Modbus Slave Tool User Manual

V1.00, Aug. 2025

Written by Mac Cho
Edited by Sunny Chiu

TABLE OF CONTENTS

1. Data Format Differences in Modbus Communication

Modbus was developed by Modicon in 1979. It is a master/slave data communications protocol primarily used for communication between industrial electronic devices, especially programmable logic controllers (PLCs). Today, it has become a de facto standard communication protocol for communication between industrial control systems. The Modbus protocol typically uses serial communication lines (RS-232, RS-485, or RS-422) or Ethernet interface. Serial communication interfaces are well-suited for multi-device and cost-sensitive applications. Ethernet communication is represented by Modbus TCP, which transmits data over Ethernet networks and is commonly used in networked automation equipment and remote monitoring systems.

Although most Modbus devices follow the same protocol standard, slight variations still exist between products from different manufacturers. These minor differences often require engineers to invest considerable time and effort in testing and debugging. The Modbus Slave Tool is designed to help users quickly verify whether the software and hardware are communicating with consistent data formats.

Two discrepancies that may occur between devices and Master software are register addressing basics and data transmission formats.

1.1. Differences in Register Address Basics

The Modbus protocol has 4 types of data, each corresponding to specific I/O operations of a device. The I/O data is accessed with its Modbus address, also called register address. Modbus addresses start with an integer number to specify the data type, a common example is the use of 4xxxx addresses for holding registers. The “4” designates the Modbus data type, holding registers in this case. For the typical PLCs and SCADA systems, the first holding register is denoted as 40001 (1-based addressing). In some Modbus device documentation, the first register address may be labeled as “0000”, which is known as 0-based addressing.

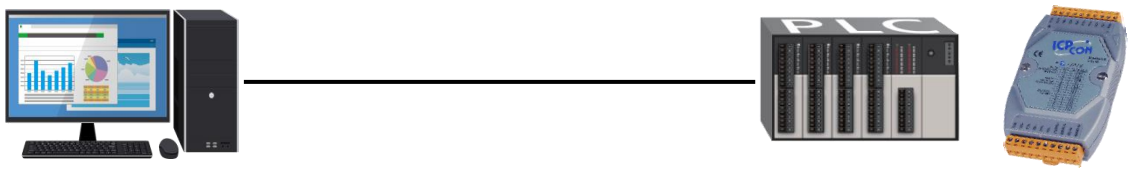
Modbus Data Type

Data Type	PLC/SCADA	Address	Abbr.	Access	I/O
Coils	00001 ~ 09999	0000H~FFFFH	0x	R/W	DO
Discrete Inputs	10001 ~ 19999	0000H~FFFFH	1x	R	DI
Input Registers	30001 ~ 39999	0000H~FFFFH	3x	R	AI
Holding Registers	40001 ~ 49999	0000H~FFFFH	4x	R/W	AO

Each type of Modbus data can be corresponding to specific I/O operations of a device:

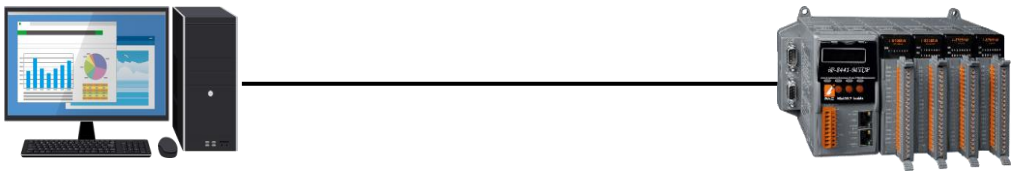
Modbus address	I/O operation
0xxxx	Outputs or reads back the DO data
1xxxx	Reads the DI data
3xxxx	Reads the AI data
4xxxx	Outputs or reads back the AO data

Different Modbus slave devices or Master software from different manufacturers may follow different documentation standards. In typical PLC and SCADA software documentation, 1-based addressing is used. If both the Master software and the slave device (e.g., M-7000) use 1-based addressing, the read/write addresses will match exactly.



