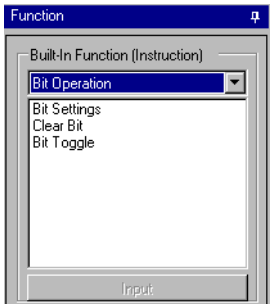
21 Program Instructions and Descriptions

In this chapter, we will talk about instructions and descriptions used in scripts of GP-Pro EX. For information about script programming, see "Chapter 20 Programming Scripts (Programming that does not use Parts)" (page 20-1).

21.1 Bit Operation

Bit Operation	Function Summary
	Bit Settings ☞ "21.1.1 Bit Settings" (page 21-2) Changes the specified bit address from 0 to 1.
	Clear Bit ☞ "21.1.2 Clear Bit" (page 21-2) Changes the specified bit address from 1 to 0.
	Bit Toggle ☞ "21.1.3 Bit Toggle" (page 21-2) Changes the specified bit address from 1 to 0 or from 0 to 1.

21.1.1 Bit Settings

Item	Description
Summary	Changes the specified bit address from 0 to 1.
Format	set( )

Example expression:

set ([b:[#INTERNAL]LS010000])

In the above example, the 00th bit of LS0100 is changed from 0 to 1.

21.1.2 Clear Bit

Item	Description
Summary	Changes the specified bit address from 1 to 0.
Format	clear()

Example expression:

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

In the above example, the 00th bit of LS0100 is changed from 1 to 0.

21.1.3 Bit Toggle

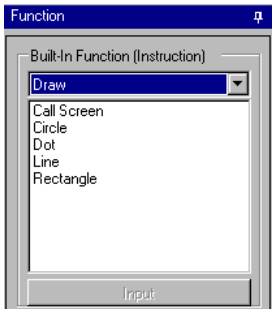
Item	Description
Summary	Changes the specified bit address from 1 to 0 or from 0 to 1.
Format	toggle ( )

Example expression:

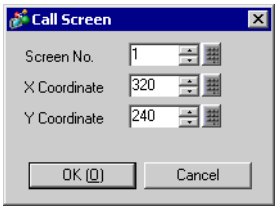
toggle ([b:[#INTERNAL]LS 010000])

In the above example, the 00th bit of LS0100 is changed from 1 to 0 or from 0 to 1.

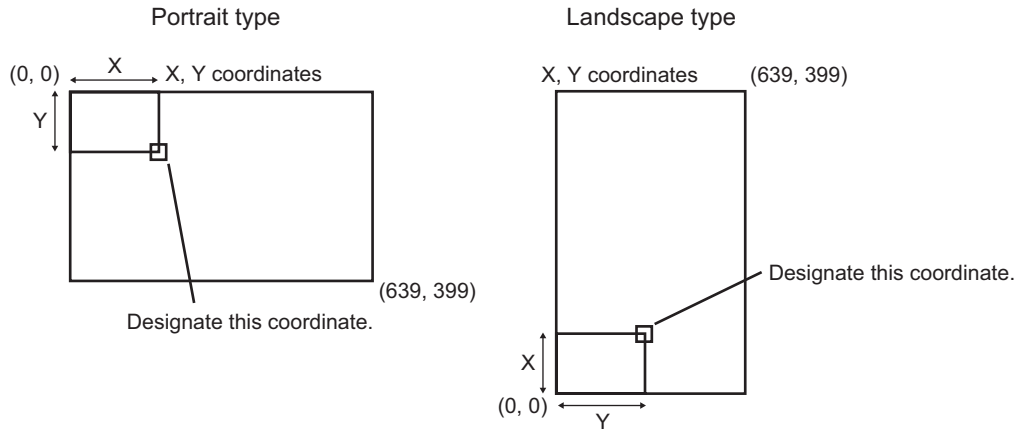
21.2 Draw

Draw	Function Summary
	Call Screen ☞ "21.2.1 Call Screen" (page 21-3) Calls the screen (base screen) with the designated screen number. It cannot be used in an Extended Script.
	Circle ☞ "21.2.2 Circle" (page 21-4) Draws the designated circle.
	Dot ☞ "21.2.3 Dot" (page 21-5) Draws the designated dot.
	Line ☞ "21.2.4 Line" (page 21-5) Draws the designated line.
	Rectangle ☞ "21.2.5 Rectangle" (page 21-6) Draws the designated rectangle.

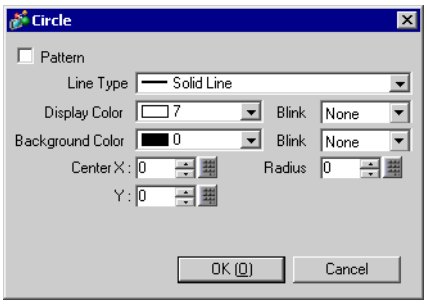
21.2.1 Call Screen

Item	Description
Summary	This function is used to call up a previously registered Library Item. The designated screen (Base screen) will be called up at the designated X,Y coordinates. It cannot be used in an Extended Script.
Format	b_call (Screen No., X Coordinate, Y Coordinate)  **NOTE** • Designate the called screen's center coordinate with the X coordinate and Y coordinate.

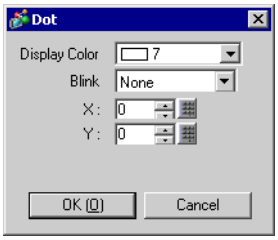
Coordinate Position



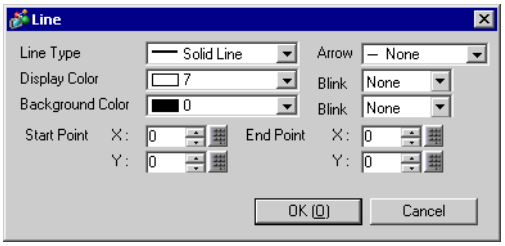
21.2.2 Circle

Item	Description
Summary	Draws a circle at the designated point. When you put a check mark next to the [Pattern] box, a filled circle will be drawn. Select and enter the line type (or fill pattern when selecting a pattern), color attributes, center coordinates, and radius value. As well, center coordinates and radius can be set indirectly.
Format	dsp_circle (X Coordinate, Y Coordinate, Radius, Display Color Blink + Display Color, Background Color Blink + Background Color, Line Type)  **NOTE** • When both black and Blink are set, the background color will become transparent.

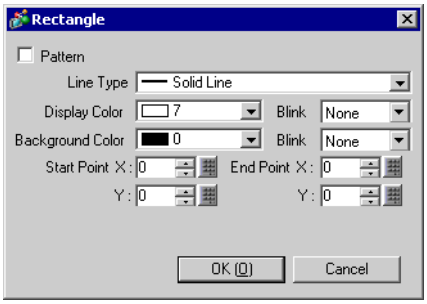
21.2.3 Dot

Item	Description
Summary	Draws a dot at the designated point. Designate the X,Y coordinates, and display color.
Format	dsp_dot (X Coordinate, Y Coordinate, Blink + Display Color)  **NOTE** • When both black and Blink are set, the background color will become transparent.

21.2.4 Line

Item	Description
Summary	Draws a line at the designated position. Designate the line's type, color attributes, and start and end coordinates.
Format	dsp_line (Start Point X Coordinate, Start Point Y Coordinate, End Point X Coordinate, End Point Y Coordinate, Display Color Blink + Display Color, Background Color Blink + Background Color, Line Type and Arrow)  **NOTE** • When both black and Blink are set, the background color will become transparent.

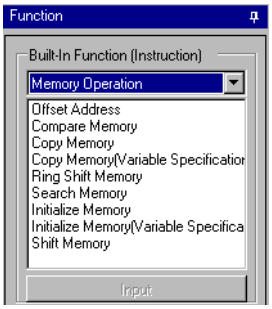
21.2.5 Rectangle

Item	Description
Summary	Draws a rectangle at the designated position. When you put a check mark next to the [Pattern] box, a filled rectangle will be drawn. Select and enter the line type (or fill pattern when selecting a pattern), color attributes, and start and end coordinates.
Format	dsp_rectangle (Start Point X Coordinate, Start Point Y Coordinate, End Point X Coordinate, End Point Y Coordinate, Display Color Blink + Display Color, Background Color Blink + Background Color, Pattern and Line Type)  **NOTE** • When both black and Blink are set, the background color will become transparent.

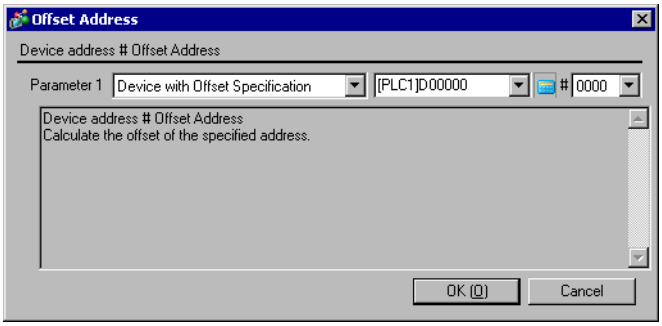
IMPORTANT

- When color-coding the draw functions, set the color codes from 0 to 255. If you set E1 to E12 and save the script, it will output an error.

21.3 Memory Operation

Memory Operation	Function Summary
	Offset Address ☞ "21.3.1 Offset Address" (page 21-8) Designates an address offset.
	Compare Memory ☞ "21.3.2 Compare Memory" (page 21-9) Compares two blocks of data at the specified positions (offset), and writes the comparison result to the storage address.
	Copy Memory ☞ "21.3.3 Copy Memory" (page 21-11) Copies device memory in one operation.
	Copy Memory (Variable Specification) ☞ "21.3.4 Copy Memory (Variable Specification)" (page 21-14) Copies device memory in one operation. The source (copy from) address, destination (copy to) address, and number of addresses can be modified.
	Ring Shift Memory ☞ "21.3.5 Ring Shift Memory" (page 21-15) Ring-shifts the data in memory by the designated number of word blocks.
	Search Memory ☞ "21.3.6 Search Memory" (page 21-17) Performs a data search in block units, and returns (saves) the search result to the specified storage address.
	Initialize Memory ☞ "21.3.7 Initialize Memory" (page 21-20) Initializes all devices at once.
	Initialize Memory (Variable Specification) ☞ "21.3.8 Initialize Memory (Variable Specification)" (page 21-21) Initializes all devices at once. The top address, set data, and number of addresses can be modified.
	Shift Memory ☞ "21.3.9 Shift Memory" (page 21-22) Shifts block units up.

21.3.1 Offset Address

Item	Description
Summary	Offset Addresses can be designated. Only temporary Word Addresses can be designated for offset value storage Addresses.
Format	[Device Address] # [Offset Address]  Constant Input Ranges

Data Type	Constant Input	
	Min Value	Max Value
Bin16	0	65535
Bin32	0	4294967295
Bin16+/–	–32768	32767
Bin32+/–	–2147483648	2147483647
BCD16	0	9999
BCD32	0	99999999

Example expression 1:

[w:[PLC1]D0200]=[w:[PLC1]D0100]#[t:0000]

In the above example, when [t:0000]'s value is 2, the value stored in D0102 will be offset to D0200.

Example expression 2:

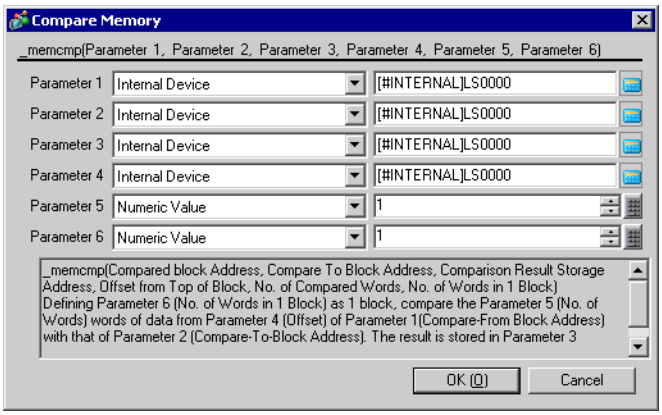
[w:[PLC1]D0100]#[t:0000]=30

In the above example, when [t:0000]'s value is 8, 30 will be offset to D0108.

IMPORTANT

- Word Addresses used in the offset address format are not counted as D-Script Addresses.
- Data from a device designated by an offset address is not continuously read out from the connected device. It is read out every time the D-Script is performed. When an error occurs during the readout, the read-out value is treated as "0". Also, Bit 12 of the GP unit's internal special relay LS2032 turns ON. When data read is completed normally, Bit 12 will be turned OFF.
- If the operation result exceeds 16 bits (Max. Value: 65535), Bit 1 to Bit 15 are treated as valid bits and Bit 16 and other bits are discarded.

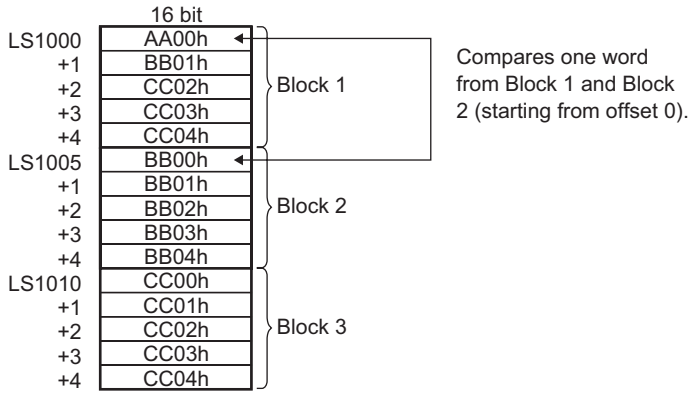
21.3.2 Compare Memory

Item	Description
Summary	Compares two blocks of data at the specified positions (offset), and writes the comparison result to the storage address. The following values are stored as the comparison result: When the values are equal: “0”. When the target data is larger than the original data: 1. When the target data is smaller than the original data: 2. When an error occurs, the error status value is written to LS9152.
Format	_memcmp ([Compared block Address], [Compare To Block Address], [Comparison Result Storage Address], Offset from Top of Block, No. of Compared Words, No. of Words in 1 Block)  Parameter 1: Internal Device Parameter 2: Internal Device Parameter 3: Internal Device Parameter 4: Numeric Value (0 to 639), Internal Device, Temporary variable Parameter 5: Numeric Value (1 to 640) Parameter 6: Numeric Value (1 to 640) Data to be stored 0: Match 1: Source is smaller than Target (Source < Target) 2: Source is larger than Target (Source > Target)

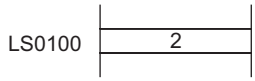
Example expression 1:

_memcmp ([w:[#INTERNAL]LS1000], [w:[#INTERNAL]LS1005], [w:[#INTERNAL]LS0100], 0, 1, 5)

(Compares one word from Block 1 and Block 2 (starting from offset 0) and saves the comparison result in LS0100).



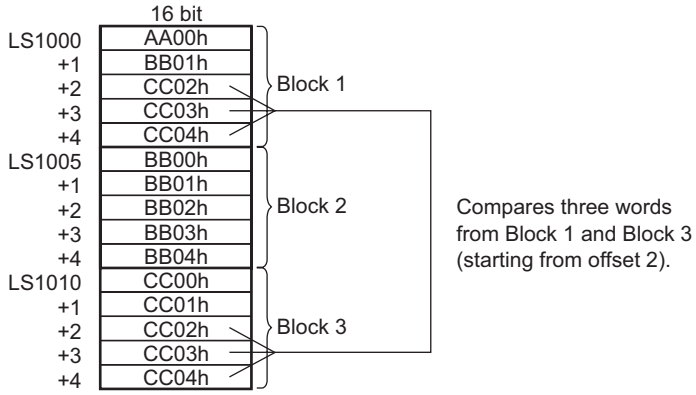
Since the source value is smaller than the target value, the comparison result “2” is stored in LS0100.



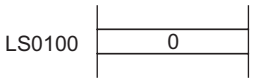
Example expression 2:

_memcmp ([w:[#INTERNAL]LS 1000], [w:[#INTERNAL]LS1010], [w:[#INTERNAL]LS0100], 2, 3, 5)

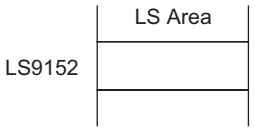
(Compares three words from Block 1 and Block 3 (starting from offset 2), and saves the comparison result in LS0100).



Since the values of the original and target data match, the comparison result “0” is stored in LS0100.



Error Status

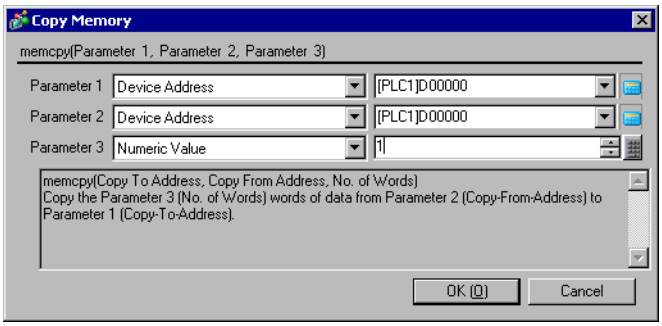


Editor Function Name	LS Area	Error Status	Cause
_memcmp ( )	LS9152	0000h	Completed Successfully
		0001h	Parameter error
		0003h	Write/Read error

IMPORTANT

- The effective LS device range that can be specified is limited to the designated user area (LS20 to LS2031 and LS2096 to LS8191).
- When you specify for the offset from the top of the block a value that is larger than the number of words in one block, this feature will not work.
- When the number of words to compare is larger than one block, this feature will not work.

21.3.3 Copy Memory

Item	Description
Summary	Copies device memory 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. The number of addresses that can be used is from 1 to 640.
Format	memcpy ([Copy To Address], [Copy From Address], No. of Words) 

Example expression:

memcpy ([w:[PLC1]D0200], [w:[PLC1]D0100], 10)

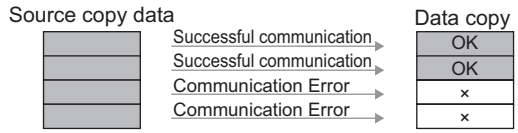
In the above example, data is copied from D0100-D0109 to D0200-D0209.

IMPORTANT

- Source copy data will be read from the connected device only once, when required. If a communication error occurs during data read, the GP's internal special relay LS2032's Bit 12 will be turned ON. When data read is completed normally, Bit 12 will be turned OFF.
- Reading from the source copy data and writing the data to the destination is performed in one operation, or it is accomplished by dividing the data into several pieces equivalent to the number of Addresses used for the source copy data. If a communication error occurs during data read, the result of the data copy varies as follows, depending on whether the data was processed in one operation or in several pieces: (Result of data copy OK: Properly copied, ×: No data copied)

(Copy in one operation)

(Copy by dividing data)

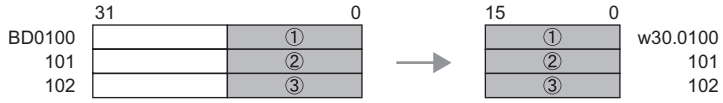


- As the number of Addresses increases, more time is required for writing data to the PLC. Depending on the number of Addresses, it may take from 20 seconds to several minutes.
- If data to be written exceeds the designated device range, a communication error occurs. In this case, you must turn the GP's power OFF and then ON again to reset the GP from the error.
- When the data are written to the LS Area with the Copy Memory (memcpy) function, the data can be written only in the User area. Data cannot be written into the System Data area (LS0000 to LS0019), Special area (LS2032 to LS2047), or Reserved area (LS2048 to LS2095). However, data can be read out from these areas.

Continued

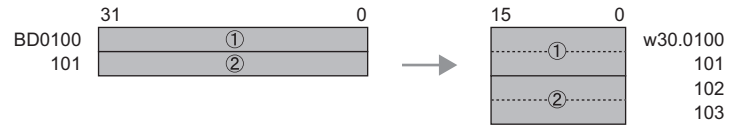
IMPORTANT

- When the 32 bit device data is copied to a 16 bit device using D-Script, and the bit length is designated as 16 bits, only the data for lower 16 bits will be copied.
  Example: memcpy ([w:[PLC1]w30.0100], [w:[PLC1]BD0100], 3)

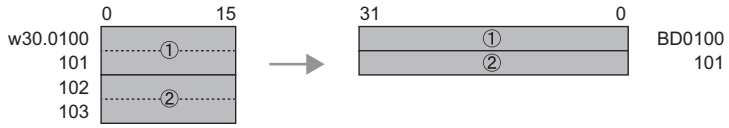


  Also, when 16 bit device data is copied to a 32 bit device, the data for the lower 16 bits will simply be copied and “0” will be designated for the upper 16 bits.
  Example: memcpy ([w:[PLC1]BD0100], [w:[PLC1]w30.0100], 3)

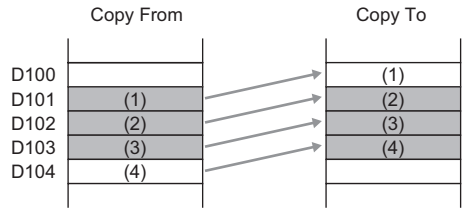
- When 32 bit device data is copied to a 16 bit device, or when 16 bit device data is copied to a 32 bit device, if the D-Script bit length designated in D-Script is 32, the copying will be as follows. When one of the devices is a 32 bit device and the other is a 16 bit device, use the 16 bit device’s no. of addresses to designate the memcpy ( ) function’s no. of address.
  Example: memcpy ([w:[PLC1]w30.0100], [w:[PLC1]BD0100], 4)



  Example: memcpy ([w:[PLC1]BD0100], [w:[PLC1]w30.0100], 4)



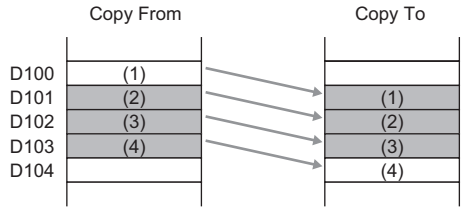
- If the original and destination data ranges overlap, all overlapping data will be rewritten as follows:
  Example: When copying D101-D104 to D100-D103
  Data is copied to a smaller number Address.



