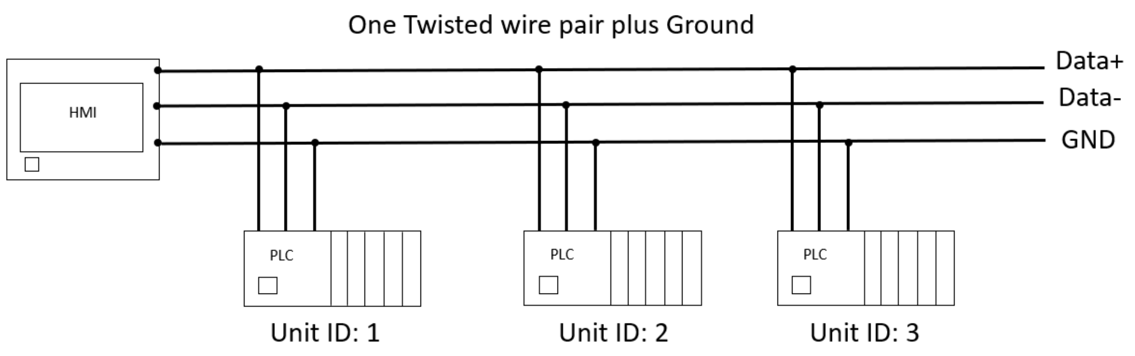
Weintek USA, Inc.
www.WeintekUSA.com
(425) 488-1100

Rev. JAN 27, 2020

Weintek HMI to Multiple Modbus RTU (RS-485) Slave Devices

Introduction: This instruction manual discusses how to communicate with Modbus RTU slave devices through a RS-485 network. The Modbus RTU protocol is widely used on many industrial sites and adopted by many manufacturers because this protocol is free, open, and simple. Modbus RTU enables master-slave communication between devices connected through serial RS-485 using two-wires. In a RS-485 network, a master queries one or more slave devices for data acquisition or parameter settings. The slaves return a response to all queries addressed to them individually and only respond to the queries transmitted from the master. Typically, a Modbus RS-485 communication network requires a 120 ohm resistor at each physical end of a network. That means there should be a resister at the master and the last slave. The terminating resistor should match the characteristic impedance of the cable being terminated.



RS-485 Wiring Diagram

Equipment & Software:

1. EasyBuilder Pro v6.03.02.294
2. Weintek HMI cMT3090
3. Modbus RTU (RS-485) slave devices

Knowledge of Modbus RTU Protocol:

A Modbus slave device provides a Modbus master device with the following memory tables to access data.

Object Type	Access (Read-write)	Address Range	Read	Write Single	Write Multiple
Coil (Bit)	R/W	00001-09999 (0x)	FC01	FC05	FC15
Discrete input (Bit)	R	10001-19999 (1x)	FC02	N/A (Read only)	N/A (Read only)
Input register (16-bits)	R	30001-39999 (3x)	FC04	N/A (Read only)	N/A (Read only)
Holding register (16-bits)	R/W	40001-49999 (4x)	FC03	FC06	FC16

Note: FC means Modbus **F**unction **C**ode

The supported Modbus Function Codes vary from the manufacturers. The common function codes are shown below.

Function Code (Decimal)	Access	Access Object Type
01	Read	Coil
02	Read	Discrete input
03	Read	Holding register
04	Read	Input register
05	Write single	Coil
06	Write single	Holding register
15	Write multiple	Coil
16	Write multiple	Holding register

A Modbus map is a list of parameters stored in Modbus addresses. It provides the essential information for users to access data. Most slave devices are built with fixed map defined by the manufacturer. While some Modbus slave devices, such as PLCs or HMIs, allow programmers to configure custom maps. You will need to know the following information defined by your devices.

- What is the unit ID of the device? (fixed or configurable?)
- Where is data stored? (which tables and addresses)
- How is data stored? (data types and byte, word ordering)

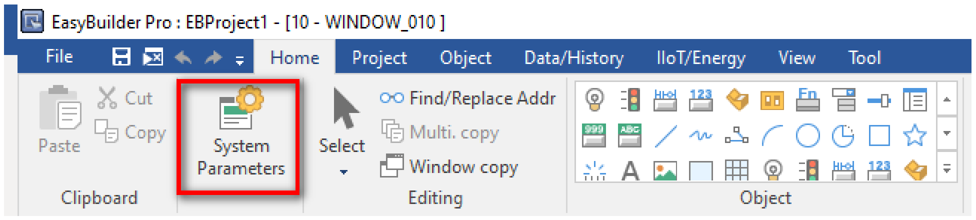
- How do you find the range?

