Memory Link Driver

Introduction

This manual describes how to connect the Display and the External Device (target PLC).

In this manual, the connection procedure will be described by following the below sections:

1	System Configuration This section shows the types of External Devices which can be connected and SIO type.	☞ "1 System Configuration" (page 3)
2	Selection of External Device Select a model (series) of the External Device to be connected and connection method.	☞ "2 Selection of Driver" (page 6)
3	Example of Communication Settings This section shows setting examples for communicating between the Display and the External Device.	☞ "3 Example of Communication Setting" (page 7)
4	Setup Items This section describes communication setup items on the Display. Set communication settings of the Display with GP-Pro EX or in off-line mode.	☞ "4 Setup Items" (page 15)
5	Cable Diagram This section shows cables and adapters for connecting the Display and the External Device.	☞ "5 Cable Diagram" (page 22)

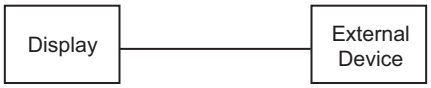
Operation

1 System Configuration

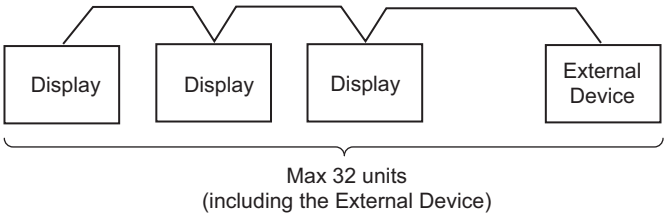
When using the Memory Link, the connection configuration is shown below.

■ Serial

- 1:1 Connection



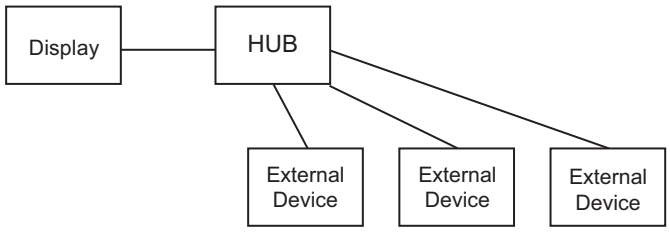
1:n Connection



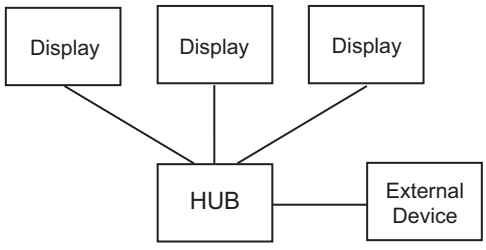
■ Ethernet

- 1:1 Connection

- 1:n Connection



- n:1 Connection (only available when selecting "Ethernet (UDP)")



■ COM Port of IPC

When connecting IPC with External Device, the COM port which can be used changes with series and SIO type. Please refer to the manual of IPC for details.

Usable port

Series	Usable port		
	RS-232C	RS-422/485(4 wire)	RS-422/485(2 wire)
PS-2000B	COM1[*1] , COM2, COM3[*1], COM4	-	-
PS-3650A, PS-3651A	COM1[*1]	-	-
PS-3700A (Pentium®4-M) PS-3710A	COM1[*1], COM2[*1], COM3[*2] , COM4	COM3[*2]	COM3[*2]
PS-3711A	COM1[*1], COM2[*2]	COM2[*2]	COM2[*2]

*1 The RI/5V can be switched. Please switch with the change switch of IPC.

*2 It is necessary to set up the SIO type with the Dip switch. Please set up as follows according to SIO type to be used.

Dip switch setting: RS-232C

Dip switch	Setting	Description
1	OFF	Reserve (always OFF)
2	OFF	SIO type: RS-232C
3	OFF	
4	OFF	Output mode of SD (TXD) data: Always output
5	OFF	Terminal resistance (220Ω) insertion to SD (TXD): None
6	OFF	Terminal resistance (220Ω) insertion to RD (RXD): None
7	OFF	Short-circuit of SDA (TXA) and RDA (RXA): Does not Exist
8	OFF	Short-circuit of SDB (TXB) and RDB (RXB): Does not Exist
9	OFF	RS (RTS) Auto control mode: Disable
10	OFF	

Dip switch setting: RS-422/485 (4 wire)

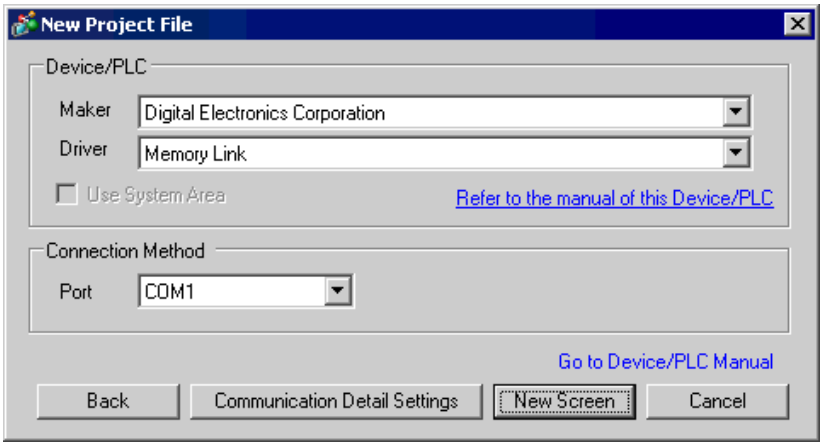
Dip switch	Setting	Description
1	OFF	Reserve (always OFF)
2	ON	SIO type: RS-422/485
3	ON	
4	OFF	Output mode of SD (TXD) data: Always output
5	OFF	Terminal resistance (220Ω) insertion to SD (TXD): None
6	OFF	Terminal resistance (220Ω) insertion to RD (RXD): None
7	OFF	Short-circuit of SDA (TXA) and RDA (RXA): Does not Exist
8	OFF	Short-circuit of SDB (TXB) and RDB (RXB): Does not Exist
9	OFF	RS (RTS) Auto control mode: Disable
10	OFF	

Dip switch setting: RS-422/485 (2 wire)

Dip switch	Setting	Description
1	OFF	Reserve (always OFF)
2	ON	SIO type: RS-422/485
3	ON	
4	OFF	Output mode of SD (TXD) data: Always output
5	OFF	Terminal resistance (220Ω) insertion to SD (TXD): None
6	OFF	Terminal resistance (220Ω) insertion to RD (RXD): None
7	ON	Short-circuit of SDA (TXA) and RDA (RXA): Exist
8	ON	Short-circuit of SDB (TXB) and RDB (RXB): Exist
9	ON	RS (RTS) Auto control mode: Enable
10	ON	

2 Selection of Driver

Select the External Device to be connected to the Display.



Setup Items	Setup Description
Maker	Select the maker of the External Device to be connected. Select "Digital Electronics Corporation".
Driver	Select a model (series) of the External Device to be connected and connection method. Select "Memory Link". Check the connection configuration in "Memory Link" in system configuration. ☞ "1 System Configuration" (page 3)
Port	Select the Display port to be connected to the External Device.

3 Example of Communication Setting

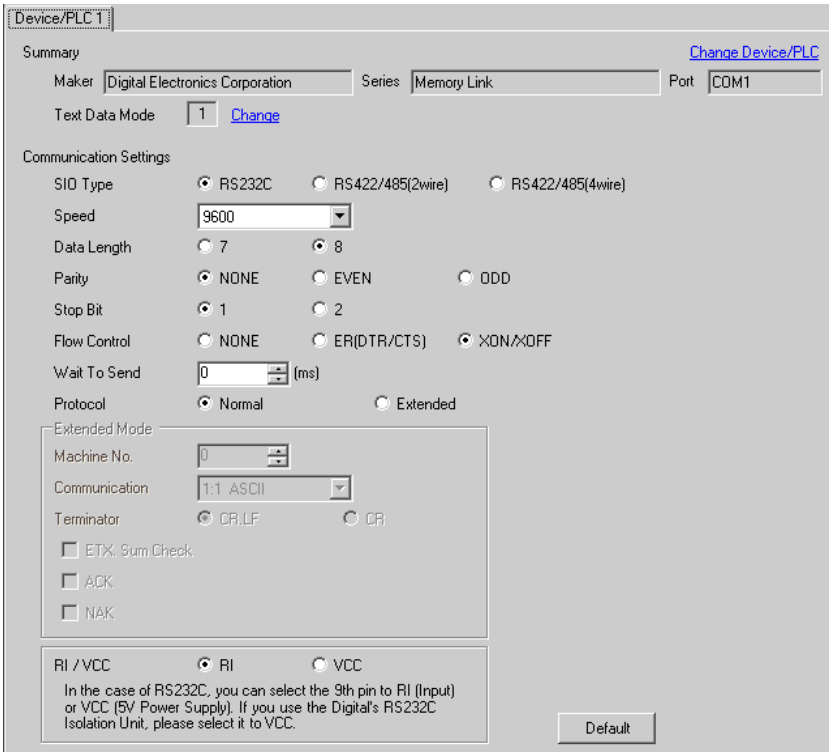
Examples of communication settings of the Display and the External Device, recommended by Pro-face, are shown.

3.1 Setting Example 1

■ Settings of GP-Pro EX (RS232C connection: Convert mode)

◆ Communication Settings

To display the setting screen, select [Device/PLC Settings] from [System setting window] in workspace.



■ Settings of External Device

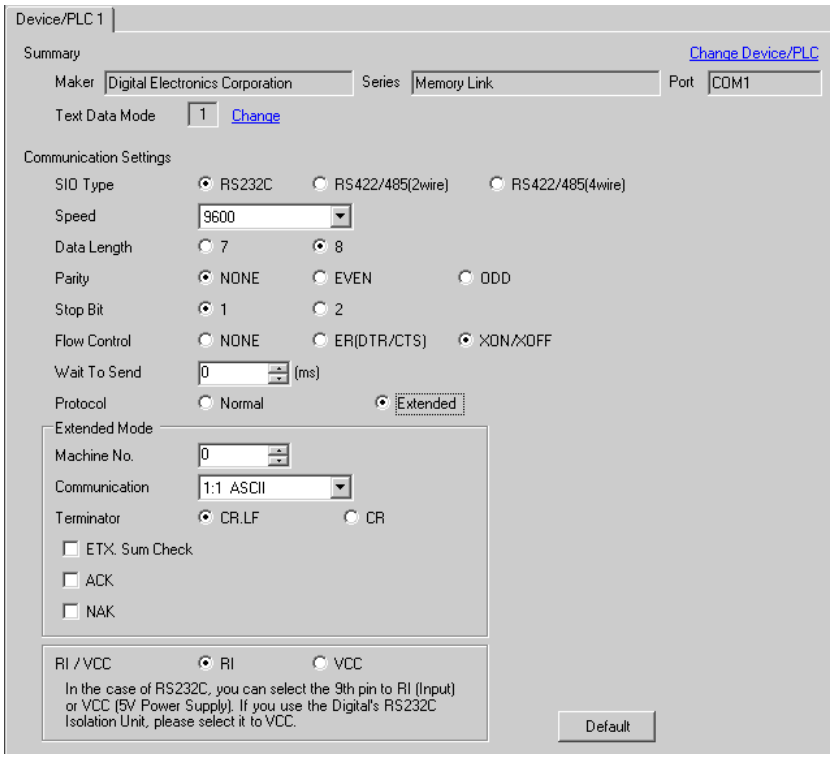
Depends on the specification of the External Device.

3.2 Setting Example 2

■ Settings of GP-Pro EX (RS232C connection: Extend mode)

◆ Communication Settings

To display the setting screen, select [Device/PLC Settings] from [System setting window] in workspace.



■ Settings of External Device

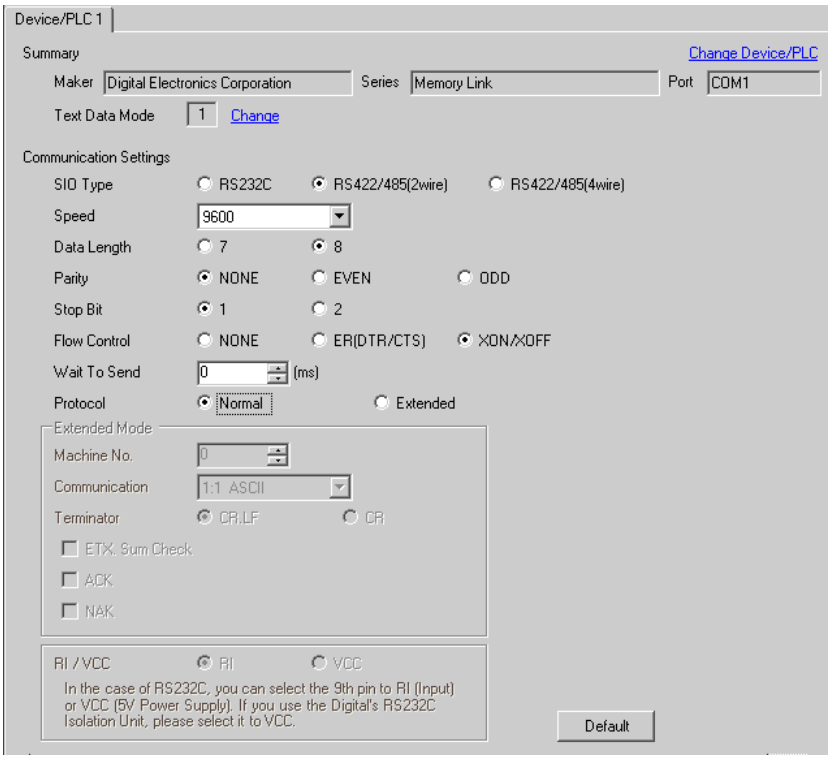
Depends on the specification of the External Device.

3.3 Setting Example 3

■ Settings of GP-Pro EX (RS422/485 (2wire) connection: Convert mode)

◆ Communication Settings

To display the setting screen, select [Device/PLC Settings] from [System setting window] in workspace.



■ Settings of External Device

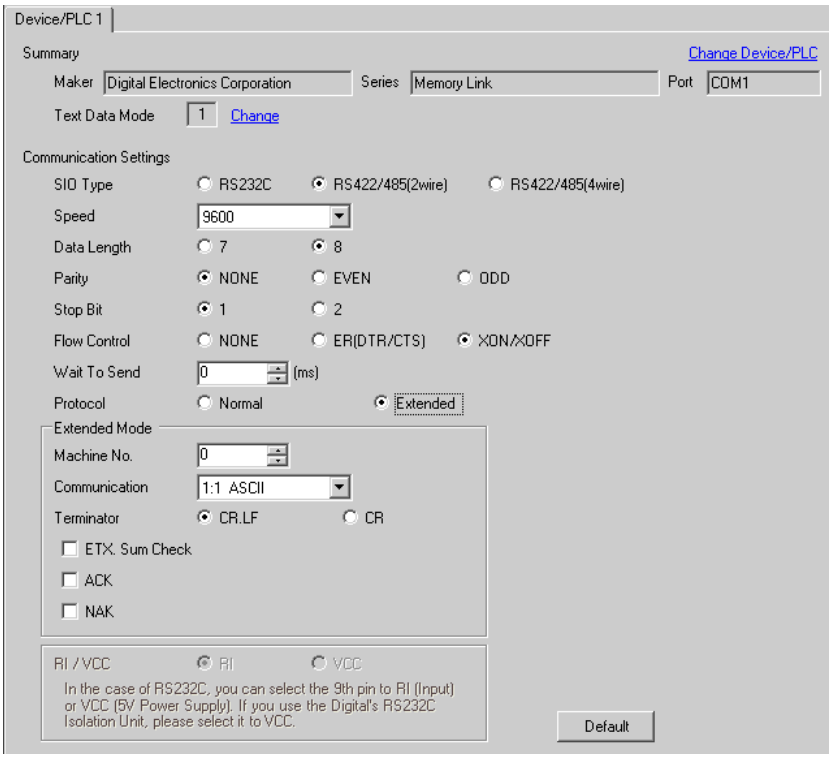
Depends on the specification of the External Device.

3.4 Setting Example 4

■ Settings of GP-Pro EX (RS422/485 (2wire) connection: Extend mode)

◆ Communication Settings

To display the setting screen, select [Device/PLC Settings] from [System setting window] in workspace.



■ Settings of External Device

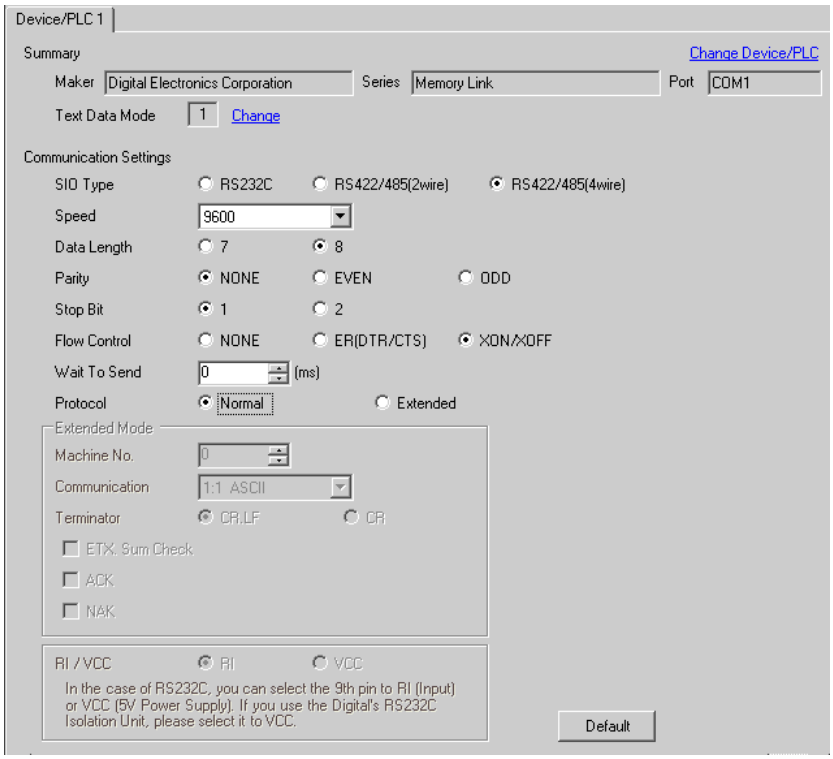
Depends on the specification of the External Device.

3.5 Setting Example 5

■ Settings of GP-Pro EX (RS422/485 (4wire) connection: Convert mode)

◆ Communication Settings

To display the setting screen, select [Device/PLC Settings] from [System setting window] in workspace.



■ Settings of External Device

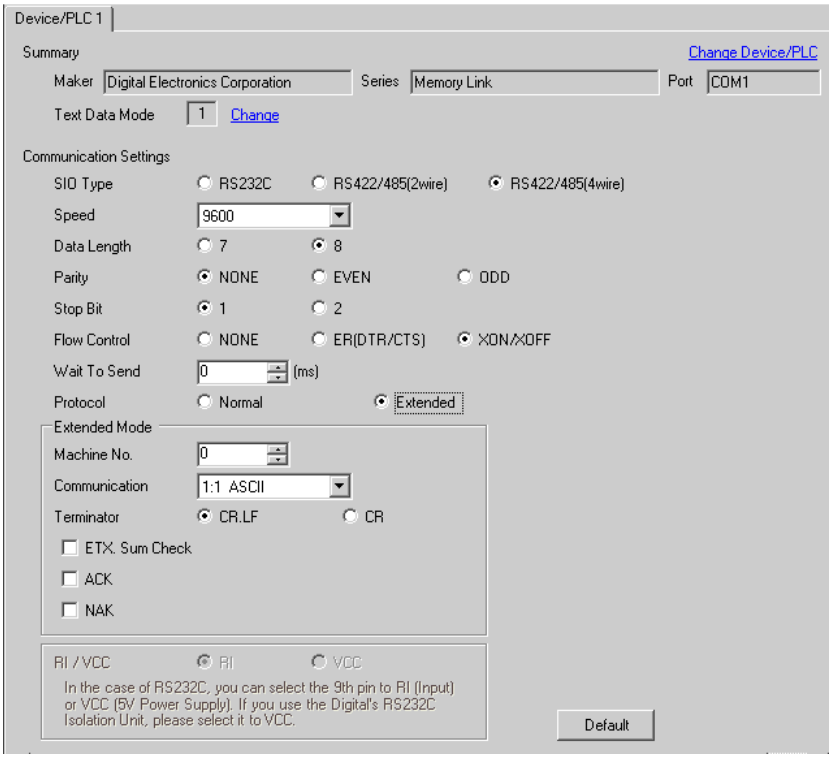
Depends on the specification of the External Device.

3.6 Setting Example 6

■ Settings of GP-Pro EX (RS422/485 (4wire) connection: Extend mode)

◆ Communication Settings

To display the setting screen, select [Device/PLC Settings] from [System setting window] in workspace.



■ Settings of External Device

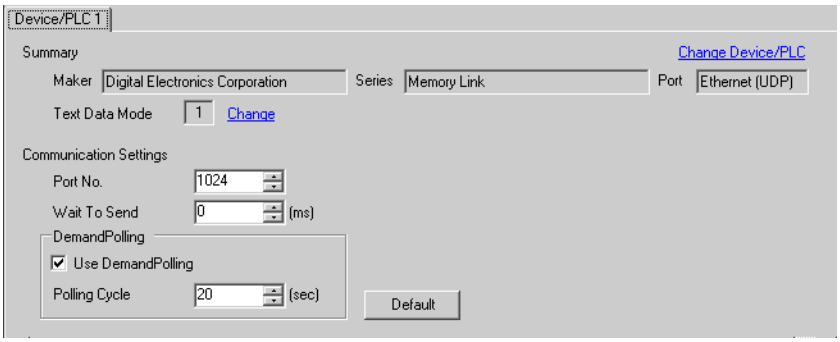
Depends on the specification of the External Device.

3.7 Setting Example 7

■ Settings of GP-Pro EX (Ethernet (UDP) connection)

◆ Communication Settings

To display the setting screen, select [Device/PLC Settings] from [System setting window] in workspace.



■ Settings of External Device

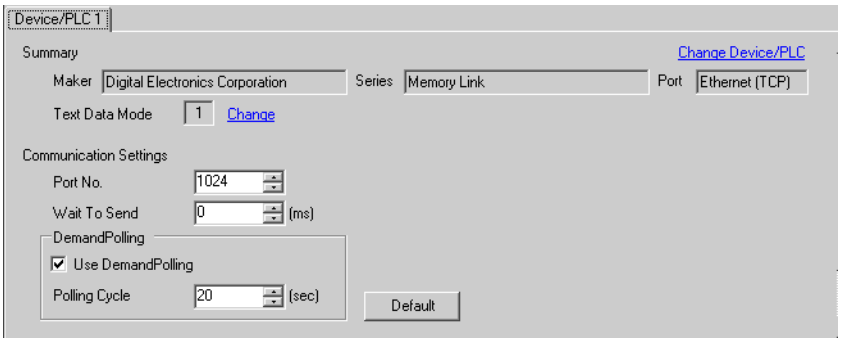
Depends on the specification of the External Device.

3.8 Setting Example 8

■ Settings of GP-Pro EX (Ethernet (TCP) connection)

◆ Communication Settings

To display the setting screen, select [Device/PLC Settings] from [System setting window] in workspace.



