

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

Copyright 2024 by ICP DAS Co., LTD. All rights reserved worldwide.

Trademark

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

Revision

Version	Date	Author	Description
3.0	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

Revision	2
1. General Information	4
1.1 <i>DeviceNet Introduction</i>	4
1.2 <i>DeviceNet Applications</i>	6
1.3 <i>DeviceNet Master Series Product Characteristics</i>	7
2. DeviceNet Master Software Utility for Windows	10
2.1 <i>Introduction</i>	12
2.2 <i>Tutorial Demos</i>	13
2.3 <i>Description of the Buttons</i>	25
3. XML Format Description	33
3.1 <i>Root(Header)</i>	33
3.2 <i>Device Information Segment: <Device></Device></i>	34
3.3 <i>Data Configuration Segment: <DataAttributes><Attribute> </Attribute></DataAttributes></i>	36
3.4 <i>XML Code Example</i>	40

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.

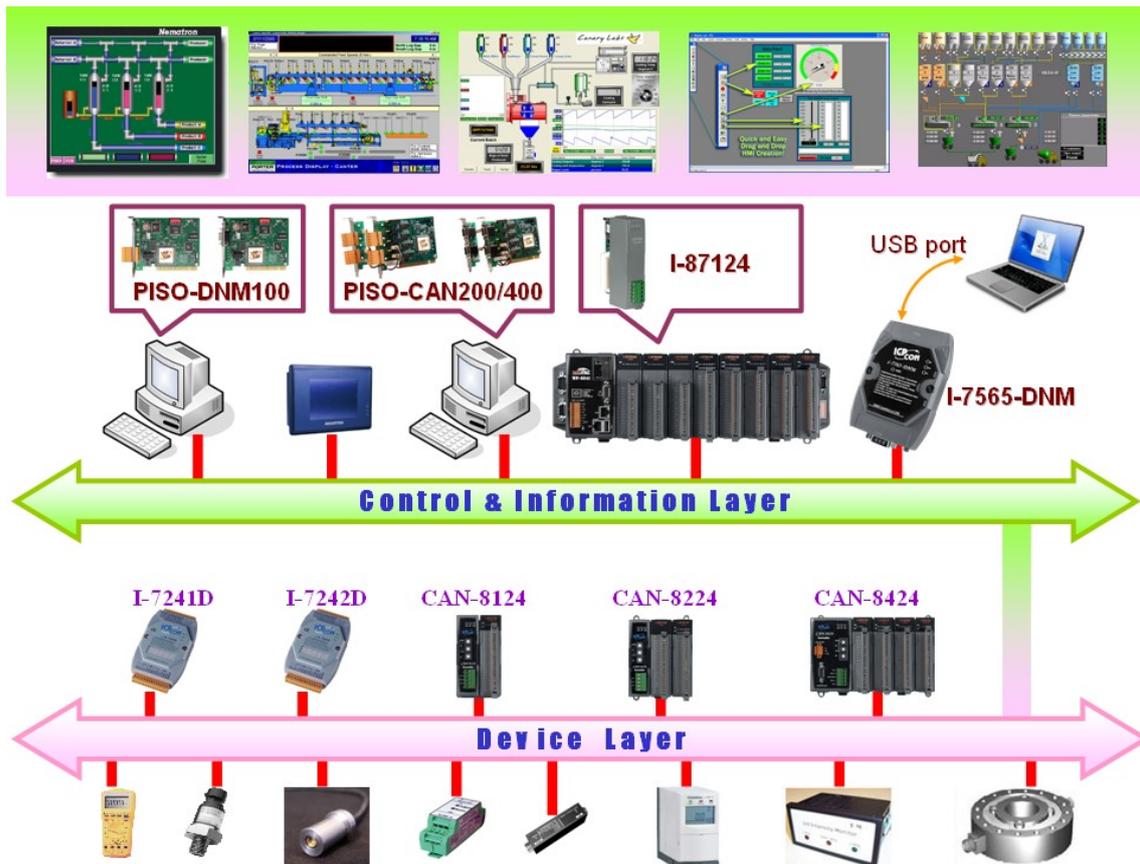


Figure 1.1.1 Example of the DeviceNet network

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.