Prerequisites:

- Each Modbus RTU slave device must use the same serial configuration, including Baud rate, Data bits, Stop bits, and Parity.
- Each Modbus RTU slave devices must be assigned a unique slave ID number (unit address) from 1 to 247.

These settings should be configurable by software of your Modbus devices.

Detail of the HMI Programming: Open a new project and choose the HMI model cMT3090. To get the HMI talking to the Modbus slaves, go to the [HOME] tab on the top of the menu and then click on the [System Parameters] button.



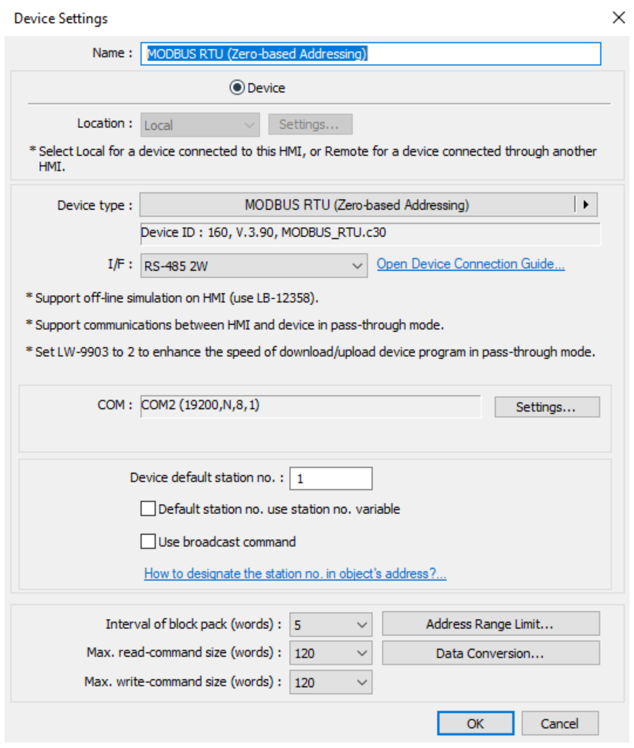
You will need to select one of the drivers based on the specification of your devices.

Driver Name	Description
Modbus RTU, RTU over TCP	The addresses for the parameters start from 1 (1 based)
Modbus RTU (Zero-based Addressing)	The addresses for the parameters start from 0 (0 based)
Modbus RTU (HEX Addressing)	Use this driver when the addresses are Heximal format and start from 0

In this case, the **Modbus RTU(Zero-based Addressing)** driver is used.

I/F: **RS-485 2W**

Device default station no.: You can enter the station number of the first slave device.


Device Settings
Device
Settings...
Open Device Connection Guide...
How to designate the station no. in object's address?...
Address Range Limit...
Data Conversion...
OK
Cancel

Name : MODBUS RTU (Zero-based Addressing)

Location : Local

* Select Local for a device connected to this HMI, or Remote for a device connected through another HMI.

Device type : MODBUS RTU (Zero-based Addressing)

Device ID : 160, V.3.90, MODBUS_RTU.c30

I/F : RS-485 2W

* Support off-line simulation on HMI (use LB-12358).

* Support communications between HMI and device in pass-through mode.

* Set LW-9903 to 2 to enhance the speed of download/upload device program in pass-through mode.