Continued

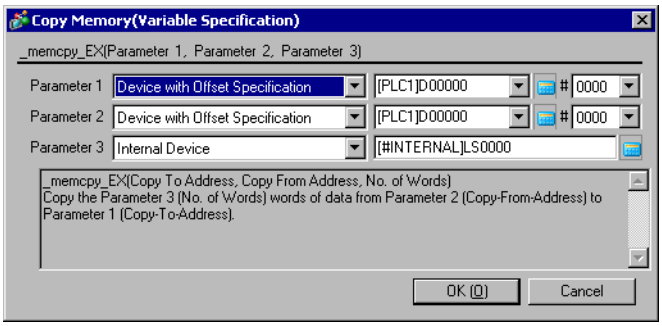
IMPORTANT

Example: When copying D100-D103 to D101-D104
Data is copied to a larger number Address.



- Although this example's function designates 2 Addresses, these Addresses will not be counted as D-Script Addresses.
- When using a device address for assignment, there is communication with the device/PLC therefore the written value will not be assigned right away.

21.3.4 Copy Memory (Variable Specification)

Item	Description
Summary	Copies device memory in one operation. The data of addresses specified with Parameter 3 are copied from the source (copy from) word address specified with Parameter 2 to the destination (copy to) word address specified with Parameter 1. The number of addresses that can be used is from 1 to 640. With the "_memcpy_EX" function, the source address, destination address, and number of addresses can be designated indirectly.
Format	_memcpy_EX ([Copy To Address], [Copy From Address], No. of Words) Parameter 1: Device address + Temporary address Parameter 2: Device address + Temporary address Parameter 3: Numeric Value, Internal Device, Temporary address (The valid range for Parameter 3 is from 1 to 640.) 

Example expression:

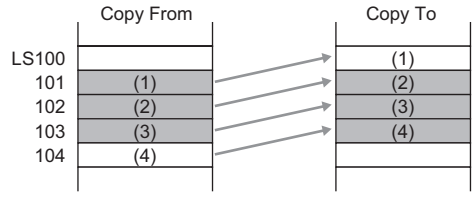
[t:0000]=10, [t:0001]=20

_memcpy_EX ([w:[#INTERNAL]LS 0100]#[t:0000], [w:[PLC1]D0100]#[t:0001], 5)

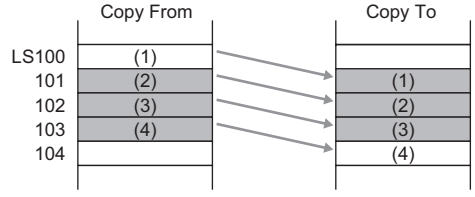
In the example above, five words of data will be read out from D0120 and written into LS0110 to LS0114.

IMPORTANT

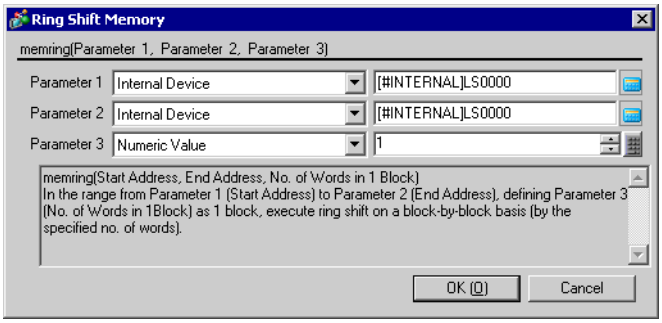
- If the original and destination data ranges overlap, all overlapping data will be rewritten as follows:
  Example: When copying LS101-LS104 to LS100-LS103
  Data is copied to a smaller number Address.



  Example: When copying LS100-LS103 to LS101-LS104
  Data is copied to a larger number Address.



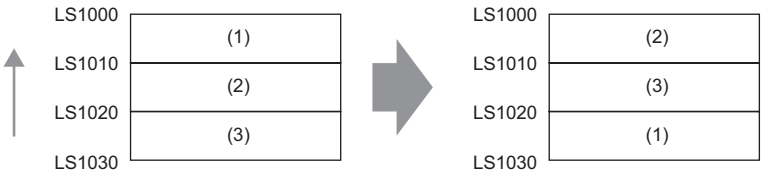
21.3.5 Ring Shift Memory

Item	Description
Summary	Ring-shifts the data in memory in blocks. Performs ring-shift between the start and ending addresses in block units (by the specified number of words). When an error occurs, the error status is written to LS9150.
Format	memring ([Start Address], [End Address], No. of Words in 1 Block)  Parameter 1: Internal Device Parameter 2: Internal Device Parameter 3: Numeric Value (1 to 640) When Parameter 1 is smaller than Parameter 2 (P1 < P2), the block data is shifted upward. When Parameter 1 is larger than Parameter 2 (P1 > P2), the block data is shifted downward. IMPORTANT • Make sure that the Start Address and End Address are set to the same type of device (LS or USR).

Example expression 1:

memring ([w:[#INTERNAL]LS1000], [w:[#INTERNAL]LS1030], 10)

(When Parameter 1 is smaller than Parameter 2 (P1 < P2))

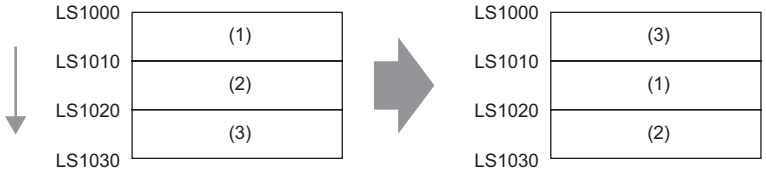


Data moves upward in 10-word block units.

Example expression 2:

memring ([w:[#INTERNAL]LS1030], [w:[#INTERNAL]LS1000], 10)

(When Parameter 1 is greater than Parameter 2 (P1 > P2))

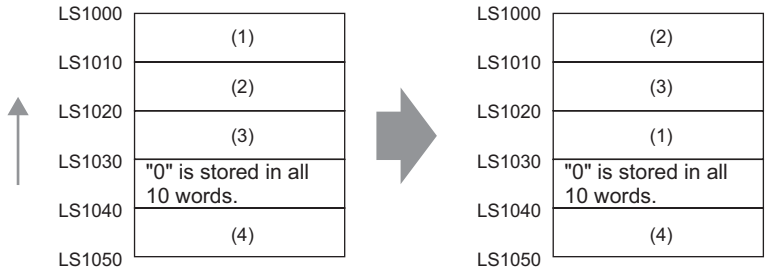


Data moves downward in 10-word block units.

Example expression 3:

memring ([w:[#INTERNAL]LS1000], [w:[#INTERNAL]LS1050], 10)

(When the range contains a block where all words are “0”.)

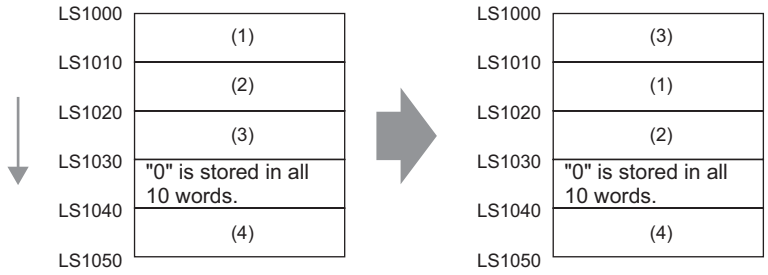


Data moves upward in 10-word block units only, from the starting block to the block with “0” data. If data exists after the block with “0” data, the data will be ignored.

Example expression 4:

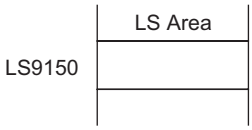
memring ([w:[#INTERNAL]LS1050], [w:[#INTERNAL]LS1000], 10)

(When a block with “0” data exists within the range.)



Data moves downward in 10-word block units only, from the starting block to the block with “0” data. If data exists after the block with “0” data, the data will be ignored.

Error Status

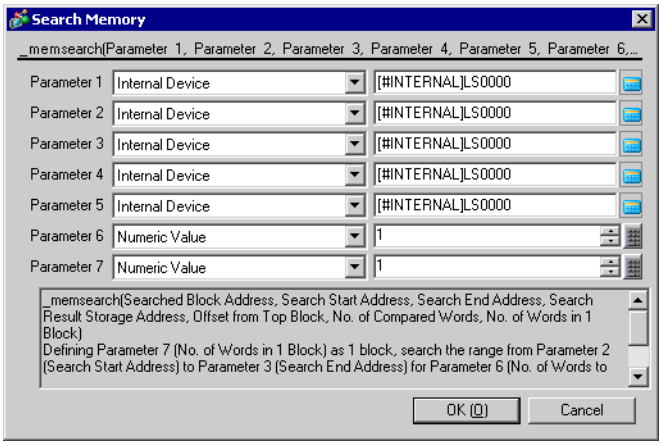


Editor Function Name	LS Area	Error Status	Cause
memring ( )	LS9150	0000h	Completed Successfully
		0001h	Parameter error
		0003h	Write/Read error

IMPORTANT

- The processing time required is proportional to the range designated by the start and end addresses. The larger the designated range, the longer the processing time becomes. The Part will not be refreshed until processing is completed.
- The effective LS device range that can be specified is limited to the designated user area (LS20 to LS2031 and LS2096 to LS8191).

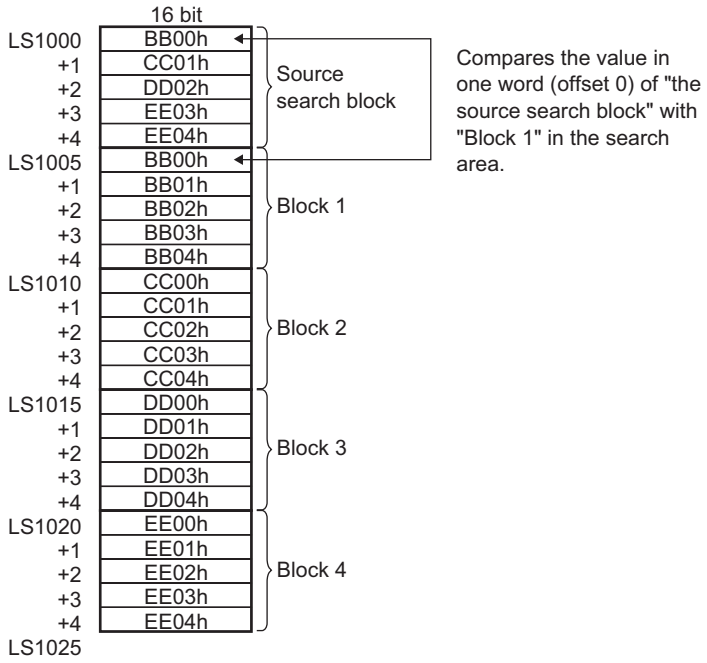
21.3.6 Search Memory

Item	Description
Summary	Performs a data search in block units, starting from the first item in the specified range. Compares data blocks, starting from the specified (offset) blocks and returns (saves) the search result to the specified storage address. When a matching block is found, the offset value of the block (1 or higher) is saved. When no matching block is found, “FFFFh” is saved. When an error occurs, the error status value is written to LS9153.
Format	_memsearch ([Searched Block Address], [Search Start Address], [Search End Address], [Search Result Storage Address], Offset from Top Block, No. of Compared Words, No. of Words in 1 Block)  Parameter 1: Internal Device Parameter 2: Internal Device Parameter 3: Internal Device Parameter 4: Internal Device Parameter 5: Numeric Value (0 to 639), Internal Device, Temporary variable Parameter 6: Numeric Value (1 to 640) Parameter 7: Numeric Value (1 to 640) Data to be written When there are matching blocks: The block’s offset value (“1” or higher) When there are no matching blocks: “FFFFh” IMPORTANT • Make sure that the search start address and search ending address are set to the same type of device (LS or USR). However, the [Searched Block Address] and [Search Result Storage Address] can be set to the Internal Device. • Be sure that [Parameter 2] is smaller than [Parameter 3] (Parameter 2 < Parameter 3). Otherwise, an error will result.

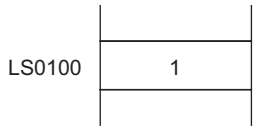
Example expression 1:

_memsearch ([w:[#INTERNAL]LS1000], [w:[#INTERNAL]LS1005], [w:[#INTERNAL]LS1025], [w:[#INTERNAL]LS0100], 0, 1, 5)

(Searches from LS1005 to LS1025 for a block with the same value. Starts from offset 0 of the source search block, and stores the result in LS0100.)



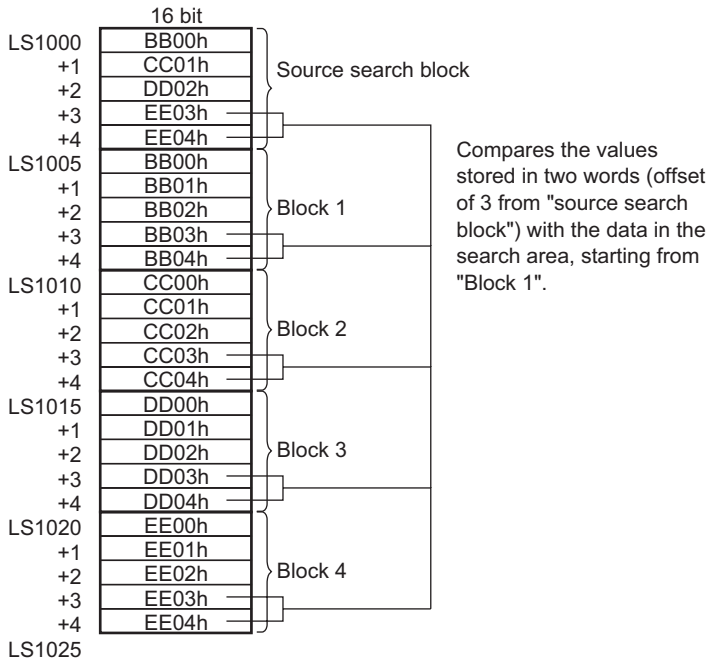
In this case, the value of “Block 1” matches the value of “the source search block”; As a result the search result “1” is stored in LS0100.



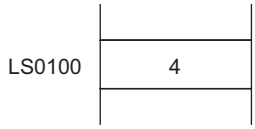
Example expression 2:

_memsearch ([w:[#INTERNAL]LS1000], [w:[#INTERNAL]LS1005], [w:[#INTERNAL]LS1025], [w:[#INTERNAL]LS0100], 3, 2, 5)

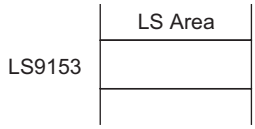
(Searches from LS1005 to LS1025 for a block with the same value. Uses two words, starting from an offset of 3, and stores the result in LS0100.)



In this case, the value of “Block 4” matches the value of “the source search block”. As a result the search result “4” is stored in LS0100.



Error Status

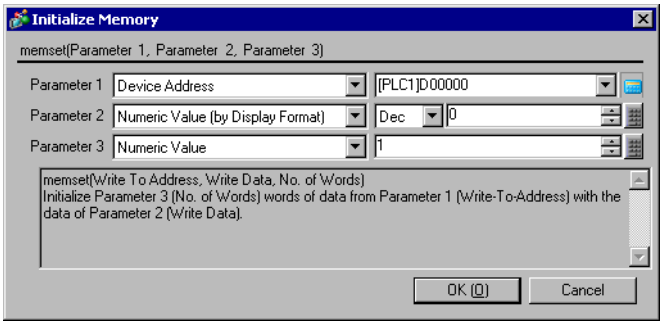


Editor Function Name	LS Area	Error Status	Cause
_memsearch ( )	LS9153	0000h	Completed Successfully
		0001h	Parameter error
		0003h	Write/Read error

IMPORTANT

- The processing time required is proportional to the range designated by the start and end addresses. The larger the designated range, the longer the processing time becomes. The Part will not be refreshed until processing is completed.
- The effective LS device range that can be specified is limited to the designated user area (LS20 to LS2031 and LS2096 to LS8191).

21.3.7 Initialize Memory

Item	Description
Summary	Initializes all devices at once. Setting data for the number of Addresses is taken from the Set Word Address. The valid range for the number of addresses is from 1 to 640.
Format	memset ([Write-To Address], Write Data, No. of Words) 

Example expression:

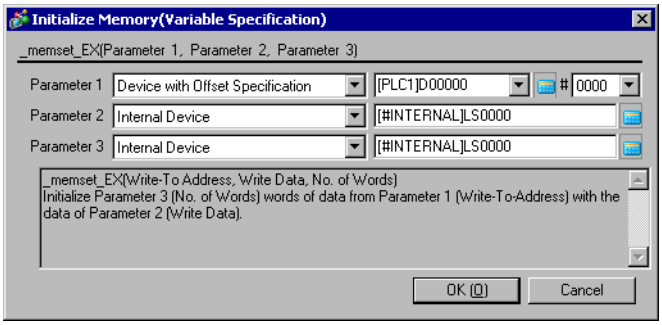
memset ([w:[PLC 1]D0100], 0, 10)

In the above example, “0” is set for the addresses D0100 to D0109.

IMPORTANT

- As the number of Addresses increases, more time is required for writing data to the PLC. Depending on the number of Addresses, it may take from 20 seconds to several minutes.
- If data to be written exceeds the designated device range, a communication error occurs. In this case, you must turn the GP’s power OFF and then ON again to reset the GP from the error.
- Although this function designates Address(es), they are not counted as D-Script Address(es).
- When writing data to the LS Area with the Memory Reset (memset) function, the data can be written only into the User area. Data cannot be written into the System Data area (LS0000 to LS0019), Special area (LS2032 to S2047), or Reserved area (LS2048 to LS2095).
- When using device addresses for the Assign operation, the write values will not be assigned immediately, due to the GP to PLC transmission time.
  (Example)
  memset ([w:D 0100], 0, 10) // Initializes “D100 to D109” as 0
  [w:D200] = [w:D100] // Assigns D100 data to D200.
  In this case, value 0 written to D100 as the operation result has not been assigned to D200 yet.

21.3.8 Initialize Memory (Variable Specification)

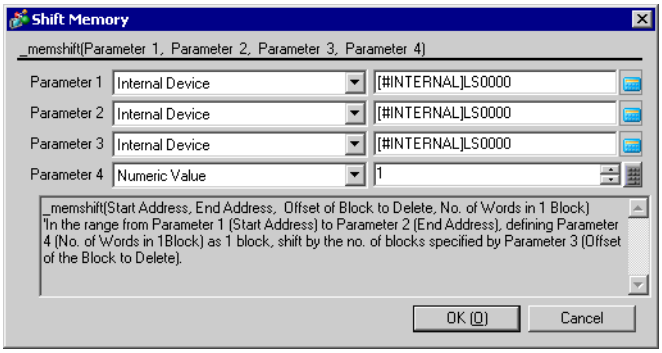
Item	Description
Summary	Initializes all devices at once. The Set data specified with Parameter 2 are set from the Set Word Address specified with Parameter 1 into the addresses specified with Parameter 3. The valid range for the number of addresses is from 1 to 640. The Write-To Address, Write Data, and number of addresses can each be designated indirectly.
Format	_memset_EX ([Write-To Address], Write Data, No. of Words)  Parameter 1: Device address + Temporary address Parameter 2: Numeric Value, Internal Device, Temporary address (The valid range for Parameter 2 is from 0 to 65535 for Dec, and from 0 to FFFF for Hex.) Parameter 3: Numeric Value, Internal Device, Temporary address (The valid range for Parameter 3 is from 1 to 640.)