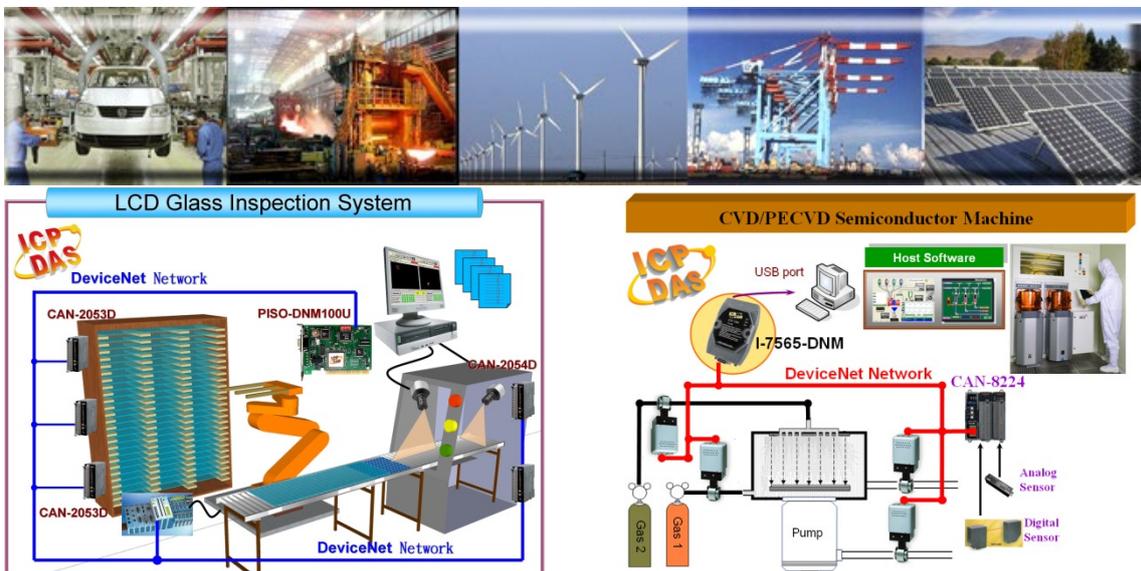
Baud rate (bit/s)	Max. Bus length (m)
500 K	100
250 K	250
125 K	500

Table 1.1.1 The Baud rate and the Bus length

1.2 DeviceNet 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
- Isolation wall manufacturing
- Overhead storage bin production
- Pocket-bread bakery
- Dinnerware production
- HVAC module production
- Textile machines
- Trawler automation system
- LCD manufacturing plant
- Rolling steel door production
- Bottling line
- Tight manufacturing