■ Settings of External Device

Depends on the specification of the External Device.

4 Setup Items

Set communication settings of the Display with GP-Pro EX or in off-line mode of the Display.

The setting of each parameter must be identical to that of External Device.

☞"3 Example of Communication Setting" (page 7)

IMPORTANT

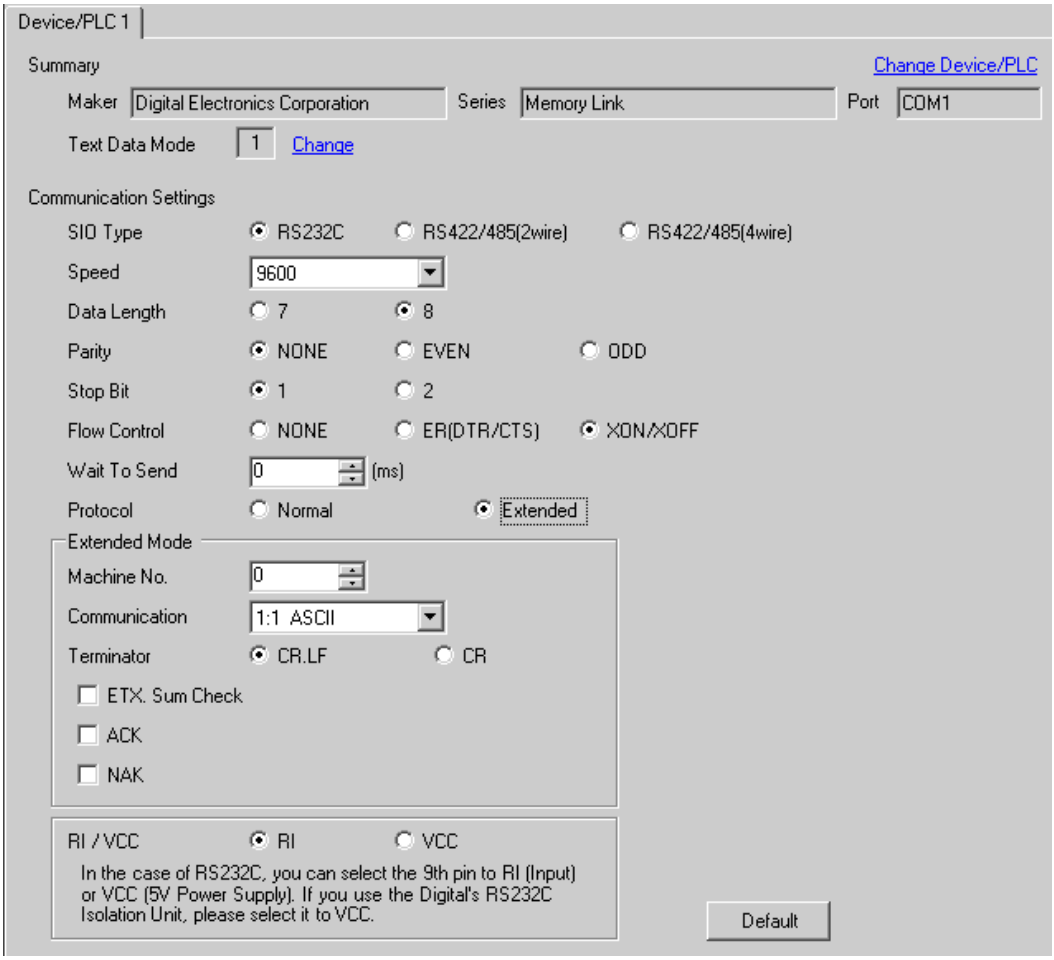
- You need to set IP address on the display in the off-line mode of the display.
  Cf. Maintenance / Troubleshooting “2.5 Ethernet Settings“

4.1 Serial Connection

■ Setup Items in GP-Pro EX

◆ Communication Settings

To display the setting screen, select [Device/PLC Settings] from [System setting window] in workspace.



Setup Items	Setup Description
SIO Type	Select the SIO type to communicate with the External Device.

continued to next page

Setup Items	Setup Description
Speed	Select speed between the External Device and the Display.
Data Length	Select data length.
Parity	Select how to check parity.
Stop Bit	Select stop bit length.
Flow Control	Select the communication control method to prevent overflow of transmission and reception data.
Wait To Send	Use an integer from 0 to 255 to enter standby time (ms) for the Display from receiving packets to transmitting next commands.
Protocol	Select the communication protocol.
Machine No.	Use an integer 0 to 31 to enter the unit number of the External Device to communicate.
Communication	Select any of "1:1 ASCII code", "1:1 Binary code", "1:N ASCII code" and "1:N Binary code" for SIO type.
Terminator	Select the terminator to be used. The terminator is available only in case that the communication type is [1:1 ASCII] or [1:n ASCII].
ETX. Sum Check	Set whether the sum check code is added or not for data communication.
ACK	Check this option when you use ACK.
NAK	Check this option when you use NAK.
RI/VCC	You can switch RI/VCC of the 9th pin when you select RS232C for SIO type. It is necessary to change RI/5V by changeover switch of IPC when connect with IPC. Please refer to the manual of the IPC for more detail.

■ Setup Items in Off-Line Mode

NOTE

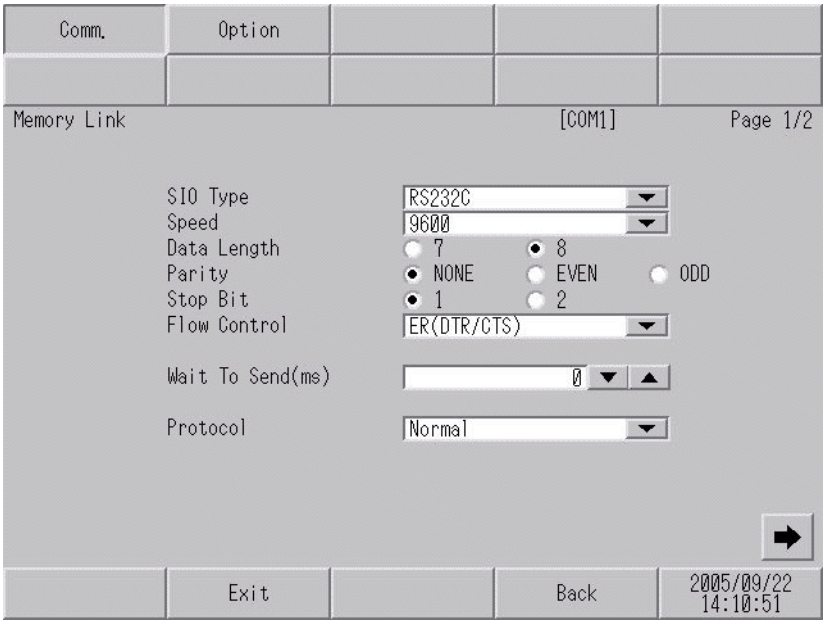
- Please refer to Maintenance/Troubleshooting for more information on how to enter off-line mode or about operation.

  Cf. Maintenance/Troubleshooting "2.2 Offline Mode"

◆ Communication Settings

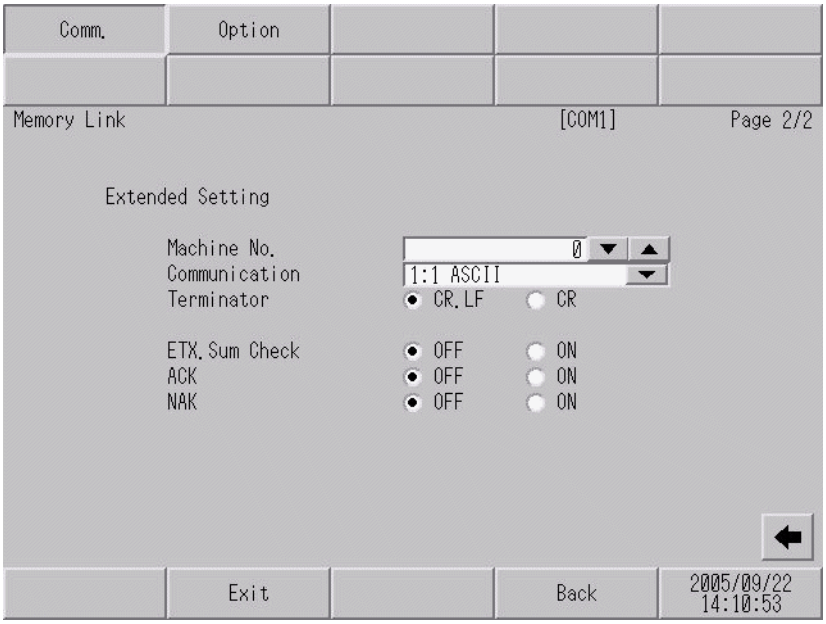
To display the setting screen, touch [Device/PLC Settings] from the [Peripheral Settings] in the off-line mode. Touch the External Device you want to set from the displayed list, and touch the [Communication Settings].

(Page 1/2)



Setup Items	Setup Description
SIO Type	Select the SIO type to communicate with the External Device. **IMPORTANT** To make the communication settings correctly, confirm the serial interface specifications of Display unit for [SIO Type]. We cannot guarantee the operation if a communication type that the serial interface does not support is specified. For details concerning the serial interface specifications, refer to the manual for Display unit.
Speed	Select speed (bps) between the External Device and the Display.
Data Length	Select data length.
Parity	Select how to check parity.
Stop Bit	Select stop bit length.
Flow Control	Select the communication control method to prevent overflow of transmission and reception data.
Wait To Send	Use an integer from 0 to 255 to enter standby time (ms) for the Display from receiving packets to transmitting next commands.
Protocol	Select either "Normal" or "Extended" for the communication protocol.

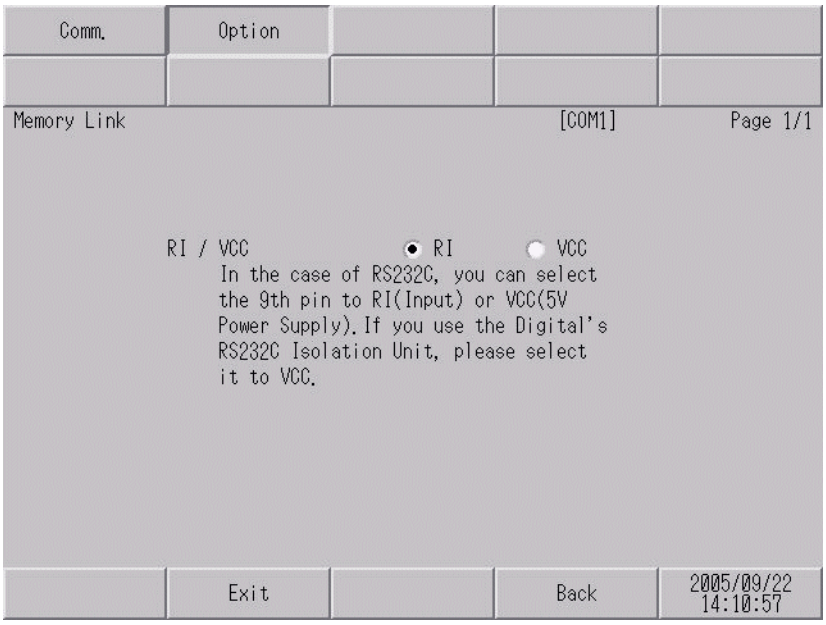
(Page 2/2)



Setup Items	Setup Description
Machine No.	Use an integer 0 to 31 to enter the unit number of the External Device to communicate.
Communication	Select any of "1:1 ASCII code", "1:1 Binary code", "1:N ASCII code" and "1:N Binary code" for SIO type.
Terminator	Select either of "CR, LF" or "CR" for the terminator to be used. The terminator is available only in case that the communication type is [1:1 ASCII] or [1:n ASCII].
ETX. Sum Check	Set whether the sum check code is added or not for data communication.
ACK	Check this option when you use ACK.
NAK	Check this option when you use NAK.

◆ Option

To display the setting screen, touch [Device/PLC Settings] from [Peripheral Settings] in the off-line mode. Touch the External Device you want to set from the displayed list, and touch [Option].



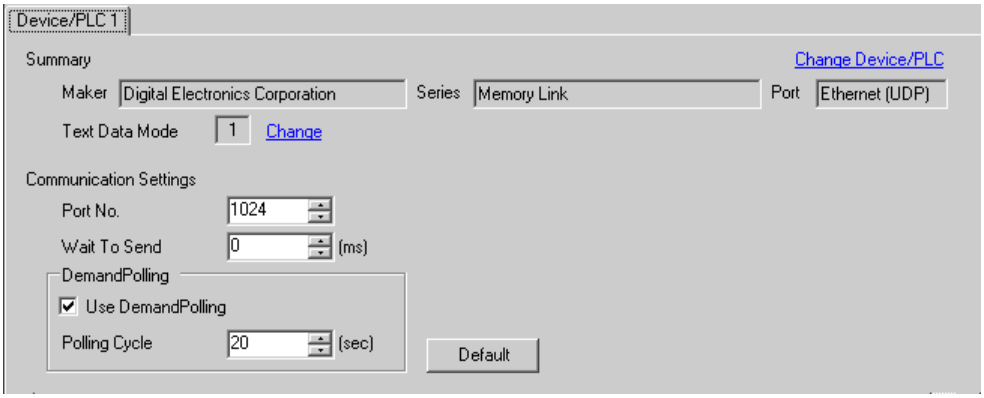
Setup Items	Setup Description
RI/VCC	You can switch RI/VCC of the 9th pin when you select RS232C for SIO type. It is necessary to change RI/5V by changeover switch of IPC when connect with IPC. Please refer to the manual of the IPC for more detail.

4.2 Ethernet Connection

■ Steup Items in GP-Pro EX

◆ Communication Settings

To display the setting screen, select [Device/PLC Settings] from [System setting window] in workspace.



Setup Items	Setup Description
Port No.	Use an integer 1024 to 65535 to enter the port number of the External Device to communicate.
Wait To Send	Use an integer from 0 to 255 to enter standby time (ms) for the Display from receiving packets to transmitting next commands.
Use Demand Polling	Check this option when you use the demand polling command to confirm the presence of the External Device.
Polling Cycle	Use an integer from 10 to 100 to enter the polling cycle (sec).

■ Setup Items in Off-Line Mode

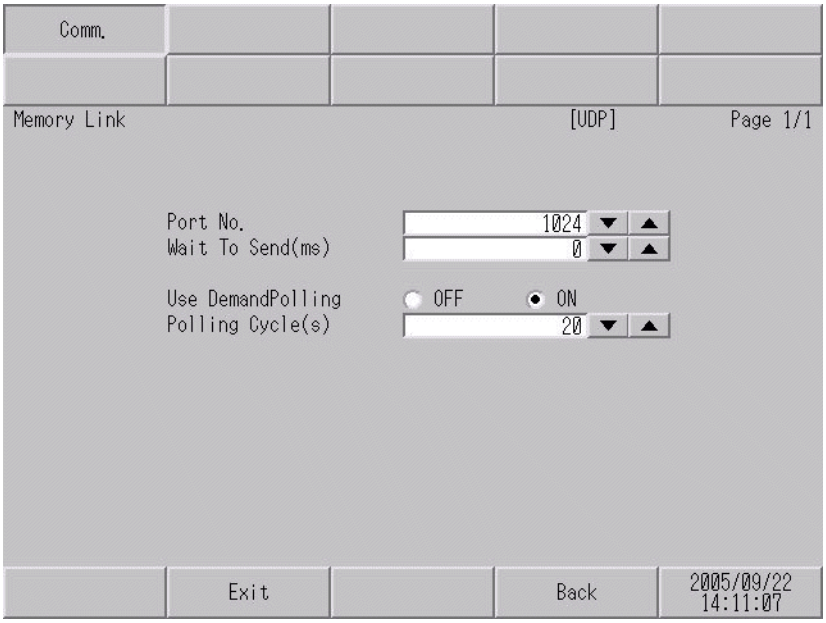
NOTE

- Please refer to Maintenance/Troubleshooting for more information on how to enter off-line mode or about operation.

  Cf. Maintenance/Troubleshooting "2.2 Offline Mode"

◆ Communication Settings

To display the setting screen, touch [Device/PLC Settings] from the [Peripheral Settings] in the off-line mode. Touch the External Device you want to set from the displayed list, and touch the [Communication Settings].



Setup Items	Setup Description
Port No.	Use an integer 1024 to 65535 to enter the port number of the External Device to communicate.
Wait To Send	Use an integer from 0 to 255 to enter standby time (ms) for the Display from receiving packets to transmitting next commands.
Use Demand Polling	Check this option when you use the demand polling command to confirm the presence of the External Device.
Polling Cycle (s)	Use an integer from 10 to 100 to enter the polling cycle (sec).

5 Cable Diagram

The cable diagram shown below may be different from the cable diagram recommended by the maker of the External Device. Please be assured there is no operational problem in applying the cable diagram shown in this manual.

- The FG pin of the main body of the external device must be D-class grounded. Please refer to the manual of the external device for more details.
- SG and FG are connected inside the Display. When connecting SG to the External Device, design the system not to form short-circuit loop.
- Connect the isolation unit, when communication is not stabilized under the influence of a noise etc..

Cable Diagram 1

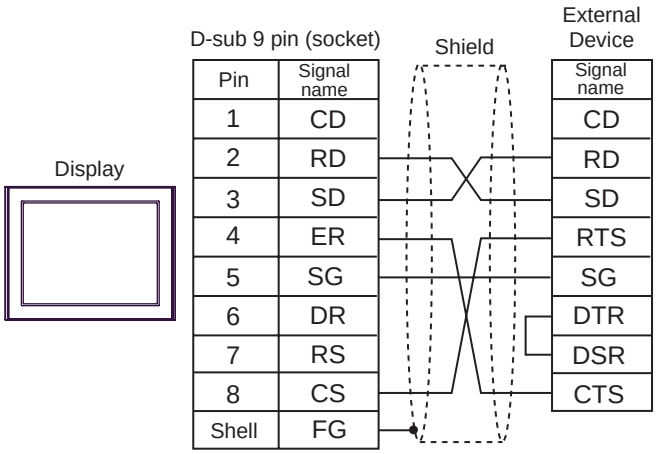
Display (Connection Port)	Cable		Remarks
GP (COM1) IPC*1	A	Your Own Cable (ER Control)	The cable length must be 15m or less.
	B	Your Own Cable (X Control)	
	C	Your Own Cable (without control method)	

*1 Only the COM port which can communicate by RS-232C can be used.
☞"■ COM Port of IPC" (page 4)

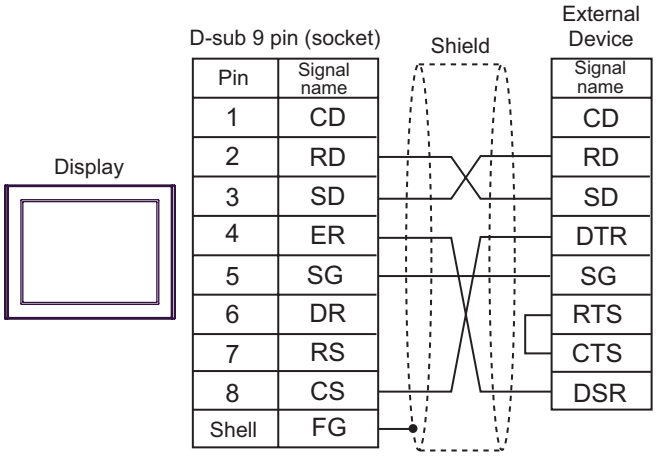
IMPORTANT
- Correspondence of the RS232C connector type or pin number with the signal name varies depending on the host device. Connect properly according to the interface specification of the host device.

A) When using your own cable

- When the External Device supports RTS/CTS control



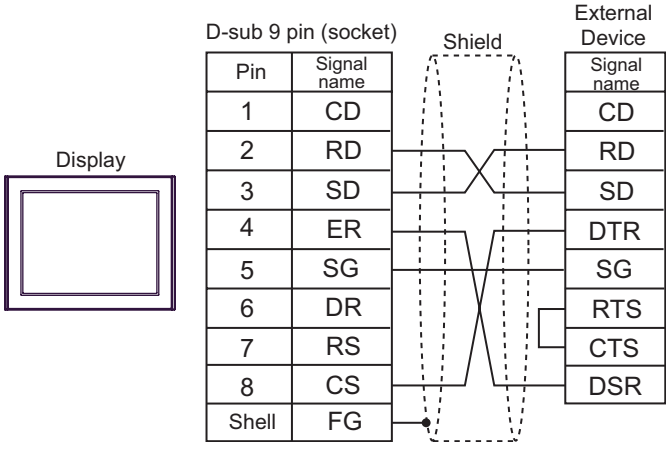
- When the External Device supports DTR/DSR control



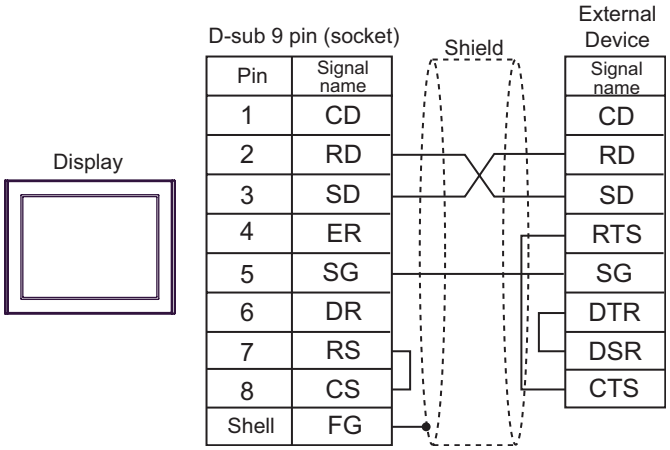
Prohibited:

- When ER in the GP is OFF, do not allow the host device to send.

B) When using your own cable (X control)



C) When using your own cable (without control method)



Cable Diagram 2

Display (Connection Port)	Cable		Remarks
GP[*1] (COM1) AGP-3302B (COM2) IPC[*2]	A	COM port conversion adapter by Pro-face CA3-ADPCOM-01 + Terminal block conversion adapter by Pro-face CA3-ADPTRM-01 + Your own cable	
	B	COM port conversion adapter by Pro-face CA3-ADPCOM-01 + 422 cable for AGP by Pro-face CA3-CBL422-01	
	C	COM port conversion adapter by Pro-face CA3-ADPCOM-01 + Multi-link cable for AGP by Pro-face CA3-CBLMLT-01	
	D	Your own cable	
GP[*3] (COM2)	E	Online adapter by Pro-face CA4-ADPONL-01 + Terminal block conversion adapter by Pro-face CA3-ADPTRM-01 + Your own cable	
	F	Online adapter by Pro-face CA4-ADPONL-01 + 422 cable for AGP by Pro-face CA3-CBL422-01	
	G	Online adapter by Pro-face CA4-ADPONL-01 + Multi-link cable for AGP by Pro-face CA3-CBLMLT-01	
	H	Online adapter by Pro-face CA4-ADPONL-01 + Your own cable	

*1 All GP models except AGP-3302B

*2 Only the COM port which can communicate by RS-422/485 (4 wire) can be used.
☞"■ COM Port of IPC" (page 4)

*3 All GP models except GP-3200 series and AGP-3302B

NOTE

- Control method when using the RS422 cable is XON/XOFF only. XON/XOFF control is enabled only for ASCII.