Example expression:

[t:0000]=10
[w:LS0050]=0
[w:LS0051]=5
_memset_EX ([w:[#INTERNAL]LS0100]#[t:0000], [w:[#INTERNAL]LS0050], [w:[#INTERNAL]LS0051])

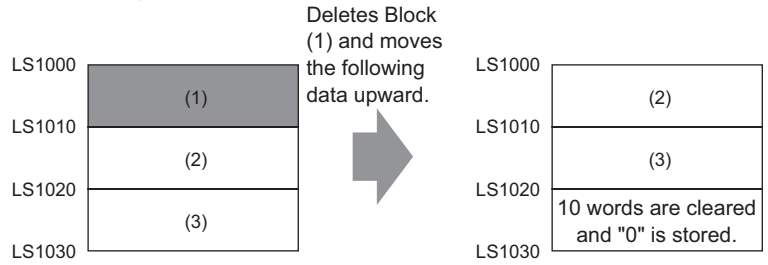
In the example above, “0” will be written into the five words from LS0100 to LS0114.

21.3.9 Shift Memory

Item	Description
Summary	Deletes the specified block and moves the following data blocks upward. The block to be deleted is designated using an offset. When an error occurs, the error status is written to LS9151.
Format	_memshift ([Start Address], [End Address], Offset of Block to Delete, No. of Words in 1 Block)  Parameter 1: Internal Device Parameter 2: Internal Device Parameter 3: Numeric Value (1 to 65,535), Internal Device, Temporary variable Parameter 4: Numeric Value (1 to 640) IMPORTANT • Make sure that the Start Address and End Address are set to the same type of device (LS or USR). • Be sure that [Parameter 1] is smaller than [Parameter 2] (Parameter 1 < Parameter 2). Otherwise, an error will result.

Example expression 1:

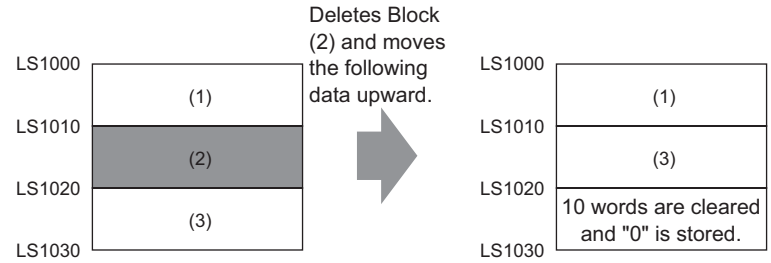
_memshift ([w:[#INTERNAL]LS1000], [w:[#INTERNAL]LS1030], 1, 10)



Data moves upward in block units (1 block = 10 words), and the last block (10 words) is cleared to zero.

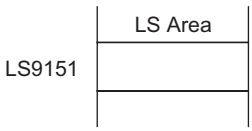
Example expression 2:

_memshift ([w:[#INTERNAL]LS 1000], [w:[#INTERNAL]LS1030], 2, 10)



The data moves upward in block units (1 block = 10 words) starting from the offset 2 position, and the last block (10 words) is cleared to zero.

Error Status

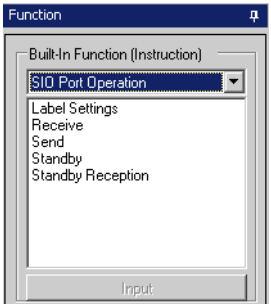


Editor Function Name	LS Area	Error Status	Cause
_memshift ( )	LS9151	0000h	Completed Successfully
		0001h	Parameter error
		0003h	Write/Read error

IMPORTANT

- The processing time required is proportional to the range designated by the start and end addresses. The larger the designated range, the longer the processing time becomes. The Part will not be refreshed until processing is completed.
- When a value exceeding the range specified for the start and ending addresses is designated as the offset of the block to be deleted, this feature will not operate correctly.
- The effective LS device range that can be specified is limited to the designated user area (LS20 to LS2031 and LS2096 to LS8191).

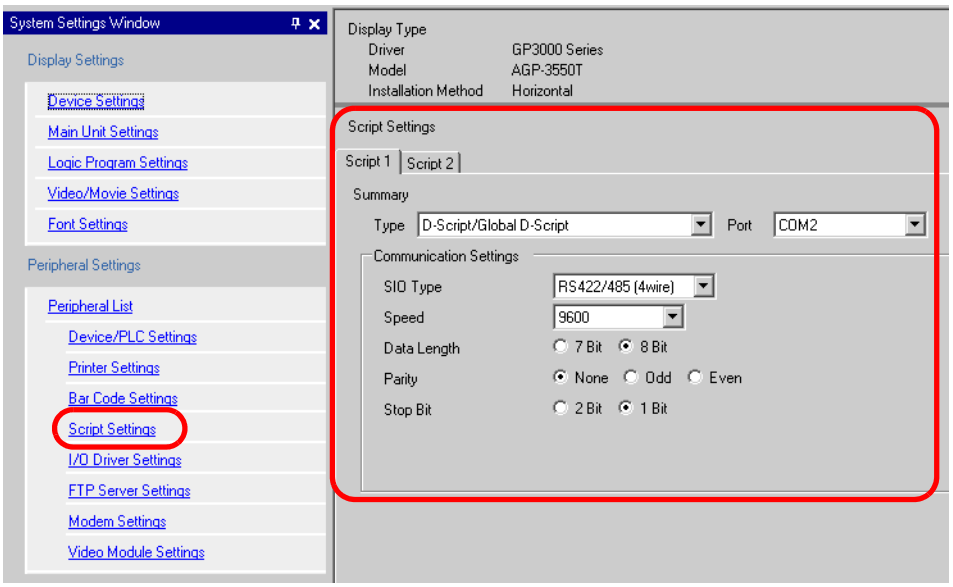
21.4 SIO Port Operation

SIO Port Operation	Function Summary
	Label Settings ☞ "21.4.1 Label Settings" (page 21-26) Designated from the Control, Status, Receive Data Count, Receive Function, and Send Function.
	Receive ☞ "21.4.2 Receive" (page 21-30) Reads received data from the designated serial port (COM1 or COM2).
	Send ☞ "21.4.3 Send" (page 21-31) Writes to the designated serial port (COM1 or COM2).
	Extended Receive ☞ "21.4.4 Extended Receive" (page 21-32) Reads received data from the designated serial port (COM1 or COM2). It can only be used in an Extended Script.
	Extended Send ☞ "21.4.5 Extended Send" (page 21-33) Writes to the designated serial port (COM1 or COM2). It can only be used in an Extended Script.
	Standby Reception Function ☞ "21.4.6 Standby Reception Function" (page 21-34) Stays in standby receive mode until it receives specified strings. It can only be used in an Extended Script.
	Standby Function ☞ "21.4.7 Standby Function" (page 21-35) The system waits (suspends operation) for the specified period of time until it executes the process. It can only be used in an Extended Script.

IMPORTANT

- Label Settings, Send, and Receive can be easily included in a D-Script/Global D-Script.
- To communicate with D-Scripts/Global D-Scripts, please make sure to designate the following script settings. If script settings are not designated, they can not execute.

  [D-Script/Global D-Script Settings Procedure]
  (1) Click [Project] - [System Settings] - [Script Settings].
  Set the [Type] to “D-Script/Global D-Script”.



  There are 2 tabs in the Script Settings. “Script 1” is shown above.
  Set the [Port] to COM1 or COM2, and set the [Communication Settings] to match the Extended SIO.
- When creating a communication program with more advanced functionality than the SIO port operation, it is recommended to use an [Extended Script]. For examples on how to use extended scripts, refer to
  ☞ “20.5 Communicating with Peripheral Devices not Supported by Regular Scripts” (page 20-21)

21.4.1 Label Settings

◆ **Control**

When designating the bit: [c:EXT_SIO_CTRL **] (write-only)
When designating the word: [c:EXT_SIO_CTRL] (write-only)

◆ **Status**

When designating the bit: [s:EXT_SIO_STAT **] (read-only)
When designating the word: [s:EXT_SIO_STAT] (read-only)

◆ **Received Data Size**

[r:EXT_SIO_RCV] (read-only)

◆ **Receive Function**

IO_READ ([p:EXT_SIO], LS storage address, Number of bytes)

◆ **Send Function**

IO_WRITE ([p:EXT_SIO], LS storage address, Number of bytes)

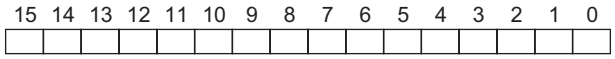
■ Control

Item	Description
Summary	This control variable is used to clear the Send buffer, Receive buffer, and error status. This control variable is write-only.
Format	When designating the bit: [c:EXT_SIO_CTRL**] (** : 00 to 15) When designating the word: [c:EXT_SIO_CTRL]

Example expression:

When designating the bit: [c:EXT_SIO_CTRL00] = 1
When designating the word: [c:EXT_SIO_CTRL] = 0x0007

EXT_SIO_CTRL



Bit	Content
15	Reserved
14	
13	
12	
11	
10	
9	
8	
7	
6	
5	
4	
3	1: Clear Receive timeout
2	1: Clear error
1	1: Clear Receive buffer
0	1: Clear Send buffer

NOTE

- When a word is designated (when two or more bits are set simultaneously), the processing will be executed in the following order:
  Clear Error → Clear Receive Buffer → Clear Send Buffer

■ Status

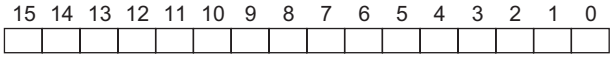
Item	Description
Summary	Status includes the following information. This status variable is write-only.
Format	When designating the bit: [s:EXT_SIO_STAT**] (** : 00 to 15) When designating the word: [s:EXT_SIO_STAT]

Example expression:

When designating the bit: if ([s:EXT_SIO_STAT 00] == 1)

When designating the word: if (([s:EXT_SIO_STAT] & 0x0001) <> 0)

Contents of EXT_SIO_STAT



Bit	Content
15	0: No D-Script/Global D-Script 1: D-Script/Global D-Script exists
14	0: No extended script 1: Extended script exists
13	Reserved
12	
11	
10	
9	
8	
7	
6	
5	
4	0: Normal 1: Receive timeout
3	0: Normal 1: Receive error
2	0: No receive data 1: Receive data exists
1	0: Normal 1: Send error
0	0: Data exists in Send buffer 1: Send buffer is empty

NOTE

- The reserved bits may be assigned in the future. Therefore, be sure to check only the necessary bits.
- Two types of transmission errors exist: the transmission timeout error and the transmission buffer-full error. When either of the two errors occurs, the transmission error bit turns ON. The transmission timeout period is five seconds.
- There are four types of receive errors: parity error, overrun error, framing error, and overflow. When one of these four errors occurs, the bit for the receive error turns ON.
- If a transmission error is detected, the send data will remain in the transmission buffer. If a transmission error cannot be detected, the send data will be sent from the transmission buffer.
- When using the serial interface COM2, which is RS-422, the CS (CTS) signal cannot be detected. As a result, disconnection of a cable cannot be detected.

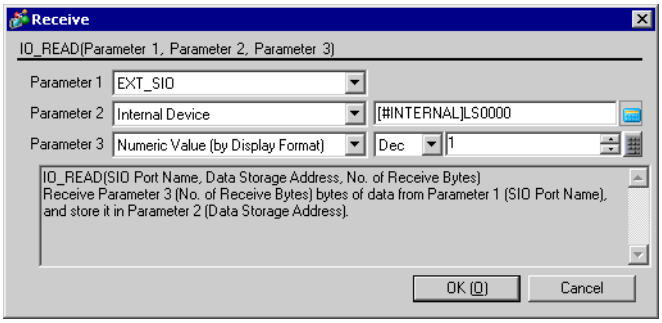
■ Received Data Size

Item	Description
Summary	Shows the quantity of data (number of bytes) that has been received at that time. The received data size is a read-only feature.
Format	[r:EXT_SIO_RECV]

IMPORTANT

- Label name of the No. of Received Data (number of bytes)
  With GP-PRO/PB III V.6.0 and earlier versions, the Label name designated for the received data size is [r: EXT_SIO_RCV]. However, you are not required to revise the description because the function will be the same whether [r: EXT_SIO_RCV] or [r: EXT_SIO_RECV] expression is selected.

21.4.2 Receive

Item	Description
Summary	Write the statement as follows when reading out the received data from the Extended SIO.
Format	IO_READ ([p:EXT_SIO], Data Storage Address, No. of Receive Bytes)  Parameter 1: EXT_SIO Parameter 2: Internal Device Parameter 3: Numeric Value

Example expression:

IO_READ ([p:EXT_SIO], [w:[#INTERNAL]LS0100], 10)

In the above example, the number of bytes received is stored in LS0100. 10 bytes of data is stored starting from LS0101.The following image shows the stored received data.

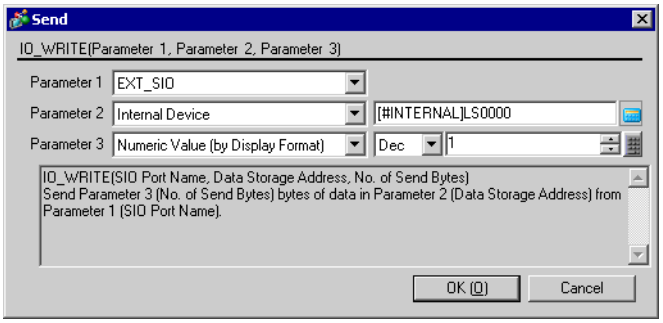
NOTE

- The maximum number of transfer bytes during data reception is 2,011. The data is written to each word address in units of 1 byte.

LS0100	Received Data Size		... 10 bytes
LS0101	00	Byte 1	
LS0102	00	Byte 2	
LS0103	00	Byte 3	
LS0104	00	Byte 4	
LS0105	00	Byte 5	
LS0106	00	Byte 6	
LS0107	00	Byte 7	
LS0108	00	Byte 8	
LS0109	00	Byte 9	
LS0110	00	Byte 10	

Received Data Storage Method

21.4.3 Send

Item	Description
Summary	Write the statement as follows when writing data to the Extended SIO.
Format	IO_WRITE ([p:EXT_SIO], Data Storage Address, No. of Send Bytes)  Parameter 1: EXT_SIO Parameter 2: Internal Device Parameter 3: Numeric Value

Example expression:

IO_WRITE ([p:EXT_SIO], [w:[#INTERNAL]LS0100], 10)

In the above example, 10 bytes of data starting from LS0100 are sent. The following image shows the stored sent data.

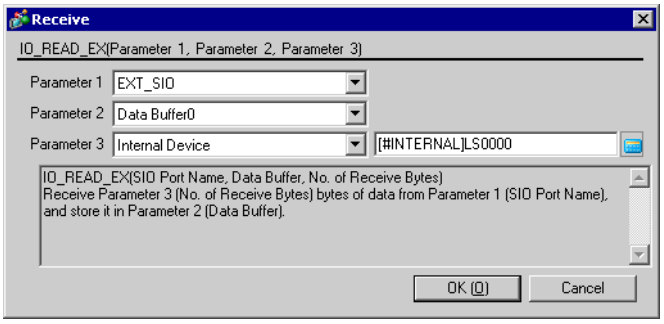
NOTE

- The maximum number of transfer bytes when receiving data is 2,012.
- As the LS device for the Send buffer, write the data in single bytes to each word address.

LS0100	00	Byte 1
LS0101	00	Byte 2
LS0102	00	Byte 3
LS0103	00	Byte 4
LS0104	00	Byte 5
LS0105	00	Byte 6
LS0106	00	Byte 7
LS0107	00	Byte 8
LS0108	00	Byte 9
LS0109	00	Byte 10

Sent Data Storage Method

21.4.4 Extended Receive

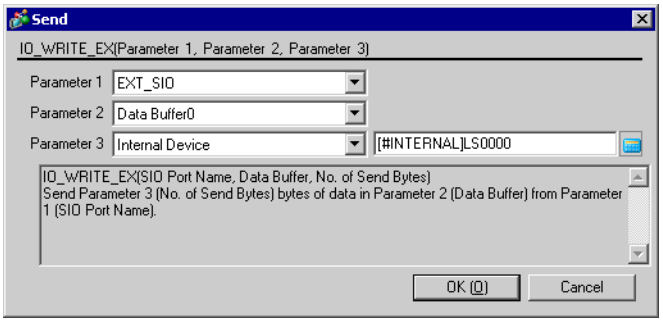
Item	Description
Summary	Receives data of the size indicated in Received Data Size (bytes) from the Extended SIO and stores it in the data buffer. The number of bytes specified with Parameter 3 is received from the Extended SIO and stored in the data buffer specified with Parameter 2. It can only be used in an Extended Script.
Format	IO_READ_EX ([p:EXT_SIO], Data Buffer, No. of Receive Bytes)  Parameter 1: [p:EXT_SIO] Parameter 2: Data Buffer Parameter 3: Numeric Value, Internal Device, Temporary address (The valid range for Parameter 3 is from 1 to 1,024.)

Example expression:

IO_READ_EX ([p:EXT_SIO], databuf 1, 10)

In the above example, 10 bytes of data in the data received by the Extended SIO are received and stored in “databuf1”.

21.4.5 Extended Send

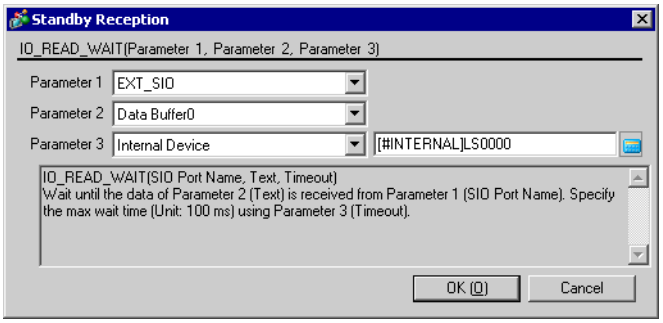
Item	Description
Summary	Sends the data in the data buffer with Extended SIO according to the size of No. of Send Bytes. The contents of the data buffer specified with Parameter 2 are sent from Extended SIO by the length specified with Parameter 3. It can only be used in an Extended Script.
Format	IO_WRITE_EX ([p:EXT_SIO], Data Buffer, No. of Send Bytes)  Parameter 1: [p:EXT_SIO] Parameter 2: Data Buffer Parameter 3: Numeric Value, Internal Device, Temporary address (The valid range for Parameter 3 is from 1 to 1,024.)

Example expression:

IO_WRITE_EX ([p:EXT_SIO], databuf 0, 10)

In the example above, 10 bytes of data in “databuf0” are sent from Extended SIO.

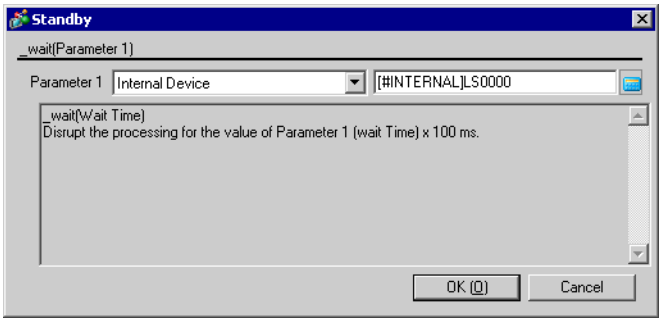
21.4.6 Standby Reception Function

Item	Description
Summary	Stays in standby receive mode until it receives specified text. After the timeout period has expired, Bit 4 (Receive time-out error) of Status [s:EXT_SIO_STAT] is set. The timeout duration can be set in 100-ms increments. The system is in standby receive mode until it receives the character string or character code specified with Parameter 2. Configure the timeout duration with Parameter 3. It can only be used in an Extended Script.
Format	IO_READ_WAIT([p:EXT_SIO], Text, Timeout)  Parameter 1: [p:EXT_IO] Parameter 2: Numeric Value, Text, Data Buffer Parameter 3: Numeric Value, Internal Device, Temporary address (The valid range for Parameter 3 is from 1 to 600.)

IMPORTANT

- The received data cannot be used until the specified text is received. (Otherwise, the data are abandoned.)
- Up to 128 characters (bytes) can be specified. Note that the standby receive operation cannot be performed successfully when strings exceeding the limit are specified.

21.4.7 Standby Function

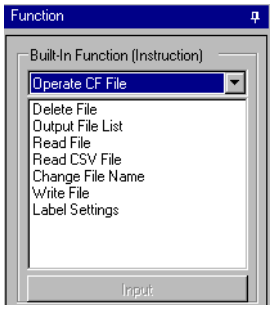
Item	Description
Summary	The system waits (suspends operation) for the specified period of time. The time can be configured in 100-ms increments. It can only be used in an Extended Script.
Format	_wait (Wait Time)  Parameter 1: Internal Device, Temporary address, Numeric Value (The valid range for Parameter 1 is from 1 to 600.)

Example expression:

_wait (10)

In the example above, the system waits one second.