1.3 DeviceNet 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

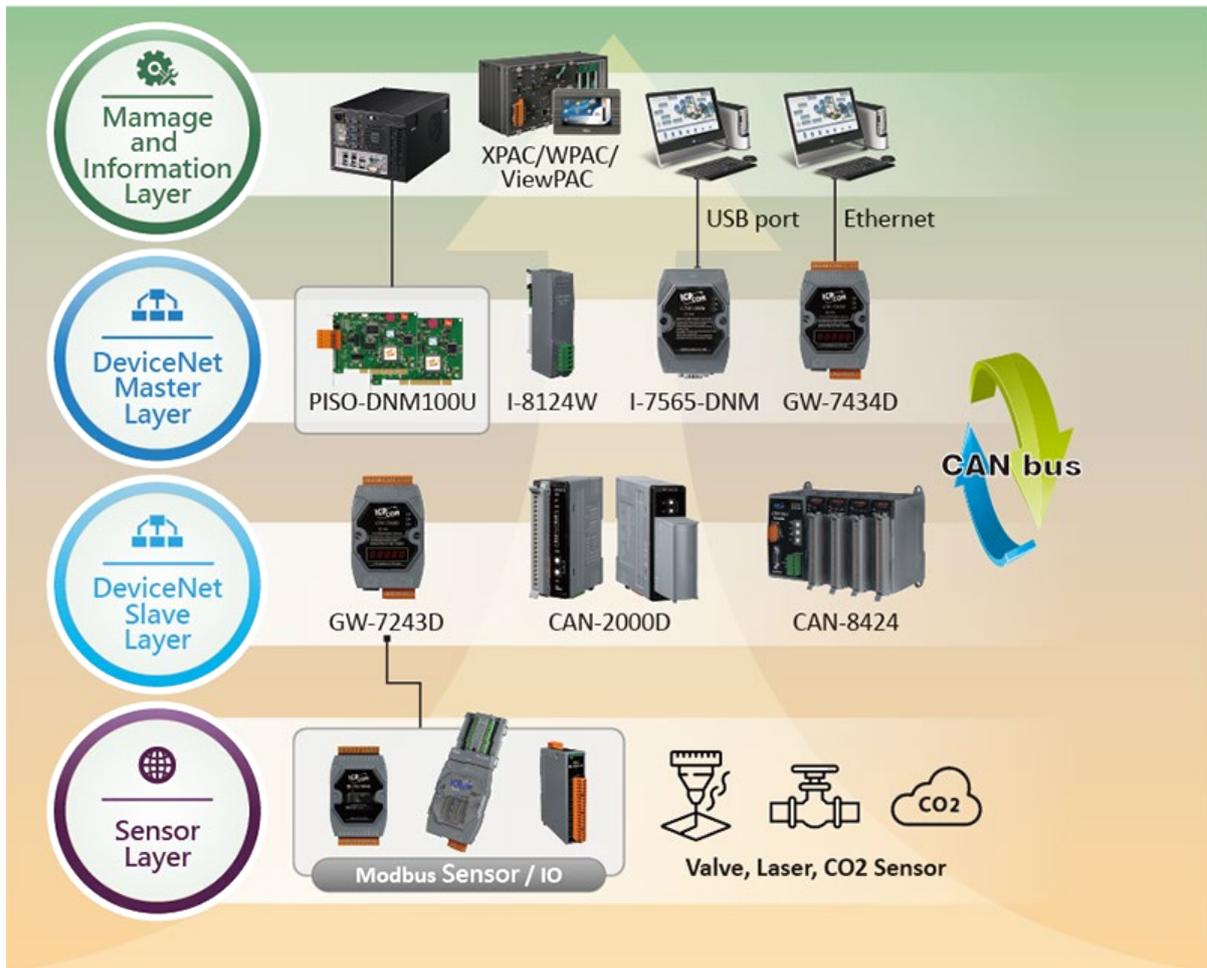


Figure 1.3.1 Application architecture

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

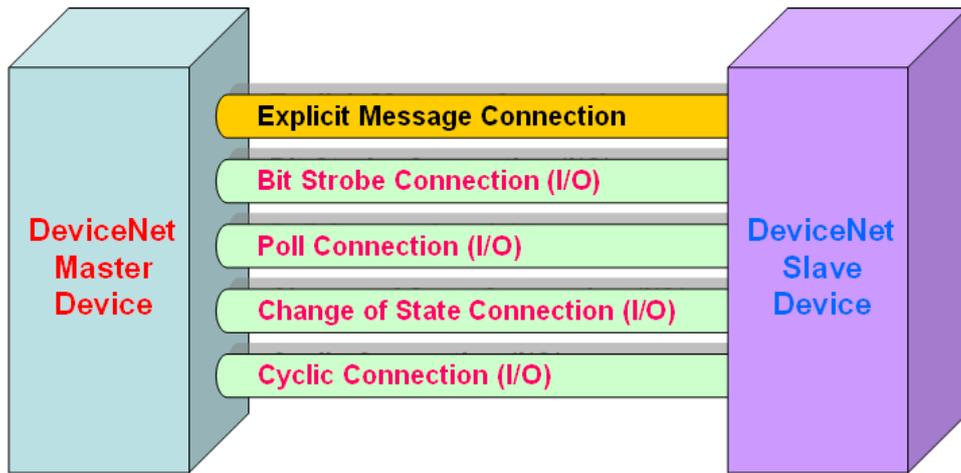


Figure 1.3.2 DeviceNet Messaging

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:

