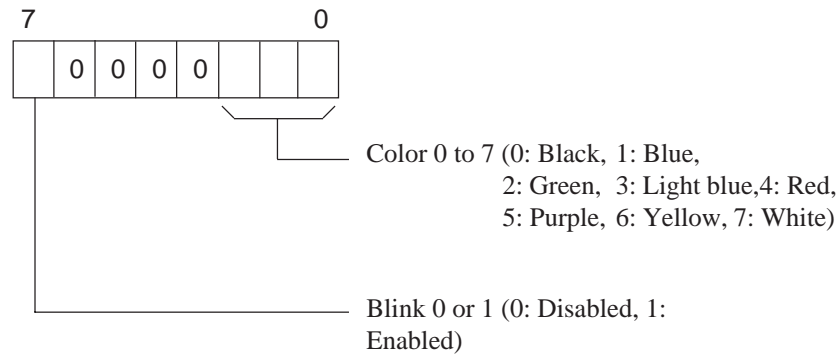
After the specified processing is completed,  
the API system will return with the processing  
result code.

Other than NULL: Pointer to the function will be called back after  
completion of processing. When this  
parameter is specified, this API system will  
return with MTOM\_CONTINUE immediately  
after the processing is requested.

(Asynchronous Transmission)

After the system completes the processing, the  
specified callback function is called with the  
corresponding socket handle and processing  
result code.

GPCOLOR cDisplayColor Specifies the display color in the following range:  
 GPCOLOR cBackColor Specifies the background color in the following range:



GPLINESTYLE cLineStyle Specifies the line type in the following range:  
 (00: \_\_\_\_\_ 01: - - - - 02: \_ \_ \_ \_ 03: \_ \_ \_ \_  
 04: \_\_\_\_\_ 05: - - - - 06: \_ \_ \_ \_ 07: \_ \_ \_ \_)

INT iX1 Specifies the X-coordinate of the start point in the following range:

INT iY1 Specifies the Y-coordinate of the start point in the following range:

INT iX2 Specifies the X-coordinate of the end point in the following range:

INT iY2 Specifies the Y-coordinate of the end point in the following range:  
 X-coordinate: 0000h ~ (0 ~)  
 Y-coordinate: 0000h ~ (0 ~)

## 5-3-6 Drawing Filled Rectangle

To draw a filled rectangle, use the following commands:

### ■ Syntax

```
int WINAPI MtoMESC_S
(LPMtoMSOCK pMSock, MTOMCALLBACK pfFinish, GPCOLOR
cDisplayColor, GPCOLOR cBackColor, INT iX1, INT iY1, INT iX2, INT iY2, GPTILE
cTile)
```

### ■ Return Value

When pfFinish is NULL

00 : Normal termination  
Other : Error code

When pfFinish is any value other than NULL

MTOM\_CONTINUE : The system is informed of the processing request.  
Completion of the processing is communicated when  
pfFinish is called back.  
Other : Error code

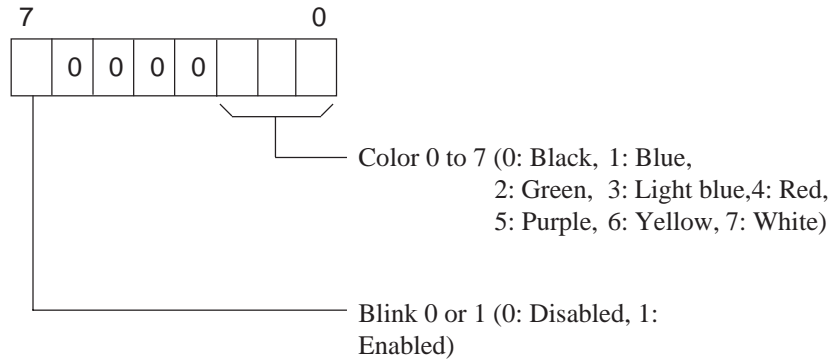
### ■ Argument

LPMtoMSOCK pMSock Handle of memory link socket

MTOMCALLBACK pfFinishNULL: This API system will not be completed until the specified processing is completed. (Synchronous Transmission)  
After the specified processing is completed, the API system will return with the processing result code.

Other than NULL: Pointer to the function will be called back after completion of processing. When this parameter is specified, this API system will return with MTOM\_CONTINUE immediately after the processing is requested. (Asynchronous Transmission)  
After the system completes the processing, the specified callback function is called with the corresponding socket handle and processing result code.

GPCOLOR cDisplayColor Specifies the display color in the following range:



- GPCOLOR cBackColor Specifies the background color in the following range:
- INT iX1 Specifies the X-coordinate of the start point in the following range:
- INT iY1 Specifies the Y-coordinate of the start point in the following range:
- INT iX2 Specifies the X-coordinate of the end point in the following range:
- INT iY2 Specifies the Y-coordinate of the end point in the following range:  
X-coordinate: 0000h ~ (0 ~)  
Y-coordinate: 0000h ~ (0 ~)
- GPTILEcTile Specifies the tiling pattern <sup>\*1</sup> in the range of 00h to 08h.

<sup>\*1</sup> Tiling Patterns

Tiling pattern No.	Tiling pattern	Tiling pattern No.	Tiling pattern	Tiling pattern No.	Tiling pattern
0		3		6	
1		4		7	
2		5		8	



## 5-3-7 Drawing a Circle

To draw a circle, use the following commands:

### ■ Syntax

```
int WINAPI MtoMESC_C
(LPMtoMSOCK pMSock, MTOMCALLBACK pfFinish, GPCOLOR
cDisplayColor, GPCOLOR cBackColor, GPLINESTYLE cLineStyle, INT iX1, INT
iY1, INT iRadius)
```

### ■ Return Value

When pfFinish is NULL

00 : Normal termination  
Other : Error code

When pfFinish is any value other than NULL

MTOM\_CONTINUE : The system is informed of the processing request.  
Completion of the processing is communicated when  
pfFinish is called back.  
Other : Error code

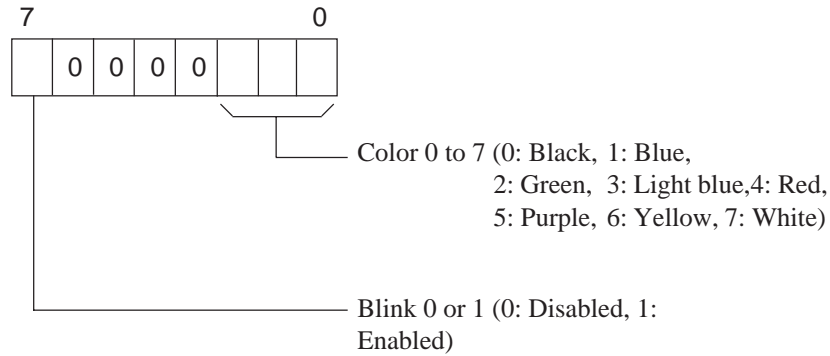
### ■ Argument

LPMtoMSOCK pMSock Handle of memory link socket

MTOMCALLBACK pfFinishNULL: This API system will not be completed until the specified processing is completed. (Synchronous Transmission)  
After the specified processing is completed, the API system will return with the processing result code.