21.5 CF File Operation

Operate CF File	Function Summary
	Label Settings ☞ "21.5.1 Label Settings" (page 21-37) Designated from the No. of Files Listed, No. of Read Bytes, and CF-Card Error Status.
	Write File ☞ "21.5.2 Write File" (page 21-44) Writes the specified number of bytes of data from the source address to the specified file.
	Change File Name ☞ "21.5.3 Change File Name" (page 21-47) Modifies the file name.
	Read CSV File ☞ "21.5.4 Read CSV File" (page 21-49) Reads data in cell units from a CSV file and writes it to a word address.
	Read File ☞ "21.5.5 Read File" (page 21-51) Reads the specified number of bytes of data in the file after the specified offset and writes it in the destination address.
	Output File List ☞ "21.5.6 Output File List" (page 21-53) The list of files that exist in the specified folder is written in the Internal Device.
	Delete File ☞ "21.5.7 Delete File" (page 21-54) Deletes the file.

21.5.1 Label Settings

The following statuses are used for CF-Card status:

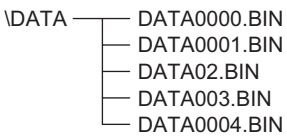
Status name	Label name	Description
Listed Files	[s:CF_FILELIST_NUM]	Stores the number of files actually listed when the File List Output function "_CF_dir ( )" is executed.
No. of Read Bytes	[s:CF_READ_NUM]	Stores the number of bytes that can be read out when the File Read function "_CF_read ( )" is executed.
CF-Card Error Status	[s:CF_ERR_STAT]	Stores the error status generated when the CF-Card is accessed.

■ Listed Files

When the File List Output function "_CF_dir ( )" is executed, the number of file lists that are actually written in the LS Area is stored in "Listed Files [s:CF_FILELIST_NUM]".

Usage example

_CF_dir ("\DATA*.*", [w:[#INTERNAL]LS0100], 10, 0)
[w:LS0200] = [s:CF_FILELIST_NUM]



When an attempt is made to obtain a file list of 10 files but the specified folder contains only five files, "5" is stored in [s:CF_FILELIST_NUM].

IMPORTANT
- When no files are written, the total number of files contained in the specified folder is written in [s:CF_FILELIST_NUM].

■ No. of Read Bytes

When the File Read function "_CF_read ( )" is executed, the number of bytes actually read out is stored in "Readout Bytes [s:CF_READ_NUM]".

Usage example

_CF_read ("\DATA", "DATA 0001.BIN", [w:[#INTERNAL]LS0100], 16, 16)
[w:[#INTERNAL]LS0200] = [s:CF_READ_NUM]

When an attempt is made to read 16 bytes but only 12 bytes are read successfully, "12" is stored in [s:CF_READ_NUM].

■ CF-Card Error Status

Stores error statuses generated when the CF-Card is accessed.

Bit Position	Error Name	Description
15	Reserved	Reserved
14		
13		
12		
11		
10		
9		
8		
7		
6	File rename error	• CF-Card was removed during operation. • Specified file does not exist. • An attempt was made to rename a file with a read-only attribute.
5	File delete error	• CF-Card was removed during operation. • Specified file does not exist. • An attempt was made to delete a file with a read-only attribute.
4	File write error	• CF-Card was removed during operation. • No available space remains on CF-Card. • An attempt was made to write data to a file with a read-only attribute. • For attempting to “overwrite”, the designated file does not exist.
3	File read error	• CF-Card was removed during operation. • Specified file does not exist.
2	File list error	• CF-Card was removed during operation. • Specified folder does not exist.
1	CF-Card Error	• CF-Card is invalid. • The media inserted is not a CF-Card.
0	No CF-Card	• No CF-Card is inserted. • Cover is open.

- Even when a CF-Card error occurs, operation continues. Be sure to write a script to check the error whenever you use the CF-Card file operation function.
  Example)

```
_CF_dir (“\DATA\*.*”, [w:[#INTERNAL]LS0100], 2, 1) Outputs a file list.
if ([s:CF_ERR_STAT02] <> 0)          // Checks the error status.
{
    set ([b:[#INTERNAL]LS 005000])// Sets the bit address for error display.
}
endif
```

CF-Card Error Detailed Status - Storage Area

If an error occurs, the appropriate bits are set. You can check the cause of the error by referring to the detailed status. The detailed status for each function is stored in LS9132 through LS9137 of the extended system area. These areas are read-only.

	LS Area	
LS0000		
:		
LS9132		Status of CF-card list operation
LS9133		Status of CF-card read operation
LS9134		Status of CF-card write operation
LS9135		Status of CF-card delete operation
LS9136		Status of CF-card rename operation
LS9137		Status of CSV Read
:		
LS9999		

Error list for each function

Editor Function Name		Error Status	Cause
_CF_dir ( )	LS9132	0010h	Invalid D-Script data (Error in retrieving folder name specified with fixed string)
		0012h	File name (path name) error
		0018h	LS Area writing range error
		0020h	No CF-Card
		0021h	Invalid CF-Card
		0100h	Directory open error

Editor Function Name		Error Status	Cause
_CF_read ( )	LS9133	0010h	Invalid D-Script data (Error in retrieving folder name/file name specified with fixed string)
		0011h	LS Area reading range error
		0012h	File name (path name) error
		0018h	LS Area writing range error
		0020h	No CF-Card
		0021h	Invalid CF-Card
		0101h	File seek error (Offset error)
		0102h	Number of readout bytes error
		0110h	File creation (open) error

Continued

Editor Function Name		Error Status	Cause
_CF_write ( )	LS9134	0010h	Invalid D-Script data (Error in retrieving folder name/file name specified with fixed string)
		0011h	LS Area reading range error
		0012h	File name (path name) error
		0020h	No CF-Card
		0021h	Invalid CF-Card
		0101h	File seek error (Offset error)
		0104h	Folder creation error
		0108h	Write mode error
		0110h	File creation (open) error
		0111h	File write error (Example Insufficient space on CF-Card)
_CF_delete ( )	LS9135	0010h	Invalid D-Script data (Error in retrieving folder name/file name specified with fixed string)
		0011h	LS Area reading range error
		0012h	File name (path name) error
		0020h	No CF-Card
		0021h	Invalid CF-Card
		0112h	File delete error (Example Specified file does not exist. Specified file is read-only.)
_CF_rename ( )	LS9136	0010h	Invalid D-Script data (Error in retrieving folder name/file name specified with fixed string)
		0011h	LS Area reading range error
		0012h	File name (path name) error
		0020h	No CF-Card
		0021h	Invalid CF-Card
		0114h	File rename error (Example Specified file does not exist. Specified file is read-only. File name already exists.)
_CF_read_csv()	LS9137	0001h	Parameter error
		0002h	CF-Card error (No CF-Card, Open file error, File read error)
		0003h	Write Error

Data Store Mode

When data is read/written from/to device addresses at the execution of the File Read/File Write function, the storage order of the written (readout) data can be specified.
Setting the data storage mode in LS9130 can change the storage order. The mode can be selected from four options: 0, 1, 2 and 3.

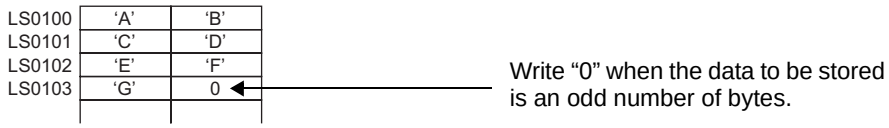
◆ Mode 0

Example: When the File Read function is used to write a string “ABCDEFG” in a device address

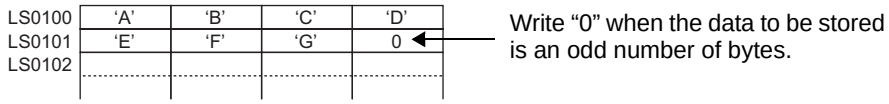
[w:[#INTERNAL]LS9130] = 0

_CF_read (“\DATA”, “DATA0001.BIN”, [w:[#INTERNAL]LS0100], 0, 7)

- When the device address length is 16 bits



- When the device address length is 32 bits



◆ Mode 1

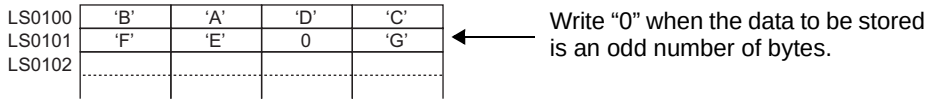
Example: When the File Read function is used to write a string “ABCDEFG” in a device address

[w:[#INTERNAL]LS9130] = 1

_CF_read (“\DATA”, “DATA0001.BIN”, [w:[#INTERNAL]LS0100], 0, 7)

- When the device address length is 16 bits

- When the device address length is 32 bits



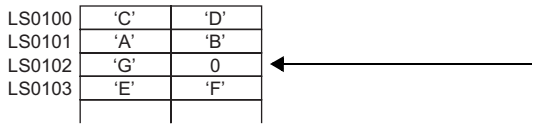
◆ Mode 2

Example: When the File Read function is used to write a string “ABCDEFG” in a device address

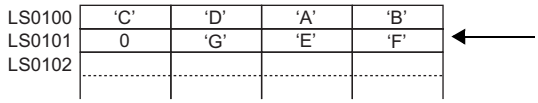
[w:[#INTERNAL]LS9130] = 2

_CF_read (“\DATA”, “DATA0001.BIN”, [w:[#INTERNAL]LS0100], 0, 7)

- When the device address length is 16 bits



- When the device address length is 32 bits



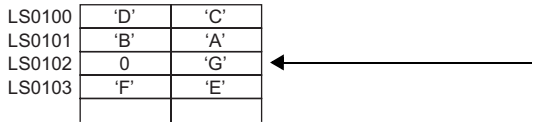
◆ Mode 3

Example: When the File Read function is used to write a string “ABCDEFG” in a device address

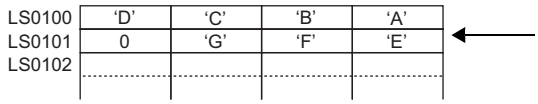
[w:[#INTERNAL]LS9130] = 3

_CF_read (“\DATA”, “DATA0001.BIN”, [w:[#INTERNAL]LS0100], 0, 7)

- When the device address length is 16 bits



- When the device address length is 32 bits



IMPORTANT

- The data storage mode is not the same as the string data mode in the system setting. The relationship with the string data mode is shown in the following table.

Data Device Storage Order	Byte in Word LH/HL Storage Order	LH/HL Storage Order In Double Word	D-Script data storage mode	Text Data Mode
Store from Top Data	HL Order	HL Order	0	1
	LH Order		1	2
	HL Order	LH Order	2	5
	LH Order		3	4
Store from Last Data	HL Order	HL Order	—	3
	LH Order		—	7
	HL Order	LH Order	—	8
	LH Order		—	6

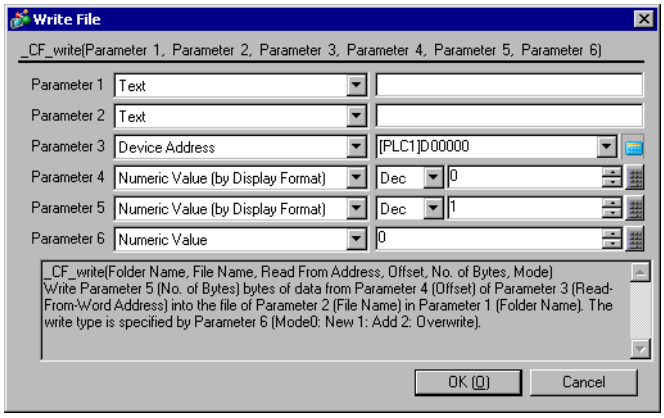
IMPORTANT

- There is a limit to the frequency that data can be rewritten to the CF-Card. Therefore, be sure to backup all CF-Card data regularly to another storage media. (assuming that 500KB of DOS format data is overwritten, the limit is 100,000 times)
- If an error occurs during CF-Card processing, the error is written to the CF-Card Error Status [s:CF_ERR_STAT]. For more details, refer to ☞ " ■ CF-Card Error Status" (page 21-38) .
- The following symbols and characters cannot be used in folder names or file names. Use of these symbols and characters in a folder name or file name will generate an error.

:	,	=	+	/	"	[
]	|	<	>	(space)	?	

- To specify a root folder (directory), specify " " (empty string) as the folder name.

21.5.2 Write File

Item	Description
Summary	Writes the specified number of bytes of data from the source address to the specified file. Any one of three modes can be selected: “New”, “Add” or “Overwrite”. See the “Data Storage Mode” section below for more details about data storage order.
Format	_CF_write (Folder Name, File Name, Read From Address, Offset, No. of Bytes, Mode)  Parameter 1 Folder name: Fixed string (Maximum length: 32 single-byte characters) Parameter 2 File name: Fixed string, Internal Device (Maximum length: 32 single-byte characters), Internal Device + Temporary address Parameter 3 Read From Address: Device address, Device address + Temporary address Parameter 4 Offset: Numeric Value, Device address, Temporary address (Maximum number that can be specified: 65,535 for 16-bit length, 4,294,967,295 for 32-bit length) Parameter 5 Number of bytes: Numeric Value, Device address, Temporary address (Maximum length: 1280) Parameter 6 Mode: Numeric Value, Device address, Temporary address (Available values: 0,1,2)

Storage Format Overview

Mode	Name	Description
0	New	Create a new file. If a file with the same name exists, it is deleted.
1	Add	Add the data to a specified file. If the specified file does not exist, a new file is created.
2	Overwrite	Overwrite part of the file. If the specified offset is larger than the file size, the surplus area is filled with 0s and the data is written after the area. If the offset is specified at the end of the file data, the operation is equivalent to adding the data to the file. If the file does not exist, an error will occur. For more information about this error, please refer to “ ■ CF-Card Error Status” (page 21-38) .

Example expression:

```
[w:[#INTERNAL]LS0200] = 0    //Offset (“0” when the mode is “New”)
[w:[#INTERNAL]LS0202] = 100// No. of Bytes (100 bytes)
[w:[#INTERNAL]LS0204] = 0    //Mode (New)
_CF_write (“\DATA”, “DATA0001.BIN”, [w:[#INTERNAL]LS0100],
[w:[#INTERNAL]LS0200], [w:[#INTERNAL]LS0202], [w:[#INTERNAL]:LS0204]
( [#INTERNAL]LS0202], [w:[#INTERNAL]:LS0204])
```

The above example creates a new file, DATA0001.BIN, in the \DATA folder and stores 100 bytes of data read from LS0100.

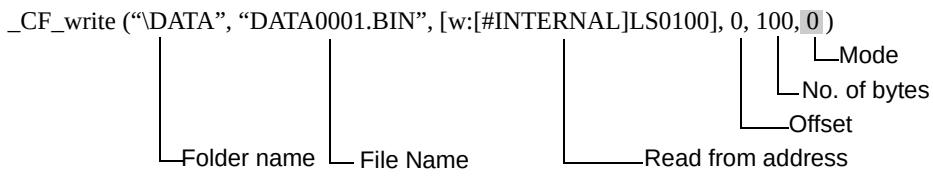
When the Internal Device is specified for the offset, the number of bytes or the mode, they can be designated indirectly.

IMPORTANT

- The offset setting is effective only in “Overwrite” mode. The offset setting is disabled in “New” and “Add” modes. Set the offset value to “0” in modes other than “Overwrite” mode.
- When “New” mode is specified and a file with the same name already exists, it is overwritten.
- When the LS Area is specified for “File name”, “Read From Address” is not counted as a D-Script address.
- When a PLC device is specified for “Read From Address” data is read from the PLC only once when the function is executed. If an error occurs during data read, the error is set in the CF-Card Error Status [s:CF_ERR_STAT]. The error is cleared when the data read is successfully completed.
- The data is divided into pieces and read from the source, although this depends on the number of bytes to be read. Therefore, even if a communication error occurs during data read, the data may have been partially written to the specified file.
- To specify a full path for a file name, specify “*” (asterisk) as the folder name. Example: _CF_read (“*” “\DATA\DATA0001.BIN” [w:[#INTERNAL]LS0100], 0, 10)

Storage format example expression

◆ **When “New” mode is specified**



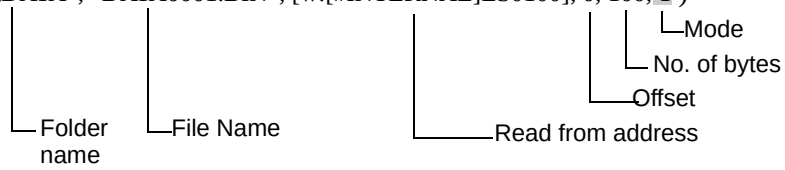
When the above example is executed, 100 bytes of data are read from LS0100 and following areas and written into the DATA0001.BIN file, which is a new file created in the \DATA folder.

IMPORTANT

- Only the 8.3 format (Up to 12 characters total: 8 characters for the file name and 3 characters for the extension) can be used for the file name. A file name longer than this format cannot be used.

◆ When “Add” mode is specified

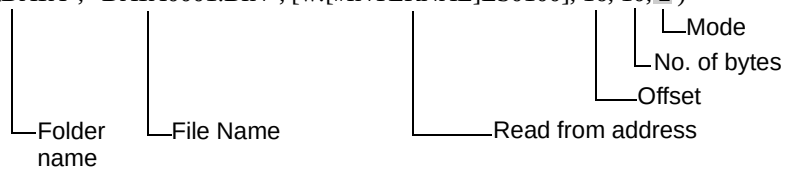
_CF_write (“\DATA”, “DATA0001.BIN”, [w:[#INTERNAL]LS0100], 0, 100, 1)



If the specified file (DATA0001.BIN in the example) already exists and the statement above is executed, 100 bytes of data are read from LS0100 and following areas and added to the DATA0001.BIN file in the \DATA folder.

◆ When “Overwrite” mode is specified (1)

_CF_write (“\DATA”, “DATA0001.BIN”, [w:[#INTERNAL]LS0100], 16, 10, 2)

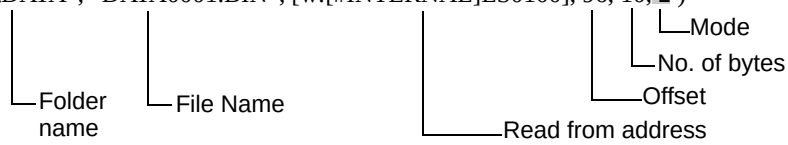


If the specified file (DATA0001.BIN in the example) already exists and the above statement is executed, 10 bytes of data stored in LS0100 and following areas are read and overwrite 10 bytes of data stored in the 17th and following bytes after the offset in the DATA0001.BIN file in the \DATA folder.

◆ When “Overwrite” mode is specified (2)

(The file to be overwritten is less than the sum of the offset value and number of bytes.)

_CF_write (“\DATA”, “DATA0001.BIN”, [w:[#INTERNAL]LS0100], 96, 10, 2)

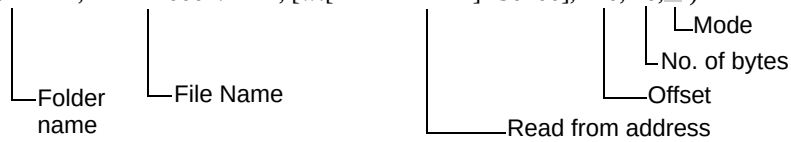


The specified file (DATA0001.BIN in the example) already exists and the file size is 100 bytes. When the offset is set to 96 bytes and the number of bytes is set to 10 bytes for the overwrite operation, 10 bytes of data stored in LS0100 and following areas are read. Then, the first 4 bytes of readout data overwrite the 4 bytes of data stored in the 97th and following bytes in the file, and the remaining 6 bytes of data are added to the end of the file data. The resulting file contains 106 bytes of data.

◆ When “Overwrite” mode is specified (3)

(The file to be overwritten is smaller than the offset value.)

_CF_write (“\DATA”, “DATA0001.BIN”, [w:[#INTERNAL]LS0100], 110, 10, 2)



The specified file (DATA0001.BIN in the example) already exists and the file size is 100 bytes. When the offset is set to 110 bytes and the number of bytes is set to 10 bytes for the overwrite operation, the area between the 101st byte and 110th bytes is filled with 0s and the 10 bytes of data read from LS0100 and following areas are written in the 111th and following bytes. The resulting file contains 120 bytes of data.

IMPORTANT

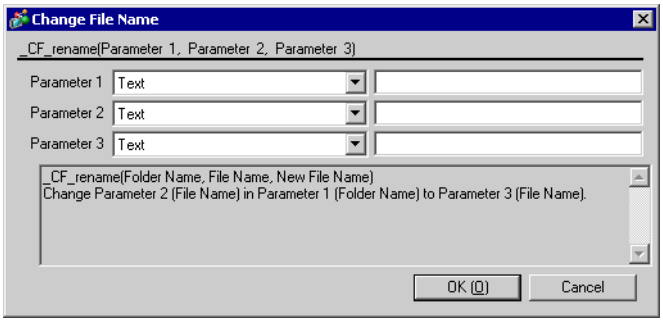
- The maximum allowable number of characters for the first parameter “Folder name” and the second parameter “File name” is 32 single-byte characters.
- The Internal Device can be specified for the second parameter “File name” Specifying the Internal Device allows the indirect addressing of a file name. Also, up to 32 single-byte characters can be used to specify a file name.
  Example: _CF_write (“\DATA” [w:LS0100], [w:[#INTERNAL]LS0200], 0, 100, 0)
  Storing a file name in LS0100 allows indirect addressing of a file name. In this example, a file name is stored in LS0100 through LS0106 as follows.