Forced:

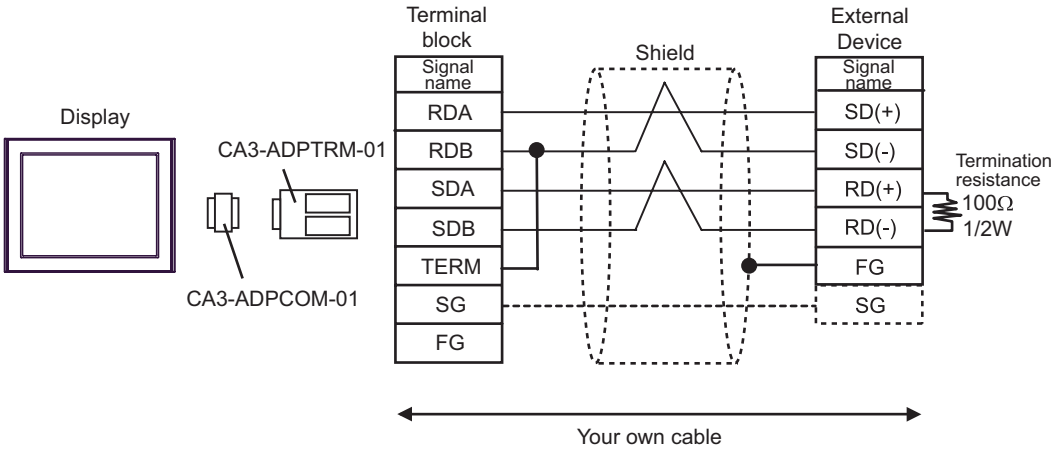
- Use the twist pair cable with approx. 50pF/m capacitance, 100 Ω characteristic impedance, made of 24AWG rod.

IMPORTANT

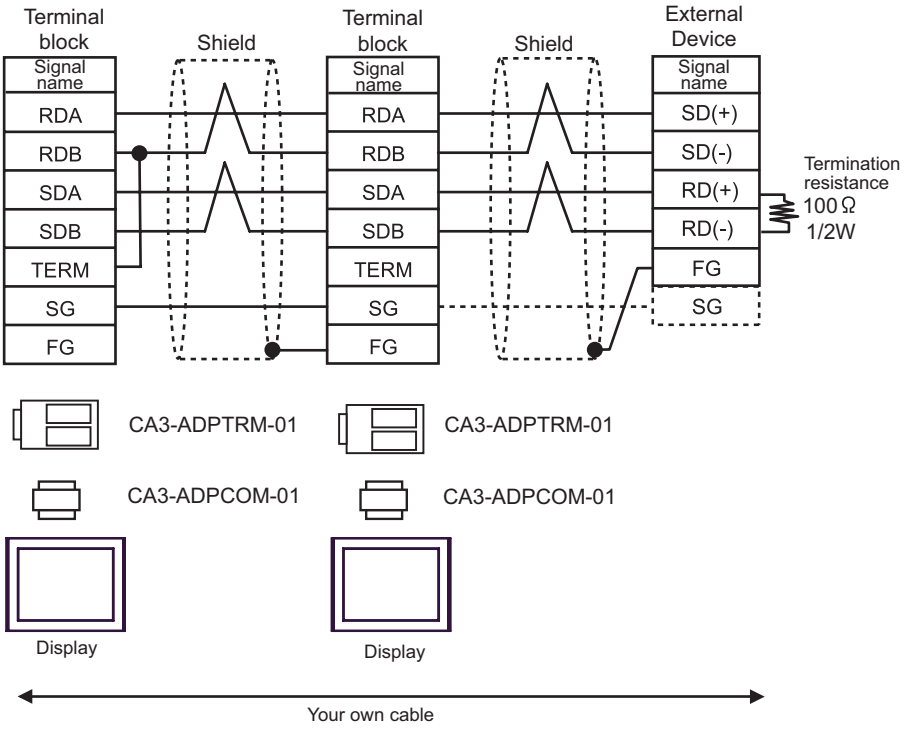
- The RS422 cable length is normally 1000m at maximum, but the cable length has the limit depending on the connecting host device. For connection, be sure to refer to the manual of the connecting host device.
- The connecting method or termination resistance varies depending on the connecting host device.
- Not isolated on the GP side.
- Always connect SG between GPs.
- When the External Device is isolated, SG connection between the External Device and the GP may not be required.

A) When using the COM port conversion adapter (CA3-ADPCOM-01), the connector terminal block conversion adapter (CA3-ADPTRM-01) by Pro-face and your own cable

- 1:1 Connection



1:n Connection

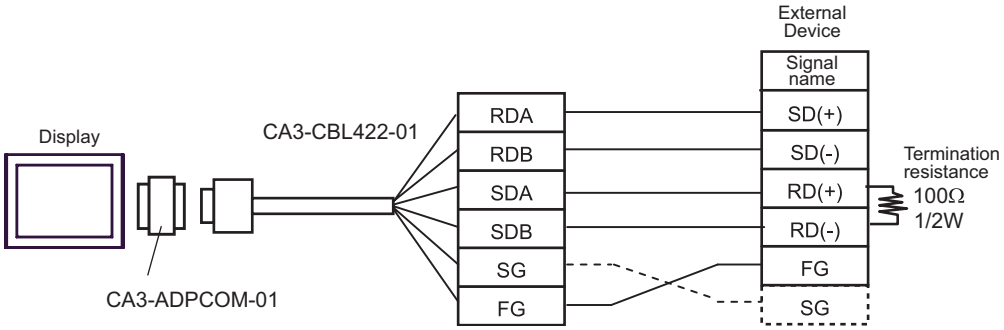


NOTE

- Connect RDB of CA3-ADPTRM-01 with TERM to insert the 100Ω 1/2W termination resistance between RDA and RDB on the GP side.

B) When using the COM port conversion adapter (CA3-ADPCOM-01), the 422 cable for AGP (CA3-CBL422-01) by Pro-face and your own cable

- 1:1 Connection

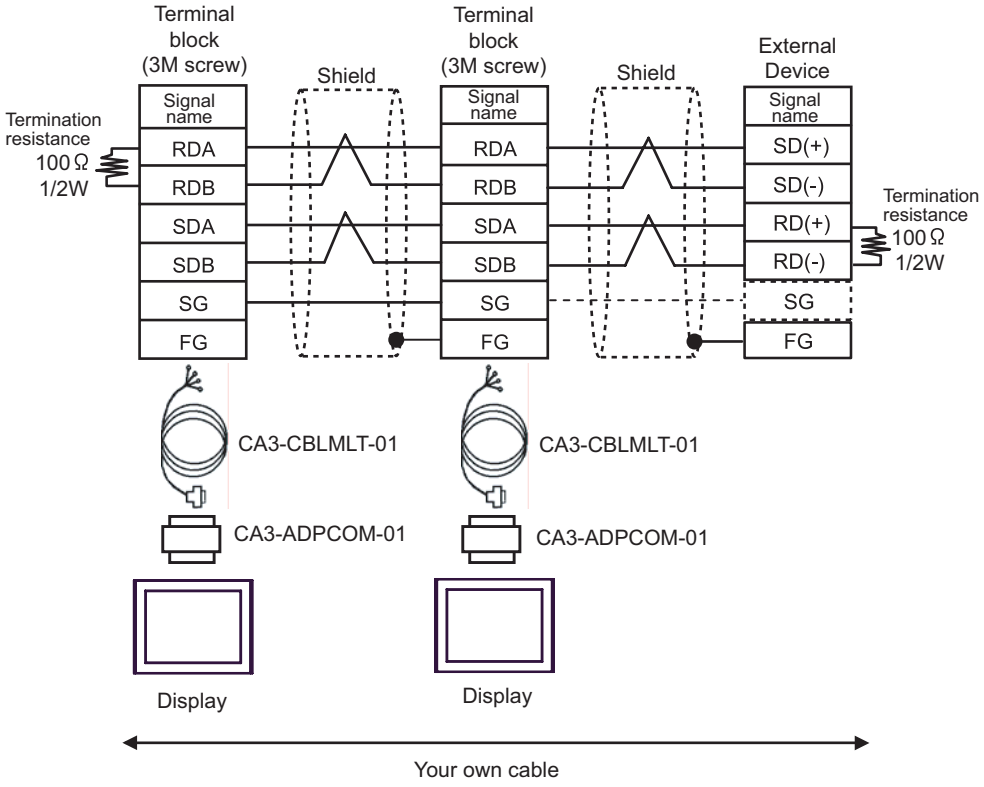


NOTE

- 100Ω termination resistance is inserted between RDA and RDB in CA3-CBL422-01.

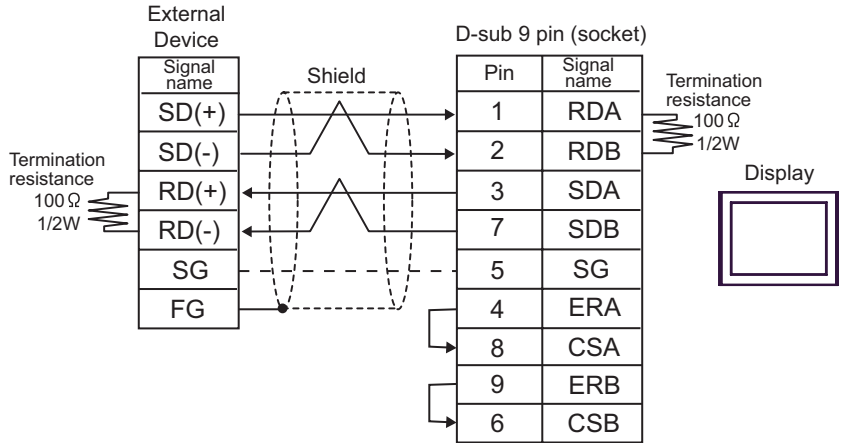
C) When using the COM port conversion adapter (CA3-ADPCOM-01), the multi-link cable for AGP (CA3-CBLMLT-01) by Pro-face and your own cable

- 1:n Connection

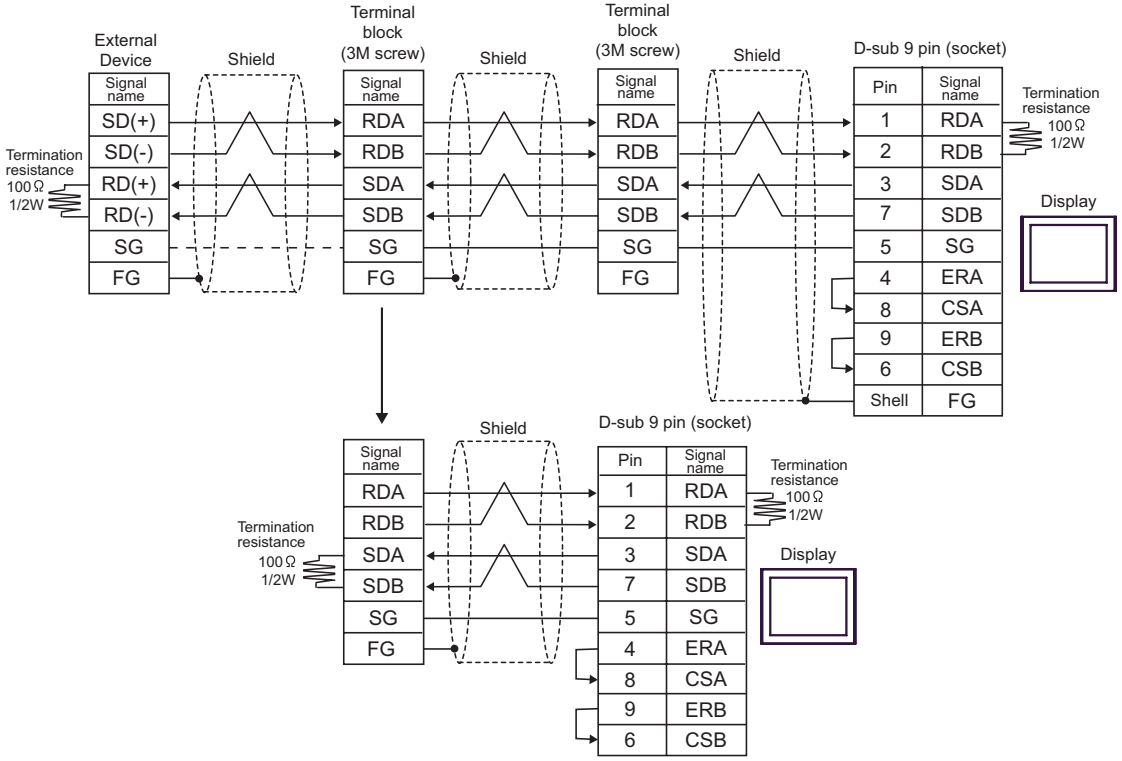


D) When using your own cable

- 1:1 Connection

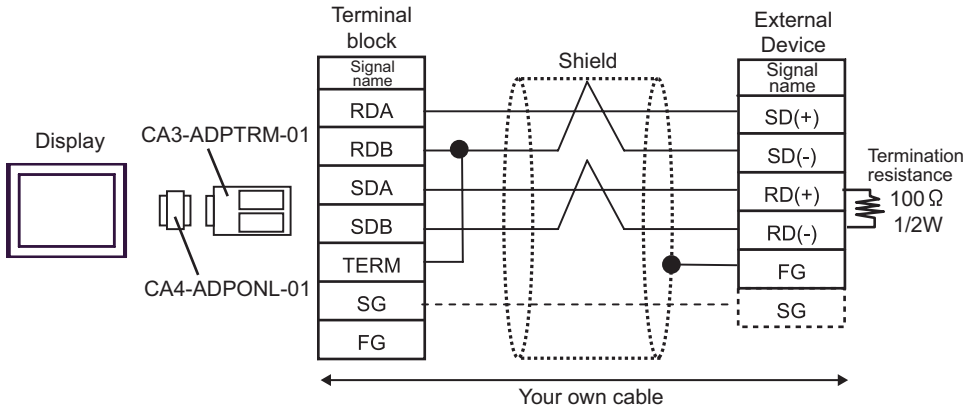


- 1:n Connection

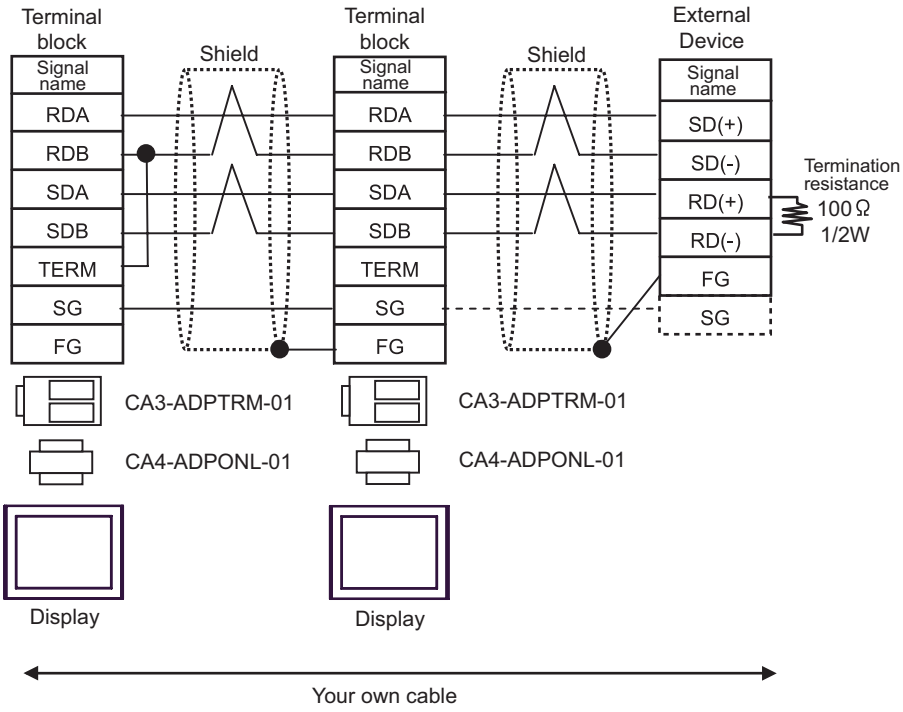


E) When using the online adapter (CA4-ADPONL-01), the connector terminal block conversion adapter (CA3-ADPTRM-01) by Pro-face and your own cable

- 1:1 Connection



- 1:n Connection

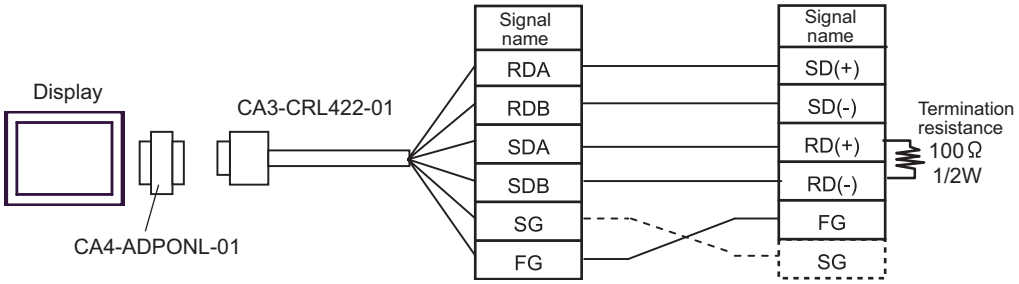


NOTE

- Connect RDA of CA3-ADPTRM-01 with TERM to insert the 100Ω1/2W termination resistance between RDA and RDB on the GP side.

F) When using the online adapter (CA4-ADPONL-01), the 422 cable for AGP (CA3-CBL422-01) by Pro-face.

- 1:1 Connection

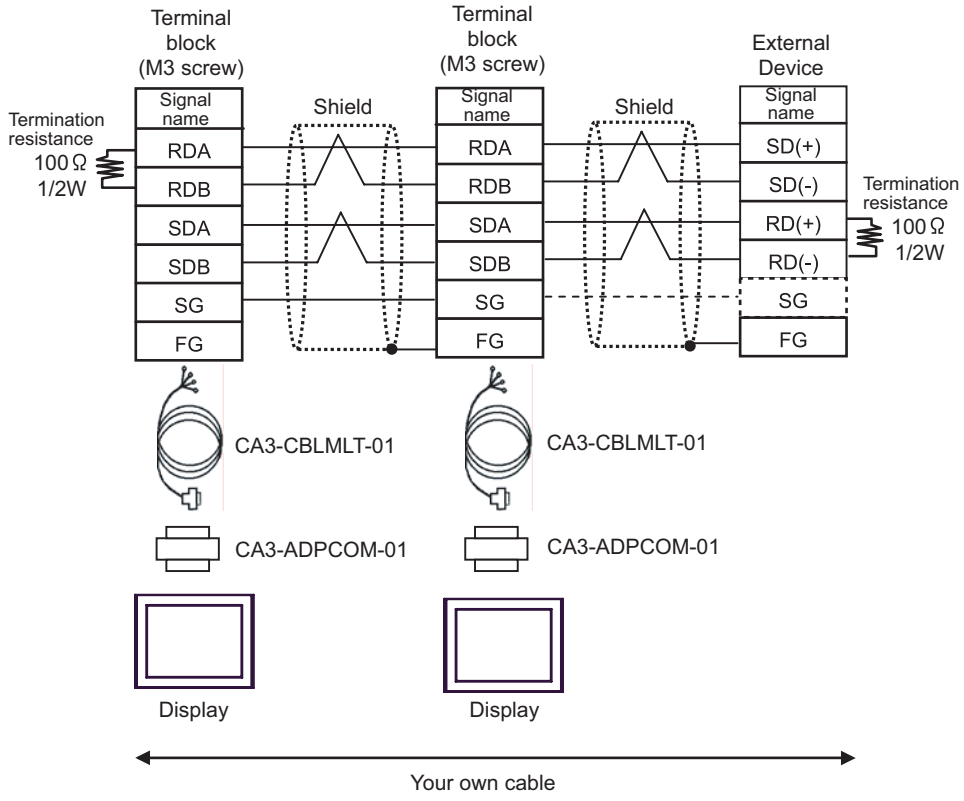


NOTE

- 100Ω termination resistance is inserted between RDA and RDB in CA3-CBL422-01.

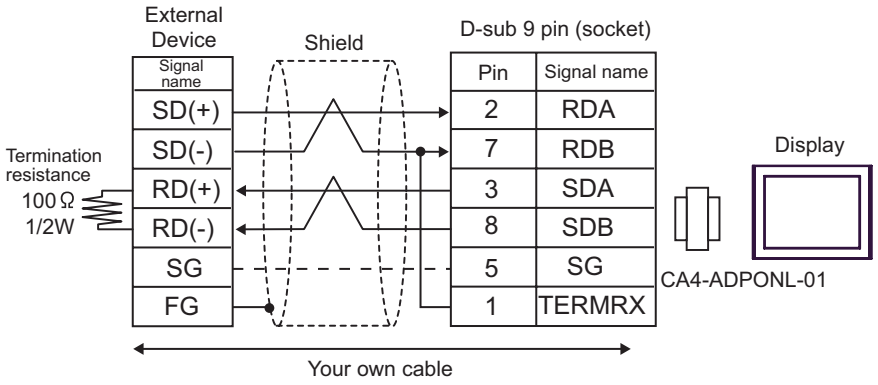
G) When using the online adapter (CA4-ADPONL-01) and the multi-link cable for AGP (CA3-CBLMLT-01) by Pro-face and your own cable

- 1:n Connection

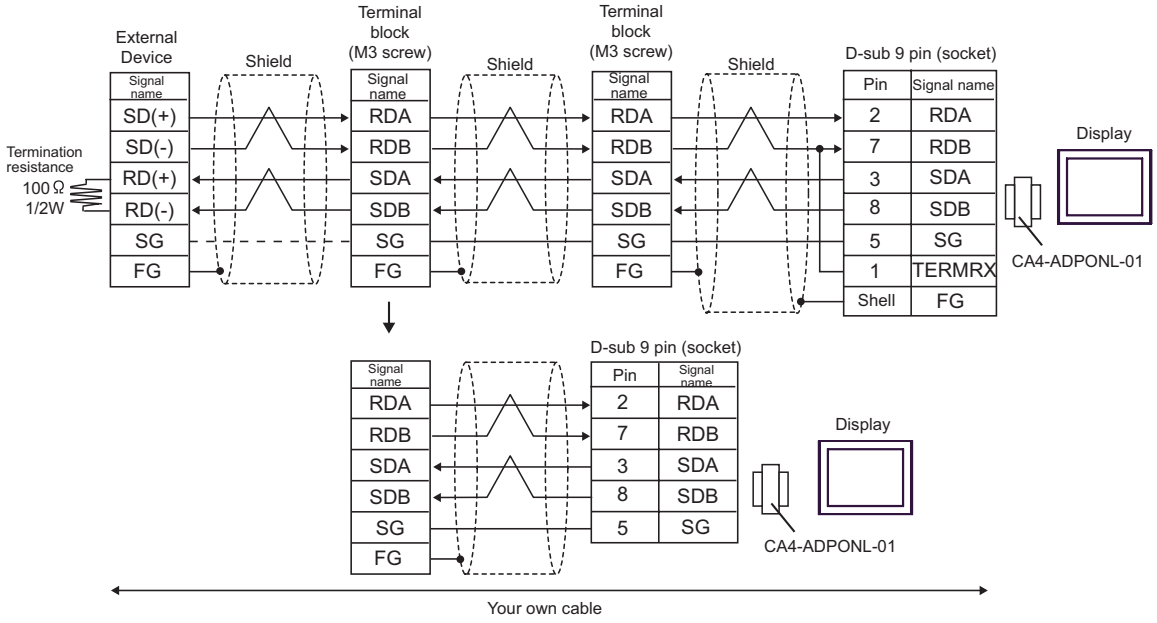


H) When using the online adapter (CA4-ADPONL-01) by Pro-face and your own cable

- 1:1 Connection



- 1:n Connection



Cable Diagram 3 (RS-422 (2 wire) connection)

Display (Connection Port)	Cable		Remarks
GP[*1] (COM1) AGP-3302B (COM2)	A	COM port conversion adapter by Pro-face CA3-ADPCOM-01 + Terminal block conversion adapter by Pro-face CA3-ADPTRM-01 + Your own cable	
	B	Your own cable	
GP[*2] (COM2)	C	Online adapter by Pro-face CA4-ADPONL-01 + Terminal block conversion adapter by Pro-face CA3-ADPTRM-01 + Your own cable	
	D	Online adapter by Pro-face CA4-ADPONL-01 + Your own cable	
IPC[*3]	E	COM port conversion adapter by Pro-face CA3-ADPCOM-01 + Terminal block conversion adapter by Pro-face CA3-ADPTRM-01 + Your own cable	
	F	Your own cable	

*1 All GP models except AGP-3302B

*2 All GP models except GP-3200 series and AGP-3302B

*3 Only the COM port which can communicate by RS-422/485 (2 wire) can be used.
☞"■ COM Port of IPC" (page 4)

NOTE

- Control method when using the RS422 cable is XON/XOFF only. XON/XOFF control is enabled only for ASCII.