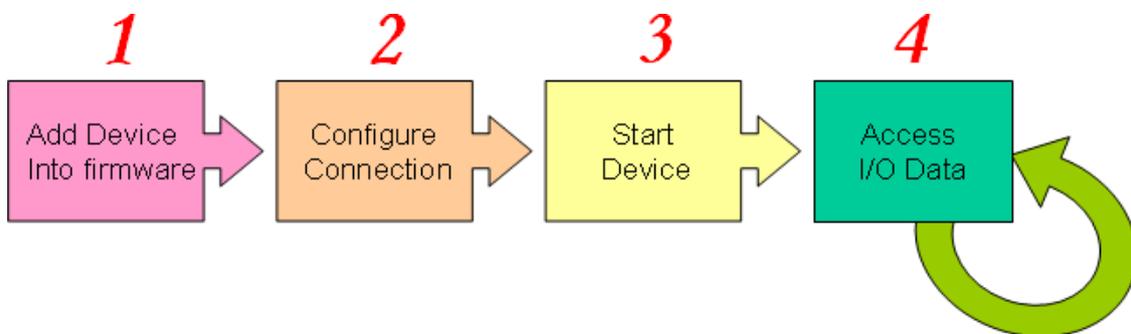


Figure 1.3.3 Four steps to establish connection

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.

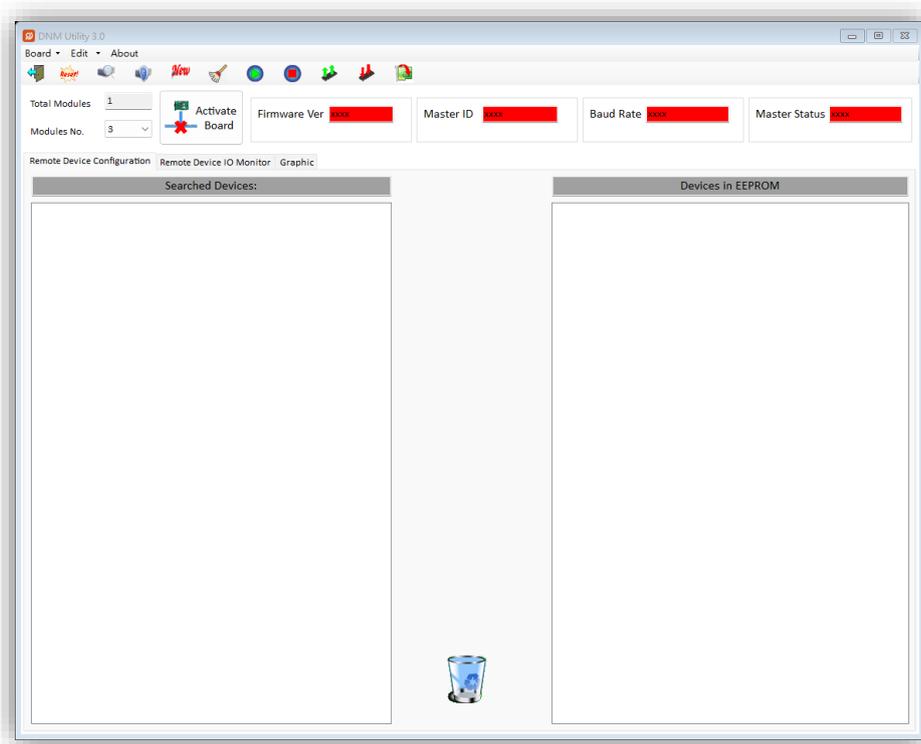
2. DeviceNet Master Software Utility for Windows

