Name	GetData

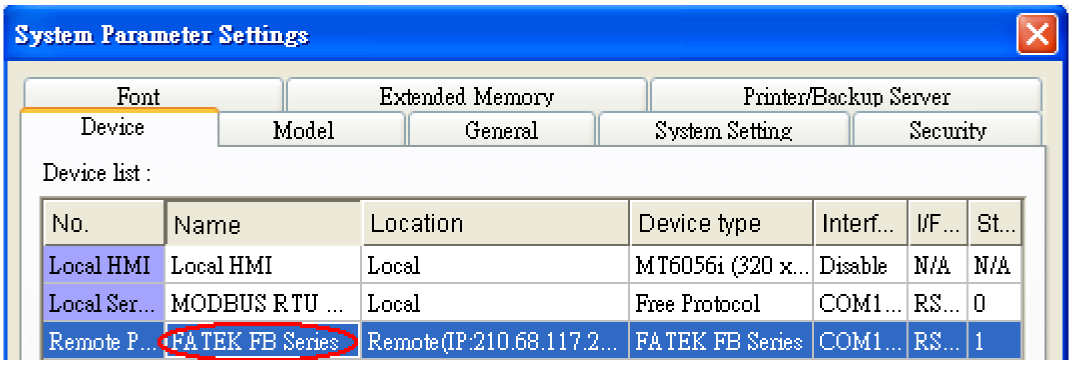
Syntax	GetData(read_data[start], device_name, device_type, address_offset, data_count)
	or
	GetData(read_data, device_name, device_type, address_offset, 1)

Description

Receives data from the PLC. Data is stored into read_data[start]~ read_data[start + data_count - 1].
Data_count is the amount of received data. In general, read_data is an array, but if data_count is 1, read_data can be an array or an ordinary variable. Below are two methods to read one word data from PLC.

```
macro_command main()
short read_data_1[2], read_data_2
GetData(read_data_1[0], “FATEK KB Series”, RT, 5, 1)
GetData(read_data_2,    “FATEK KB Series”, RT, 5, 1)
end macro_command
```

Device_name is the PLC name enclosed in the double quotation marks (“) and this name has been defined in the device list of system parameters as follows (see FATEK KB Series):



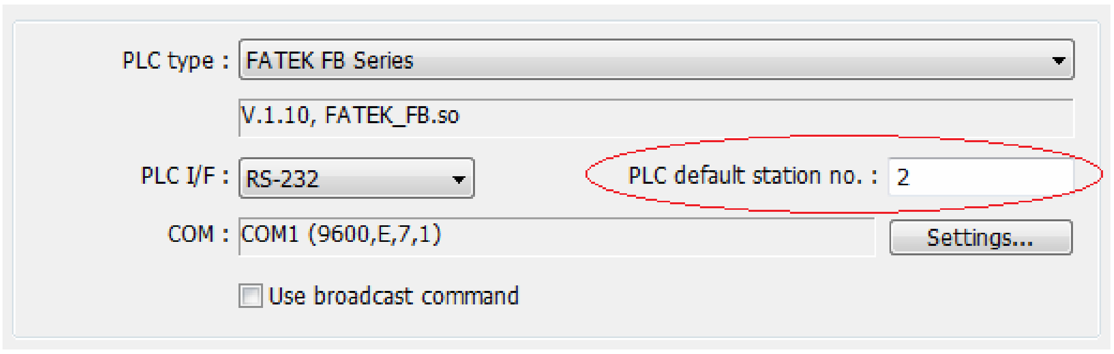
No.	Name	Location	Device type	Interf...	I/F...	St...
Local HMI	Local HMI	Local	MT6056i (320 x...	Disable	N/A	N/A
Local Ser...	MODBUS RTU ...	Local	Free Protocol	COM1 ...	RS...	0
Remote P...	FATEK FB Series	Remote (IP:210.68.117.2...	FATEK FB Series	COM1 ...	RS...	1

Device_type is the device type and encoding method (binary or BCD) of the PLC data. For example, if device_type is LW_BIN, it means the register is LW and the encoding method is binary. If use BIN encoding method, “_BIN” can be ignored.

If device_type is LW_BCD, it means the register is LW and the encoding method is BCD.

Address_offset is the address offset in the PLC.
For example, GetData(read_data_1[0], “FATEK KB Series”, RT, 5, 1) represents that the address offset is 5.

