Multi-Channel Communication Module

. Connect 8 vibration/temperature modules

. 16 analog input channels

. Supports various communication methods

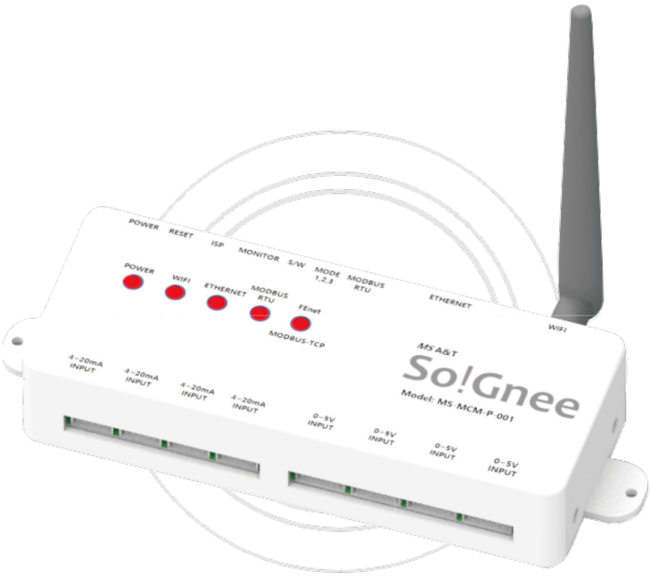
. User can choose communication method

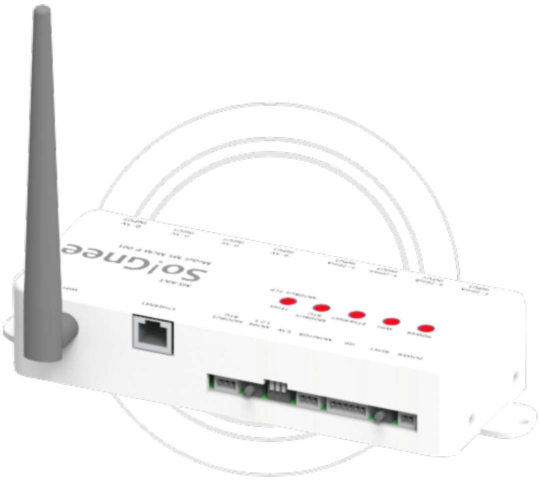
. Voltage and current type sensors available

. PLC and HMI (SCADA) communication

. SSID, PASSWORD, IP , MAC Address Set via Monitor

Mode		
No.	FUNCTION	
	H	L
1	WIFI	Ethernet
2	FEnet	Modbus-TCP/IP
3	WIFI/Ethernet	Modbus-RTU