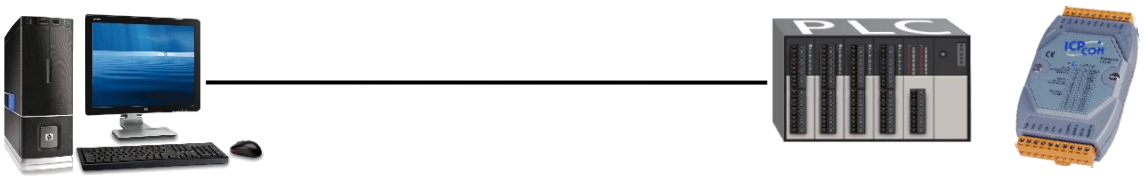
SCADA (1-Based)	Device Address (1-Based)
40001	0001
40002	0002
40003	0003
40004	0004
40005	0005

When the Master software (e.g., Win-GRAF) uses 1-based addressing and the slave device (e.g., iP-8411-MRTU) uses 0-based addressing, the data read from the device will correspond to the hardware's address at [software address - 1].



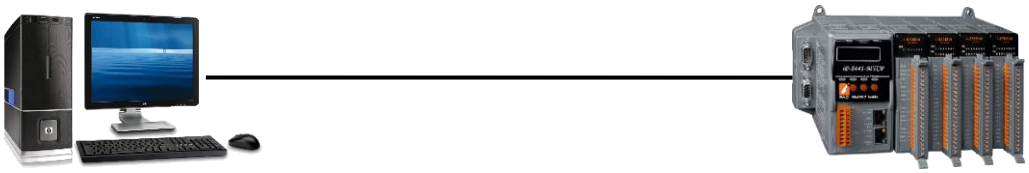
SCADA (1-Based)	Device Address (0-Based)
40001	0000
40002	0001
40003	0002
40004	0003
40005	0004

Conversely, if the master software (e.g., LabVIEW) uses 0-based addressing and the slave device (e.g., M-7000) uses 1-based addressing, the master software will access the data at [hardware address - 1].



Software (0-Based)		Device Address (1-Based)
40000	⟷	0001
40001	⟷	0002
40002	⟷	0003
40003	⟷	0004
40004	⟷	0005

If both master software (e.g., Modbus Utility) and the slave device (e.g., iP-8411-MRTU) use 0-based addressing, the addresses will match directly.



Software (0-Based)		Device Address (0-Based)
40000	⟷	0000
40001	⟷	0001
40002	⟷	0002
40003	⟷	0003
40004	⟷	0004

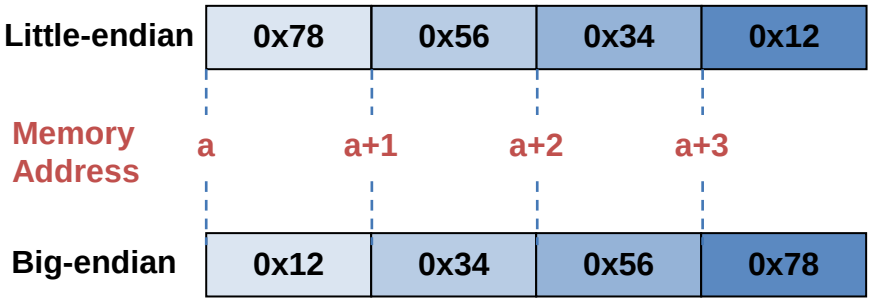
1.2. Differences in Data Transmission Format

When exchanging data between different machines or networks, it's essential to consider that systems may follow different byte-ordering conventions. The two most common conventions are Big-endian and Little-endian:

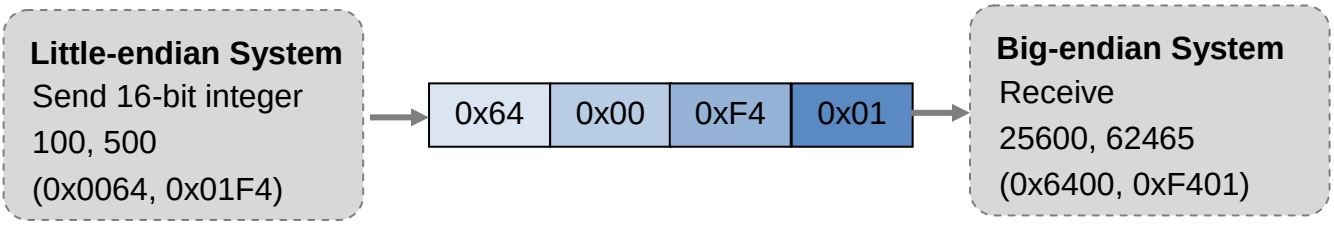
Big-endian: Stores the most significant byte at the lowest memory address.