If address_offset uses the format – “N#AAAAA”, N indicates that PLC’s station number is N. AAAAA represents the address offset. This format is used while multiple PLCs or controllers are connected to a single serial port. For example, GetData(read_data_1[0], “FATEK KB Series”, RT, 2#5, 1) represents that the PLC’s station number is 2. If GetData() uses the default station number defined in the device list as follows, it is not necessary to define station number in address_offset.


PLC type : FATEK FB Series
V.1.10, FATEK_FB.so
PLC I/F : RS-232
PLC default station no. : 2
COM : COM1 (9600,E,7,1)

Use broadcast command

The number of registers actually read from depends on both the type of the read_data variable and the value of the number of data_count.

type of read_data	data_count	actual number of 16-bit register read
char (8-bit)	1	1
char (8-bit)	2	1
bool (8-bit)	1	1
bool (8-bit)	2	1
short (16-bit)	1	1
short (16-bit)	2	2
int (32-bit)	1	2

int (32-bit)	2	4
float (32-bit)	1	2
float (32-bit)	2	4

When a GetData() is executed using a 32-bit data type (int or float), the function will automatically convert the data. For example,

```
macro_command main()
float f
GetData(f, "MODBUS", 6x, 2, 1)    // f will contain a floating point value
end macro_command
```

Example

```
macro_command main()
bool a
bool b[30]
short c
short d[50]
int e
int f[10]
double g[10]

//   get the state of LB2 to the variable a
GetData(a, “Local HMI”, LB, 2, 1)

//   get 30 states of LB0 ~ LB29 to the variables b[0] ~ b[29]
GetData(b[0], “Local HMI”, LB, 0, 30)

//   get one word from LW2 to the variable c
GetData(c, “Local HMI”, LW, 2, 1)

//   get 50 words from LW0 ~ LW49 to the variables d[0] ~ d[49]
GetData(d[0], “Local HMI”, LW, 0, 50)

//   get 2 words from LW6 ~ LW7 to the variable e
//   note that the type of e is int
GetData(e, “Local HMI”, LW, 6, 1)

//   get 20 words (10 integer values) from LW0 ~ LW19 to variables f[0] ~ f[9]
```

	// since each integer value occupies 2 words
	GetData(f[0], “Local HMI”, LW, 0, 10)
	
	// get 2 words from LW2 ~ LW3 to the variable f
	GetData(f, “Local HMI”, LW, 2, 1)
	
	end macro_command

Name	GetDataEx
Syntax	GetDataEx (read_data[start], device_name, device_type, address_offset, data_count) or GetDataEx (read_data, device_name, device_type, address_offset, 1)
Description	Receives data from the PLC and continue executing next command even if no response from this device. Descriptions of read_data, device_name, device_type, address_offset and data_count are the same as GetData.

Example	macro_command main()
	bool a
	bool b[30]
	short c
	short d[50]
	int e
	int f[10]
	double g[10]
	
	// get the state of LB2 to the variable a
	GetDataEx (a, “Local HMI”, LB, 2, 1)
	
	// get 30 states of LB0 ~ LB29 to the variables b[0] ~ b[29]
	GetDataEx (b[0], “Local HMI”, LB, 0, 30)
	
	// get one word from LW2 to the variable c
	GetDataEx (c, “Local HMI”, LW, 2, 1)
	
	// get 50 words from LW0 ~ LW49 to the variables d[0] ~ d[49]

	GetDataEx (d[0], “Local HMI”, LW, 0, 50)
	
	// get 2 words from LW6 ~ LW7 to the variable e
	// note that he type of e is int
	GetDataEx (e, “Local HMI”, LW, 6, 1)
	
	// get 20 words (10 integer values) from LW0 ~ LW19 to f[0] ~ f[9]
	// since each integer value occupies 2 words
	GetDataEx (f[0], “Local HMI”, LW, 0, 10)
	
	// get 2 words from LW2 ~ LW3 to the variable f
	GetDataEx (f, “Local HMI”, LW, 2, 1)
	
	end macro_command

Name	SetData

Syntax	SetData(send_data[start], device_name, device_type, address_offset, data_count)
	or
	SetData(send_data, device_name, device_type, address_offset, 1)

Description	Send data to the PLC. Data is defined in send_data[start]~ send_data[start + data_count - 1]. data_count is the amount of sent data. In general, send_data is an array, but if data_count is 1, send_data can be an array or an ordinary variable. Below are two methods to send one word data. macro_command main() short send_data_1[2] = { 5, 6}, send_data_2 = 5 SetData(send_data_1[0], “FATEK KB Series”, RT, 5, 1) SetData(send_data_2, “FATEK KB Series”, RT, 5, 1) end macro_command device_name is the PLC name enclosed in the double quotation marks (“) and this name has been defined in the device list of system parameters. device_type is the device type and encoding method (binary or BCD) of the PLC data. For example, if device_type is LW_BIN, it means the register is

LW and the encoding method is binary. If use BIN encoding method, "_BIN" can be ignored.