Forced:

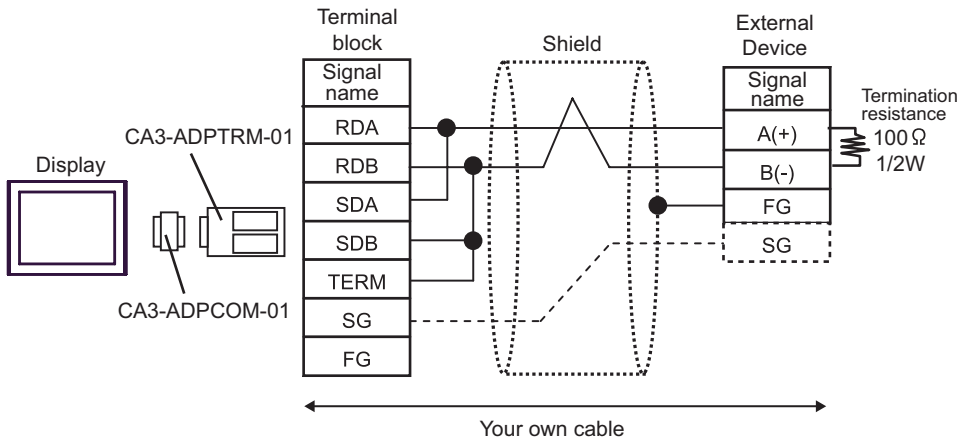
- Use the twist pair cable with approx. 50pF/m capacitance, 100Ω characteristic impedance, made of 24AWG rod.

IMPORTANT

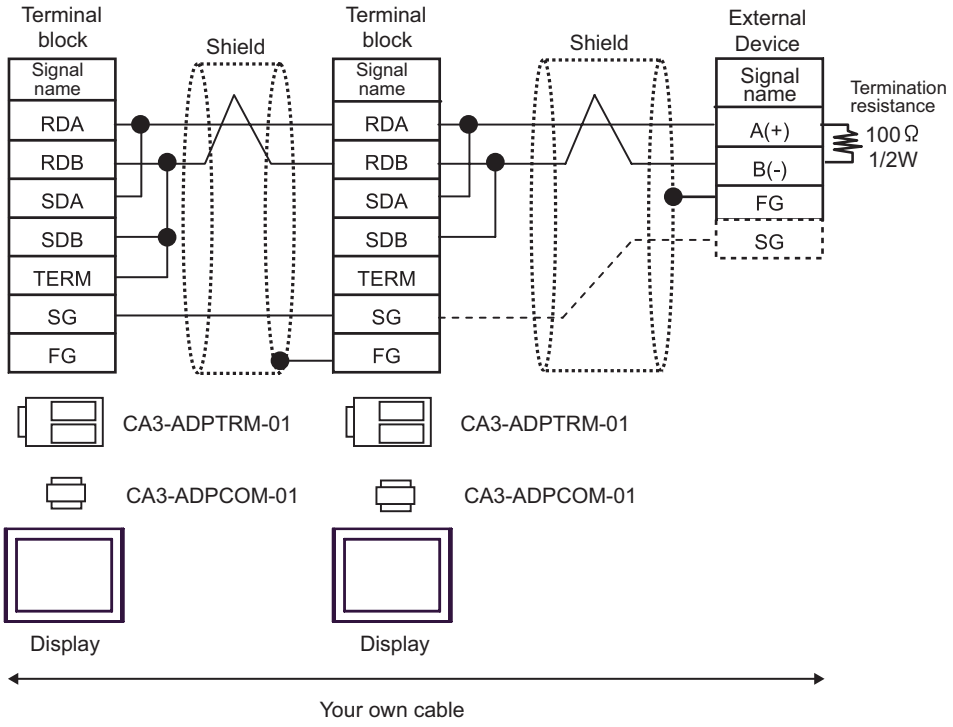
- The RS422 cable length is normally 1000m at maximum, but the cable length has the limit depending on the connecting host device. For connection, be sure to refer to the manual of the connecting host device.
- The connecting method or termination resistance varies depending on the connecting host device. Connect SG if provided.

A) When using the COM port conversion adapter (CA3-ADPCOM-01), the connector terminal block conversion adapter (CA3-ADPTRM-01) by Pro-face and your own cable

- 1:1 Connection



- n:1 Connection

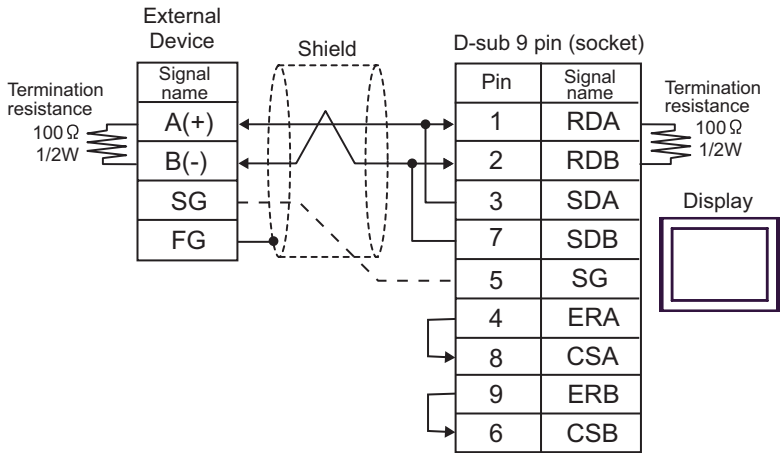


NOTE

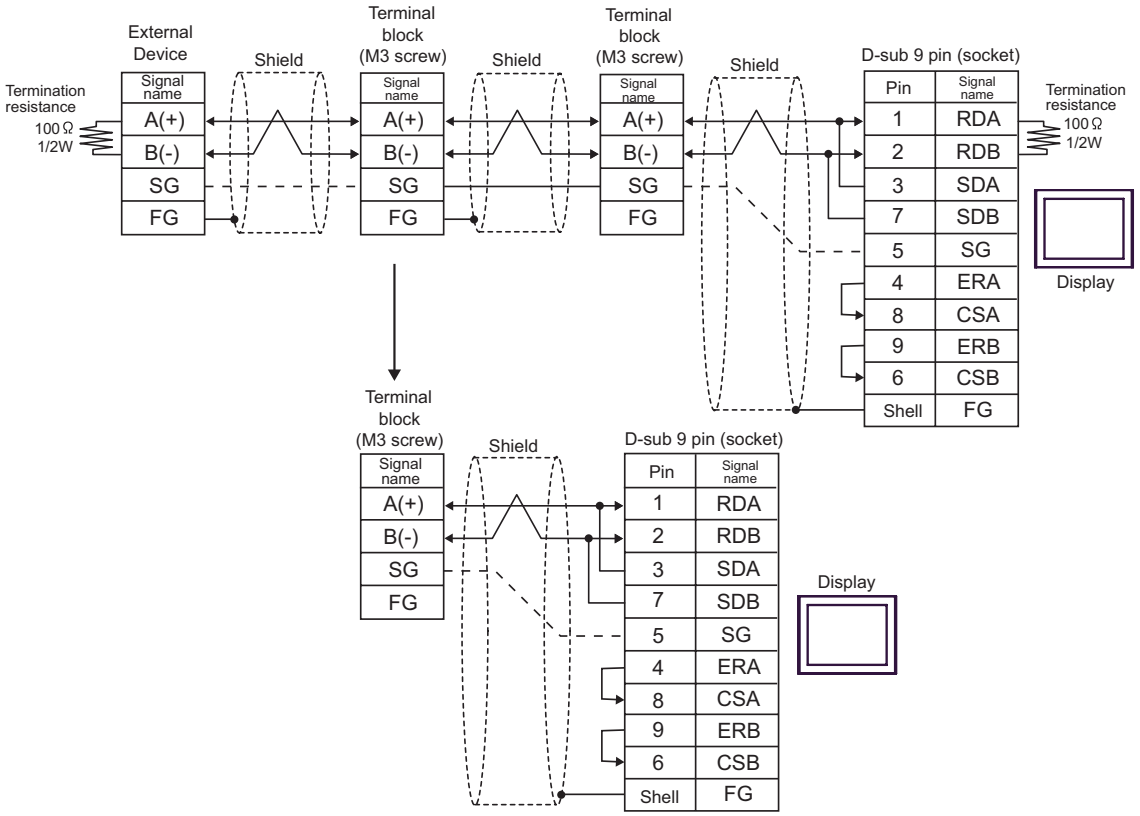
- Connect RDB of CA3-ADPTRM-01 with TERM to insert the 100Ω1/2W termination resistance between RDA and RDB on the Display.

B) When using your own cable

- 1:1 Connection

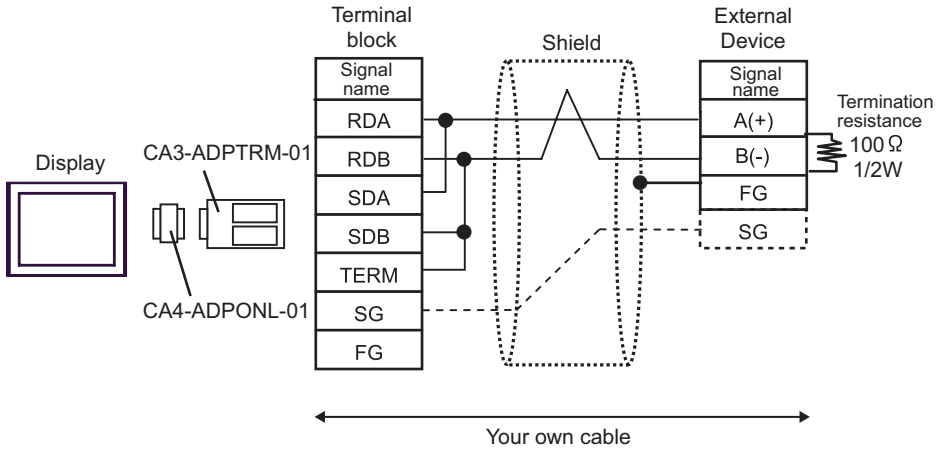


- n:1 Connection

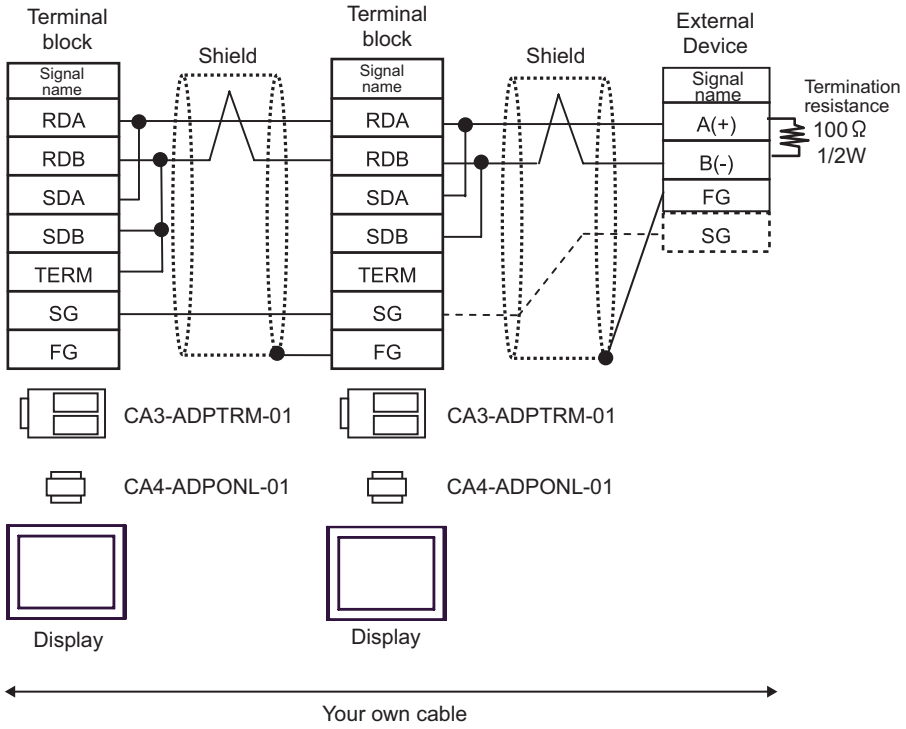


C) When using the online adapter (CA4-ADPONL-01), the connector terminal block conversion adapter (CA3-ADPTRM-01) by Pro-face and your own cable

- 1:1 Connection

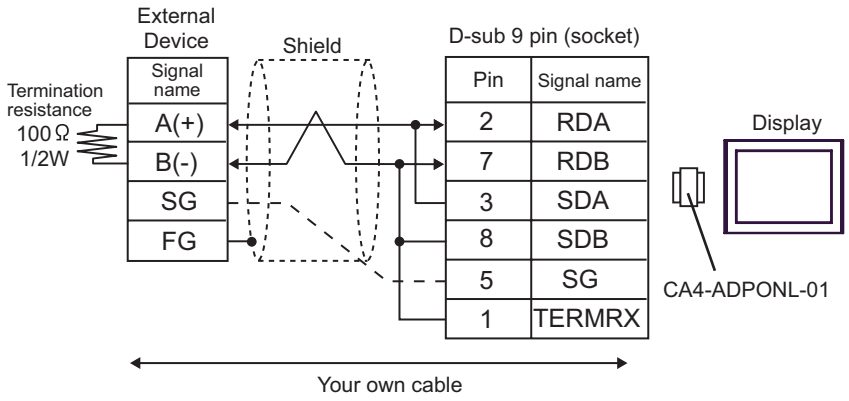


- n:1 Connection

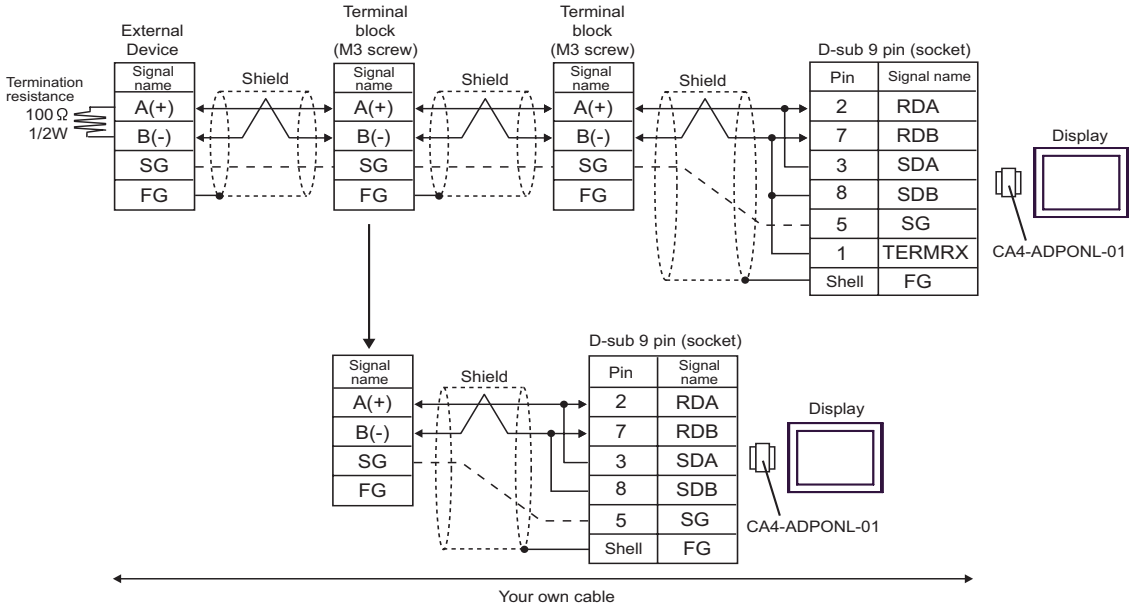


D)When using the online adapter (CA4-ADPONL-01) by Pro-face and your own cable

- 1:1 Connection

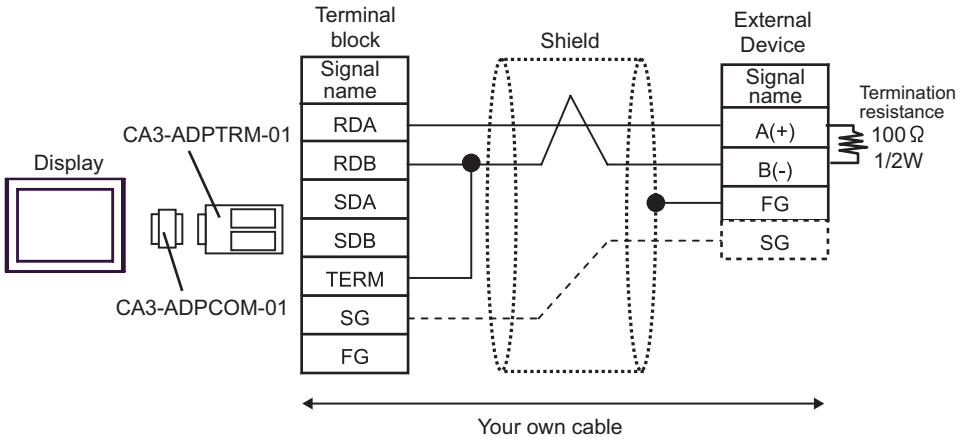


- n:1 Connection

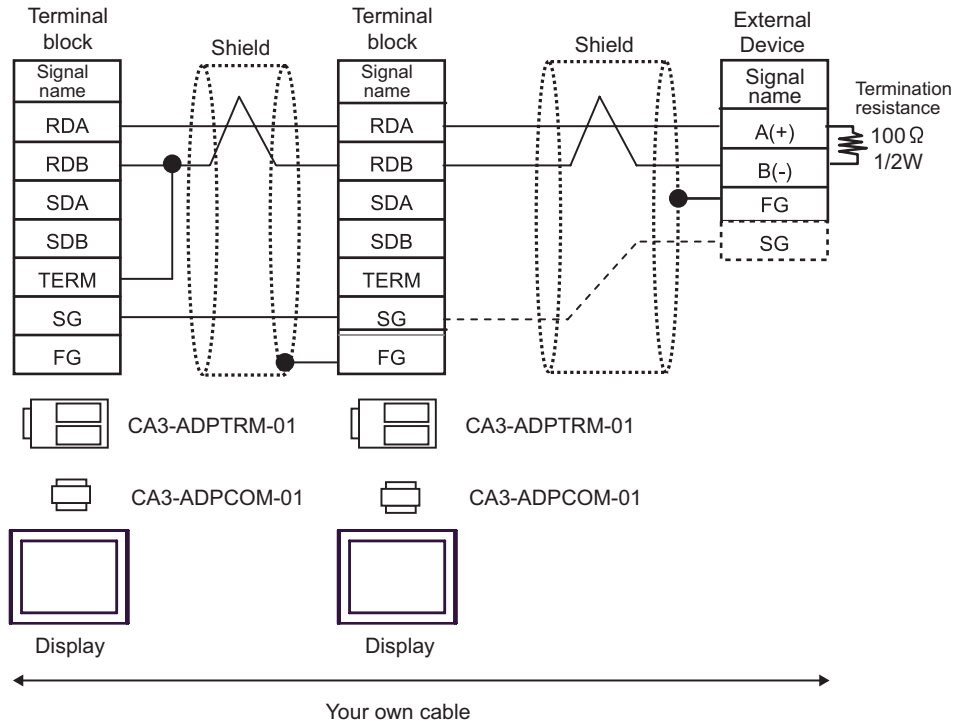


E) When using the COM port conversion adapter (CA3-ADPCOM-01), the connector terminal block conversion adapter (CA3-ADPTRM-01) by Pro-face and your own cable

- 1:1 Connection



- n:1 Connection

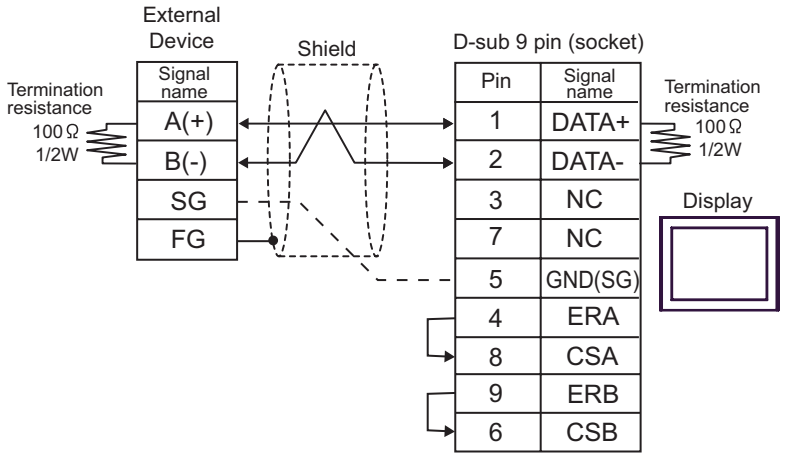


NOTE

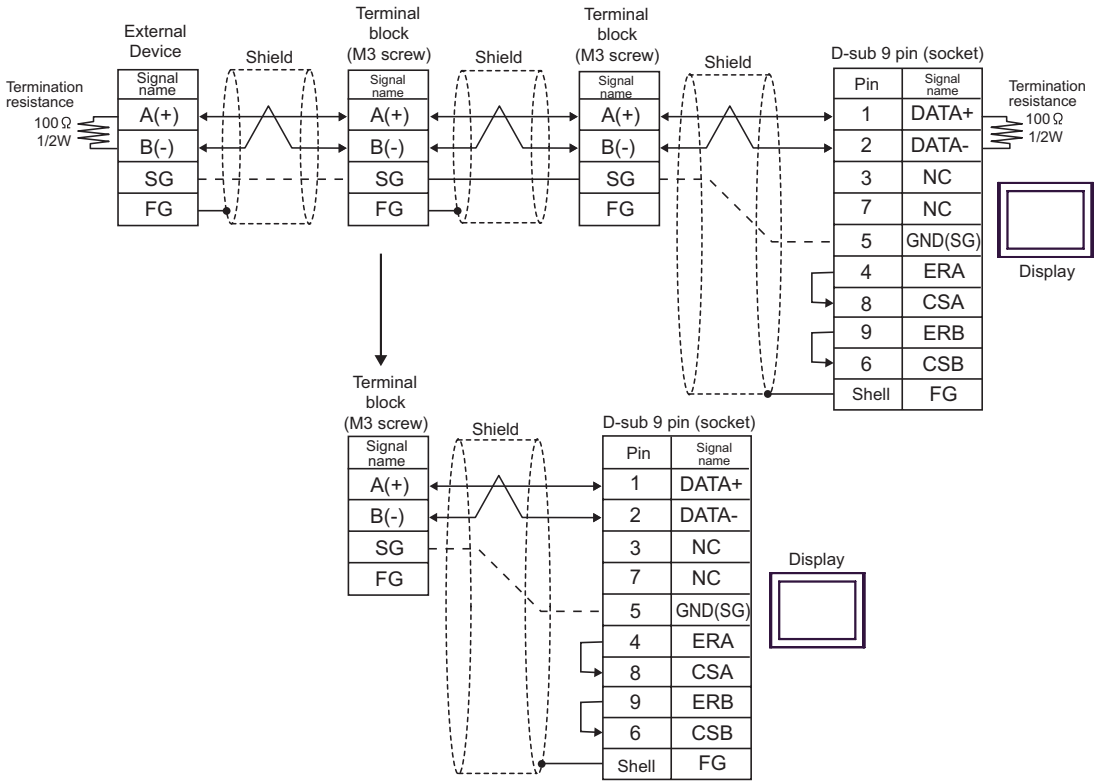
- Connect RDB of CA3-ADPTRM-01 with TERM to insert the 100Ω1/2W termination resistance between RDA and RDB on the Display.