COM : COM2 (19200,N,8,1)

Device default station no. : 1

☐ Default station no. use station no. variable

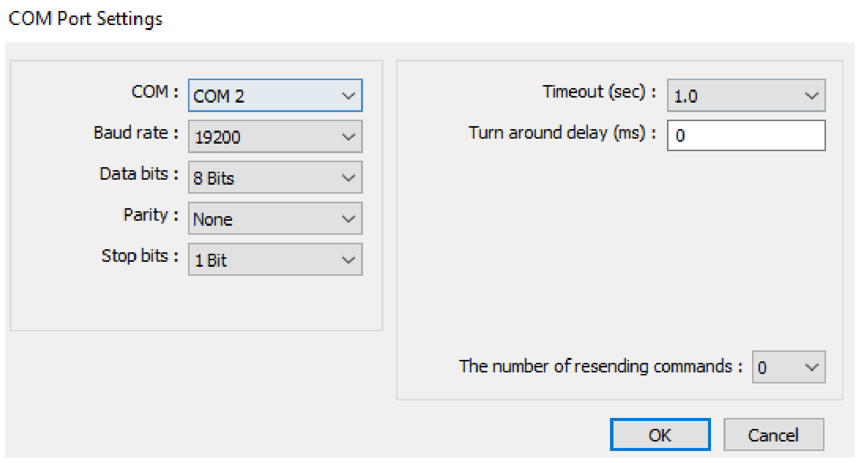
☐ Use broadcast command

Interval of block pack (words) : 5

Max. read-command size (words) : 120

Max. write-command size (words) : 120

Click on the [Settings..] button to enter the serial settings of your Modbus slave devices, including Baud rate, Data bits, Stop bits, and Parity.



COM : COM 2

Baud rate : 19200

Data bits : 8 Bits

Parity : None

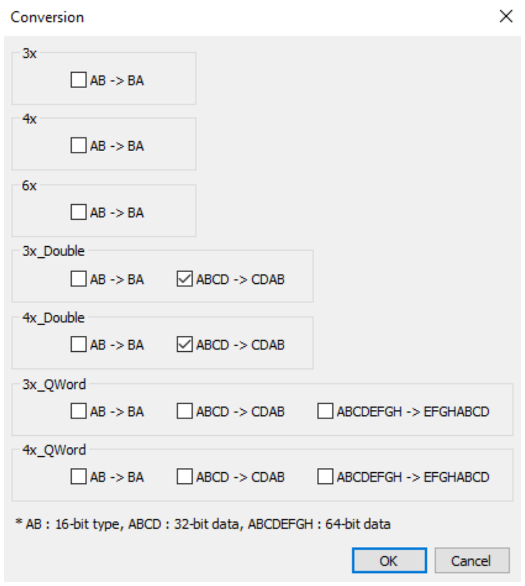
Stop bits : 1 Bit

Timeout (sec) : 1.0

Turn around delay (ms) : 0

The number of resending commands : 0

Click on the [Data Conversion] button to implement **Byte swap**, **Word swap**, or **Double Word swap**.



3x
- [ ] AB -> BA

4x
- [ ] AB -> BA

6x
- [ ] AB -> BA

3x_Double
- [ ] AB -> BA
- [✓] ABCD -> CDAB

4x_Double
- [ ] AB -> BA
- [✓] ABCD -> CDAB

3x_QWord
- [ ] AB -> BA
- [ ] ABCD -> CDAB
- [ ] ABCDEFGH -> EFGHABCD

4x_QWord
- [ ] AB -> BA
- [ ] ABCD -> CDAB
- [ ] ABCDEFGH -> EFGHABCD

* AB : 16-bit type, ABCD : 32-bit data, ABCDEFGH : 64-bit data

- Each character, such as “A”, represents one byte.
- **AB->BA** does a byte swap.
- **ABCD->CDAB** does a word swap.
- You can do a byte swap and word swap with **3x_Double** and **4x_Double**.
- **ABCDEFGH->EFGHABCD** does a double word swap (for CMT HMIs only).
- You can do a byte swap, word swap, and double word swap with **3x_QWord 4x_QWord** (for CMT HMIs only).

Since the Modbus protocol does not define exactly how data is stored in the registers. You will need to check with the manufacturer to find out which ordering format your slave device stores data.

By default the Modbus RTU master driver in Weintek HMI uses **Low byte (or word, double word)** first as ordering.

16-bit data (0x4E20)	
Low byte (0x20)	High byte (0x4E)

Low byte first

32-bit data (0xAE41,5652)	
Low word (0x5652)	High word (0xAE41)

Low word first

64-bit data (0x7048,860F,9180)	
Low double word (0x860F,9180)	High double word (0x7048)

Low double word first

Accessible device memory in EasyBuilder Pro

The Weintek HMI uses the following Modbus Function Codes.

