

# 20 | Programming Scripts

**(Programming that does not use Parts)**

This chapter explains the basics of “Programming Using Scripts” in GP-Pro EX and the methods of creating a script.

Please start by reading “20.1 Settings Menu” (page 20-2) and then turn to the corresponding page.

20.1	Settings Menu .....	20-2
20.2	Conditional Operations.....	20-5
20.3	Copying Data in Blocks .....	20-12
20.4	Displaying an Alarm when an Error Occurs .....	20-17
20.5	Communicating with Peripheral Devices not Supported by Regular Scripts ..	20-21
20.6	Procedure for Creating Scripts.....	20-38
20.7	Trigger Condition Setup .....	20-42
20.8	Settings Guide.....	20-48
20.9	Restrictions .....	20-53

## 20.1 Settings Menu

D-Scripts are a simple language users can program for themselves. Using this feature, you can perform operations in the GP or communicate between the GP and unsupported peripheral devices.

---


### WARNING

---

Be sure to not use D-Scripts/Global D-Scripts to control systems that can cause life-threatening or seriously damaging actions.

---

**NOTE**

- D-Scripts are set up on a Base Screen. That Base Screen looks at the conditions while it is displayed and executes the script.
  - Global D-Scripts, when the GP is running, regardless of the displayed screen, a program executes based on the condition (trigger).
  - Extended Scripts are used for high-level communication programs.
  - In addition to scripts, you can use logic programs to control systems.
-  “28.1 Settings Menu” (page 28-2)
-

### Conditional Operations

Create a script which automatically changes screens to screen number 7 after 3 seconds.

Time →

Process Script →

After 1 second      After 2 seconds      After 3 seconds

D100=1      D100=2      D100=3

D100 is not 3 so portion after "if" does not execute.      D100 is not 3 so portion after "if" does not execute.      D100 = 3 so condition is true and [w:LS0008]=7 executes.

☞ Setup Procedure (page 20-6)  
☞ Details (page 20-5)

### Copying Data in Blocks

Create a script which detects the rising edge (0 to 1) of bit address M0100 and copies data stored in the connected device into another address.

D0099      C

⋮

B

D0000      A

D0200      C

⋮

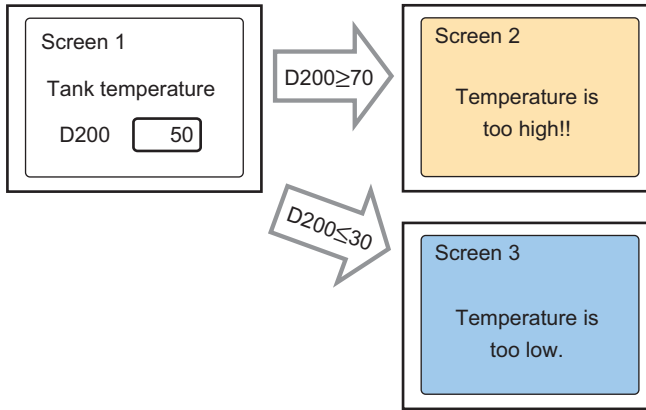
B

D0101      A

☞ Setup Procedure (page 20-13)  
☞ Details (page 20-12)

**Displaying an Alarm when an Error Occurs**

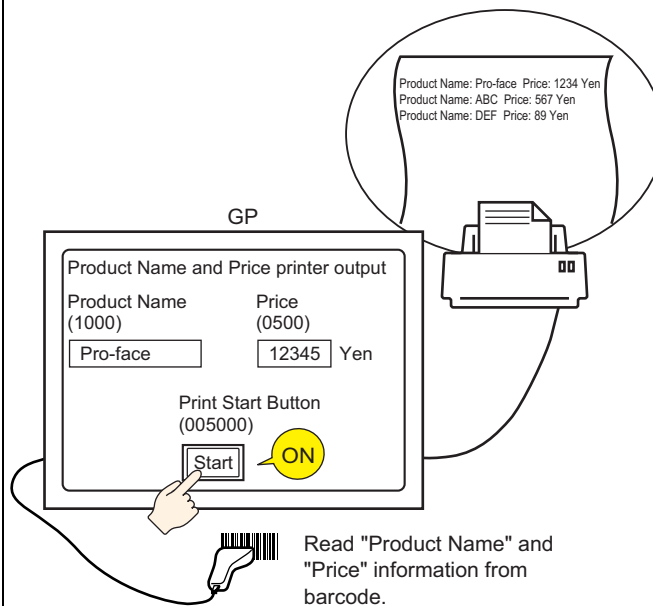
The temperature management system detects an error bit from the connected device and displays alarm messages when the temperature information storage address (D200) rises to 70 °C or higher, or falls to 30 °C or lower. Also, this script will count the number of detected errors.



- ☞ Setup Procedure (page 20-18)
- ☞ Details (page 20-17)

**Communicating with Peripheral Devices not Supported by Regular Scripts**

Create an extended script to output data read from a barcode connected to the USB to a serial printer connected to COM1.



- ☞ Setup Procedure (page 20-34)
- ☞ Details (page 20-21)

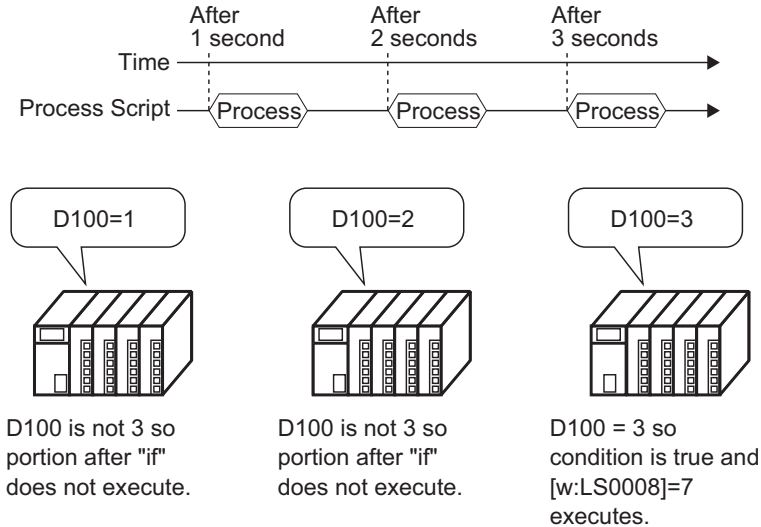
## 20.2 Conditional Operations

**NOTE**

- Please refer to the settings guide for details.  
 ☞ “20.8.1 Common Settings Guide for D-Script” (page 20-48)

**Action**

Create a script which automatically changes screens to screen number 7 after 3 seconds.

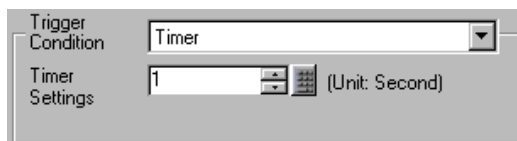


**Commands used**

Command	Function Summary
Assignment (=)	Assigns the right side value to the left side.
Addition (+)	Adds a constant to a word device’s data.
if ( )	When a condition enclosed with brackets “( )” following “if” becomes true, the process following the “if ( )” statement is executed.
Equivalent (==)	Compares the value on the right and left sides. Becomes true if the left side equals the right side.
LS0008	Changes to the screen number stored in this value. ☞ “A.1.4.2 System Data Area” (page A-10)

**Trigger Condition**

Select the timer as below and set the [Timer Settings] to 1 second.



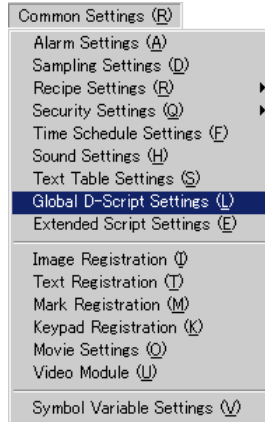
### Completed script

```

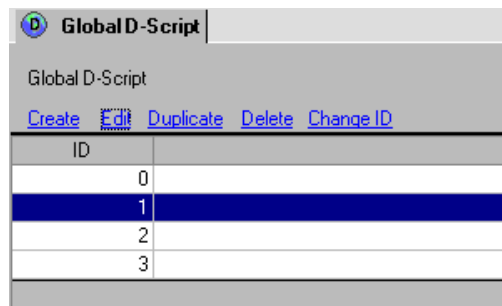
Execution Expression  Enlarge Execution Expression  Address Input
0001 [w:[PLC1]D00100]=[w:[PLC1]D00100]+1
0002 if ([w:[PLC1]D00100]==3)
0003 {
0004   [w:[#INTERNAL]LS0008]=7
0005 }
0006 endif
0007
0008
0009
    
```

### Creation procedure

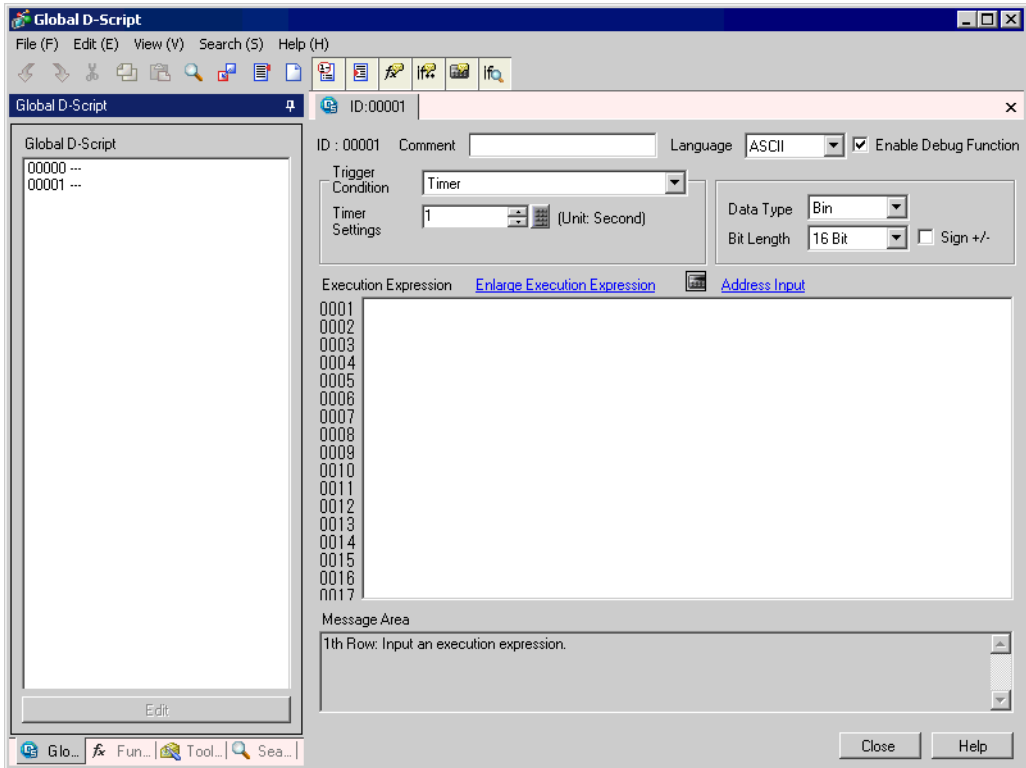
- 1 Select the [Common Settings (R)] menu - [Global D-Script Settings (L)] command.



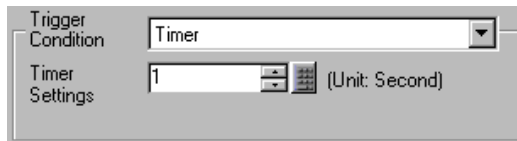
- 2 Click [Create]. If accessing a previously registered script, designate the ID number and click [Edit], or double-click the ID No.'s row.



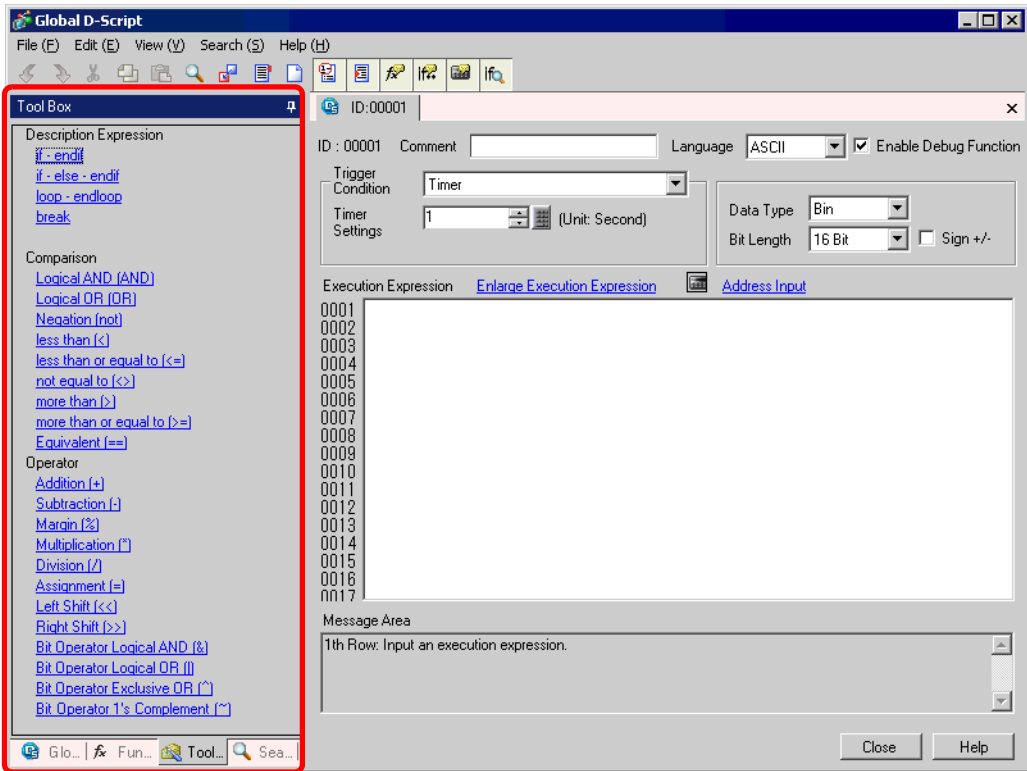
3 The [Global D-Script] dialog box will appear.



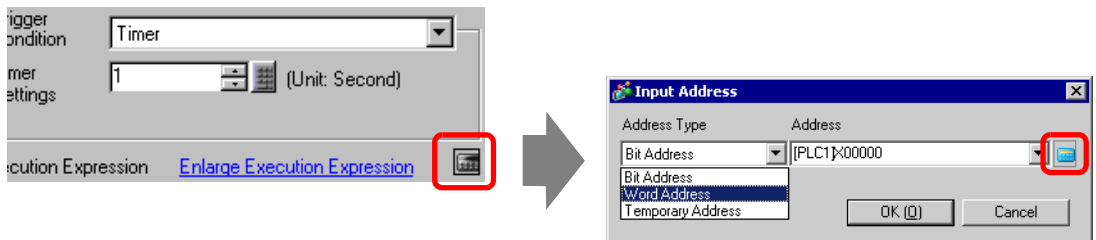
4 Set the script's execution condition (trigger). Select the timer as below and set the [Timer Settings] to 1 second.



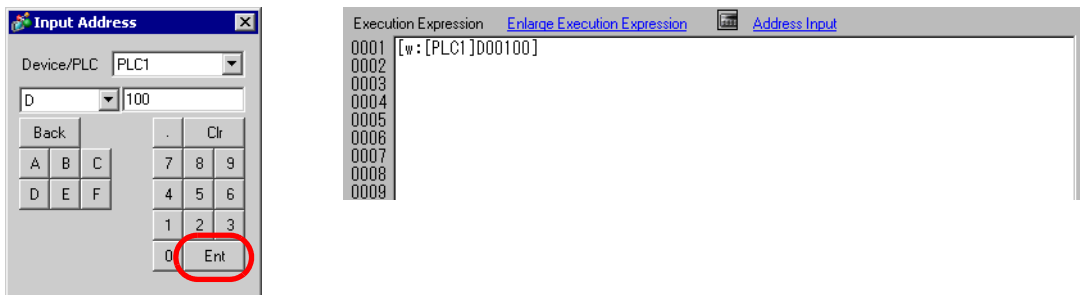
5 Click the [Tool Box] tab. The toolbox allows you to easily place a command that can be used in the script by clicking on it.



6 Create the first row of the script. The first row's action will set D00100 to an initial value of 0 and increment it by 1 each time the line is processed. Click the [Address Input] Dialog, select [Word Address], and click.

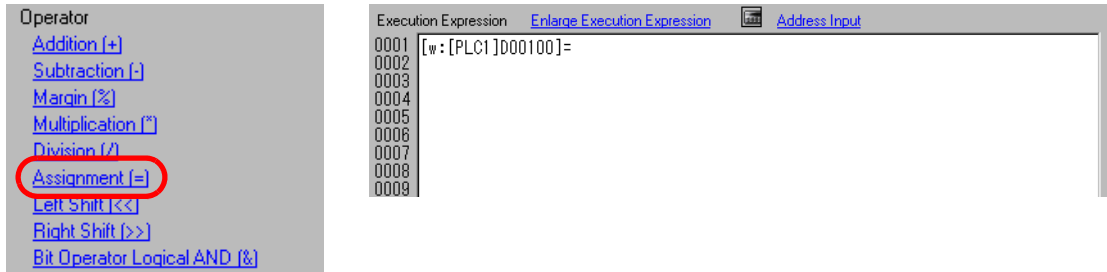


7 Input D00100, and click [ENT].

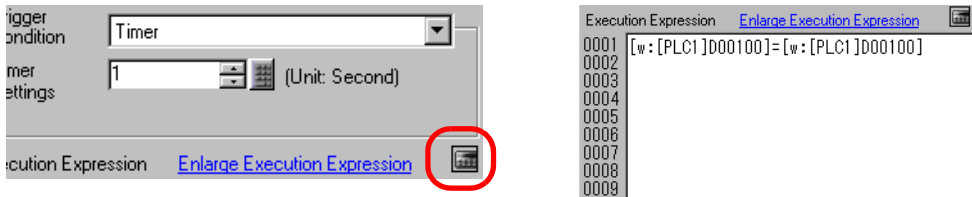




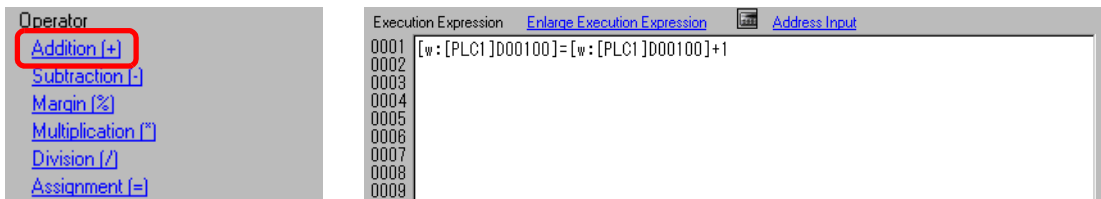
8 Click the [Tool Box]’s [Assignment (=)].