F) When using your own cable

- 1:1 Connection



- n:1 Connection



6 Supported Device

Range of supported device address is shown in the table below. Please note that the actually supported range of the devices varies depending on the External Device to be used. Please check the actual range in the manual of your External Device.

Device	Bit Address	Word Address	32 bits	Remarks
Internal Device	000000 - 999915	0000 - 9999	H/L	

NOTE

- Please refer to the precautions on manual notation for icons in the table.
  ☞"Manual Symbols and Terminology"

7 Device Code and Address Code

Use device code and address code when you select "Device Type & Address" for the address type in data displays.

Device	Device Name	Device Code (HEX)	Address Code
Internal Device	-	0000	Value of word address

8 Error Messages

Error messages are displayed on the Display screen as follows: "No.: Device Name: Error Message (Error Occurrence Area)". Each description is shown below.

Item	Description
No.	Error No.
Device Name	Name of the External Device where error occurs. Device name is a title of the External Device set with GP-Pro EX. (Initial value [PLC1])
Error Message	Displays messages related to the error which occurs.
Error Occurrence Area	Displays IP address or device address of the External Device where error occurs, or error codes received from the External Device. NOTE • IP address is displayed such as "IP address(Decimal): MAC address(Hex)". • Device address is diplayed such as "Address: Device address". • Received error codes are displayed such as "Decimal[Hex]".

Display Examples of Error Messages

"RHAA035: PLC1: Error has been responded for device write command (Error Code: 2 [02])"

NOTE
- Please refer to the manual of the External Device for more detail of received error codes.
- Please refer to "When an error message is displayed (Error code list)" of "Maintenance/ Troubleshooting" for a common error message to the driver.

■ Error Codes Specific to the External Device

Error Code	Description
06	The checksum code is not corresponding.
10	Undefined code has been received.
12	The specified number of data elements does not match the number of data elements received.
24	The specified attribute code is out of the permissible range.
25	The contrast adjustment command was sent to the model that cannot adjust the contrast.
26	The specified contrast setting is out of the permissible range.
27	The brightness adjustment command was sent to the model that cannot adjust the brightness.
28	The specified brightness setting is out of the permissible range.
FA	The specified address in the system area is out of the permissible range.
FB	An attempt has been made to write to or read from outside the system area.
FC	A data block of an improper format has been received by the Display Unit.
FF	The Display Unit could not send data for longer than 10 seconds.

■ Error Messages Specific to this Driver

Message ID	Error Message	Cause and Solution
RHxx128	Memory Link: Checksum does not match the data actually received (ErrorCode: 06 Destination: %s)	The check sum is wrong in the sending sentence. Correct the sending sentence.
RHxx129	Memory Link: Undefined code has been received (ErrorCode: 10 Destination: %s)	The command in the sending sentence is wrong. Correct the sending sentence.
RHxx130	Memory Link: The specified number of data elements does not match the number of data elements received (ErrorCode: 12 Destination: %s)	The number of data elements in the sending sentence is wrong. Correct the sending sentence.
RHxx146	Memory Link: The specified address in the system area is out of the permissible range (ErrorCode: FA Destination: %s)	The system area specification in the sending sentence is wrong. Correct the sending sentence.
RHxx147	Memory Link: An attempt has been made to write to or read from outside the system area (ErrorCode: FB Destination: %s)	Exceeded the system area range. Correct the sending sentence.
RHxx148	Memory Link:A data block of an improper format has been received by the AGP (ErrorCode: FC Destination: %s)	The format in the sending sentence is wrong. Correct the sending sentence.
RHxx149	Memory Link: The AGP has been unable to send data (ErrorCode: FF Destination: COM port name)	The GP cannot send the command. Check the cable wiring.
RHxx150	Memory Link: The specified attribute code is out of the permissible range (ErrorCode: 24 Destination: %s)	The attribute designation in the sending sentence is wrong. Correct the sending sentence.
RHxx151	Memory Link: The contrast cannot be adjusted with this model (ErrorCode: 25 Destination: %s)	The contrast adjustment command was sent to the model that cannot adjust the contrast. Check the GP model.
RHxx152	Memory Link: The specified contrast setting is out of the permissible range (ErrorCode: 26 Destination: %s)	The contrast setting value in the sending sentence is wrong. Correct the sending sentence.
RHxx153	Memory Link: The brightness cannot be adjusted with this model (ErrorCode: 27 Destination: %s)	The brightness adjustment command was sent to the model that cannot adjust the brightness. Check the GP model.
RHxx154	Memory Link: The specified brightness setting is out of the permissible range (ErrorCode: 28 Destination: %s)	The brightness setting value in the sending sentence is wrong. Correct the sending sentence.

*COM port name for SIO, source port No. for TCP and destination IP address for UDP are shown in %s after Destination.

Memo

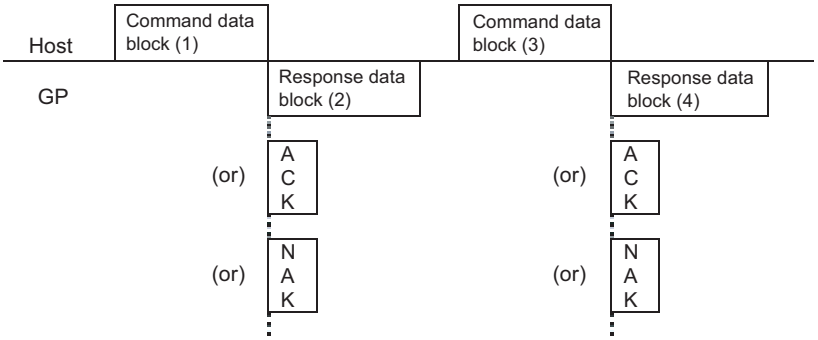
9 Memory Link Command (Serial Communication)

9.1 Basic Communication Protocol Control

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

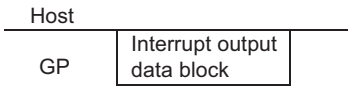
9.1.1 SIO

■ Host to GP Data Transfer



- (1) and (3) (Command Data area) store the data to be transmitted from the host device to the GP.
- After the GP analyzes the Command Data, (2) and (4) (Response Data area) store the result of "ACK" or "NAK", or no response.
- Please send the Command Data (3) from the host device after receiving the Response Data (2) from the GP.

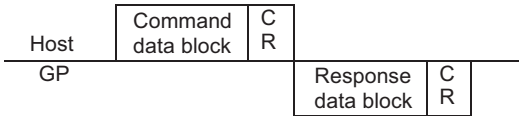
■ GP to Host Data Transfer



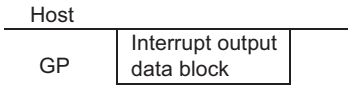
- When entered with the touch panel, the GP sends the data stored in the interrupt data area to the host device. (Interrupt Output)
- Interrupt output does not occur in case of 1:n or RS422/485 (2wire) connection. Please refer to the interrupt output request.

9.1.2 Communication in SIO Convert Mode

■ Host to GP Data Transfer



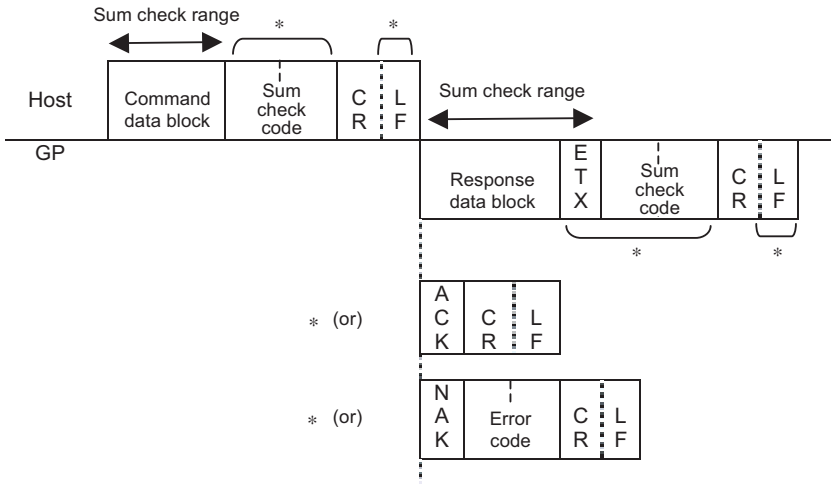
■ GP to Host Data Transfer (Interrupt Output)



- You cannot use the interrupt output in case of RS422/485(2wire).

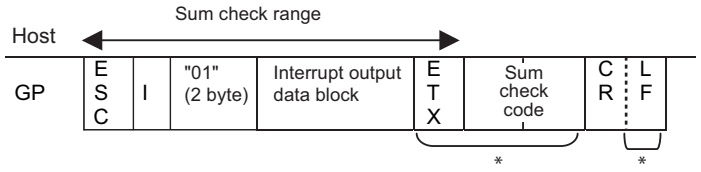
9.1.3 Communication in SIO Extend Mode (1:1 ASCII)

■ Host to GP Data Transfer



- Marked area with asterisk (*) may not be added depending on the setting.

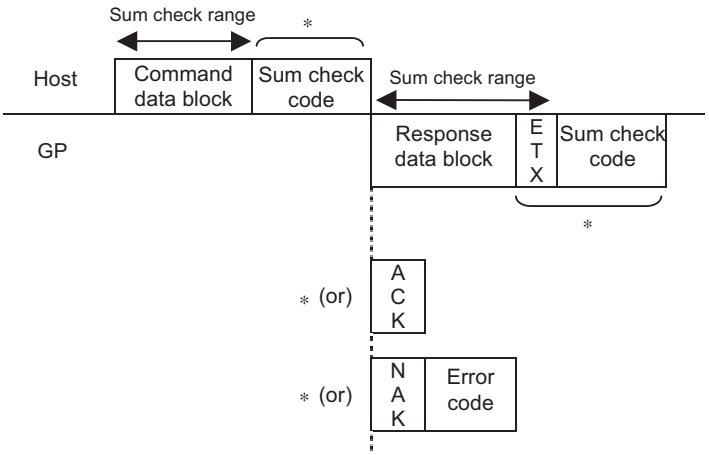
■ GP to Host Data Transfer (Interrupt Output)



- Marked area with asterisk (*) may not be added depending on the setting.
- In case of RS422/485 (2wire) or UDP connection, please use "Interrupt Output Request Command" to perform the interrupt output.

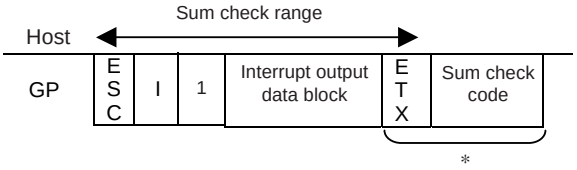
9.1.4 Communication in SIO Extend Mode (1:1 Binary)

■ Host to GP Data Transfer



- Marked area with asterisk (*) may not be added depending on the setting.

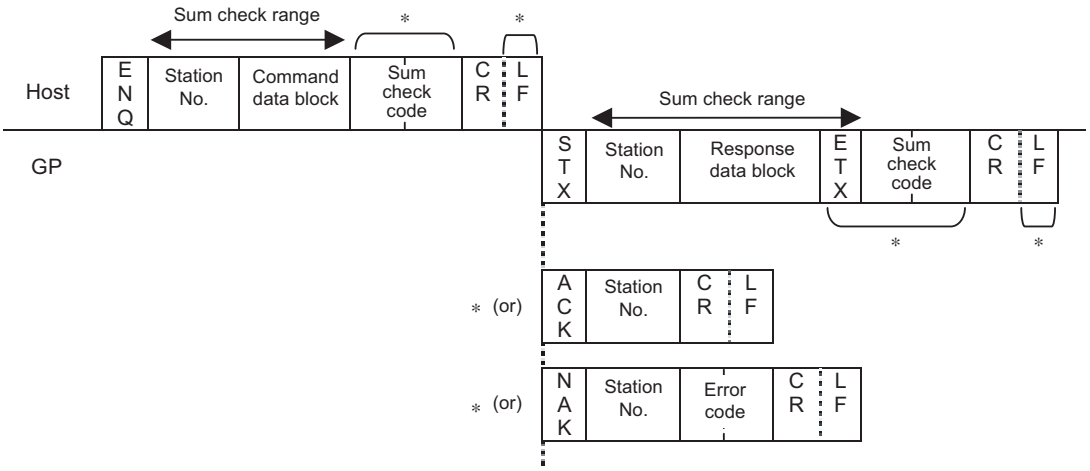
■ GP to Host Data Transfer (Interrupt Output)



- Marked area with asterisk (*) may not be added depending on the setting.
- You cannot use the XON/XOFF control in the binary mode. Use the ER control and enable the response (ACK/NAK) for communication.
- In case of RS422/485 (2wire), please use "Interrupt Output Request Command" to perform the interrupt output.

9.1.5 Communication in SIO Extend Mode (1:n ASCII)

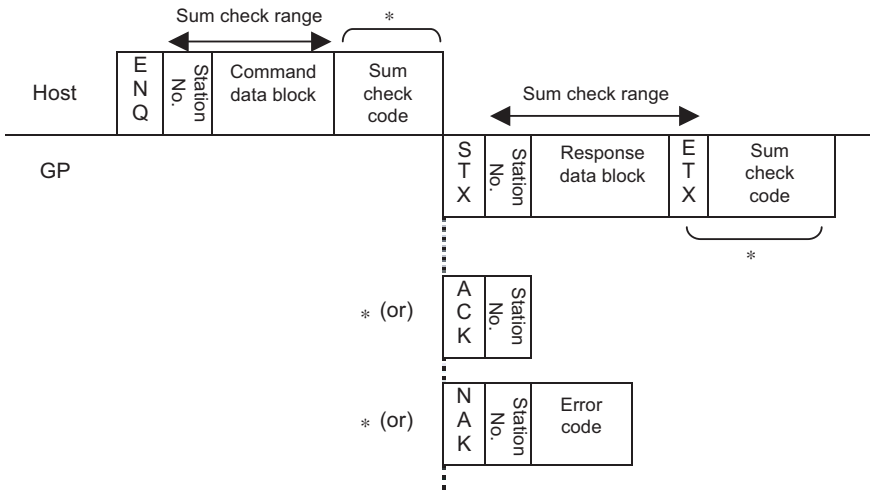
■ Host to GP Data Transfer



- Marked area with asterisk (*) may not be added depending on the setting.
- You can set the station No. to "FF" to transfer the command to all stations simultaneously. Note that ACK or NAK response will not be performed. In this case, please set the interval of 100ms or more until sending next command after sending the first command.
  In addition, note that you cannot use the "Read from System Area" (ESC R) or "Brightness/Contrast Current Value" (ESC $) command which requires the response data.
- In case of 1:n connection, please use "Interrupt Output Request Command" to perform the interrupt output.

9.1.6 Communication in SIO Extend Mode (1:n Binary)

■ Host to GP Data Transfer



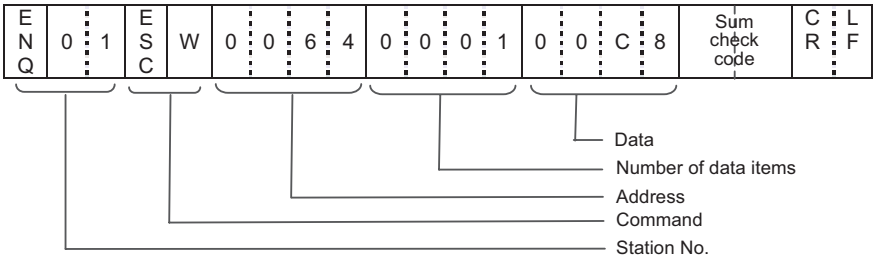
- Marked area with asterisk (*) may not be added depending on the setting.
- You can set the station No. to "FF" to transfer the command to all stations simultaneously. Note that ACK or NAK response will not be performed. In this case, please set the interval of 100ms or more until sending next command after sending the first command.
  In addition, note that you cannot use the "Read from System Area" (ESC R) or "Brightness/Contrast Current Value" (ESC $) command which requires the response data.
- You cannot use the XON/XOFF control in the binary mode. Use the ER control and enable the response (ACK/NAK) for communication.
- In case of 1:n connection, please use "Interrupt Output Request Command" to perform the interrupt output.

9.1.7 Sum Check Code

The sum check code is the lower one byte (8 bits) of the sum of all data included in the sum check range.

In the ASCII mode, data is converted into ASCII code before summing. Then, the lower 2 digits of the hexadecimal sum of all data is used as the sum check code.
In the binary mode, the lower byte of the sum of all data is used as the sum check code.

Example: Extend Mode, 1:n ASCII
The following data block writes "200" (decimal) to address 100 in the system area:



ASCII	30H	+	31H	+	1BH	+	57H	+	30H	+	30H	+	36H	+	34H	+	30H	+	30H	+				
	(0)		(1)		(ESC)		(W)		(0)		(0)		(6)		(4)		(0)		(0)					
ASCII	30H	+	31H		30H	+	30H	+	43H	+	38H													
	(0)		(1)		(0)		(0)		(C)		(8)													
										=	339H													

Lower two digits, "39" (33H, 39H) are used as the sum check code.

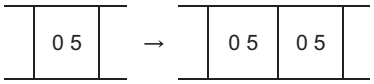
9.2 Notes on SIO 1:N Binary Communication

In the SIO Extend Mode/1:N binary communication, double process occurs.

9.2.1 Host to GP Data Transfer

■ ENQ

In the transmission from the host, when the data for "Sum Check Range" or "Sum Check" includes "05h" that is same as the ENQ code, add "05h" just before the data to transmit.

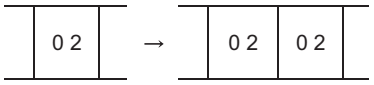


Note that the added "05h" is not included in the data number when the Command Data area has "Data Number".

9.2.2 Host to GP Data Transfer

■ STX

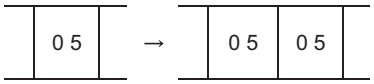
In the response from the GP series, when the data for "Sum Check Range" or "Sum Check" includes "02h" that is same as the STX code, add "02h" just before the data to respond.



Note that the added "02h" is not included in the data number when the Response Data area has "Data Number".

■ ENQ

For the 2-wire 1:N connection, when the data for "Sum Check Range" or "Sum Check" includes "05h" that is same as the ENQ code in the response from the GP series, add "05h" just before the data to transmit.

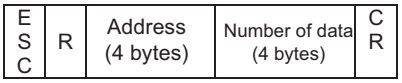


9.3 Command Format

9.3.1 Read Format

■ SIO Convert Mode

◆ Command data block (from Host)



Setting range

Address: 0000H to 270FH (0 to 9999)

Number of data packets: 0001H to 0100H (1 to 256)

Be sure to make all data entries in ASCII code format.

◆ GP Response data block (from GP)

- When there is no error

<Setting range>

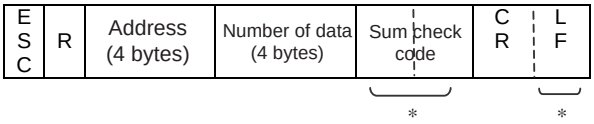
Data: 0000H to FFFFH

- If an error occurs

NAK response

■ SIO Extend Mode, ASCII

◆ Command data block (from Host)



Marked area with asterisk (*) may not be added depending on the setting.

<Setting range>

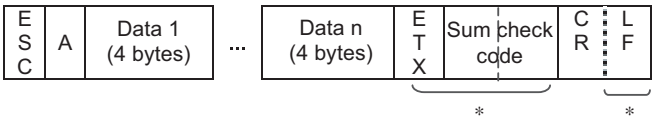
Address: 0000H to 270FH (0 to 9999)

Number of data packets: 0001H to 0100H (1 to 256)

Be sure to make all data entries in ASCII code format.

◆ GP Response data block (from GP)

- When there is no error



Marked area with asterisk (*) may not be added depending on the setting.

<Setting range>

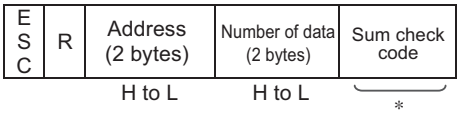
Data: 0000H to FFFFH

- If an error occurs

NAK response

■ SIO Extend Mode, Binary

◆ Command data block (from Host)



Marked area with asterisk (*) may not be added depending on the setting.

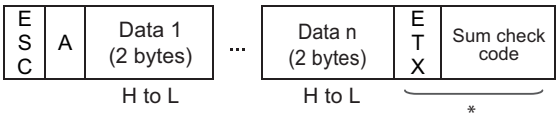
<Setting range>

Address: 0000H to 270FH (0 to 9999)

Number of data packets: 0001H to 0200H (1 to 512)

◆ GP Response data block (from GP)

- When there is no error



Marked area with asterisk (*) may not be added depending on the setting.