Little-endian: Stores the most significant byte at the highest memory address.

For instance, a 32-bit integer value 0x12345678 would be arranged differently in memory depending on the endianness format as shown in the figure below.



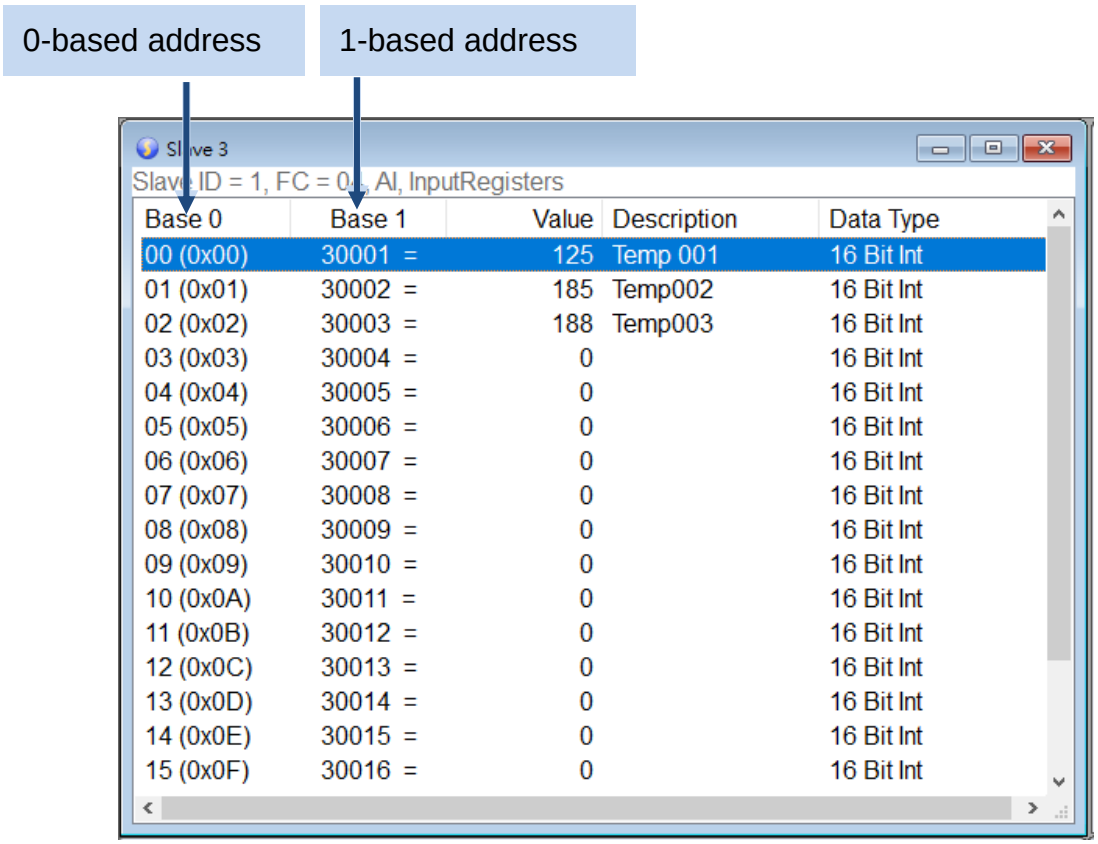
Most systems use the little-endian, so issues caused by endianness differences are relatively rare. However, if you have confirmed that the problematic data is not due to differences in the register address base between the Modbus Master and Slave, you should consider the possibility of endianness mismatch. The example below illustrates a case where inconsistent endianness between systems leads to incorrect data interpretation:



2. Modbus Slave Tool

To assist users in verifying how their master software interprets Modbus data, ICP DAS developed the Modbus Slave Tool, which emulates a Modbus slave device to communicate with the user's Master software. By providing test data, users can easily observe how their Master software processes and displays the information.