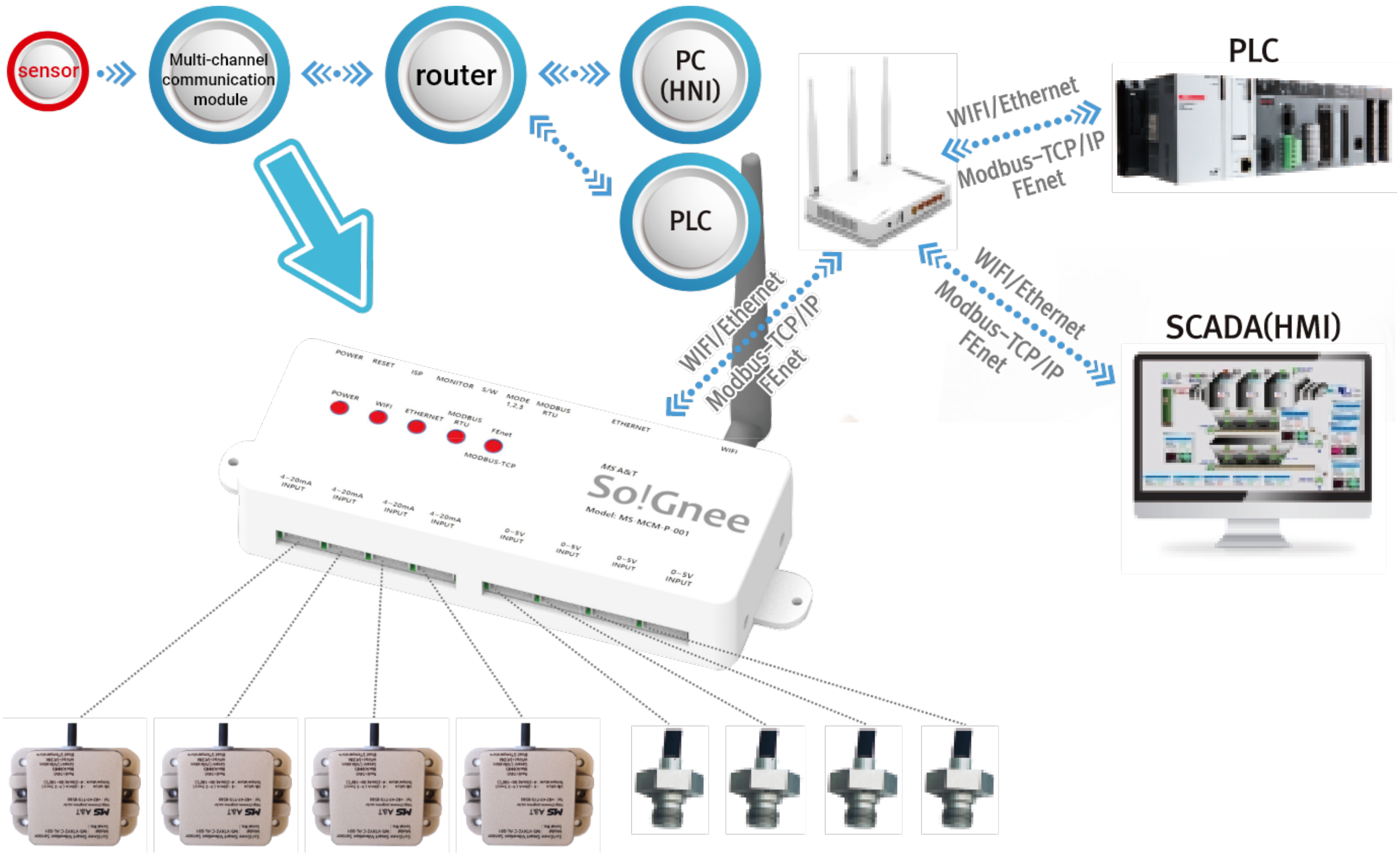




Specification

Item	MS-MCM-P-001	Unit
INPUT POWER	DC 20 ~ 26	V
WIFI	802.11b/g/n, 2.4GHz	
Ethernet	10/100 base-T	
Modbus-RTU	19600(RS485communication infrastructure)	BPS
Modbus-TCP	Industrial Network	
FEnet	LS dedicated communication (XGT,XGK)	
Analog Voltage Input	0 ~ 5 (8-Channel)	V
Analog Current Input	4 ~ 20 (8-Channel)	mA
Monitor	Communication status monitoring / Setup	V
S/W (Switch)	Setup Mode	
Reset	reboot / reset	