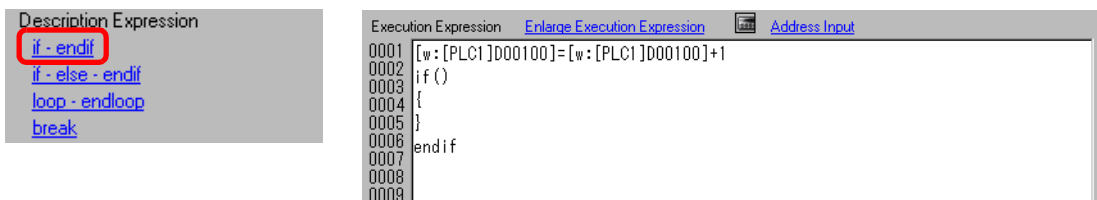
9 Place D00100 in the same way as procedures 6-7.



10 Click [Addition (+)] and input “1”. The first row is now complete.



11 Create the second row of the script. In the second row, when a condition enclosed with brackets “( )” following “if” becomes true, the process following the “if ( )” statement is executed. Click [if - endif].



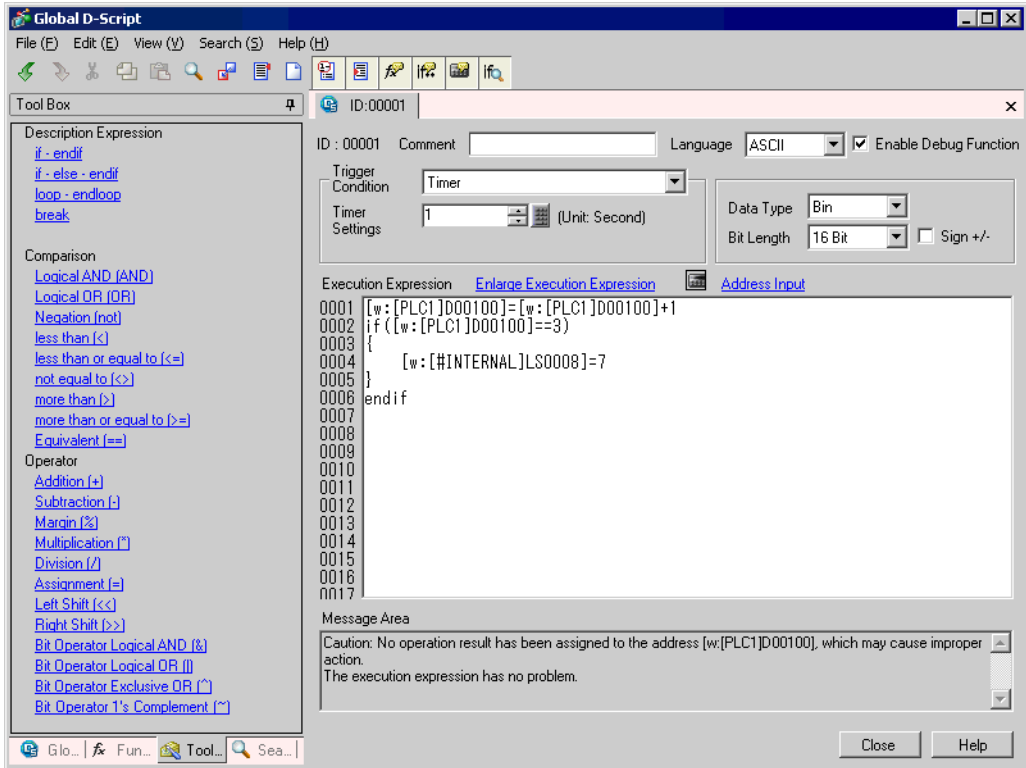
- 12 Create the condition expression inside the brackets “( )” following “if”. The condition expression compares the value stored in D00100 to “3”, and turns true if they are equal. Place the cursor inside the brackets “( )” and repeat steps 6 to 7 to place another D00100.

- 13 Click [Equivalent (==)] and input “3”. The second row is now complete.

- 14 Place the cursor inside the “{ }” brackets and enter a line feed. Repeat steps 6 to 7 to place another LS0008.

- 15 Click [Assignment (=)] and input “7”.

16 The script is complete.



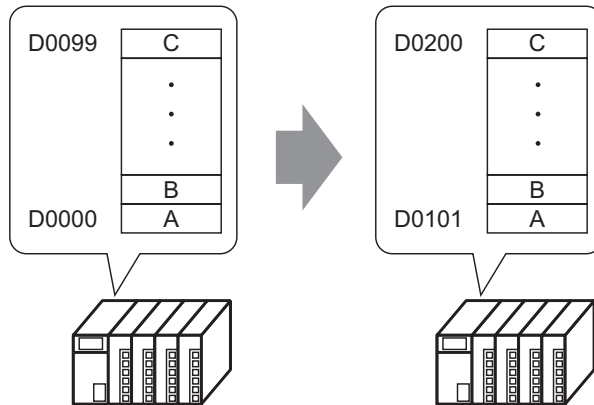
## 20.3 Copying Data in Blocks

**NOTE**

- Please refer to the settings guide for details.  
 ➔ “20.8.1 Common Settings Guide for D-Script” (page 20-48)

**Action**

Create a script which detects the rising edge (0 to 1) of bit address M0100 and copies data stored in the connected device into another address.

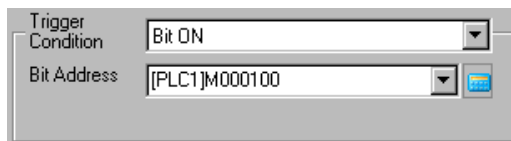


**Commands used**

Command	Function Summary
Copy Memory memcpy ( )	Copies a stored value into a device in one operation. Data for the number of Addresses will be copied to the copy destination Word Addresses beginning from the source data's first Word Address. [Format] memcpy ([Copy To Address], [Copy From Address], No. of Words)

**Trigger Condition**

Select the rising bit as follows, and set the [Bit Address] to M000100.




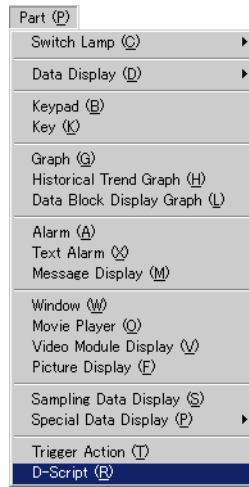
**Completed script**

```

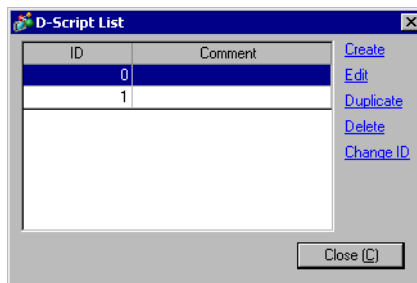
Execution Expression  Enlarge Execution Expression Address Input
0001 memcpy ([w:[PLC1]D00101], [w:[PLC1]D00000], 100)
0002
0003
0004
0005
0006
0007
    
```

### Creation procedure

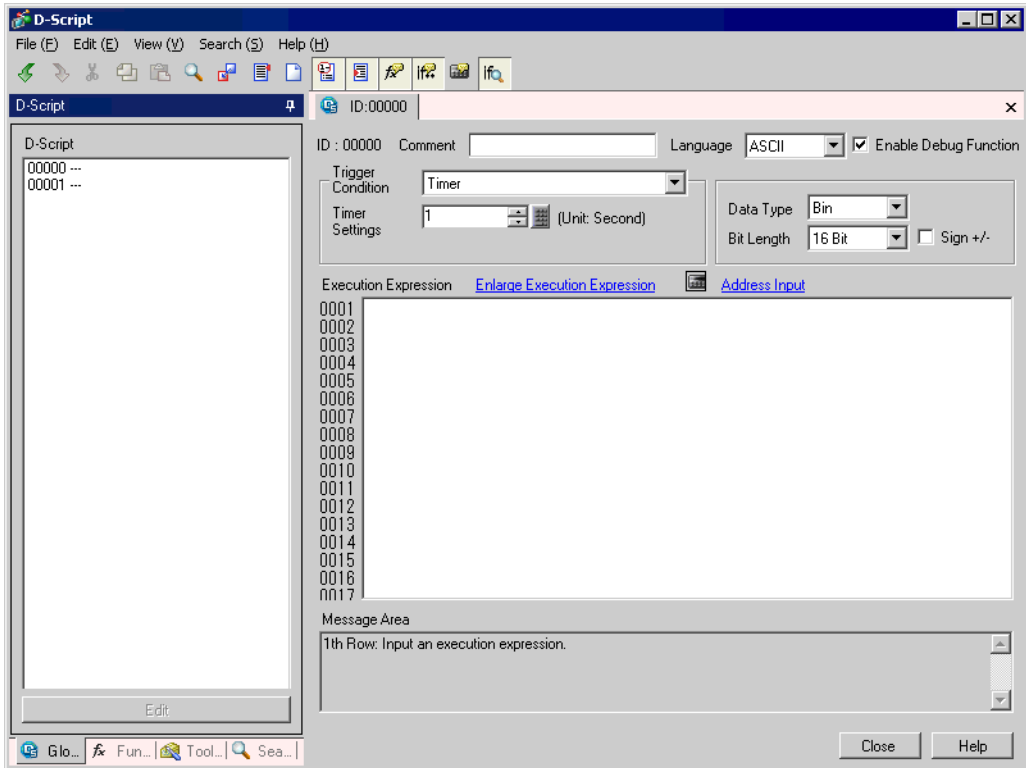
- 1 Select the [Part (P)] menu - [D-Script (R)] command,  or click.



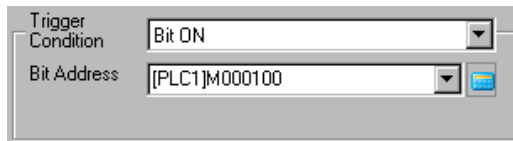
- 2 Click [Create]. If the script is already registered, the ID number will be displayed.



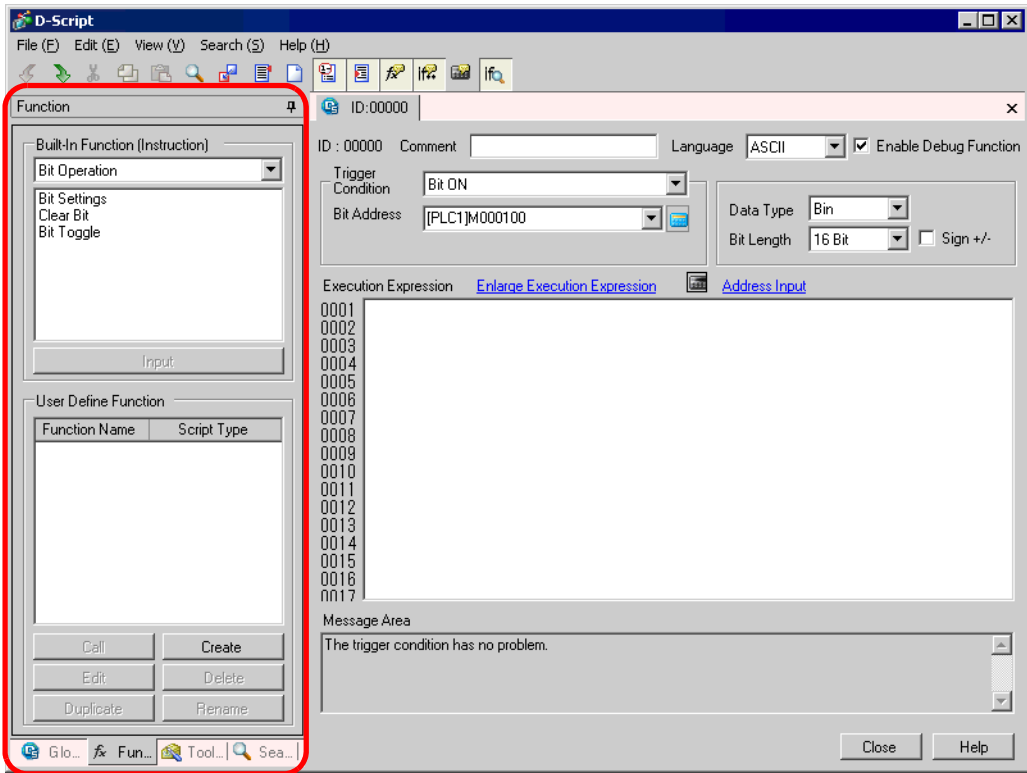
3 The [D-Script] dialog box will appear.



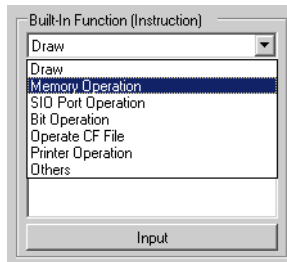
4 Set the script's execution condition (trigger). Select the rising bit as follows, and set the [Bit Address] to M000.



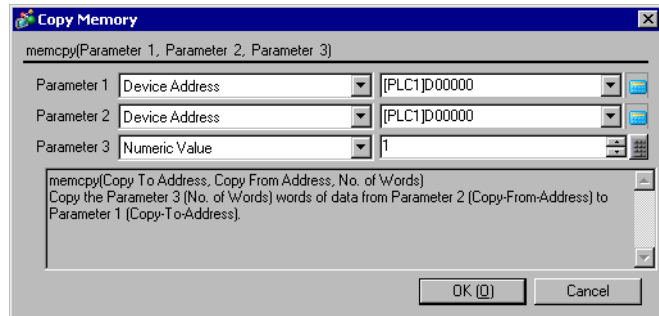
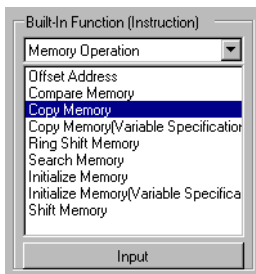
5 Click the [Function] tab. The built-in functions allow you to easily place a command that can be used in the script by clicking on it.



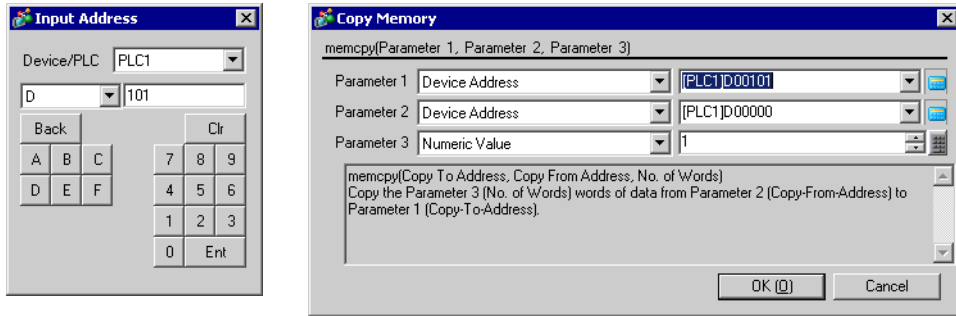
6 Select [Memory Operation] from the [Built-in Function (Instruction)] pull down menu.



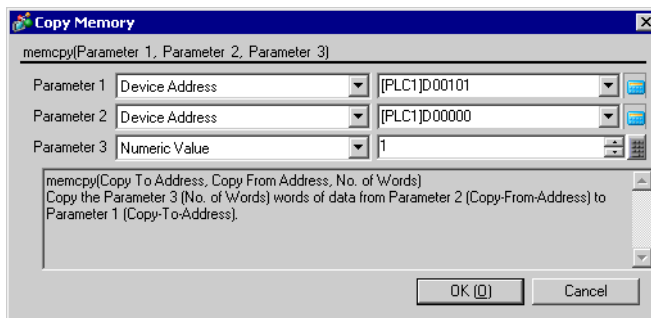
7 Double-click [Copy Memory] and in the following dialog box set the destination address, source address, and number of addresses. Click .