The Modbus Slave Tool opens four windows corresponding to the four Modbus function codes (01, 02, 03, 04), displaying each data point with both 0-based and 1-based addresses. All data is transmitted in Little-endian format for consistency.



2.1. Preparation

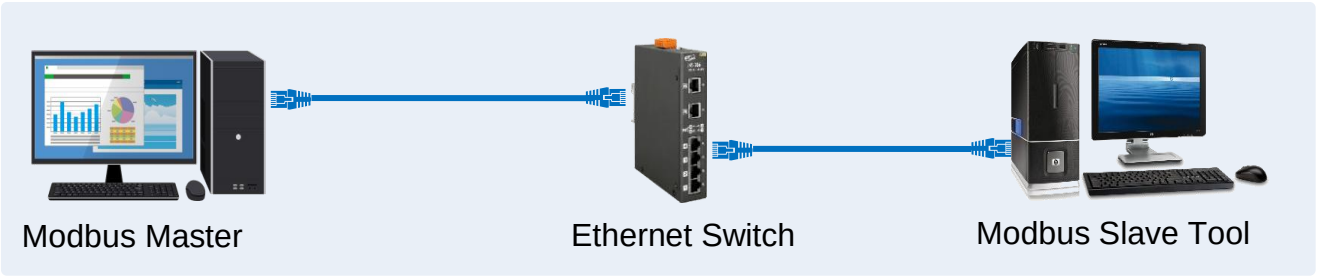
1. Download the “Modbus Slave Tool” software and unzip it.

   https://www.icpdas.com/en//download/index.php?nation=US&kw=Modbus+Slave+Tool

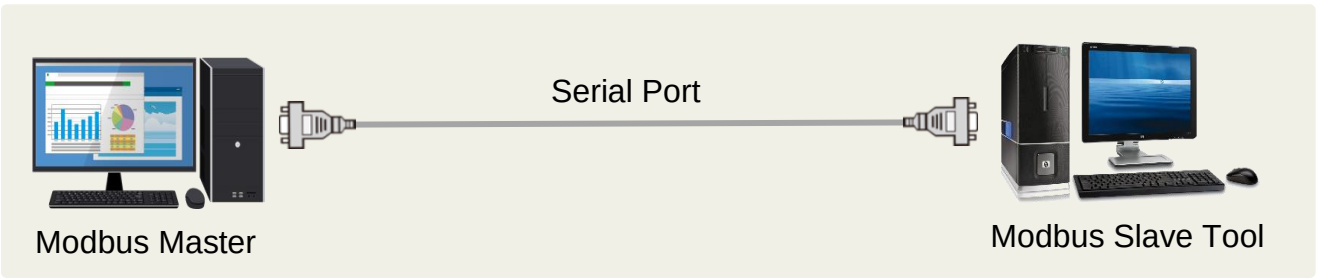
2. Connect to Modbus Master

   The Modbus Slave Tool can simulate devices communicating over both TCP/IP and Serial Port interfaces. Before launching the tool, ensure that the PC running Modbus Slave Tool is properly connected to the Master host either through a serial connection or within the same network environment.

