DeviceNet Master Utility
User's Manual

Warranty

All products manufactured by ICP DAS are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICP DAS assumes no liability for damages consequent to the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, or for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

The names used for identification only may be registered trademarks of their respective companies.

Revision

Version	Date	Author	Description
3.2.1	2025 09/30	Terry	1. Optimize Task Log output format. 2. Add support for Win7 x86.
3.2.0	2025 06/27	Terry	3. Add the Task Helper feature. 4. Add a Delay to the Batch Access Object. 5. Modify the CSV export format of the Batch Access Object. (This change is not compatible with CSV files exported by Batch Access Object versions prior to 3.2.0.)
3.1.1	2025 03/11	Terry	1. Add new function to set the EPR value. 2. Adjust the firmware update program.
3.1.0	2024 10/18	Terry	Add batch access functionality for slave objects.
3.0.1	2024 5/21	Terry	1. DNM_Utility will issue an error message if the SDK is not installed correctly. 2. Supplementary instructions for using Graphic.
3.0.0	2024 1/12	Terry	This manual is for the DNM_Utility software (3.0), which supports the PISO-DNM100U PCI board and the I-7565-DNM module.

Contents

1. General Information

1.1 DeviceNet Introduction

The CAN (Controller Area Network) is a serial communication protocol, which efficiently supports distributed real-time control with a very high level of security. It is an especially suited for networking "intelligent" devices as well as sensors and actuators within a system or sub-system. In CAN networks, there is no addressing of subscribers or stations in the conventional sense, but instead, prioritized messages are transmitted. DeviceNet is one kind of the network protocols based on the CAN bus and mainly used for machine control network, such as textile machinery, printing machines, injection molding machinery, or packaging machines, etc. DeviceNet is a low level network that provides connections between simple industrial devices (sensors, actuators) and higher-level devices (controllers), as shown in Figure 1.1.1.

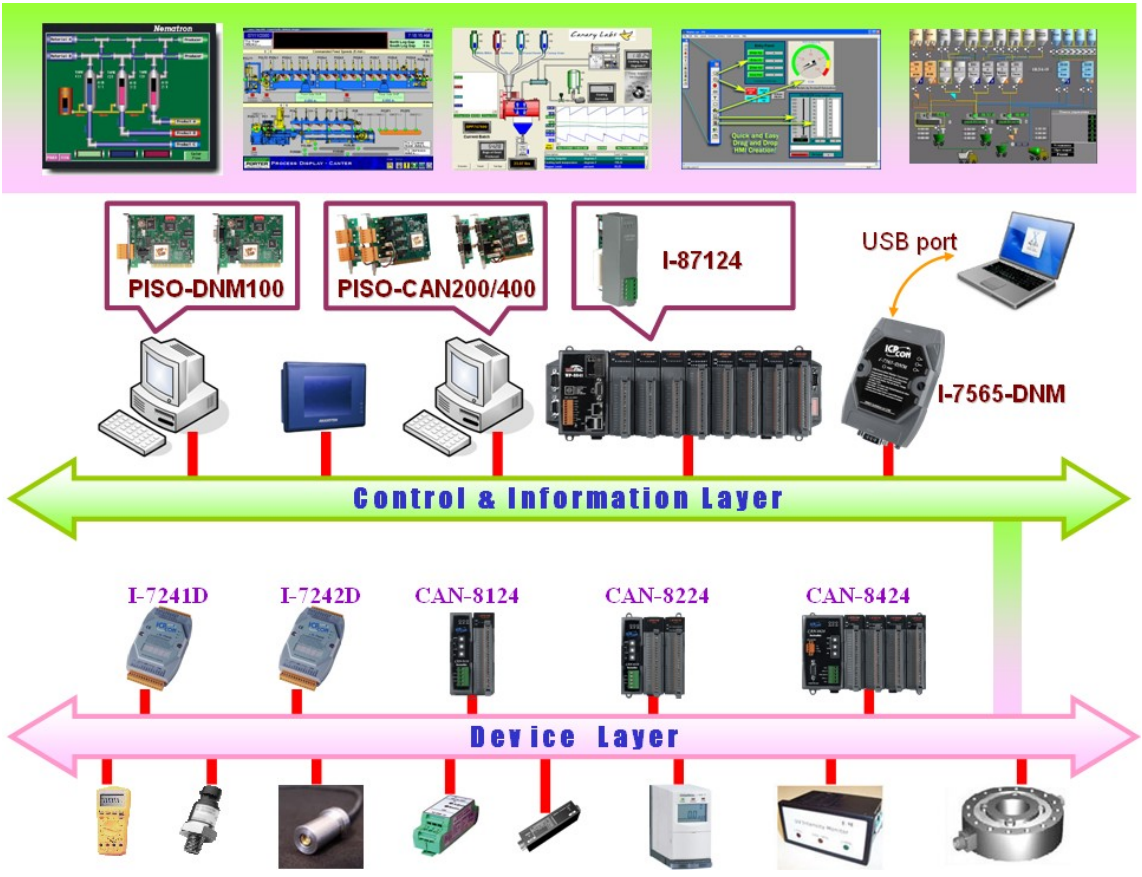


Figure 1.1.1 Example of the DeviceNet network

DeviceNet is a cost effective solution to one kind application of control c\area network. It reduces the connection wires between devices and provides rapid troubleshooting rejection function. The transfer rate can be up to 500Kbps within 100 meters. The transfer distance can be up to 500 meters in 125Kbps (See Table 1.1.1). It allows direct peer to peer data exchange between nodes in an organized and, if necessary, deterministic manner. Master/Slave connection model can be supported in the same network. Therefore, DeviceNet is able to facilitate all application communications based on a redefine a connection scheme. However, DeviceNet connection object strands as the communication path between multiple endpoints, which are application objects that is needed to share data.

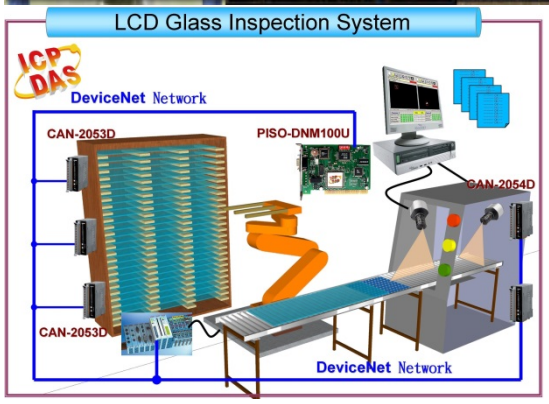
Baud rate (bit/s)	Max. Bus length (m)
500 K	100
250 K	250
125 K	500

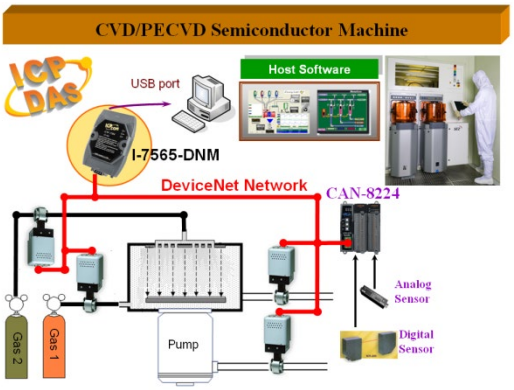
Table 1.1.1 The Baud rate and the Bus length

1.2DeviceNet Applications

DeviceNet is the standardized network application layer optimized for factory automation. It is mainly used in low- and mid-volume automation systems. Some users have also implemented DeviceNet for machine control systems. The main DeviceNet application fields include the following application area (For more information, please refer to www.odva.org):

- Production cell builds and tests CPUs
- Beer brewery
- Equipment for food packing
- Fiberglass twist machine
- Sponge production plant
- Automobile Manufacturing Plant
- Wafer Pod Storage System
- Chemical Vapor Deposition (CVD)
- Dinnerware production
- HVAC module production
- Textile machines
- Trawler automation system
- LCD manufacturing plant
- Large Glass Defect Inspection
- Bottling line
- Equipment for CoWoS





1.3DeviceNet Master Series Product Characteristics

Users don't need to take care of the detail of the DeviceNet protocol. The firmware inside the product mainly supports the Predefined Master-Slave Connection Set and UCMM functions to allow users to merge third party's DeviceNet devices into the DeviceNet network. It can help users to establish the connection with DeviceNet slave devices easily. The general application architecture is demonstrated as Figure 1.3.1

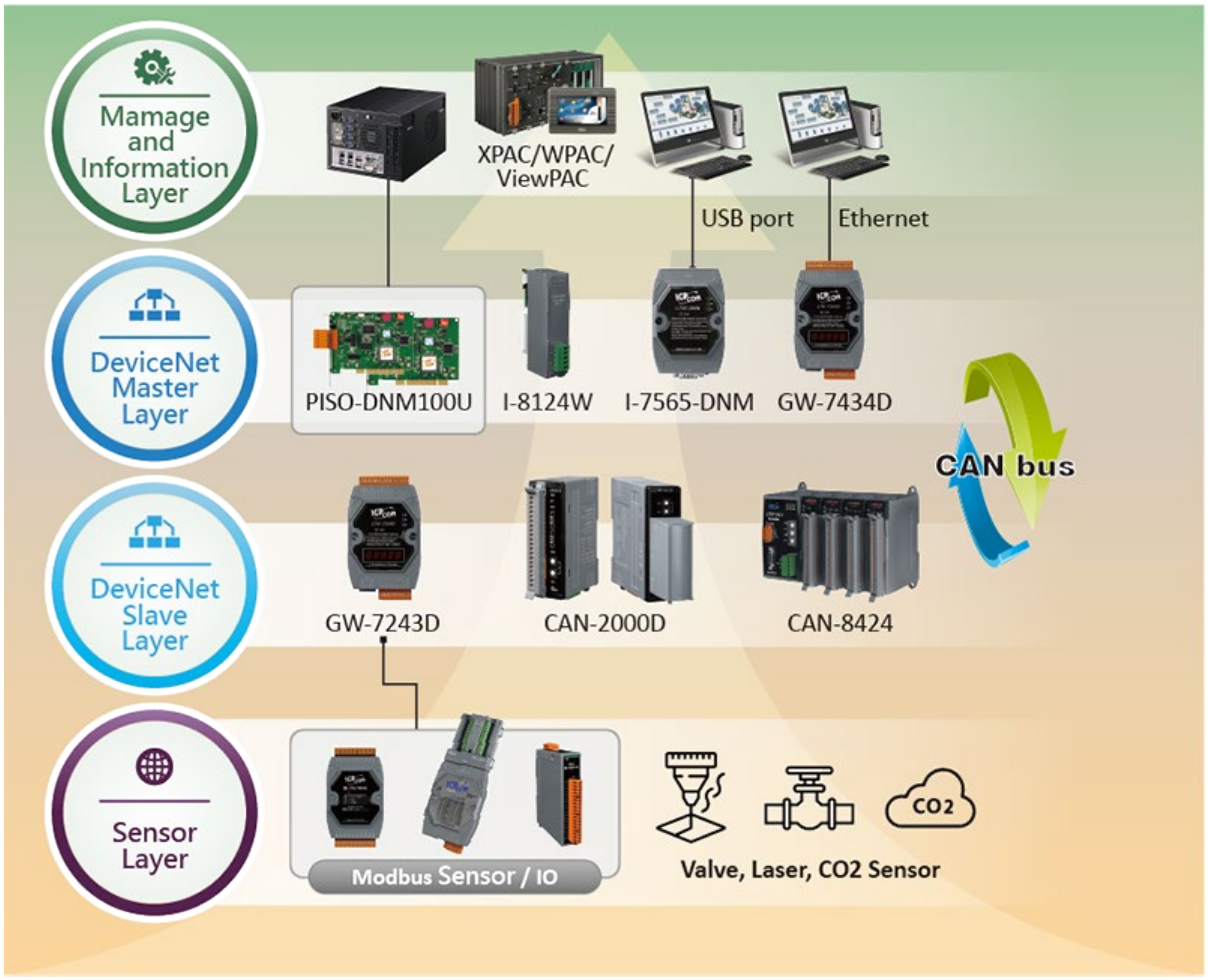


Figure 1.3.1 Application architecture

The DeviceNet protocol firmware provides the DeviceNet Master mechanism to communicate with slave devices by the Predefined Master/Slave Connection Set and UCMM Connection Set. In the DeviceNet communication protocol can be clarify as two forms: One is the Explicit Message and others are I/O Messages. Here, we only provide one explicit message connection and four I/O connections as depicted in Figure 1.3.2

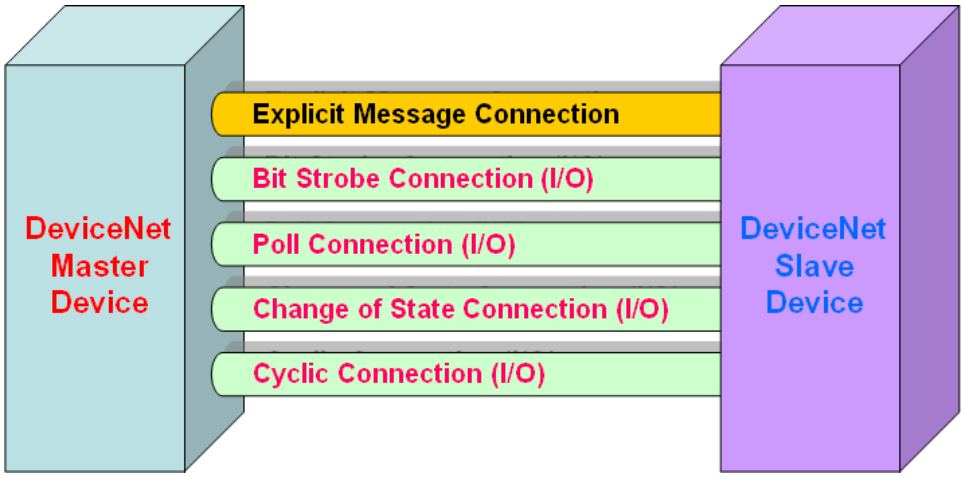


Figure 1.3.2 DeviceNet Messaging

The DeviceNet Communication Protocol is based on the concept of connections method. Master should create connections with slave devices based on the command of exchanging information and I/O data. To establish the master control mechanism, there are only four main steps to be followed. Figure 1.3.3 demonstrates the basic process for the DeviceNet master communication. The every step function is described in below:

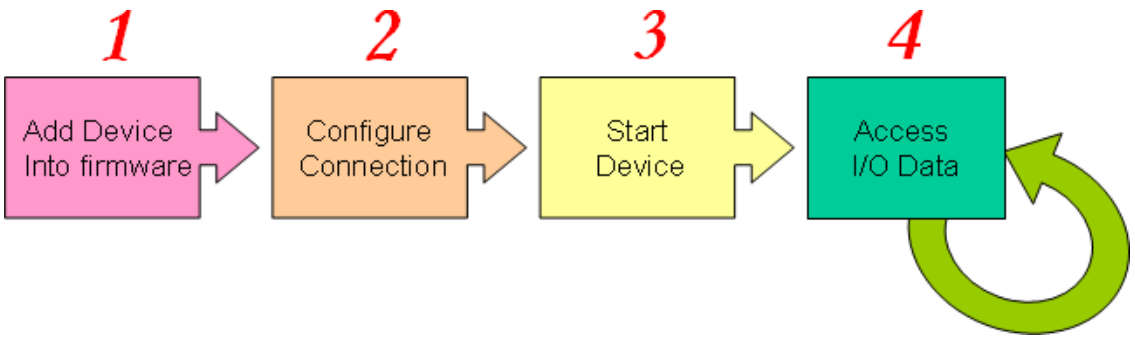


Figure 1.3.3 Four steps to establish connection

1. Add device into firmware

You should provide the slave device's MAC ID to add into firmware.

2. Configure connection

You can check the slave device's I/O connection type and the I/O data length. When configuring the I/O connection, you should provide these parameters.

3. Start Device

After configuring connections, users should start device to establish the connection between the master and the slave devices.

4. Access I/O data

After communicating with slave devices, you can access the I/O data with corresponding read/write function.

2. DeviceNet Master Software Utility for Windows

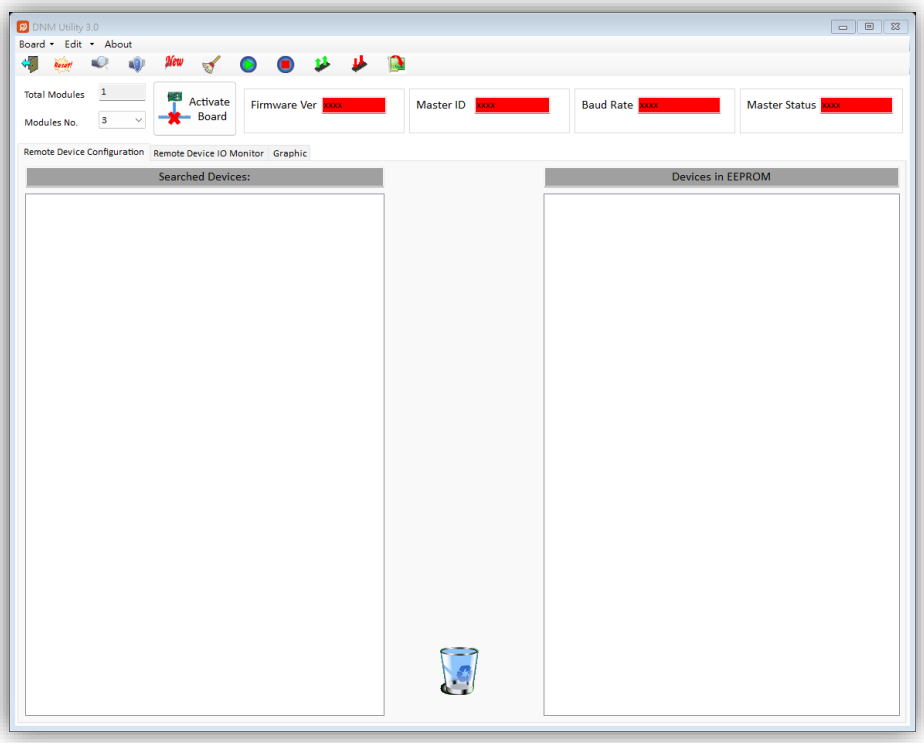
System requirements: Windows 7/10/11 x64 or Windows 7 x86.