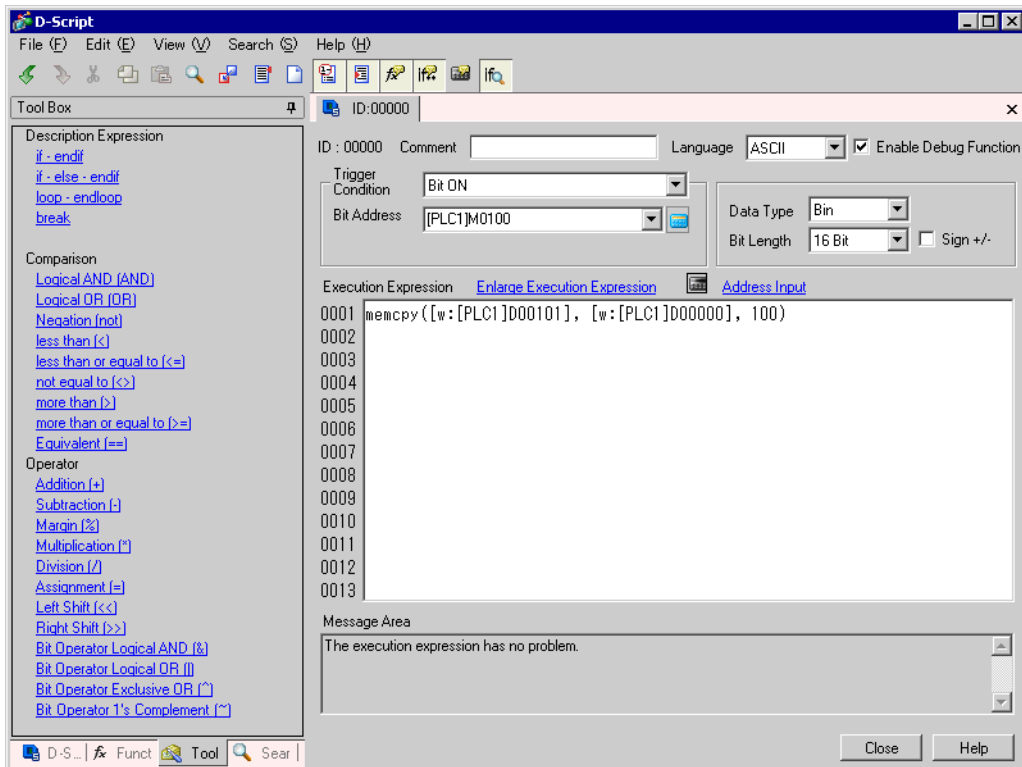
8 Input D00101, and click [ENT].



9 Input 100 for the number of addresses, repeat step 8 to set the source word address to D00000, and click [OK].



10 The script is complete.



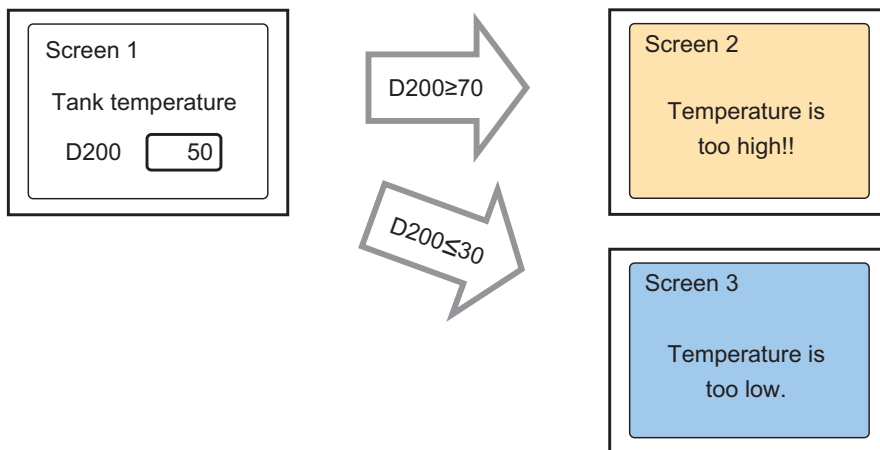


## 20.4 Displaying an Alarm when an Error Occurs

**NOTE** • Please refer to the settings guide for details.  
 ☞ “20.8.1 Common Settings Guide for D-Script” (page 20-48)

### Action

The temperature management system detects an error bit from the connected device and displays alarm messages when the temperature information storage address (D200) rises to 70 °C or higher, or falls to 30 °C or lower. Also, this script will count the number of detected errors.



The address that counts each time D200 rises to 70 °C or higher and stores the number of times: LS0300

The address that counts each time D200 falls to 30 °C or lower and stores the number of times: LS0301

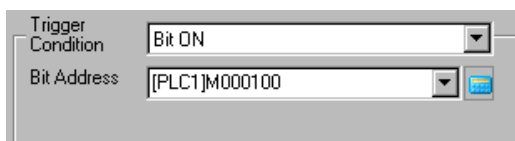
The address that stores the screen number to display an alarm screen: LS0302

### Commands used

Command	Function Summary
if ( )	When a condition enclosed with brackets “( )” following “if” becomes true, the process following the “if ( )” statement is executed.
more than or equal to (>=)	True if N1 is more than or equal to N2 (N1 ≥ N2).
Assignment (=)	Assigns the right side value to the left side.
Addition (+)	Adds a constant to a word device’s data.
less than or equal to (<=)	True if N1 is less than or equal to N2 (N1 ≤ N2).

### Trigger Condition

Select the rising bit as follows, and set the [Bit Address] to M000100.




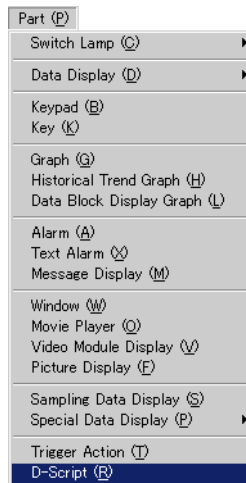
## Completed script

```

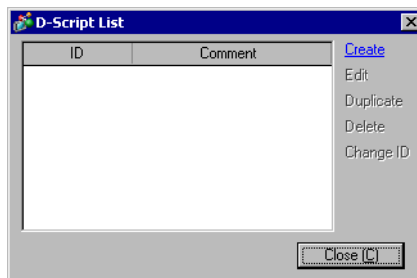
Execution Expression  Enlarge Execution Expression  Address Input
0001  if ([w:[PLC1]D00200]>=70) // When temp is greater than 70 degrees
0002  {
0003      [w:[#INTERNAL]LS0302]=100 // Greater than 70 degrees alarm screen number 100
0004      [w:[#INTERNAL]LS0300]=[w:[#INTERNAL]LS0300]+1 // Increase error count
0005  }
0006  endif
0007
0008  if ([w:[PLC1]D00200]<=30) // When temp is less than 30 degrees
0009  {
0010      [w:[#INTERNAL]LS0302]=101 // Less than 30 degrees alarm screen number 101
0011      [w:[#INTERNAL]LS0301]=[w:[#INTERNAL]LS0301]+1 // Increase error count
0012  }
0013  endif
0014
0015
0016
0017
    
```

## Creation procedure

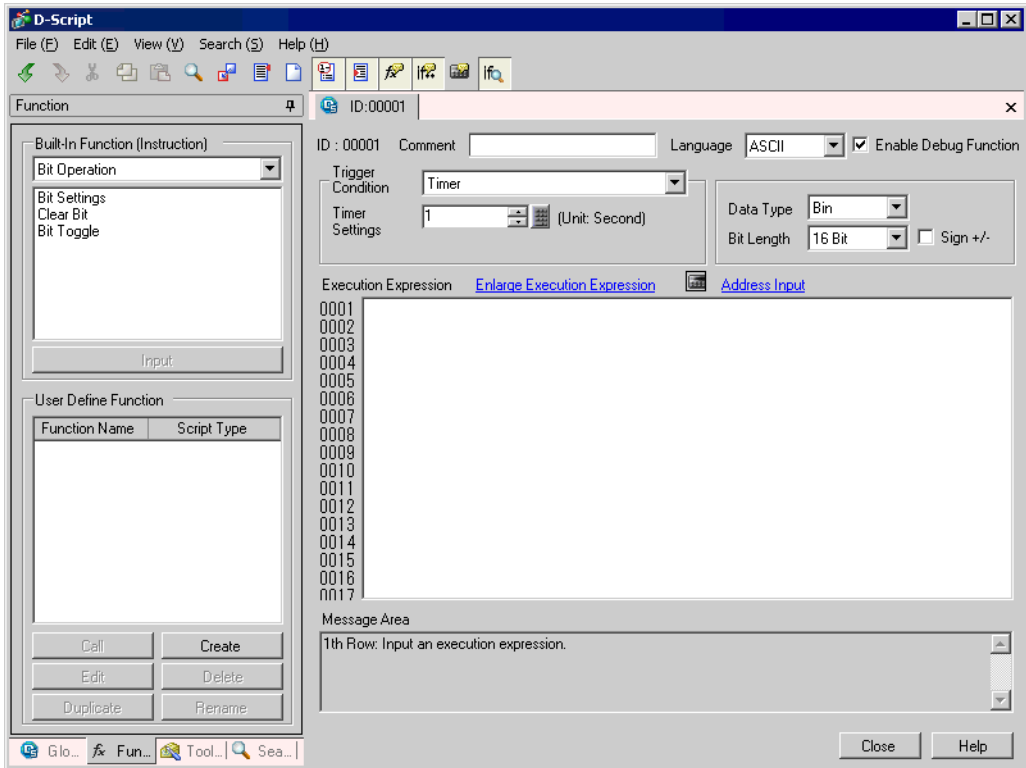
1 Select the [Part] menu - [D-Script] command, or click .



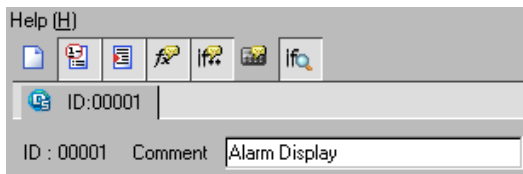
2 Click [Create]. If the script is already registered, the ID number will be displayed.



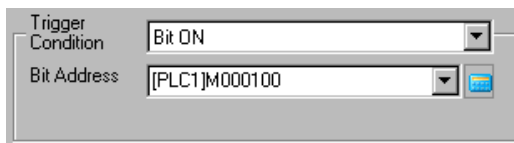
3 The [D-Script] dialog box will appear.




4 Set the comment. Input “Alarm Display”.



5 Set the script’s execution condition (trigger). Select the rising bit as follows, and set the [Bit Address] to M000100.




### 6 Create a program in the Action area by entering Functions, Statements, and Expressions and the setting is complete.

```
Execution Expression Enlarge Execution Expression  Address Input
0001 if ([w:[PLC1]D00200]>=70) // When temp is greater than 70 degrees
0002 {
0003     [w:[#INTERNAL]LS0302]=100 // Greater than 70 degrees alarm screen number 100
0004     [w:[#INTERNAL]LS0300]=[w:[#INTERNAL]LS0300]+1 // Increase error count
0005 }
0006 endif
0007
0008 if ([w:[PLC1]D00200]<=30) // When temp is less than 30 degrees
0009 {
0010     [w:[#INTERNAL]LS0302]=101 // Less than 30 degrees alarm screen number 101
0011     [w:[#INTERNAL]LS0301]=[w:[#INTERNAL]LS0301]+1 // Increase error count
0012 }
0013 endif
0014
0015
0016
0017
```

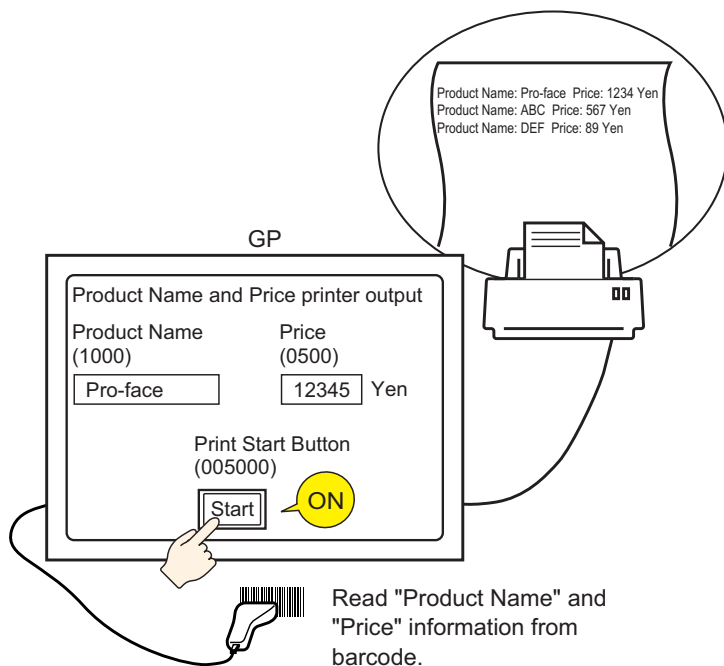
## 20.5 Communicating with Peripheral Devices not Supported by Regular Scripts

**NOTE**

- Please refer to the settings guide for details.  
 "20.8.1 Common Settings Guide for D-Script" (page 20-48)

■ **Action**

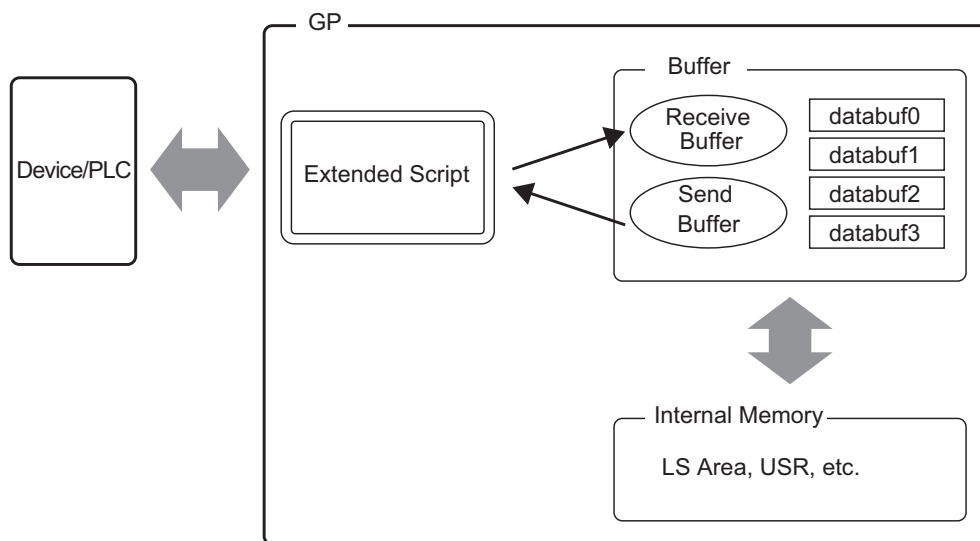
Create an extended script to output the data read from a barcode connected to the USB to a serial printer connected to COM1.



## ■ Extended Script Structure

Extended Scripts are specialty scripts for communicating between the GP's internal Serial Port and connected input/output devices.

For Extended Script data management, as shown in the following picture, data is stored in databuf0 to databuf3 via the Send/Receive Buffer. Databuf is not divided by address, so when editing data from the device/PLC, please store it in internal memory before editing it.



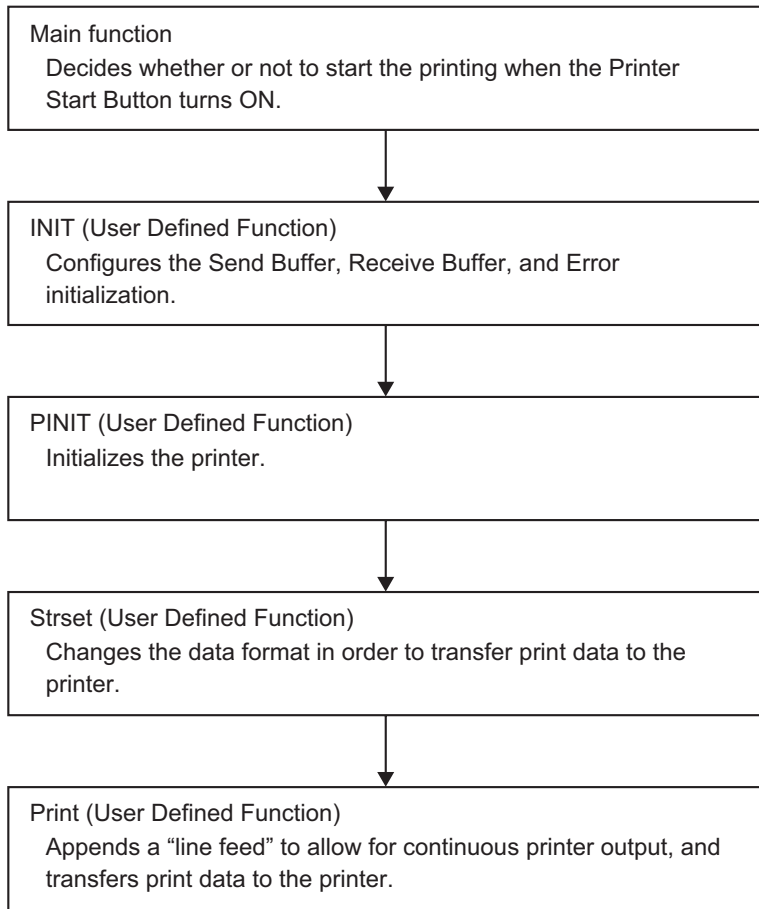
### Receive Buffer/Send Buffer

For communication with the device/PLC, this acts as a bit memory space which distinguishes received/sent data in real time.