<Setting range>

Data: 0000H to FFFFH

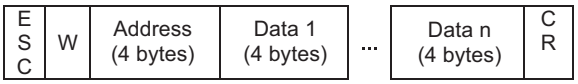
- If an error occurs

NAK response

9.3.2 Write Format

■ SIO Convert Mode

◆ Command data block (from Host)



<Setting range>

Address: 0000H to 270FH (0 to 9999)

Data: 0000H to FFFFH

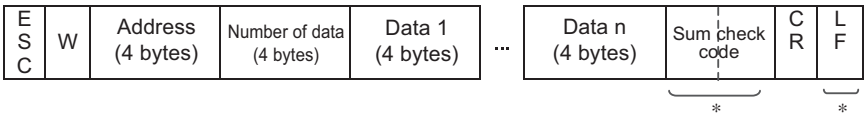
Be sure to make all data entries in ASCII code format.

In the Convert Mode, there is no response command from the GP.

In the Convert Mode, there is no limit for number of write data packets.

■ SIO Extend Mode, ASCII

◆ Command data block (from Host)



Marked area with asterisk (*) may not be added depending on the setting.

<Setting range>

Address: 0000H to 270FH (0 to 9999)

Number of data packets: 0001H to 0100H (1 to 256)

Data: 0000H to FFFFH

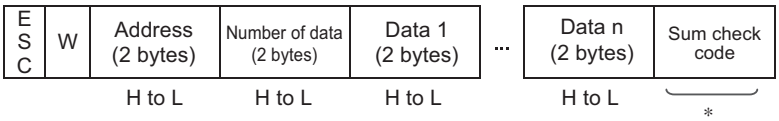
Be sure to make all data entries in ASCII code format.

◆ GP Response data block (from GP)

ACK or NAK response

■ SIO Extend Mode, Binary

◆ Command data block (from Host)



Marked area with asterisk (*) may not be added depending on the setting.

<Setting range>

Address: 0000H to 270FH (0 to 9999)

Number of data packets: 0001H to 0200H (1 to 512)

Data: 0000H to FFFFH

◆ GP Response data block (from GP)

ACK or NAK response

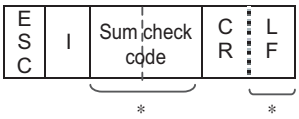
9.3.3 Interrupt Output Requests

Here, commands are explained that are used by the GP in Extend Mode, when using [1:n ASCII], [1:n Binary] or 2-wire type communication, to output an interrupt code via the GP unit's System Area's Absolute Value Write, etc. from the GP unit to the Host.

When using 2-wire type connection, be sure to perform the following settings even for a 1:1 connection.

■ SIO, ASCII Mode

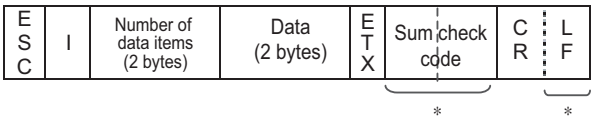
◆ Command data block (from Host)



Marked area with asterisk (*) may not be added depending on the setting.

◆ GP Response data block (from GP)

- When there is no error



- If an error occurs

  NAK response

<Setting range>

Number of data packets

When a request command is sent from the host, this value defines the previously issued interrupt output's number of data items.

When all the previously issued interrupt output data is acquired, this data frequency (number) must be sent.

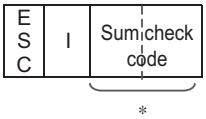
Data

The data (00H to FEH) is converted into a 2-digit ASCII code (HEX) before being output.

"00" will be entered in this field if there is no data to be output.

■ SIO, Binary Mode

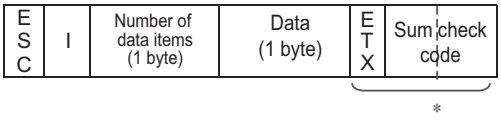
◆ Command data block (from Host)



Marked area with asterisk (*) may not be added depending on the setting.

◆ GP Response data block (from GP)

- When there is no error



Marked area with asterisk (*) may not be added depending on the setting.

- If an error occurs

  NAK response

<Setting range>

Number of data packets

When a request command is sent from the host, this value defines the previously issued interrupt output's number of data items.

When all the previously issued interrupt output data is acquired, this data frequency (number) must be sent.

Data

The data value (00H to FEH) is output.

"00" will be entered in this field if there is no data to be output.

9.3.4 Brightness and Contrast Adjustments

The format of the command data block containing the ESC # command (brightness and contrast adjustment command) is shown below. Note that brightness or contrast cannot be adjusted with some GP types.

■ SIO Convert Mode

◆ Command data block (from Host)

ESC	#	Attribute (4 bytes)	Setting (4 bytes)	CR

<Setting range>
Attribute: 0000H to 0001H (0: Contrast, 1: Brightness)
Settings: Please refer to "■ Brightness/Contrast Table" (page 45).

Be sure to make all data entries in ASCII code format.

◆ GP Response data block (from GP)

No response data.

■ SIO Extend Mode, ASCII

◆ Command data block (from Host)

ESC	#	Attribute (4 bytes)	Setting (4 bytes)	Sum check code	CR	LF
				*		*

Marked area with asterisk (*) may not be added depending on the setting.

<Setting range>
Attribute: 0000H to 0001H (0: Contrast, 1: Brightness)
Settings: Please refer to "■ Brightness/Contrast Table" (page 45).

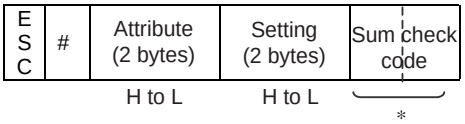
Be sure to make all data entries in ASCII code format.

◆ GP Response data block (from GP)

ACK or NAK response

■ SIO Extend Mode, Binary

◆ Command data block (from Host)



<Setting range>

Attribute: 0000H to 0001H (0: Contrast, 1: Brightness)

Settings: Please refer to " ■ Brightness/Contrast Table" (page 61).

Be sure to make all data entries in ASCII code format.

◆ GP Response data block (from GP)

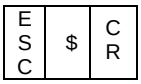
ACK or NAK response

9.3.5 Brightness and Contrast Current Value

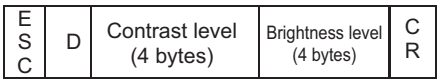
The format of the command data block to acquire the brightness and contrast current values with the command is shown below. Note that the brightness or contrast level is not available with some GP types.

■ SIO Convert Mode

◆ Command data block (from Host)

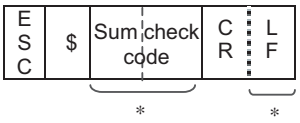


◆ GP Response data block (from GP)



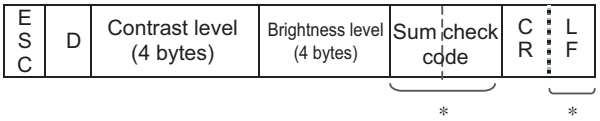
■ SIO Extend Mode, ASCII

◆ Command data block (from Host)



Marked area with asterisk (*) may not be added depending on the setting.

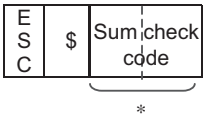
◆ GP Response data block (from GP)



Marked area with asterisk (*) may not be added depending on the setting.

■ SIO Extend Mode, Binary

◆ Command data block (from Host)



Marked area with asterisk (*) may not be added depending on the setting.

◆ GP Response data block (from GP)

Marked area with asterisk (*) may not be added depending on the setting.

■ Brightness/Contrast Table

GP	Brightness Setting Range	Contrast Setting Range
AGP-3302B	0(Bright) to 7(Dark)	0(Bright) to 7(Dark)
AGP-3301L	0(Bright) to 7(Dark)	0(Bright) to 7(Dark)
AGP-3301S	0(Bright) to 7(Dark)	0(Bright) to 7(Dark)
AGP-3300L	0(Bright) to 7(Dark)	0(Bright) to 7(Dark)
AGP-3300S	0(Bright) to 7(Dark)	0(Bright) to 7(Dark)
AGP-3300T	0(Bright) to 7(Dark)	-
AGP-3400S	0(Bright) to 7(Dark)	0(Bright) to 7(Dark)
AGP-3400T	0(Bright) to 7(Dark)	-
AGP-3500L	0(Bright) to 7(Dark)	0(Bright) to 7(Dark)
AGP-3500S	0(Bright) to 7(Dark)	0(Bright) to 7(Dark)
AGP-3500T	0(Bright) to 7(Dark)	-
AGP-3600T	0(Bright) to 7(Dark)	-
AGP-3450T	0(Bright) to 7(Dark)	-
AGP-3550T	0(Bright) to 7(Dark)	-
AGP-3650T	0(Bright) to 7(Dark)	-
AGP-3750T	0(Bright) to 7(Dark)	-

10 Sample Program (Serial Communication)

10.1 Sample System

This section provides examples of the Host's program and the GP's parts setup which are necessary for data transmissions between the GP and the Host. Plus, when the parts setup below is run with the sample program, it demonstrates a GP screen change.

Use the following steps to create the screens shown below.

When the [Motor ON], [Motor OFF], [Display], or [Error] switch is pressed, that switch's respective interrupt code is output to the host system, starting the following operations.

Switch Explanation

[Motor ON] .....Starts the motor to supply 50% of the sediment into the sedimentation tank.

[Motor OFF].....Stops the motor.

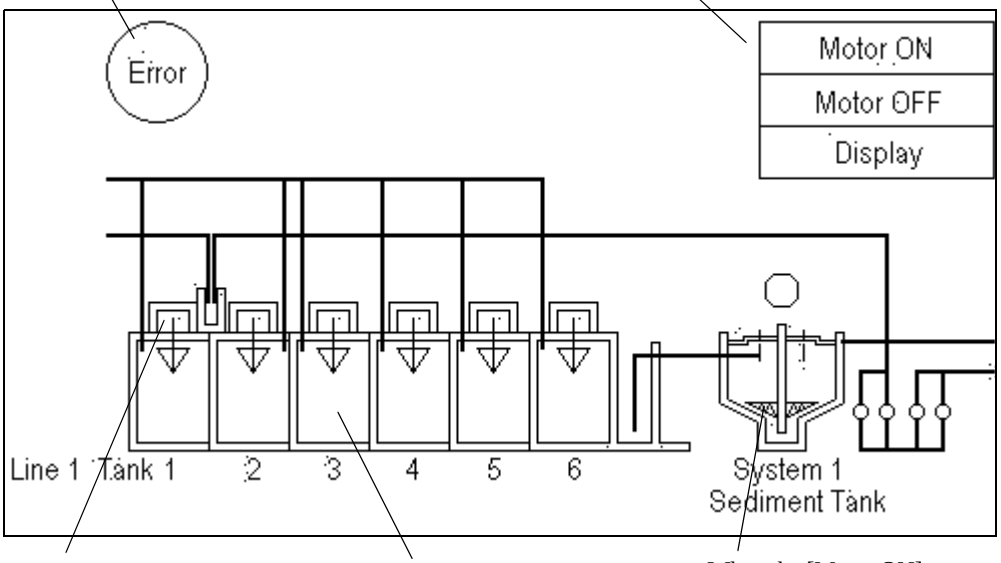
[Display].....50% of the sediment is being supplied to the sedimentation tank.

[Error].....Only 20% of the sediment has been supplied to the sedimentation tank.

■ System Example

This is the Error switch. When pressed, the lamp lights.

These are the motor start switch, motor stop switch and the switch for displaying the level of the sedimentation tank. When pressed, the respective lamp lights.



When the motor ON switch is pressed, the mark is displayed.

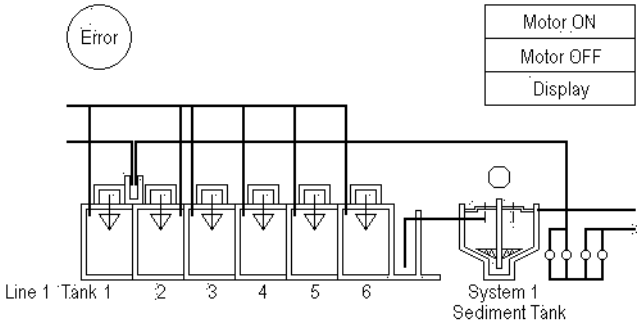
When the program is run, an image of the tank with material inside is displayed.

When the [Motor ON], [Display] and [Error] switches are pressed, the current sediment level is displayed.

■ Screen Creation

(1) Use the GP-Pro EX to create the screens.

This screen is displayed when the GP is operating.



(2) Use the GP-Pro EX to setup Parts

■ Parts Setup Example

Switch List

Screen No.	Parts Name	Switch	Word Address	Word Action	Fixed No.
Base Screen 1	Motor ON	Word Switch	#MEMLINK 13	Write Data 16 bit Dec	0031
	Motor OFF				0032
	Display				0033
	Error				0034

◆ Address Map

Parts shown in the Parts Setup Example are allotted to their corresponding address as follows.

Switch -> Address 13

Writing data to Address 13 (Interrupt) causes an output of the bottom 1 byte code from the RS232C port. For this reason, the Switch (Parts) uses word write.

Motor ON .....word write 0031 to address 13

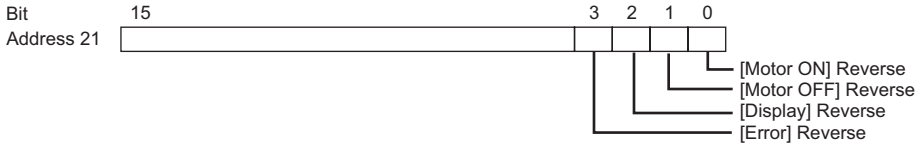
Motor OFF.....word write 0032 to address 13

Display .....word write 0033 to address 13

Error .....word write 0034 to address 13

Tank -> Address 20

Switch -> Address 21



Motor -> Address 22

Sediment Tank -> Address 23

(3) The host unit's company creates the program for data transfer between the GP and the host.

◆ Sample Program

E.g. If an IBM PC/AT-compatible machine and the C language are used:

```
/*****************************************************************************************/
/**/
/*  GP series     Sample program for memory link communications*/
/**/
/*****************************************************************************************/

#include<stdio.h>
#include<dos.h>
#include<string.h>
#include<stdlib.h>
#include<conio.h>

#define  data_size_str220/*The data size of str2 is 20 bytes*/
#define  data_size_wr_data24/*The data size of wr_data is 24 bytes*/

#define  serial_port_BIOS0x14/*PC serial port BIOS*/
#define  serial_port_number0x00/*Serial port number used*/
#define  serial_port_INT0xE7/*The serial port is initialized.*/
#define  serial_port_parameter0xE7/*9600bps,8bit,stopbit;1,parity;none*/

#define  get_status0x03/*The status of the serial port is acquired.*/
#define  serial_port_write0x01/*The serial port is written.*/
#define  serial_port_read0x02/*The serial port is read out.*/

#define  status_bit_60000x60000/*Port status bits 13 and 14*/
#define  status_bit_00200x0020/*Port status bit 5*/

/*****************************************************************************************/
/*                Communications settings for the SI0*/
/*****************************************************************************************/
void  open_SIO (void);/*Communications settings for RS232C*/

/*****************************************************************************************/
/*              Acquisition and identification of port status*/
/*****************************************************************************************/
int   err_status (void);  /*The port status is acquired.*/
void  write_ready (void); /*The transmission buffer register and the transmission register statuses are acquired.*/
int.  read_ready (void);  /*Confirmation of data set status*/
```

```
/*****************************************************************************************/
/*                          Writing data*/
/*****************************************************************************************/
void  write_data (char wr_data);/*The data is written to the registers.*/
void  write (char *wr_data);/*The data is written to the GP.*/

/*****************************************************************************************/
/*                          Reading data*/
/*****************************************************************************************/
int   read_data (void);/*The data is read from the GP.*/
void  change_screen (int interrupt_data);/*The received data in an interruption from the GP is identified.*/
int   read (void);/*The received data in an interruption from the GP is read.*/

/*****************************************************************************************/
/*                      Confirmation of key entries*/
/*****************************************************************************************/
int   kbhit (void);

/*****************************************************************************************/
/*                          Global variables*/
/*****************************************************************************************/
int   interrupt_data,port_status+;
char  *str2;

void main (void)
{
        int no_data;
        str2 =  (char*) malloc (sizeof (char) *data_sezi_str2); /*The memory for str2 is secured.*/
        char *wr_data =  (char*) malloc (sizeof (char) *data_size_wr_data);
                                        /*The memory for wr_data is secured.*/
        open_SIO ();          /*Communication settings for RS232C*/
        wr_data = "\x1bW000F0001\x0d\0" ; /*0x1 is written to address 15: screen number 1 setup*/
        write (wr_data);
        wr_data = "\x1bW0014003F\x0d\0";
              /*0x3F is written to address 20: Materials are put into aeration tanks Nos. 1 to 6.*/
        write (wr_data);
/*************************************************************************/
/*      The data reception from the GP is identified.  */
/*  If the Write key is pressed, the execution is completed.*/
/*************************************************************************/
       while (1)
       {
              no_data = read ();
```

```
            if (no_data == 1)      /*If there is any key entry, no_data=1.*/
            {
                break;
            }
            else
            {
                wr_data = str2;
                write (wr_data);
            }
        }
        getch ();                   /*The codes for keys are removed from the key buffer.*/
        free (wr_data);                 /*The memory area for wr_data is freed up.*/
        free (str2);                /*The memory area for str2 is freed up.*/
}

/*The transmission buffer register status and the transmission register status are acquired.*/
void write_ready (void)
{
        int err6000;

        err6000 = 0;
        while (status_bit_6000 != err6000)
        {
            err6000 = err_status () & status_bit_6000;
        }
        return;
}

/*Confirmation of data set status*/
int read_ready (void)
{
        int no_data,err0020;

        err0020 = 0;
        while (status_bit_0020 != err0020)
        }
            err0020 = 344_status () & status bit_0020;
            if (kbhit ())       /*Confirms whether there is a key entry or not.*/
            {
                no_data = 1;/*If there is a key entry, no_data=1.*/
                break;      /*The program is terminated.*/
            }
```

```
    }
     return (no_data);
}
/*Data is written to the GP.*/

void write (char *wr_data)
{
     while (*wr_data != '\0')       /*The data is written until it becomes NULL.*/
     {
          write_ready ();
          write_data (*wr_data);
          wr_data++;        /*The address pointed to by the pointer is incremented.*/
     }
     return;
}
/*****************************************************************************/
/*          The interrupt data received from the GP is confirmed.        */
/*          The data is written to addresses 20, 21, 22, and 23.         */
/*****************************************************************************/
void change_screen (int interrupt_data)
{
      switch (interrupt_data)
/*If interrupt_data is 1, 0x1 is written to address 21, 0x3F to address 22, and 0x50 to address 23.*/
               case   1: str2 = "\x1bW00150001003F0050\x0d\0";
                        break;

/*If interrupt_data is 2, 0x2 is written to address 21, 0x0 to address 22, and 0x0 to address 23.*/
               case   2: str2 = "\x1bW0015000200000000\x9d\0";
                        break;

/*If interrupt_data is 3, 0x4 is written to address 21, 0x0 to address 22, and 0x50 to address 23.*/
               case   3: str2 = "\x1bW00150000400000050\x0d\0";
                        break;

/*If interrupt_data is 4, 0x8 is written to address 21, 0x0 to address 22, and 0x20 to address 23.*/
               case   4; str2 = "\x1bW0015000800000020\x0d\0";
                        break;

/*If interrupt_data is other than 1 to 4, NULL is written.*/
              default  : str2 = "\0";
                       break;
       {
       return;
```

```
}

/*****************************************************************************/
/*          The interrupt data received from the GP is read.              */
/*  Reading is performed until the interrupt_data becomes other than NULL.  */
/*****************************************************************************/
int read (void)
{
        int no_data;
        do
        {
                no_data = read_ready (); /*Confirmation of data set status*/
                if (no_data == 1)       /*If there is a key entry, no_data=1.*/
                {
                    break;
                }
                else
                {

                    read_data ();      /*The data received from the GP is read out./*
                    change_screen (interrupt_data); /*The data received from the GP is identified.*/
                }
         } while (*str2 == '\0');
         return (no_data);
}
/*Communications settings for RS232C*/

void open_SIO (void)
{
        union REGS regs ;
                regs.x.dx = serial_port_number;
                regs.h.ah = serial_port_INT;
                regs.h.al = serial_port_parameter;
                int86 (serial_port_BIOS,&regs,&regs);
        return;
}

/*The port status is acquired.*/
int err_status  (void)
{
        union REGS regs;
                regs.x.dx = serial_port_number;
                regs.h.ah = get_status;
```

```
            int86 (serial_port_BIOS,&regs,&regs);
            port_status = regs.x.ax;
        return (port_status);
}

/*The data is written to the registers*/
void write_data (char wr_data)
{
        union REGS regs;
            regs.x.dx = serial_port number;
            regs.h.ah = serial_port_write;
            regs.h.al = wr_data;
            int86 (serial_port_BIOS,&regs,&regs);
        return;
}

/*The data is read from the GP*/
int read_data (void)
{
        union REGS regs;
            regs.x.dx = serial_port_number;
            regs.h.ah = serial_port_read;
            int86 (serial_port_BIOS,&regs,&regs);
            interrupt_data = regs.h.al;
        return (interrupt_data);
}
```

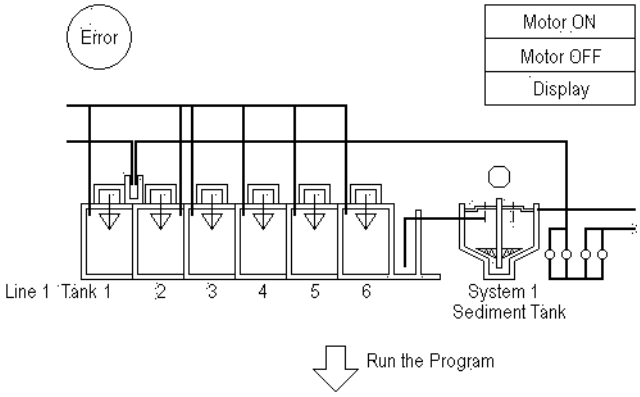
NOTE

- The availability of open_SI0 (void), err_status (void), write_data (char wr_data), and read_data (void) will depend on the models used. If the program is written on a personal computer that is not IBM -compatible, it must be modified in order to be used.

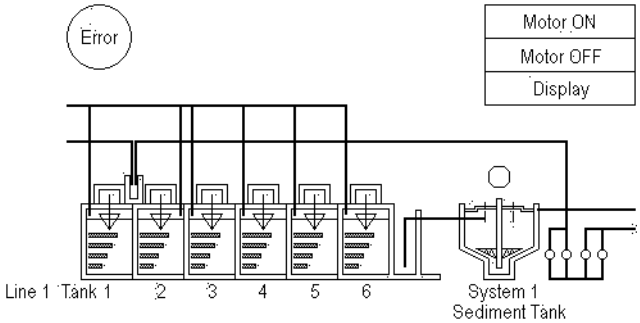
(4) After screen data is transferred to the GP, display (operation) can begin.