Other than NULL: Pointer to the function will be called back after completion of processing. When this parameter is specified, this API system will return with MTOM\_CONTINUE immediately after the processing is requested. (Asynchronous Transmission)  
After the system completes the processing, the specified callback function is called with the corresponding socket handle and processing result code.

GPCOLOR cDisplayColor Specifies the display color in the following range:  
 GPCOLOR cBackColor Specifies the background color in the following range:



GPLINESTYLE cLineStyle Specifies the line type in the following range:  
 00h ~ 03h  
 (00: ——— 01: - - - - 02: — - — 03: — - - - )

INT iX1 Specifies the X-coordinate of the center in the following range:

INT iY1 Specifies the Y-coordinate of the center in the following range:  
 X-coordinate: 0000h ~ (0 ~)  
 Y-coordinate: 0000h ~ (0 ~)

INT iRadius Specifies the radius in the range of 0001h ~ (1 ~).

## 5-3-8 Drawing an Arc

To draw an arc, use the following command:

### ■ Syntax

```
int WINAPI MtoMESC_A
(LPMtoMSOCK pMSock,MTOMCALLBACK pfFinish,GPCOLOR
cDisplayColor,GPCOLOR cBackColor,GPLINESTYLE cLineStyle,INT iX1,INT
iY1, INT iRadius,INT iStartAngle,INT iEndAngle)
```

### ■ Return Value

When pfFinish is NULL

00 : Normal termination  
Other : Error code

When pfFinish is any value other than NULL

MTOM\_CONTINUE : The system is informed of the processing request.  
Completion of the processing is communicated when  
pfFinish is called back.  
Other : Error code

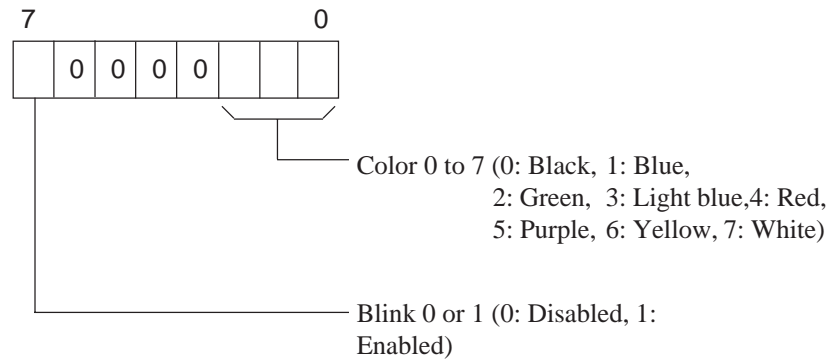
### ■ Argument

LPMtoMSOCK pMSock Handle of memory link socket

MTOMCALLBACK pfFinishNULL: This API system will not be completed until the specified processing is completed. (Synchronous Transmission)  
After the specified processing is completed, the API system will return with the processing result code.

Other than NULL: Pointer to the function will be called back after completion of processing. When this parameter is specified, this API system will return with MTOM\_CONTINUE immediately after the processing is requested. (Asynchronous Transmission)  
After the system completes the processing, the specified callback function is called with the corresponding socket handle and processing result code.

GPCOLOR cDisplayColor Specifies the display color in the following range:  
 GPCOLOR cBackColor Specifies the background color in the following range:



GPLINESTYLE cLineStyle Specifies the line type in the following range:  
 00h ~ 03h  
 (00:———— 01: - - - - 02: — - — 03: — - - - )

INT iX1 Specifies the X-coordinate of the center in the following range:

INT iY1 Specifies the Y-coordinate of the center in the following range:  
 X-coordinate: 0000h ~ (0 ~)  
 Y-coordinate: 0000h ~ (0 ~)

INT iRadius Specifies the radius in the range of 0001h ~ (1 ~).

INT iStartAngle Specifies the start angle in the following range:

INT iEndAngle Specifies the end angle in the following range:  
 Angle: 0000h to 0168h (0 to 360)



- *The drawing direction is counter-clockwise.*
- *Do not enter the same value for both the start and end angles.*

## 5-3-9 Drawing a Sector

To draw a sector, use the following commands:

### ■ Syntax

```
int WINAPI MtoMESC_G
(LPMtoMSOCK pMSock, MTOMCALLBACK pfFinish, GPCOLOR
cDisplayColor, GPCOLOR cBackColor, GPLINESTYLE cLineStyle, INT iX1, INT
iY1, INT iRadius, INT iStartAngle, INT iEndAngle)
```

### ■ Return Value

When pfFinish is NULL

00 : Normal termination  
Other : Error code

When pfFinish is any value other than NULL

MTOM\_CONTINUE : The system is informed of the processing request.  
Completion of the processing is communicated when  
pfFinish is called back.  
Other : Error code

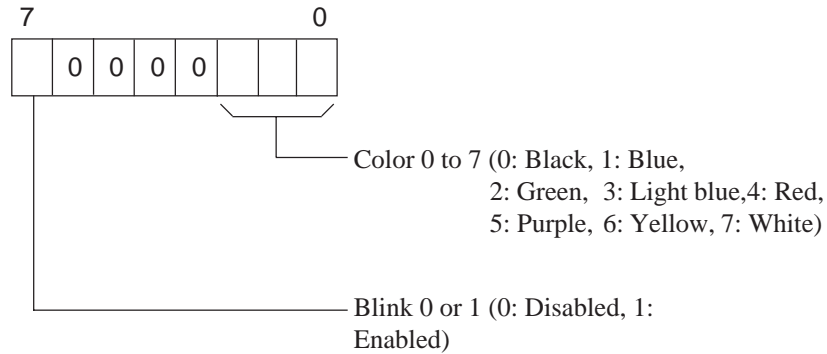
### ■ Argument

LPMtoMSOCK pMSock Handle of memory link socket

MTOMCALLBACK pfFinish NULL: This API system will not be completed until the specified processing is completed. (Synchronous Transmission)  
After the specified processing is completed, the API system will return with the processing result code.

Other than NULL: Pointer to the function will be called back after completion of processing. When this parameter is specified, this API system will return with MTOM\_CONTINUE immediately after the processing is requested. (Asynchronous Transmission)  
After the system completes the processing, the specified callback function is called with the corresponding socket handle and processing result code.