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

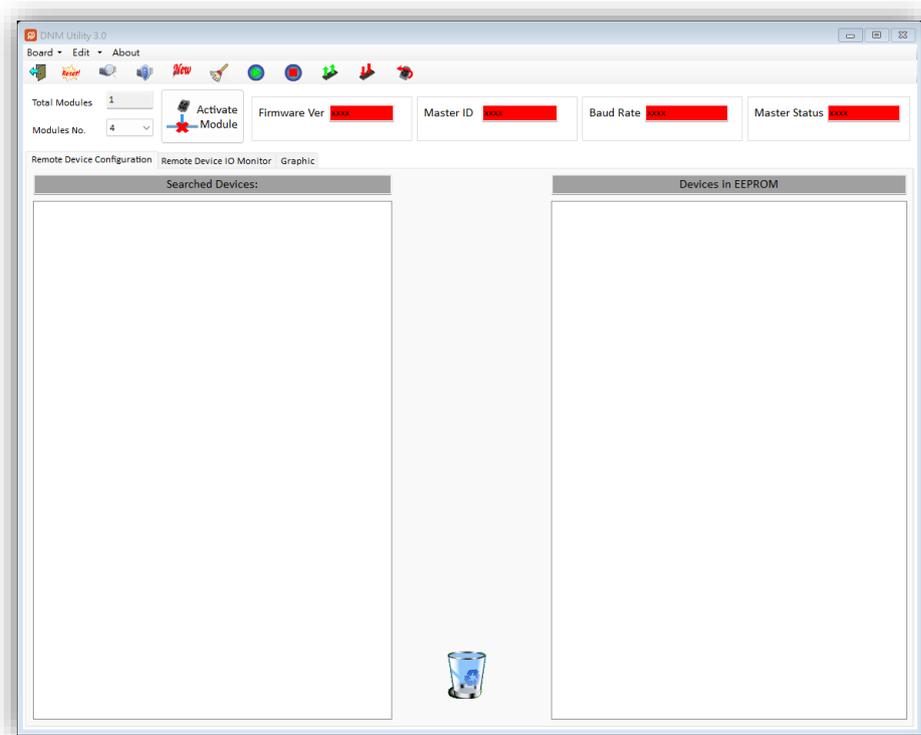
After running the utility program, the following window would be shown. Please select the hardware which you are using.



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



After choosing the “I-7565-DNM” button, the screenshot 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

This 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

This software 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

The software 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.

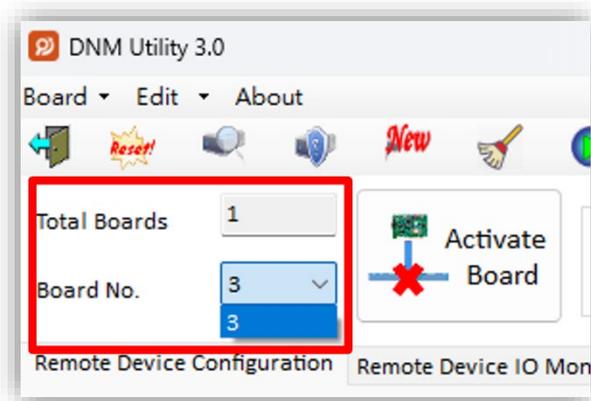
- Graphic

This software 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.

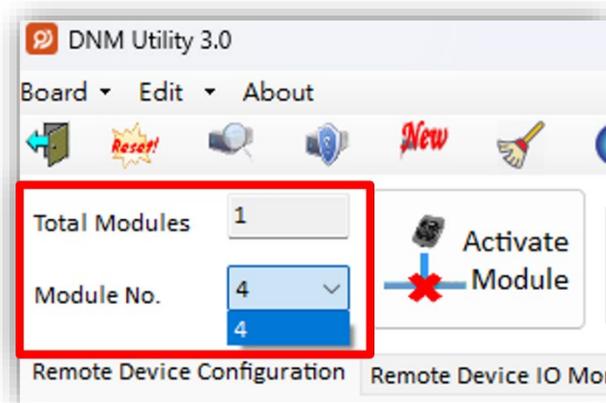
2.2 Tutorial Demos

2.2.1 Where to find the Hardware Information

1. The utility would search the number of master devices in your PC automatically. It shows the count of the boards which have been found.
2. For PISO-DNM100U, the utility also lists the ID of all boards in the “Board No.”field.
3. For I-7565-DNM, the utility also lists the ID of all modules in the “Module No.”field.