Connecting with the Modbus Master via Ethernet



Connecting with the Modbus Master via Serial Port

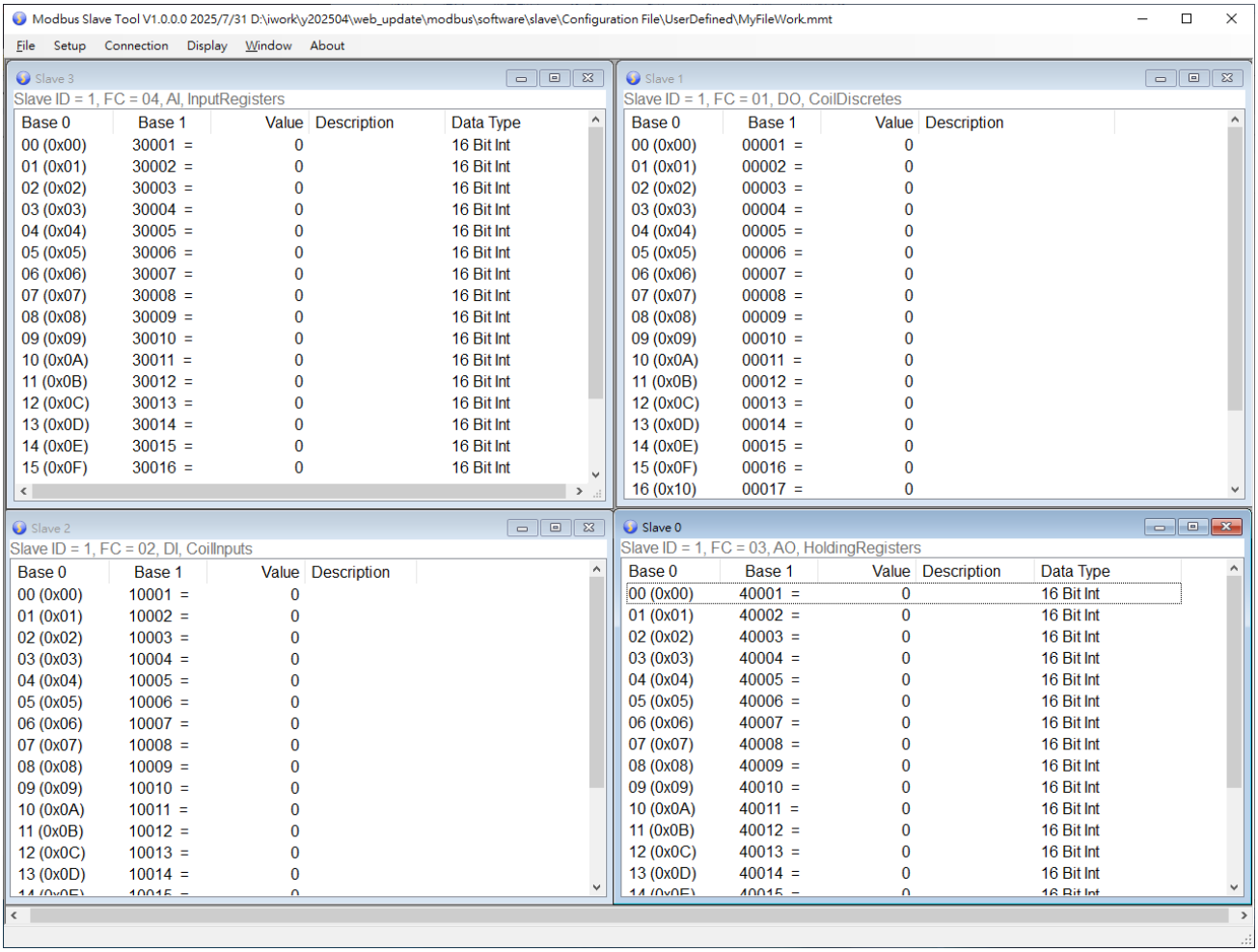


2.2. Configuring Modbus Slave Tool

Modbus Data Type

Data Type	PLC/SCADA	Address	Abbr.	Access	I/O
Coils	00001 ~ 09999	0000H~FFFFH	0x	R/W	DO
Discrete Inputs	10001 ~ 19999	0000H~FFFFH	1x	R	DI
Input Registers	30001 ~ 39999	0000H~FFFFH	3x	R	AI
Holding Registers	40001 ~ 49999	0000H~FFFFH	4x	R/W	AO

When launching ModbusSlaveToolPC.exe, the program will open four separate windows corresponding to the Modbus function code 01, 02, 03 and 04. Each data point has a right-click menu allowing users to add description or configure data value.