GPCOLOR cDisplayColor Specifies the display color in the following range:  
 GPCOLOR cBackColor Specifies the background color in the following range:



GPLINESTYLE cLineStyle Specifies the line type in the following range:  
 00h ~ 03h  
 (00:————— 01: - - - - - 02: — — — — 03: — — — —)

INT iX1 Specifies the X-coordinate of the center in the following range:

INT iY1 Specifies the Y-coordinate of the center in the following range:  
 X-coordinate: 0000h ~ (0 ~)  
 Y-coordinate: 0000h ~ (0 ~)

INT iRadius Specifies the radius in the range of 0001h ~ (1 ~).

INT iStartAngle Specifies the start angle in the following range:

INT iEndAngle Specifies the end angle in the following range:  
 Angle: 0000h to 0168h (0 to 360)



- *Do not enter the same value for both the start and end angles.*
- *The drawing direction is counter-clockwise.*

## 5-3-10 Filling an Object

To fill an object, use the following command:

### ■ Syntax

```
int WINAPI MtoMESC_P
(LPMtoMSOCK pMSock, MTOMCALLBACK pfFinish, GPCOLOR
cDisplayColor, GPCOLOR cBackColor, GPTILE cTile, GPCOLOR cLimitColor, INT
iX1, INT iY1)
```

### ■ Return Value

When pfFinish is NULL

00 : Normal termination  
Other : Error code

When pfFinish is any value other than NULL

MTOM\_CONTINUE : The system is informed of the processing request.  
Completion of the processing is communicated when  
pfFinish is called back.  
Other : Error code

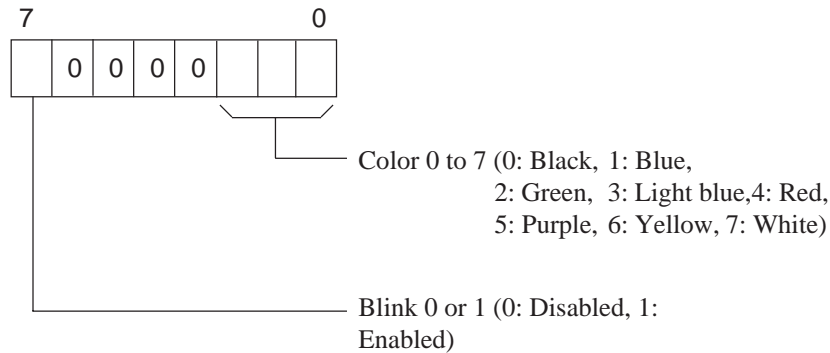
### ■ Argument

LPMtoMSOCK pMSock Handle of memory link socket

MTOMCALLBACK pfFinishNULL: This API system will not be completed until the specified processing is completed. (Synchronous Transmission)  
After the specified processing is completed, the API system will return with the processing result code.

Other than NULL: Pointer to the function will be called back after completion of processing. When this parameter is specified, this API system will return with MTOM\_CONTINUE immediately after the processing is requested. (Asynchronous Transmission)  
After the system completes the processing, the specified callback function is called with the corresponding socket handle and processing result code.

GPCOLOR cDisplayColor Specifies the display color in the following range:  
 GPCOLOR cBackColor Specifies the background color in the following range:  
 GPCOLOR cLimitColor Specifies the border line color in the following range:



• *Be sure to set the border line blink function to disabled.*

GPTILE cTile

Specifies the tiling pattern in the range of 00h to 08h.

**Reference** *"Tiling Patterns" in "5-3-6 Drawing Painted Rectangle"*

INT iX1

Specifies the X-coordinate of the start point in the following range:

INT iY1

Specifies the Y-coordinate of the start point in the following range:

X-coordinate: 0000h ~ (0 ~)

Y-coordinate: 0000h ~ (0 ~)



## 5-3-11 Inquisition of Touch Panel Input

To inquire whether the touch panel has been pressed or not, use the following commands:

After the processing of this API system is normally completed, check the pbHave value.

If this area is TRUE, refer to the pdwCode value.

### ■ Syntax

```
int WINAPI MtoMESC_I
(LPMtoMSOCK pMSock, MTOMCALLBACK pfFinish, BOOL* pbHave, DWORD
*pdwCode)
```

### ■ Return Value

When pfFinish is NULL

00 : Normal termination

Other : Error code

When pfFinish is any value other than NULL

MTOM\_CONTINUE : The system is informed of the processing request.  
Completion of the processing is communicated when  
pfFinish is called back.

Other : Error code

### ■ Argument

LPMtoMSOCK pMSock	Handle of memory link socket
MTOMCALLBACK pfFinish	<p>NULL: This API system will not be completed until the specified processing is completed. (Synchronous Transmission) After the specified processing is completed, the API system will return with the processing result code.</p> <p>Other than NULL: Pointer to the function will be called back after completion of processing. When this parameter is specified, this API system will return with MTOM_CONTINUE immediately after the processing is requested. (Asynchronous Transmission) After the system completes the processing, the specified callback function is called with the corresponding socket handle and processing result code.</p>
BOOL* pbHave	<p>This area indicates whether the touch panel has been pressed or not. If this area is "TRUE" after the processing of this API system is completed, this means that the touch panel has been pressed, and the corresponding code is entered for the pdwCode.</p>
DWORD *pdwCode	When the touch panel has been pressed, the corresponding code is entered in this area.

## 5-3-12 Displaying a Character String Using an Extended Function

To display a character string by specifying extended functions, use the following commands: The Turn, Direction and Highlighting functions are added as the extended functions.

### ■ Syntax

```
int WINAPI MtoMESC_t
(LPMtoMSOCK pMSock, MTOMCALLBACK pfFinish, GPCOLOR
cDisplayColor, GPCOLORBackColor,
GPFONT cFont, INT iTurn, INT iDirection, INT iHalfcentering, INT
iEmphasis, INT iSculpture,
INT iX1, INT iY1, GPFONTSIZE cFontSize, CHAR* szString)
```

### ■ Return Value

When pfFinish is NULL

00 : Normal termination  
Other : Error code

When pfFinish is any value other than NULL

MTOM\_CONTINUE : The system is informed of the processing request.  
Completion of the processing is communicated when  
pfFinish is called back.  
Other : Error code

### ■ Argument

LPMtoMSOCK pMSock

Handle of memory link socket

MTOMCALLBACK pfFinishNULL: This API system will not be completed until the specified processing is completed.