.Net Framework 4.8

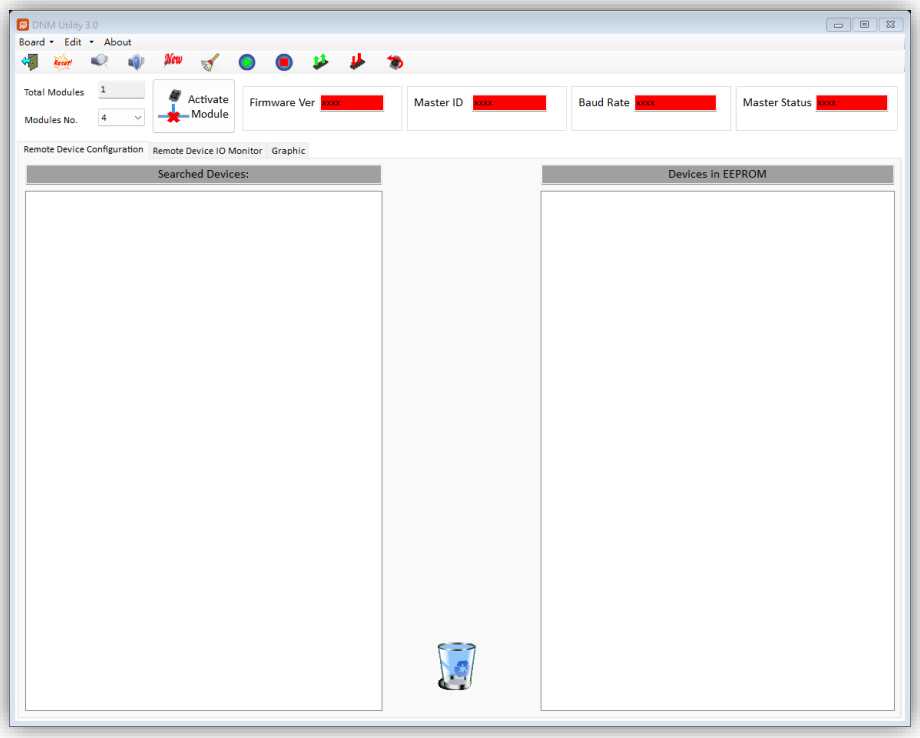
The utility does not work normally if the DeviceNet master series hardware driver is not installed correctly.

After running the utility program, the following window would be shown. Please select the hardware which you are using.

After choosing the “PISO-DNM100U” button, the screenshoot is shown below.



After choosing the “I-7565-DNM” button, the screenshoot is shown below.



2.1 Introduction

The software utility includes various useful functions. These functions help users to diagnose and access the DeviceNet devices. There are four main parts of these functions.

- Diagnosis

DNM_Utility supports to search all devices and specific devices in the network. These functions help the users to configure the connection of the slave devices. Anymore, the software also can diagnose the remote slave devices when building the DeviceNet network.

- Configuration

DNM_Utility supports the users to configure the I/O connection of the devices by searching devices or manual setting. After configuring the I/O connection, the information would be saved into the EEPROM of the PISO-DNM100 or I-7565-DNM. The users can export the data from the EEPROM easily. Correspondingly, the users can import the data into the EEPROM.

- Remote I/O access

DNM_Utility can easily to access the I/O data of all the slave devices. The users can monitor the input data of the specific slave device and change the output data to the remote slave device with this utility.

- Graphic

DNM_Utility assists in analyzing the received I/O data. The users only need to load the information of the slave devices. The utility will convert the raw data into meaningful physical quantity information based on the information you provide and present it in the form of a chart. You can easily observe the changes in all received data through the chart.

- Batch Access Object

DNM_Utility provides a batch transmission function for Explicit messages, allowing users to handle parameter settings all at once.

- Task Helper

DNM_Utility provides a task scheduling function. Here, you can configure I/O sequence number output, I/O return value analysis, and Explicit return value analysis. Task scheduling helps you test the slave's modules.

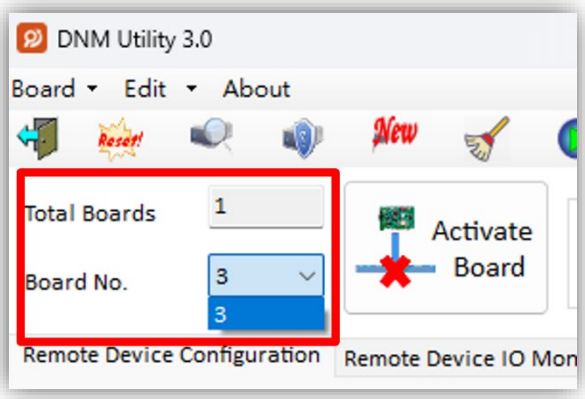
2.2Common Features

2.2.1Where to find the Hardware Information

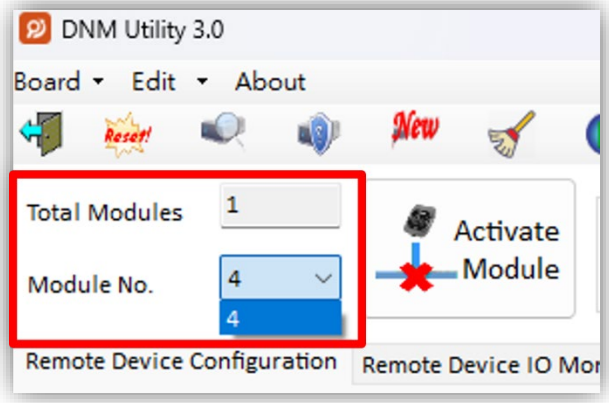
The utility would search the number of master devices in your PC automatically. It shows the count of the boards which have been found.

For PISO-DNM100U, the utility also lists the ID of all boards in the “Board No.” field.

For I-7565-DNM, the utility also lists the ID of all modules in the “Module No.” field.



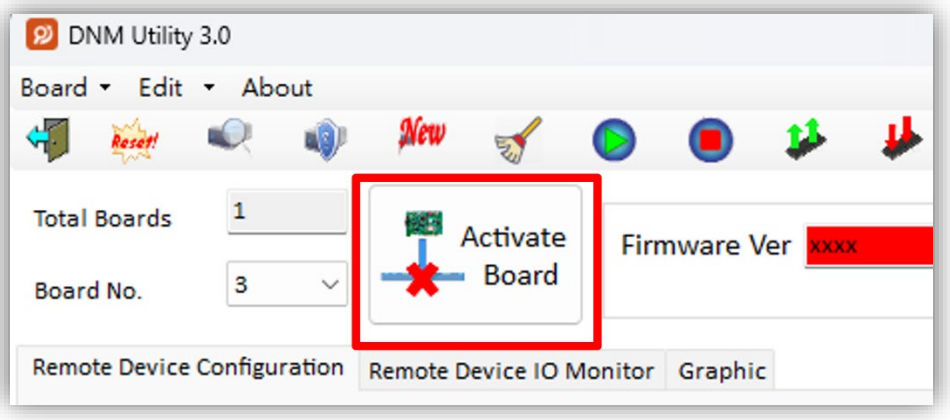
The picture for the PISO-DNM100U



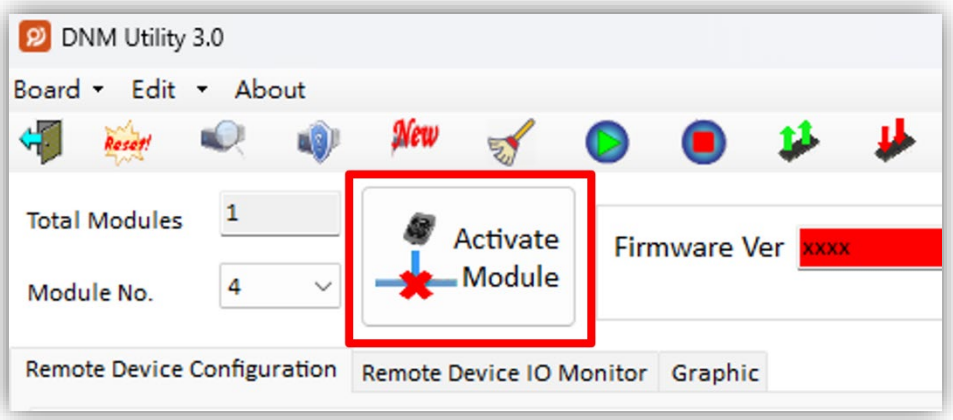
The picture for the I-7565-DNM

2.2.2 How to start using the utility

Before using this utility, the users should click “ActivateBoard” or “ActivateModule” button to activate the specific DeviceNet master hardware. That would initialize the DeviceNet master device which you have selected in the “Board No.” or “Module No.” field.

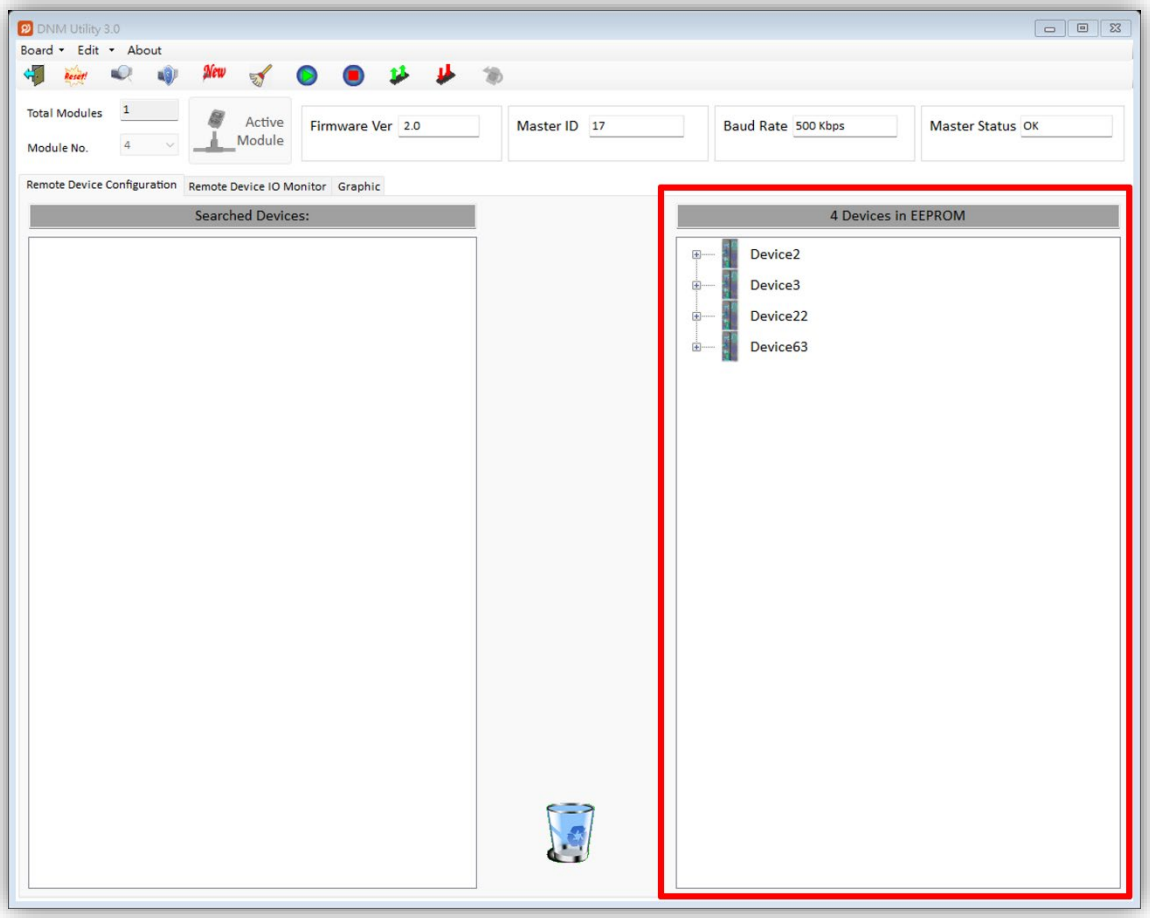


The picture for the PISO-DNM100U



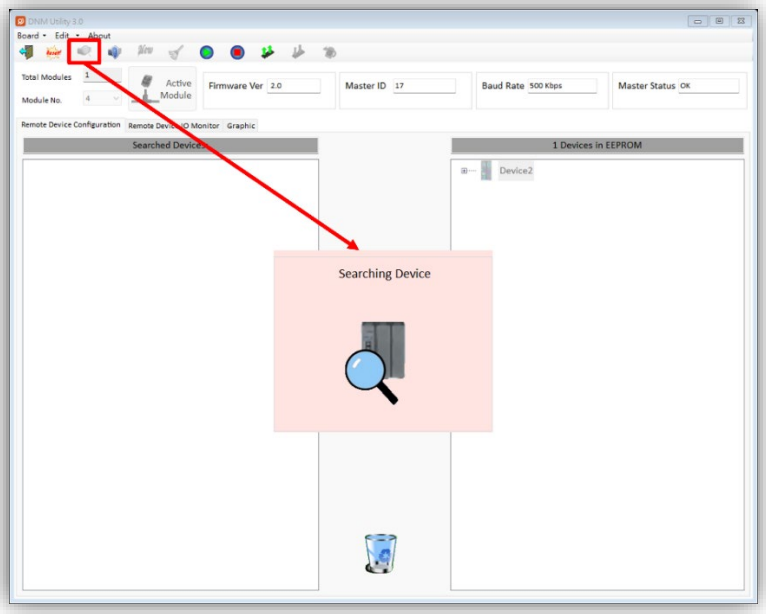
The picture for the I-7565-DNM

After activating the board, the utility will read all configurations from the EEPROM. After reading the configuration from EEPROM of DeviceNet master device successfully, the utility shows the information in the “Devices in EEPROM” field.

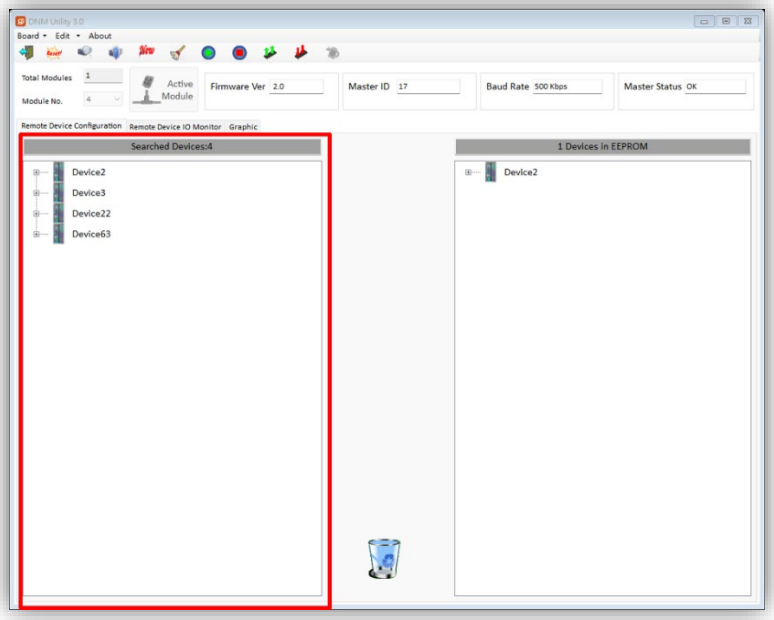


2.2.3 How to search the slave devices

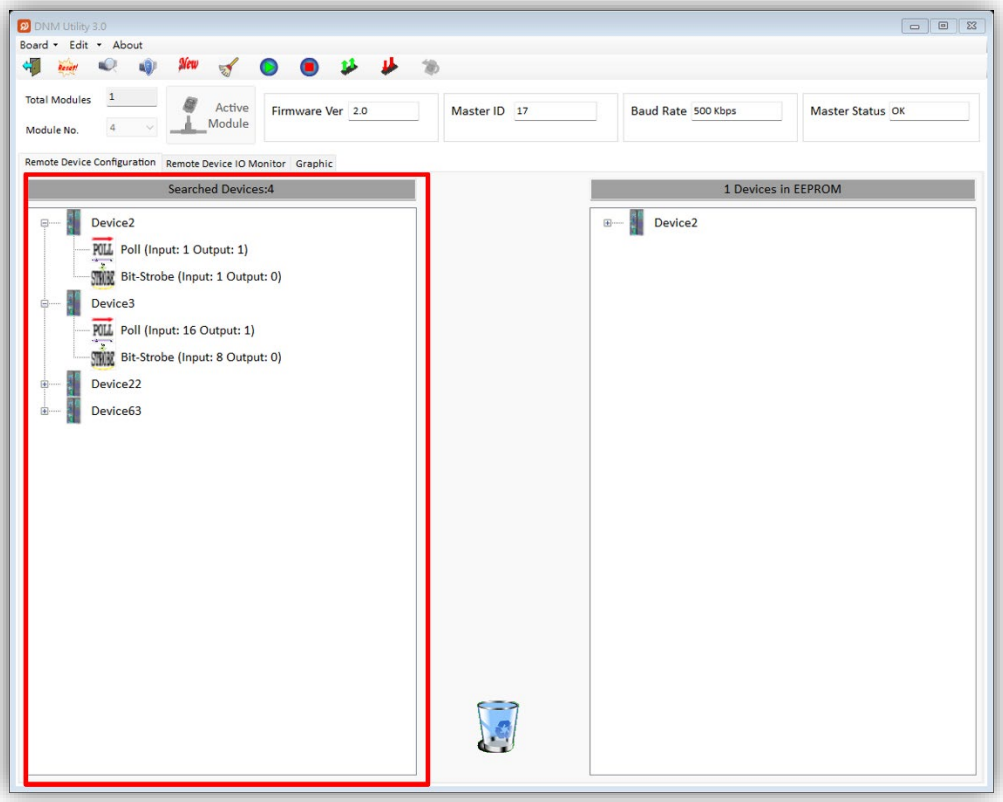
After the board has been activated, the users can press the “Search all Devices” button shown below. As the users press the button, the warning message would pop-up. In this example, we can ignore it. The screen shows a waiting dialog. It takes about 30 seconds to search the whole slave devices in the network. The number of scanned devices is 64.



After finishing the searching procedure, the utility shows the information of all the slave devices in the “Searched Devices” field.

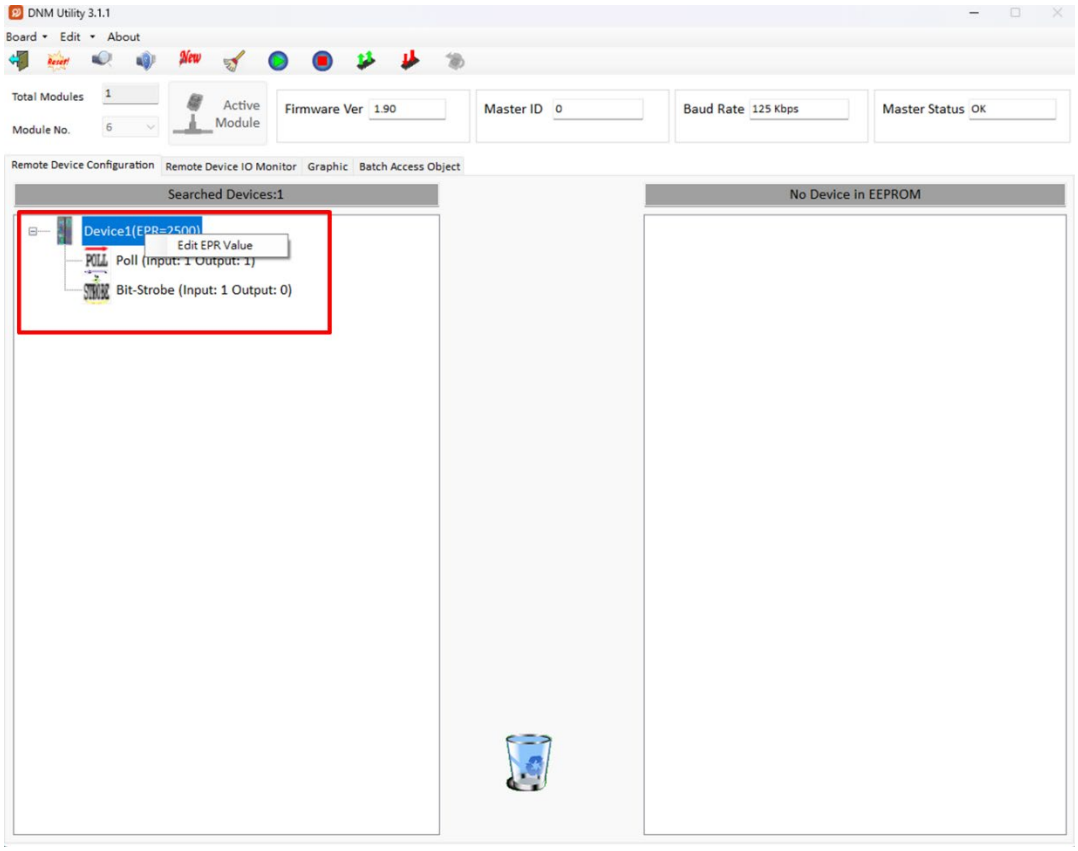


The users can expand the device to find out more I/O connection information of those devices. The users can use this I/O information to develop your configuration in the EEPROM.