Address	**Read/Write**	**Use Function Code (Decimal)**	**Description**
0x *1	R	01	Reads **coils**
	W	05	Writes a single **coil**
1x *1	R	02	Reads **discrete inputs**
0x_multi_coil	R	01	Reads **coils**
	W	15	Writes multiple **coils**
3x_Bit *1	R	04	Reads **input register (3x)**'s bit
4x_Bit *1	R	03	Reads **holding register (4x)**'s bit
	W	16	Writes multiple **holding register (4x)**'s bit
6x_Bit *1	R	03	Reads **holding register**'s bit
	W	06	Writes multiple **holding register (4x)**'s bit
3x	R	04	Reads **input registers**
4x	R	03	Reads **holding registers**
	W	16	Writes multiple **holding registers**
5x *2	R	03	Reads **holding registers**
	W	16	Writes multiple **holding registers**
6x *3	R	03	Reads **holding register**
	W	06	Writes a single **holding register**
3x_Double	R	04	Reads **input register** (32-bit data) Defaults to 32-bit numeric format
4x_Double	R	03	Reads **holding register** (32-bit data) Defaults to 32-bit numeric format
	W	16	Writes **holding register** (32-bit data) Defaults to 32-bit numeric format

*1. The **Modbus RTU (Zero-based Addressing)** driver reads a group of 16 bits at a time. Bit groups are 0-15, 16-31, 32-47,48-63, etc. All bits in the group must be available in the controller for the HMI to read. Otherwise, errors will result.

*2. The 5x is exactly the same as the 4x. Use the 5x when reading/writing to a 32-bit registers using the low word first format. For example,

4x contains the following data,

Address	1	2
Data (word)	0x4E20	0x7530
Data (Double word)	0x075304E20	

Then use 5x instead of 4x, it will be

Address	1	2
Data (word)	0x4E20	0x7530
Data(Double word)	0x4E207530	

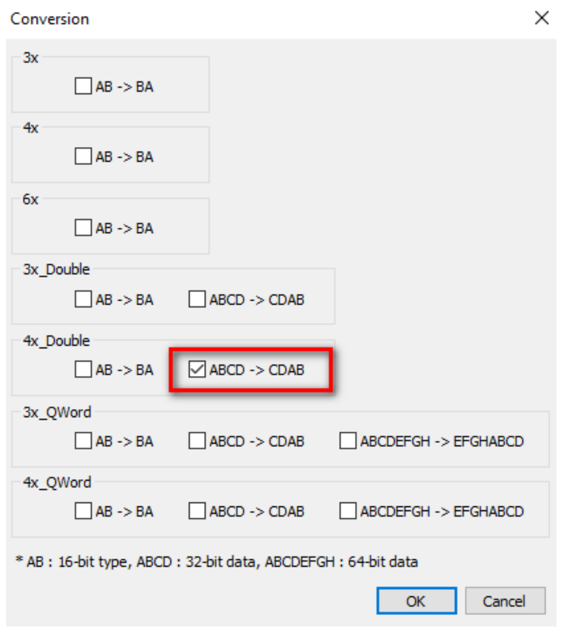
*3. By default the Weintek HMI uses a Function Code **16** to write multiple registers, even if it is only writing to one register. The 6x forces the HMI to transmit a Function Code **06** to write a single register.

Special device memory in EasyBuilder Pro

Address	Read/Write	Description
0x_single_bit	R/W	Reads a single **0x** bit at a time instead of a group of 16 consecutive bits
1x_single_bit	R	Reads a single **1x** bit at a time instead of a group of 16 consecutive bits
0x_single_coil	R/W	Reads/writes a bit at a time. (Word level access of **0x**) If the value of **0x_single_coil** -1 is 15, the Bits from **0x**-0 to **0x**-3 will be ON

How to read/write 32-bit unsigned data

To read 32-bit unsinged data from register 40106 (combined with 40107 to generate 32-bit data) with **high word first** format, please check the **Word swap [ABCD ->CDAB]** option on [Data Conversion].