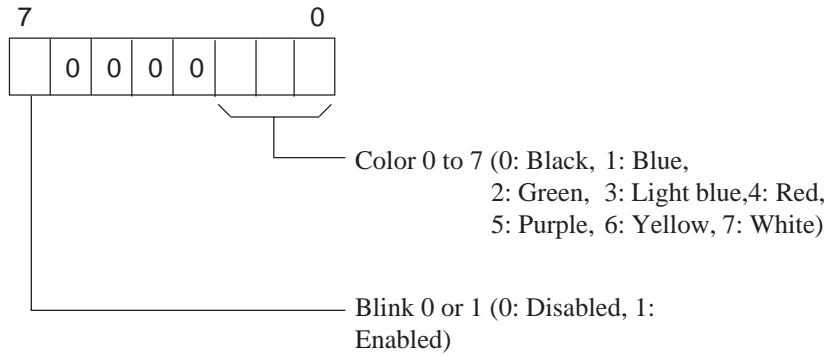
(Synchronous Transmission)

After the specified processing is completed, the API system will return with the processing result code.

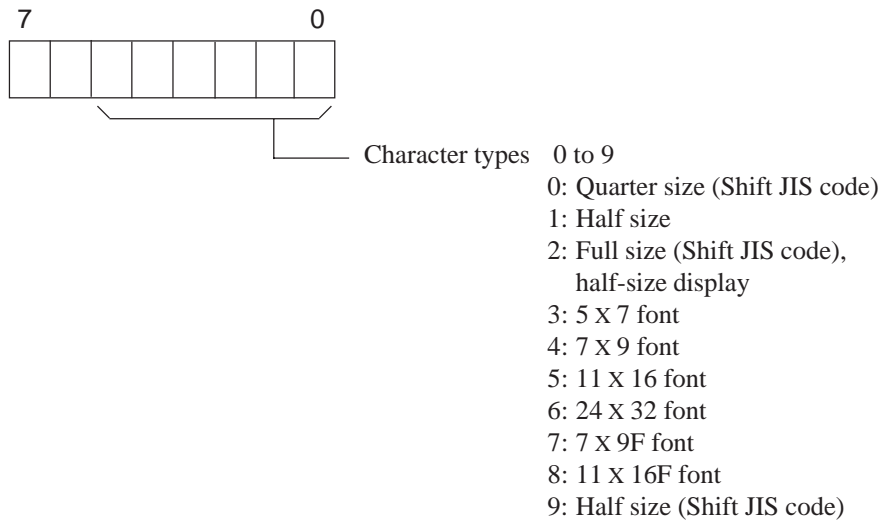
Other than NULL: Pointer to the function will be called back after completion of processing. When this parameter is specified, this API system will return with MTOM\_CONTINUE immediately after the processing is requested. (Asynchronous Transmission)

After the system completes the processing, the specified callback function is called with the corresponding socket handle and processing result code.

GPCOLOR cDisplayColor Specifies the display color in the following range:  
 GPCOLOR cBackColor Specifies the background color in the following range:



PFONT cFont Specifies the character type in the following range:



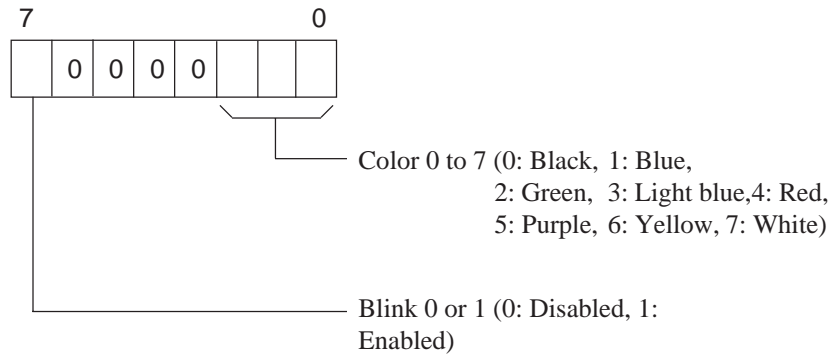
INT iTurn Specifies the character angle in the following range:  
 00f to 03h (0: 0°, 1: 90°, 2: 180°, 3: 270°)

INT iDirection Specifies the characters' direction in the following range:  
 00h to 01h (0: Landscape, 1: Portrait)

INT iHalfcentering Specifies the centering of half-size characters in the following range:  
 00h to 01h (0: Disabled, 1: Enabled: Effective only for the portrait mode)

INT iEmphasis Specifies the highlighting mode in the following range:  
 00h to 02h (0: Normal, 1: Highlighted 2: Relief)

**INT iSculpture** Specifies the relief color in the following range:



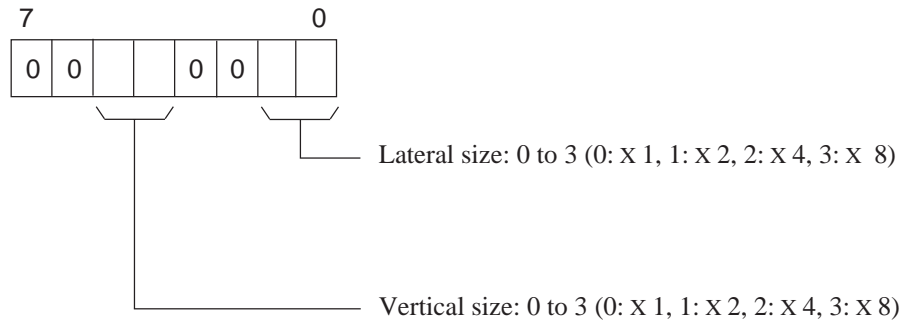
**INT iX1** Specifies the X-coordinate in the following range:

**INT iY1** Specifies the Y-coordinate in the following range:

X-coordinate: 0000h ~ (0 ~)

Y-coordinate: 0000h ~ (0 ~)

**GPFONTSIZE cFontSize** Specifies the font size in the following range:



**CHAR\* szString**

Specifies Shift JIS codes.

An ANK character uses one byte. A chinese character uses 2 bytes.

## 5-3-13 Drawing a Line Using Extended Function

To draw a line by specifying the extended function, use the following commands:  
The arrow function is added as the extended function.