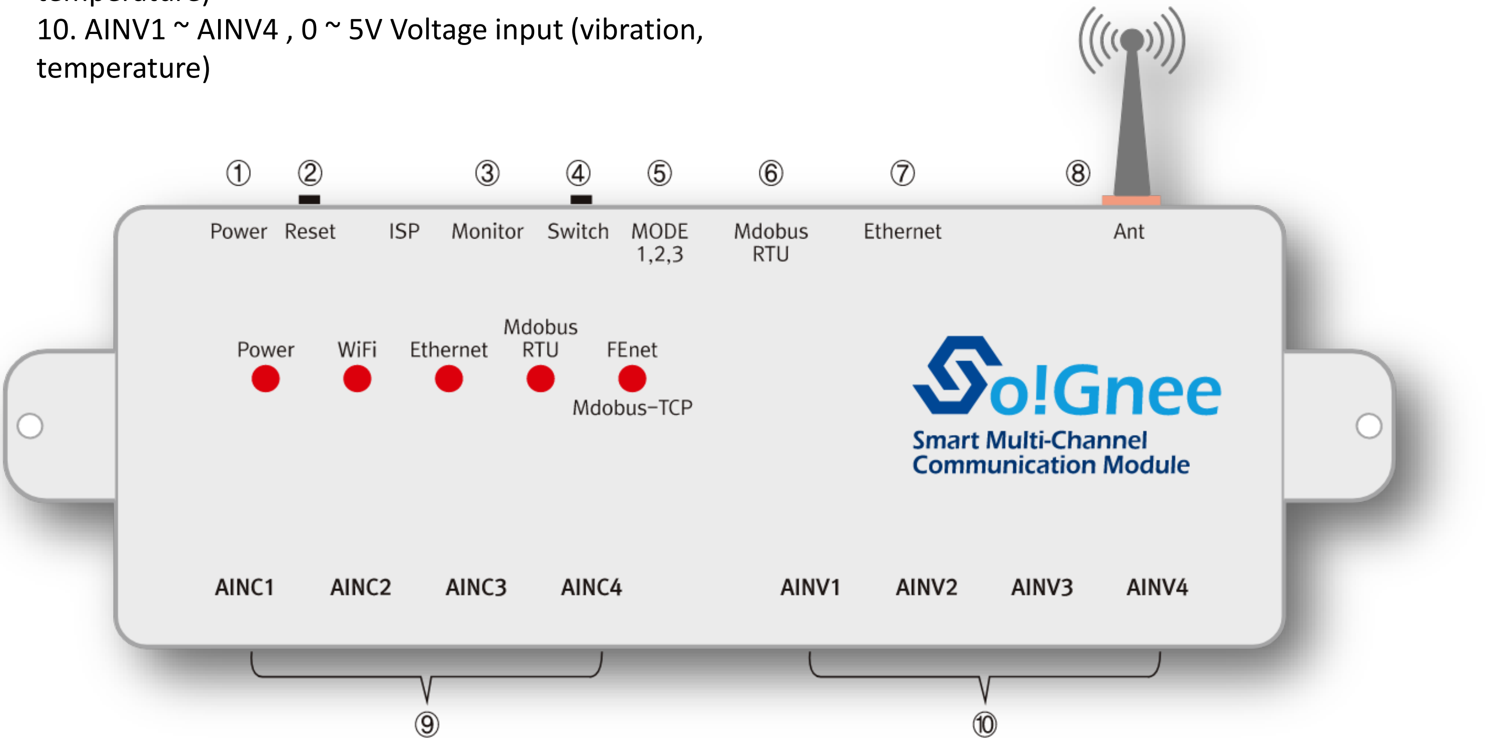
System configuration diagram using multi-channel communication module



Description of each port

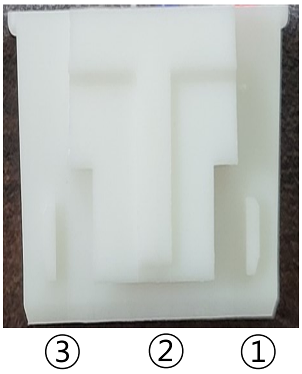
1. Power : 24V input
2. Reset : Reset processing after setting up MODE and Monitor
3. Monitor : Setup settings, IP and Mac address settings,
   communication output status monitoring
4. Switch : Entering the setting mode from the monitor state
is done through the ‘q’ or “Switch” button.
5. MODE 1,2,3 : see MODE
6. Modbus RTU : Modbus RTU communication port 19600
   (based on RS485 communication)
7. Ethernet : Ethernet communication port
8. ANT : Antenna for WiFi
9. AINC1 ~ AINC4 , 4 ~ 20mA Current input (vibration, temperature)
10. AINV1 ~ AINV4 , 0 ~ 5V Voltage input (vibration, temperature)

Mode		
No.	FUNCTION	
	H	**L**
1	WIFI	Ethernet
2	FEnet	Modbus-TCP/IP
3	WIFI/Ethernet	Modbus-RTU



Serial Monitor Harness Specifications (USB to SERIAL CONVERTOR)

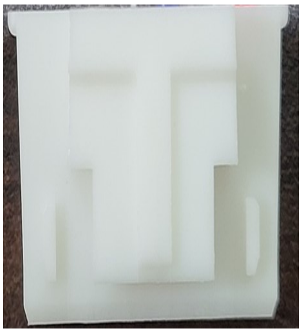
SMH250-03 (connector)



connector	Harness cable					
	1	2	3			
SMH250-03	GND	RX	TX			

Modbus-RTU Harness Specifications

SMH250-03 (connector)



connector	Harness cable					
	1	2	3			
SMH250-03	B(-)	A(+)	GND			

basic command

command format	factor format	Explanation	Command example
`all`		View all currently set setting values	`all`
`apply`		Apply the changed setting value to initialize the communication module, etc.	`apply`
`q`		Pause device motion	`q`
`resume`		Restart device behavior	`resume`