	16 bit	
LS0100	‘D’	‘A’
LS0101	‘T’	‘A’
LS0102	‘0’	‘0’
LS0103	‘0’	‘1’
LS0104	‘.’	‘B’
LS0105	‘I’	‘N’
LS0106	‘\0’	‘\0’
	:	

The end of the file name must be a NULL character. The display device recognizes the data before the NULL character as the file name.

  In the example above, 100 bytes of data are read from LS0200 and following areas and a new file, “\DATA\DATA0001.BIN”, is created for storing the data.
- As for the file name, only the “8.3 format” (a maximum of 12 characters, with 8 characters for the file name and 3 characters for the extension) may be used. Long file names cannot be used.

21.5.3 Change File Name

Item	Description
Summary	Modifies the file name. Parameter 1 designates the CF-Card data folder. Parameter 2 designates the original file name. Parameter 3 designates the new name.
Format	_CF_rename (Folder Name, File Name, New File Name) The file name can also be designated indirectly with the LS Address.  Parameter 1 Folder name: Fixed text Parameter 2 File name: Fixed text, Internal device, Internal device + Temporary address Parameter 3 File name: Fixed text, Internal device, Internal device + Temporary address

Example expression:

_CF_rename (“\DATA”,“DATA0001.BIN”,“DATA1234.BIN”)

The example above changes the file name from “\DATA\DATA0001.BIN” to “\DATA\DATA1234.BIN”.

IMPORTANT

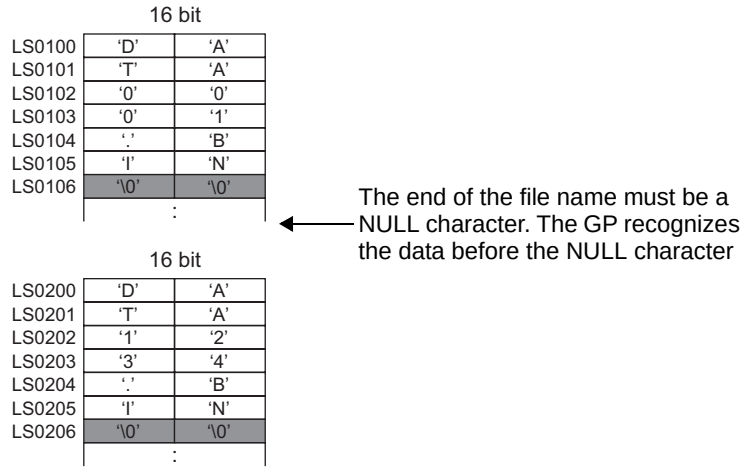
- As for the file name, only the “8.3 format” (a maximum of 12 characters, with 8 characters for the file name and 3 characters for the extension) may be used. Long file names cannot be used.
- The maximum allowable number of characters for the first parameter “Folder name” and the second parameter “File name” is 32 single-byte characters.
- The Internal Device can be specified for the second and third parameter’s “File names”. Specifying the Internal Device allows the indirect addressing of a file name. Also, up to 32 single-byte characters can be used to specify a file name.

  Example:

  _CF_rename (“\DATA”, [w:[#INTERNAL]LS0100], [w:[#INTERNAL]LS0200])

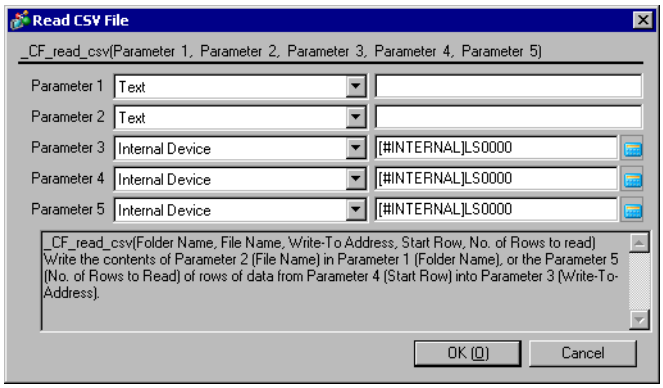
  Storing the file name in LS0100 and LS0200 enables indirect addressing of the file name.

- Store the file names in LS0100 through LS0106 as follows:



  In the example above, the name of the “\DATA\DATA0001.BIN” file is changed to “\DATA\DATA1234.BIN”.
- When the LS Area is specified for “File name”, it is not counted as a D-Script Address.
- To specify a root folder (directory), specify “ “ (empty string) as the folder name.
- To specify a full path for a file name, specify “*” (asterisk) as the folder name.

21.5.4 Read CSV File

Item	Description
Summary	Reads data in cell units from a CSV file (constructed from a cell image delimited with ","), and writes it to a word address.
Format	_CF_read_csv (Folder Name, File Name, [Write-To Address], Start Row, No. of Rows to read)  Parameter 1: Text (Up to 32 single-byte characters) Parameter 2: Text (Up to 32 single-byte characters), Internal Device, Internal Device + Temporary address Parameter 3: Internal Device, Internal Device designated with offset Parameter 4: Numeric Value (1 to 65,535), Internal Device, Temporary variable Parameter 5: Numeric Value (1 to 65,535), Internal Device, Temporary variable

Example expression:

_CF_read_csv ("\CSV", "SAMPLE.CSV", [w:[#INTERNAL]LS1000], 1, 2)

(When reading two lines of data, starting from the first line of the [\CSV\SAMPLE.CSV] file in the CF memory card using the "_CF_read_csv ( )" function.)

SAMPLE.CSV

```
001,"DAT01-01" ," DAT01-2"
002,"DAT02-01" ,"DAT02-2"
```

Reads two lines of data, starting from the first line of the CSV file. When the first character is a numerical value ("0" to "9" "_"), the data is stored as a numerical value. When the first character is ["], the data is treated as a character and "00h" is stored at the end of the text string. For example, when storing "DAT01-01" the data size is 8 characters, which is an even number, and a total of five words are used: Four words are used for storing the text string, and one word is used for storing "00h" at the end. For example, when storing "DAT01-2" the data size is 7 characters, which is an odd number, and a total of 4 words are used to store the text, with "00h" stored at the end.

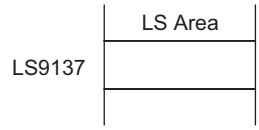
16 bit

LS1000	1	
+1	'D'	'A'
+2	'T'	'0'
+3	'1'	'-'
+4	'0'	'1'
+5	00h	00h
+6	'D'	'A'
+7	'T'	'0'
+8	'1'	'-'
+9	'2'	00h
LS1010	2	
+1	'D'	'A'
+2	'T'	'0'
+3	'2'	'-'
+4	'0'	'1'
+5	00h	00h
+6	'D'	'A'
+7	'T'	'0'
+8	'2'	'-'
+9	'2'	00h

When the Data Storage Mode is 0

NOTE
- When the first character in the cell is a numerical value (“0” to “9”, “_”), it converts the value to numerical data and then writes the data to the LS device. The allowed range is from -32,768 to 32,767.
- When the first character in the cell is [”], it writes the range with [”] to the LS device as text string data. When the size of the text string data is an odd number of bytes, “0x00” is appended to the end. When the size of the text data is an even number of bytes, “0x0000” is written to the address following the last address. Up to 32 single-byte characters can be entered in one cell.
- When a CSV file has two or more lines of data, the desired number of lines can be read out starting from the specified line. Up to 200 single-byte characters can be entered in a line, and up to 65,535 lines can be entered in a CSV file.
- When an error occurs, the error status is written to LS9137.
- When writing CSV file text data to the LS device, the data storage order depends on the data storage mode.

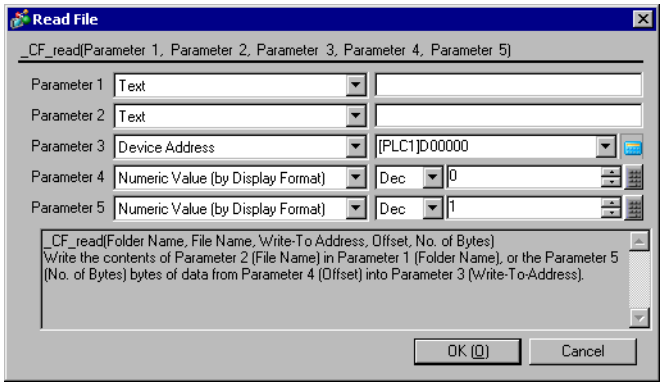
Error Status



Editor Function Name	LS Area	Error Status	Cause
_CF_read_csv ()	LS9137	0000h	Completed Successfully
		0001h	Parameter error
		0002h	CF-Card Error (No CF-Card/File Open Error/ File Read Error)
		0003h	Write/Read error

IMPORTANT
- When “ ∗ ” is specified for the folder name, the full path can be designated for the file name.
- Only the 8.3 format (Up to 12 characters total: 8 characters for the file name and 3 characters for the extension) can be used for the file name. A file name longer than this format cannot be used.
- The effective LS device area for storing data imported from a CSV file is limited to the designated user area (LS20 to LS2031 and LS2096 to LS8191).
- The processing time required for importing data is proportional to the data volume of the CSV file to be read out. Note that Parts will not be refreshed until processing is completed. (It takes approximately 10 seconds to read the data from the first to the 100th line of a CSV file containing 100 lines, with 40 characters per line.)
- Unlike the [_CF_read ( )] function, the status will not be saved to [s:CF_ERR_STAT] immediately after the function is executed. (In some cases, undefined values may be stored.)
- Be sure to insert [”] at the beginning and end of text strings that start with a numeral.
  (Example)
  [123, 2-D4EA] [123, “2-D4EA”]
  × OK

21.5.5 Read File

Item	Description
Summary	Reads the specified number of bytes of data in the file after the specified offset and writes it in the destination address. See the “Data Storage Mode” section below for more details about data storage order.
Format	_CF_read (Folder Name, File Name, Write-To Address, Offset, No. of Bytes)  Parameter 1 Folder name: Fixed string (Maximum length: 32 single-byte characters) Parameter 2 File name: Fixed string, Internal Device, Internal Device + Temporary address (Maximum length: 32 single-byte characters) Parameter 3 Write-To Address: Device Address, Device Address + Temporary address Parameter 4 Offset: Numeric Value, Device address, Temporary address (Maximum number that can be specified: 65,535 for 16-bit length, 4,294,967,295 for 32-bit length) Parameter 5 Number of bytes: Numeric Value, Device address, Temporary address (Maximum length: 1280)

Example expression:

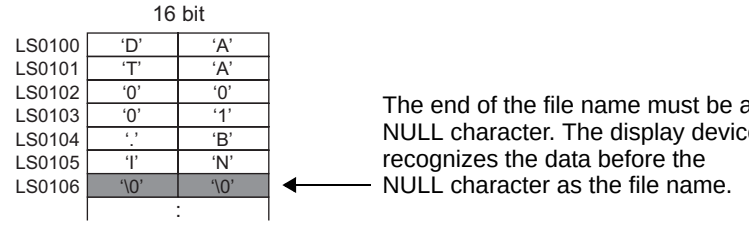
To read 16 bytes of data in the specified file when the offset is 16:

_CF_read (“\DATA”, “DATA0001.BIN”, [w:[#INTERNAL]LS0100], 16, 16)

In the example above, the 16 bytes of data from the 17th and later bytes in the “\DATA\DATA0001.BIN” file are written to LS0100 and later areas.

IMPORTANT

- As for the file name, only the “8.3 format” (a maximum of 12 characters, with 8 characters for the file name and 3 characters for the extension) may be used. Long file names cannot be used.
- The maximum allowable number of characters for the first parameter “Folder name” and the second parameter “File name” is 32 single-byte characters.
- The Internal Device can be specified for the second parameter “File name”. Specifying the Internal Device allows the indirect addressing of a file name. Also, up to 32 single-byte characters can be used to specify a file name.
  Example:
  To read 10 bytes of data stored in a file when the file is specified in LS0100 and later and the offset is 0:
  _CF_read (“\DATA”, [w:LS0100], [w:LS0200], 0, 10)
  Storing a file name in LS0100 allows indirect addressing of a file name. In this example, a file name is stored in LS0100 through LS0106 as follows.

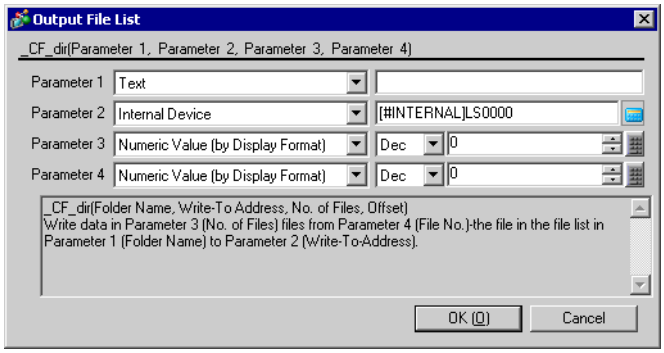


  In the example above, 10 bytes of data at the beginning of the “\DATA\DATA0001.BIN” file are read and written into LS0200 and later areas.

- The number of bytes that are successfully read is written in CF-Card Readout Bytes [s:CF_READ_NUM]. For more details, refer to “21.5.1 Label Settings ■ CF-Card Error Status” (page 21-38) .
- The internal device designated in “File Name” and the “Write-To Address” are not counted as D-Script Addresses.
- When a PLC device is specified for the Write-To Address, more time is required for writing data to the PLC as the number of words (bytes) increases. Several seconds may be required, depending on the number of words.
- If the data read out from the file exceeds the designated device range of the PLC, a communication error occurs. In this case, you must turn the power to the PLC OFF and ON once to reset the PLC from the error.
- When a PLC device is specified as a destination, the values are not written immediately due to the GP to PLC transmission time.
  Example:
  In the script below, statement (1) reads 10 bytes of data from the file and writes the data into [w:D0100]. The data, however, has not yet been written into [w:[PLC1]D0100] at the execution of statement (2) due to the transmission time.
  _CF_read (“\DATA”, “DATA0001.BIN”, [w:[PLC1]D0100], 0, 10) ..... (1)
  [w:[PLC1]D0200] = [w:[PLC1]D0100] + 1 ..... (2)

  In such a case, store the data once in the LS Area and then execute the second statement, as follows.
  _CF_read (“\DATA”, “DATA0001.BIN”, [w:[PLC1]D0100], 0, 10)
  memcpy ([w:[#INTERNAL]LS0100], [w:[PLC1]D0100], 10)
  [w:[PLC1]D0200] = [w:[#INTERNAL]LS0100] + 1

21.5.6 Output File List

Item	Description
Summary	The list of files that exist in the specified folder is written in the Internal Device. Parameter 1 designates the CF-Card data folder. Parameter 4 designates the offset used to select a file/files within that folder. Parameter 3 designates the number of files selected within that folder. Parameter 2 specifies the LS Area into which the files will be written. When the offset is specified as “0” the list starts from the first (starting) file.
Format	_CF_dir (Folder Name, Write-To Address, No. of Files, Offset)  Parameter 1 Folder name: Fixed text (Maximum length: 32 single-byte characters) Parameter 2 Write-To Address: Internal Device, Internal Device designated with offset Parameter 3 No. of files: Numeric Value, Device address, Temporary address (Maximum length: 32) Parameter 4 Offset: Numeric Value, Device address, Temporary address

Example expression:

To output a file list containing two files when the offset is 1 (second file):

_CF_dir (“\DATA*.*”, [w:[#INTERNAL]LS0100], 2, 1)

When the statement above is executed while the following files exist in the DATA folder, file names “DATA0001.BIN” and “DATA02.BIN” are written to LS0100 and later areas.

Contents of folder

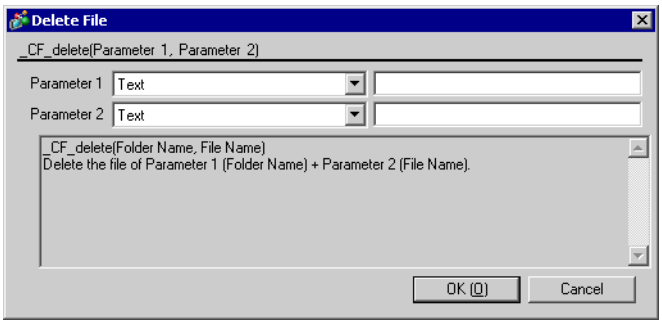
Contents of LS Area

	16 bit		
LS0100	‘D’	‘A’	7 words are used.
LS0101	‘T’	‘A’	
LS0102	‘0’	‘0’	
LS0103	‘0’	‘1’	
LS0104	‘.’	‘B’	
LS0105	‘I’	‘N’	
LS0106	‘\0’	‘\0’	
LS0107	‘D’	‘A’	7 words are used.
LS0108	‘T’	‘A’	
LS0109	‘0’	‘2’	
LS0110	‘.’	‘B’	
LS0111	‘I’	‘N’	
LS0112	‘\0’	‘\0’	
LS0113	‘\0’	‘\0’	

IMPORTANT

- When the offset is specified as “0”, the list starts from the first (starting) file.
- As for the file name, only the “8.3 format” (a maximum of 12 characters, with 8 characters for the file name and 3 characters for the extension) may be used. Long file names cannot be used.
- If the specified folder does not have enough files as specified, the remaining LS Area is filled with NULL characters (‘\0’).
- If a file name has fewer than 12 characters, the empty positions are filled with NULL characters (‘\0’).
- To specify a folder name, be sure to add “∗.∗” (Example “\DATA\∗.∗”). “∗.∗” means to list all files.Just like ∗” make sure you describe “∗.∗”. “∗.∗” means list all the files.
- The number of files actually listed is written in CF-Card Listed Files [s:CF_FILELIST_NUM]. For more details, refer to “ ■ CF-Card Error Status” (page 21-38) .
- Write-To LS Addresses are not counted as D-Script Addresses.
- The file names are not sorted when they are written into the LS Area. They are written in order of creation (the order of FAT entry).
- You can create the list by specifying a file extension. To list the files with a certain extension, use a format such as “\DATA\∗.BIN”. However, you cannot use “∗” within a file name.

21.5.7 Delete File

Item	Description
Summary	Deletes the specified file from the CF-Card. Parameter 1 designates the CF-Card data folder. Parameter 2 designates the name of the file to be deleted.
Format	_CF_delete (Folder Name, File Name) The file name can also be designated indirectly with the LS Address.  Parameter 1 Folder name: Fixed text Parameter 2 File name: Fixed text, Internal device, Internal device + Temporary address

Example expression:

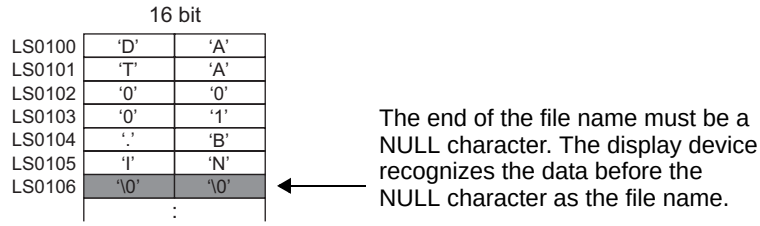
_CF_delete (“\DATA”, “DATA0001.BIN”)

The above example deletes the “\DATA \DATA0001.BIN” file.

IMPORTANT

- As for the file name, only the “8.3 format” (a maximum of 12 characters, with 8 characters for the file name and 3 characters for the extension) may be used. Long file names cannot be used.
- The maximum allowable number of characters for the first parameter “Folder name” and the second parameter “File name” is 32 single-byte characters.
- The Internal Device can be specified for the second parameter “File name”. Specifying the Internal Device allows the indirect addressing of a file name. Also, up to 32 single-byte characters can be used to specify a file name.

  In this example, a file name is stored in LS0100 through LS0106 as follows.



  In the example above, the “\DATA\DATA0001.BIN” file is deleted.
- To specify a root folder (directory), specify “ ” (empty string) as the folder name.
- When the LS Area is specified for “File name”, “Write-To Addresses” are not counted as D-Script Addresses.
- To specify a full path for a file name, specify “*” (asterisk) as the folder name.

21.6 Printer Operation

Printer Operation	Function Summary
	Label Settings ☞ "21.6.1 Label Settings" (page 21-56) Designated from the Control and Status variables. Send ☞ "21.6.2 Send" (page 21-58) Outputs the designated number of bytes to the COM port.