### ■ Syntax

```
int WINAPI MtoMESC_1
(LPMtoMSOCK pMSock, MTOMCALLBACK pfFinish, GPCOLOR cDisplayColor,
GPCOLORcBackColor, GPLINESTYLE cLineStyle, INT iArrowPattern, INT
ArrowDirection,
INT iX1, INT iY1, INT iX2, INT iY2)
```

### ■ Return Value

When pfFinish is NULL

00 : Normal termination  
Other : Error code

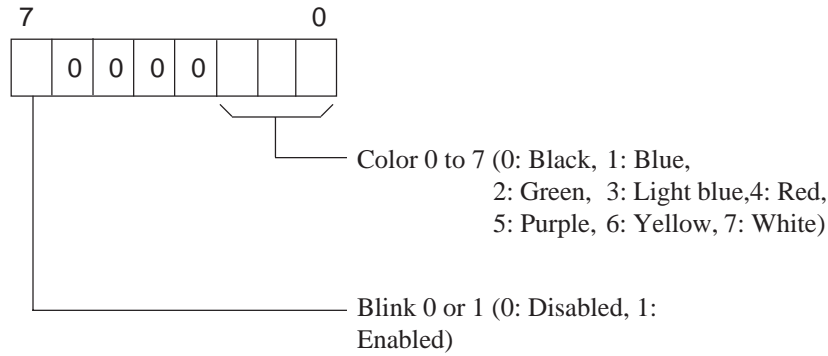
When pfFinish is any value other than NULL

MTOM\_CONTINUE : The system is informed of the processing request.  
Completion of the processing is communicated when  
pfFinish is called back.  
Other : Error code

### ■ Argument

LPMtoMSOCK pMSock : Handle of memory link socket  
MTOMCALLBACK pfFinishNULL: This API system will not be completed until  
the specified processing is completed.  
(Synchronous Transmission)  
After the specified processing is completed,  
the API system will return with the processing  
result code.  
Other than NULL: Pointer to the function will be called back after  
completion of processing. When this  
parameter is specified, this API system will  
return with MTOM\_CONTINUE immediately  
after the processing is requested.  
(Asynchronous Transmission)  
After the system completes the processing, the  
specified callback function is called with the  
corresponding socket handle and processing  
result code.

GPCOLOR cDisplayColor Specifies the display color in the following range:  
 GPCOLOR cBackColor Specifies the background color in the following range:



GPLINESTYLE cLineStyle Specifies the line type in the following range:  
 00 to 07h  
 (00: ——— 01: - - - - 02: — - - - 03: — - - - )  
 04: ——— 05: - - - - 06: — - - - 07: — - - - )

INT iArrowPattern Specifies the arrow pattern in the following range:  
 00 to 03h  
 (00: None, 01: Arrow 1, 02: Reserved, 03: Reserved)

INT iCornerRadius Specifies the arrow direction in the following range:  
 00 to 01h (00: Both ends, 01: Endpoint)

INT iX1 Specifies the X-coordinate of the start point in the following range:

INT iY1 Specifies the Y-coordinate of the start point in the following range:

INT iX2 Specifies the X-coordinate of the end point in the following range:

INT iY2 Specifies the Y-coordinate of the end point in the following range:  
 X-coordinate: 0000h ~ (0 ~)  
 Y-coordinate: 0000h ~ (0 ~)

## 5-3-14 Drawing a Rectangle Using an Extended Function

To draw a rectangle by specifying the extended function, use the following command:  
The chamfering function is added as the extended function.

### ■ Syntax

```
int WINAPI MtoMESC_b
(LPMtoMSOCK pMSock,MTOMCALLBACK pfFinish,GPCOLOR
cDisplayColor,GPCOLORcBackColor,
GPLINESTYLE cLineStyle,INT iCornerDirection,INT iCornerRadius,
INT iX1,INT iY1,INT iX2,INT iY2)
```

### ■ Return Value

When pfFinish is NULL

00 : Normal termination  
Other : Error code

When pfFinish is any value other than NULL

MTOM\_CONTINUE : The system is informed of the processing request.  
Completion of the processing is communicated when  
pfFinish is called back.  
Other : Error code

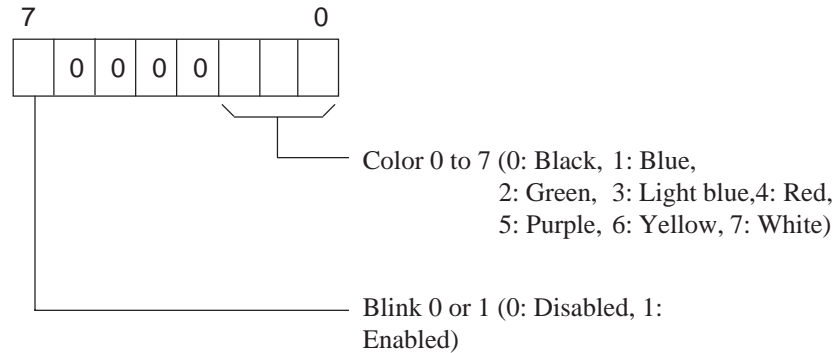
### ■ Argument

LPMtoMSOCK pMSock : Handle of memory link socket

MTOMCALLBACK pfFinishNULL: This API system will not be completed until the specified processing is completed. (Synchronous Transmission)  
After the specified processing is completed, the API system will return with the processing result code.

Other than NULL: Pointer to the function will be called back after completion of processing. When this parameter is specified, this API system will return with MTOM\_CONTINUE immediately after the processing is requested. (Asynchronous Transmission)  
After the system completes the processing, the specified callback function is called with the corresponding socket handle and processing result code.

GPCOLOR cDisplayColor Specifies the display color in the following range:  
 GPCOLOR cBackColor Specifies the background color in the following range:



GPLINESTYLE cLineStyle Specifies the line type in the following range:  
 (00: ——— 01: - - - - 02: — — — 03: — — — —  
 04: ——— 05: - - - - 06: — — — 07: — — — —)



- Line types 0 to 3 are 1-dot lines, line type 8 is 3-dot line and line type 9 is 5-dot line.

INT iArrowPattern Specifies the arrow pattern in the following range:  
 00 to 02h (00: None, 01: Curve, 02: Line)

INT iCornerRadius Specifies the chamfering radius in the following range:  
 00 to 32h

INT iX1 Specifies the X-coordinate of the start point in the following range:

INT iY1 Specifies the Y-coordinate of the start point in the following range:

INT iX2 Specifies the X-coordinate of the end point in the following range:

INT iY2 Specifies the Y-coordinate of the end point in the following range:  
 X-coordinate: 0000h ~ (0 ~)  
 Y-coordinate: 0000h ~ (0 ~)



## 5-3-15 Drawing a Filled Rectangle Using an Extended Function

To draw a filled rectangle by specifying the extended function, use the following command:  
The chamfering function is added as the extended function.