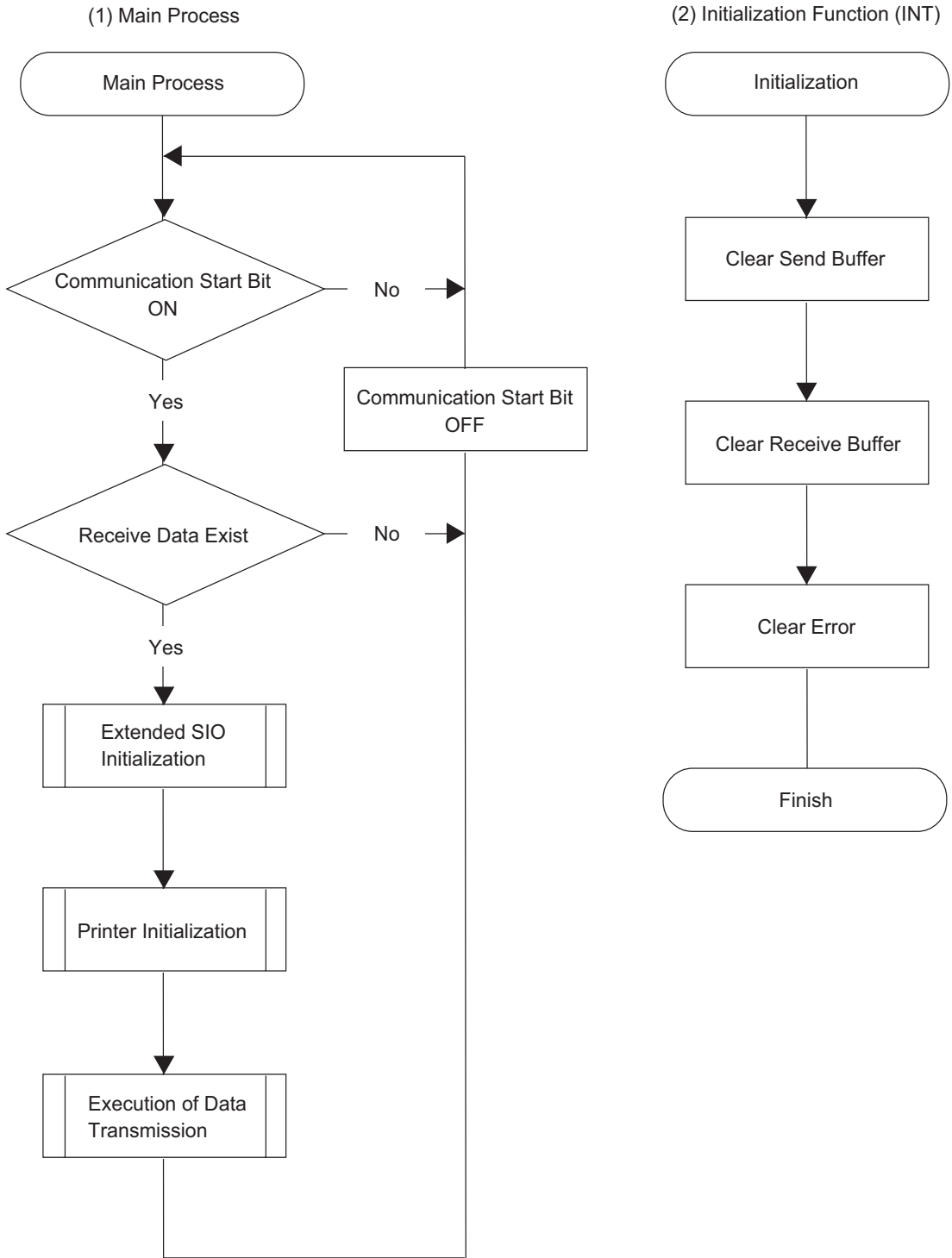
### databuf0-databuf3

These are byte (8-bit) memory spaces that act as data storage locations. Each buffer's size is 1 KB.

## ■ Procedure to Create Scripts

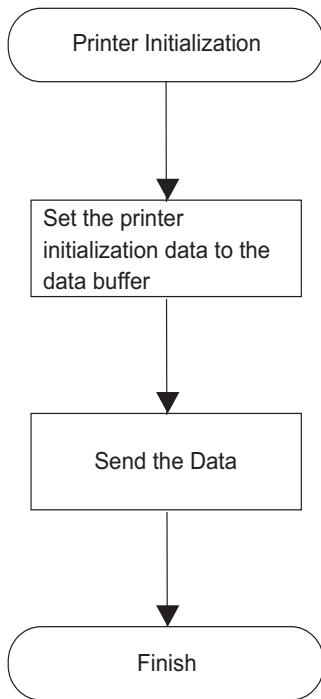


■ Flow Chart

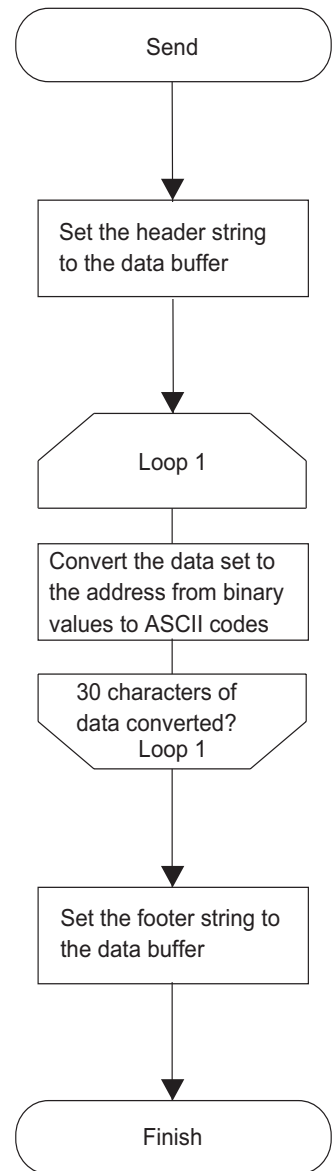




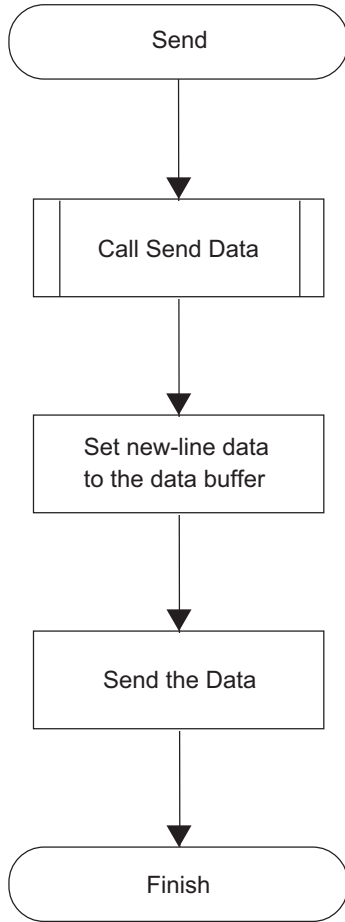
(3) Printer Initialization Function (PINIT)



(4) String Function (Strset)



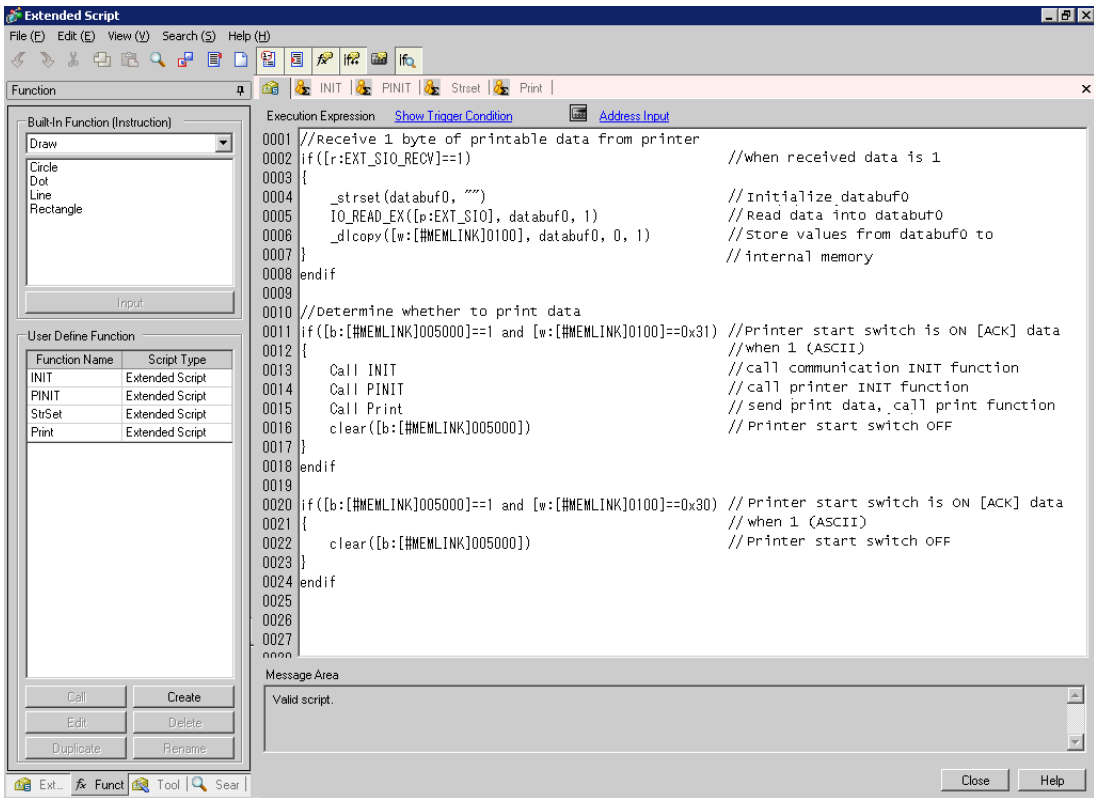
(5) Send Function (Print)



## ■ Script Function Summary

### ◆ Main Function

Completed script



### Function Summary

When the Printer Start Button (internal memory 005000) turns ON, decides whether or not to start printing from the 1st byte of Print Permit data.

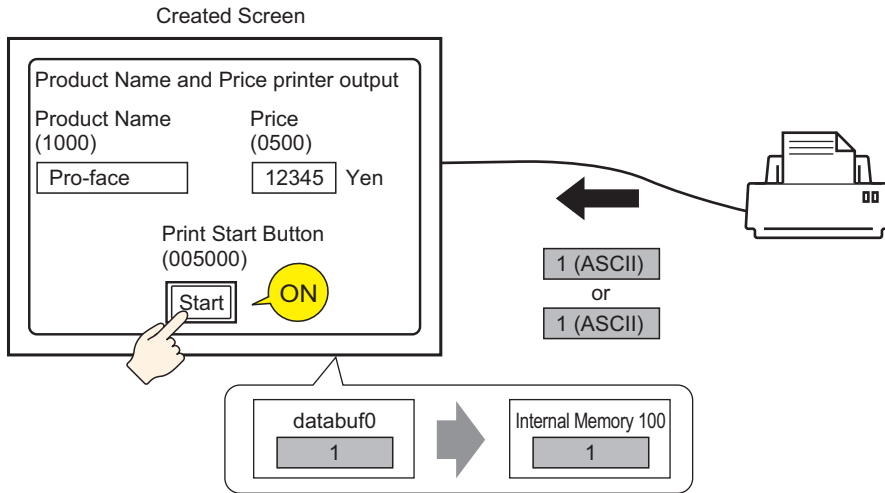
The Print Permit data will perform the following actions as an example of the printer's specifications.

Print Preparation OK: Send 0x31 (ASCII code "1" ) to the device/PLC.

Print Preparation Invalid: Send 0x30 (ASCII code "0") to the device/PLC.

The GP receives the Print Permit data in databuf0 and this data is moved to easily accessible internal memory 100 with the following script handling.

When internal memory 100 is 0 x 31 (ASCII code “1”), printing starts, when it is 0 x 30 (ASCII code “0”), it returns to the beginning and repeats this process until 0x31’s data is received.



## ◆ INIT (User Defined Function)

Completed script

```

Execution Expression  Enlarge Execution Expression  Address Input
0001 [c:EXT_S10_CTRL00]=1 //Send buffer clear
0002 [c:EXT_S10_CTRL01]=1 //Receive buffer clear
0003 [c:EXT_S10_CTRL02]=1 //Error clear
0004
    
```

### Function Summary

Configure the Send Buffer, Receive Buffer, and Error initialization.

## ◆ PINIT (User Defined Function)

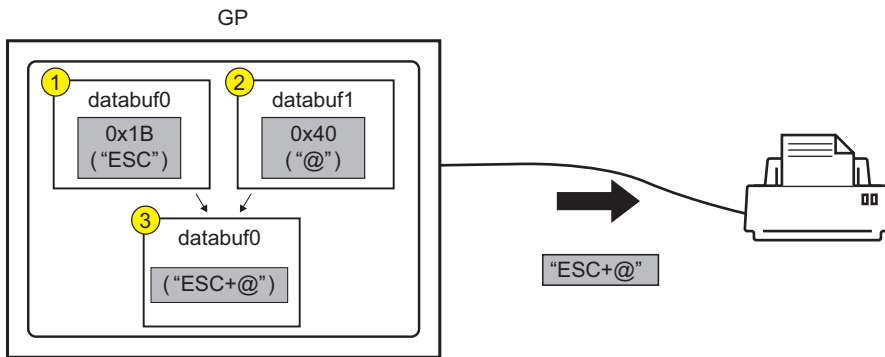
Completed script

```

Execution Expression  Enlarge Execution Expression  Address Input
0001 //Printer initialization (ESC/P command "ESC + @")
0002
0003 _strset(databuf0, "") //Clear databuf0
0004 _strset(databuf0, 0x1B) //Set ASCII code "ESC"
0005 _strset(databuf1, "") //Clear databuf1
0006 _strset(databuf1, 0x40) //Set ASCII code "@"
0007 _strcat(databuf0, databuf1) //Add databuf1 to end of databuf0
0008 _strlen([t:0000], databuf0) //Store data length to temporary address
0009
0010 //Send data over serial port
0011
0012 IO_WRITE_EX([p:EXT_S10], databuf0, [t:0000]) //Send databuf0, amount defined by temporary address value
0013
    
```

### Function Summary

Initializes the printer. Send the ESC/P command "ESC+@" to the printer.



◆ **Strset (User Defined Function)**

Completed script

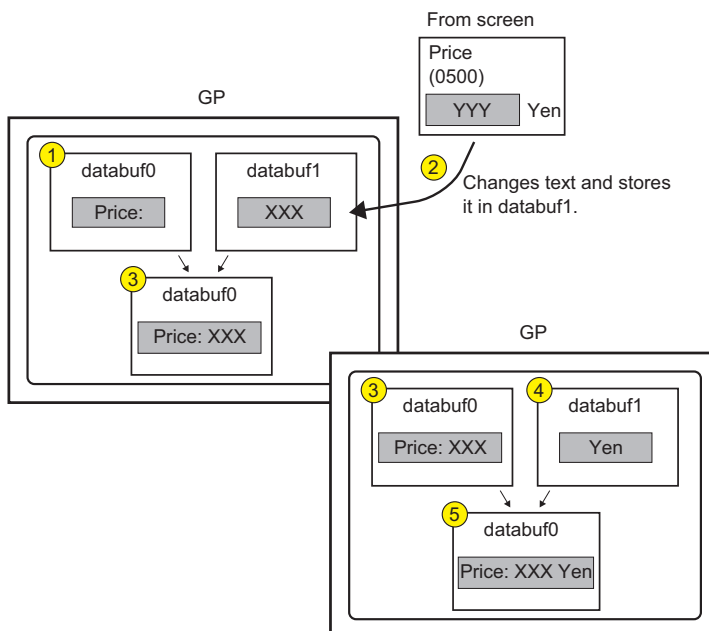
```

Execution Expression  Enlarge Execution Expression  Address Input
0001 //String example, add "Price :" and "$"
0002 _strset(databuf0, "") //Initialize databuf0
0003 _strset(databuf0, "Price : ") //Store text "Price :" to databuf0
0004 _bin2decasc(databuf0, [w:[#MEMLINK]0500]) //Convert value to string and store in databuf1
0005 _strcat(databuf0, databuf1) // Add databuf1 to end of databuf0
0006 _strset(databuf1, "") //Initialize databuf1
0007 _strset(databuf1, "$") //Store text "$" to databuf1
0008 _strcat(databuf0, databuf1) // Add databuf1 to end of databuf0
0009
0010 //Initialize temporary address
0011 [t:0001]=0
0012 [t:0002]=0
0013
0014 //Store to internal memory word units, consecutive characters into byte units (30 characters)
0015 loop()
0016 {
0017     [w:[#MEMLINK]2000]#[t:0002]=[w:[#MEMLINK]1000]#[t:0001] >> 8 //Store top byte into bottom byte
0018     [w:[#MEMLINK]2001]#[t:0002]=[w:[#MEMLINK]1000]#[t:0001] & 0xFF //Erase top byte and store in next address
0019     [t:0001]=[t:0001]+1 //Address offset + 1
0020     [t:0002]=[t:0002]+2 //Address offset + 2
0021     if([t:0001]==15) //Store 2 words into 2 bytes and repeat 15 times
0022     {
0023         break
0024     }
0025     endif
0026 }
0027 endloop
0028 _ldcopy(databuf2, [w:[#MEMLINK]2000], 30) //Store internal memory 2000~2030 to data buffer as characters
0029
0030 //Add string "Item :"
0031 _strset(databuf1, "") //Initialize databuf1
0032 _strset(databuf1, "Item :") //Store string "Item :" into databuf1
0033 _strcat(databuf1, databuf0) // Add databuf0 to end of databuf1
0034
0035 //Add Item and Price strings
0036 _strcat(databuf1, databuf0) // Add databuf0 to end of databuf1
0037
0038

```

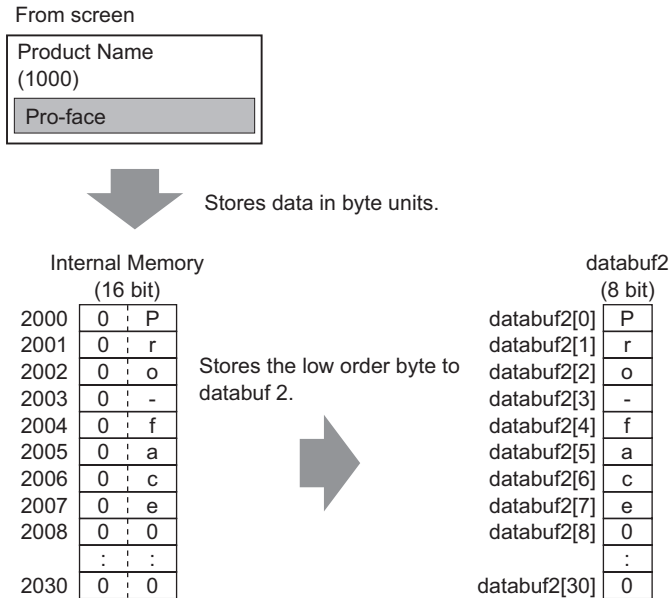
Function Summary

- 1 Append the text "Price:" and "Circle" to the price data stored internal memory 0500.