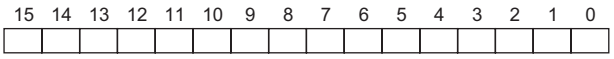
IMPORTANT
- COM1 or USB/PIO (USB-PIO) are ports which can be used as a Printer Operation Function.

21.6.1 Label Settings

■ Control

Control (PRN_CTRL) is a variable to clear the Send Buffer and the Error Status. This variable is write-only.

- Control (PRN_CTRL) Summary



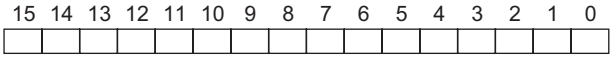
Bit	Content
15	Reserved
14	
13	
12	
11	
10	
9	
8	
7	
6	
5	
4	
3	
2	1: Clear error
1	Reserved
0	1: Clear Send buffer

IMPORTANT
- When a word is designated (when two or more bits are set simultaneously), the processing will be executed in the following order:
  Clear error
  ↓
  Clear send buffer
- The reserved bits may be used in the future; therefore, designate only the bits that are required.

■ Status

The status variable (PRN_STAT) is used in order to check for the presence/absence of data in the Send Buffer and to get the Error Status. This status variable is write-only.

- Contents of Status Variable (PRN_STAT)

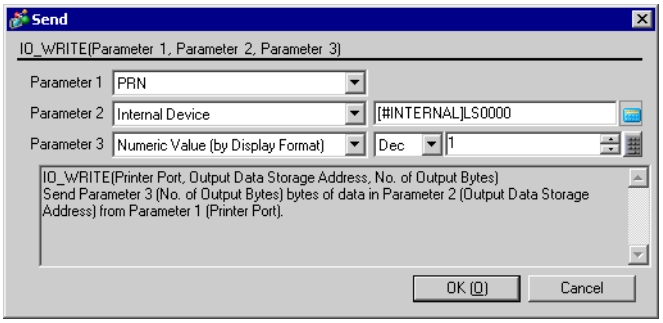


Bit	Content
15	Reserved
14	The status of the Printer I/F ERROR signal Printer Error (Input): 0: Error 1: Normal
13	The status of the Printer I/F SLCT signal Select (Input): 0: Offline 1: Online
12	The status of the Printer I/F PE signal Paper Empty (Input): 0: Normal 1: Paper Empty
11	Reserved
10	
9	
8	
7	
6	
5	
4	
3	
2	
1	0: Normal 1: Send error
0	0: Data exists in Send buffer 1: Send buffer is empty

IMPORTANT

- If the Send buffer overflows, it will result in an error. When this error occurs, the transmission error bit turns ON.
- The Send buffer is 8,192 bytes.
- The reserved bits may be assigned in the future. Therefore, be sure to check only the necessary bits.

21.6.2 Send

Item	Description
Summary	Outputs the designated number of bytes to the COM port. The data is output regardless of the printer type specified.
Format	IO_WRITE ([p:PRN], Output Data Storage Address, No. of Output Bytes)  Parameter 1: [p:PRN] Parameter 2: Internal Device Parameter 3: Integer value, Device address, Temporary address

IMPORTANT
- The maximum numerical value that can be specified for Parameter 3 is 1024. Even when values larger than 1024 are specified, only 1024 bytes of data are output from the COM port.

Example expression 1:

IO_WRITE ([p:PRN], [w:[#INTERNAL]LS1000], 10)

In the example above, 10 bytes of data stored in LS1000 and later areas are output from the COM port.

Example expression 2:

IO_WRITE ([p:PRN], [w:[#INTERNAL]LS1000], [w:[#INTERNAL]LS0800])

In the example above, the data stored in LS1000 and later areas are output from the COM port. The number of bytes is that same as that written in LS0800.

Example expression 3:

IO_WRITE ([p:PRN], [w:[#INTERNAL]LS 1000], [t:0010])

In the example above, the data stored in LS1000 and later areas are output from the COM port. The number of bytes is that same as that written in the Temporary address [t:0010].

Data Storage Mode

When data is read from device addresses upon execution of the COM Port Operation function, you can specify the storage order of the readout data.
Setting the data storage mode in LS9130 can change the storage order.
The mode can be selected from four options: 0, 1, 2 and 3.

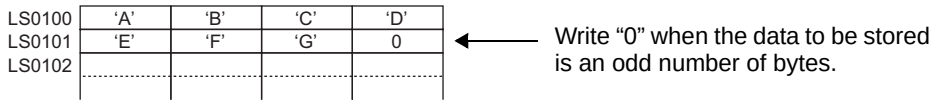
◆ Mode 0

Example: When the COM Port Operation function is used to read the string “ABCDEFG” from a device address

[w:[#INTERNAL]LS9130] = 0
IO_WRITE ([p:PRN], [w:[#INTERNAL]LS1000], 7)

- When the device address length is 16 bits

- When the device address length is 32 bits



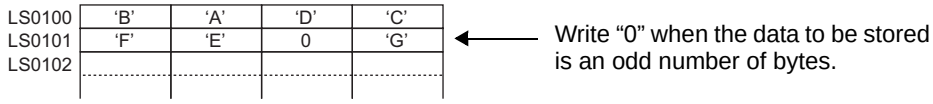
◆ Mode 1

Example: When the COM Port Operation function is used to read the string “ABCDEFG” from a device address

[w:[#INTERNAL]LS9130] = 1
IO_WRITE ([p:PRN], [w:[#INTERNAL]LS1000], 7)

- When the device address length is 16 bits

- When the device address length is 32 bits



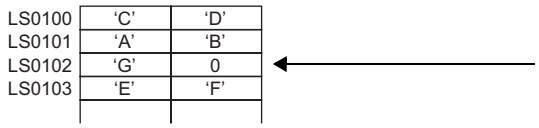
◆ Mode 2

Example: When the COM Port Operation function is used to read the string “ABCDEFG” from a device address

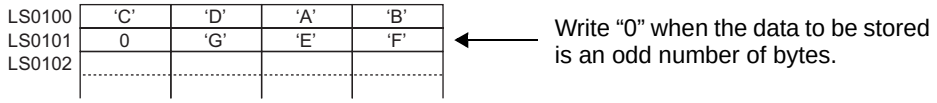
[w:[#INTERNAL]LS9130] = 2

IO_WRITE ([p:PRN], [w:[#INTERNAL]LS1000], 7)

- When the device address length is 16 bits



- When the device address length is 32 bits



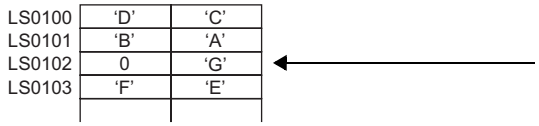
◆ Mode 3

Example: When the COM Port Operation function is used to read the string “ABCDEFG” from a device address

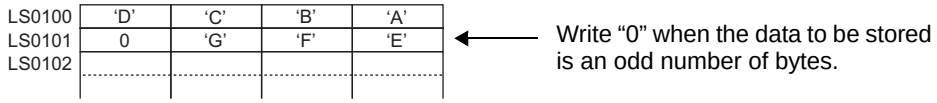
[w:[#INTERNAL]LS9130] = 3

IO_WRITE ([p:PRN], [w:[#INTERNAL]LS1000], 7)

- When the device address length is 16 bits



- When the device address length is 32 bits

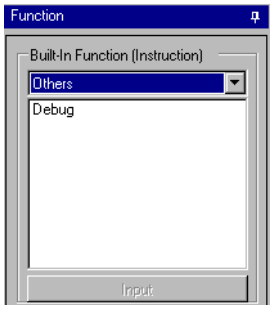


IMPORTANT

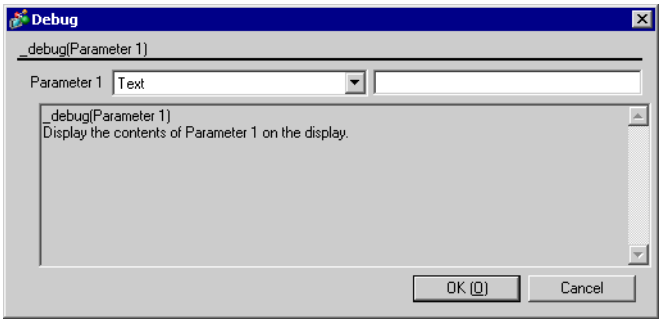
- The data storage mode is not the same as the string data mode in the system setting. The relationship with the string data mode is shown in the following table.

Data Device Storage Order	Byte in Word LH/HL Storage Order	LH/HL Storage Order In Double Word	D-Script data storage mode	Text Data Mode
Store from Top Data	HL Order	HL Order	0	1
	LH Order		1	2
	HL Order	LH Order	2	5
	LH Order		3	4
Store from Last Data	HL Order	HL Order	—	3
	LH Order		—	7
	HL Order	LH Order	—	8
	LH Order		—	6

21.7 Others

Others	Function Summary
	Debug Function ☞ "21.7.1 Debug Function" (page 21-61) Displays the designated address or text on the screen to debug it.

21.7.1 Debug Function

Item	Description
Summary	Displays the designated address or text on the screen to debug it. After you finish debugging, remove the check mark next to the script editor's [Enable Debug Function] and the debug function will disappear from the screen.
Format	_debug (Parameter 1)  Parameter 1: Text (Up to 32 single-byte characters)

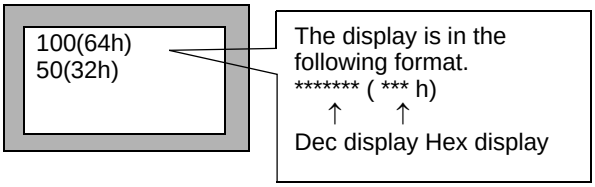
■ Contents of Parameter 1

Parameter 1	Format	Description
Text	_debug (“ABC”)	Displays the text inside “ ”. The text can be up to 32 single-byte characters.
Word Address or Temporary Address	_debug (w:D1000)	Displays the value of the set Word Address or Temporary Address.
Line Feed	_debug (_CRLF)	Moves the cursor to the start of the next line.
Carriage Return	_debug (_CR)	Moves the cursor to the start of the same line.

Example expression 1:

The following script displays the value of the Word Address.

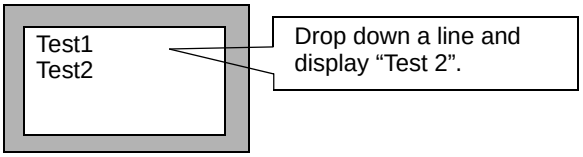
```
[w:[#INTERNAL]LS0100]=100
_debug ([w:[#INTERNAL]LS0100])
_debug (_CRLF)
[w:[#INTERNAL]LS0100]=50
_debug ([w:[#INTERNAL]LS0100])
```



Example expression 2:

The following script displays a line feed and text.

```
_debug (“Test1”)
_debug (_CRLF)
_debug (“Test2”)
```



21.8 Description Expression

Description Expression	Function Summary
Description Expression if - endif if - else - endif loop - endloop break return	if - endif ☞ "21.8.1 if - endif" (page 21-63) When a condition enclosed with brackets "( )" following "if" becomes true, the process following the "if ( )" statement is executed.
	if - else - endif ☞ "21.8.2 if - else - endif" (page 21-63) When a condition enclosed with brackets "( )" following "if" becomes true, the process following the "if ( )" statement is executed. When the condition is false, the statement after "else" is executed.
	loop - endloop ☞ "21.8.3 loop - endloop" (page 21-64) Loop (repetitive) processing is repeated according to the number stored in the temporary Addresses designated in the brackets "( )" following "loop".
	break ☞ "21.8.4 break" (page 21-66) Halts loop operation while the loop ( ) equation is being executed.
	return ☞ "21.8.5 return" (page 21-66) Executes again from the beginning. It can only be used in an Extended Script.

21.8.1 if - endif

When a condition enclosed with brackets "( )" following "if" becomes true, the process following the "if ( )" statement is executed.

NOTE • The Assign "=" character cannot be used in a conditional expression.

21.8.2 if - else - endif

When a condition enclosed with brackets "( )" following "if" becomes true, the process following the "if ( )" statement is executed. When the condition is false, the statement after "else" is executed.

NOTE • The Assign "=" character cannot be used in a conditional expression.

21.8.3 loop - endloop

Loop (repetitive) processing is repeated according to the number stored in the temporary Addresses designated in the brackets “( )” following “loop”.

Infinite Loop

The loop operation is set to infinite loop when no statement is entered in the bracket ( ) for the “loop ( )” statement.
An infinite loop can only be used in an Extended Script.

Example expression:

```
loop ( )
{
    [w:[#INTERNAL]LS0100]=[w:[#INTERNAL]LS0100]+1
    if ( [w:[#INTERNAL]LS0100] >10)
    {
        break
    }
    endif
}
endloop
```

NOTE

- The loop ( ) format is as follows:
  Example:

```
loop (number of loops)<= Designates the temporary Address where the loop
    {                    repetition number is designated.
        Action equation
        break            <= Stated when escaping from the loop halfway (can
                            be omitted)
    } endloop            <= Stated at the end of the loop
```

- Only a temporary Word Address can be entered (in the parentheses). (Example: loop ([t:000]))
- “loop ( )” cannot be used for a trigger equation.
- The temporary Word Address value used to designate the “infinite loop” will decrease every time loop operation is performed. When the value changes to 0, the loop’s operation is finished. If the temporary Word Address value designated for the “infinite loop” is modified, the loop will become endless. Also, the temporary Word Address used is designated as Global. Therefore, simultaneously using this temporary Word Address for another item means the loop’s operation may be performed forever.
- Until loop operation finishes, screen displays of Parts, etc. will not be updated/ refreshed.

Continued

NOTE

- loop ( ) can also be nested. When it is nested, the inner-most loop ( ) will be skipped via the “break” command.

```
loop ([t:0000])        // loop 1
{
   loop ([t: 0001])    // loop2
   {
   break               // Escape from loop2
   }endloop

   break               // Escape from loop1
}endloop
```

- If loop operation is finished without using the escape command, the temporary Word Address value becomes 0.
- The range available for the temporary Word Address value will differ depending on the data format (Bin, BCD), bit length, and code +/– used. If code +/– has been designated and the temporary Word Address becomes a negative value, the condition is judged at the beginning of the loop and the loop processing stops.
- DO NOT use a PLC device in the loop formula. Instead, use an address from the GP’s internal LS area’s user area device, or a temporary Word Address. For example, the following description performs data write to the PLC many times in a short period (100 times in the following example). This can cause a system error since communication processing (the time required to write to the PLC) cannot be performed at this speed.

  Example)

```
[t:0000] = 100                                          // Loop Count: 100
loop ([t:0000])
{
  [w:[PLC1]D0200] = [w:[#INTERNAL]LS0100]               // Write to D0200
  [w:[#INTERNAL]LS0100] =
  [w:[#INTERNAL]LS0100] + 1                             // Increment LS0100
}endloop
```

  Please change as follows:

```
[t:0000] = 100                                          // Loop Count: 100
loop ([t:0000])
{
  [w:[#INTERNAL]LS0200] =
  [w:[#INTERNAL]LS0100]                                 // Write to D0200
  [w:[#INTERNAL]LS0100] =
  [w:[#INTERNAL]LS0100] + 1                             // Increment LS0100
}endloop
[w:[PLC1]D0200]=[w:[#INTERNAL]LS0200]                   //Write LS0200 data to D0200
                                                        Write
```

- Using “loop” or “break” as a function name for a D-Script function will cause an error.

21.8.4 break

Halts loop operation while the loop ( ) equation is being executed.

NOTE • The “break” command can be used only in the { } section of loop ( ).

21.8.5 return

When the “User Defined Function” includes “return”
The processing of the Function is terminated and the control returns to the caller of the Function.

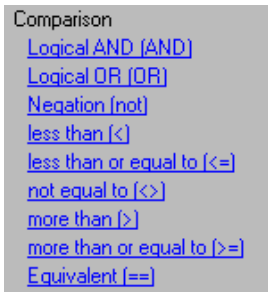
When Execution (main Function) includes “return”
The processing of the main Function is aborted for the moment, and is restarted from the start of the main Function.

NOTE • The Assign “=” character cannot be used in a conditional expression.

Example expression:

```
[w:[#INTERNAL]LS0100]=([w:[#INTERNAL]LS0200]>> 8) & 0xFF
if ([w:[#INTERNAL]LS0100]==0)     //  When LS0100 is “0”, processing is no longer
                                      executed
{
    set([b:[#INTERNAL]LS005000])  //  Sets the bit address for error display
    return                        //  End
}
endif
```

21.9 Comparison

Comparison	Function Summary
	Logical AND (AND) ☞ "21.9.1 Logical AND (AND)" (page 21-67) N1 and N2: True if both N1 and N2 are ON.
	Logical OR (OR) ☞ "21.9.2 Logical OR (OR)" (page 21-67) N1 or N2: True if either N1 and N2 are ON.
	Negation (not) ☞ "21.9.3 Negation (not)" (page 21-68) notN1: Becomes 0 if N1 is 1, and 1 if N1 is 0.
	less than (<) ☞ "21.9.4 Less than (<)" (page 21-68) True if N1 is less than N2 (N1 < N2).
	less than or equal to (<=) ☞ "21.9.5 Less than or equal to (<=)" (page 21-68) True if N1 is less than or equal to N2 (N1 <= N2).
	not equal to (<>) ☞ "21.9.6 Not equal to (<>)" (page 21-68) True if N1 is not equal to N2 (N1 <> N2).
	more than (>) ☞ "21.9.7 Greater than (>)" (page 21-68) True if N1 is more than N2 (N1 > N2).
	more than or equal to (>=) ☞ "21.9.8 Greater than or equal to (>=)" (page 21-68) True if N1 is more than or equal to N2 (N1 >= N2).
	Equivalent (==) ☞ "21.9.9 Equal to (==)" (page 21-68) True if N1 is equal to N2 (N1 = N2).

21.9.1 Logical AND (AND)

ANDs the right and left sides. Value 0 (zero) is regarded as OFF, and other values as ON.
N1 and N2: True if both N1 and N2 are ON. Otherwise false.

21.9.2 Logical OR (OR)

ORs the right and left sides. Value 0 (zero) is regarded as OFF, and other values as ON.
N1 or N2: True if either N1 and N2 are ON. Otherwise false.

21.9.3 Negation (not)

NOTs the right side. Value 0 (zero) is regarded as 1, and other values as 0.
notN1: Becomes 0 if N1 is 1, and 1 if N1 is 0.

21.9.4 Less than (<)

Compares the data in two word addresses, or the data in a word address and a constant.
True if N1 is less than N2 (N1 < N2).

21.9.5 Less than or equal to (<=)

Compares the data in two word addresses, or the data in a word address and a constant.
True if N1 is less than or equal to N2 (N1 <= N2).

21.9.6 Not equal to (<>)

Compares the data in two word addresses, or the data in a word address and a constant.True if N1 is not equal to N2 (N1 <> N2).

21.9.7 Greater than (>)

Compares the data in two word addresses, or the data in a word address and a constant.
True if N1 is more than N2 (N1 > N2).

21.9.8 Greater than or equal to (>=)

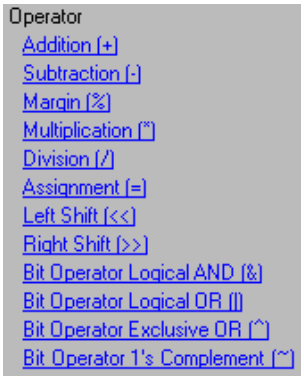
Compares the data in two word addresses, or the data in a word address and a constant.
True if N1 is more than or equal to N2 (N1 >= N2).

21.9.9 Equal to (==)

Compares the data in two word addresses, or the data in a word address and a constant.
True if N1 is equal to N2 (N1 = N2).