### ■ Syntax

```
int WINAPI MtoMESC_s
(LPMtoMSOCK pMSock, MTOMCALLBACK pfFinish, GPCOLOR
cDisplayColor, GPCOLOR cBackColor, GPTILE cTile, INT iCornerDirection, INT
iCornerRadius, CINT iX1, INT iY1, INT iX2, INT iY2,)
```

### ■ Return Value

When pfFinish is NULL

00 : Normal termination  
Other : Error code

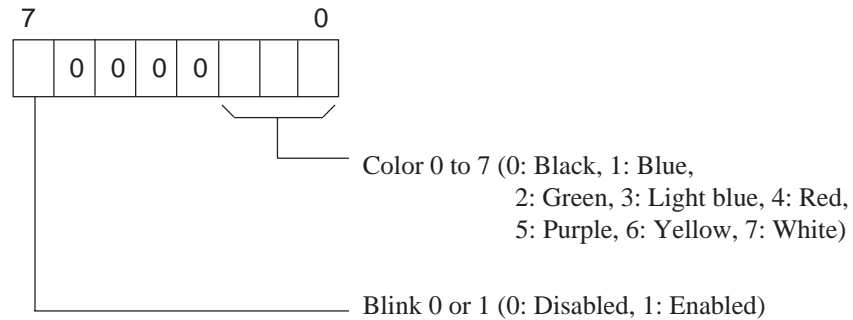
When pfFinish is any value other than NULL

MTOM\_CONTINUE : The system is informed of the processing request.  
Completion of the processing is communicated when  
pfFinish is called back.  
Other : Error code

### ■ Argument

LPMtoMSOCK pMSock : Handle of memory link socket  
MTOMCALLBACK pfFinishNULL: This API system will not be completed until the  
specified processing is completed.  
(Synchronous Transmission)  
After the specified processing is completed, the  
API system will return with the processing result  
code.  
Other than NULL: Pointer to the function will be called back after  
completion of processing. When this parameter  
is specified, this API system will return with  
MTOM\_CONTINUE immediately after the  
processing is requested. (Asynchronous  
Transmission)  
After the system completes the processing, the  
specified callback function is called with the  
corresponding socket handle and processing  
result code.

GPCOLOR cDisplayColor Specifies the display color in the following range:  
 GPCOLOR cBackColor Specifies the background color in the following range:



GPTILE cTile Specifies the tiling pattern in the range of 00h to 08h.

**Reference** 5-3-6 Tiling Patterns in "5-3-6  
 Drawing a Filled Rectangle"

INT iCornerDirection Specifies the chamfering type in the following range:  
 00 to 02h (00: None, 01: Curve, 02: Line)

INT iCornerRadius Specifies the chamfering radius in the following range:  
 00 to 32h

INT iX1 Specifies the X-coordinate of the start point in the  
 following range:

INT iY1 Specifies the Y-coordinate of the start point in the  
 following range:

INT iX2 Specifies the X-coordinate of the end point in the  
 following range:

INT iY2 Specifies the Y-coordinate of the end point in the  
 following range:

X-coordinate: 0000h ~ (0 ~)

Y-coordinate: 0000h ~ (0 ~)

## 5-3-16 Drawing a Circle Using an Extended Function

To draw a circle by specifying the extended function, use the following command:  
The tilling pattern is added as the extended function.

### ■ Syntax

```
int WINAPI MtoMESC_c
(LPMtoMSOCK pMSock, MTOMCALLBACK pfFinish, GPCOLOR cDisplayColor,
GPCOLOR cBackColor, GPTILE cTile, CINT iX1, INT iY1, INT iRadius)
```

### ■ Return Value

When pfFinish is NULL

00 : Normal termination

Other : Error code

When pfFinish is any value other than NULL

MTOM\_CONTINUE : The system is informed of the processing request.  
Completion of the processing is communicated when  
pfFinish is called back.

Other : Error code

### ■ Argument

LPMtoMSOCK pMSock

Handle of memory link socket

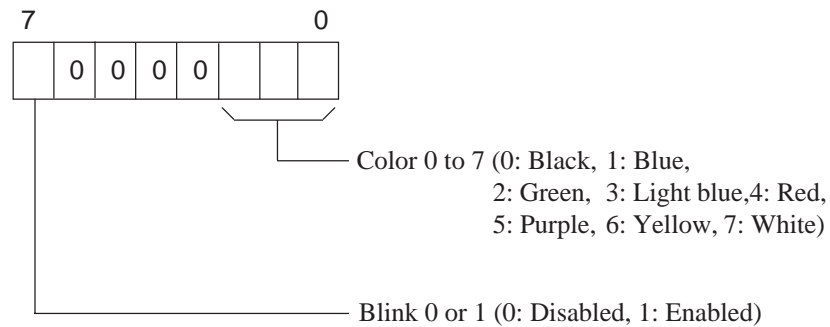
MTOMCALLBACK pfFinishNULL: This API system will not be completed until  
the specified processing is completed.  
(Synchronous Transmission)

After the specified processing is completed,  
the API system will return with the processing  
result code.

Other than NULL: Pointer to the function will be called back after  
completion of processing. When this  
parameter is specified, this API system will  
return with MTOM\_CONTINUE immediately  
after the processing is requested.  
(Asynchronous Transmission)

After the system completes the processing, the  
specified callback function is called with the  
corresponding socket handle and processing  
result code.

GPCOLOR cDisplayColor Specifies the display color in the following range:  
 GPCOLOR cBackColor Specifies the background color in the following range:



GPTILE cTile Specifies the tiling pattern in the range of 00h to 08h.  
**▼Reference▲** *5-3-6 Tiling Patterns in "5-3-6 Drawing a Filled Rectangle"*

INT iX1 Specifies the X-coordinate of the center in the following range:

INT iY1 Specifies the Y-coordinate of the center in the following range:  
 X-coordinate: 0000h ~ (0 ~)  
 Y-coordinate: 0000h ~ (0 ~)

INT iRadius Specify the radius in the range of 0001h ~ (1~).

## 5-3-17 Drawing a Sector Using an Extended Function

To draw a sector by specifying the extended function, use the following command:  
The line type is added as the extended function.