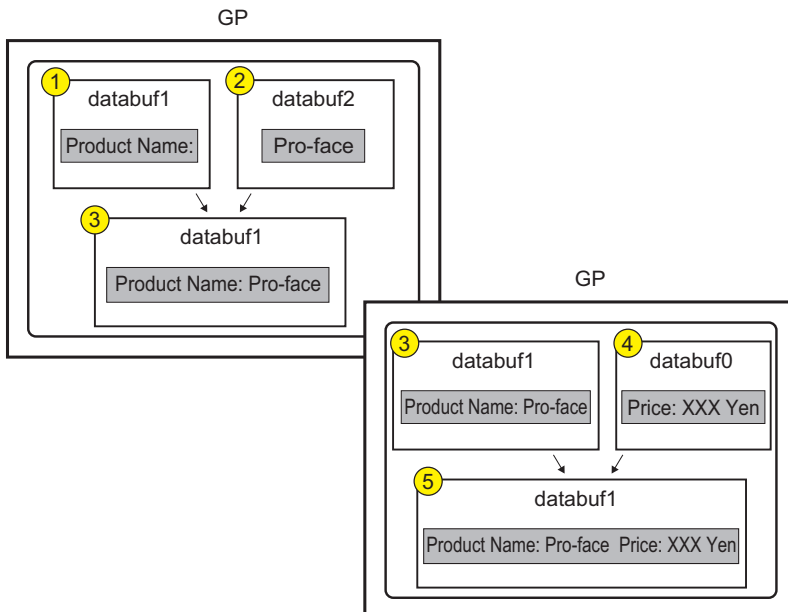


2 Change the data format in order to send print data to the printer. Divide the string data (Product Name) stored sequentially in internal memory 1000 into byte units, and store into internal memory 2000 to 2030 as low order byte string data. Use the function `_ldcopy` and store the data in `databuf2` in order of the consecutive word address's lowest byte.

**NOTE** • The `_ldcopy` function takes data stored as words, and stores only the lower order bytes in the buffer, while higher order byte data is ignored.



3 Append the text “Product Name:” and “Price” to `databuf2`.



◆ **Print (User Defined Function)**

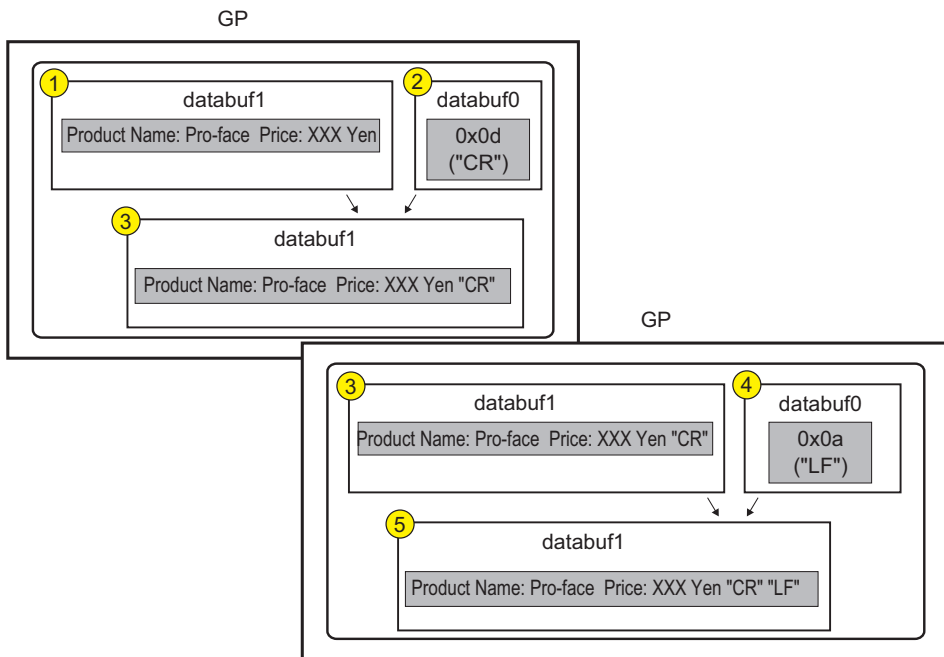
Completed script

```

Execution Expression  Enlarge Execution Expression  Address Input
0001 Call Strset //Call string data function
0002 _strset(databuf0, "") //Clear databuf1
0003
0004 //Text delimiter
0005
0006 _strset(databuf0, 0x0d) //Return to start of row
0007 _strcat(databuf1, databuf0) // Add databuf1 to end of databuf0
0008 _strset(databuf0, "") //Clear databuf1
0009 _strset(databuf0, 0x0a) //New line
0010 _strcat(databuf1, databuf0) // Add databuf1 to end of databuf0
0011
0012 _strlen([t:0000], databuf1) // Store data length to temporary address
0013
0014 //Send data over serial port
0015
0016 IO_WRITE_EX([p:EXT_SIO], databuf1, [t:0000]) // Send databuf0, amount defined by temporary address value
0017
    
```

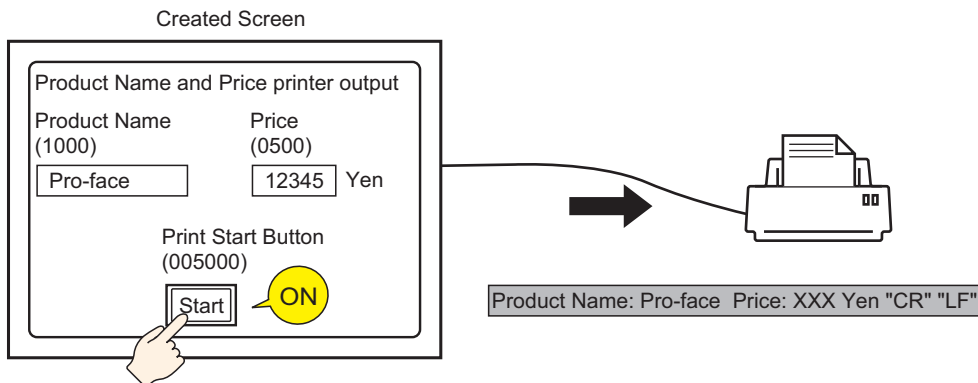
Function Summary

- 1 Append a “line feed” to allow for continuous printer output.





2 Set the print data to the printer.

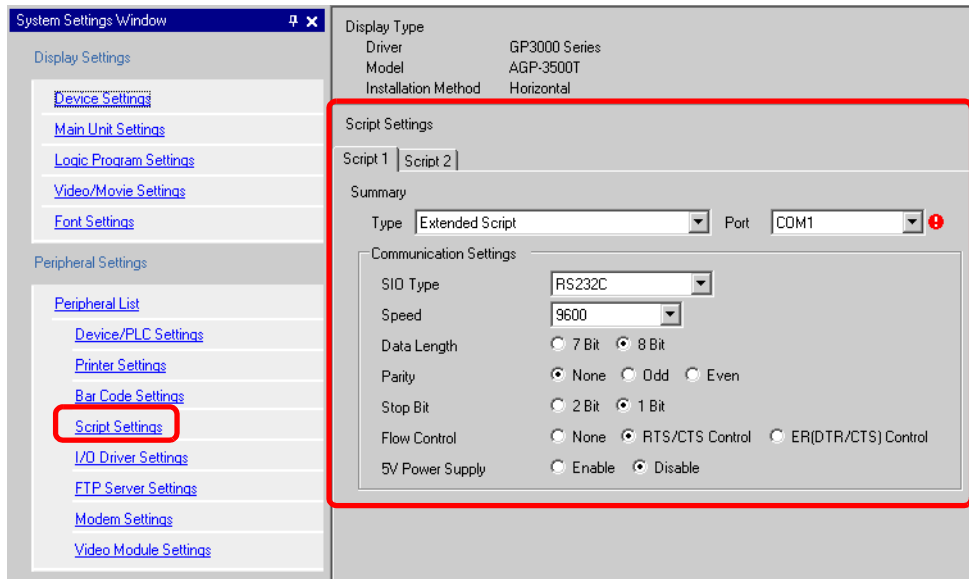


■ Commands used

Command	Function Summary
if ( )	When a condition enclosed with brackets “( )” following “if” becomes true, the process following the “if ( )” statement is executed.
Label Settings [r:EXT_SIO_RECV]	Shows the quantity of data (number of bytes) that has been received at that time. The received data size is a read-only feature.
Equivalent (==)	True if N1 is equal to N2 (N1 = N2).
Text Settings (_strset)	Stores a fixed string in the data buffer.
Extended Receive (IO_READ_EX)	Receives data of the size indicated in Received Data Size (bytes) from the Extended SIO and stores it in the data buffer.
From Data Buffer to Internal Device (_dlcopy)	Each byte of string data stored in the offset of the data buffer is copied to the LS area according to the number of strings.
Label Settings [c:EXT_SIO_CTRL **]	This control variable is used to clear the Send buffer, Receive buffer, and error status.
Connect Text (_strcat)	Concatenates a character string or character code with the text buffer.
Text Length (_strlen)	Obtains the length of the stored string.
Extended Send (IO_WRITE_EX)	Sends the data in the data buffer with Extended SIO according to the size of No. of Send Bytes.
Assignment (=)	Assigns the right side value to the left side.
Addition (+)	Adds a constant to a word device’s data.
Numeric Value Decimal String Conversion (_bin2decasc)	This function is used to convert an integer to a decimal string.
From Internal Device To Data Buffer (_ldcopy)	The data of the string stored in the LS area is copied to the data buffer according to the number of strings in a byte-by-byte transfer.

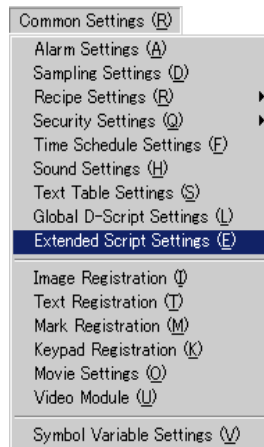
## ■ Creation procedure

- 1 Set the Extended Script which will be used to communicate. Click [Project(F)] - [System Settings (C)] - [Script Settings]. Make sure to set the [Type] to [Extended Script].

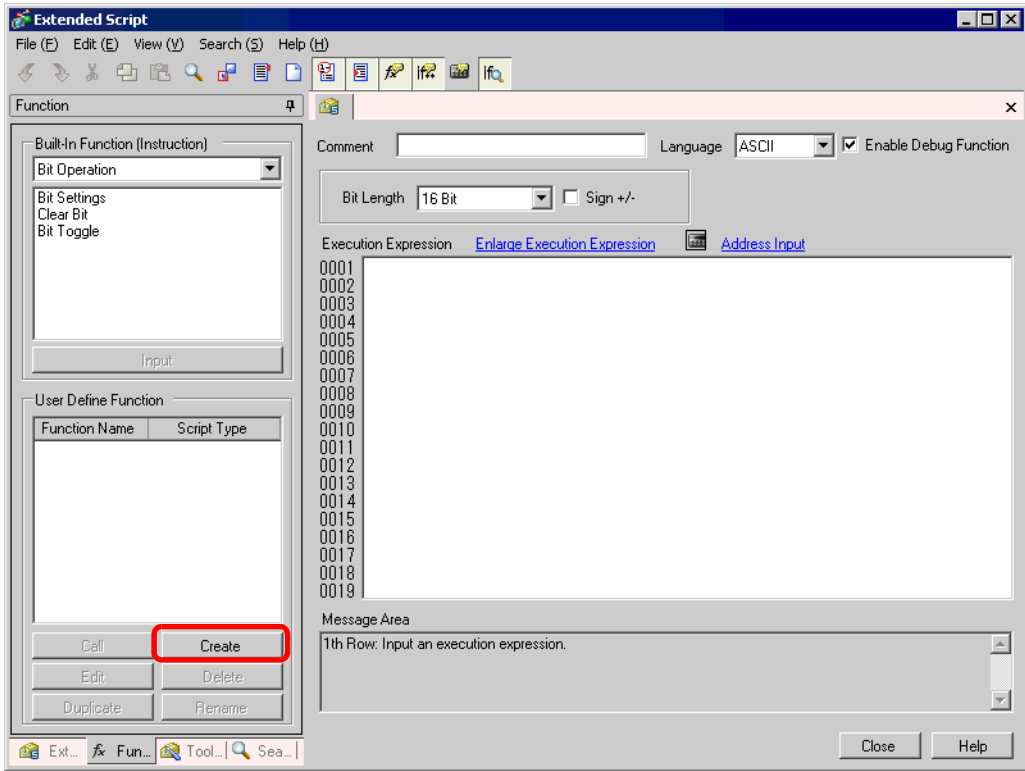


There are two tabs for the script settings. The above picture is using the [Script 1] tab. Set the [Port] to COM1 or COM2, and set the [Communication Settings] to match the Extended SIO.

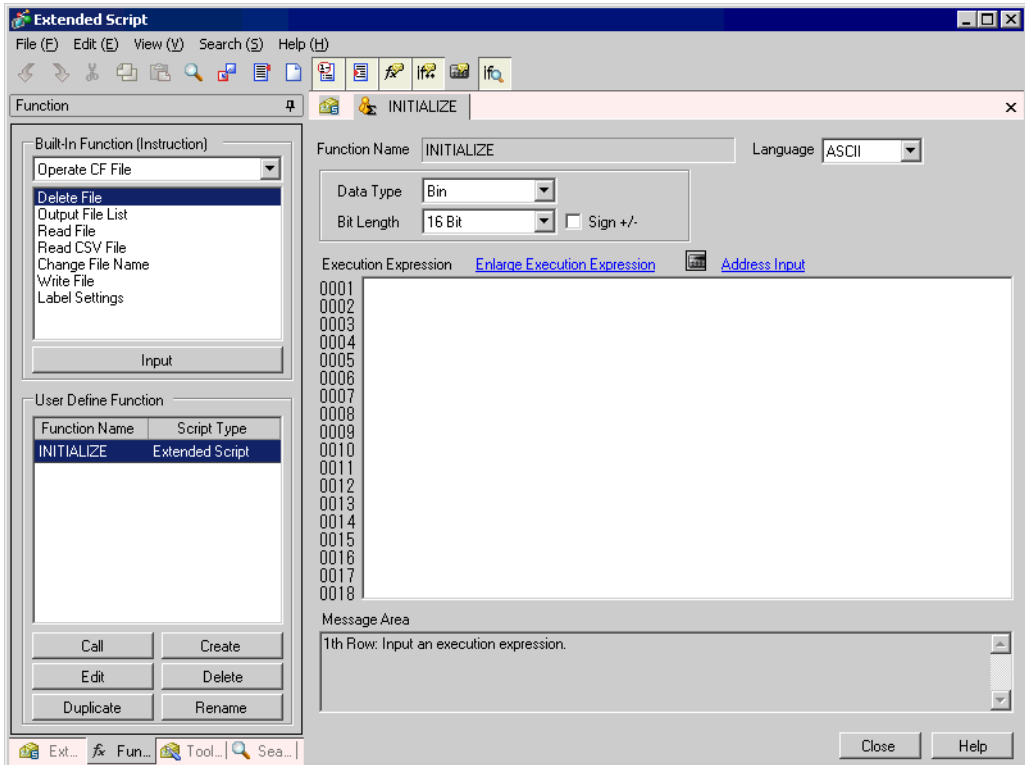
- 2 Select [Extended Script Settings (E)] from the [Common Settings (R)] menu.



3 Register “INIT” as a User-Defined Function. Click the [Function] tab and click the user-defined function frame’s [Create] button.