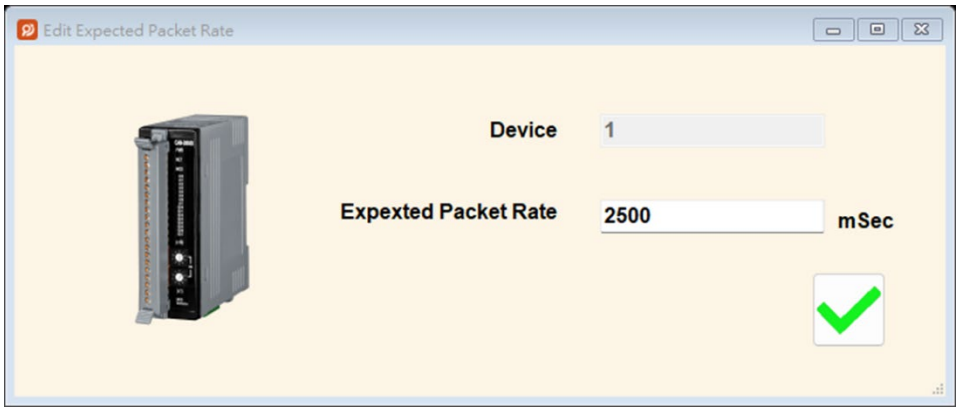


2.2.4 How to Adjust the EPR Value

After searching for all modules, you can change the I/O Expected Packet Rate value by right-clicking with the mouse.



After completing the settings, click the green checkmark to save the configuration.

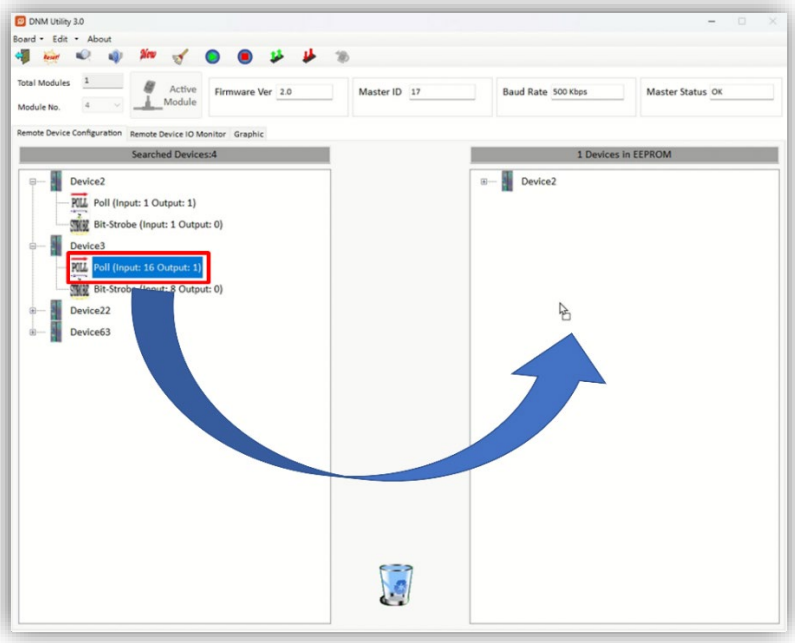


2.2.5 How to add I/O information into the EEPROM

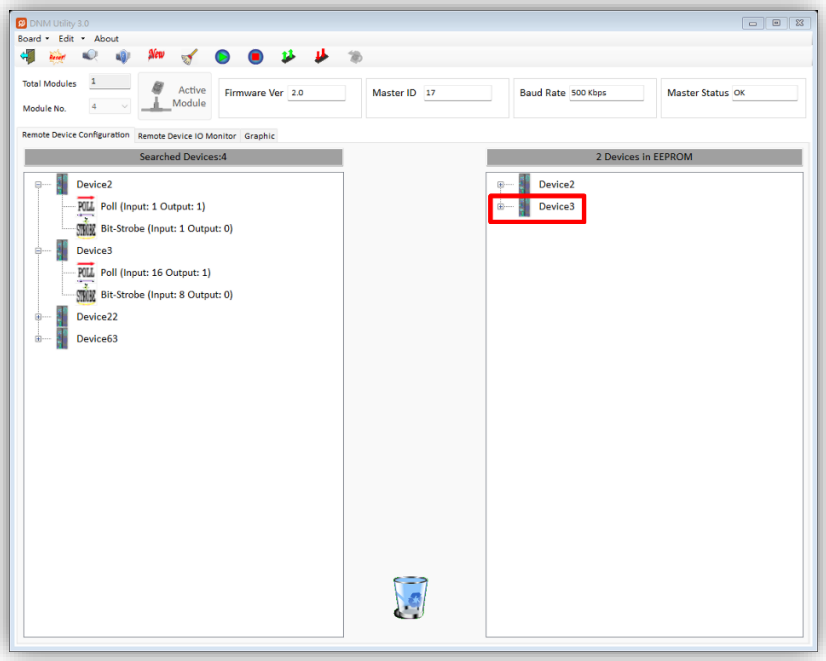
Please activate your board or refer to section 2.2.2

Please search all devices or refer to section 2.2.3

Please select one of the I/O connection items in the “Searched Devices” field. And drag the item into the “EEPROM“ field.



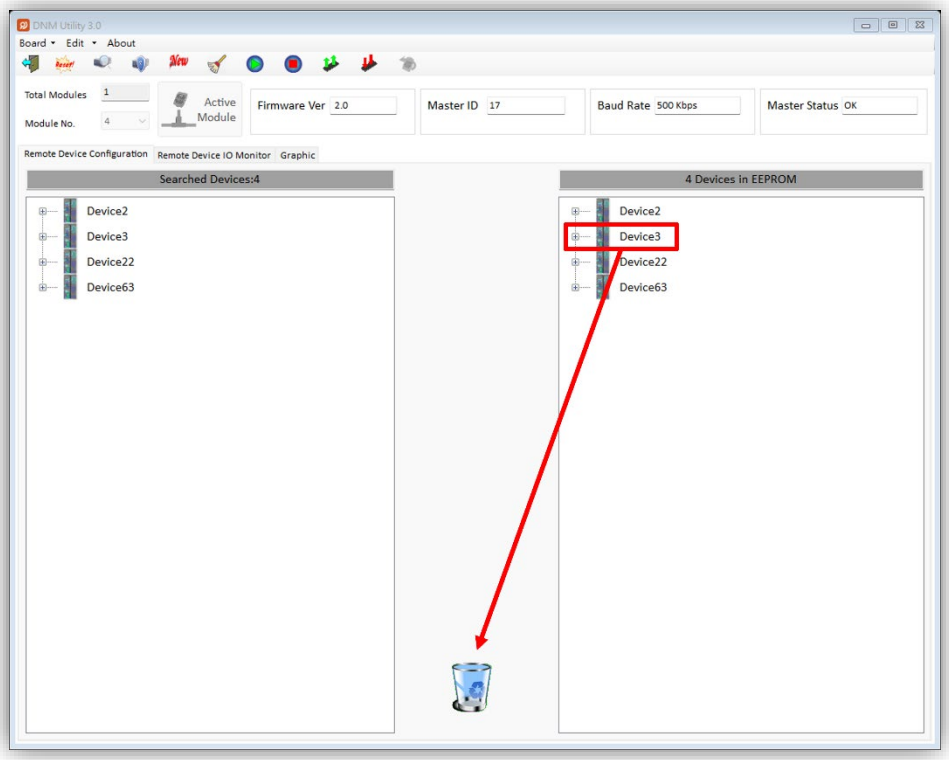
If the mission is successful, the users can find the selected item has been added into the “EEPROM” field.



2.2.6 How to remove I/O information from the EEPROM

Please activate your board or refer to section 2.2.2

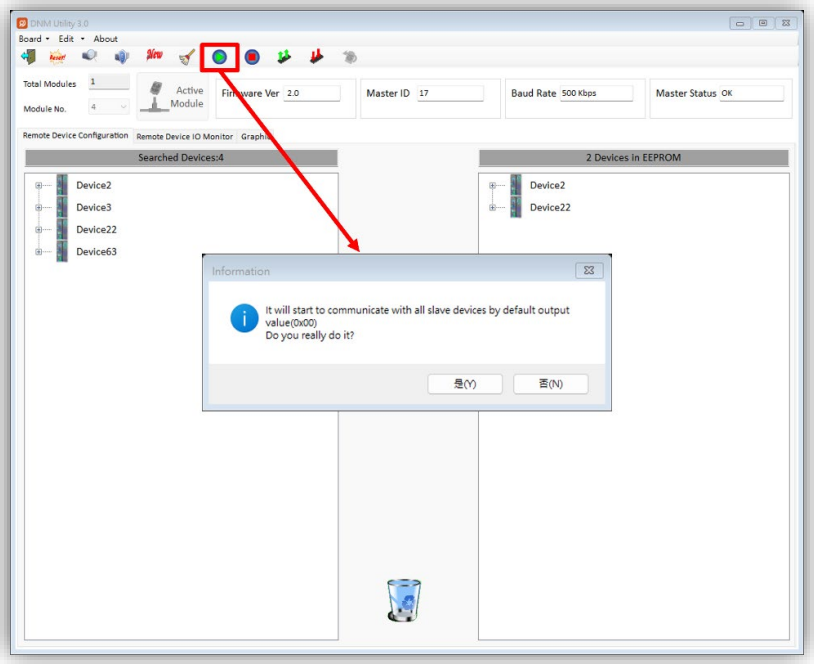
Please select one of the device items in the “EEPROM” field. And drag the item into the “trash can” image.



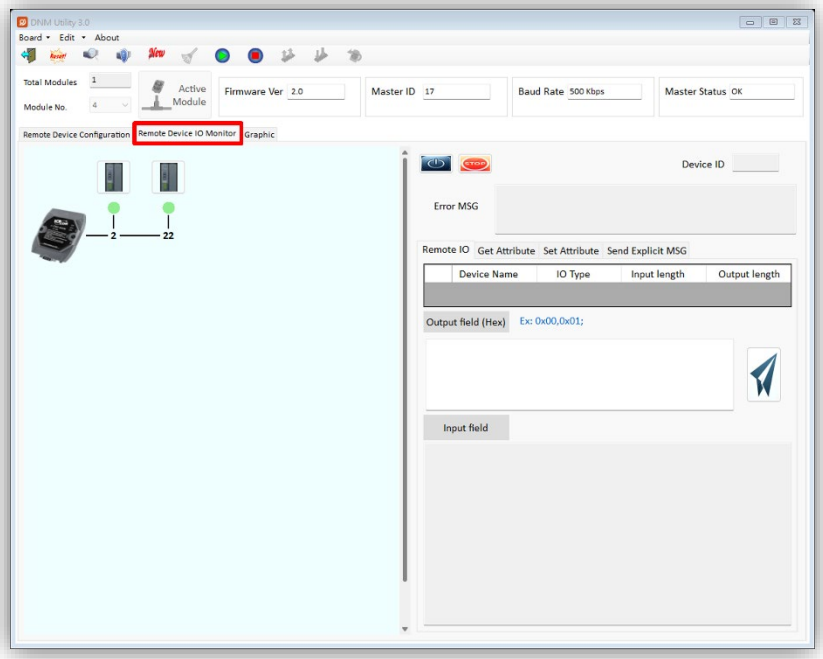
2.2.7 How to read/write the I/O data form the slave device

If the users have no I/O configuration in the EEPROM, please refer to section 2.2.4 to add at least one I/O configuration.

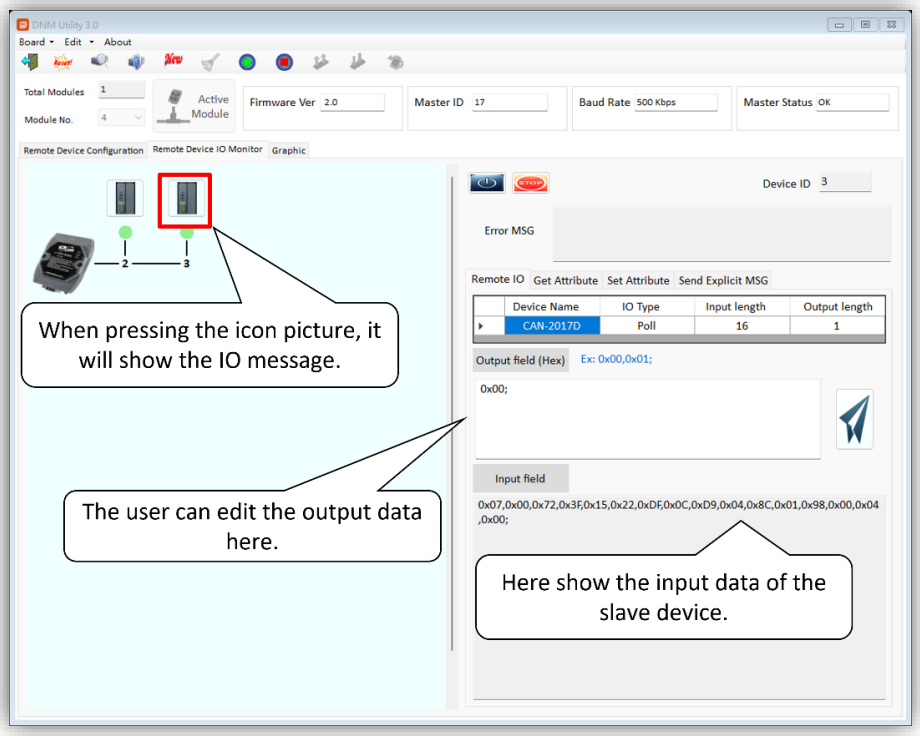
Please press “Start all Device” button to communicate with all slave devices. The information message would pop-up. In this example, we can ignore it.



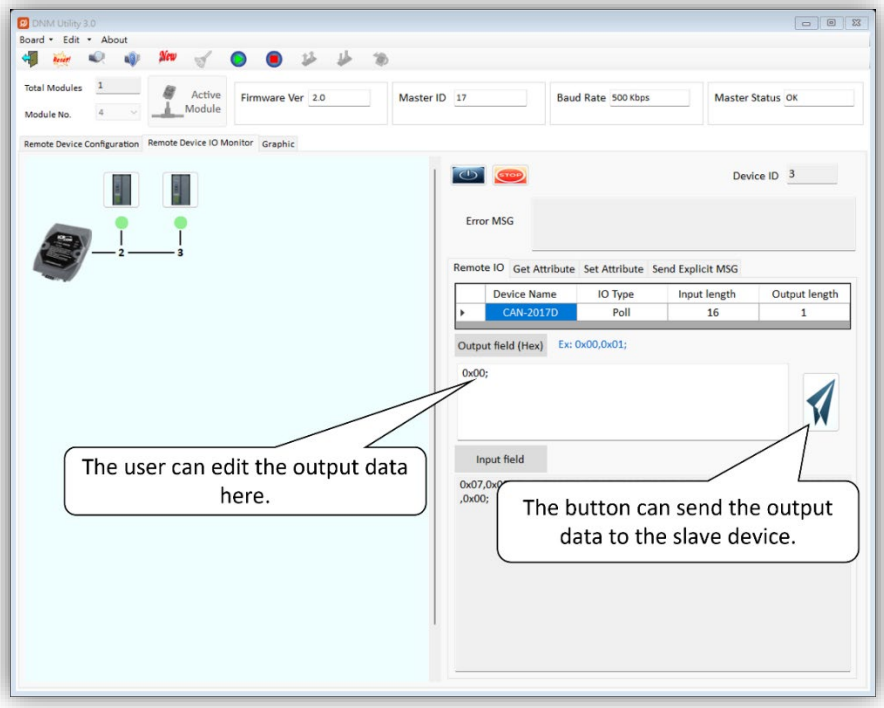
The users can click “Remote Device I/O Monitor” page to view the I/O data of the slave devices.


Board
Edit
About
Total Modules
Module No.
Active Module
Firmware Ver 2.0
Master ID 17
Baud Rate 500 Kbps
Master Status OK
Remote Device Configuration
Remote Device IO Monitor
Graphic
2
22
Device ID
Error MSG
Remote IO
Get Attribute
Set Attribute
Send Explicit MSG
Device Name
IO Type
Input length
Output length
Output field (Hex)
Ex: 0x00,0x01;
Input field

The users can press the icon picture to display the device information, including the device name, device ID and input data.



The users can press “Write output data” button to send the output data to the slave device.

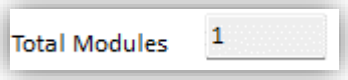


2.3 Description of the Buttons

Here is the description of the buttons in the software utility.

2.3.1 Total Board Number

or



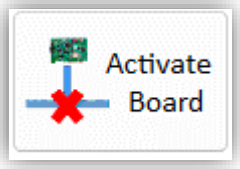
This field shows the total number of all DeviceNet series hardware in the PC. It will detect the DeviceNet hardware automatically when running this software. If the number is 0, the users can not use this software. Please check the driver of the PISO-DNM100U and I-7565-DNM.

2.3.2 Board Number

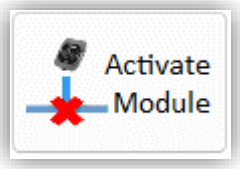
or

There is a DIP-Switch on the PISO-DNM100U. The number of the DIP-Switch is called “Board No”. The I-7565-DNM uses the USB port in the PC. The USB port number is called “Module No”. The drop-down list will show the DIP-switch number or USB port number of all DeviceNet hardware in the PC. The users can select a “Board No” or “Module No” to be activated.