WiFi related setup commands

command format	factor format	Explanation	Command example
`ssid`		Inquire the currently set WiFi SSID value	`ssid`
`set ssid` <factor>	string without spaces	Set to use the SSID for WiFi communication	`set ssid` *`iptime`*
`Pass`		Inquire the currently set WiFi password value	`pass`
`set pass` <factor>	string without spaces	Set to use the password for WiFi communication	`set pass` *`12345678`*

※ According to the IEEE 802.11 standard, WiFi SSID supports setting up to 32 bytes.

※ The WiFi password supports setting from a minimum of 8 bytes to a maximum of 63 bytes based on the WPA2-PSK standard of the IEEE 802.11i standard.

WiFi and Ethernet related setup commands

command format	factor format	Explanation	Command example
`ip`		Inquire the currently set static IP address	`ip`
`set ip` <factor>	Decimal IPv4 address (delimiter: .)	Set to use the static IP address for communication	`set ip` *192.168.1.1*
`Mac`		Current MAC Address Lookup	`mac`
`set mac` <factor>	Hexadecimal MAC address (separator: -)	Set to use that MAC address for communication	`set mac` *02-CA-FE-BA-BE-00*

Modbus TCP and RTU related setting commands

command format	factor format	Explanation	Command example
`slave`		Search the currently set Slave ID	`slave`
`set slave` <factor>	1-byte unsigned integer	Set to use the corresponding Slave ID for communication	`set slave` *123*

Function related setting commands

command format	factor format	Explanation	Command example
`pmode`		Inquire currently set port mode	`pmode`
`set pmode` *<factor>*	port mode number	Set to operate in the corresponding port mode	`set pmode` *1*
`scale`		Query the currently set scale up value for all ports	`scale`
`scale` <factor>	port number	Query the currently set scale up value for a specific port	`scale` *1*

`set scale all` <factor>	integer or real number	Set all ports to use that scale up value	`set scale all 1`
			`set scale all 1.23`

`set scale` <factor1> <factor2>	<factor1>: port number
	<factor2>: integer or real number
		Set a specific port to use its scale up value	`set scale 10 1`
			`set scale 10 1.23`

※ In the 'scale <factor>' and 'set scale <factor 1> <factor 2>' commands, the port number must be the active port number in the current port mode.

※ The scale up value set for each port remains in the set value even if the corresponding port is deactivated. That is, the corresponding value is applied when the corresponding port is re-enabled.

※ The 'set scale all <factor>' command sets the scale-up value of all ports, including ports disabled in the current port mode, to the corresponding value to prevent errors.

Port mode type

Port mode	active port	number of active ports	data	number of sampling
0	0~1	2	Voltage (even port) Temperature Algorithm 1 (odd port)	500 times
1	0~7	8		
2	0~1	2	Current Algorithm 1 (Even Port) Temperature Algorithm 1 (odd ports)	500 times
3	0~7	8		
4	8~15	8	vibration (Ports 8, 10, 12, 14) Temperature Algorithm 2 (Ports 9, 11, 13, 15)	10 times
5	0~15	16	Voltage (Ports 0, 2, 4, 6) Temperature Algorithm 1 (Ports 1, 3, 5, 7) vibration (Ports 8, 10, 12, 14)	500 times (voltage, temperature) 10 times (vibration, temperature)

			Temperature Algorithm 2 (Ports 9, 11, 13, 15)	
6	0~15	16	Current Algorithm 2	500 times

Error message type

- `Too few or many arguments`
- `Invalid argument value`
- `Unknown command`
- `Port #`*`n`*` is not available in this port mode`

XGT FEnet TCP and Modbus TCP/RTU Communication Manual

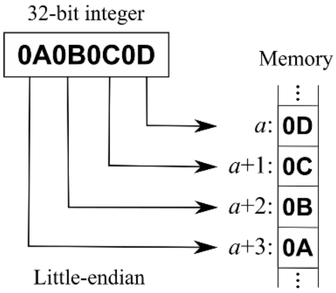
This document is about the XGT FEnet TCP, Modbus TCP, and Modbus RTU protocol-based network communication of the sensor module developed by the head office. Describes how to use the test tool software to virtually simulate the operation of

1.protocol specification

The sensor module supports some read request commands of XGT FEnet TCP, Modbus TCP, and Modbus RTU protocols, and the details of each protocol are as follows. In addition, when receiving a read request packet for all protocols, the sensor verifies that the request is valid and then transmits a read response packet including measurement data.