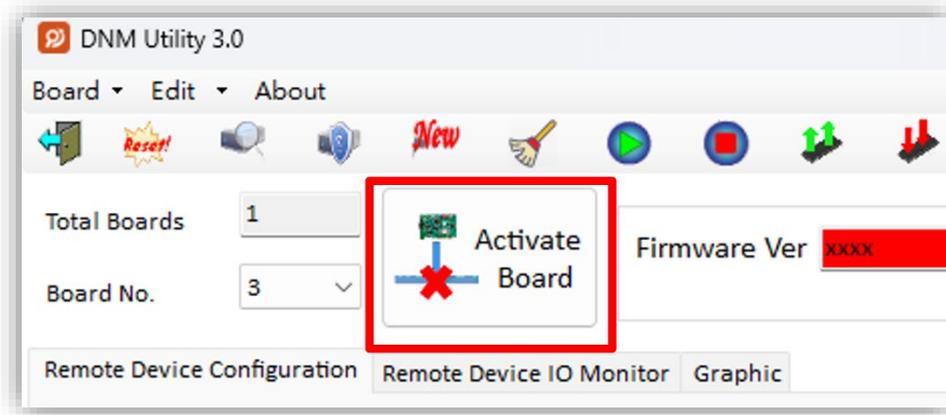
The picture for the PISO-DNM100U



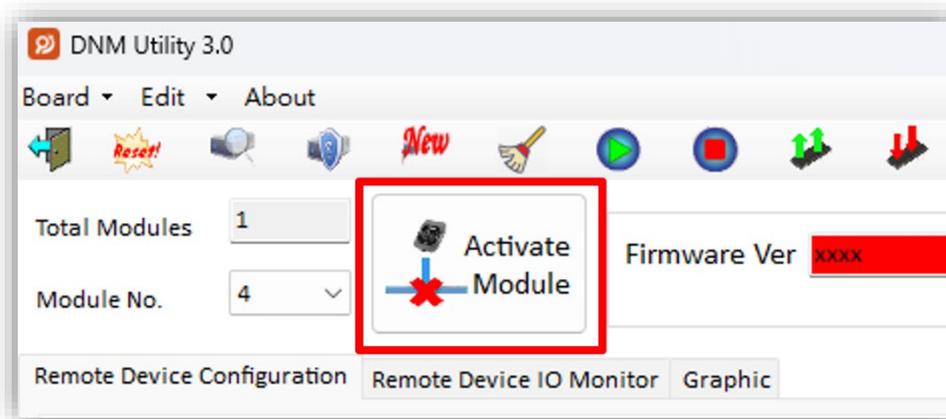
The picture for the I-7565-DNM

2.2.2 How to start using the utility

1. 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.