Conversion

3x
- [ ] AB -> BA

4x
- [ ] AB -> BA

6x
- [ ] AB -> BA

3x_Double
- [ ] AB -> BA
- [ ] ABCD -> CDAB

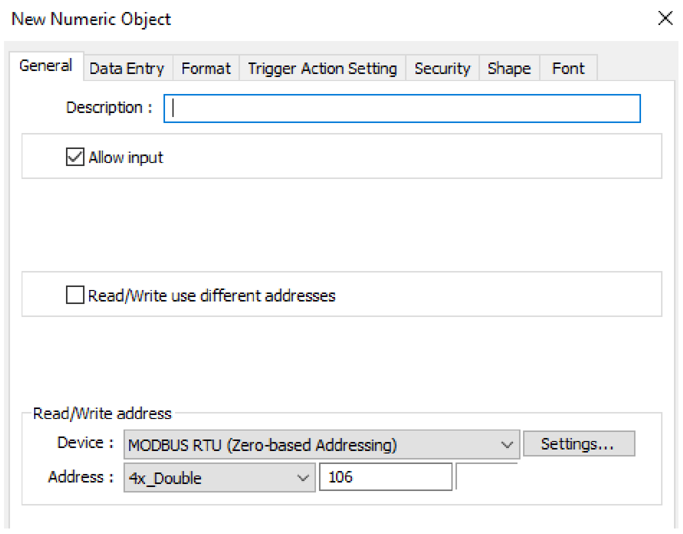
4x_Double
- [ ] AB -> BA
- [x] ABCD -> CDAB

3x_QWord
- [ ] AB -> BA
- [ ] ABCD -> CDAB
- [ ] ABCDEFGH -> EFGHABCD

4x_QWord
- [ ] AB -> BA
- [ ] ABCD -> CDAB
- [ ] ABCDEFGH -> EFGHABCD

* AB : 16-bit type, ABCD : 32-bit data, ABCDEFGH : 64-bit data

Create a Numeric object and specify the address **4x_Double - 106** on the [General] tab as below.


New Numeric Object

Description :

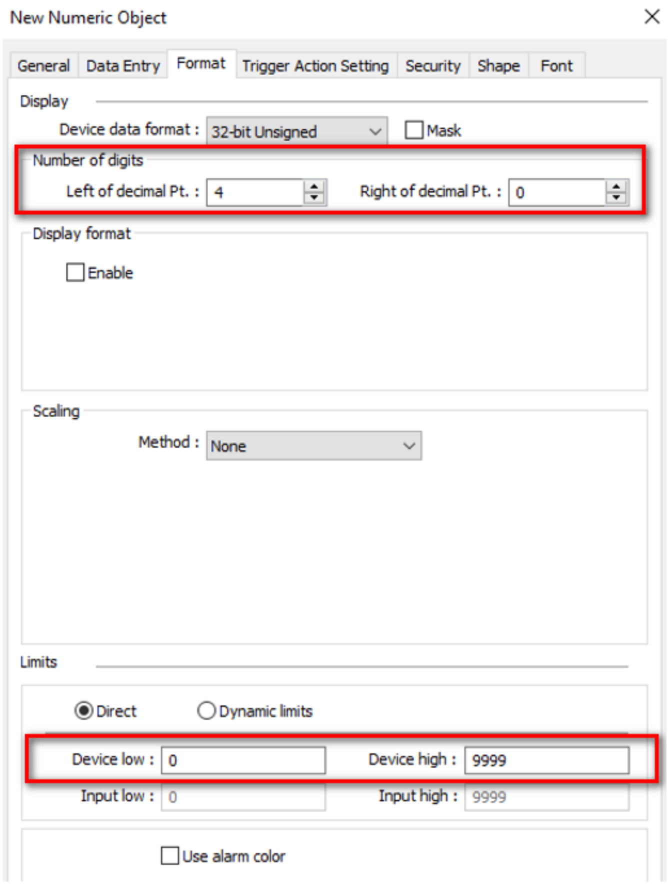
- [x] Allow input
- [ ] Read/Write use different addresses

Read/Write address

Device : MODBUS RTU (Zero-based Addressing)

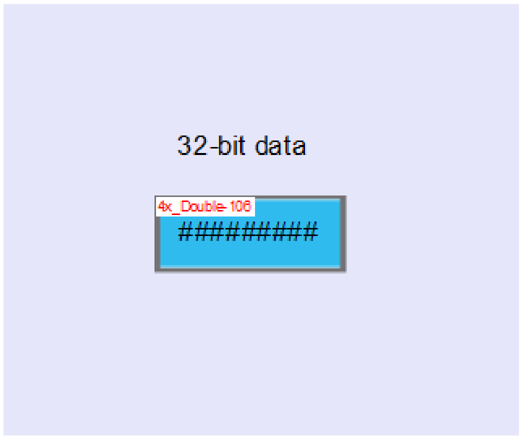
Address : 4x_Double 106

Under the [Format] tab, enter the number of digits used in this parameter as well as the device's low limit and high limit. Click the [OK] button to finish setting up this object.