Command		Example
Conjunction	and	if ((Operation) and (Operation))
Disjunction	or	if ((Operation) or (Operation))
Negation	not	if (not (Operation))
less than	<	(Term 1) < (Term 2)
less than or equal to	<=	(Term 1) <= (Term 2)
not equal to	<>	(Term 1) <> (Term 2)
more than	>	(Term 1) > (Term 2)
more than or equal to	>=	(Term 1) >= (Term 2)
Equivalent	==	(Term 1) == (Term 2)

21.10 Operator

Operator	Function Summary
	Addition (+) ☞ "21.10.1 Addition (+)" (page 21-70) Adds the data in two word addresses, or the data in a word address and a constant.
	Subtraction (-) ☞ "21.10.2 Subtraction (-)" (page 21-70) Subtracts the data in two word addresses, or the data in a word address and a constant.
	Margin (%) ☞ "21.10.3 Modulus (%)" (page 21-70) Detects a remainder of a division performed on the data in two word addresses, or the data in a word address and a constant.
	Multiplication (*) ☞ "21.10.4 Multiplication (*)" (page 21-70) Multiplies the data in two word addresses, or the data in a word address and a constant.
	Division (/) ☞ "21.10.5 Division (/)" (page 21-70) Performs division the data in two word addresses, or the data in a word address and a constant.
	Assignment (=) ☞ "21.10.6 Assignment (=)" (page 21-70) Assigns the right side value to the left side.
	Left Shift (<<) ☞ "21.10.7 Left Shift (<<)" (page 21-70) Shifts the data on the left side to the left by the number on the right side.
	Right Shift (>>) ☞ "21.10.8 Right Shift (>>)" (page 21-71) Shifts the data on the left side to the right by the number on the right side.
	Bit Operator Logical AND (&) ☞ "21.10.9 Bitwise AND (&)" (page 21-71) Performs logical AND of data between word devices, or between word device data and constant.
	Bit Operator Logical OR (|) ☞ "21.10.10 Bitwise OR (|)" (page 21-71) Performs logical OR of data between word devices, or between word device data and constant.
	Bit Operator Exclusive OR (^) ☞ "21.10.11 Bitwise Exclusive OR (^)" (page 21-71) Performs exclusive OR of data between word devices, or between word device data and constant.
	Bit Operator 1's Complement (~) ☞ "21.10.12 Bitwise 1's Complement (~)" (page 21-71) Inverts the bits.

21.10.1 Addition (+)

Adds the data in two word addresses, or the data in a word address and a constant. Any overflowing digits resulting from the operation are rounded.

21.10.2 Subtraction (-)

Subtracts the data in two word addresses, or the data in a word address and a constant. Any overflowing digits resulting from the operation are rounded.

21.10.3 Modulus (%)

Detects a remainder of a division performed on the data in two word addresses, or the data in a word address and a constant. The operation result may depend on the sign of the left and right sides.

21.10.4 Multiplication (*)

Multiplies the data in two word addresses, or the data in a word address and a constant. Any overflowing digits resulting from the operation are rounded.

21.10.5 Division (/)

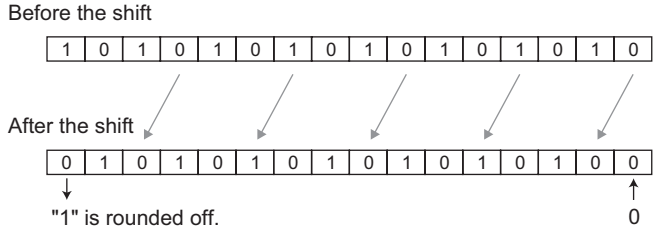
Performs division the data in two word addresses, or the data in a word address and a constant. Decimal places resulting from the operation are rounded. Any overflowing digits resulting from the operation are rounded.

21.10.6 Assignment (=)

Assigns the right side value in the left side. The left side can state a device address only. The right side can describe both a device address and a constant. Any overflowing digits resulting from the operation are rounded.

21.10.7 Left Shift (<<)

Shifts the data on the left side to the left by the number on the right side. This feature supports logical shifts only.
(Example: Left Shift operation (Shifts to the left by one bit.)



21.10.8 Right Shift (>>)

Shifts the data on the left side to the right by the number on the right side. This feature supports logical shifts only.

21.10.9 Bitwise AND (&)

Performs logical AND of data between word devices, or between word device data and constant. Used to extract a specific bit or to mask a specific string of bits.

21.10.10 Bitwise OR (|)

Performs logical OR of data between word devices, or between word device data and constant. Used to turn ON a specific bit.

21.10.11 Bitwise Exclusive OR (^)

Performs exclusive OR of data between word devices, or between word device data and constant.

21.10.12 Bitwise 1's Complement (~)

Inverts the bits.

NOTE

- For information about rounding decimal numbers or overflowing digit caused by operation results, see .

☞ "20.9.4 Notes on Operation Results" (page 20-58)

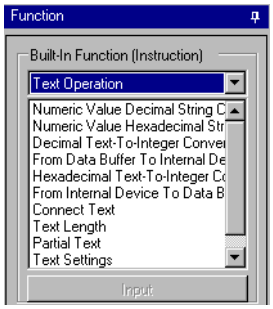
Priority and Associativity

The following table shows the priority of the trigger conditions. If two or more operators have the same priority, follow the direction shown by the associativity.

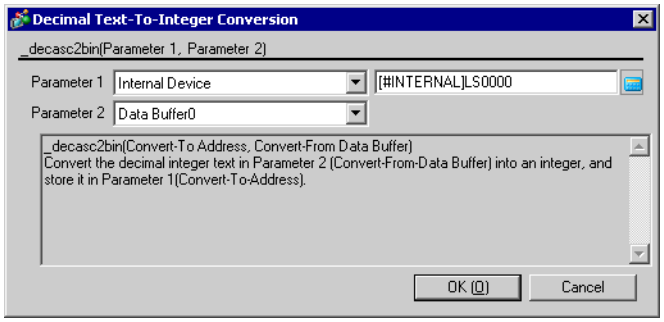
Priority	Operator	Associativity
High	( )	→
	not ~	←
	* / %	→
	+ –	→
	<< >>	→
	< <= > >=	→
	== <>	→
	& ^ |	→
	and or	→
Low	=	←

21.11 Text Operation

Text Operation functions can only be used in an Extended Script.

Text Operation	Function Summary
	Decimal Text-To-Integer Conversion ☞ "21.11.1 Decimal Text-To-Integer Conversion" (page 21-73) This function is used to convert decimal text to integers.
	Hexadecimal Text-To-Integer Conversion ☞ "21.11.2 Hexadecimal Text-To-Integer Conversion" (page 21-75) This function converts hexadecimal text to integers.
	Copy From Internal Device To Data Buffer ☞ "21.11.3 From Internal Device to Data Buffer" (page 21-77) The data of the string stored in the Internal Device is copied to the data buffer.
	Copy From Data Buffer to Internal Device ☞ "21.11.4 From Data Buffer To Internal Device" (page 21-79) The data of the string stored in the data buffer is copied to the Internal Device.
	Status ☞ "21.11.5 Text Operation Error Status" (page 21-81) Stores any error that has occurred.
	Numeric Value Decimal String Conversion ☞ "21.11.6 Numeric Value Decimal String Conversion" (page 21-82) This function is used to convert an integer to a decimal string.
	Numeric Value Hexadecimal String Conversion ☞ "21.11.7 Numeric Value Hexadecimal String Conversion" (page 21-83) This function is used to convert binary data into a hexadecimal string.
	Partial Text Function ☞ "21.11.8 Partial Text" (page 21-84) Data are retrieved from the specified offset of the string according to the length of the string and stored in another data buffer.
	Text Settings ☞ "21.11.9 Text Settings" (page 21-85) Stores a fixed string in the data buffer.
	Get Text Length ☞ "21.11.10 Text Length" (page 21-86) Obtains the length of the stored string.
	Connect Text ☞ "21.11.11 Connect Text" (page 21-87) Concatenates a character string or character code with the text buffer.

21.11.1 Decimal Text-To-Integer Conversion

Item	Description
Summary	This function is used to convert a decimal string to integers. Convert the decimal integer text in Parameter 2 (Convert-From Data Buffer) into an integer, and store it in Parameter 1 (Convert-To Address).
Format	_decasc2bin ([Convert-To Address], [Convert-From Data Buffer])  Parameter 1: Internal Device, Temporary address Parameter 2: Data Buffer

Example expression 1 (When the data length is 16 bits)

_decasc2bin ([w:[#INTERNAL]LS0100], databuf0)

The content of “databuf0” is as follows:

	8 bit	
databuf0[0]	31h	‘1’
databuf0[1]	32h	‘2’
databuf0[2]	33h	‘3’
databuf0[3]	34h	‘4’
databuf0[4]	00h	NULL

The above data are converted as follows.

	16 bit
LS0100	1234

Example expression 2 (When the data length is 32 bits)

_decasc2bin ([w:[#INTERNAL]LS0100], databuf0)

The content of “databuf0” is as follows:

	8 bit	
databuf0[0]	31h	‘1’
databuf0[1]	32h	‘2’
databuf0[2]	33h	‘3’
databuf0[3]	34h	‘4’
databuf0[4]	35h	‘5’
databuf0[5]	36h	‘6’
databuf0[6]	37h	‘7’
databuf0[7]	38h	‘8’
databuf0[8]	00h	NULL

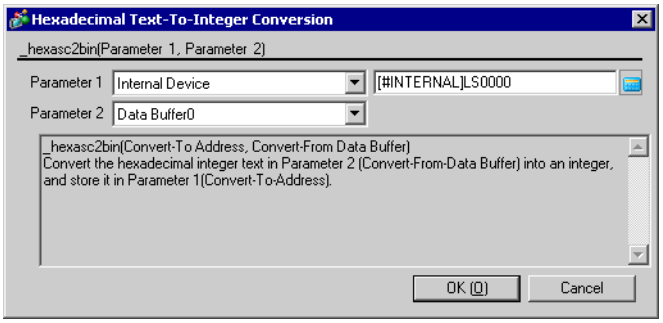
The above data are converted as follows.

	32 bit
LS0100	12345678
LS0102	

IMPORTANT

- An error occurs when the converted bit length is greater than the bit length of the D-Script Editor.
  Example: When the bit length of the script is 16 bits:
  _strset (databuf 0, “123456”) // When a 6-digit decimal string is set accidentally
  _decasc2bin ([w:[#INTERNAL]LS0100], databuf0)
  When the above expression is executed, Error No. 2 (string conversion error) of the String error status [e: STR_ERR_STAT] is triggered. However, the bit returns to the beginning of the Main function when an error occurs. Therefore, you cannot reference other functions directly after _decasc2bin executes. (If the command comes while a function is running, it returns to the line that called that function.)
- An error occurs during conversion of a string of data containing characters other than “0” to “9”.
  Example: When the bit length of the script is 16 bits:
  _strset (databuf0, “12AB”) // When a non-decimal string is set accidentally
  _decasc2bin ([w:[#INTERNAL]LS0100], databuf0)
  When the above expression is executed, Error No. 2 (string conversion error) of the String error status [e: STR_ERR_STAT] is triggered. However, the bit returns to the beginning of the Main function when an error occurs. Therefore, you cannot reference other functions directly after _decasc2bin executes. (If the command comes while a function is running, it returns to the line that called that function.)
- The processing is terminated when an error occurs and returns to the beginning of the Main function. (If the command comes while a function is running, it returns to the line that called that function.)

21.11.2 Hexadecimal Text-To-Integer Conversion

Item	Description
Summary	This function converts a hexadecimal string to binary data. Convert the hexadecimal integer text in Parameter 2 (Convert-From Data Buffer) into an integer, and store it in Parameter 1(Convert-To Address).
Format	_hexasc2bin ([Convert-To Address], [Convert-From Data Buffer])  Parameter 1: Internal Device, Temporary address Parameter 2: Data Buffer

Example expression 1 (When the data length is 16 bits)

_hexasc2bin ([w:[#INTERNAL]LS0100], databuf0)

The content of “databuf0” is as follows:

	8 bit	
databuf0[0]	31h	‘1’
databuf0[1]	32h	‘2’
databuf0[2]	33h	‘3’
databuf0[3]	34h	‘4’
databuf0[4]	00h	NULL

The above data are converted as follows.

	16 bit
LS0100	1234h

Example expression 2 (When the data length is 32 bits)

_hexasc2bin ([w:[#INTERNAL]LS0100], databuf0)

The content of “databuf0” is as follows:

	8 bit	
databuf0[0]	31h	‘1’
databuf0[1]	32h	‘2’
databuf0[2]	33h	‘3’
databuf0[3]	34h	‘4’
databuf0[4]	35h	‘5’
databuf0[5]	36h	‘6’
databuf0[6]	37h	‘7’
databuf0[7]	38h	‘8’
databuf0[8]	00h	NULL

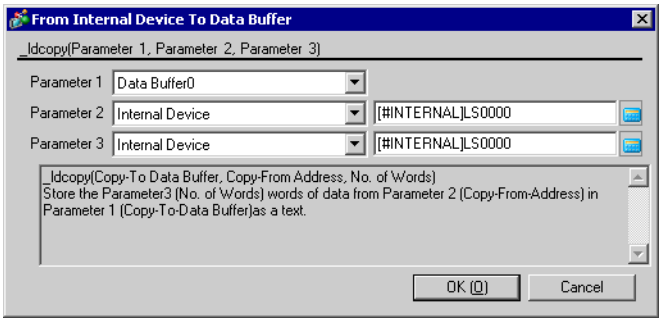
The above data are converted as follows.

	32 bit
LS0100	12345678h
LS0102	

IMPORTANT

- An error occurs when the converted string is greater than 16 bits or 32 bits.
  Example: When the bit length of the script is 16 bits:
  _strset (databuf0, “123456”)
  _hexasc2bin ([w:[#INTERNAL]LS0100], databuf0)
  When the above expression is executed, Error No. 2 (string conversion error) of the String error status [e: STR_ERR_STAT] is triggered.
- An error occurs during conversion of a string of data containing characters other than “0” to “9”, “A” to “F”, or “a” to “f”.
  Example: When the bit length of the script is 16 bits:
  _strset (databuf 0, “123G”)
  _hexasc2bin ([w:[#INTERNAL]LS0100], databuf0)
  When the above expression is executed, Error No. 2 (string conversion error) of the String error status [e: STR_ERR_STAT] is triggered.
- The processing is terminated when an error occurs and returns to the beginning of the Main function. (If the command comes while a function is running, it returns to the line that called that function.)

21.11.3 From Internal Device to Data Buffer

Item	Description
Summary	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. Store the Parameter 3 (No. of Words) words of data from Parameter 2 (Copy-From Address) in Parameter 1 (Copy-To Data Buffer) as a text.
Format	_ldcopy (Copy-To Data Buffer, [Copy-From Address], No. of Words)  Parameter 1: Data Buffer Parameter 2: Internal Device Parameter 3: Integer value, Internal Device, Temporary address (The valid range for Parameter 3 is from 1 to 1,024.)

Example expression 1:

_ldcopy (databuf0, [w:[#INTERNAL]LS0100], 4)

	16 bit
LS0100	31h
LS0101	32h
LS0102	33h
LS0103	34h

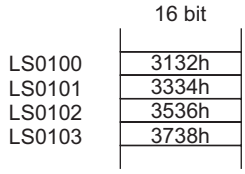
The data in LS0100 to LS0103 is written into the 4 bytes of the data buffer sequentially starting from “databuf0” The LS area is read in each byte (the lowest bits).

	8 bit	
databuf0[0]	31h	‘1’
databuf0[1]	32h	‘2’
databuf0[2]	33h	‘3’
databuf0[3]	34h	‘4’
databuf0[4]	00h	NULL

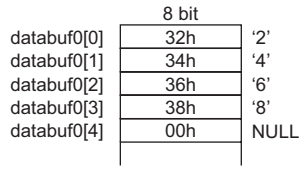
IMPORTANT

- The low 1 byte of the LS area is read out and the specified quantity of data is written into the data buffer.
- The maximum value that can be assigned for Parameter 3 is 1,024. When a value exceeding the limit is set, Error No. 1 (string overflow) of the String error status [e: STR_ERR_STAT] is triggered.
- Even when data are stored in the significant byte in the LS area, only the data in the low 1 byte is read out.
- The processing is terminated when an error occurs and returns to the beginning of the Main function. (If the command comes while a function is running, it returns to the line that called that function.)

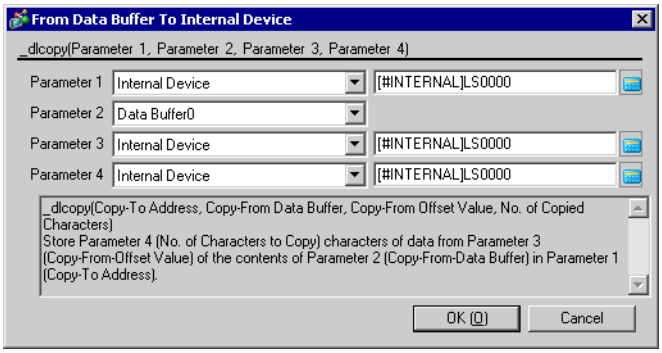
  _ldcopy (databuf0, [w:[#INTERNAL]LS0100], 4)



When data are stored as illustrated above, the data of the low 1 byte is read out and written into the data buffer.



21.11.4 From Data Buffer To Internal Device

Item	Description
Summary	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. Stores Parameter 4 (No. of Characters to Copy) characters of data from Parameter 3 (Copy-From Offset Value) of the contents of Parameter 2 (Copy-From Data Buffer) in Parameter 1 (Copy-To Address).
Format	_dlcopy ([Copy-To Address], Copy-From Data Buffer, Copy-From Offset Value, No. of Copied Characters)  Parameter 1: Internal Device Parameter 2: Data Buffer Parameter 3: Numeric Value, Internal Device, Temporary address (The valid range for Parameter 3 is from 0 to 1,024.) Parameter 4: Numeric Value, Internal Device, Temporary address (The valid range for Parameter 4 is from 1 to 1,024.)

Example expression 1:

_dlcopy ([w:[#INTERNAL]LS0100], databuf0, 2, 4)

	8 bit	
databuf0[0]	31h	'1'
databuf0[1]	32h	'2'
databuf0[2]	33h	'3'
databuf0[3]	34h	'4'
databuf0[4]	35h	'5'
databuf0[5]	36h	'6'
databuf0[6]	37h	'7'
databuf0[7]	38h	'8'

4 bytes of data retrieved from "offset 2" of "databuf0" are written into LS0100 to LS0103. The data are written into the LS area in units of 1 byte.

	16 bit
LS0100	33h
LS0101	34h
LS0102	35h
LS0103	36h

IMPORTANT

- 1 byte of data is read out from the data buffer and written into the LS area. That means only the lowest 8 bits (1 byte) of the LS area will be used. The significant 8 bits (1 byte) will be cleared with “0”.
- When the specified value [source offset value + number of characters to be copied] is greater than the data buffer size, error No. 3 (string extraction error) of the string error status [e: STR_ERR_STAT] is issued.
- The processing is terminated when an error occurs and returns to the beginning of the Main function. (If the command comes while a function is running, it returns to the line that called that function.)

21.11.5 Text Operation Error Status

When an error occurs during execution of text operation, an error is set to the Text Operation Error Status [e: STR_ERR_STAT]. “0” in [e: STR_ERR_STAT] indicates a normal condition, and values other than “0” stored in [e: STR_ERR_STAT] indicate error states. The most recent error is stored in the Text Operation Error Status [e: STR_ERR_STAT]. The Text Operation Error Status can be set up with [SIO Port Operation/Label Settings] under the D-Script Toolbox menu. The following table lists the text operation errors.

Error No.	Error Name	Description
0	Normal	No error
1	Text overflow	A string of at least 256 bytes is directly included in the argument for the following Functions: _strset ( ), _strlen ( ), _strcat ( ), _strmid ( ), and IO_READ_WAIT ( ). Or, a string exceeding the data buffer size is created during execution of the _strcat ( ) or _ldcopy ( ) function. Example: _strcat (databuf0, databuf1) The above function is executed when a string of 1,020 bytes is stored in databuf0, and a string of 60 bytes is stored in databuf1. (A string exceeding 1,024 bytes, the size of the data buffer, results in an error status.)
2	String conversion error	Invalid character code is given to the _hexasc2bin ( ) or _decasc2bin ( ) Function. Example: A character code other than “0” to “9”, “A” to “F”, or “a” to “f” is included in the second argument of _hexasc2bin ( ).
3	String retrieval error	Retrieval of a character string longer than the character string specified with the “_strmid ( )” Function is attempted. Or, an offset value greater than the specified string is designated. Example: _strmid (databuf0, “12345678”, 2, 8) Retrieval of an 8-character string from offset 2 is attempted.

The String Control Error Status cannot be used with D-Scripts and Global D-Scripts. If it is read out accidentally, “0” will be loaded.

It is stored in the Error Status during execution of each function.

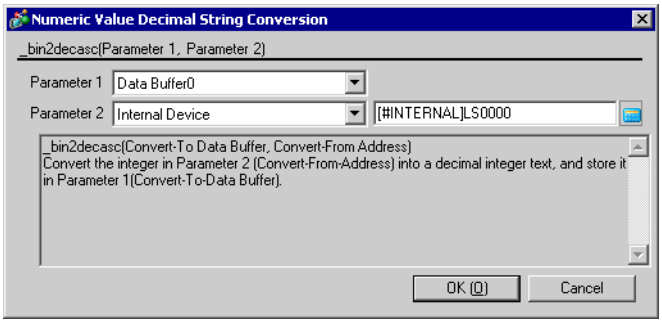
To check the error [e: STR_ERR_STAT], write the following statements. You can confirm the error with the following expression.

Example expression:

```
if ([e:STR_ERR_STAT] <> 0)          // Checks the error status.
{
set([b:[#INTERNAL]LS005000])        // Sets bit on Error Display Lamp
}
endif
```

IMPORTANT
- The processing is terminated when an error occurs and returns to the beginning of the Main function. (If the command comes while a function is running, it returns to the line that called that function.)