1.1. XGT FEnet TCP

XGT FEnet TCP is a protocol developed by LS ELECTRIC (formerly LSIS) for network communication of FEnet I/F module. . This content is based on our 'XGT FEnet I/F module protocol specification (2005. 3. 30.)', which can be found in the download data room of the LS ELECTRIC website.



The data frame of this protocol has a little-endian byte order (for example, 2-byte data 0x1234 is arranged in reverse byte order like 0x3412 in the data frame), and the frame consists of an application header and an application instruction.

Also, since this protocol basically uses the TCP 2004 port, when communicating with this sensor, it must communicate using the corresponding port.

Application Header

This area is commonly included in the front of the XGT FEnet TCP data frame and includes basic information of the data frame itself, such as frame direction, frame order, and byte length.

Company ID	Reserved	PLC Info	CPU Info	Source of Frame	Invoke ID	Length	FEnet Position	Reserved (BCC)
8 bytes	2 bytes	2 bytes	1 byte	1 byte	2 bytes	2 bytes	1 byte	1 byte

- Company ID (8 bytes): ASCII code value of character string “LSIS-XGT”. The sensor module checks whether it is a valid XGT FEnet protocol packet through the value of this area of the read request frame.
- Reserved (2 bytes): Reserved area and has a value of 0x0000. Since the sensor module does not check the value of this area of the read request frame, an arbitrary value may be included.
- PLC Info (2 bytes): In case of client → server, 0x0000 value, in case of server → client. It has a value other than 0x0000 and contains information such as CPU Type, Master/Slave, CPU operation status, and system status.may include Since the sensor module does not check the value of this area of the request frame, an arbitrary value may be included, and the sensor module also does not provide any information through this area in the response frame.
- CPU Info (1 byte): This is an area for displaying CPU information (XGK, XGI, XGR series). In the sensor module. Since the value of this area of the request frame is not checked, an arbitrary value may be included, and the sensor module also does not provide any information through this area in the response frame.
- Source of Frame (1 byte): As a frame direction value, 0x33 in case of client → server (sensor), In case of server (sensor) → client, it has a value of 0x11. The sensor module checks the value of this area in the request frame. If it is not 0x33, it may be regarded as an invalid request and may not respond.
- Invoke ID (2 bytes): ID that can be arbitrarily designated to distinguish the order between frames. sensor When the module receives the read request frame, the value of this area is included in the read response frame as the same value and transmitted.
- Length (2 bytes): Indicates the byte size of the Application Instruction. The sensor module requests If the value of this area of the frame does not match the byte size of the actual Application Instruction, it is regarded as an invalid request and may not respond.
- FEnet Position (1 byte): The upper 4 bits and the lower 4 bits are the slot number and base of the FEnet I/F module, respectively. indicates the number. Since the sensor module does not check the value of this area of the request frame, an arbitrary value may be included, and the sensor module also does not provide any information through this area of the response frame.
- Reserved (BCC) (1 byte): Reserved area. It also represents the Least Significant Byte of the value. (For example, if the byte sum of the Application Header excluding this area is 0x1234, it takes 0x34.) Since TCP itself guarantees data integrity, the sensor module does not check the value of this area of the request frame, so any The value may be included, and the sensor module transmits the LSB value of the sum of bytes in the main area of the response frame.

Application Instruction structure of read request instruction

This area is included in the XGT FEnet TCP data frame for a read request, located behind the Application Header, and includes information about the data that is the target of the read request. In order to read the data of the sensor, this request must be sent.