2.3.3 Active Board



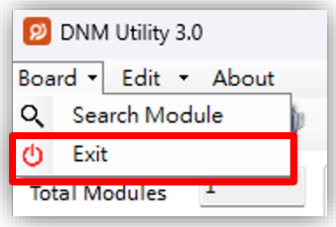
or



This button could activate the DeviceNet hardware which is selected in the “Board No” or “Module No” field. The users should click this button before using other functions. (Except for “Update New Firmware”)

2.3.4 Exit

or



This button will exit the utility.

2.3.5 Reset Firmware

This button can restart the firmware of the PISO-DNM100U or I-7565-DNM. If the users have changed the master's ID or baud rate, you must restart firmware to make change enable.

2.3.6 Search All Device

This button can search all the slave devices in the network.

Notice: When the master is communicating with the slave devices, please don't use this function to avoid breaking the connection to the slave devices.

2.3.7 Diagnose Specific Device

This button can diagnose the specific slave device in the network.

Notice: When the master is communicating with the slave devices, please don't use this function to avoid breaking the connection with the slave devices.

2.3.8 Start All Device

This button can start to communicate with all slave devices which have configured in the EEPROM.

Notice! If the slave device contains output channels, the master will send default value (0) to the output channels. If you do not wish to set the initial value to 0, please use the "Start Device" button in Section 2.3.15.

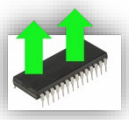
2.3.9 Stop All Device

This button would disconnect the communication with all slave devices which have configured in the EEPROM. All remote slave devices will change to the “off-line” state.

2.3.10 Clear the EEPROM

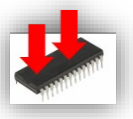
This button will clear all the slave devices information stored in the EEPROM.

2.3.11 Export from EEPROM



This button can export the information from the EEPROM of the PISO-DNM100 or I-7565-DNM. The information will be saved a file (*.EEP).

2.3.12 Import into EEPROM

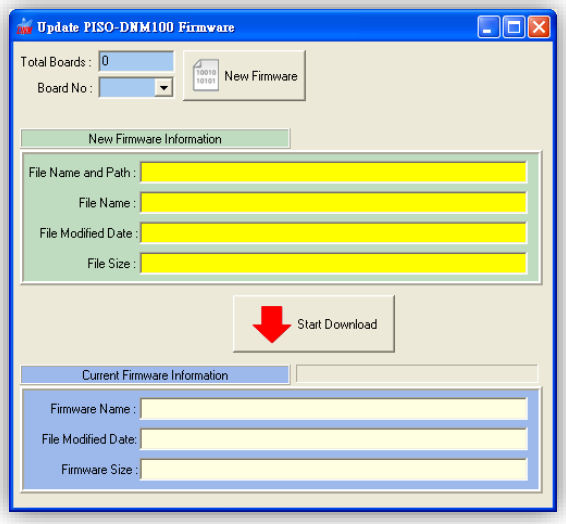


This button can import the EEP file into the EEPROM of the PISO-DNM100 or I-7565-DNM from the specific EEP file.

2.3.13 Update New Firmware

or

This button can update the firmware of the PISO-DNM100U or I-7565-DNM. If ICPDAS have new version of the firmware, the users can click this button to update it. After clicking the button, the users can see the following window as figure 2.3.1.



or

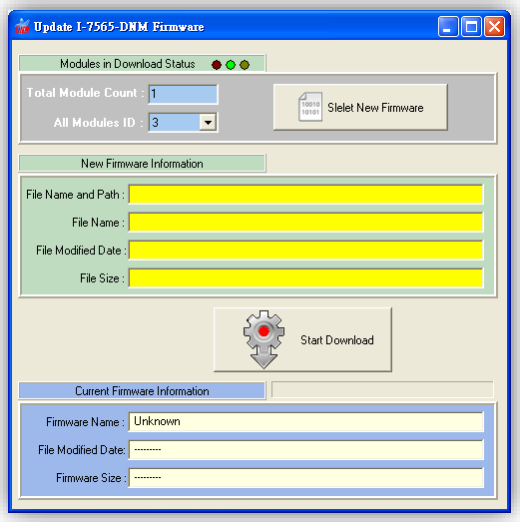


Figure 2.3.1 Update Firmware

Please follow the steps to update the new firmware.

1 Please select the number in the “Board No” or “Module No” field.

2 Click the “New Firmware” button to open the new firmware.

3 Click the “Start Download” button to download the new firmware.

4 Wait for a while, the user will see a “Download OK” message box.

2.3.14 Add New Device by user-defined

This button can add a new device into the EEPROM by your need. When click the button, the “New Device configuration” dialog would be shown as picture below.

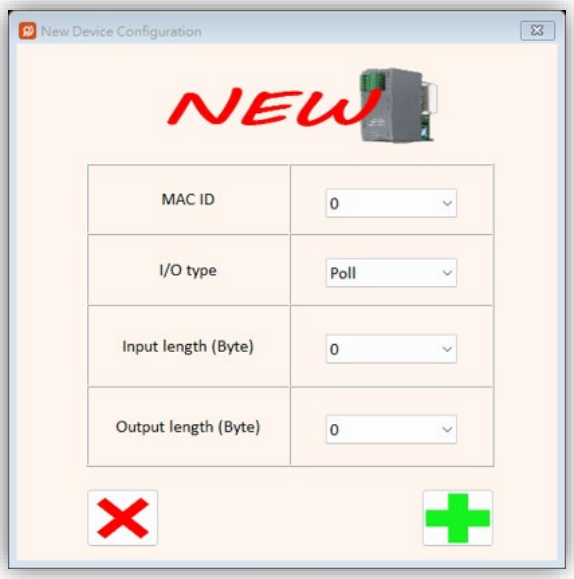


Figure: Add New Device by user-defined

2.3.15 Start Device

This button can start to communicate with the selected devices. If the function is successful, the information will be displayed in the “Device Name” field, “Device ID” field and “Input Data” field and then the LED image will turn green as shown in picture below.

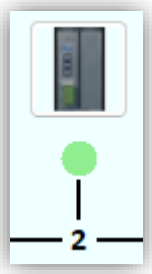


Figure: Green color LED

2.3.16 Stop Device

This button can stop to communicate with the device which the users have selected. If the function is successful, the LED image will turn red as shown in picture below.

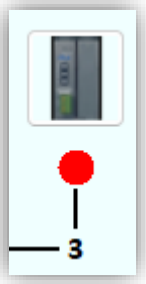
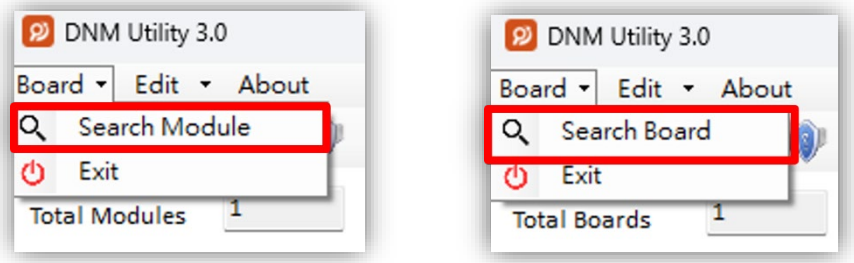


Figure: Red color LED

2.3.17 Search Board

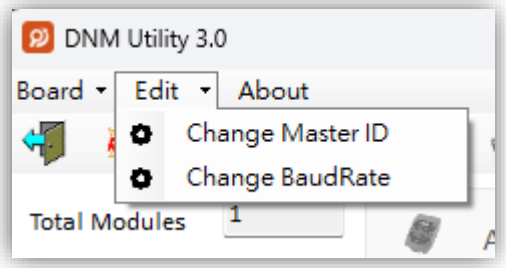


or

"Search Module" or "Search Board" will search for the number of master devices on your PC. After selecting this function, a warning window will pop up. Choosing yes will disconnect from the currently connected master device and search for all master devices on your PC. The results will be displayed in the "Total Modules/Boards" field and the "Module/Board No." field.

Notice! This function will disconnect from the master device and interrupt all current ongoing tasks.

2.3.18 Change Master ID and Baud Rate



If the users want to change the MAC ID of the DeviceNet Master or the baud rate of the network, you can click one of these two items. The users will see the following window as picture below.

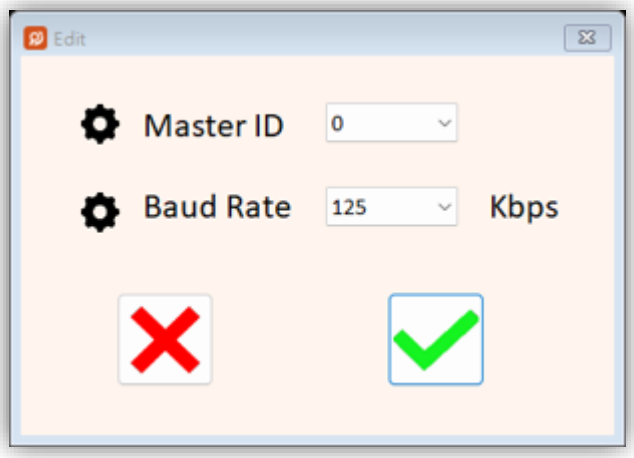


Figure: Change MAC ID and Baud Rate Dialog

The users can select your own setting and then press the "✔" button. The "✖" button is to close the dialog without changing any setting.

After completion, call Reset Firmware to apply all settings.

3. Advanced Feature: Chart

In the Graphic page, DNM_Utility assists in converting the received slave information into physical quantities and presents them in a trend graph format on the screen. The user only needs to provide the slave information in XML format (please refer to Section 3.2 for the XML content). After starting Rx analysis, the user can observe the data and its changes after the slave return values are converted into physical quantities. The figure below shows the voltage values after converting the slave return values into physical quantities.

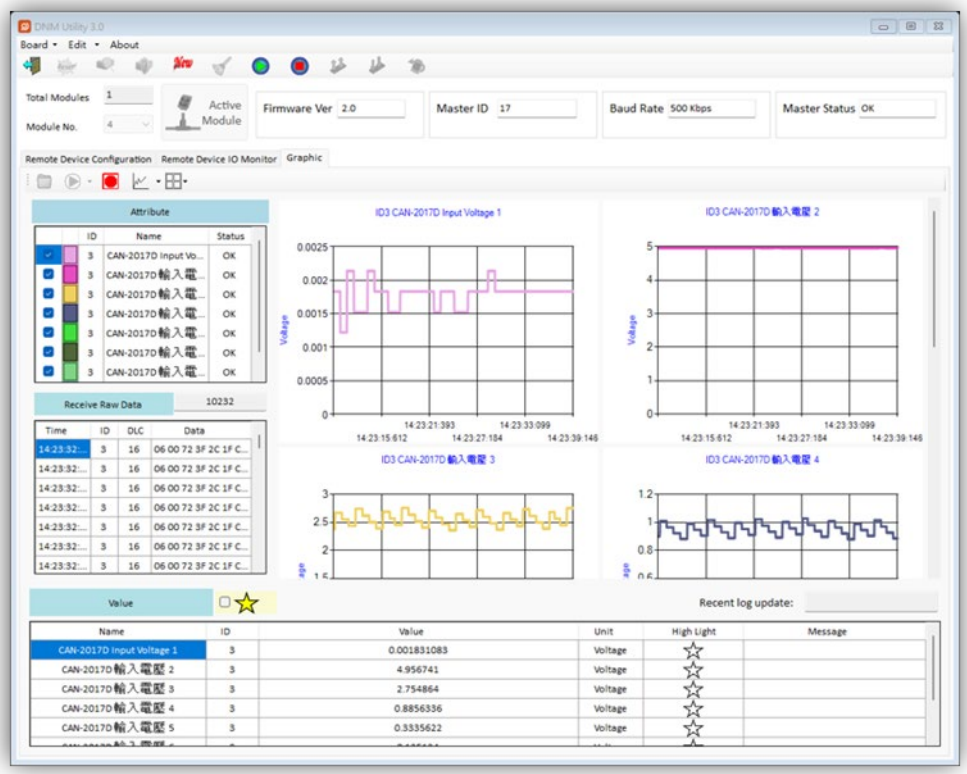


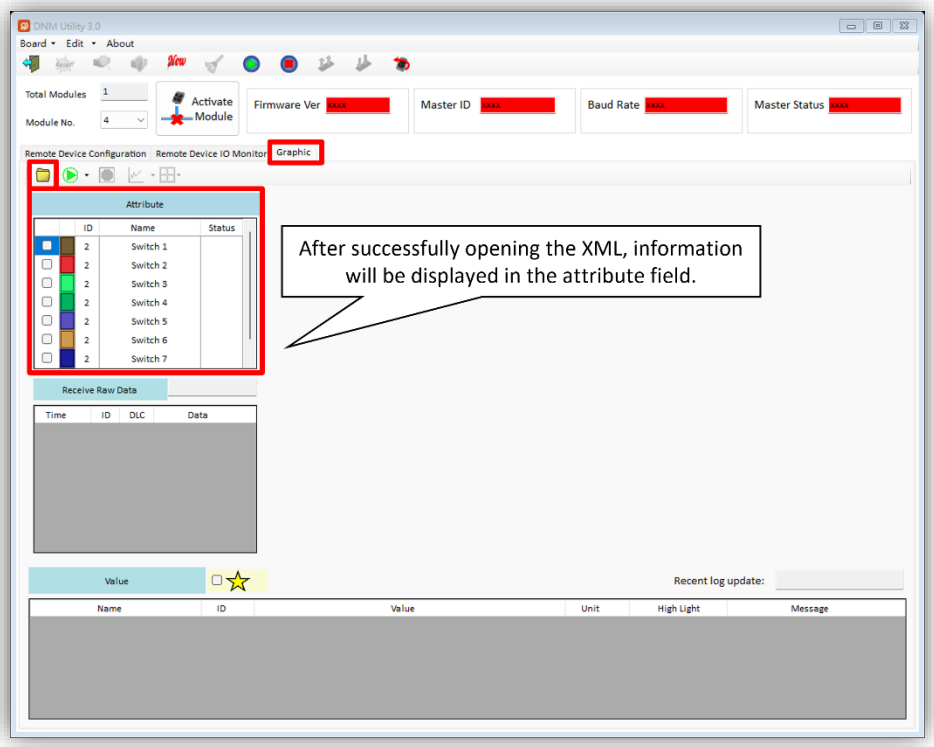
Figure: Monitoring Voltage Value Changes Using DNM_Utility

3.1 Common Features

3.1.1 How to import XML file

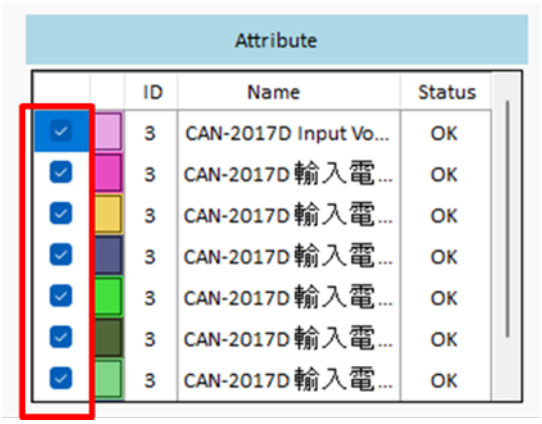
On the Graphic page, selecting “Open XML file” opens a file dialog. After selection, the system verifies whether the XML format is correct. The XML format will be introduced in Section 3.2.

After successful loading, the information from your input XML will be displayed in the attribute field.



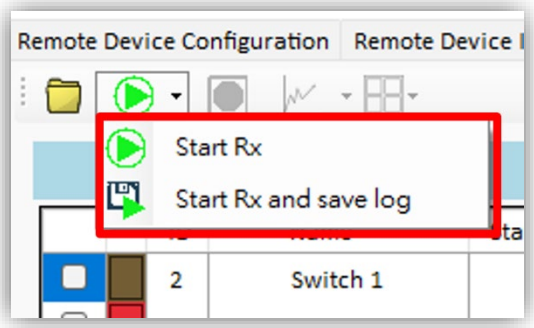
3.1.2 How to Start Rx Data Analysis After Importing XML

After successfully importing the XML, the Attribute form will display all the input data. Please first check the attributes that need to be analyzed.

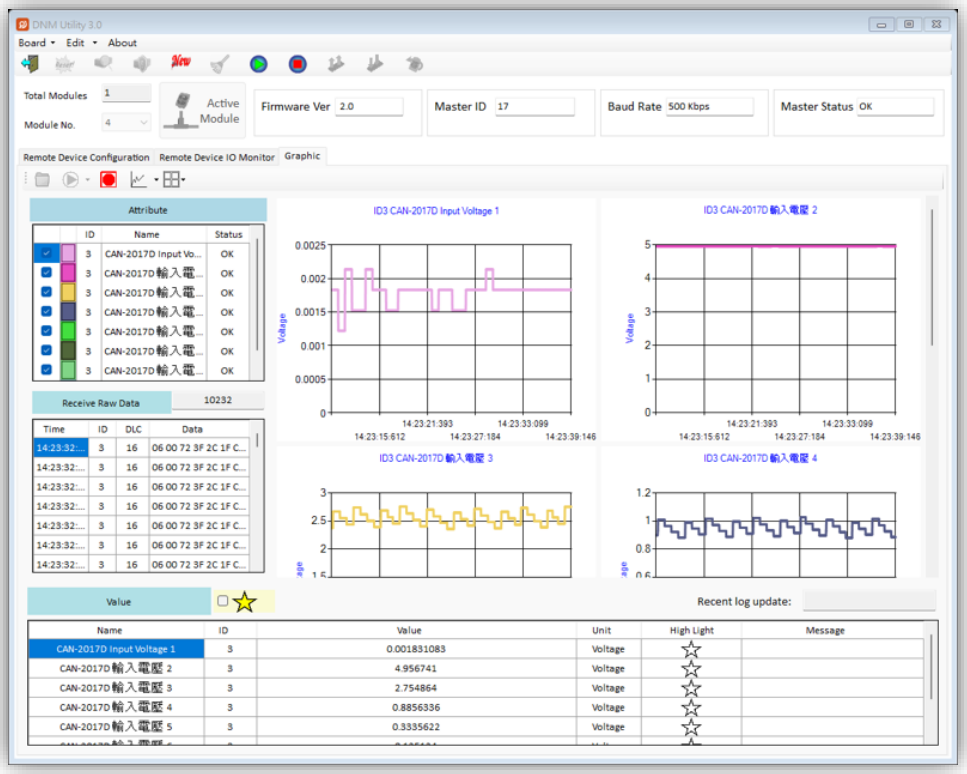


Next, select the data analysis mode from the toolbar. Choosing "Start Rx" will directly initiate a connection with the slave device, converting the received IO data into physical quantities and displaying them on the trend graph. Selecting "Start Rx and save log" will open a file dialog, allowing you to choose the storage location for the log file. After confirmation, it will start analyzing the IO data and save the results to the designated location.

The log file is in CSV format and includes the following fields: time, device ID, attribute name, converted physical quantity, and raw data.