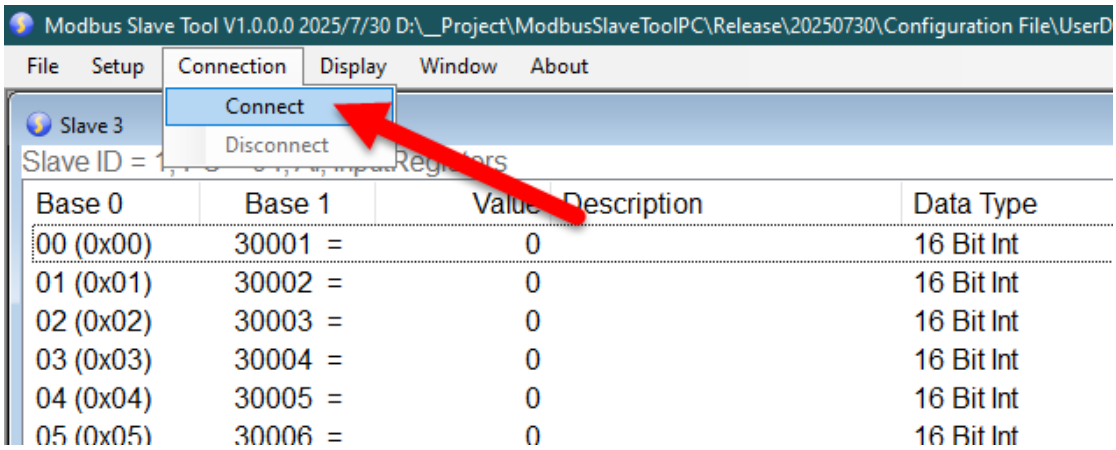


Modbus communication accesses different types of data based on function codes.

Function Code	Description	Address
01 (0x01)	Read the Status of the Coils (Readback DOs)	0xxxx
02 (0x02)	Read the Status of the Input (Reads DIs)	1xxxx
03 (0x03)	Read the Holding Registers (Readback AOs)	4xxxx
04 (0x04)	Read the Input Registers (Reads AIs)	3xxxx
05 (0x05)	Force a Single Coil (Writes DO)	0xxxx
06 (0x06)	Preset a Single Register (Writes AO)	4xxxx
15 (0x0F)	Force Multiple Coils (Writes DOs)	0xxxx
16 (0x10)	Preset Multiple Registers (Writes AOs)	4xxxx

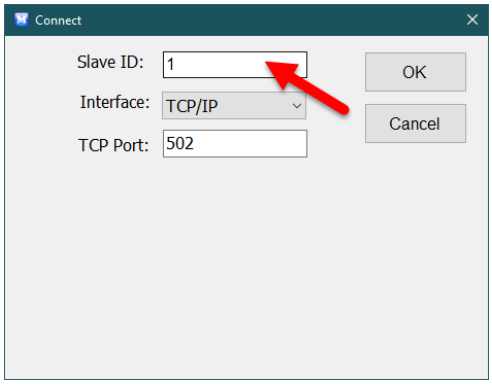
Opening Connection

1. Click on the **Connect** item from the Connection menu.

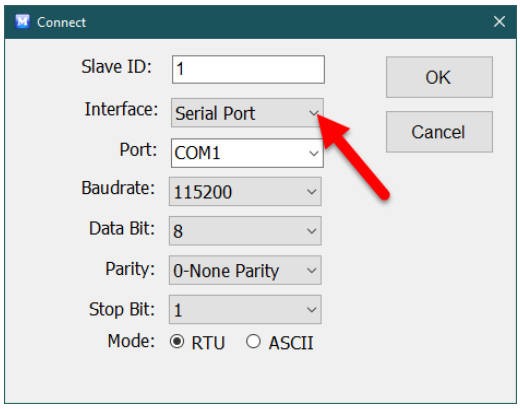


2. Select the interface used to connect to the Modbus Master.

- If TCP/IP is selected to use Ethernet connection, make sure the Slave ID number



- If Serial Port is selected, set the COM Port number and a valid Slave ID number