### ■ Syntax

```
int WINAPI MtoMESC_g
(LPMtoMSOCK pMSock, MTOMCALLBACK pfFinish, GPCOLOR
cDisplayColor, GPCOLOR cBackColor,
GPLINESTYLE cLineStyle, INT iX1, INT iY1, INT iRadius, INT iStartAngle, INT
iEndAngle)
```

### ■ Return Value

When pfFinish is NULL

00 : Normal termination  
Other : Error code

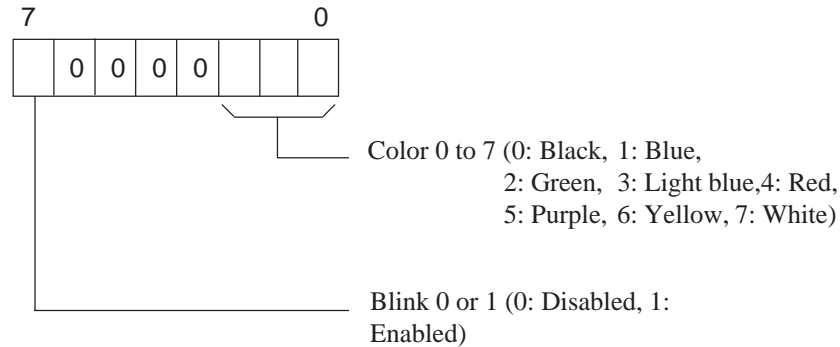
When pfFinish is any value other than NULL

MTOM\_CONTINUE : The system is informed of the processing request.  
Completion of the processing is communicated when  
pfFinish is called back.  
Other : Error code

### ■ Argument

LPMtoMSOCK pMSock Handle of memory link socket  
MTOMCALLBACK pfFinishNULL: This API system will not be completed until  
the specified processing is completed.  
(Synchronous Transmission)  
After the specified processing is completed,  
the API system will return with the processing  
result code.  
Other than NULL: Pointer to the function will be called back after  
completion of processing. When this  
parameter is specified, this API system will  
return with MTOM\_CONTINUE immediately  
after the processing is requested.  
(Asynchronous Transmission)  
After the system completes the processing, the  
specified callback function is called with the  
corresponding socket handle and processing  
result code.

GPCOLOR cDisplayColor Specifies the display color in the following range:  
 GPCOLOR cBackColor Specifies the background color in the following range:



GPLINESTYLE cLineStyle Specifies the line type in the following range:  
 (00: ——— 01: - - - - 02: — - — 03: — - - -  
 04: ——— 05: - - - - 06: — · — 07: — - - - )



- Note:** • Line types 0 to 3 are 1-dot lines, line type 8 is 3-dot line and line type 9 is 5-dot line.

INT iX1 Specifies the X-coordinate of the center in the following range:  
 INT iY1 Specifies the Y-coordinate of the center in the following range:  
 X-coordinate: 0000h ~ (0 ~)  
 Y-coordinate: 0000h ~ (0 ~)  
 INT iRadius Specify the radius in the range of 0001h ~ (1~).  
 INT iStartAngle Specifies the start angle in the following range:  
 INT iEndAngle Specifies the end angle in the following range:  
 Angle: 0000h to 0168h (0 to 360)



- *Do not enter the same value for both the start and end angles.*
- *The drawing direction is counter-clockwise.*

## 5-3-18 Brightness/Contrast Setup

To set up the brightness/contrast, use the following commands:

### ■ Syntax

DWORD WINAPI SetContrase  
(LPMtoMSOCK pMSock, MTOMCALLBACK pfFinish, DWORD dwContrast,  
DWORD dwLight)

### ■ Return Value

Setting of the brightness/contrast to be changed

### ■ Argument

LPMtoMSOCK pMSock	Handle of memory link socket
MTOMCALLBACK pfFinish	Other than NULL: This API system will not be completed until the specified processing is completed. (Synchronous Transmission) After the specified processing is completed, the API system will return with the processing result code.
MTOMCALLBACK pfFinish	Other than NULL: Pointer to the function will be called back after completion of processing. When this parameter is specified, this API system will return with MTOM_CONTINUE immediately after the processing is requested. (Asynchronous Transmission) After the system completes the processing, the specified callback function is called with the corresponding socket handle and processing result code.
dwContrast	Contrast adjustment (0000h to 0007h) 0: Bright ~ 7: Dark "FFFFFFFFh" indicates "No setting" (this model disables construct adjustment).
dwLight	Brightness adjustment (0000h to 0001h) 0: Bright, 1:: Dark "FFFFFFFFh" indicates "No setting" (this model disables brightness adjustment).

## 5-3-19 Acquiring Brightness/Contrast

To acquire the current setting of brightness/contrast, use the following command:

### ■ Syntax

DWORD WINAPI GetContrase  
(LPMtoMSOCK pMSock, MTOMCALLBACK pfFinish, DWORD\*dwContrast,  
DWORD \*dwLight)

### ■ Return Value

Current brightness/contrast setting

### ■ Argument

LPMtoMSOCK pMSock	Handle of memory link socket
MTOMCALLBACK pfFinishNULL:	This API system will not be completed until the specified processing is completed. (Synchronous Transmission) After the specified processing is completed, the API system will return with the processing result code.
Other than NULL:	Pointer to the function will be called back after completion of processing. When this parameter is specified, this API system will return with MTOM_CONTINUE immediately after the processing is requested. (Asynchronous Transmission) After the system completes the processing, the specified callback function is called with the corresponding socket handle and processing result code.
dwContrast	Current value of contrast (0000h to 0007h) 0: Bright ~ 7: Dark "FFFFFFFFh" indicates "No setting" (contrast cannot be adjusted with this model).
dwLight	Current value of brightness (0000h to 0001h) 0: Bright, 1:: Dark "FFFFFFFFh" indicates "No setting" (brightness cannot be adjusted with this model).



	Code	Description
MTOM_OK	00	Processing has been normally completed.
MTOM_CONTINUE	01	Processing is in progress.
MTOM_USERS_STOPED	03	The processing has been canceled by a user application. (The MtoMStop() function was called, but the callback function returned FALSE.)
MTOM_EVENT_TOUCH	40	The touch panel has been pressed. (It is not the API system's return value, but the callback function is informed of this code when an event registered by the SetMtoMEventCallBack() function occurs.)
MTOM_EVENT_CLOSED	41	Connection has been closed.
MTOM_ERROR	80	Error response from the GP unit.
MTOM_ERROR_INVALID	81	An API parameter error occurred, or the API was illegally called.
MTOM_ERROR_LAN	82	An error occurred on the line. (Winstock returned an error code.)
MTOM_ERROR_TOUT_RES	83	Response timeout error.
MTOM_ERROR_TOUT_CHAR	84	Character-to-character transmission timeout error. (Transmission of data frames from GP was interrupted.)
MTOM_ERROR_NAK	85	GP returned NAK.



- *When the system has received an error response from the GP unit, call the MtoMGetLastError() function to acquire details of the error.*

# Chapter 6: Sample Program

This chapter describes the sample program (GpM.EXE) that uses the Memory Link LAN API included with the GP Ethernet I/F Unit. The GpM.EXE program is the drawing software that enables the objects created with Windows to be displayed on the GP panel on a real time basis through 1:1 or 1:n (multi-link) connection between the GP unit(s) and Windows.