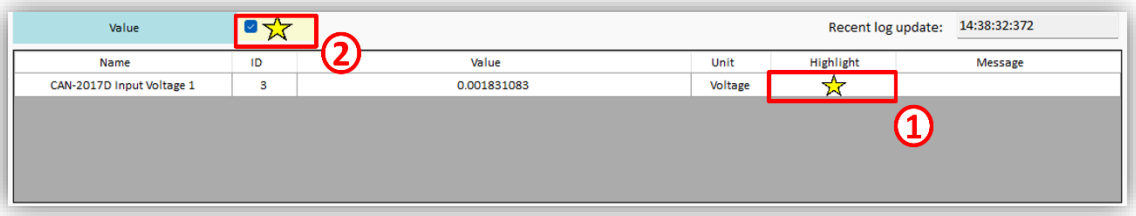


When "Start Rx" is in progress, the screen will display all the receive data and converted information.



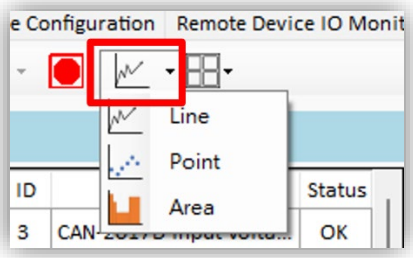
3.1.3 How to use highlight star

In the table within the “Value” field, you can click on the “highlight” star to toggle its state. Then, select the checkbox with a star next to the “Value” field to control whether the table displays the highlighted items.

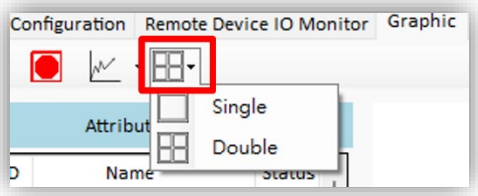


3.1.4 How to change chart display format

You can change the chart’s display format to line, point, or area, or adjust the chart’s scale on the screen.



and



On the Graphic page, DNM_Utility can assist in converting received slave information into physical quantities and displaying them as trend graphs on the screen. The user only needs to provide the slave information in XML format (refer to Section 3.1 for the XML content). After starting Rx, the user can observe the values and changes of the physical quantities converted from the slave return values. The following image shows the voltage values screen after converting the slave return values into physical quantities.

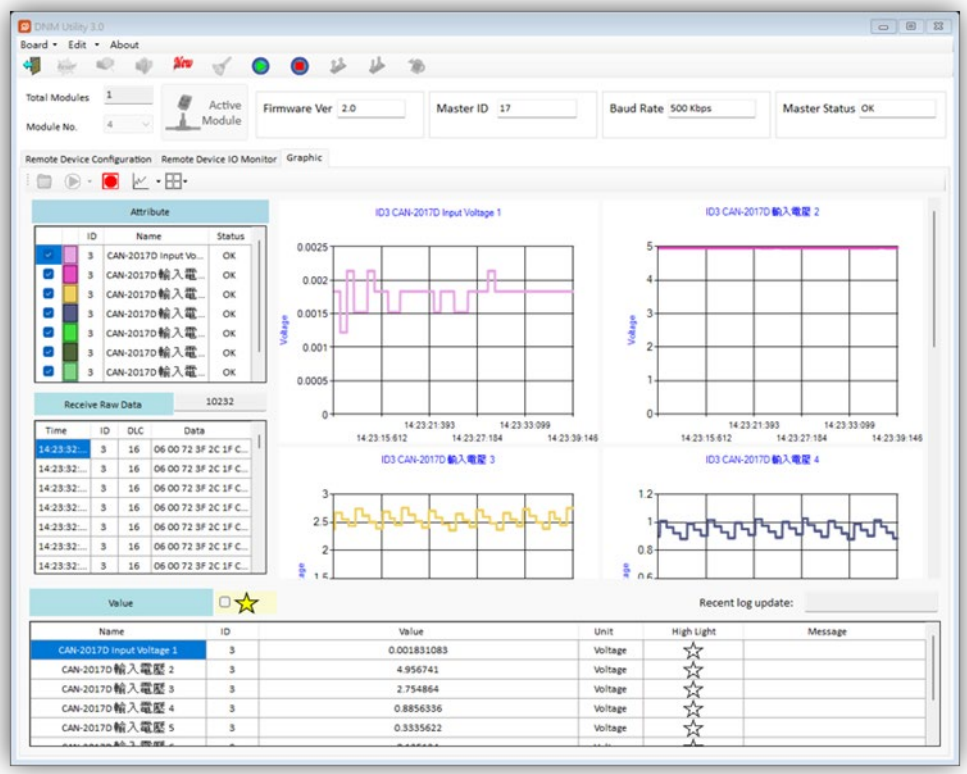


Figure 3: Monitoring Voltage Value Changes Using DNM_Utility

3.2XML Format Description

We use the XML format to load the data and computational methods necessary for analyzing raw data. All information must adhere to the specified rules; otherwise, successful loading of the information will not be possible.

The XML primarily consists of three parts. The first part, “Root,” sets up the basic structure of the XML. The second part, “Device Information Segment,” provides basic information about the slave device. Finally, the “Data Configuration Segment” details the data attributes. The XML flowchart is shown in figure 3.1.1:

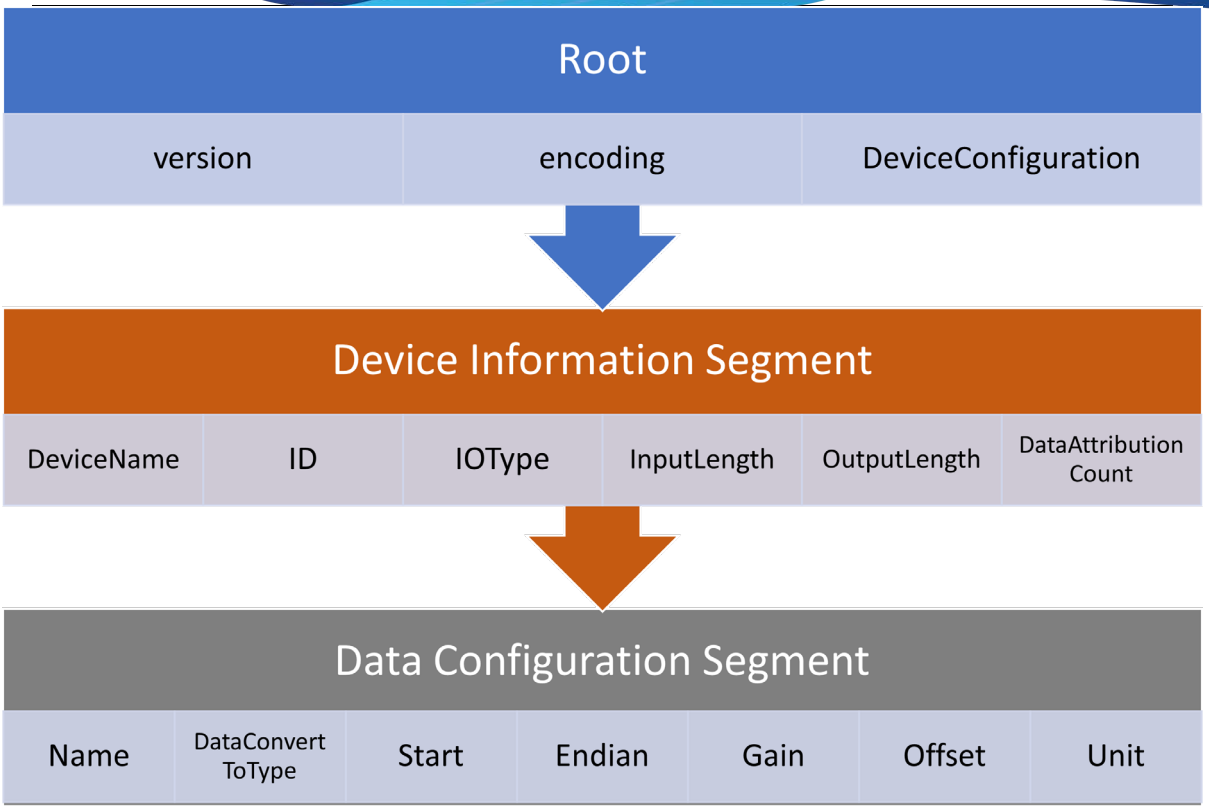


Figure: XML flowchart

3.2.1 Root(Header)

The “xml version”, “encoding”, and “DeviceConfiguration” **must** be included at the beginning of the document. Please don’t make any adjustments to the content inside.

```
<?xml version="1.0" encoding="utf-16"?>
<DeviceConfiguration  xmlns="http://tempuri.org/XMLSchema1.0.xsd">
```

Code:

<?xml version="1.0" encoding="utf-16"?>

<DeviceConfiguration xmlns="http://tempuri.org/XMLSchema1.0.xsd">

3.2.2 Device Information Segment: <Device></Device>

The content within <Device> includes the basic information of the device. Using CAN-2017D as an example. picture below shows the XML format after completing the device information for CAN-2017D.

```
<Device>
    <DeviceName>CAN-2017D</DeviceName>
    <ID>3</ID>
    <IOType>1</IOType>
    <!-- Poll=1, Bit=2, COS=3, Cyclic=4 -->
    <InputLength>16</InputLength>
    <OutputLength>1</OutputLength>
    <DataAttributionCount>8</DataAttributionCount>
</Device>
```

Figure: Device information code

3.2.2.1 Required and Optional Parameters

Table below provides all the required or optional parameters of <device> field.

Required	<ID>, <IOType>, <DataAttributionCount>
Optional	<DeviceName>, <InputLength>, <OutputLength>

Table : <Device> field used parameters

3.2.2.2 Accept Value for Device Information Field

Table below provide all the possible values that can be entered in the <Device> field.

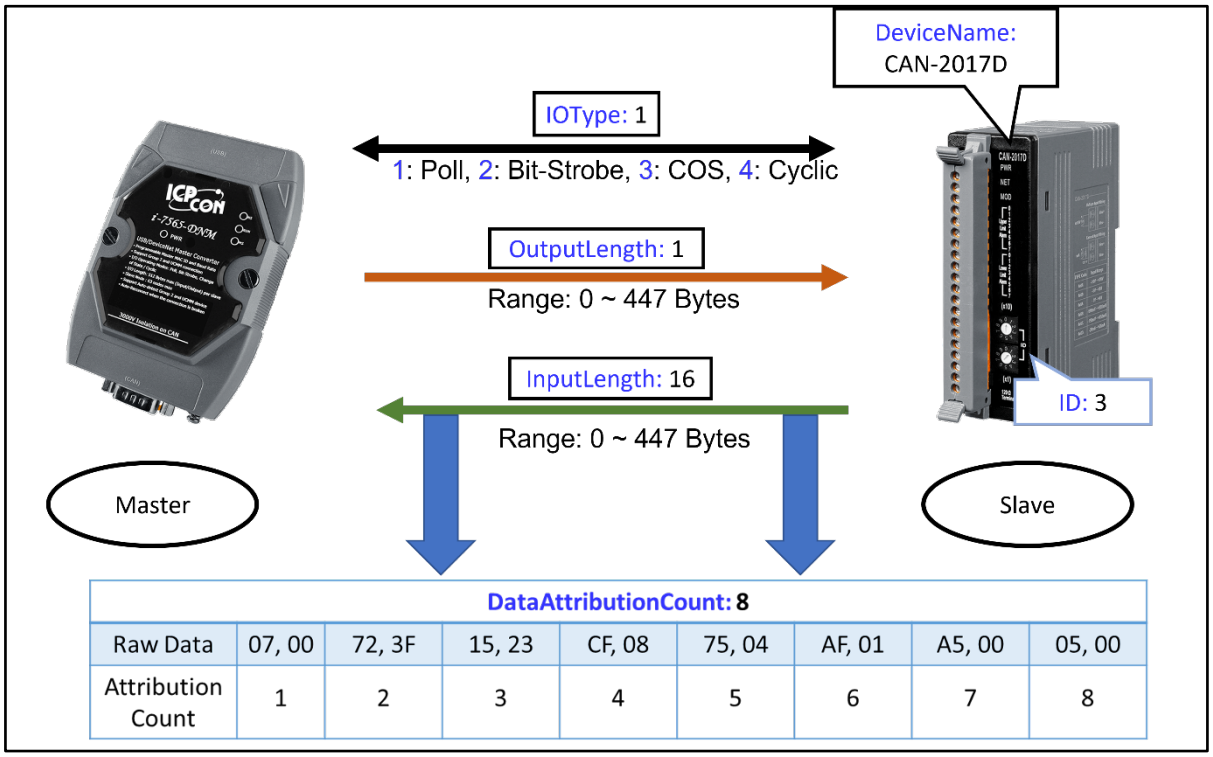
In IOType, 1 represents Poll connection, 2 represents Bit-Strobe connection, 3 represents COS connection, and 4 represents Cyclic connection.

Field Name	Accept Value
ID	0 ~ 63
IOType	1 、 2 、 3 、 4
InputLength	0 ~ 447
OutputLength	0 ~ 447
DataAttributionCount	0 ~ 10

Table : Accept value for <Device> field

Based on the field information above, picture below provides more details about these attributes. It should be noted that the “DataAttributionCount” is limited to a **maximum of 10**. Additionally, except for the “DeviceName” field, the correct numerical values should be entered in the other fields.

Notice! All fields must be in the specified order.



DataAttributionCount: 8								
Raw Data	07, 00	72, 3F	15, 23	CF, 08	75, 04	AF, 01	A5, 00	05, 00
Attribution Count	1	2	3	4	5	6	7	8

Figure : Device information diagram

3.2.3 Data Configuration Segment: <DataAttributes><Attribute> </Attribute></DataAttributes>

Inside <Attribute>, there is information about the starting position of the data, the data's length, and the data's conversion method. Using CAN-2017D as an example. Picture below shows the XML format after completing the attribute for CAN-2017D.

```
<DataAttributes>
    <Attribute>
        <Name> CAN-2017D Input Voltage 1</Name>
        <DataConvertToType>Int16</DataConvertToType>
        <Start>0</Start>
        <Endian>Little</Endian>
        <Gain>0.00030518043793392841</Gain>
        <Offset>0.0</Offset>
        <Unit>Voltage</Unit>
    </Attribute>
</DataAttributes>
```

Figure : Attribute segment code

3.2.3.1 Required and Optional Parameters

Table below provides all the required or optional parameters of attribute field.

If the optional fields are not specified, the default values are as follows:

1. Endian: Big.
2. Gain: 1.0.
3. Offset: 0.

Required	<Name>, <DataConvertToType>, <Start>
Optional	<Endian>, <Gain>, <Offset>, <Unit>

Table : Attribute field used parameters

3.2.3.2 Accept Value for Data Attribute Field

Table below provide all the possible values that can be entered in the "DataConvertToType" and "Endian" fields.

DataConvertToType	Bool, SignedByte, UnsignedByte, Int16, UInt16, Int32, UInt32, Float
Endian	Big, Little

Table : DataConvertToType and Endian field accept value

3.2.3.3 Attribute Field While DataConvertToType is Set to Int16

The "DataConvertToType" is determined by the type of the IO input data, and the value of "Start" is dependent on the byte offset from the IO raw data. Picture below illustrates the relationship between the "Start" field of each attribute and the byte offset when "DataConvertToType" is set to Int16.

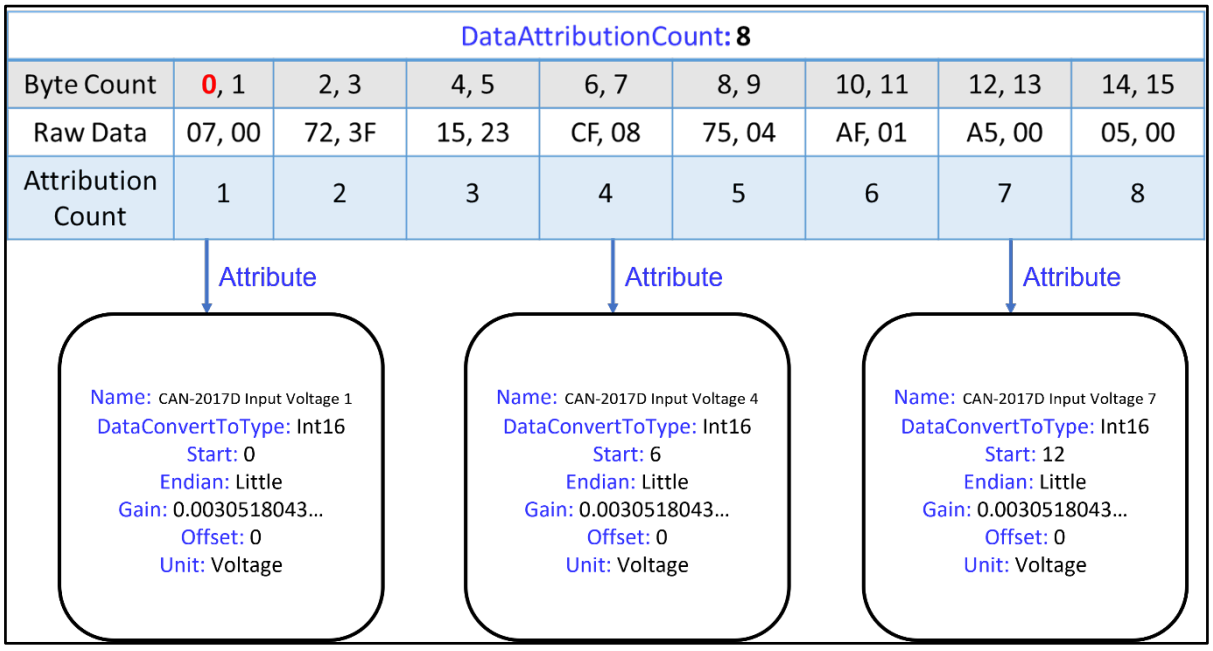


Figure : CAN-2017D data attributes

3.2.3.4 Attribute Field While DataConvertToType is Set to Bool

Only the "Bool" type accepts a floating-point value for the start parameter. For other types, please enter an integer. Picture below shows the input format for "Start" when "DataConvertToType" is set to bool.