21.11.6 Numeric Value Decimal String Conversion

Item	Description
Summary	This function is used to convert an integer to a decimal string. Convert the integer in Parameter 2 (Convert-From Address) into a decimal integer text, and store it in Parameter 1 (Convert-To Data Buffer).
Format	_bin2decasc (Convert-To Data Buffer, [Convert-From Address])  Parameter 1: Data Buffer Parameter 2: Internal Device, Temporary address

Example expression 1 (When the data length is 16 bits)

_bin2decasc (databuf0, [w:[#INTERNAL]LS0100])

	16 bit
LS0100	1234

The above data are converted as follows: Note that “NULL (0x00)” is added.

	8 bit	
databuf0[0]	31h	‘1’
databuf0[1]	32h	‘2’
databuf0[2]	33h	‘3’
databuf0[3]	34h	‘4’
databuf0[4]	00h	NULL

Example expression 2 (When the data length is 32 bits)

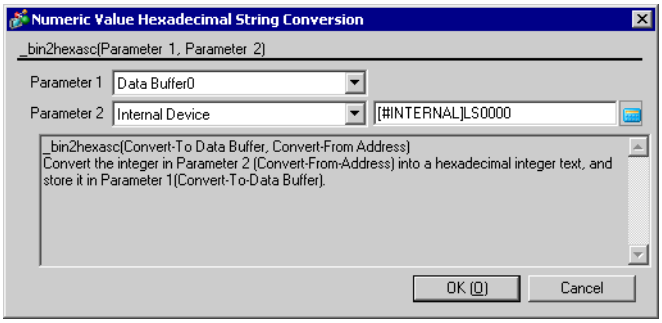
_bin2decasc (databuf0, [w:[#INTERNAL]LS0100])

	32 bit
LS0100	12345678
LS0102	

The above data are converted as follows.

	8 bit	
databuf0[0]	31h	‘1’
databuf0[1]	32h	‘2’
databuf0[2]	33h	‘3’
databuf0[3]	34h	‘4’
databuf0[4]	35h	‘5’
databuf0[5]	36h	‘6’
databuf0[6]	37h	‘7’
databuf0[7]	38h	‘8’
databuf0[8]	00h	NULL

21.11.7 Numeric Value Hexadecimal String Conversion

Item	Description
Summary	This function is used to convert binary data into a hexadecimal string. Convert the integer in Parameter 2 (Convert-From Address) into a hexadecimal integer text, and store it in Parameter 1 (Convert-To Data Buffer).
Format	_bin2hexasc (Convert-To Data Buffer, [Convert-From Address])  Parameter 1: Data Buffer Parameter 2: Internal Device, Temporary address

Example expression 1 (When the data length is 16 bits)

_bin2hexasc (databuf0, [w:[#INTERNAL]LS0100])

	16 bit
LS0100	1234h

The above data are converted as follows: Note that “NULL (0x00)” is added.

	8 bit	
databuf0[0]	31h	‘1’
databuf0[1]	32h	‘2’
databuf0[2]	33h	‘3’
databuf0[3]	34h	‘4’
databuf0[4]	00h	NULL

Example expression 2 (When the data length is 32 bits)

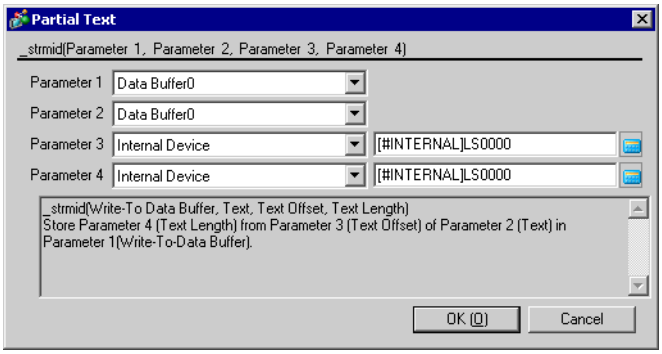
_bin2hexasc (databuf0, [w:[#INTERNAL]LS0100])

	32 bit
LS0100	12345678h
LS0102	

The above data are converted as follows.

	8 bit	
databuf0[0]	31h	‘1’
databuf0[1]	32h	‘2’
databuf0[2]	33h	‘3’
databuf0[3]	34h	‘4’
databuf0[4]	35h	‘5’
databuf0[5]	36h	‘6’
databuf0[6]	37h	‘7’
databuf0[7]	38h	‘8’
databuf0[8]	00h	NULL

21.11.8 Partial Text

Item	Description
Summary	Data are retrieved from the specified offset of the string according to the length of the string and stored in another data buffer. Store Parameter 4 (Text Length) from Parameter 3 (Text Offset) of Parameter 2 (Text) in Parameter 1 (Write-To Data Buffer).
Format	_strmid (Write-To Data Buffer, Text, Text Offset, Text Length)  Parameter 1: Data Buffer Parameter 2: String, Data Buffer Parameter 3: Numeric Value, Internal Device, Temporary address (The valid range for Parameter 3 is from 0 to 1024.) Parameter 4: Numeric Value, Internal Device, Temporary address (The valid range for Parameter 4 is from 1 to 1024.)

Example expression:

_strmid (databuf0, “12345678”, 2, 4)

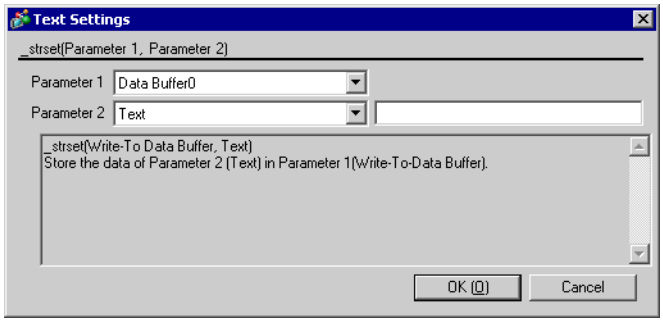
4 bytes of data retrieved from offset 2 of string “12345678” are stored in “databuf0”.

	8 bit	
databuf0[0]	33h	‘3’
databuf0[1]	34h	‘4’
databuf0[2]	35h	‘5’
databuf0[3]	36h	‘6’
databuf0[4]	00h	NULL

IMPORTANT

- When attempting to retrieve a string longer than the string specified with the “strmid ( )” function, or when specifying an offset value greater than the specified string, error No. 3 (string extraction error) of the string error status [e: STR_ERR_STAT] is issued.
- The processing is terminated when an error occurs and returns to the beginning of the Main function. (If the command comes while a function is running, it returns to the line that called that function.)

21.11.9 Text Settings

Item	Description
Summary	A fixed string is stored in the data buffer. Stores the data of Parameter 2 (Text) in Parameter 1 (Write-To Data Buffer).
Format	_strset (Write-To Data Buffer, Text)  Parameter 1: Data Buffer Parameter 2: Text, Numeric Value (Text Code) (The valid range for Parameter 2 is 0 and from 1 to 255.)

Example expression:

_strset (databuf0, “ABCD”)

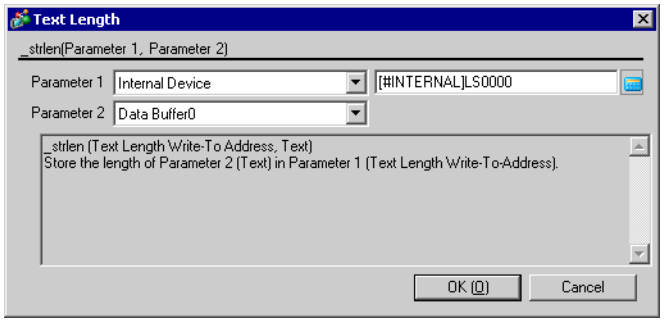
The string is stored in the data buffer as illustrated below:

	8 bit	
databuf0[0]	41h	‘A’
databuf0[1]	42h	‘B’
databuf0[2]	43h	‘C’
databuf0[3]	44h	‘D’
databuf0[4]	00h	NULL

IMPORTANT

- A string of up to 255 characters can be specified. To create strings longer than this limit, store the string in another buffer and concatenate the strings with the string-concatenating function (_strcat).
- To clear the data buffer, create an empty string “”.
  Example)_strset (databuf0,“”)
  _strset (databuf0,0)

21.11.10 Text Length

Item	Description
Summary	Obtains the length of the stored string. Stores the length of Parameter 2 (Text) in Parameter 1 (Text Length Write-To Address). (The NULL character is not included.)
Format	_strlen (Text Length Write-To Address, Text)  Parameter 1: Internal Device, Temporary address Parameter 2: String, Data Buffer

Example expression 1:

_strlen ([w:[#INTERNAL]LS0100], “ABCD”)

When the above statement is executed, the length of the string is written into LS0100 as illustrated below.

LS0100	4

Example expression 2:

_strlen ([t:0000], databuf0)

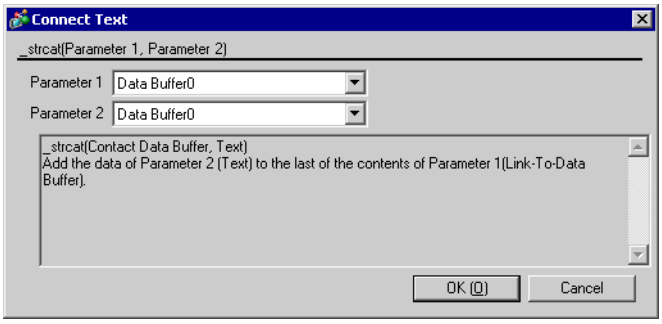
The content of “databuf0” is as follows:

	8 bit	
databuf0[0]	31h	‘1’
databuf0[1]	32h	‘2’
databuf0[2]	33h	‘3’
databuf0[3]	34h	‘4’
databuf0[4]	00h	NULL

When the above statement is executed, the length of the string is written into [t: 0000] as illustrated below.

t0000	4

21.11.11 Connect Text

Item	Description
Summary	A character string or character code is concatenated with the text buffer. Adds the data of Parameter 2 (Text) to the last of the contents of Parameter 1(Contact Data Buffer).
Format	_strcat (Contact Data Buffer, Text)  Parameter 1: Data Buffer Parameter 2: Text, Numeric Value (Text Code), Data Buffer (The valid range for Parameter 2 is 0 and from 1 to 255.)

Example expression 1:

_strcat (databuf0, “ABCD”)

	8 bit	
databuf0[0]	31h	‘1’
databuf0[1]	32h	‘2’
databuf0[2]	33h	‘3’
databuf0[3]	34h	‘4’
databuf0[4]	00h	NULL

When “ABCD” is concatenated according to the above, the result is as follows. Note that “NULL (0x00)” is added.

	8 bit	
databuf0[0]	31h	‘1’
databuf0[1]	32h	‘2’
databuf0[2]	33h	‘3’
databuf0[3]	34h	‘4’
databuf0[4]	41h	‘A’
databuf0[5]	42h	‘B’
databuf0[6]	43h	‘C’
databuf0[7]	44h	‘D’
databuf0[8]	00h	NULL

IMPORTANT

- A string of up to 255 characters can be specified.
- If you set an empty string “” or the numeric value 0 to Parameter 2, Parameter 1’s data buffer does not change.
  Example:_strcat (databuf0,“”)
  _strcat (databuf0,0)

21.12 Operation Examples

21.12.1 Logical Operation Examples

■ **The following shows logical operation examples.**

◆ **((100 > 99) and (200 <> 100))**
Result: ON

◆ **((100 > 99) and (200 <> 200))**
Result: OFF

◆ **((100 > 99) or (200 <> 200))**
Result: ON

◆ **((100 < 99) or (200 <> 200))**
Result: OFF

◆ **not (100 > 99)**
Result: OFF

◆ **not (100 < 99)**
Result: ON

◆ **[w:D200] < 10**
Result: True if D200 is smaller than 10.

◆ **not [w:D200]**
Result: True if D200 is 0.

◆ **([w:D200] == 2) or ([w:D200] == 5)**
Result: True if D200 is 2 or 5.

◆ **([w:D200] < 5) and ([w:D300] < 8)**
Result: True if D200 is smaller than 5, and D300 is smaller than 8.

◆ **[w:D200] < 10**
Result: True if D200 is smaller than 10.

◆ **not [w:D200]**
Result: True if D200 is 0.

◆ **([w:D200] == 2) or ([w:D200] == 5)**
Result: True if D200 is 2 or 5.

◆ **([w:D200] < 5) and ([w:D300] < 8)**
Result: True if D200 is smaller than 5, and D300 is smaller than 8.

21.12.2 Bit Operation Examples

■ **The following shows bit operation examples.**

◆ **[w:D200] << 4**

Result: The data in D200 is shifted 4 bits to the left.

◆ **[w:D200] >> 4**

Result: The data in D200 is shifted 4 bits to the right.

◆ **12(0000Ch) is stored in D301, using the BIN format.**

[w:D200] = [w:D300] >> [w:D301]
Result: The data in D300 is shifted 12 bits to the right and assigned to D200.

◆ **[w:D200] << 4**

Result: The data in D200 is shifted 4 bits to the left.

◆ **[w:D200] >> 4**

Result: The data in D200 is shifted 4 bits to the right.

◆ **12(0000Ch) is stored in D310, using the BIN format.**

[w:D200] = [w:D300] >> [w:D310]
Result: Shifts data in D300 12 bits to the right and assigns it to D200.

◆ **Bitwise AND**

0 & 0	Result: 0
0 & 1	Result: 0
1 & 1	Result: 1
0x1234 & 0xF0F0	Result: 0x1030

◆ **Bitwise OR**

0 | 0	Result: 0
0 | 1	Result: 1
1 | 1	Result: 1
0x1234 | 0x9999	Result: 0x9BBD

◆ **Bitwise XOR**

0 ^ 0	Result: 0
0 ^ 1	Result: 1
1 ^ 1	Result: 0

◆ **Bitwise 1's Complement (When the Data Format is BIN16+)**

~ 0	Result: 0xFFFF
~ 1	Result: 0xFFFE

21.12.3 Conditional Branch Usage Calculation Examples

■ Control program flow using “if-endif” and “if-else-endif”

◆ if-endif

```
if (condition)
{Process1}
endif
```

If the condition is true, Process1 is run. If false, skips Process1.

Example:

```
if ( [ w:D200 ] < 5 )
{
    [ w:D100 ] = 1
}
endif
```

If data in D200 is less than 5, then assigns 1 to D100.

◆ if-else-endif

```
if (condition)
{Process1}
else
{Process2}
endif
```

If the condition is true, runs Process1. If false, runs Process2.

Example:

```
if ( [ w:D200 ] < 5 )
{
    [ w:D100 ] = 1
}
else
{
    [ w:D100 ] = 0
}
endif
```

If the value in D200 is less than 5, assigns 1 to D100. Otherwise, assigns 0.

21.12.4 Offset Address Usage Calculation Examples

■ Offset Specification: Special Calculation Examples Using [w:D00100]#[t:0000].

◆ Script Settings: 16 bit unsigned, [t:0000]= 65526, the resulting address is [w:D00090].

100 + 65526 = 64(Hex) + FFF6(Hex) = 1005A(Hex) → 005A(Hex) = 90

Bottom 16 bits are valid

◆ Script Settings: 16 bit signed, [t:0000]= –10, the resulting address is [w:D00090].

100 + (–10) = 64(Hex) + FFF6(Hex) = 1005A(Hex) → 005A(Hex) = 90

Bottom 16 bits are valid

◆ Script Settings: 32 bit unsigned, [t:0000]= 4294901840, the resulting address is [w:D00180].

100 + 4294901840 = 64(Hex) + FFFF0050(Hex) = FFFF00B4(Hex) → 00B4(Hex) = 180

Bottom 16 bits are valid

◆ Script Settings: 32 bit signed, [t:0000]= –65456, the resulting address is [w:D00180].

100 + (–65456) = 64(Hex) + FFFF0050(Hex) = FFFF00B4(Hex) → 00B4(Hex) = 180

Bottom 16 bits are valid

IMPORTANT
- Offset addresses are always treated as 16 bit Bin values, regardless of the script's Bit Length and Data Type settings. If the result exceeds 16 bits (Maximum Value: 65535), Bits 0 to 15 are treated as the valid bits, and bits 16 and higher are ignored.

21.13 Command List

Item	Command/Function	D-Script/ Global D-Script	Extended Script
Data Type	Bin, BCD	OK	Bin only
Bit Length	16 bit, 32 bit	OK	OK
Signed/	Unsigned	OK	OK
Trigger	Timer Setting	OK	×
	Rising bit	OK	×
	Falling bit	OK	×
	Toggle bit	OK	×
	Expression is true	OK	×
	Expression is false	OK	×
Draw	Load Screen	OK	×
	Dot	OK	OK
	Line	OK	OK
	Circle	OK	OK
	Rectangle	OK	OK
Operator	Addition (+)	OK	OK
	Subtraction (–)	OK	OK
	Modulus (%)	OK	OK
	Multiplication (*)	OK	OK
	Division (/)	OK	OK
	Assignment (=)	OK	OK
Comparison	Logical AND	OK	OK
	Logical OR	OK	OK
	Negation (NOT)	OK	OK
	Less than (<)	OK	OK
	Less than or equal to (<=)	OK	OK
	Not equal to (<>)	OK	OK
	Greater than (>)	OK	OK
	Greater than or equal to (>=)	OK	OK
	Equals (==)	OK	OK

Continued

Item	Command/Function	D-Script/ Global D-Script	Extended Script
Memory Operation	Copy Memory: memcpy ( )	OK	OK
	Initialize Memory: memset ( )	OK	OK
	Copy Memory (Specifying Variable): _memcpy_EX ( )	OK	OK
	Initialize Memory (Specifying Variable): _memset_EX ( )	OK	OK
	Offset Address	OK	OK
	Shift Memory	OK	OK
	Ring Shift Memory	OK	OK
	Search Memory	OK	OK
	Compare Memory	OK	OK
Bit Operation	Shift Left (<<)	OK	OK
	Shift Right (>>)	OK	OK
	Bitwise AND (&)	OK	OK
	Bitwise OR (|)	OK	OK
	Bitwise XOR (^)	OK	OK
	1's Complement	OK	OK
	Set Bit: set ( )	OK	OK
	Clear Bit: clear ( )	OK	OK
	Toggle Bit: toggle ( )	OK	OK
Description Expression	if ( )	OK	OK
	if ( ) else	OK	OK
	loop ( ), break	OK	OK
	loop ( ) infinite loop	×	OK
Address	Bit Address	OK	Internal Device
	Word Address	OK	Internal Device
	Temporary Working Address	OK	OK*1

Continued

Item	Command/Function	D-Script/ Global D-Script	Extended Script
Constant	Dec, Hex, Oct	OK	OK
SIO Function	Receive: IO_READ ([p:SIO])	OK	OK
	Send: IO_WRITE ([p:SIO])	OK	OK
	Extended Receive: _IO_READ_EX ( )	×	OK
	Extended Send: _IO_WRITE_EX ( )	×	OK
	Standby Receive Function: _IO_READ_WAIT ( )	×	OK
	Control [c:EXT_SIO_CTRL]	OK	OK
	Status [s:EXT_SIO_STAT]	OK	OK
	No. of Received Data [r:EXT_SIO_RCV]	OK	OK
	Pause: _wait ( )	×	OK
Text Operation	Text	×	OK
	Data Buffer: databuf0, databuf1, databuf2, databuf3	×	OK
	Write String: _strset ( )	×	OK
	Cop from Data Buffer to Internal Device: _dlcopy ( )	×	OK
	Copy from Internal Device to Data Buffer: _ldcopy ( )	×	OK
	Hexadecimal Text-To-Integer Conversion: _hexasc2bin ( )	×	OK
	Decimal Text-To-Integer Conversion: _decasc2bin ( )	×	OK

Continued

Item	Command/Function	D-Script/ Global D-Script	Extended Script
Text Operation	Hexadecimal Number to String Conversion: _bin2hexasc ( )	×	OK
	Decimal Number to String Conversion: _bin2decasc ( )	×	OK
	String Length: _strlen ( )	×	OK
	String Concatenate: _strcat ( )	×	OK
	Copy Partial String: _strmid ( )	×	OK
	Status: [e:STR_ERR_STAT]	×	OK
Function	Call	OK	OK
	return	X	OK
CF File Operation	Read CSV File	OK	OK
	Output File List: _CF_dir ( )	OK	OK
	Read File: _CF_read ( )	OK	OK
	Write File: _CF_write ( )	OK	OK
	Delete File: _CF_delete ( )	OK	OK
	Edit File Name: _CF_rename ( )	OK	OK
Printer Operation	Output COM Port: IO_WRITE ([p:PRN])	OK	OK
Debug:	_debug ( )	OK	OK

*1 The temporary address exists separate from the D-script and global D-script.

Memo