◆ GP Run Screen

GP Screen (Before running program)



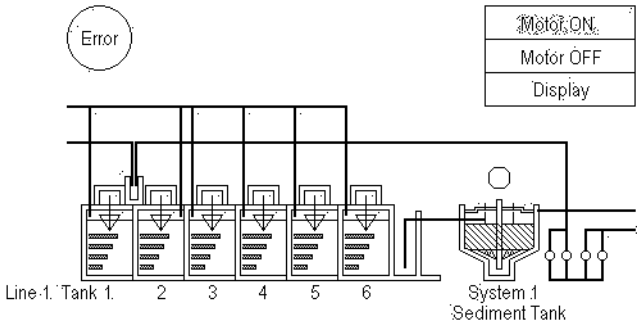
GP Screen (After running program)



Six Libraries  appear.

Press the [Motor ON] switch

ASCII Code "31" = Data "1" is output to the Host, causing the screen to change.



10.2 Troubleshooting Multiple GP (Multi-drop) Communication

The host plays the following two roles when controlling multiple GP units:

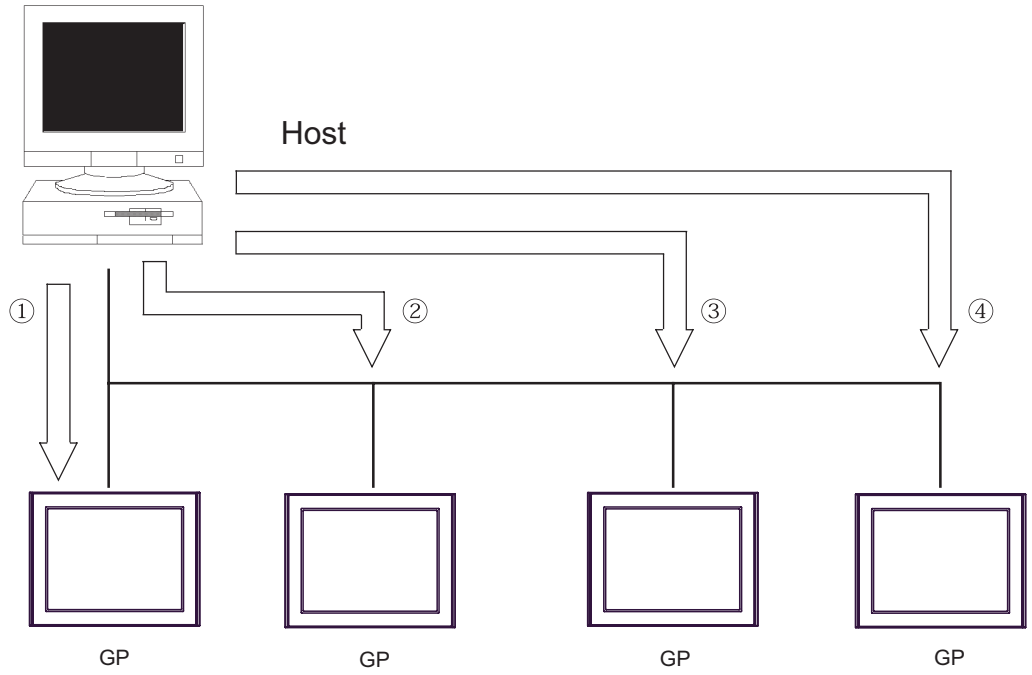
1. transferring data to be displayed
2. reading touch panel inputs from GP units through polling

Note that the more GP units to be controlled, and the more data to be transferred, the more burdened the host will be. In addition, an excessive number of GP units or an excessive amount of data can degrade the response speed of the GP units (slower display switching and slower response to touch panel inputs), substantially affecting the system operation. Therefore, you should consider the number of GP units and amount of data when designing a multi-drop system.

■ Sending Display Data to All GP units at the Same Time

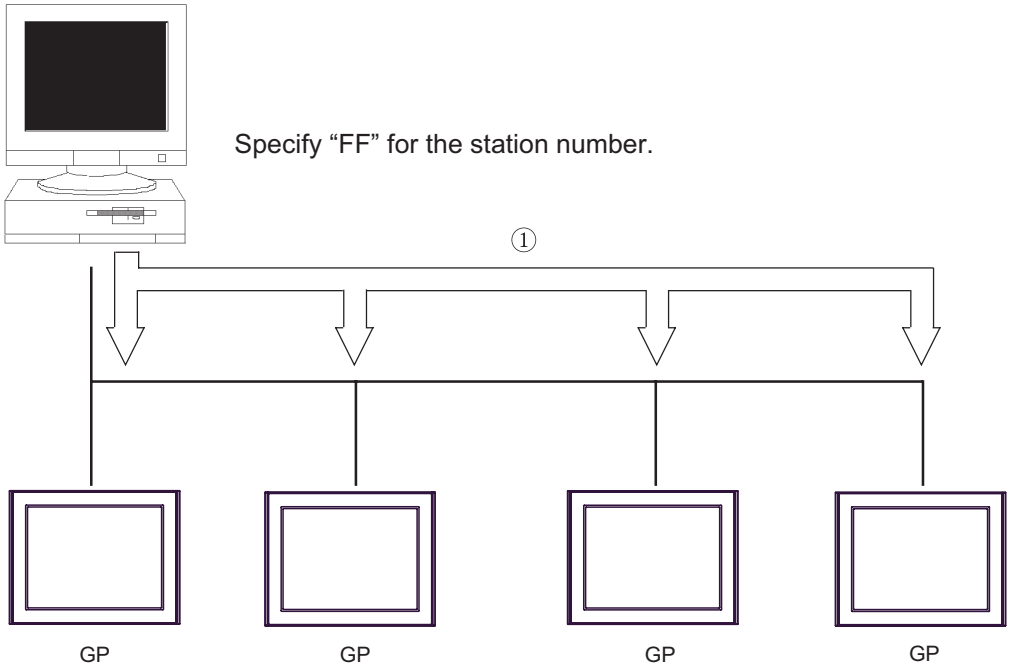
When you need to send the same data to all GP units, try sending it to all GP units at the same time for improved efficiency, instead of sending it to one GP unit at a time.
(This can be accomplished by specifying "FF" for the station number.)

◆ When sending data to one GP at a time



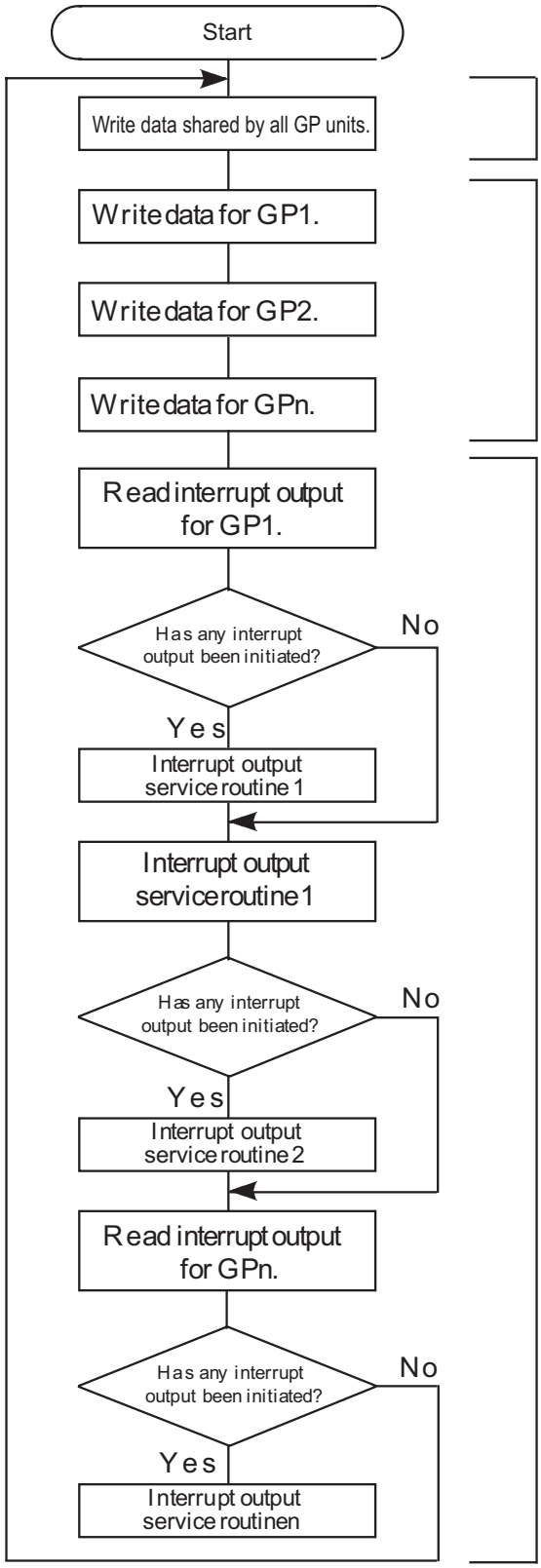
Excessive amount of time required (4 times longer than the case shown below)

◆ When sending data to all GP units at the same time



Substantially reduced communication time (4 times shorter than the case shown above)

10.3 Program Flowchart for Multi-drop System



(1) Writing data shared by all GP units
Use the ESC W command to write display data shared by all GP units to the system area. (At this time, specify "FF" for station number.)
(2) Writing data for a specific GP
Use the ESC W command to write data for a specific GP to the system area.

(3) Polling
Use the ESC I command to poll each GP unit to determine whether any touch panel input has been made. Touch panel inputs are serviced accordingly.

NOTE

- To improve GP response speed for touch panel input, insert a polling sequence after each write sequence (sequence in which data is written for a specific GP unit).
- Make sure that the amount of data to be written to the system area is minimal. In order to accomplish this, youcan, for example, choose to update only data items that have been changed.

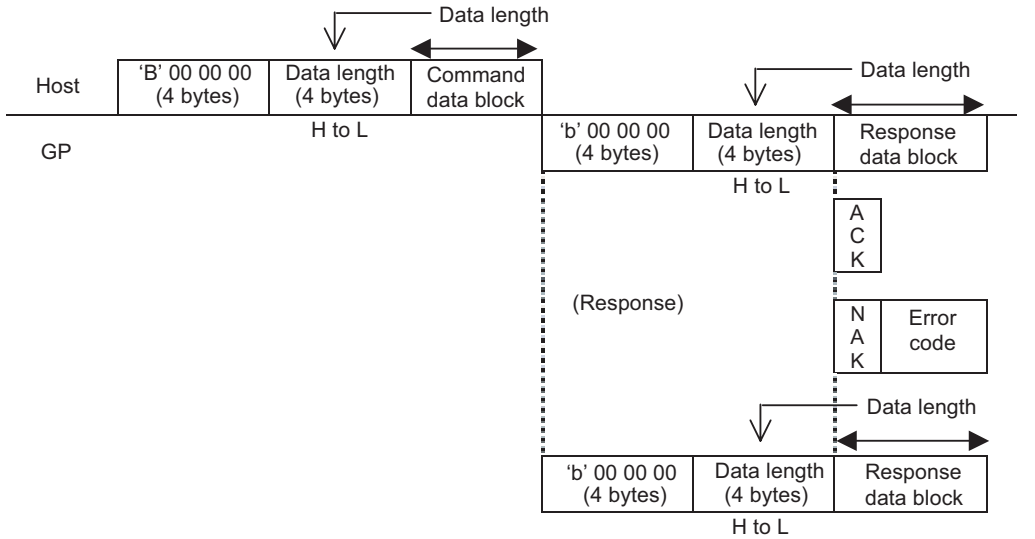
11 Memory Link Command (Ethernet Communication)

11.1 Basic Communication Protocol Control

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

11.1.1 LAN

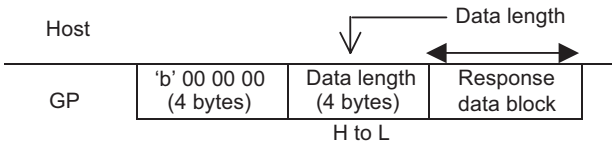
■ Host to GP Data Transfer



(When the response frame is followed by another frame)

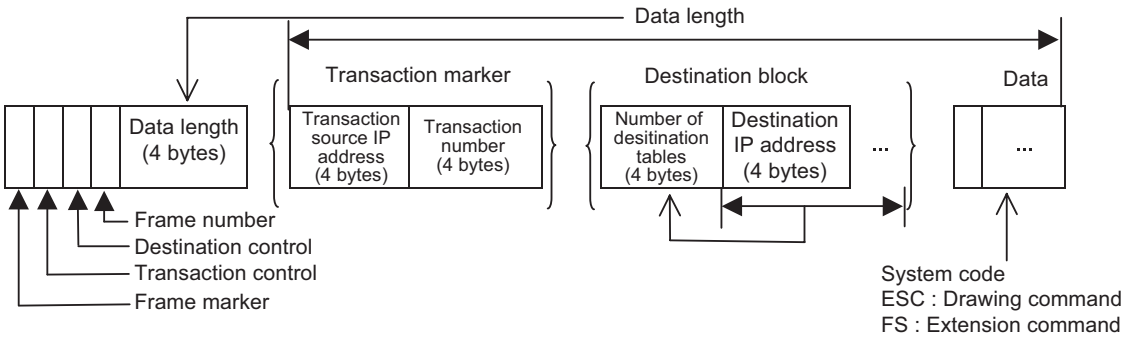
- Command Data area stores the data to be transmitted from the host device to the GP.
- After the GP analyzes the Command Data, Response Data area stores the result of "ACK" or "NAK", or no response.

■ GP to Host Data Transfer



■ Details on Frame Format

The memory link LAN frame is structured as follows:

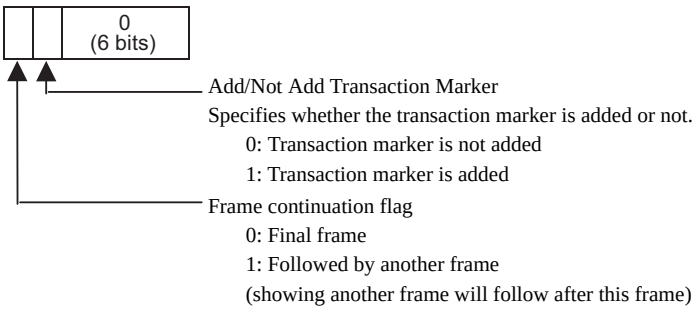


The initial 8 bytes, from frame marker to data length, are provided in all memory link LAN frames.
Therefore, during a frame check, the system checks the initial 8 bytes first, and then checks the subsequent data based on the data length specified in the initial 8 bytes.

◆ Frame Marker (1 byte)

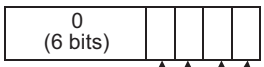
The frame marker is used to identify the frame type.
'B': Binary command frame
'b': Binary response frame
Only binary frames are supported.

◆ Transaction Control (1 byte)



During the transmission of large amounts of data, the data will be divided into several frames for sending and receiving. 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.

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

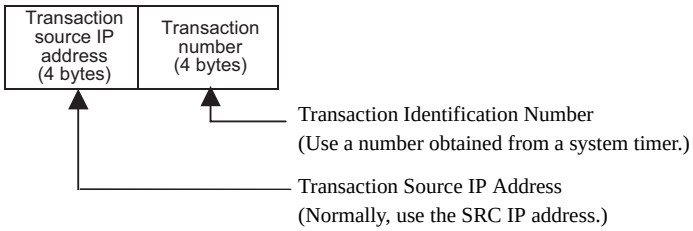
The GP sends back a response after waiting for time duration of the "least-significant 7 bits of the SRC IP address x 1 ms". This function prevents several nodes from responding simultaneously. To request a response from all the nodes, enter "09h".

◆ 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.

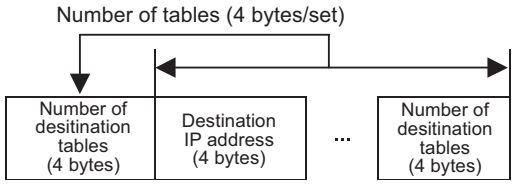
◆ Frame Number

When a command or response is divided into several frames, serial numbers (starting from 0) are assigned as the frame numbers. The maximum number is 255.

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".



11.2 Demand Polling

In TCP connection, 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 presence of the Host.

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

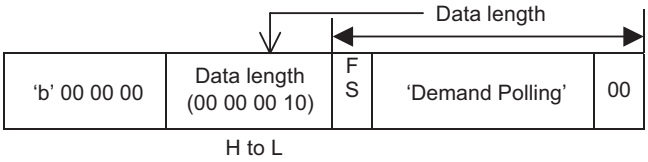
If you wish to use Digital Electronics Corporation APIs for the Host, the response processing of the Demand Polling request will be performed automatically by the API.

11.2.1 Demand Polling (FS Demand)

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

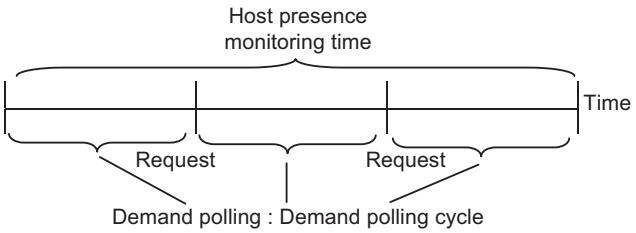
Host: Nothing

GP: Response Data



Data Name

- Data: "Demand Polling"



Demand Polling: When the Host Presence Monitoring Time elapses, a Demand Polling request is sent.

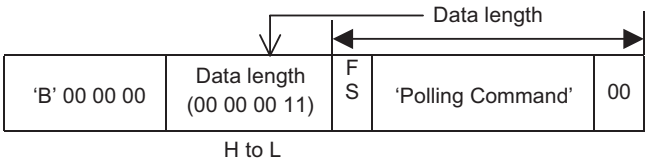
This request demands the Host send its own Polling command. This type of request allows the GP to not have to wait for a polling request.

Host					
GP	'b' 00 00 00	00 00 00 10	F S	'Demand Polling'	00

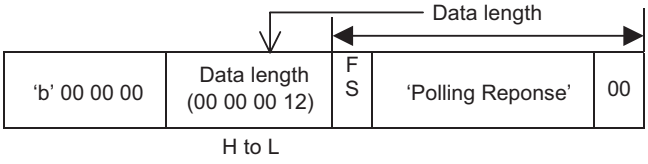
11.2.2 Polling Command (FS Polling)

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:

Host	'B' 00 00 00	00000011	F S	'Polling Command'	00
GP	'b' 00 00 00	00000012	F S	'Polling Response'	00

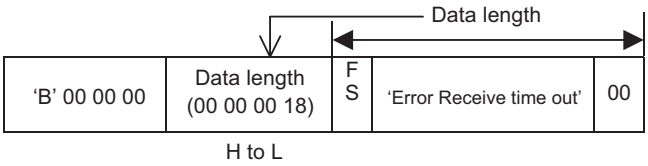
11.2.3 Error Detection (FS Error)

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.

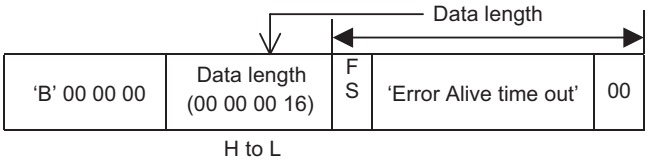
Host: Command Data

- Inter-character timeout error frame



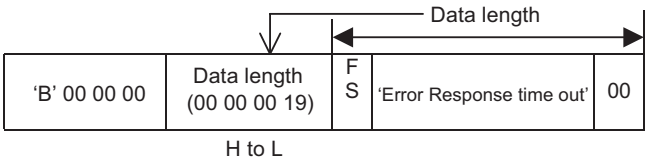
Data Name

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



Data Name

- Data: "Error Alive time out"
- Inter-protocol timeout error frame

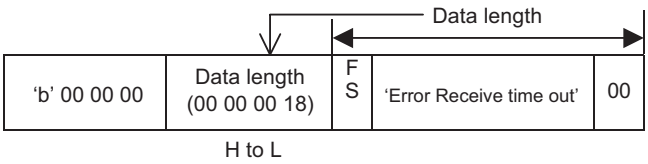


Data Name

- Data: "Error Response time out"

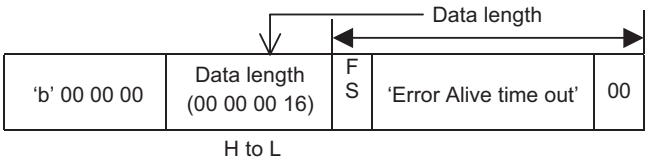
GP: Response Data

- Inter-character timeout error frame



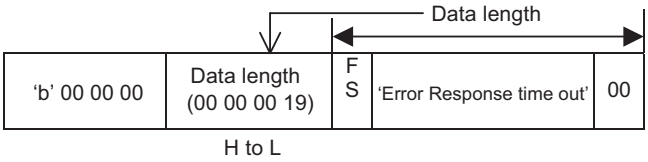
Data Name

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



Data Name

- Data: "Error Alive time out"
- Inter-protocol timeout error frame



Data Name

- Data: "Error Receive time out"

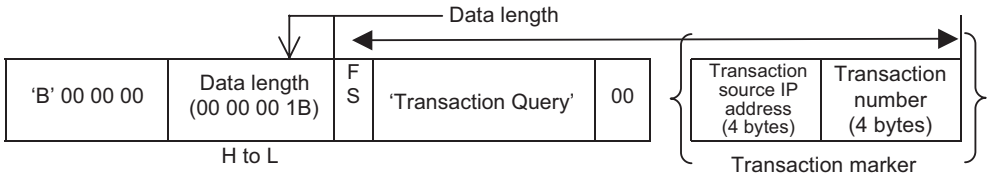
11.3 Transaction Result Request Command

This command is enabled only when using the Ethernet.

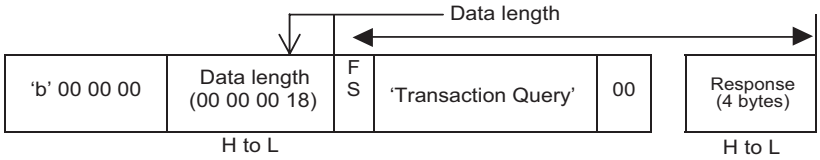
11.3.1 Transaction Result Request

The contents of the transaction result request sent from the host device to the GP are as follows:

Host: Command Data



GP: Response Data



Data Name

- Data: 'Transaction Query'

Result Executed

- Data: 'Transaction Result'

Value Meaning

0x00000000 Processing ends properly.

0x00000001 Error

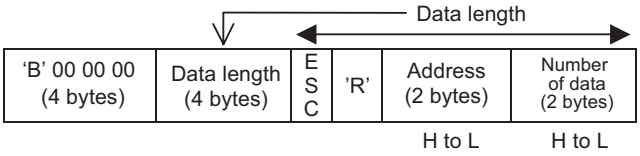
0x00000002 Designated transaction marker is not stored in the GP.