Command	Data type	Reserved area	Number of variables	Variable name length	Variable name	Number of data
2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	*N* bytes	2 bytes

- Command (2 bytes): The read request command has a value of 0x0054. The sensor module checks the value of this area of the request frame and supports only the read request command.
- Data type (2 bytes): The continuous data type has a value of 0x0014. The sensor module checks the value of this area of the request frame and supports only continuous data types.
- Reserved area (2 bytes): It has a value of 0x0000. Since the sensor module does not check the value of this region of the request frame, an arbitrary value may be included.
- Number of variables (2 bytes): Indicates the number of variables to be read request. The sensor module checks the value of this area of the request frame, and supports only one variable number .
- Variable name length (2 bytes): Indicates the byte length of the variable name to be read request. The sensor module checks the value of this area of the request frame to read the value of the subsequent area. If this value does not match the actual byte length of the variable name area, it may incorrectly recognize the data and not respond.
- Variable name (*N* bytes): Indicates the ASCII code value of the variable name that is the target of the read request, and the byte length of this area is the same as the value of the 'Variable name length' area. Since the sensor module does not check the value of this field in the request frame, it may include any value.
- Number of data (2 bytes): Indicates the byte length of data to be read request. The sensor module checks the value of this area in the request frame, and if it does not match the byte length of the data provided by the module (the number of sensors × the byte length of each sensor data), it may be regarded as an invalid request and may not respond. . For example, if the sensor provides 8 data, each data is a signed 16-bit (2 byte) integer, so 16 bytes must be requested.

Sample data frame of read request command

Original (Hexadecimal) ※ The gray part is the Application Header
4C 53 49 53 2D 58 47 54 00 00 00 00 A0 33 00 00 12 00 00 40 54 00 14 00 00 00 01 00 06 00 25 44 57 35 30 30 10 00

Original (Hexadecimal)	area	meaning
Application Header		
4C 53 49 53 2D 58 47 54	Company ID	ASCII code for “LSIS-XGT”
00 00	Reserved	
00 00	PLC Info	
A0	CPU Info	
33	Source of Frame	Client → Server (sensor) direction
00 00	Invoke ID	0th frame
12 00	Length	The length of the Application Instruction 18 (=0x12) bytes
00	FEnet Position	
40	Reserved (BCC)	Byte Consensus Least Significant Byte
Application Instruction		
54 00	command	read request
14 00	data type	continuous data
00 00	reserved area	
01 00	number of variables	1 variable requested
06 00	variable name length	Request variable name is 6 bytes
25 44 57 35 30 30	variable name	ASCII code for “%DW500”
10 00	number of data	16 (=0x10) bytes of data request

Application Instruction structure of read response

This area is included in the XGT FEnet TCP data frame corresponding to the read response, located behind the Application Header, and includes information on the read result. The sensor transmits a data frame including this area as a response to a read request.

Command	data type	reserved area	error state	number of variables	data size	data
2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	*N* bytes

- Command (2 bytes): A read response has a value of 0x0055.
- data type (2 bytes): Since it is a continuous data type, it has a value of 0x0014.
- reserved area (2 bytes): It has a value of 0x0000.
- error state (2 bytes): It has a value of 0x0000 in case of a normal response, and a value other than 0x0000 in case of an error.
- number of variables (2 bytes): Since it is one variable, it has a value of 0x0001.
- data size (2 bytes): Indicates the byte length of the read result data.
- data (*N* bytes): Read result data.The sensor arranges each data in descending order of port number, where each data is a signed 16-bit (2 byte) integer.

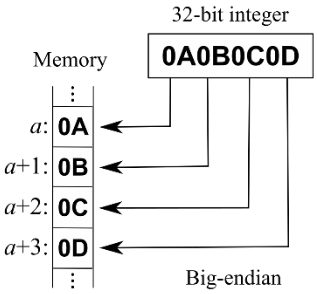
Sample data frame in read response

Original (Hexadecimal) ※ The gray part is the Application Header
4C 53 49 53 2D 58 47 54 00 00 00 00 A0 11 00 00 1C 00 00 28 **55 00 14 00 00 00 00 00 01 00 10 00 57 04 AE 08 05 0D 5C 11 B3 15 0A 1A 61 1E B8 22**