## 6-1

## Memory Link LAN API Sample Program

### ■ Start-up Environments

- ① The GpM.EXE program runs on the Windows® 95 operating system.
- ② Since the GpM.EXE program uses the MtomLAN.DLL file, copy the MtomLAN.DLL file into a Windows directory.
- ③ The Memory Link LAN uses TCP/IP protocol; so, you must first install the TCP/IP protocol. (Install Microsoft® TCP/IP by selecting [Start] - [Control Panel] - [Network].)



**Note:**

- If "DLL: LAN initialize error" appears at the start-up of the GpM.EXE program and the program cannot be started, the TCP/IP settings may be incorrect. Check the TCP/IP settings.

### ■ Development Environment

The GpM.EXE program has been developed for use in the following environments:  
The sample program's source code is contained on the GP-PRO/PBIII for Windows CD-ROM.

When the source code found in the CD-ROM's [MTOMLAN] folder is compiled in the following environment, the file [GpM.EXE] is created.

Compiler : Microsoft® Visual C++™ Ver 4.1  
OS : Microsoft® Windows® 95

### ■ Basic Structure

The GpM.EXE program uses MDI (Multi Document Interface). In the 1:1 communication mode, a single GP unit is controlled by "One Document - One View".

On the other hand, in the 1:n communication mode, all the GP units connected to the network are controlled by "One Document - One View" through the broadcast function.

To understand this program, you must learn how its documents and views are managed and related by the Microsoft® MFC MDI framework.

### ■ How to Access Memory Link API

To access the Memory Link API, the GpM.EXE program defines and uses Class CMSock. Class CMSock fully includes the Memory Link API as "One Socket - One Object".

The GpM.EXE program provides a callback from the Memory Link API by overriding the method of Class CMSock.

### ■ Derivation of CMSock

The GpM.EXE program uses two classes derived from CMSock: One is inherited by Class CgpMApp for 1:n communication and node search, and another is inherited by Class CGpMDoc for 1:1 communication.

CcpMApp is the application class of GpM, and CGpMDoc is the document class of GpM. In other words, the application class manages 1:n communication, and the document class manages 1:1 communication.

### ■ Drawing Method

The GpM.EXE program provides two objects for one drawing method (e.g. method of displaying a character string, or displaying a line.)

#### ◆ Drawing pen object

This object allows users to specify an object. (It is noted as "Display Designation" in the source program.) This object is divided into the following two parts:

One is the property dialog that allows users to specify the properties of objects. The other allows users to specify the object position using the mouse. It corresponds to the class specified as CgpPEsc\*, which is derived from CGpPenEsc.

#### ◆ Contents object

This object is used to transmit objects (data) that the user has specified with the pen object to the GP unit using CMSock, to display the objects in the window, and to save/read (serialize) the data.

It corresponds to the class specified as CContentsEsc\*, which is derived from CContents.

**■ Class CGpMDOC**

This class is the core of the CpM.Exe program. It is used for document data management, including an array of the contents objects. Also, this class manages the connection with a GP unit in the 1:1 communication mode.

**■ Class CGpMView**

This class displays the related contents object of the CGpMDoc class in the window.

**■ Class CGpList**

This class searches for a GP unit participating in the network. CGpMApp is used as the socket.

**■ GpWin. Cpp**

This is not a class, but the library for simulating the created GP panel screen on the Windows® program.

With this library, however, GP panel screens cannot completely be simulated. Only the Windows® GPI's simulation range can be supported.



**Careful!** • *Since Windows® 95 does not support GDI, bold dotted lines cannot be displayed*

**■ MtoMAPI.H and MtoMLAN.LI**

The CpM.Exe program includes the MtoMAPI.H file in the external device.

The MtoMAPI.H file is stored in the MtomLAN directory of the floppy disk included in the GP Ethernet I/F Unit. Copy this file into an appropriate directory, and specify the location by changing the #include statement of defsfile.h.

The CpM.Exe program includes the MtoMLAN.LIB file to call the MtoMLAN.DLL program. Copy this file into an appropriate directory, and specify the location by selecting **Setup** - **Linker** - **Object/Library Module**.

# *Memo*

# Chapter 7: Error Messages

This chapter describes the error messages that may be displayed during memory link communication. Check the contents of the error message, and take the necessary action.

## 7-1 Error Message List

If the GP's communication parameter settings do not match those of the host controller, or the data sent from the host controller is invalid, a communication error will occur, and the following error message will be displayed.

The "Host Communication Error" message is followed by the following message:

Host Communication Error (02:\*\*)

\*\* : Error code

Error code	Description	Action
10	An undefined command has been received.	The data sent from the host controller may be invalid.
15	The specified display attribute does not conform to the format.	Check that the transmission data are valid, and re-transmit the correct data.
16	The specified character size does not conform to the format.	An SIO error may have occurred. Check the communication parameter settings and connecting environments (e.g. noise interference).
17	The specified coordinate does not conform to the format.	
18	The specified line type does not conform to the format.	
19	The specified tiling pattern does not conform to the format.	
1A	The specified radius exceeds the display range.	
1B	The specified start angle/end angle does not conform to the format.	
1C	The specified character type code does not conform to the format.	
1D	The specified "Turn" code does not conform to the format.	
1E	The specified "Direction" code does not conform to the format.	
1F	The specified "Highlighting" code does not conform to the format.	

Error code	Description	Action
20	The specified arrow pattern does not conform to the format.	The data sent from the host controller may be invalid. Check that the transmission data are valid, and re-transmit the correct data. An SIO error may have occurred. Check the communication parameter settings and connecting environments (e.g. noise interference).
21	The specified "Arrow Direction" code does not conform to the format.	
22	The specified chamfering type does not conform to the format.	
23	The specified "Centering" code does not conform to the format.	
FA	The specified System Area address exceeds the setting range.	
FB	Written/read data exceeds the capacity of the System Area.	
FC	The GP unit has received an invalid data format.	
FF	The GP unit cannot send data for ten or more seconds.	The communication control may not have been properly performed. Check the communication cable.

**■ Protocol Stack Error Codes**

The following error code will be displayed when a protocol stack error occurs.

If the following error occurs, check the GP's Ethernet Information and host controller's settings.

Host Communication Error (02:FE:\*\*)  
 \*\*: Error code

Error code	Description
00	SRC IP address error during initialization
05	Failed to initialize.
06	Failed to cancel communication.
07	Attempted to open connection before initialization is normally completed.
08	SRC port number error
09	Destination port number error
0A	Destination IP address error
0B	The same port number has been used to open the UDP connection.
0C	The same port number has been used to open the TCP connection with the same destination node.
0D	The protocol stack rejects "Open Connection" request.
0E	The protocol stack returns "Failed to Open".
0F	The connection has been closed.
10	All connections are being used. (There is no unused connection.)