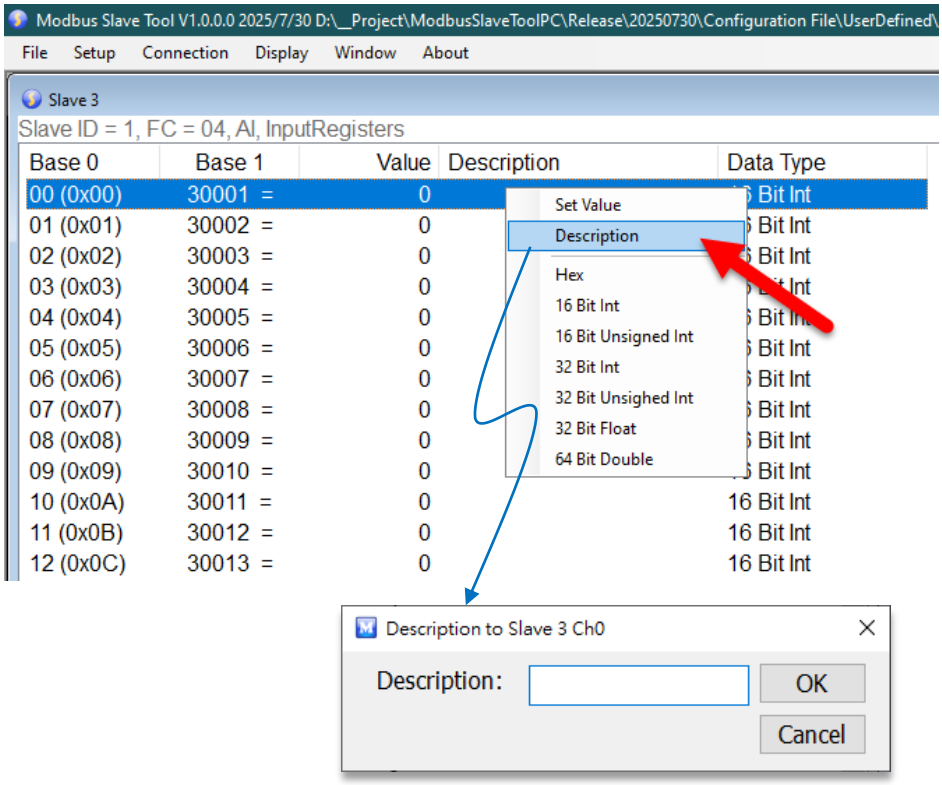


3. After the connection is open, Modbus Slave Tool updates data about once per second.

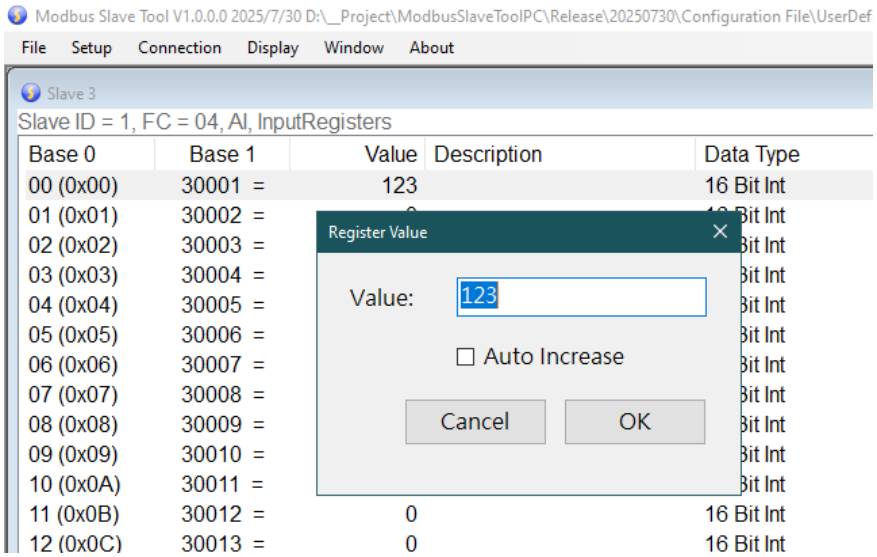
Setting Data

1. Add the description or set the value for each data point by selecting the relevant item from its right-click menu.

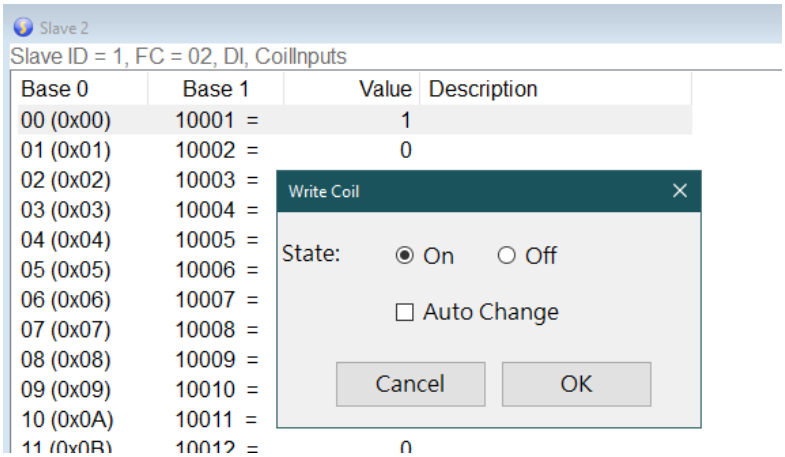
■ Add the description



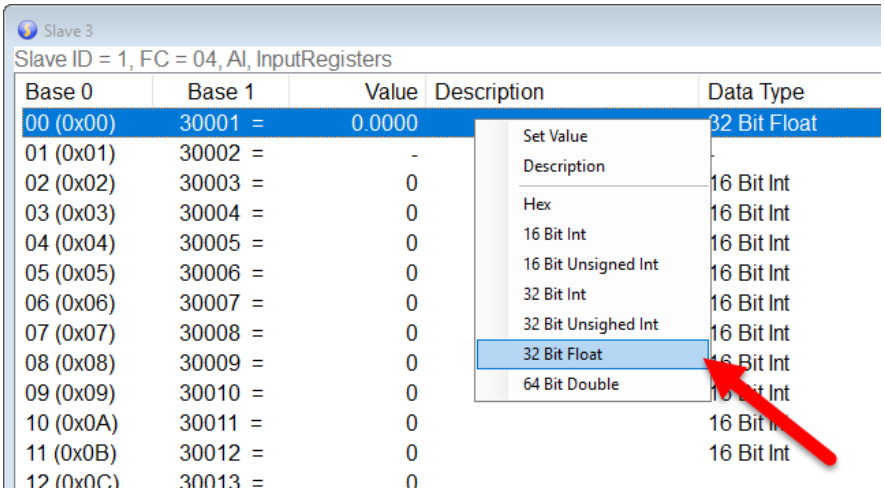
■ Set the value for input registers or holding registers