Display
Device data format : 32-bit Unsigned
Mask
Number of digits
Left of decimal Pt. : 4
Right of decimal Pt. : 0
Display format
Enable
Scaling
Method : None
Limits
Direct
Dynamic limits
Device low : 0
Device high : 9999
Input low : 0
Input high : 9999
Use alarm color

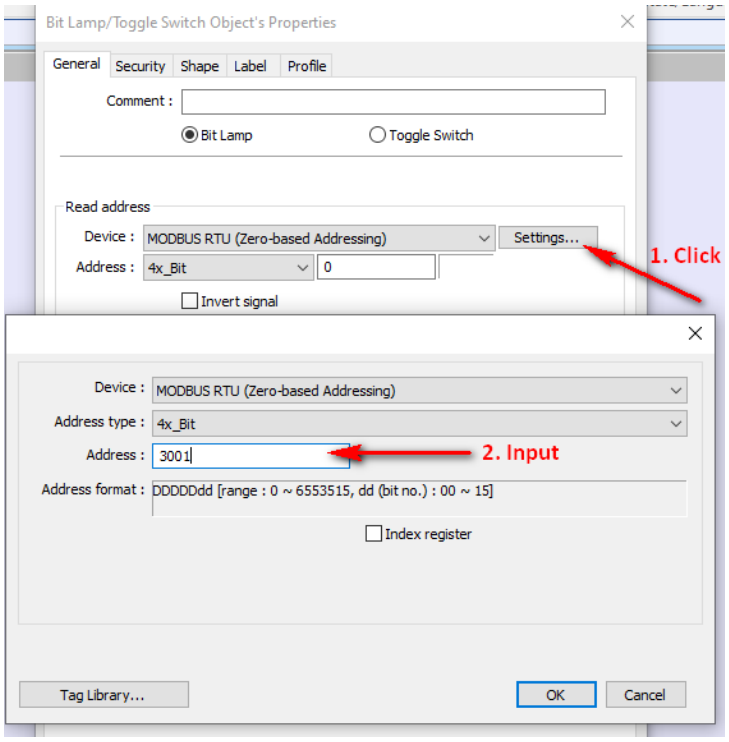
Place the Numeric object onto the editing area.



How to read/write bits in the 4x/3x memory tables

The 4x_Bit is used to read/write to individual bits in the 4x memory table. To access a bit in 4x memory table, select the **4x_Bit** as the address for bit-type objects such as Bit Lamp. Under the **Address**, use the format DDDDDdd to enter the **word** memory area, followed by the two-digit bit reference.

For example, to monitor the second bit of 40030, enter "3001" into the Address. (DD=30, dd=01)



Device : MODBUS RTU (Zero-based Addressing)
Address : 4x_Bit 0
Invert signal

Device : MODBUS RTU (Zero-based Addressing)
Address type : 4x_Bit
Address : 3001
Address format : DDDDDdd [range : 0 ~ 6553515, dd (bit no.) : 00 ~ 15]
Index register

The **3x_Bit** works the same as for the **4x_Bit**, except that it is used for accessing bits in a 3x memory table (input register, read only).