4 Input [INIT] as the function name, click [OK], and the following screen will appear.



5 Create a script in the Execution Expression with Commands, statements, and Constant input.

```

Execution Expression  Enlarge Execution Expression  Address Input
0001 [c:EXT_SIO_CTRL00]=1 //Send buffer clear
0002 [c:EXT_SIO_CTRL01]=1 //Receive buffer clear
0003 [c:EXT_SIO_CTRL02]=1 //Error clear
0004
    
```

6 In the same manner, register “PINIT” as a User-Defined Function. Input [PINIT] as the function name and create the following script in Execution Expression.

```

Execution Expression  Enlarge Execution Expression  Address Input
0001 //Printer initialization (ESC/P command "ESC + @")
0002
0003 _strset(databuf0, "") //Clear databuf0
0004 _strset(databuf0, 0x1B) //Set ASCII code "ESC"
0005 _strset(databuf1, "") //Clear databuf1
0006 _strset(databuf1, 0x40) //Set ASCII code "@"
0007 _strcat(databuf0, databuf1) //Add databuf1 to end of databuf0
0008 _strlen([t:0000], databuf0) //Store data length to temporary address
0009
0010 //Send data over serial port
0011
0012 IO_WRITE_EX([p:EXT_SIO], databuf0, [t:0000]) //Send databuf0, amount defined by temporary address value
0013
    
```

7 In the same manner, register “Strset” as a User-Defined Function. Input [Strset] as the function name and create the following script in Execution Expression.

```

Execution Expression  Enlarge Execution Expression  Address Input
0001 //String example, add "Price :" and "$"
0002 _strset(databuf0, "") //Initialize databuf0
0003 _strset(databuf0, "Price : ") //Store text "Price :" to databuf0
0004 _bin2deasc(databuf0, [w:[#MEMLINK]0500]) //Convert value to string and store in databuf1
0005 _strcat(databuf0, databuf1) // Add databuf1 to end of databuf0
0006 _strset(databuf1, "") //Initialize databuf1
0007 _strset(databuf1, "$") //Store text "$" to databuf1
0008 _strcat(databuf0, databuf1) // Add databuf1 to end of databuf0
0009
0010 //Initialize temporary address
0011 [t:0001]=0
0012 [t:0002]=0
0013
0014 //Store to internal memory word units, consecutive characters into byte units (30 characters)
0015 loop()
0016 {
0017     [w:[#MEMLINK]2000]#[t:0002]=[w:[#MEMLINK]1000]#[t:0001] >> 8 //Store top byte into bottom byte
0018     [w:[#MEMLINK]2001]#[t:0002]=[w:[#MEMLINK]1000]#[t:0001] & 0xFF //Erase top byte and store in next address
0019     [t:0001]=[t:0001]+1 //Address offset + 1
0020     [t:0002]=[t:0002]+2 //Address offset + 2
0021     if ([t:0001]==15) //Store 2 words into 2 bytes and repeat 15 times
0022     {
0023         break
0024     }
0025     endif
0026 }
0027 endloop
0028 _ldcopy(databuf2, [w:[#MEMLINK]2000], 30) //Store internal memory 2000~2030 to data buffer as characters
0029
0030 //Add string "Item :"
0031 _strset(databuf1, "") //Initialize databuf1
0032 _strset(databuf1, "Item :") //Store string "Item :" into databuf1
0033 _strcat(databuf1, databuf2) // Add databuf1 to end of databuf0
0034
0035 //Add Item and Price strings
0036 _strcat(databuf1, databuf0) // Add databuf0 to end of databuf1
    
```

(1)

(2)

(3)

8 In the same manner, register “Print” as a User-Defined Function. Input [Print] as the function name and create the following script in Execution Expression.

```

Execution Expression Enlarge Execution Expression  Address Input
0001 Call Strset //Call string data function
0002 _strset(databuf0, "") //Clear databuf1
0003
0004 //Text delimiter
0005
0006 _strset(databuf0, 0x0d) //Return to start of row
0007 _strcat(databuf1, databuf0) // Add databuf1 to end of databuf0
0008 _strset(databuf0, "") //Clear databuf1
0009 _strset(databuf0, 0x0a) //New line
0010 _strcat(databuf1, databuf0) // Add databuf1 to end of databuf0
0011
0012 _strlen([t:0000], databuf1) // Store data length to temporary address
0013
0014 //Send data over serial port
0015
0016 IO_WRITE_EX([p:EXT_SIO], databuf1, [t:0000]) // Send databuf0, amount defined by temporary address value
0017
    
```

9 Create the main script. Create the following script in Execution Expression and the settings are complete.

```

Execution Expression Enlarge Execution Expression  Address Input
0001 //Receive 1 byte of printable data from printer
0002 if([r:EXT_SIO_RECV]==1) //When received data is 1
0003 {
0004     _strset(databuf0, "") //Initialize databuf0
0005     IO_READ_EX([p:EXT_SIO], databuf0, 1) //Read data into databuf0
0006     _d1copy([w:#MEMLINK]0100), databuf0, 0, 1) //Store values from databuf0 to internal memory
0007 }
0008 endif
0009
0010 //Determine whether to print data
0011 if([b:#MEMLINK]005000)==1 and [w:#MEMLINK]0100]==0x31) //Printer start switch is ON [ACK] data
0012 { //when 1 (ASCII)
0013     Call INIT //call communication INIT function
0014     Call PINIT //call printer INIT function
0015     Call Print //send print data, call print function
0016     clear([b:#MEMLINK]005000) //Printer start switch OFF
0017 }
0018 endif
0019
0020 if([b:#MEMLINK]005000)==1 and [w:#MEMLINK]0100]==0x30) //Printer start switch is ON [ACK] data
0021 { //when 0 (ASCII)
0022     clear([b:#MEMLINK]005000) //printer start switch OFF
0023 }
0024 endif
    
```

**NOTE** • When placing the user-defined functions created in steps 3 to 9 into the main script, select the function to be placed and click [Call] on the [Function] tab. The function will be placed using “Call Function Name”

## 20.6 Procedure for Creating Scripts

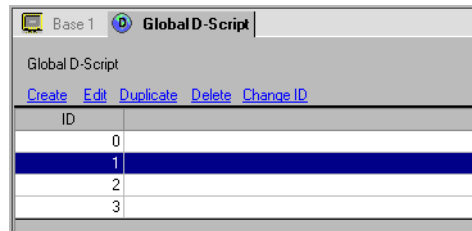
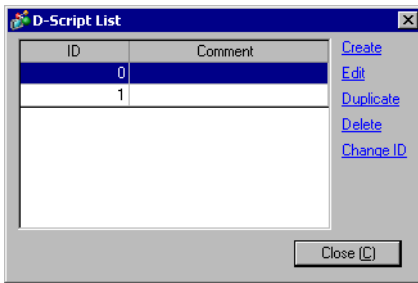
### 20.6.1 Procedure for Creating D-Scripts/Global D-Scripts

Open [D-Script (R)] from the [Part (P)] menu.

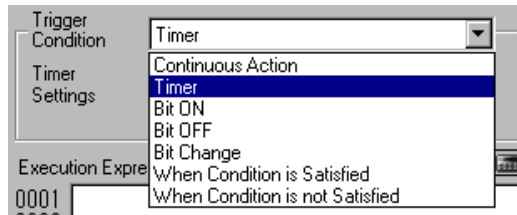
Open [Global D-Script Settings (L)] from the [Common Settings (R)] menu.

Click [Create]. If accessing a previously registered script, designate the ID number and click [Edit], or double-click the ID No.'s row.

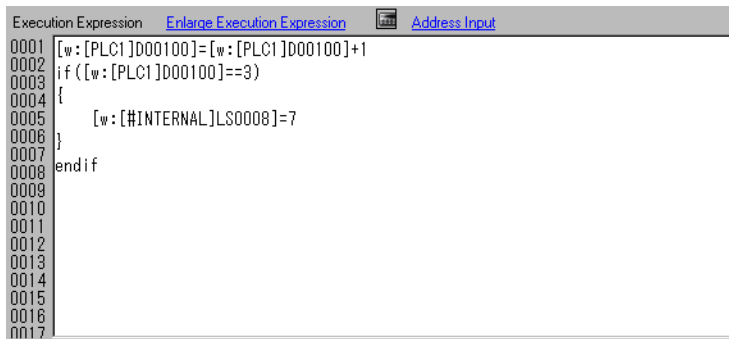
Click [Create]. If accessing a previously registered script, designate the ID number and click [Edit], or double-click the ID No.'s row.



Set the trigger condition that will cause the script to execute. For more information about this function, please refer to “20.7 Trigger Condition Setup” (page 20-42) .



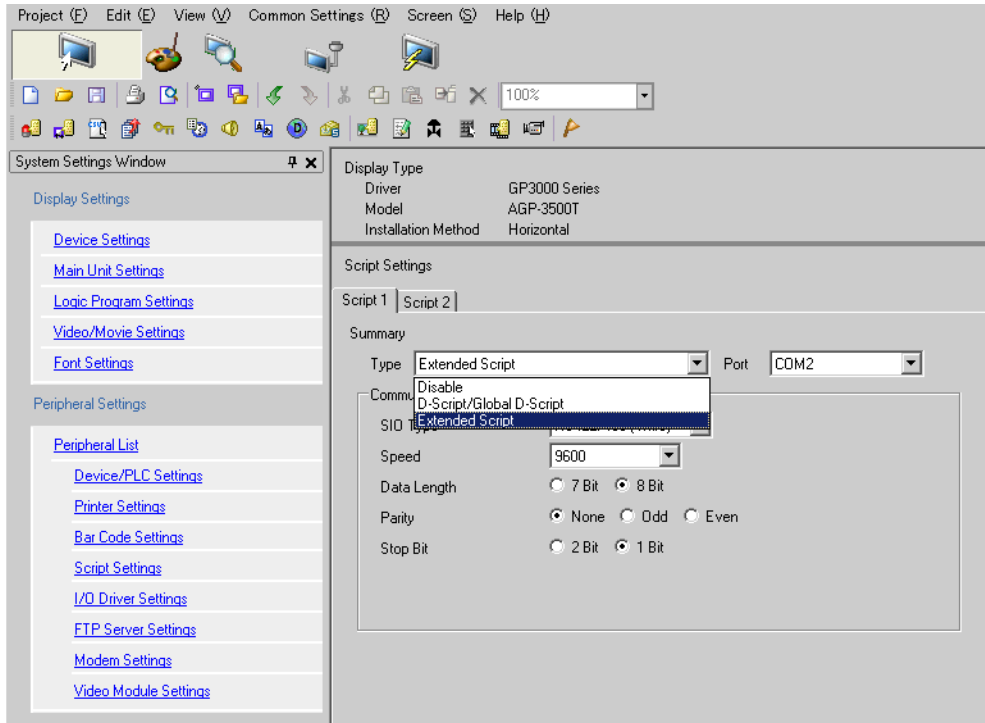
Create the script (Execution Expression). For more information about commands and functions, please refer to “21.13 Command List” (page 21-92) .



## 20.6.2 Procedure for Creating Extended Scripts

Open [System Settings (C)] from the [Project (F)] menu. Click on [Script Settings] and the following dialog box will be displayed.

When using an Extended Script, select [Type] as [Extended Script] and designate the connection port.



Open [Extended Script Settings (E)] from the [Common Settings (R)] menu.



Create the script (Execution Expression). For more information about commands and functions, please refer to “21 Program Instructions and Descriptions” (page 21-1) .

```

Execution Expression  Enlarge Execution Expression  Address Input
0001  [w:[PLC1]D00100]=[w:[PLC1]D00100]+1
0002  if ([w:[PLC1]D00100]==3)
0003  {
0004  {
0005      [w:[#INTERNAL]LS0008]=7
0006  }
0007  }
0008  endif
0009
0010
0011
0012
0013
0014
0015
0016
0017
    
```

### 20.6.3 Setting Procedure for Used-Defined Functions

Register a created script as a user-defined function and it can be used by other scripts. The registered function can be used by a D-Script, Global D-Script, or Extended Script.

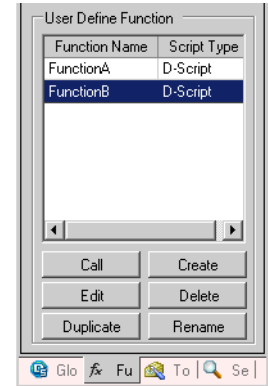
#### ■ Setting Procedure

When creating a new User-Defined Function

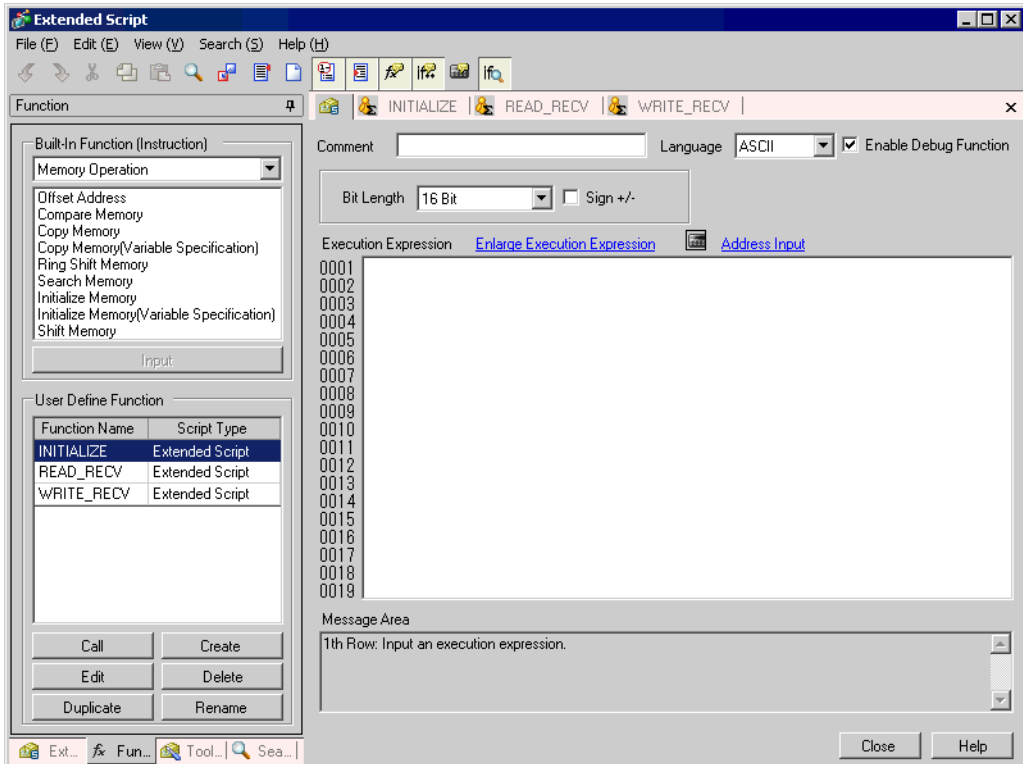
Click on [Create] and the User-Defined Function dialog box will be displayed.

When edited a previously registered User-Defined Function

Select the User-Defined Function you want to modify and click [Edit]. The registered user-defined function will appear.



Input the function name and create the script in the Execution field. Click [OK] and the user-defined function is registered.

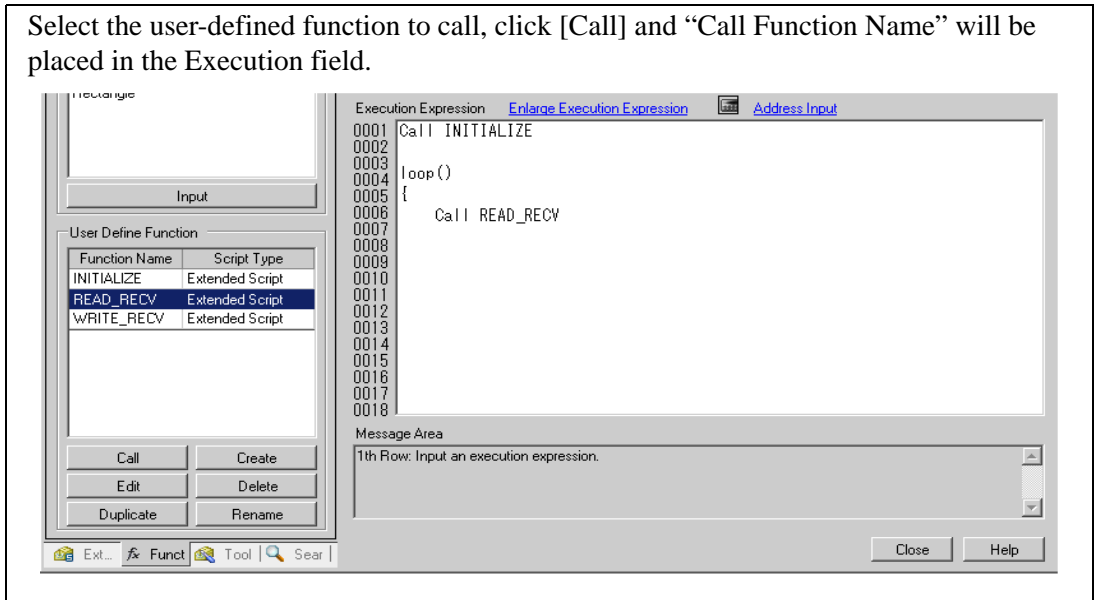


#### NOTE

- There are restrictions on which names you can use for Function Names. For more information, please refer to “20.9.3 Restrictions on User-Defined Functions” (page 20-57).



Select the user-defined function to call, click [Call] and “Call Function Name” will be placed in the Execution field.



**IMPORTANT**

- When a user-defined function calls another script, it cannot use functions created in an Extended Script in D-Scripts or Global D-Scripts.

## 20.7 Trigger Condition Setup

A created script can use any of the following 7 types of trigger conditions.

Setting		Description
Continuous Action		The script is triggered regularly.
Timer		The script is triggered after a designated time elapses.
Bit	Bit ON	When the GP detects the designated bit rise from 0 to 1, the script is triggered.
	Bit OFF	When the GP detects the falling edge of the designated bit, the script is triggered.
	Bit Change	When the GP detects the designated bit rise from 0 to 1 or fall from 1 to 0, the script is triggered.
Condition Expression	When Condition is Satisfied	When the GP detects true for a designated expression, the script is triggered.
	When Condition is not Satisfied	When the GP detects false for a designated expression, the script is triggered.

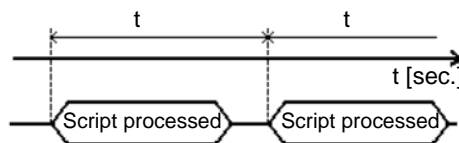
### 20.7.1 Continuous Action

Executes each display scan time.

### 20.7.2 Timer

#### ■ Timer

Each time the designated time elapses, the script is executed one time. The timer duration can be set from 1 to 32,767 seconds.

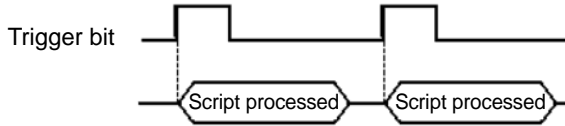


- NOTE**
- When setting the timer function's time, the time value will include the set time + display scan time error. Also, depending on the time taken to draw a screen item or to printout data, the timer function may be slowed. For more information about the Display Scan Time, please refer to “**■ Restrictions on the Trigger Bit**” (page 20-45).
  - When using D-Script, switching the screen will cause the timer function to restart counting from 0.

### 20.7.3 Bit

#### ■ Bit ON

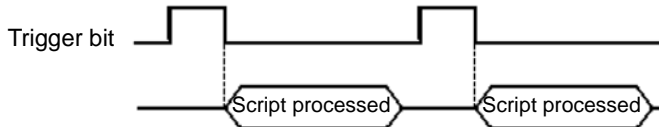
When the GP detects the designated bit address (trigger bit) rise from 0 to 1, the script is triggered.



- NOTE** • For the trigger bit’s ON/OFF, make sure to leave an interval longer than the communication cycle time or display scan time, whichever is longer. For more information about this function, please refer to “ ■ Restrictions on the Trigger Bit” (page 20-45) .