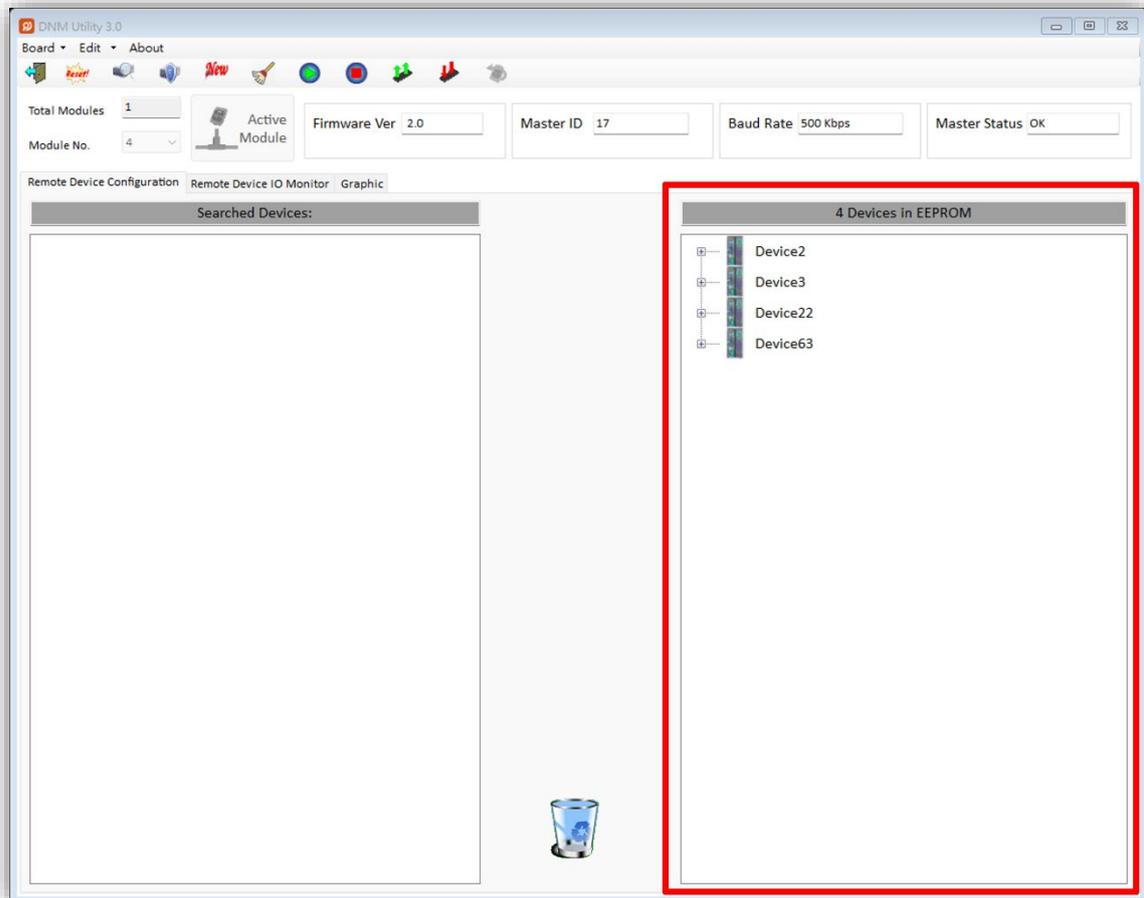


The picture for the PISO-DNM100U



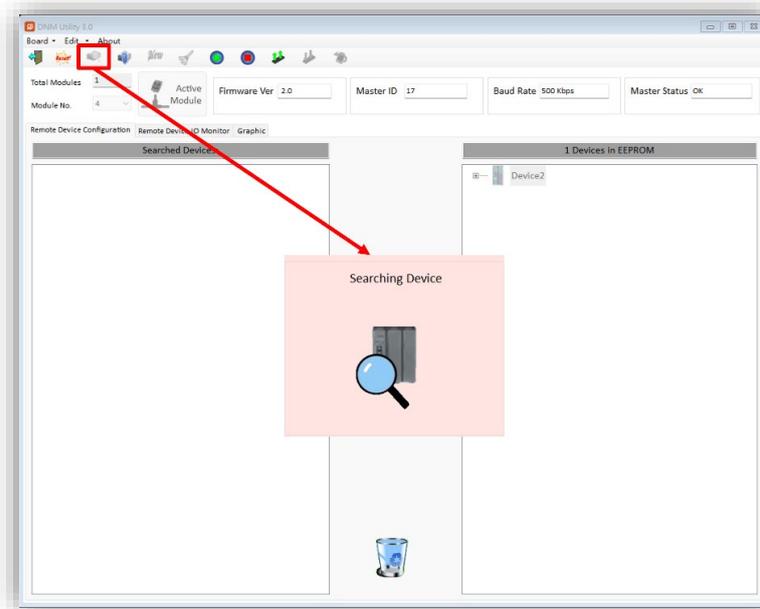
The picture for the I-7565-DNM

- 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.