If device_type is LW_BCD, it means the register is LW and the encoding method is BCD.

address_offset is the address offset in the PLC.
For example, SetData(read_data_1[0], "FATEK KB Series", RT, 5, 1) represents that the address offset is 5.

If address_offset uses the format – "N#AAAAA", N indicates that PLC's station number is N. AAAAA represents the address offset. This format is used while multiple PLCs or controllers are connected to a single serial port. For example, SetData(read_data_1[0], "FATEK KB Series", RT, 2#5, 1) represents that the PLC's station number is 2. If SetData () uses the default station number defined in the device list, it is not necessary to define station number in address_offset.

The number of registers actually sends to depends on both the type of the send_data variable and the value of the number of data_count.

type of read_data	**data_count**	**actual number of 16-bit register send**
char (8-bit)	1	1
char (8-bit)	2	1
bool (8-bit)	1	1
bool (8-bit)	2	1
short (16-bit)	1	1
short (16-bit)	2	2
int (32-bit)	1	2
int (32-bit)	2	4
float (32-bit)	1	2
float (32-bit)	2	4

When a SetData() is executed using a 32-bit data type (int or float), the

	function will automatically send int-format or float-format data to the device. For example,
	
	`macro_command main()`
	`float f = 2.6`
	`SetData(f, "MODBUS", 6x, 2, 1)    // will send a floating point value to the device`
	`end macro_command`

Example	(see code below)

```
macro_command main()
int i
bool a = true
bool b[30]
short c = false
short d[50]
int e = 5
int f[10]

for i = 0 to 29
  b[i] = true
next i

for i = 0 to 49
  d[i] = i * 2
next i

for i = 0 to 9
  f [i] = i * 3
next i

//     set the state of LB2
SetData(a, “Local HMI”, LB, 2, 1)

//    set the states of LB0 ~ LB29
SetData(b[0], “Local HMI”, LB, 0, 30)

//    set the value of LW2
SetData(c, “Local HMI”, LW, 2, 1)
```

	// set the values of LW0 ~ LW49
	SetData(d[0], “Local HMI”, LW, 0, 50)
	
	// set the values of LW6 ~ LW7, note that the type of e is int
	SetData(e, “Local HMI”, LW, 6, 1)
	
	// set the values of LW0 ~ LW19
	// 10 integers equal to 20 words, since each integer value occupies 2 words.
	SetData(f[0], “Local HMI”, LW, 0, 10)
	
	end macro_command

Name	SetDataEx
Syntax	SetDataEx (send_data[start], device_name, device_type, address_offset, data_count) or SetDataEx (send_data, device_name, device_type, address_offset, 1)
Description	Send data to the PLC and continue executing next command even if no response from this device. Descriptions of send_data, device_name, device_type, address_offset and data_count are the same as SetData.
Example	macro_command main() int i bool a = true bool b[30] short c = false short d[50] int e = 5 int f[10] for i = 0 to 29 b[i] = true next i for i = 0 to 49

	d[i] = i * 2 next i for i = 0 to 9 f [i] = i * 3 next i // set the state of LB2 SetDataEx (a, “Local HMI”, LB, 2, 1) // set the states of LB0 ~ LB29 SetDataEx (b[0], “Local HMI”, LB, 0, 30) // set the value of LW2 SetDataEx (c, “Local HMI”, LW, 2, 1) // set the values of LW0 ~ LW49 SetDataEx (d[0], “Local HMI”, LW, 0, 50) // set the values of LW6 ~ LW7, note that the type of e is int SetDataEx (e, “Local HMI”, LW, 6, 1) // set the values of LW0 ~ LW19 // 10 integers equal to 20 words, since each integer value occupies 2 words. SetDataEx (f[0], “Local HMI”, LW, 0, 10) end macro_command

Name	GetError
Syntax	GetError (err)
Description	Get an error code.
Example	macro_command main() short err char byData[10]