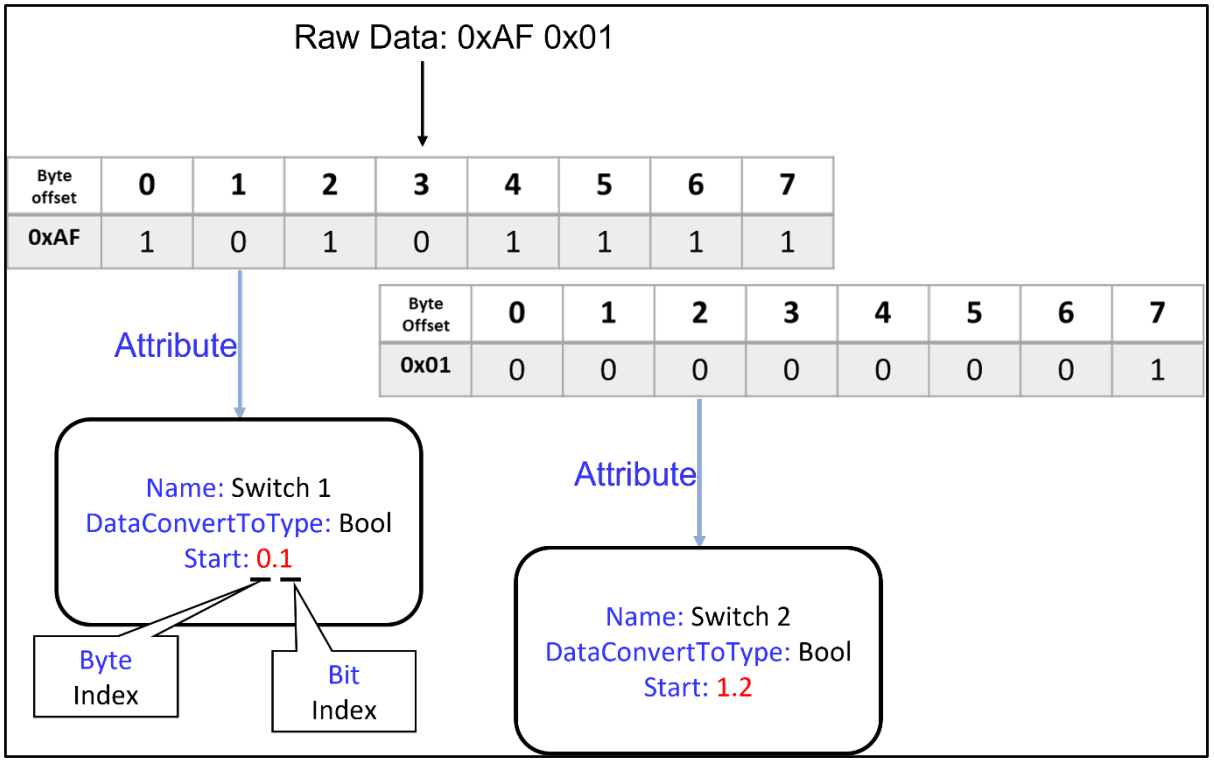


Figure: While DataConvertToType is bool data attributes

Notice! Byte Index and Bit Index **start from 0.**

3.2.3.5 Method for Calculating Gain and Offset

Gain and Offset will determine the output value. The calculation method is shown as below. The "Input Data Value" represents the numerical data obtained after converting the raw data. The default value of gain and offset is set to 1 and 0.

$$Physical\_value = Gain * Input\ data\ value + Offset$$

Using CAN-2017D as an example. Picture below shows the format of the complete XML.

```
<?xml version="1.0" encoding="utf-16"?>
<DeviceConfiguration   xmlns="http://tempuri.org/XMLSchema1.0.xsd">
    <Device>
        <DeviceName>CAN-2017D</DeviceName>
        <ID>3</ID>
        <IOType>1</IOType>
        <InputLength>16</InputLength>
        <OutputLength>1</OutputLength>
        <DataAttributionCount>8</DataAttributionCount>
    </Device>
    <DataAttributes>
        <Attribute>
            <Name> CAN-2017D Input Voltage 1</Name>
            <DataConvertToType>Int16</DataConvertToType>
            <Start>0</Start>
            <Endian>Little</Endian>
            <Gain>0.00030518043793392841</Gain>
            <Offset>0.0</Offset>
            <Unit>Voltage</Unit>
        </Attribute>
    </DataAttributes>
</DeviceConfiguration>
```

Figure : Complete XML for CAN-2017D

Notice! There will be only one <DataAttributes>, and <Attribute> should be inside <DataAttributes> and can have multiple instances (the number of <Attribute> instances should be equal to the value of <DataAttributionCount>).

3.2.4 XML Code Example

We provide XML example as following path:

C:\ICPDAS\DNM_Utility\Manual\XML Example

The “XML Example” folder contains two examples:

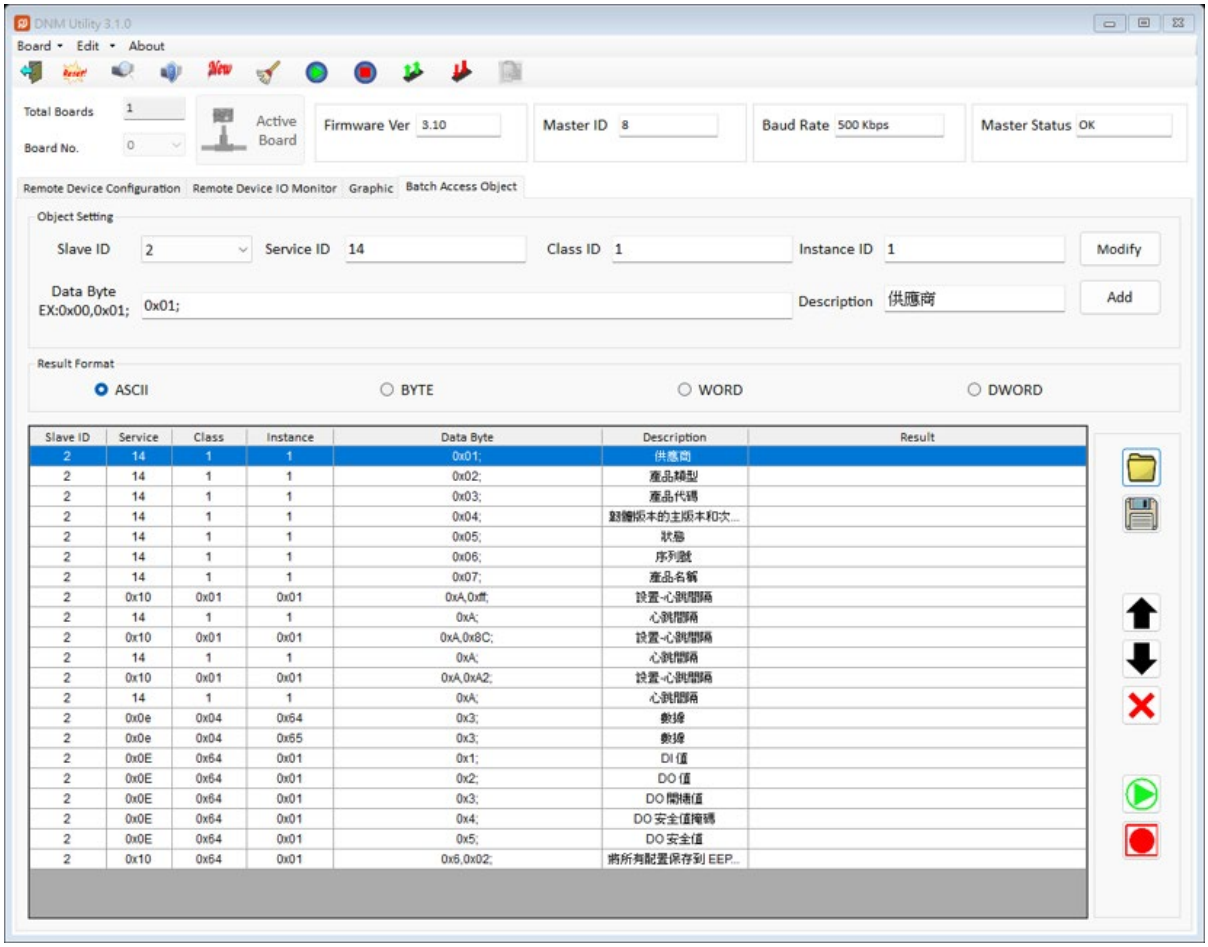
1. CAN-2017D.xml
2. XML_module.xml

In the CAN-2017D.xml file, the complete XML code for the slave device CAN-2017D is demonstrated. This allows users to directly use the code for measuring voltages of ±10V.

In XML_module.xml, the primary emphasis is on demonstrating the calculation methods between various “DataConvertToType” and “Start” values. When designing an XML, users can easily calculate the “Start” position based on the chosen “DataConvertToType”.

4. Advanced Feature: Batch Access Object

The Batch Access Object is an automated feature designed to help users handle multiple Explicit Messages more efficiently, particularly in situations that require the repeated transmission of similar messages. By automating the process of handling all input messages at once, as well as saving and importing data, it significantly reduces repetitive tasks and improves efficiency. Users can save the input data to an external file (CSV), allowing them to quickly start tasks by simply importing the data without re-entering it. This is particularly suitable for applications like manufacturing automation and equipment configuration, where similar operations need to be performed repeatedly. It greatly simplifies workflows and enhances productivity.

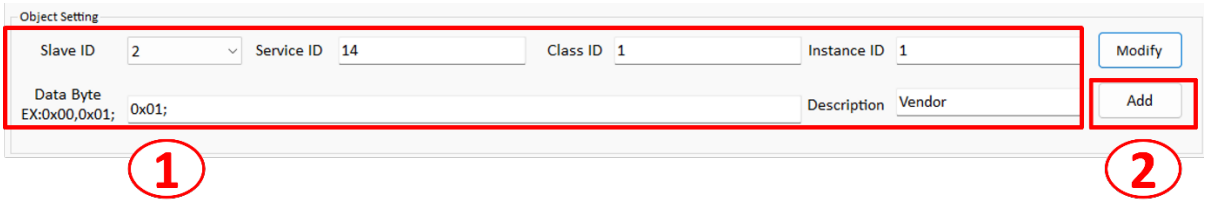


4.1. Common Features

This section will introduce the features and operations on the Batch Access Object page, including how to input object data, modify object data, and start batch retrieval of object data.

4.1.1. Input Object Data

In the "Object Setting" section, users can input the Explicit data intended for transmission. After entering the data, press the "Add" button to add the data to the scheduled transmission table.



1 **Slave ID**: The MAC ID of the remote slave device (0-63).

2 **Service ID**: The Service ID of the remote slave device.

3 **Class ID**: The Class ID of the remote slave device.

4 **Instance ID**: The Instance ID of the remote slave device.

5 **Data Byte**: Input command field. Please note the input format (e.g., "0x0A,0x91;").

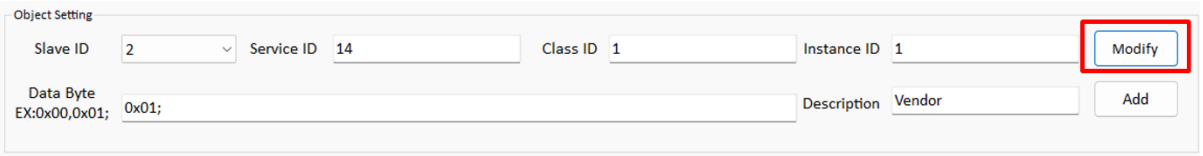
6 **Delay**: Waiting time after the command is completed. (Unit: milliseconds)

7 **Description**: Object name, can be left blank.

8 **Add**: Save the object to the form.

4.1.2. Modify Object Data

When selecting data from the table below, the corresponding data will be automatically populated in the "Object Setting" section. After modifying the data, click the "Modify" button on the right to update the changes to the form.



4.1.3. Delete Object Data

When selecting data from the table below, click the  button to remove the selected data from the form.

4.1.4. Adjust Data Order

When selecting data from the table below, click the  or  button to adjust the order of the selected data.

 or 

4.1.5. Change Display Format of Received Data

In the "Result Format" section, users can adjust the format in which the received data is displayed.

4.1.6. Save Object Data

Click the  button to save all the data in the form to a CSV file.

4.1.7. Import Object Data

Click the  button to select an external CSV file and import it into the form.

4.1.8. Start Sending Object Data

Click the  button to begin sending the object data in the form. During this process, no other operations can be performed. To stop, click the  button.

The received data will be displayed in the "Result" field of the form. The image below shows an example of retrieving the module name.

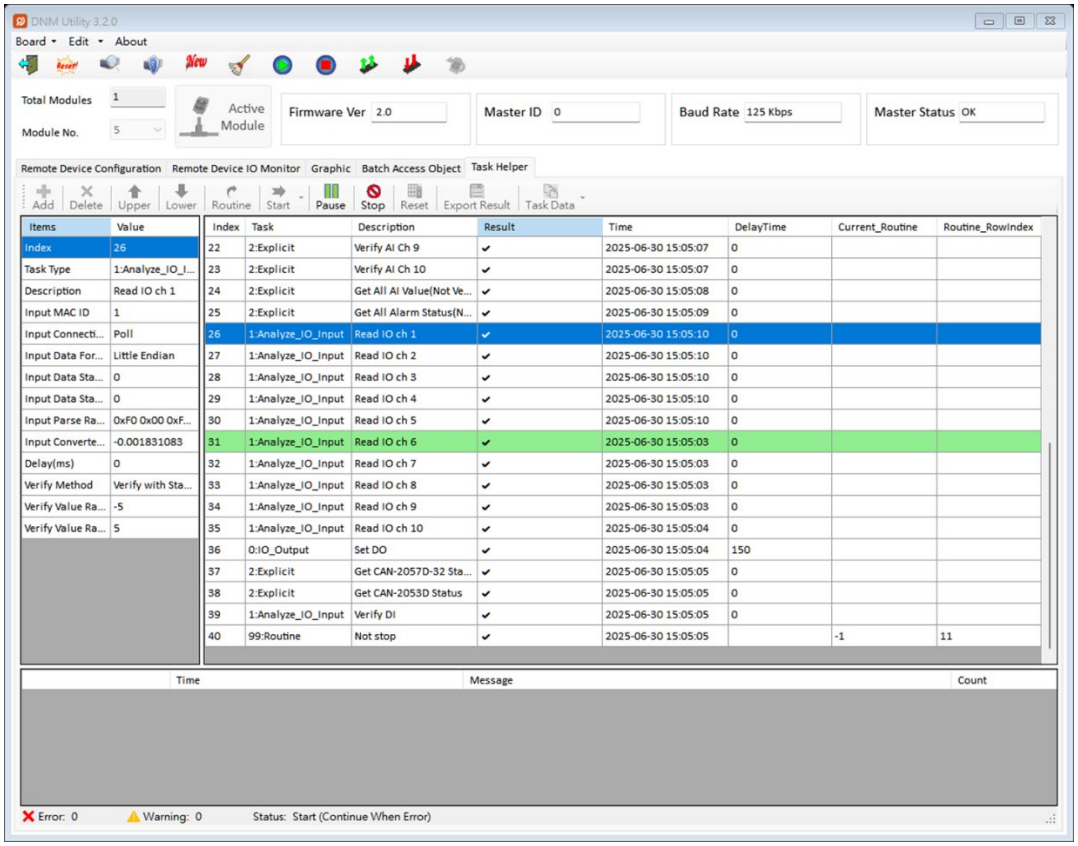
Slave ID	Service	Class	Instance	Data Byte	Description	Result
2	14	1	1	0x07;	Module Name	CAN-2054D

4.1.9. Stop Sending Object Data

Click the  button to terminate all ongoing batch retrieval tasks for the objects and restore the availability of other functionalities.

5. Advanced Feature: Task Helper

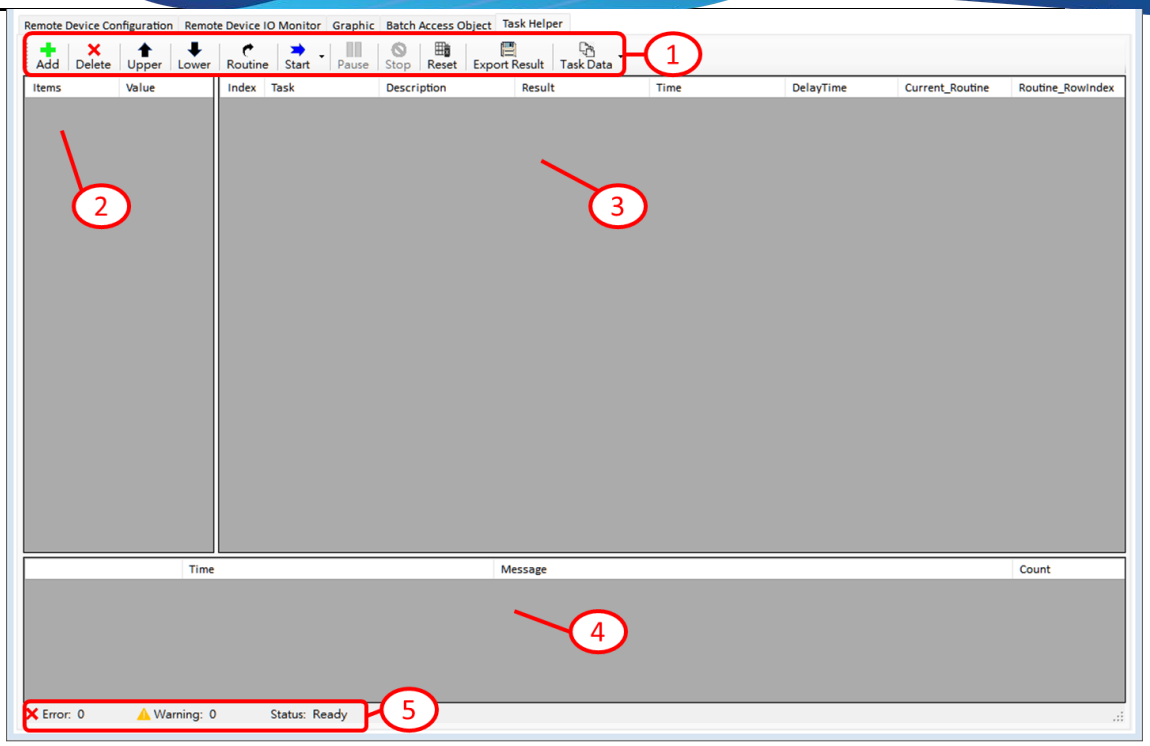
Task Helper is an automated testing tool built into DNM_Utility, offering three main task types: sending I/O data, reading and analyzing I/O data, and sending and analyzing Explicit data. With an intuitive UI, it allows you to quickly configure task parameters. Once the test is started, the system automatically records complete execution information to assist with real-time tracking, debugging, and result comparison. This enables you to verify device operation in the shortest time, significantly improving testing efficiency and accuracy.



The following sections will introduce the features and operations on the Task Helper page, including how to set up task data, modify task data, export and import task data, and export task logs.

5.1 Interface Overview

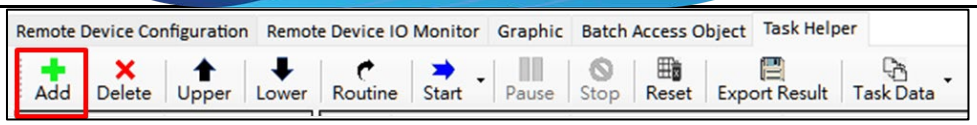
Task Helper consists of five main working sections: Function Menu, Task Detail Table, Added Task Table, Error Information Table, and Status Bar. This section provides a brief introduction to each functional area.