#### ■ Bit OFF

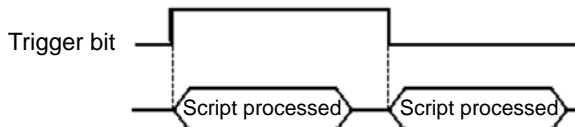
When the GP detects the designated bit address (trigger bit) fall from 1 to 0, the script is triggered.



- NOTE** • For the trigger bit’s ON/OFF, make sure to leave an interval longer than the communication cycle time or display scan time, whichever is longer. For more information about this function, please refer to “ ■ Restrictions on the Trigger Bit” (page 20-45) .

#### ■ Bit Change

When the GP detects the designated bit address (trigger bit) rise from 0 to 1 or fall from 1 to 0, the script is triggered.



- NOTE** • For the trigger bit’s ON/OFF, make sure to leave an interval longer than the communication cycle time or display scan time, whichever is longer. For more information about this function, please refer to “ ■ Restrictions on the Trigger Bit” (page 20-45) .

## 20.7.4 Condition Expression

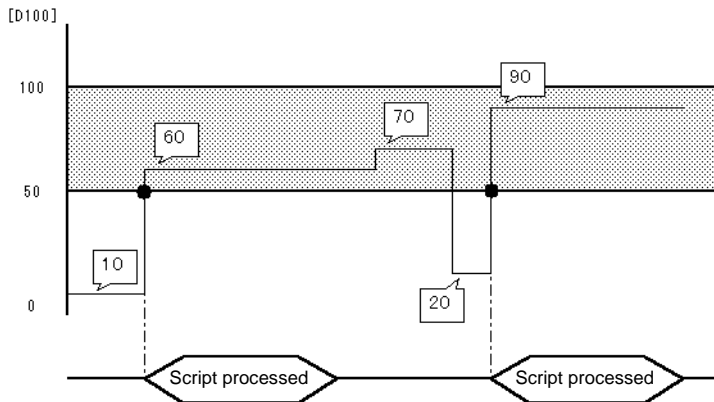
### ■ When Condition is Satisfied

When the GP detects true for a designated expression in a triggering program, the script is executed once.

e.g.: When the Trigger Condition is set to  $100 > [D100] > 50$ , the script will execute with the following timing.

[Not Satisfied] → [Satisfied] is detected, the script executes, and 70 is assigned to D100.

The script does not execute when [Satisfied] → [Satisfied].

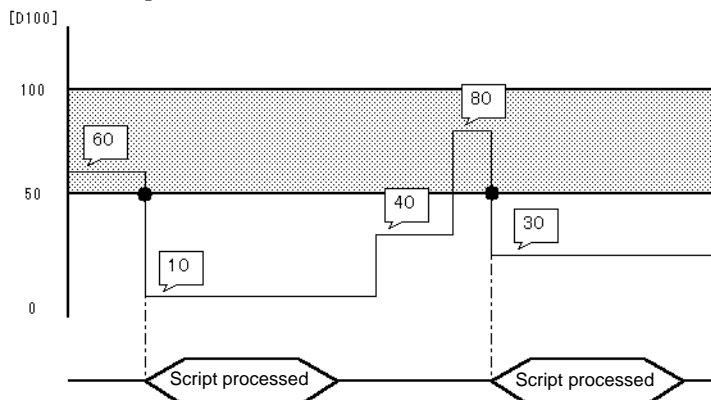


- NOTE** • For the Trigger Condition, make sure to leave an interval longer than the communication cycle time or display scan time, whichever is longer. For more information about this function, please refer to “ ■ Restrictions on the Trigger Bit” (page 20-45) .

### ■ When Condition is not Satisfied

When the GP detects false for a designated expression in a triggering program, the script is executed once.

e.g.: When the Trigger Condition is set to  $100 > [D100] > 50$ , the script will execute with the following timing. [Satisfied] → [Not Satisfied] is detected, the script executes, and 20 is assigned to D100. The script does not execute when [Not Satisfied] → [Not Satisfied].



- NOTE** • For the Trigger Condition, make sure to leave an interval longer than the communication cycle time or display scan time, whichever is longer. For more information about this function, please refer to “ ■ Restrictions on the Trigger Bit” (page 20-45) .

## ■ Restrictions on the Trigger Bit

---

- Make sure to leave an interval longer than the communication cycle time for executing write operations onto the connected device. When write operations onto the connected device are executed frequently by using the scan counter of GP internal special relay, communication errors or system errors may result.
- When the bit used for the D-Script Trigger Condition is set for “touch” and that bit turns OFF during D-Script processing, the timing used when pressing the touch area repeatedly can prevent the detection of the bit’s rise. The D-Script trigger will compare the previously read out value to the currently read out value to determine if the trigger is now “True”. However, during a single scan, the value that is stored in the bit address used during the Trigger operation is kept the same, even if the value is changed during execution. The new value is read out only after the next scan begins.

**Communication Cycle Time:** The communication cycle time is the time it takes to request and take in data from the GP unit to the PLC. It is stored in the internal device’s LS2037 as binary data. The unit is milliseconds (ms). There is an error of  $\pm 10$ ms.

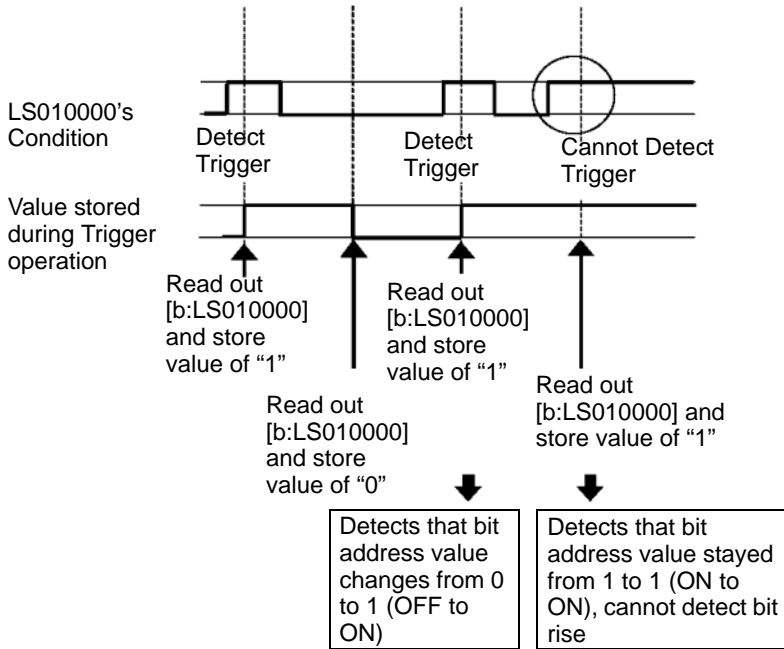
**Display Scan Time:** The display scan time is the time it takes to display/calculate 1 screen. It is stored in the internal device’s LS2036 as binary data. The unit is milliseconds (ms). There is an error of  $\pm 10$ ms.

e.g.: When Touch is used to turn ON the trigger bit (LS010000), and D-Script turns the value OFF:

Trigger Condition: Bit ON [#INTERNAL] LS010000

Execution Expression: clear ([b:[#INTERNAL] LS010000])

◆ D-Script Processing Timing Chart



As an example, if the D-Script touch timing is not used, and only detection is performed, the processing will be as follows.

Using an if ( ) statement to detect a trigger

The if statement is used to detect when the bit is set using a touch. Prior to performing the processing, the value is read out and compared.

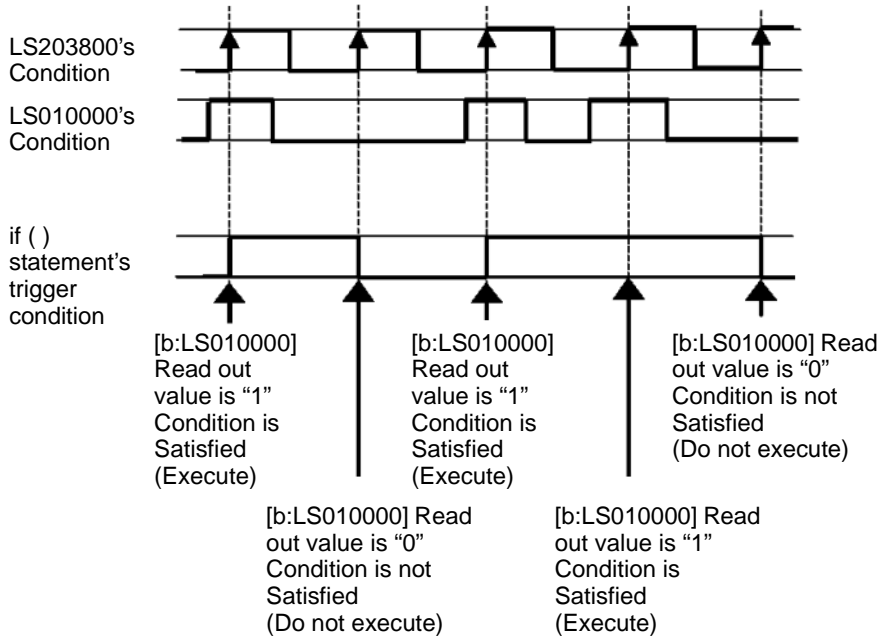
```

Trigger Condition: Bit ON ([#INTERNAL]LS203800 *1])
Execution Expression: if ([b:[#INTERNAL]LS010000]=1)
{
clear ([b:[#INTERNAL]LS010000])
:
:
}
    
```

\*1 GP's internal counter. The counter increments each time the Part set on the display screen processes.

When the above type of D-Script is created, even if touch input is done repeatedly, the tag scan is performed as shown in the following timing chart. Here, each tag scan value is read out, the condition is compared and, regardless of the previous value, if it agrees with the condition, the processing is performed.

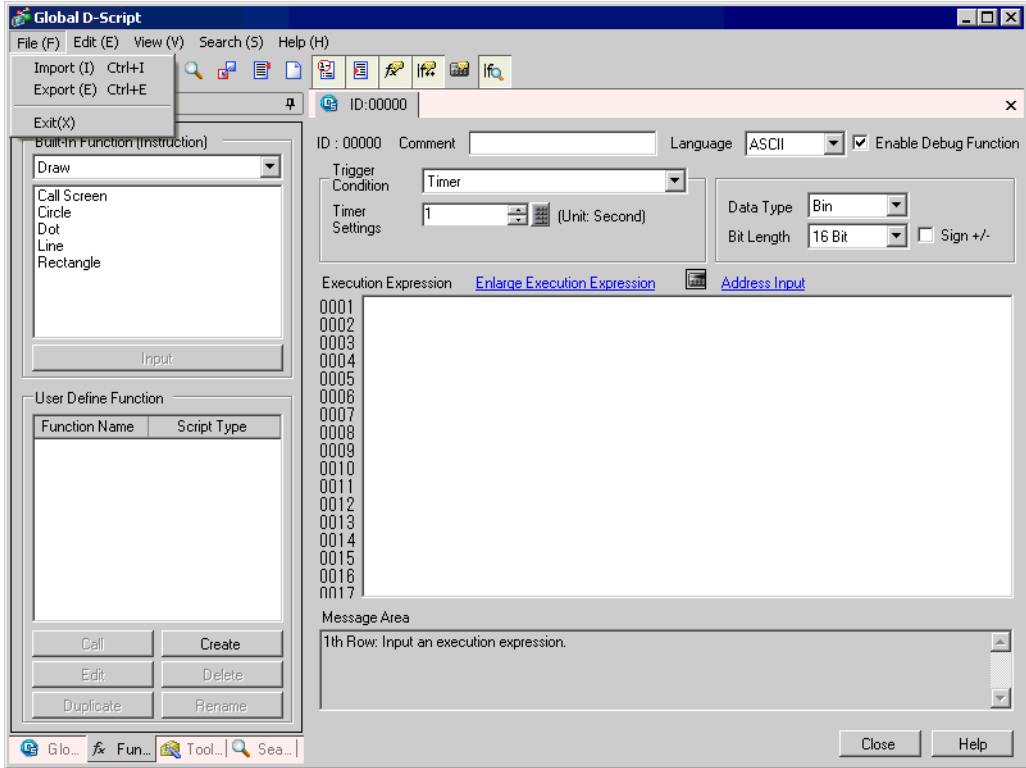
◆ D-Script Processing Timing Chart





## 20.8 Settings Guide

### 20.8.1 Common Settings Guide for D-Script


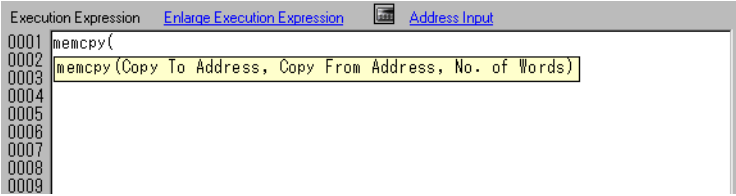



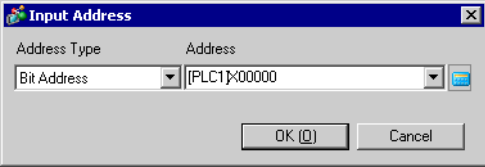
The following is the D-Script Global D-Script dialog box. For extended script, the ID and trigger settings do not exist, but the other settings are the same as below.





Setting	Description
Export	This can be selected from the File menu. Export writes a created script to a text file (.txt) which can then be imported into other scripts.
Import	This can be selected from the File menu. Import reads in an exported script (text file).
Row No. 	Shows the row number to the right of the program.
Auto Indent Control 	Automatically indents statements as below. <pre>                     Execution Expression                     0001 if ([b:[PLC1]D00000.0]==1)                     0002 {                     0003     if ([b:[PLC1]D00001.0])                     0004     {                     0005         [b:[PLC1]D00002.0]                     0006     }                     0007     endif                     0008 }                     0009 endif                     0010                     0011                     0012                     </pre>

Continued

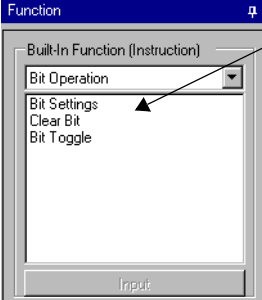
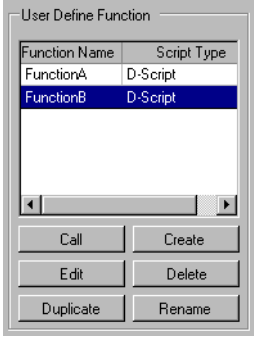
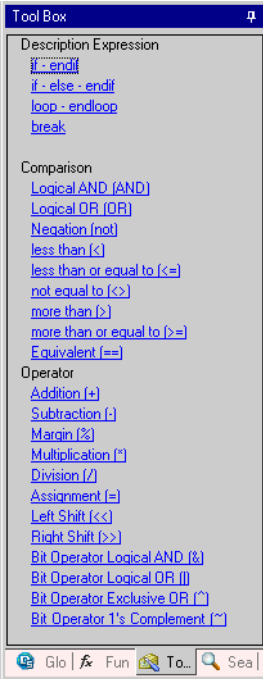


Setting	Description
<p>Function Input Assistance </p>	<p>When the function and the initial bracket “(” are inputted as below, the function’s format gets displayed.</p> 
<p>Auto Syntax Complement </p>	<p>When “if” or “loop” is inputted from the keypad, the remaining syntax is automatically placed.</p>
<p>Address Input </p>	<p>When creating a script, input a left-hand square bracket ( [ ] ) and the [Input Address] dialog box will automatically display.</p>
<p>Address Input Dialog </p>	<p>Displays the following input box for inputting an address.</p>  <p>There are 3 types of device that can be chosen.  For more information about the internal device, please refer to</p> <ul style="list-style-type: none"> <li>☞ “A.1.2 Communicating with a Device/PLC without Burdening it (Direct Access Method)” (page A-3)</li> <li>☞ “A.1.3 Communicating with Unsupported Devices/PLCs (Memory Link Method)” (page A-5)</li> <li>• Memory Link only <ul style="list-style-type: none"> <li>#INTERNAL : Designate a GP internal device’s USR.</li> <li>#MEMLINK : Designate a Memory Link device.</li> </ul> </li> <li>• Connected device only (except Memory link) <ul style="list-style-type: none"> <li>Each device name (PLC1, etc.) : Designate each device.</li> <li>#INTERNAL : Designate a GP internal device’s USR.</li> </ul> </li> <li>• Memory Link and connected device <ul style="list-style-type: none"> <li>Each device name (PLC1, etc.) : Designate each device.</li> <li>#INTERNAL : Designate a GP internal device’s USR.</li> <li>#MEMLINK : Designate a Memory Link device.</li> </ul> </li> </ul>