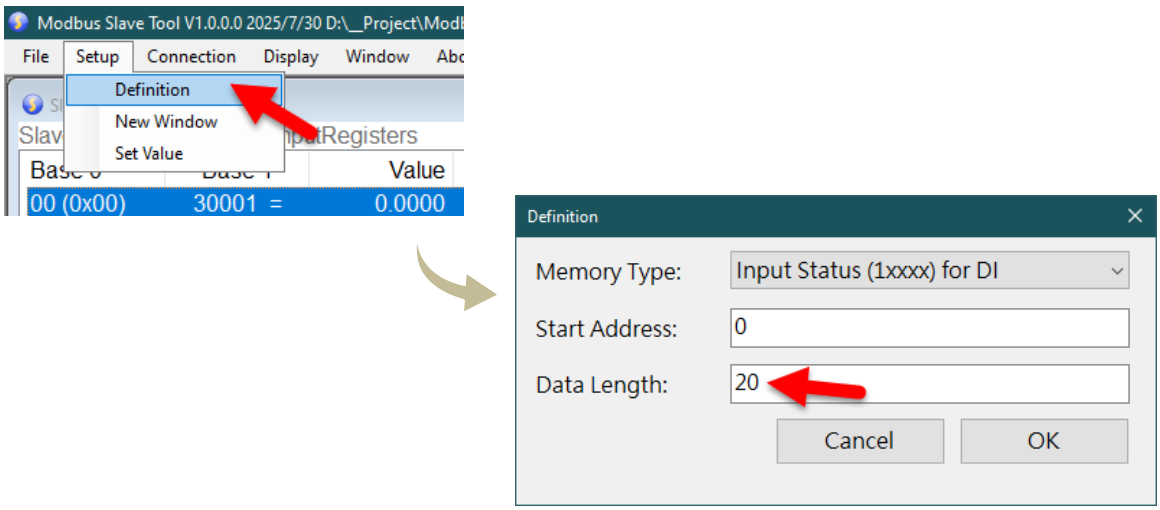
Set the status for coils or discrete inputs



Set the data display format



Set the number of data points to be displayed in the window



Save Project File

You can save your current settings as a project file by clicking File > Save on the toolbar. Project files are stored in the Configuration File > UserDefined folder within the Modbus Slave Tool directory. This allows you to easily reload your configuration for future testing.