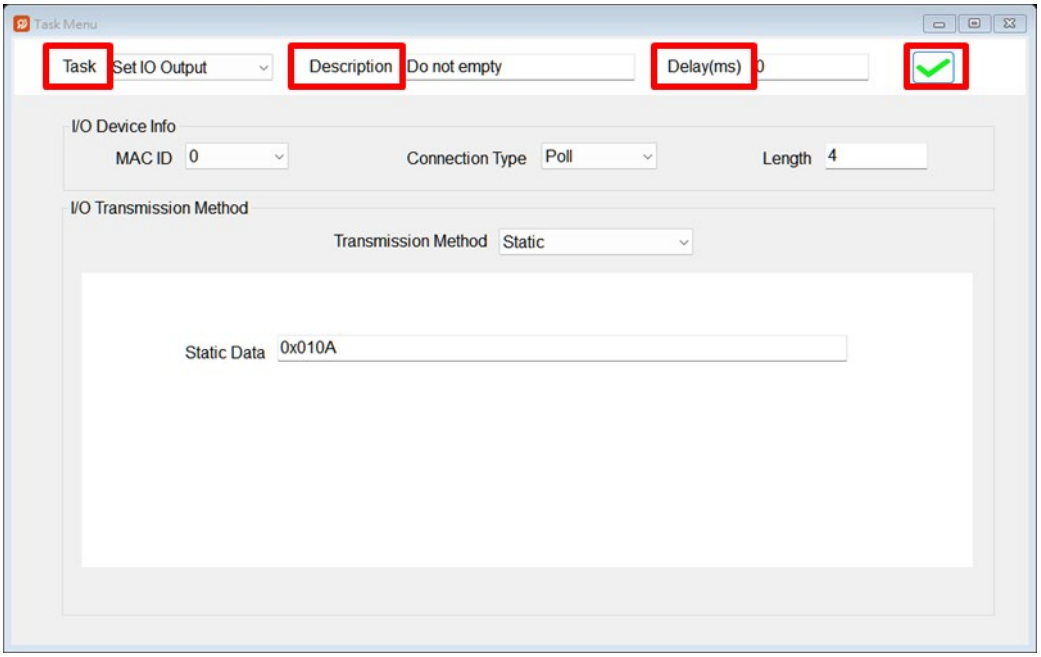
1. Function Menu: Provides features such as adding tasks, deleting tasks, exporting/importing tasks, exporting test records, and starting/stopping tests. Detailed explanations of these functions will be introduced in later sections.
2. Task Detail Table: Displays detailed information of the selected task from the Added Task Table on the right, as well as real-time data processed during task execution.
3. Added Task Table: Shows the main information of each task once added.
   - You can rearrange task order, remove tasks, and modify task parameters (by double-clicking).
   - When a task is selected, its detailed information will be displayed in the Task Detail Table on the left.
   - During task execution, the current execution phase will also be shown here.
4. Error Information Table: Displays all errors that occur during the execution phase of tasks.
5. Status Bar: Shows the total number of issues encountered and the current status of the Task Helper.

5.2 Add New Task

1 Click Add in the menu.



2 The Task Menu window will open.

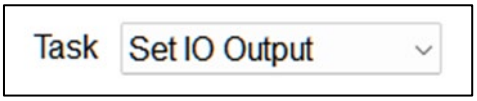


- Task: Select the task type: Send I/O Data, Read and Analyze I/O Data, or Send and Analyze Explicit Data.
- Description: Task description. You can define a custom description for the task to make it easier to review later.
- Delay: Delay time after the task is completed. Unit: milliseconds.
-  : Click to add the task after entering all information.

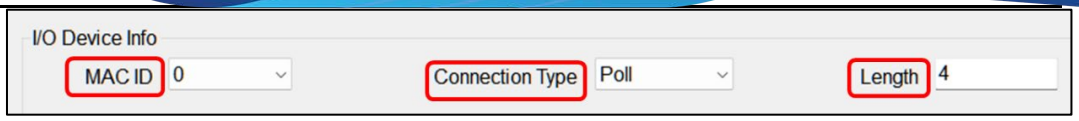
5.2.1 Task Type – Send I/O Data

There are three modes available for sending I/O data: Fixed Data, Sequential Data, and Bitwise Sequential Data. The following section provides an explanation.

1 Select “Set IO Output” from the Task dropdown list.



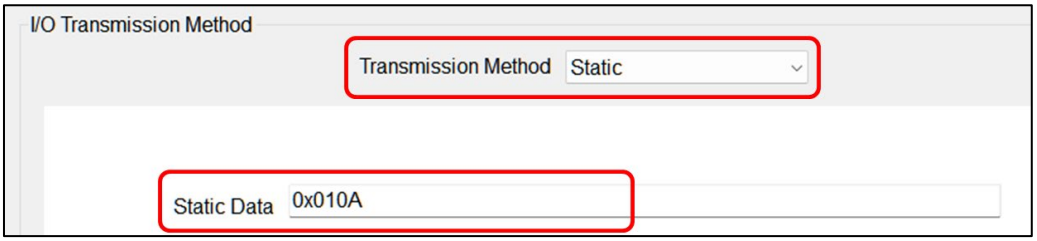
2 Enter the Slave I/O information.



- MAC ID: The ID used by the slave module.
- Connection Type: The communication mode being used.
- Length: The length of the I/O data to be sent.

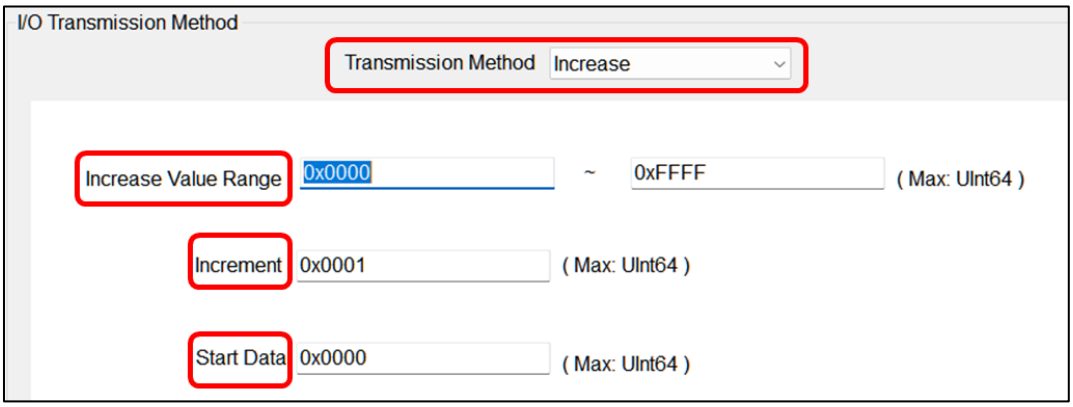
3 Select the I/O transmission mode. There are three available modes: Static Data, Sequential Data, and Bitwise Sequential Data.

Transmission Method: Select Static.



- Static Data: The I/O data to be sent. Input format: 0x____ (e.g., 0x010A08A5).

Transmission Method: Select Increase.

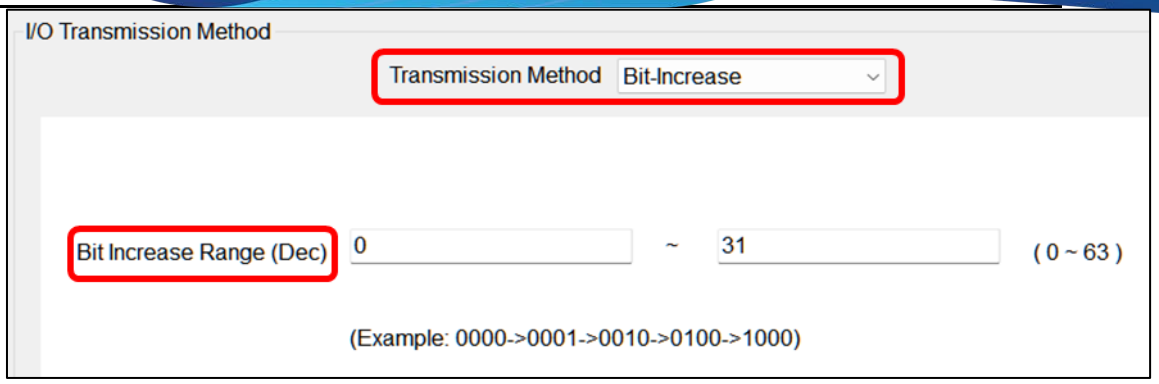


- Increase Value Range: The data value range.
- Increment: The increment step of the data.
- Start Data: The starting data value.

Tip: The sequential data function requires using it with a Routine (loop) setting.

Transmission Method: Select Bit-Increase.

This mode is suitable for testing digital output modules that use bitwise units.



- Bit Increase Range: The bit range for sequential increase. For example, 0~3 means bits 0 to 3 will be set to 1 in sequence, while the other bits remain 0.

Example: 0000 → 0001 → 0010 → 0100 → 1000 …

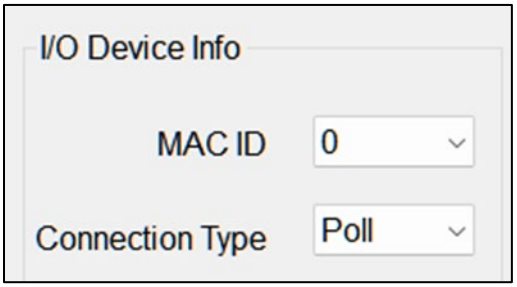
Tip: The Bitwise Sequential function needs to be used together with a Routine (loop).

5.2.2 Task Type – Read and Analyze I/O Data

Read and Analyze I/O Data provides practical data analysis features. It supports converting I/O data into physical values for comparison, and also offers the ability to compare output I/O data from other slave modules.

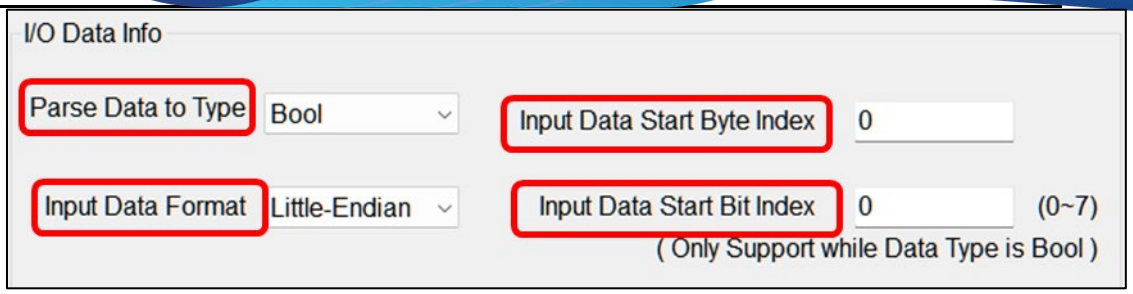
1 From the Task dropdown list, select “Analyze IO Input”.

2 Enter the Slave I/O information.



- MAC ID: The ID used by the slave module.
- Connection Type: The connection type used by the slave module.

3 Select the I/O data conversion method.

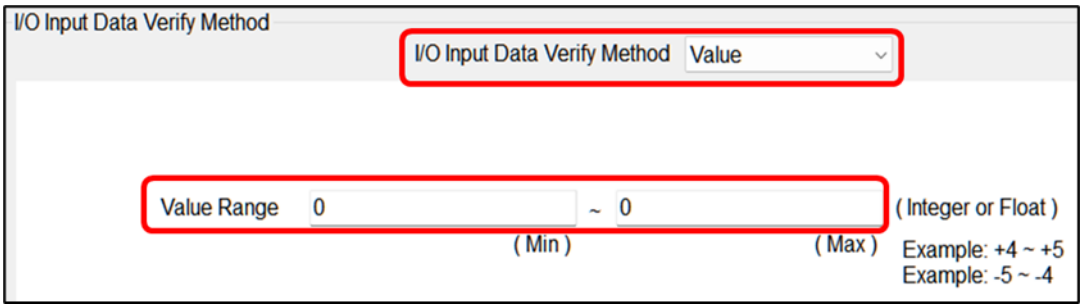


- Parse Data to Type: Select the method for converting I/O data into physical values. Currently supported types: Bool, SByte, Byte, Int16, UInt16, Int32, UInt32, Float.
- Input Data Format: Specify whether the input data is in Little-Endian or Big-Endian format.
- Input Data Start Byte Index: Starting position of the data (in bytes).
- Input Data Start Bit: Starting position of the data (in bits).

  Note: The Input Data Start Bit field is only available when “Parse Data to Type” is set to Bool.

4 Select the I/O data verification method: There are two modes — Physical Value (Value) or I/O Output Data from another slave.

Verify Method: Select Value

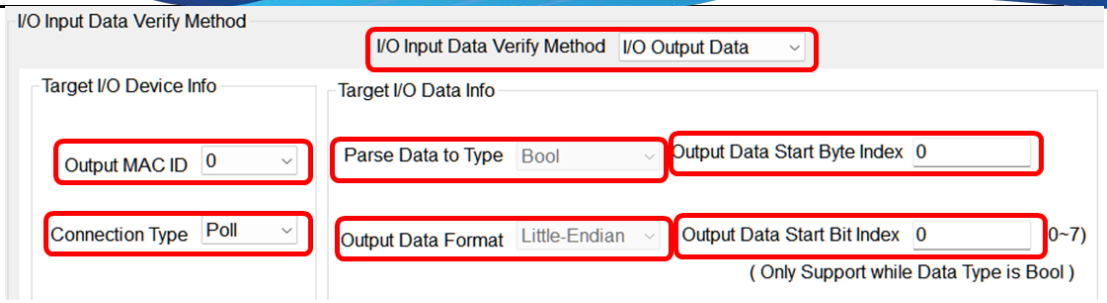


- Value Range: Verifies whether the received I/O data falls within the expected valid range.

Verify Method: Select I/O Output Data (from another slave)

This method uses the API ReadBackOutputData() to compare the received data with the output data of another slave.

For example: controlling a digital output module → reading from a digital input module, and verifying whether the data matches.



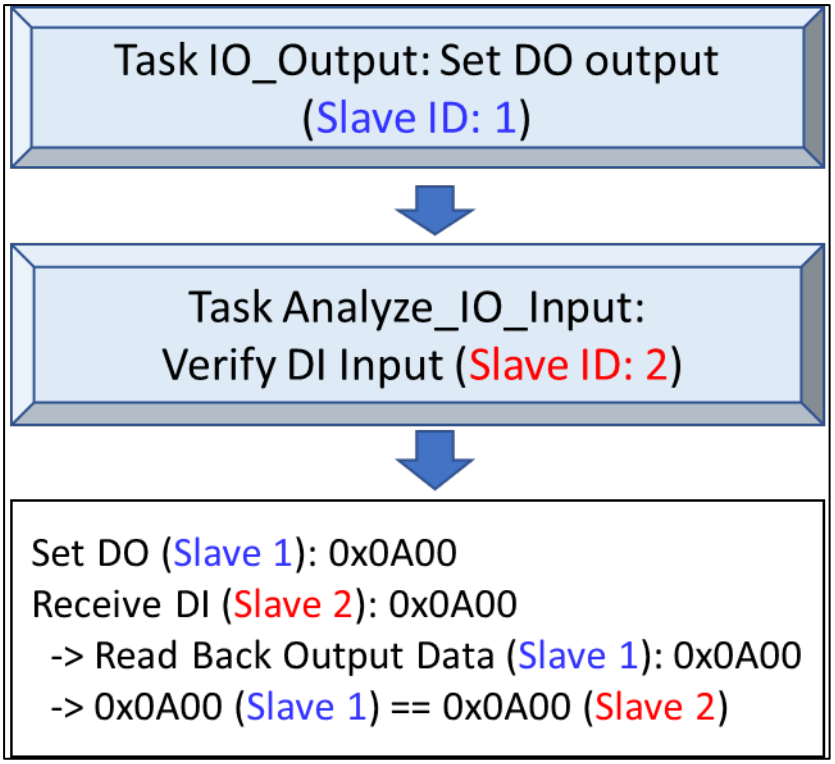
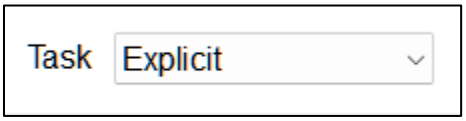


Figure: Flowchart of comparing input and output I/O values

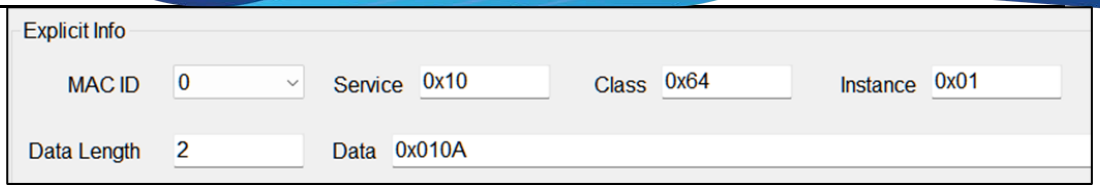
5.2.3 Task Type – Send and Analyze Explicit Data

The function for sending and analyzing Explicit data helps you configure parameters using Explicit messages and provides the ability to analyze whether the returned Explicit data meets expectations. With this function, DNM_Utility assists you in completing all parameter configurations or verifying if the parameters meet the expected values.

1 From the Task dropdown list, select “Explicit”.



2 Enter the Explicit message details.

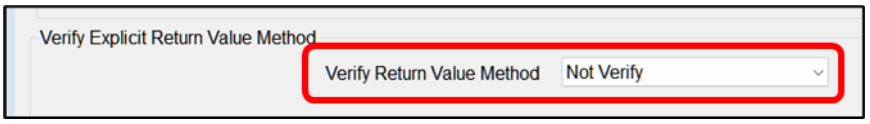


- MAC ID: Slave ID.
- Service: Explicit Service ID.
- Class: Explicit Class ID.
- Instance: Explicit Instance ID.
- Data Length: Length of the data.
- Data: Explicit data. (Format: Attribute ID + Data or Data only)

Explicit Data Analysis Method: The following conversion and analysis methods are available: No Verification, ASCII (string), Bool, SByte, Byte, Int16, UInt16, Int32, UInt32.

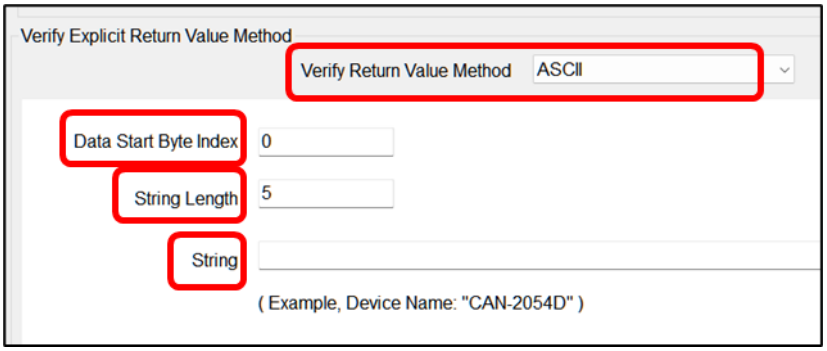
Verify Method: Not Verify

If the response data is to be ignored or the command does not carry any data (e.g., Set Attribute Single), you may select Not Verify.