11.4 Command Format

11.4.1 Read Format

■ LAN

◆ Command data block (from Host)



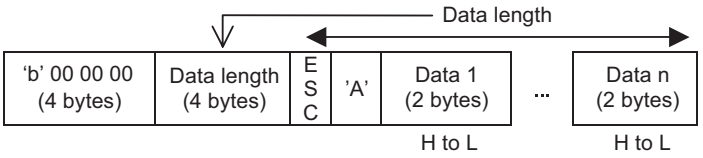
<Setting range>

Address: 0000H to 270FH (0 to 9999)

Number of data packets: 0001H to 0200H (1 to 512)

◆ GP Response data block (from GP)

- When there is no error



<Setting range>

Data: 0000H to FFFFH

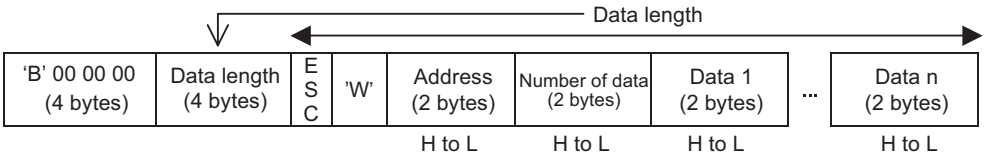
- If an error occurs

NAK response

11.4.2 Write Format

■ LAN

◆ Command data block (from Host)



<Setting range>

Address: 0000H to 270FH (0 to 9999)

Number of data packets: 0001H to 0200H (1 to 512)

Data: 0000H to FFFFH

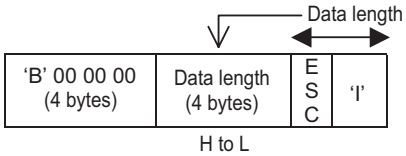
◆ GP Response data block (from GP)

ACK or NAK response

11.4.3 Interrupt Output Requests

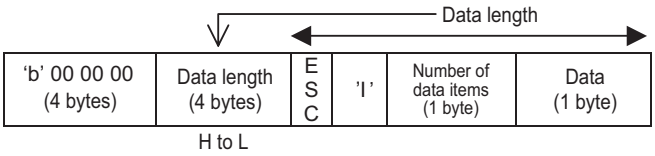
■ LAN

◆ Command data block (from Host)



◆ GP Response data block (from GP)

- When there is no error



- If an error occurs

  NAK response

<Setting range>

Number of data packets

When an enquiry command is sent from the host, this value defines the previously issued interrupt output's number of data items.

Data

The data value (00H to FEH) is output.

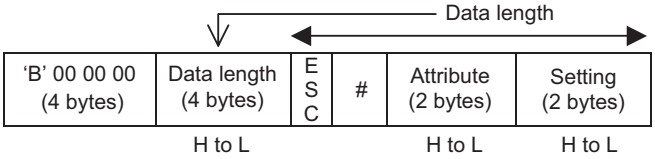
"00" will be entered in this field if there is no data to be output.

11.4.4 Brightness and Contrast Adjustments

The format of the command data block containing the ESC # command (brightness and contrast adjustment command) is shown below. Note that brightness or contrast cannot be adjusted with some GP types.

■ LAN

◆ Command data block (from Host)



<Setting range>

Attribute: 0000H to 0001H (0: Contrast, 1: Brightness)

Settings: Please refer to "■ Brightness/Contrast Table" (page 45).

Be sure to make all data entries in ASCII code format.

◆ GP Response data block (from GP)

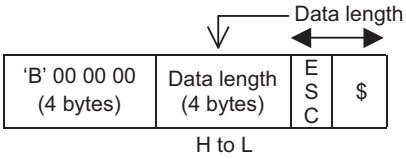
ACK or NAK response

11.4.5 Brightness and Contrast Current Value

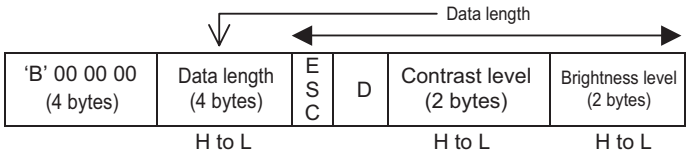
The format of the command data block to acquire the brightness and contrast current values with the command is shown below. Note that the brightness or contrast level is not available with some GP types.

■ LAN

◆ Command data block (from Host)



◆ GP Response data block (from GP)



■ Brightness/Contrast Table

GP	Brightness Setting Range	Contrast Setting Range
AGP-3302B	0(Bright) to 7(Dark)	0(Bright) to 7(Dark)
AGP-3301L	0(Bright) to 7(Dark)	0(Bright) to 7(Dark)
AGP-3301S	0(Bright) to 7(Dark)	0(Bright) to 7(Dark)
AGP-3300L	0(Bright) to 7(Dark)	0(Bright) to 7(Dark)
AGP-3300S	0(Bright) to 7(Dark)	0(Bright) to 7(Dark)
AGP-3300T	0(Bright) to 7(Dark)	-
AGP-3400S	0(Bright) to 7(Dark)	0(Bright) to 7(Dark)
AGP-3400T	0(Bright) to 7(Dark)	-
AGP-3500L	0(Bright) to 7(Dark)	0(Bright) to 7(Dark)
AGP-3500S	0(Bright) to 7(Dark)	0(Bright) to 7(Dark)
AGP-3500T	0(Bright) to 7(Dark)	-
AGP-3600T	0(Bright) to 7(Dark)	-
AGP-3450T	0(Bright) to 7(Dark)	-
AGP-3550T	0(Bright) to 7(Dark)	-
AGP-3650T	0(Bright) to 7(Dark)	-
AGP-3750T	0(Bright) to 7(Dark)	-

12 Memory Link API (Ethernet Communication)

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.

12.1 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 6.0

OS : Microsoft Windows 98 or higher

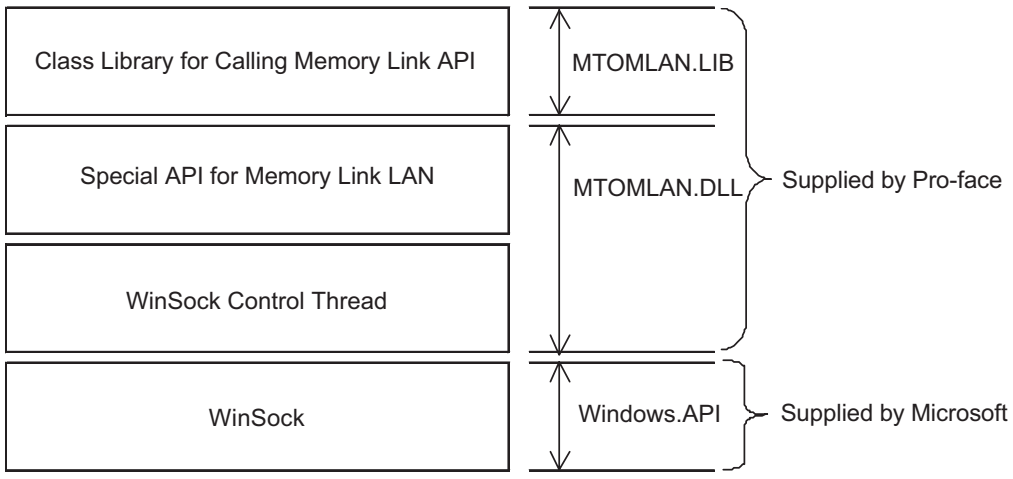
Others : The following files are contained on the GP-Pro EX CD-ROM.
To view them, open the CD-ROM's [MTOMLAN] folder and double-click on the [MTOMLAN.ZIP] file.

MTOMAPI.H

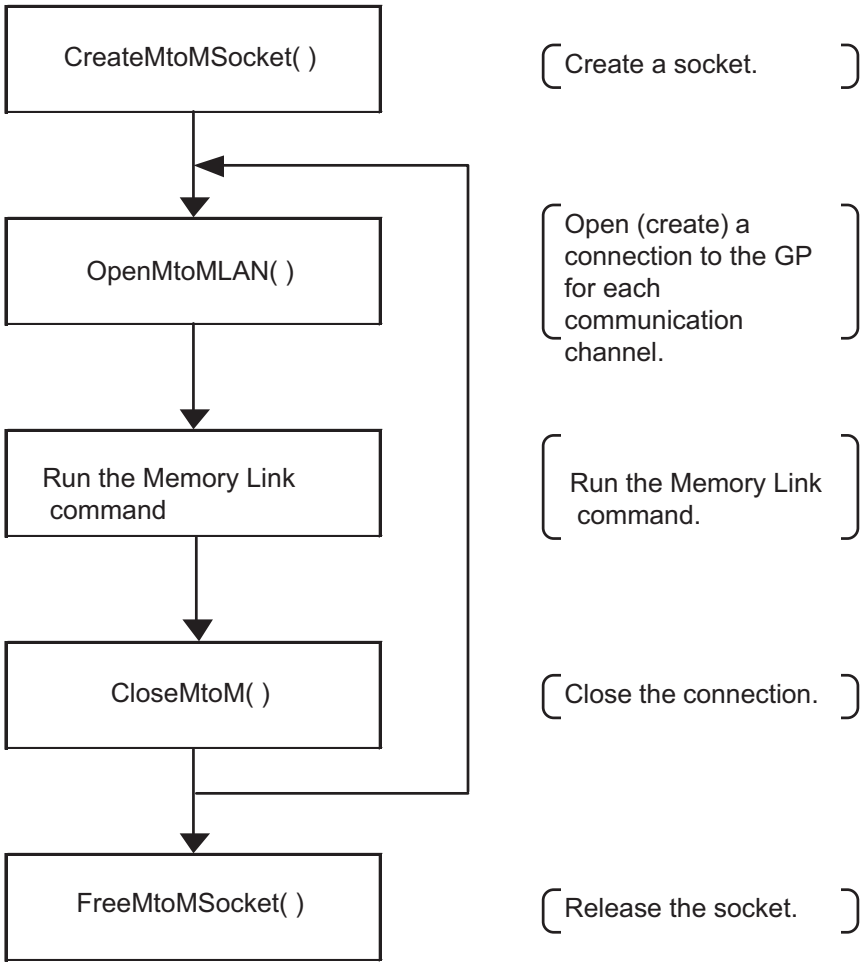
MTOMLAN.LIB

MTOMLAN.DLL

■ Memory Link API Software Structure Diagram



■ General Operation of Memory Link API



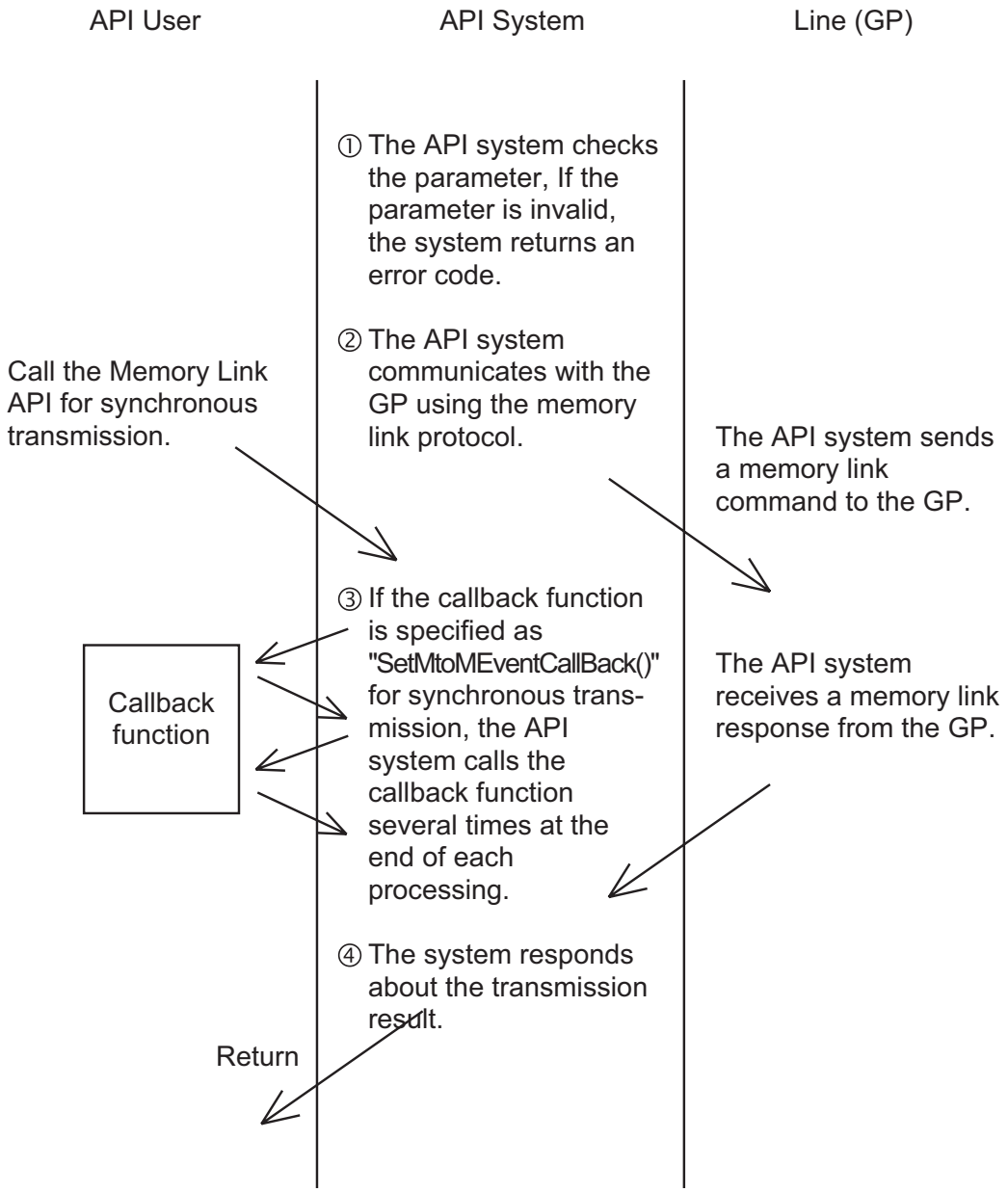
12.1.1 Synchronous and Asynchronous Transmision

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.

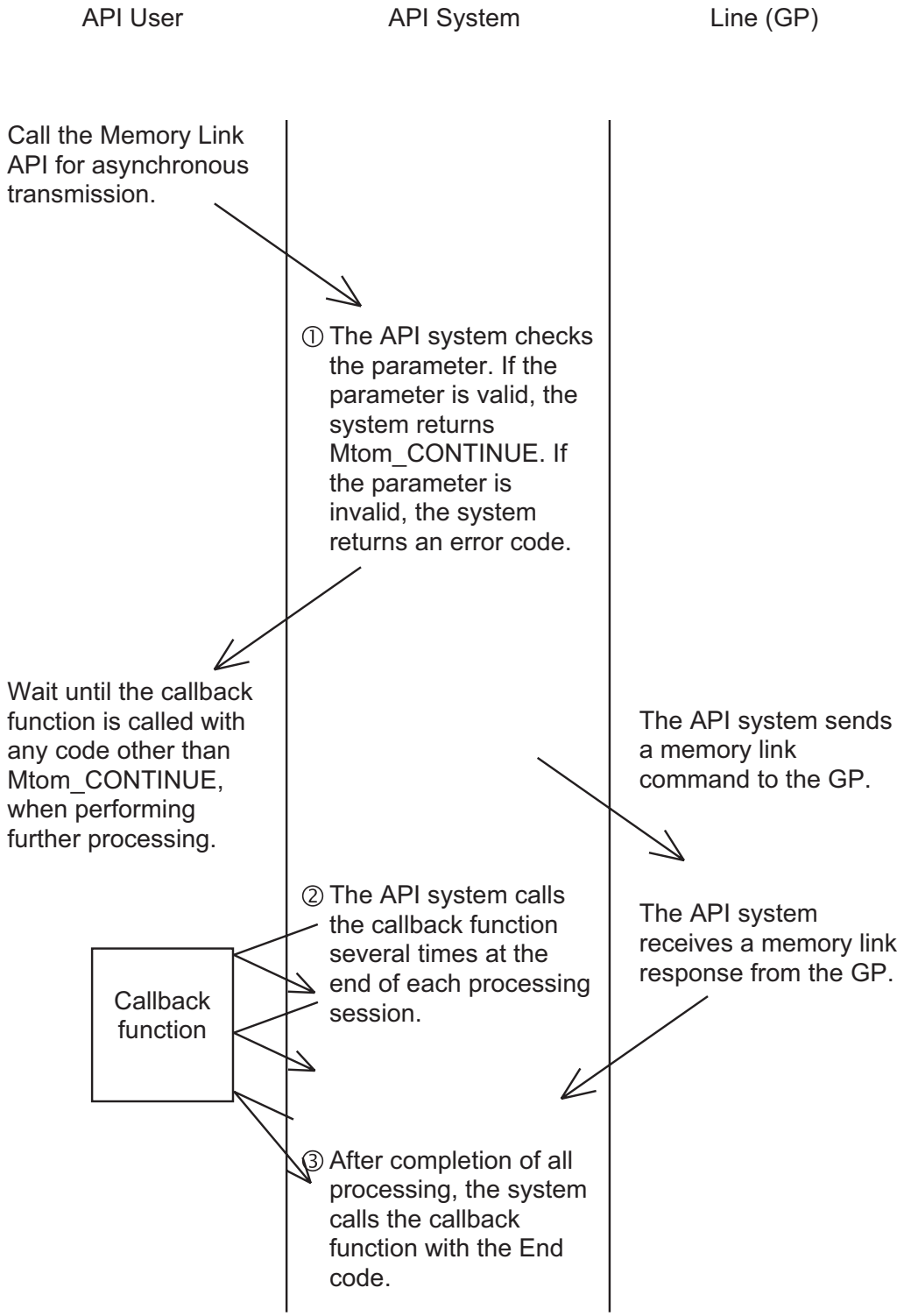
NOTE

- 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 pMSockSocket 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.

NOTE

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

12.1.2 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)
  }
 }
```

12.1.3 Transmission Methods (Transaction Types)

This Ethernet protocol supports the follwing 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 displaying 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_Specifict) 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

NOTE

- 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_Specifict) 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.

NOTE

- 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 (dwCheckButtom) 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.

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

(8) If you attempt to retry after checking the dwNodeStatus value, enter TRUE in the dwCheckButtom 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.

12.2 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.
OpenMtoMLAN	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.
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.

12.2.1 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

Other :Handle of the created socket

NULL :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

12.2.2 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 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.
LPCSTR szIPAddress	IP address of destination node (GP) For 1:n communication, specify NULL.	

NOTE

- The IP address can be specified with the following two methods:

  Separating an IP address with dots:

  Example) szipaddress="11.22.33.44";

  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 folder.

  Example) Contents of C:\Windows \ HOSTSC

  11.22.33.44 GP1

12.2.3 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.

12.2.4 Freeing a Socket

To free a socket, use the following command:

■ Syntax

int WINAPI FreeMtoMSoket(LPMtoMSOCK pMSock)

■ Return Value

Other : Handle of the created socket

NULL : Failed to create a socket

■ Argument

LPMtoMSOCK pMSock Handle of memory link socket

12.2.5 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

12.2.6 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

NOTE

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

12.2.7 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

LPMtoMSOCK pMSock	Handle of memory link socket
DWORD dwNodeCounter	Desired number of node records

12.2.8 Transmission Method Setup (Transaction Type)

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

■ Syntax

DWORD WINAPI SetTransctionType(LPMtoMSOCK pMSock,DWORD dwTranscitonType)

■ Return Value

Setting of the transaction type yet to be changed

■ Argument

LPMtoMSOCK pMSock Handle of memory link socket

DWORD dwTranscitonType 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 displaying 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.)

12.2.9 Acquiring the Current Transmission Method

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

■ Syntax

DWORD WINAPI GetTranscitonType(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 pMSockHandle of memory link socket

12.2.10 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 pMSockHandle 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.

◆ GP-related error codes

Cf. GP-Pro EX Reference Manual

◆ 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

Code	Error	Code	Error
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	-	-

12.3 Display Mode Commands

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

■ Displaying Command List

Command	Action
MtoMESC_W	Writes data into the System Area.
MtoMESC_R	Reads data from the System Area.
MtoMESC_I	Inquires if the touch panel has been pressed.
MtoMESC_SetContrast	Sets brightness/contrast.
MtoMESC_GetContrast	Acquires the brightness/contrast setting.

12.3.1 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* pwData)

■ 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	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 write data. 0000h to 0FFFh
INT iDataCount	Specifies the number of data packets to be written. 0001h to 0200h(1 to 512)
WORD* pwData	Data to be written

12.3.2 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 processingresult 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 to 0FFFh
INT iDataCount		Specifies the number of data packets to be read. 0001h to 0200h(1 to 512)
WORD pwoData		Location to store read data.

NOTE

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

12.3.3 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	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.
BOOL* pbHave	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.	
DWORD *pdwCode	When the touch panel has been pressed, the corresponding code is entered in this area.	

12.3.4 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	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.
dwContrast	Contrast adjustment (0000h to 0007h) 0: Bright to 7: Dark "FFFFFFFFh" indicates "No setting" (this model disables construct adjustment).	
dwLight	Brightness adjustment (0000h to 0007h) 0: Bright to 7: Dark "FFFFFFFFh" indicates "No setting" (this model disables brightness adjustment).	

NOTE

- For details about the range of Brightness/Contrast, refer to the Brightness/Contrast Table.
☞ " ■ Brightness/Contrast Table" (page 90)

12.3.5 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 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.
dwContrast	Current value of contrast (0000h to 0007h) 0: Bright to 7: Dark "FFFFFFFFh" indicates "No setting" (contrast cannot be adjusted with this model).	
dwLight	Current value of brightness (0000h to 0007h) 0: Bright to 7: Dark "FFFFFFFFh" indicates "No setting" (brightness cannot be adjusted with this model).	

NOTE

- For details about the range of Brightness/Contrast, refer to the Brightness/Contrast Table.
☞ "■ Brightness/Contrast Table" (page 90)

12.3.6 API Return Value Error Code List

	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.

NOTE

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

13 Sample Program (Ethernet Communication)

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

13.1 Memory Link LAN API Sample Program

■ Start-up Environments

(1) The AGPM.EXE program runs on the Windows 98 operating system.

(2) Since the AGPM.EXE program uses the MtoMLAN.DLL file, copy the MtoMLAN.DLL file into a Windows folder.

(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 AGPM.EXE program and the program cannot be started, the TCP/IP settings may be incorrect. Check the TCP/IP settings.

■ Development Environment

The AGPM.EXE program has been developed for use in the following environments:
The sample program's source code is contained on the GP-Pro EX CD-ROM.
When the source code found in the CD-ROM's [MTOMLAN] folder is compiled in the following environment, the file [AGPM.EXE] is created.
Compiler : Microsoft Visual C++ Ver 6.0
OS : Microsoft Windows 98

■ How to Access Memory Link API

To access the Memory Link API, the AGPM.EXE program defines and uses Class CMSock. Class CMSock fully includes the Memory Link API as "One Socket - One Object".
The AGPM.EXE program provides a callback from the Memory Link API by overriding the method of Class CMSock.

■ Derivation of CMSock

The AGPM.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.

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

■ Class CGpMDOC

This class is the core of the AGPM.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.

■ MtoMAPI.H and MtoMLAN.LI

The AGPM.EXE program includes the MtoMAPI.H file in the external device.
The MtoMAPI.H file is stored in the [MtoMLAN] folder. Copy this file into an appropriate folder, and specify the location by changing the #include statement of defsfile.h.

The AGPM.EXE program includes the MtoMLAN.LIB file to call the MtoMLAN.DLL program. Copy this file into an appropriate folder, and specify the location by selecting [Setup] - [Linker] - [Object/Library Module] .