Original (Hexadecimal)	area	meaning
Application Header		
4C 53 49 53 2D 58 47 54	Company ID	ASCII code for “LSIS-XGT”
00 00	Reserved	
00 00	PLC Info	
A0	CPU Info	
11	Source of Frame	Server (sensor) → client direction
00 00	Invoke ID	0th frame
1C 00	Length	The length of the Application Instruction 28 (=0x1C) bytes
00	FEnet Position	
28	Reserved (BCC)	Byte Consensus Least Significant Byte
Application Instruction		
55 00	command	read response
14 00	data type	continuous data
00 00	reserved area	
00 00	error state	normal
01 00	number of variables	1 variable read
10 00	number of data	Read 16 (=0x10) bytes of data
57 04	Sensor data of port 0	Measures: 1,111(=0x0457)
AE 08	Sensor data of port 1	Measures: 2,222 (=0x08AE)
05 0D	Sensor data of port 2	Measures: 3,333 (=0x0D05)
5C 11	Sensor data of port 3	Measures: 4,444 (=0x115C)
B3 15	Sensor data of port 4	Measures: 5,555 (=0x15B3)
0A 1A	Sensor data of port 5	Measures: 6,666 (=0x1A0A)
61 1E	Sensor data of port 6	Measures: 7,777 (=0x1E61)
B8 22	Sensor data of port 7	Measures: 8,888 (=0x22B8)

1.2. Modbus TCP

Modbus protocol was developed for PLC communication by Modicon (now Schneider Electric) in 1979, and is widely used as a de facto standard for communication between industrial electronic equipment. Modbus TCP is a kind of Modbus protocol for use in TCP/IP environment.



The data frame of this protocol has the byte order of the Big-Endian type (byte order is arranged on the data frame as it is), and the frame consists of a header and a data area.

Also, since this protocol basically uses the TCP 502 port, when communicating with this sensor, it must be communicated using the corresponding port.

Modbus TCP header

This area is commonly included in the front part of the Modbus TCP data frame and includes basic information such as frame order, byte length, and slave number.

Transaction identifier	Protocol identifier	Length field	Unit identifier
2 bytes	2 bytes	2 bytes	1 byte

- Transaction identifier (2 bytes): Same as Invoke ID of XGT FEnet TCP, it is an ID that can be arbitrarily designated to distinguish the order between frames. When the sensor module receives the read request frame, the value of this area is included in the read response frame as the same value and transmitted.
- Protocol identifier (2 bytes): As a protocol identifier, Modbus TCP has a value of 0x0000. The sensor module checks whether it is a valid Modbus TCP protocol packet through the value of this field of the read request frame.
- Length field (2 bytes): Indicates the byte size of the following data frame (including header and data area). In the sensor module, if the value of this area of the request frame does not match the actual byte size of the subsequent data frame, it may be regarded as an invalid request and may not respond.
- Unit identifier (1 byte): Slave (device) number. Since the sensor module does not check the value of this field in the request frame, it may include any value.

Data area structure of Read Registers request

Function code	Starting address	Quantity of registers
1 byte	2 bytes	2 bytes

- Function code (1 byte): It has a value of 0x03 for Read Holding Registers and 0x04 for Read Input Registers. The sensor module checks the value of this area of the request frame, and both the Read Holding Register and Read Input Register operate the same.
- Starting address (2 bytes): The address to which the read request is made. A valid address range is 0x0000~0xFFFF, but the sensor module does not check the value of this field in the request frame, so it may include any value.
- Quantity of registers (2 bytes): The number of registers (2 bytes) subject to read request.

Sample data frame from read request

Original (Hexadecimal)	※ The gray part is the header area
00 00 00 00 00 06 12 **04 34 56 00 08**	

Original (Hexadecimal)	area	meaning
header area		
00 00	Transaction identifier	0th frame
00 00	Protocol identifier	Modbus TCP
00 06	Length field	Subsequent frames are 6 bytes
12	Unit identifier	18(=0x12) slave device
data area		
04	Function code	Read Input Register
34 56	Starting address	Request data at address 0x3456
00 08	Quantity of registers	8 Register data requests

Data area structure of Read Registers response

Function code	Byte count	Register values
1 byte	1 byte	2 bytes

- Function code (1 byte): 0x03 for Read Holding Registers, Read Input Registers. It has a value of 0x04.
- Byte count (1 byte): Indicates the byte length of the read result data.
- Register values (2 bytes): Read result data. The sensor arranges each data in descending order of port number, where each data is a signed 16-bit (2 byte) integer.