Continued

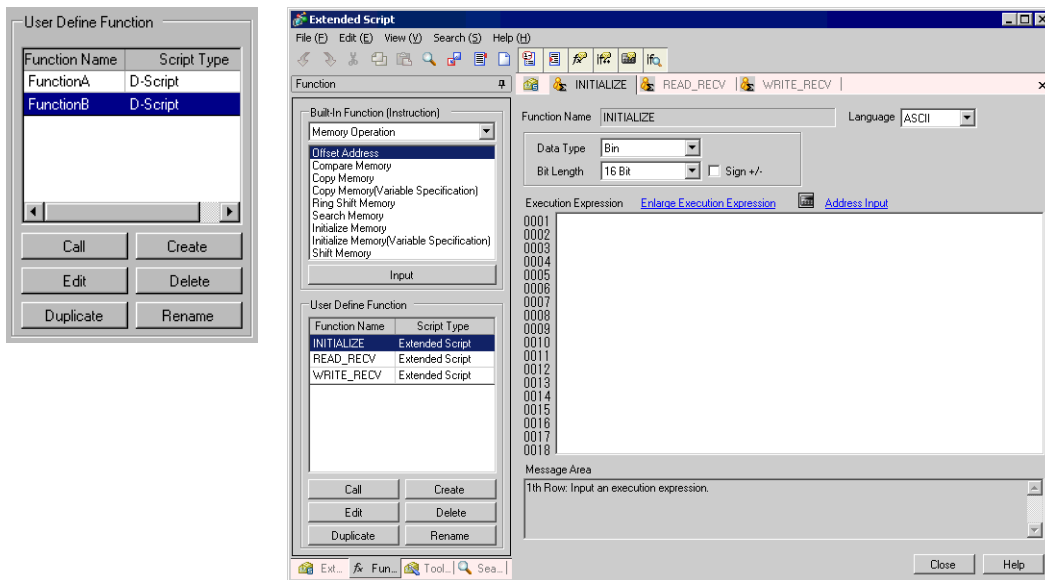
Setting	Description
Address Input Dialog 	<p><b>IMPORTANT</b></p> <ul style="list-style-type: none"> <li>In the scripts, please do NOT set any passwords, etc., that begin with "0". All numeric values beginning with "0" will be processed as Oct (base-8) data.</li> <li>How to describe different input data formats e.g.:               <ul style="list-style-type: none"> <li>DEC (Base-10) : Non-zero starting value e.g.: 100</li> <li>Hex (Base-16) : Value starting with 0x e.g.: 0x100</li> <li>Oct (Base-8) : Value starting with 0x e.g.: 0100</li> </ul> </li> <li>Example of operation with different data formats using the operator AND (Hex and BCD)               <ul style="list-style-type: none"> <li>Hex only 0x270F &amp; 0xFF00      Result: 0x2700</li> <li>BCD and Hex 9999 &amp; 0xFF00      Result: 0x9900</li> </ul> </li> </ul>
Auto Syntax Analysis 	<p>Checks the syntax during script creation. The check results will be displayed in the bottom portion of the window.</p> <div data-bbox="580 890 1061 1035" style="border: 1px solid gray; padding: 5px;"> <p>Message Area</p> <p>5th Row: A statement is required in { } of an 'if' statement. 5th Row: The expression is incorrect.</p> </div>
ID	<p>Scripts are managed by an ID number. When creating multiple scripts with different trigger conditions, set a value from 0 to 65,535.</p>
Comment	<p>Input a comment for the script.</p>
Language	<p>Choose from [ASCII], [Japanese], [Chinese (Traditional)], [Chinese (Simplified)], or [Korean].</p>
Enable Debug Function	<p>Set whether or not to enable the debug function. If the <code>_debug</code> function exists in the body of the script, the <code>_debug</code> function will execute. For more information about this function, please refer to "21.7.1 Debug Function" (page 21-61)</p>
Trigger Condition	<p>Set the trigger condition that will cause the script to execute. For more information about this function, please refer to "20.7 Trigger Condition Setup" (page 20-42). Extended scripts do not have the trigger condition setting.</p>
Data Type	<p>Set the data format for the script to Bin or BCD. For Extended Scripts, Bin is fixed.</p>
Bit Length	<p>Set the data length for the script to 16 bit or 32 bit.</p>
Sign +/-	<p>Select this when you want to insert negative numbers. This can only be set when the data type is Bin.</p>
Execution Expression	<p>The contents of the script.</p>

Continued

Setting	Description
<p>Built-in Function (Instruction)</p>	<p>Commands and functions that can be used in the script can be easily selected from icons, saving the time taken to input them.                      For more information about commands and functions that can be used, please refer to                      🖱️ “21.13 Command List” (page 21-92)                      Built-in Functions</p>  <p>Select a category from the upper pull-down menu, and the related functions will appear in the bottom area.                      Click [Input] when the function has been selected, and the corresponding settings dialog box will be displayed.</p>
<p>User Define Function</p>	<p>Register a created script as a user-defined function and it can be used by other scripts.</p> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>• For more details about user-defined functions, please refer to “20.8.2 User Define Functions Settings Guide” (page 20-52) .</li> </ul> 
<p>Tool Box</p>	<p>Commands that can be used in the script can be easily selected by clicking them, saving the time taken to input them.                      Also, you can select commands such as search and position text used in scripts.                      For more information about commands that can be used, please refer to “Chapter 21 Program Instructions and Descriptions” (page 21-1).</p> 

Continued

## 20.8.2 User Define Functions Settings Guide



Setting	Description
Call	Call a created function. Select the function to call, click [Call] and “Call Function Name” will be placed in the Execution field.
Create	Create a new function. Click on [Create] and a dialog box appears for the Function Name to create.
Edit	Edit the created function. Select the function to edit, click on [Edit] and the [D-Script Function] dialog box appears.
Delete	Delete the created function. Select the function to delete and click [Delete].
Duplicate	Copy the created function. Select the function to copy, click [Duplicate], and a dialog box will appear for you to save the duplicate as a new function name.
Rename	Change the name of the created function. Click on [Rename] and the Rename Function dialog box will be displayed.

## 20.9 Restrictions

### 20.9.1 D-Script/Global D-Script Restrictions

- As a guide for D -Script programming, three addresses occupy the same amount of memory as one Part. The maximum number of addresses available for a D-Script is 255. However, try to use the fewest possible addresses, since the more devices that are used, the slower the response.
  - The Convert Address command in the Utility menu of the Project Manager cannot convert addresses used in D-Script. Open the D-Script Editor to change these address
  - When the Connected Device Type setting is changed via the Save As window of the Project Manager-Project menu, the address used by the D-Script cannot be changed. Please use the D-Script Editor to change these addresses.
  - D-Script cannot be used for floating-point arithmetic (float variable, real variable). It cannot be used for specifying structural variables, either. It, however, can be used to specify each member of a structure.
  - The size of a D-Script affects the Display Scan Time. Note that using a large number of addresses may significantly degrade the performance of the program.
  - When calling a function from a function, the maximum number of levels (nesting) is 9. Do NOT create more than 9.
  - Up to 9 levels of nested Functions can be recalled.
  - Up to 254 Functions can be created.
- ◆ Depending on the devices specified for trigger conditions, the D-script operations activated by a trigger after the screen changes are as follows:

Trigger Condition	Connection Device or GP Internal Device (LS/USR)				Memory Link			
	Current Value or Condition	Bit "0"	Bit "1"	Condition is not Satisfied	Condition is Satisfied	Bit "0"	Bit "1"	Condition is not Satisfied
Leading edge of bit	Disable	Enable	–	–	Disable	Disable	–	–
Falling edge of bit	Enable	Disable	–	–	Disable	Disable	–	–
Bit Change	Enable	Enable	–	–	Disable	Disable	–	–
Timer setting	Disable	Disable	Disable	Disable	Disable	Disable	Disable	Disable
Detecting true	–	–	Disable	Enable	–	–	Disable	Enable
Detecting false	–	–	Enable	Disable	–	–	Enable	Disable

Enable: Operation is performed right after the screen is changed, or the power is turned ON.

Disable: Operation is not performed right after the screen is changed, or the power is turned ON.

- When the timer is operating, the timer starts counting right after the screen changes.
- When using Global D-Script, the operations mentioned above are performed only when the GP's power is turned ON. When the GP screen changes, however, the operation mentioned above will not be performed and the monitor operates using the trigger conditions that have been set.
- When a Global D-Script includes a timer, the timer starts counting right after the GP's power cord is connected.


**NOTE** • Do not use the touch panel key to set the trigger bit or to operate the start bit in a program. The timing of the touch input may not be correct, resulting in the bit being improperly entered.

- ◆ When a value is assigned to an address for switching screens while a D-Script command is being executed, the screen switching operation is processed after all D-Scripts have been processed.

e.g.:

ID	00000				
Data Type	Bin	Data Length	16 Bit	Sign +/-	None
Trigger	Leading Bit([b:M0000])				
	[w:[PLC1]D0100]=0	// (1)			
	[w:[#INTERNAL]LS0008]=30	// (2)	Switches to Base screen No. 30		
	[w:[PLC1]D0101]=1	// (3)			
	[w:[PLC1]D0102]=2	// (4)			

When the above D-Script is executed, processing of the screen switching is performed after (3) and (4) have been processed.

- ◆ When data used for D-Scripts is set by touch from the GP, detect that all the data has been written and then execute D-Scripts.
- ◆ Restrictions Specific to Global D-Script
  - When the GP's power is turned ON, the actions shown in the table on the previous page are performed. At the screen change, the above table is not applied, and the trigger conditions are continuously monitored.
  - Global D-Script operation is suspended during screen changes or other GP operations.
  - After the GP's power is turned ON, Global D-Script actions are not performed until all data reads are completed for the initial screen. However, after the initial screen changes, Global D-Script actions may be performed before the data reads are completed.
  - The maximum number of devices in Global D-Script is 255. When this number exceeds 256, the D-Script does not function. Since these devices always read data regardless of the screens, be sure to use only the minimum number of device settings in your D-Script. Otherwise, operation performance can be degraded.
  - The maximum number of Global D-Scripts available is 32. The currently used function also counts as one Global D-Script. When the number of the Global D-Scripts reaches 32, any new Global D-Scripts are ignored.
- ◆ Restrictions for SIO Port Operation
  - Addresses designated in the Send/Receive functions will not be added to the count of D-Script addresses.
  - The Control is a write-only variable, while Status and Received Data are read-only variables. Do not try reading out the Control variable or writing data into the Status variable, as doing so will cause the operation to fail.
  - Create independent D-Scripts (or functions) for Send and Receive operations. For more information about the flow charts of data transfers, please refer to  " ■ Flow Chart" (page 20-24)
  - The valid range of the LS device that can store data for Send/Receive functions is the User area (LS20 to LS2031 and LS2096 to LS8191).

- In [System Settings] - [Script Settings], when no [D-Script/Global D-Script] is set, the 13th bit of LS2032 turns ON when the readout of the Send function, Receive function, Control, Status variable, and Received Data Size is executed. For information on the structure of the internal device,
  - ☞ “A.1.4.3 Special Relay” (page A-17)
- When using the Send/Receive functions, set the bit length of the D-Script to 16 bits. Note that the operation will fail if the bit length is set to 32 bits.
- The size of the Send buffer is 2048 bytes, while the Receive buffer is 8192 bytes. The ER signal (output) RS signal (output) is turned OFF after at least 80% of the Receive buffer is full of received data.

### ◆ Limitations on BCD format operations

If a value which cannot be converted into BCD format is found during operation, the program stops running.

These values include A to F in hexadecimal format. Do not use such values.

If the program stops due to non-BCD values, bit 7 in common relay information (LS2032) in the GP turns ON. This bit does not turn OFF until the GP is turned OFF or goes offline.

e.g.:

$$[w:[PLC1]D0200]=[w:[PLC1]D0300]<<2)+80$$

If D300 is 3, shifting two bits to the left results in 0x000C, which cannot be converted into BCD format interrupts program execution.

$$[w:[PLC1]D0200]=[w:[PLC1]D0300]<<2$$

If D300 is 3, shifting two bits to the left results in 0x000C. Unlike the above example, 0x000C is the result of the operation to be stored in the memory, and does not cause the program to stop.

### ◆ Limitations of zero operations

If you divide by zero in division (/) and remainder (%) operations, execution will stop. Do not divide by zero.

If the program stops due to non-BCD values, bit 8 in common relay information (LS2032) in the GP turns ON. This bit does not turn OFF until the GP is turned OFF or goes offline.

### ◆ Notes on delay during assign operation

Using a device address in an assign operation may cause write delay because the GP has to read the address data from the connected device. Consider the following:

e.g.:

$$[w:[PLC1]D0200]=[w:[PLC1]D0300]+1 \dots (1)$$

$$[w:[PLC1]D0201]=[w:[PLC1]D0200]+1 \dots (2)$$

Statement (1) assigns (D0300+1) into D0200. However, in statement (2), the result of statement (1) has not been assigned in D0200 because of time-consuming communication with the device/PLC. In such cases, program so that the result of statement (1) is stored in the LS area before it is executed, as shown below.

$$[w:[\#INTERNAL]LS0100]=[w:[PLC1]D0300]+1$$

$$[w:[PLC1]D0200]=[w:[\#INTERNAL]LS0100]$$

$$[w:[PLC1]D0201]=[w:[\#INTERNAL]LS0100]+1$$

## 20.9.2 Extended Script Restrictions

- For Device Addresses, only the LS Area and USR Area (Extended User Area) can be used.
- The temporary addresses of D-Scripts and Global D-Scripts are managed independently from the temporary address of Extended Scripts. Therefore, changes made to the temporary addresses of D-Scripts and Global D-Scripts will not be reflected in the temporary address of Extended Scripts.
- You can call user-defined functions created with D-Script/Global D-Script, but if you access a device address outside of the internal device inside the function, it may not operate normally. Also, when transferred (during the creation of data for the GP), user-defined functions will be created independently for D-Scripts, Global D-Scripts, and Extended Scripts.
- When calling a function from a function, the maximum number of levels (nesting) is 9.
- Up to 254 functions can be called. (The number of functions available with “Call” is 254.)
- Extended Script does not affect the number of tags count.
- Functions supported only by Extended Script, for example string operations, will not function if called with D-Script or Global D-Script.
- The available data format is Bin. BCD data format is disabled.
- The size of the Send buffer is 2048 bytes, while the Receive buffer is 8192 bytes. The CTS line is turned OFF after at least 80% of the Receive buffer is full of received data.
- General protocol and Extended Script cannot be selected simultaneously. The following table shows more information about combinations.

Extended SIO Setting	D-Script/ Global D-Script Extended SIO function for Extended Script	Extended SIO function for Extended Script
General protocol	OK: Operation enabled	Invalid: Operation disabled
Extended Script	Invalid: Operation disabled	OK: Operation enabled

- Notational conventions for the character string setting  
When using character strings with “\_ strset ( )” and other functions, enclose the character string in double quotation marks (“”). To display double quotation marks in the character strings, append the “\” symbol and express as [“”]. There is no way to represent a single “\” symbol. When necessary, use the character code format setting (\_strset (databuf0, 92)).  
e.g.:

"ABC\DEF" → ABC"DEF

"ABC\DEF" → ABC\DEF

"ABC\\DEF" → ABC"DEF

"ABC\\DEF" → ABC\\DEF



- ◆ The following table shows the sizes of the dedicated buffers for Extended SIO, databuf0, databuf1, databuf2, and databuf3.

Buffer	Buffer Name	Character Size
Data Buffer0	databuf0	1K Byte
Data Buffer1	databuf1	1K Byte
Data Buffer2	databuf2	1K Byte
Data buffer 3	databuf3	1K Byte

### 20.9.3 Restrictions on User-Defined Functions

- Portions of the commands that can be used differ with each script. When using commands, please refer to “21.13 Command List” (page 21-92) .
- For the function name, you may use any English letters or the underscore character “\_.” (However, the function name must begin with an alphanumeric character.)
- Do not use the following as Function Names.

and	b_call	Bcall	_bin2hexasc	break	Call
_CF_delete	_CF_dir	_CF_read	_CF_read_csv	_CF_rename	_CF_write
clear	databuf0	databuf1	databuf2	databuf3	_decasc2bin
_dlcopy	dsp_arc	dsp_circle	dsp_dot	dsp_line	dsp_rectangle
else	endif	fall	_hexasc2bin	if	IO_READ
IO_READ_EX	IO_READ_WAIT	IO_WRITE	IO_WRITE_EX	loop	_memcmp
memcpy	_memcpy_EX	memring	_memsearch	memset	_memset_EX
_memshift	not	or	return	rise	rise_expr
set	_strcat	_strlen	_strmid	_strset	timer
toggle	_wait				

## 20.9.4 Notes on Operation Results

---

### ■ Overflowing Digits

Overflowing digits resulting from operations are rounded.

When performing an operation on unsigned 16-bit data:

- $65535 + 1 = 0$  (Produces overflowing digits)
- $(65534 * 2) / 2 = 32766$  (Produces overflowing digits)
- $(65534 / 2) * 2 = 65534$  (Does not produce overflowing digits)

### ■ Difference of Residual Processing

The result of a residual processing depends on whether the left and right sides are signed or unsigned.

- $-9 \% 5 = -4$
- $9 \% -5 = 4$

### ■ Rounded Decimal Places

Decimal places resulting from a division are rounded.

- $10 / 3 * 3 = 9$
- $10 * 3 / 3 = 10$

### ■ Notes on Operating BCD data

A BCD-data operation which produces overflowing digits does not give the correct result.

## 20.9.5 Errors

The following error message is displayed when a Script is configured incorrectly. The error will be displayed on the bottom of the GP screen.

Error codes are written to the LS91XX addresses. The number written in the error code area will be the number portion following RAAA in the table below. (For example, when error RAAA130 occurs, '130' will be written.)

List of Script Error Codes

D-Script (Error Address=LS9120)	Global D-Script (Error Address=LS9110)	Extended Script (Error Address=LS9100)
–	RAAA130	RAAA140
Unused	Global D-Script Error. (The Total No. of Global D-Scripts exceeds the maximum of 32.)	Extended D-Script Error (The total no. of functions exceeds the max of 255.)
–	RAAA131	–
Unused	Global D-Script Error. (The total no. of devices exceeds the maximum of 255.)	Unused
RAAA120	RAAA132	RAAA141
D-Script Error (The specified function does not exist or the function has an error.)	Global D-Script Error (The specified function does not exist or the function has an error.)	Extended D-Script Error (The specified function does not exist or the function has an error.)
RAAA121	RAAA133	RAAA142
D-Script Error (These functions are nested to 10 levels or more.)	Global D-Script Error (These functions are nested to 10 levels or more.)	Extended D-Script Error (These functions are nested to 10 levels or more.)
RAAA122	RAAA134	RAAA143
D-Script Error (An expression exists, that is not supported by this version.)	Global D-Script Error (An expression exists, that is not supported by this version.)	Extended D-Script Error (An expression exists, that is not supported by this version.)
RAAA123	RAAA135	RAAA144
D-Script Error (The SIO operation function is used in a condition where no Device/PLC has been set.)	Global D-Script Error (The SIO operation function is used in a condition where no Device/PLC has been set.)	Extended D-Script Error (The SIO operation function is used in a condition where no Device/PLC has been set.)
RAAA124	RAAA136	RAAA145
The D-Script has an error.	The Global D-script has an error.	The Extended D-Script has an error.

---

# *Memo*