Verify Method: ASCII

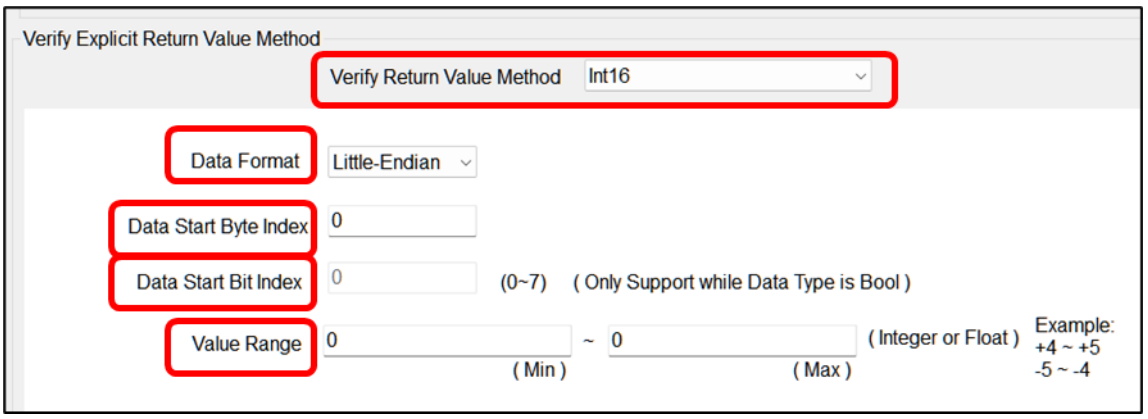
Used for parsing string format data.



- Data Start Byte Index: Starting position of the string.
- String Length: Length of the string.
- String: Expected content to be received.

Verify Method: Convert the data to Bool, SByte, Byte, Int16, UInt16, Int32,

or UInt32 for comparison.



- Data Format: Format of the received data, either Little-Endian or Big-Endian.
- Data Start Byte Index: Starting position of the data (unit: Byte).
- Data Start Bit Index: Starting bit position (unit: Bit).

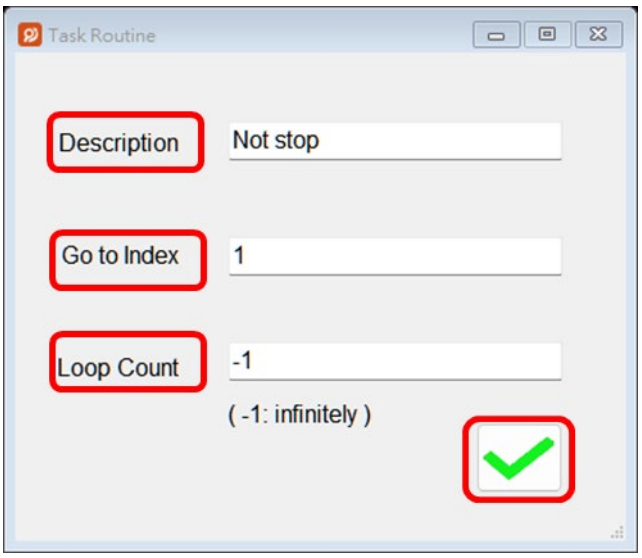
  Note: Start Bit Index can only be set when the Verify Method is set to "Bool".
- Value Range: The expected range of the received data.

5.3 Task Loop

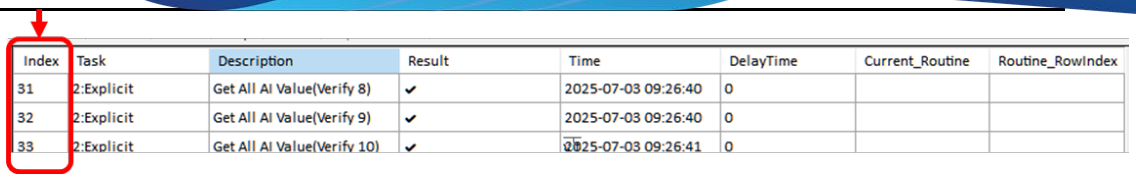
The Task Loop function helps you perform long-duration testing. Operation instructions are as follows:

Click 

to open the Task Loop settings window.



- Description: Task description. You can enter a custom description.
- Go to Index: The starting index for the loop. Refer to the index number in the Added Task Table.



Index	Task	Description	Result	Time	DelayTime	Current_Routine	Routine_RowIndex
31	2:Explicit	Get All AI Value(Verify 8)	✔	2025-07-03 09:26:40	0		
32	2:Explicit	Get All AI Value(Verify 9)	✔	2025-07-03 09:26:40	0		
33	2:Explicit	Get All AI Value(Verify 10)	✔	2025-07-03 09:26:41	0		

- Loop Count: Number of loop iterations. Enter -1 for infinite loops.
-  : Click to add the loop task after entering the data.

5.4 Delete Task

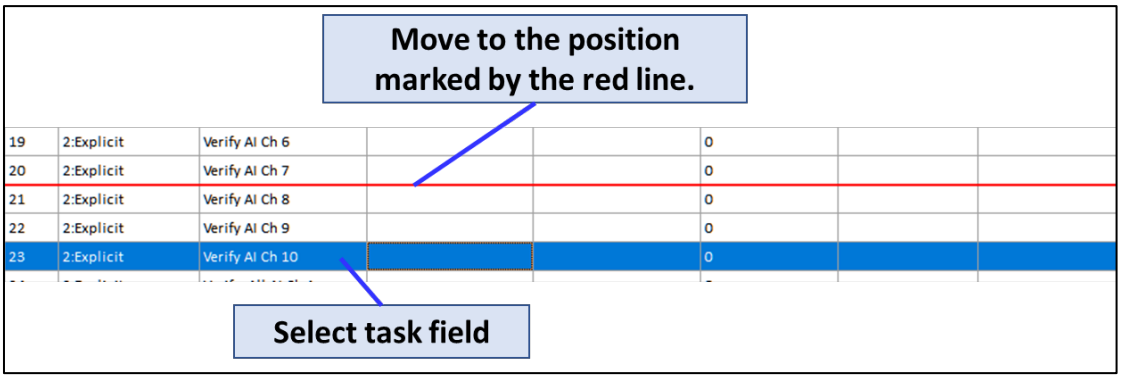
: Click to delete the currently selected task.

5.5 Adjust Task Order

There are two ways to adjust the order in the Added Task Table.

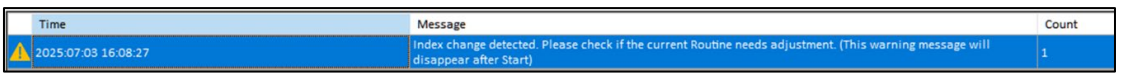
1. Use 

.
2. Click and Drag: Press and hold the mouse button to drag the task. The insertion point will be indicated by a red line.



Note: If your task schedule includes a Routine (Task Loop), a reminder message will appear when you adjust the task order, prompting you to check whether the “Index” attribute of the loop needs to be updated.

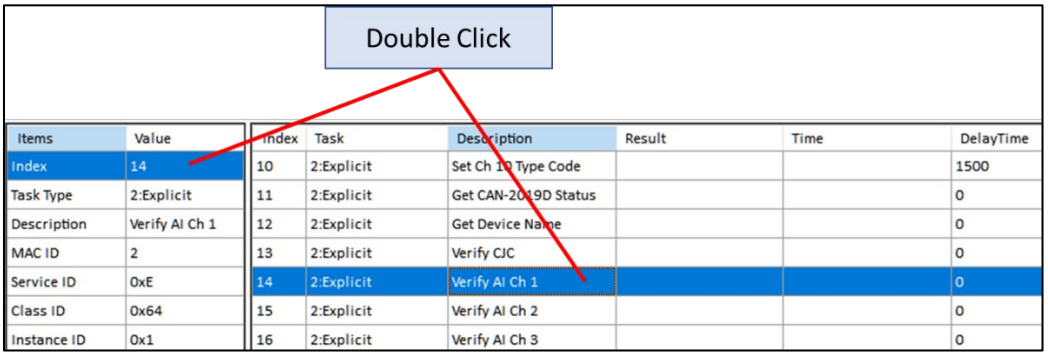
This reminder will be automatically removed after the task starts and will not be recorded in the task output log.



	Time	Message	Count
⚠	2025:07:03 16:08:27	Index change detected. Please check if the current Routine needs adjustment. (This warning message will disappear after Start)	1

5.6 Edit Task

Double-click on either the Added Task Table or the Task Detail Table to enter edit mode for the selected task.



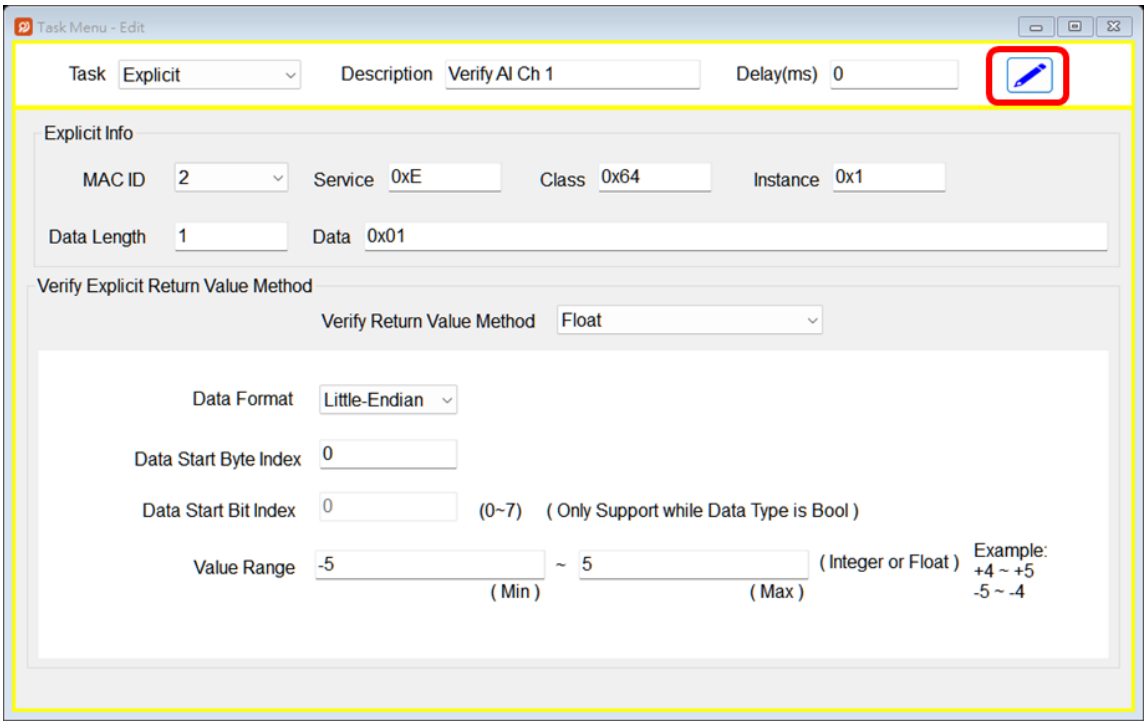


Figure: Task Editing Interface

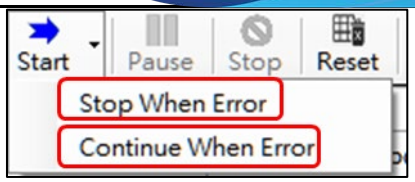
After editing, click  to apply the changes.

5.7 Start Executing the Task Schedule

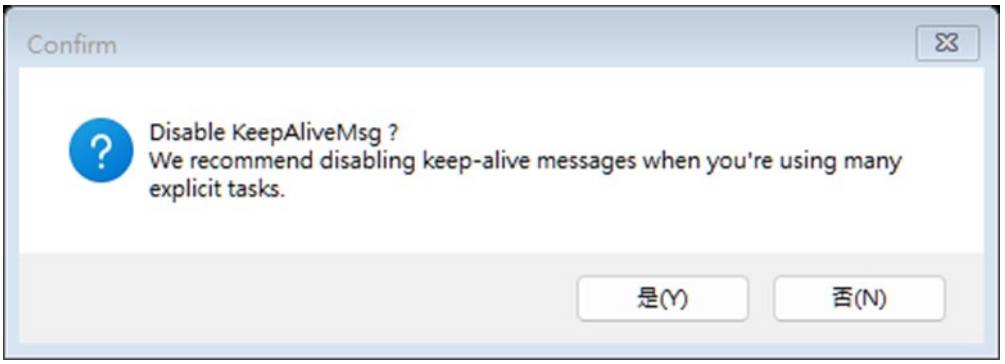
There are two modes available for executing task schedules:

1 Stop When Error: Stops task execution when an error (Error / Warning) occurs.

2 Continue When Error: Continues executing the tasks even if an error occurs.

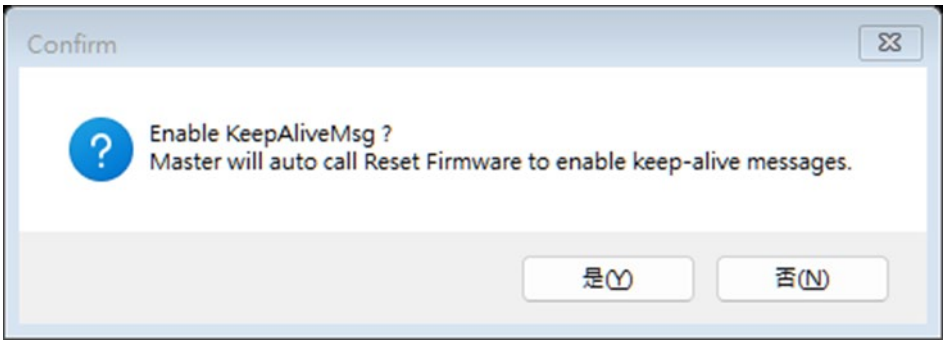


*Note: If your task involves an Explicit-type operation, DNM_Utility will ask whether to disable the **Keep Alive Message** function. The Keep Alive Message is mainly intended for slave devices that require periodic Explicit messages to maintain the connection.



If your slave device does **not** require periodic Explicit messages to maintain connectivity, we strongly recommend disabling the Keep Alive Message to avoid interference with the Explicit function during the task.

After the task is completed or terminated, if you previously disabled the Keep Alive Message, DNM_Utility will ask whether to restart the **Master** to re-enable the Keep Alive Message function. If you choose "Yes," it will trigger a **Reset Firmware** and reboot the master device. You will then need to call Start Device again to re-establish the connection with the slave device.



5.8 Pause/Resume Task Execution

When the task schedule is running, the 

becomes available. Clicking it will pause the execution before the next task.

While paused, you can edit task information.

Click 

to resume task execution.

5.9 Stop Task Execution

After selecting 

, the task schedule will be terminated.

5.10 Reset All Task Schedules

Click 

to clear all task data.

Please make sure any important information has been saved before proceeding.

5.11 Export Task Execution Log

Click 

to export the task log to the program directory (DNM_Utility).

The task log contains detailed information for each execution phase and can be used for further data analysis or as a product testing report.

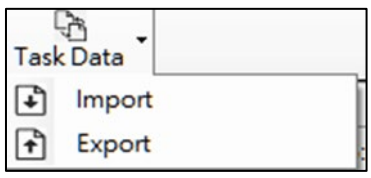
Log file name format: Date_Time_TaskProgressExport.txt

Note: A temporary log file is automatically created when the task starts. The previous temporary log will be deleted and replaced with a new one each time a task is started.

5.12 Import/Export Task Schedule

From the Task Data dropdown menu, you can select Import or Export task schedules.

Exported files are saved in CSV format.



5.13 Task Schedule Execution Field Description

During task execution, the task currently being executed will be highlighted in green.

Index	Task	Description	Result	Time	DelayTime	Current_Routine	Routine_RowIndex
26	2:Explicit	Verify_All AI Ch 3	✔	2025-07-02 09:28:00	0		
27	2:Explicit	Verify_All AI Ch 4	✔	2025-07-02 09:28:00	0		
28	2:Explicit	Verify_All AI Ch 5	✔	2025-07-02 09:28:00	0		
29	2:Explicit	Verify_All AI Ch 6	✔	2025-07-02 09:28:01	0		
30	2:Explicit	Verify_All AI Ch 7	✔	2025-07-02 09:28:01	0		
31	2:Explicit	Verify_All AI Ch 8	✔	2025-07-02 09:28:01	0		
32	2:Explicit	Verify_All AI Ch 9	✔	2025-07-02 09:28:01	0		
33	2:Explicit	Verify_All AI Ch 10	✔	2025-07-02 09:28:01	0		
34	1:Analyze_IO_Input	Verify IO Ch 1	✔	2025-07-02 09:27:56	0		
35	1:Analyze_IO_Input	Verify IO Ch 2	✔	2025-07-02 09:27:56	0		

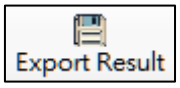
Figure: The green-highlighted task indicates the task currently being executed.

You can view the real-time status of the task in the table on the left.

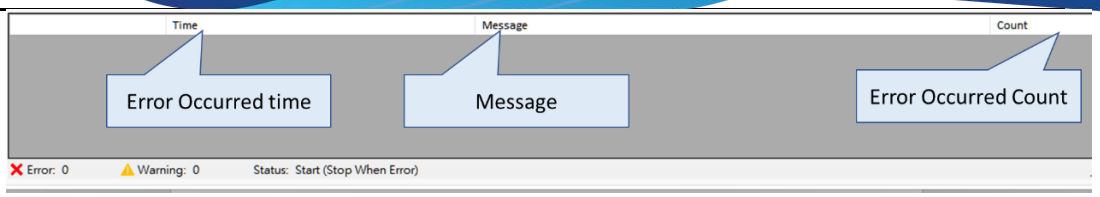
Items	Value
Index	14
Task Type	2:Explicit
Description	Verity AI ch 1
MAC ID	1
Service ID	0xE
Class ID	0x64
Instance ID	0x1
Data Length	1
Data	0x01
Delay(ms)	0
Explicit Verify Meth...	Float
Data Format	Little Endian
Start Byte	0
Start Bit	0
Range Min	-5
Range Max	5
Raw Data	0xF0 0x00 0xF0 0xBA
Value	-0.001831083

Figure: Example of real-time Explicit data

If an error occurs, it will be displayed in the table at the bottom.

If you want to know more details about the error, you can use the 

function to export the test log.


Time
Message
Count
Error Occurred time
Message
Error Occurred Count
Error: 0
Warning: 0
Status: Start (Stop When Error)

6. Frequently Asked Questions (FAQ)

6.1 Why does a message box saying “Please make sure if…” appear after selecting the master module?

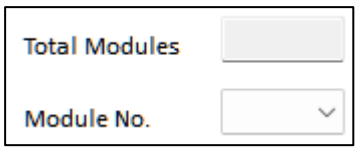
Message Example:

(The actual message content may vary depending on the system language)

Solution:

1. Go to the ICPDAS official website and search for your master module.
2. Navigate to the product page and open the Download Center.
3. Locate and download the SDK (Software Development Kit).
4. Install the SDK and restart DNM_Utility.

6.2 Why can’t the module be found, and why do Active Module or Active Board actions fail?



Solution:

1. Go to the ICPDAS official website and search for your master module.
2. Navigate to the product page and open the Download Center.
3. Locate and download the driver.
4. Install the driver and restart DNM_Utility.