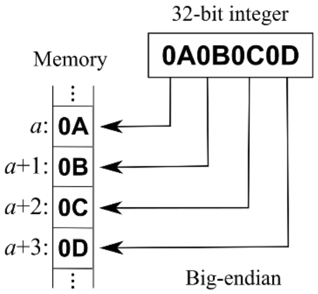
Sample data frame in read response

Original (Hexadecimal) ※ The gray part is the header area
00 00 00 00 00 13 12 **04 10 57 04 AE 08 05 0D 5C 11 B3 15 0A 1A 61 1E B8 22**

Original (Hexadecimal)	area	meaning
header area		
00 00	Transaction identifier	0th frame
00 00	Protocol identifier	Modbus TCP
00 13	Length field	The frame that follows is 19 bytes
12	Unit identifier	18(=0x12) slave device
data area		
04	Function code	Read Input Register
10	Byte count	Read 16 (=0x10) bytes of data
57 04	Sensor data of port 0	Measures: 1,111(=0x0457)
AE 08	Sensor data of port 1	Measures: 2,222 (=0x08AE)
05 0D	Sensor data of port 2	Measures: 3,333 (=0x0D05)
5C 11	Sensor data of port 3	Measures: 4,444 (=0x115C)
B3 15	Sensor data of port 4	Measures: 5,555 (=0x15B3)
0A 1A	Sensor data of port 5	Measures: 6,666 (=0x1A0A)
61 1E	Sensor data of port 6	Measures: 7,777 (=0x1E61)
B8 22	Sensor data of port 7	Measures: 8,888 (=0x22B8)

1.2. Modbus RTU

Modbus RTU is a kind of Modbus protocol for use in serial environment.



The data frame of this protocol has the byte order of the Big-Endian type (byte order is arranged on the data frame as it is), and the frame consists of a header and a data area.

Modbus RTU header

This area is commonly included in the front part of the Modbus RTU data frame, and unlike other protocols, it simply includes address information.

Address
1 byte

- Address (1 byte): Similar to Unit address of Modbus TCP, it indicates Station (device) number. Since the sensor module does not check the value of this field in the request frame, it may include any value.

Data structure of the Read Registers request

Function code	Starting address	Quantity of registers	CRC
1 byte	2 bytes	2 bytes	2 bytes

- Function code (1 byte): It has a value of 0x03 for Read Holding Registers and 0x04 for Read Input Registers. The sensor module checks the value of this area of the request frame, and both the Read Holding Register and Read Input Register operate the same.
- Starting address (2 bytes): The address to which the read request is made. A valid address range is 0x0000~0xFFFF, but since the sensor module does not check the value of this area in the request frame, it may include any value.
- Quantity of registers (2 bytes): It is the number of Registers (2 bytes) to be read request.
- CRC (2 bytes): CRC (cyclic redundancy check)

Sample data frame from read request

Original (Hexadecimal) ※ The gray part is the header area
11 03 00 6B 00 03 76 87

Original (Hexadecimal)	area	meaning
header area		
11	Address	11(=0x17) Station device
data area		
03	Function code	Read Holding Register
00 6B	Starting address	Data request at address 0x006B
00 03	Quantity of registers	3 Register data requests
76 87	CRC	CRC check value

Data structure of the Read Registers response

Function code	Byte count	Register values	CRC
1 byte	1 byte	2 bytes	2 bytes

- Function code (1 byte): 0x03 for Read Holding Registers, Read Input Registers. It has a value of 0x04.
- Byte count (1 byte): Indicates the byte length of the read result data.
- Register values (2 bytes): Read result data. The sensor arranges each data in descending order of port number, where each data is a signed 16-bit (2 byte) integer.
- CRC (2 bytes): CRC (cyclic redundancy check) check value.

Sample data frame in read response

Original (Hexadecimal) ※ The gray part is the header area
11 03 06 **AE 41 56 52 43 40** 49 AD

Original (Hexadecimal)	area	meaning
header area		
11	Address	11(=0x17) Station device
data area		
03	Function code	Read Holding Register
06	Byte count	6 (=0x6) read byte data
AE 41	Sensor data of port 0	Measures: 44,609 (=0xAE41)
56 52	Sensor data of port 1	Measures: 22,098 (=0x5652)
43 40	Sensor data of port 2	Measures: 17,216 (=0x4340)
49 AD	CRC	CRC check value