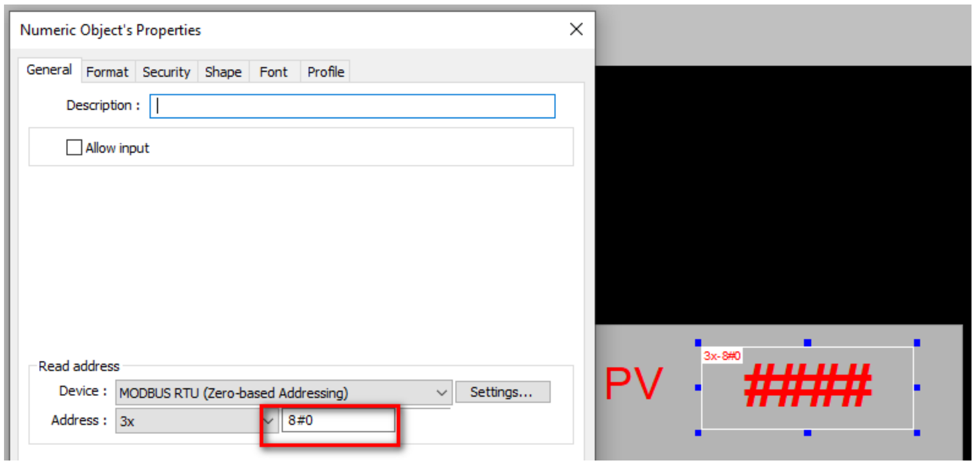
How to read data on the different station number

The address format of station number is ABC#Addr

The ABC stands for device station number and ranges from 0 to 255. The Addr stands for device address. The "#" is a sign that separates the station number and the address.

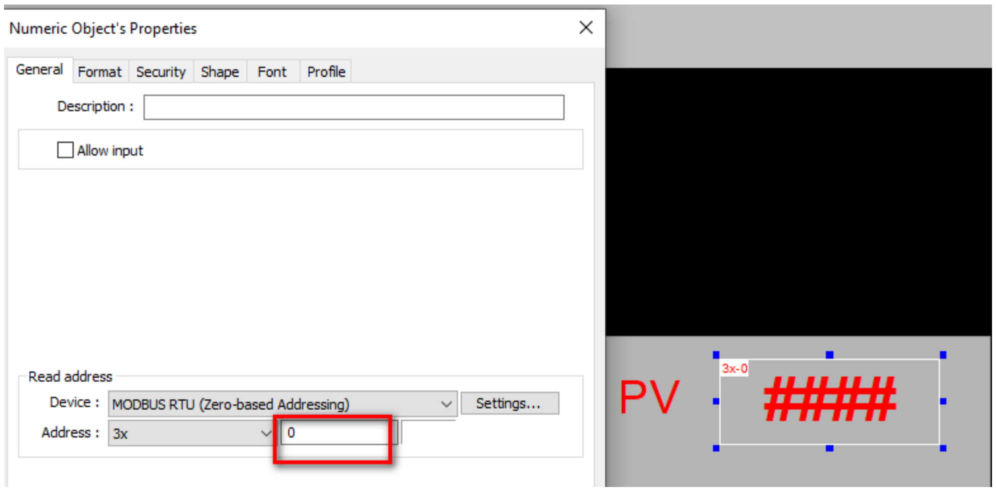
To read the station number 2 and address 0, input **2#** as prefix.

Once finishing the input, the HMI will read address 0 in the 3x (The 3xxxx table) Modbus table using Function Code 0x04.



Description :
Allow input
Read address
Device : MODBUS RTU (Zero-based Addressing)
Address : 3x 8#0

If you input the address without a station number, the HMI will query the default slave device. (where you set up on the **Modbus RTU driver**)



Description :
Allow input
Read address
Device : MODBUS RTU (Zero-based Addressing)
Address : 3x 0

How to use Station Number Variable

Station Number Variable allows you to change the station number (unit ID) during runtime to monitor different slaves. Up to 16 **Station Number Variables** can be used in one project.

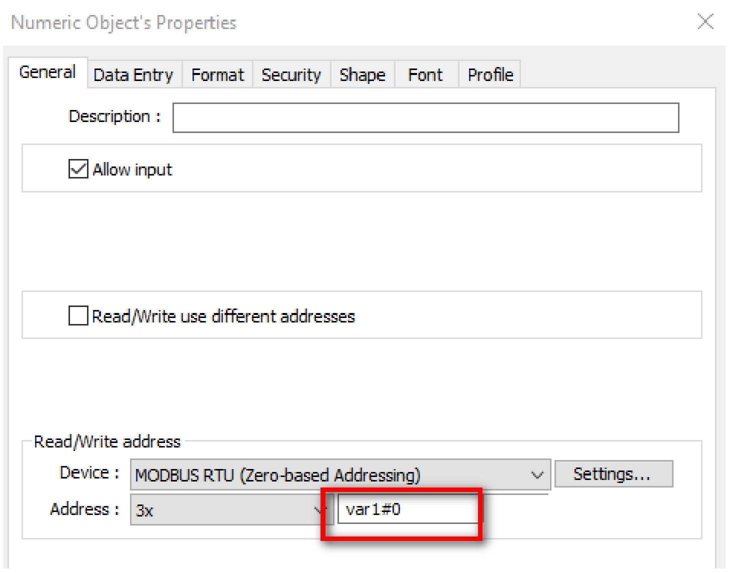
Station Number Variable	Corresponding to the System Register
Var0	LW-10000
Var1	LW-10001
:	:
Var15	LW-10015

The address format of Station Number Variables is var{i}#Addr

The i can be a constant value from 0 to 15.

The Addr stands for the device address. The "#" is a sign that separates the station number and the address.

In this example, the station number is determined by var1. You will need to enter a station number to System Register LW-10001.


Numeric Object's Properties

General | Data Entry | Format | Security | Shape | Font | Profile

Description :

☑ Allow input

☐ Read/Write use different addresses

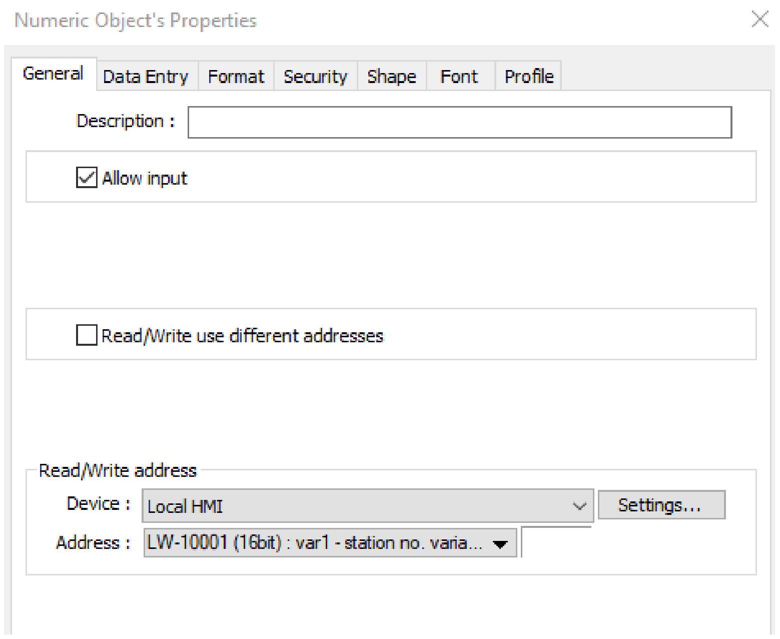
Read/Write address

Device : MODBUS RTU (Zero-based Addressing)

Settings...

Address : 3x var1#0

A Number object specified the address of the Modbus slave


Numeric Object's Properties

Description :

☑ Allow input

☐ Read/Write use different addresses

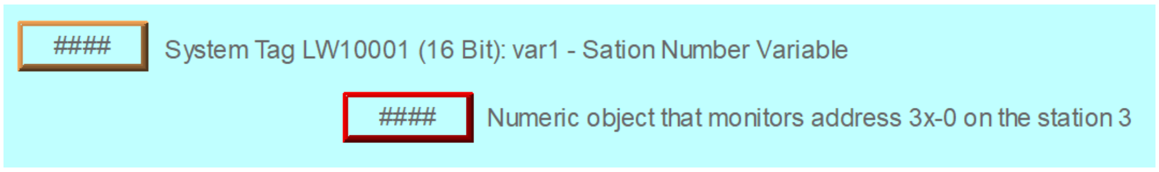
Read/Write address

Device : Local HMI

Address : LW-10001 (16bit) : var1 - station no. varia...

The other Number object specified System Register LW-10001

When you input the constant value **3** to the Numeric object that is specified address LW-10001 during runtime, the other Numeric object will display the value pulled out from the slave ID **3**.


System Tag LW10001 (16 Bit): var1 - Sation Number Variable

Numeric object that monitors address 3x-0 on the station 3