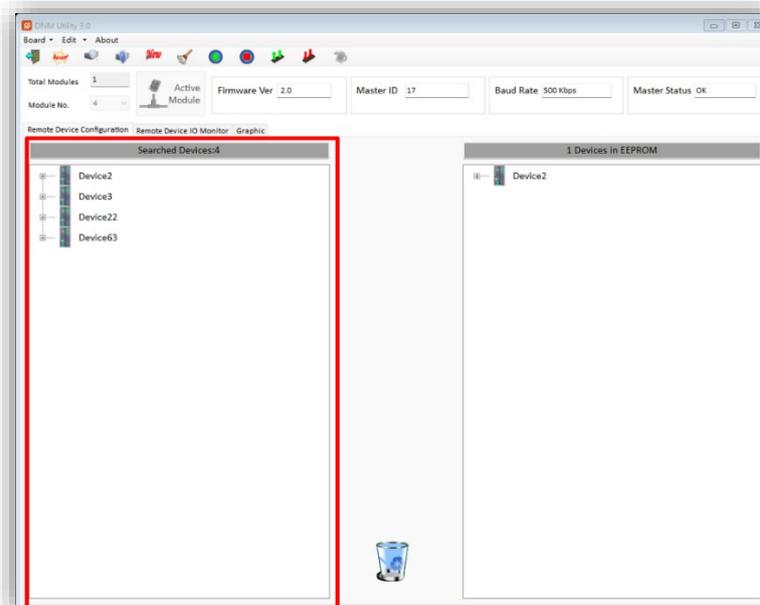


2.2.3 How to search the slave devices

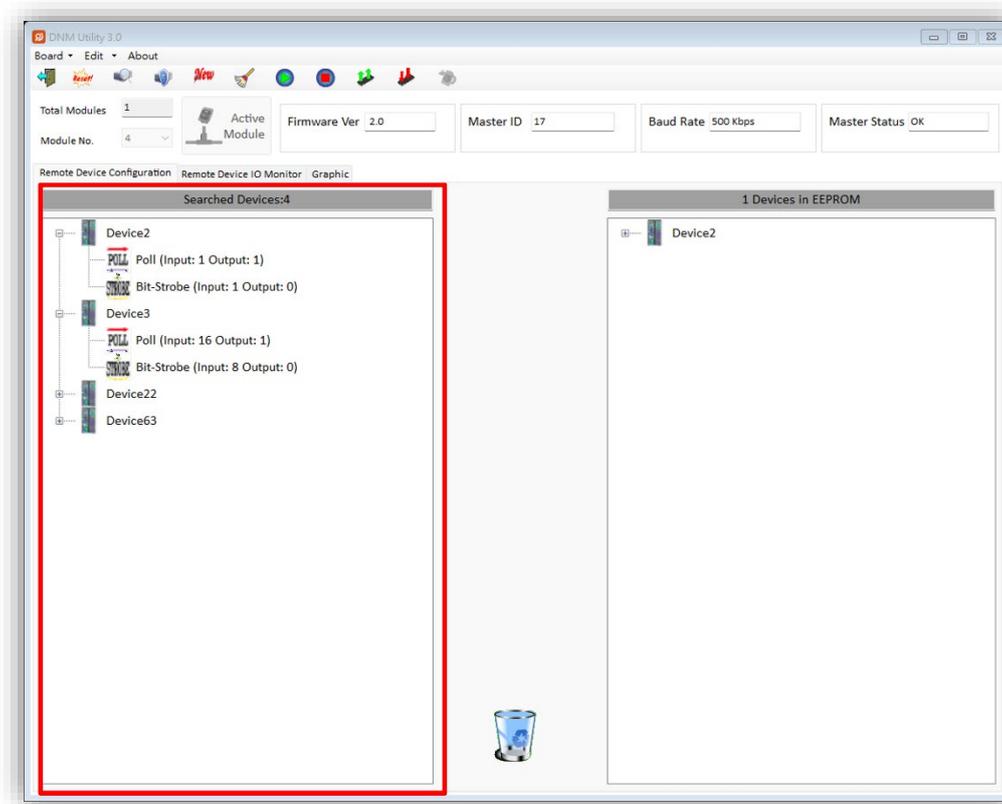
1. After the board has been activated, the users can press the “Search all Devices” button shown below. As the users press the button, the warning message would pop-up. In this example, we can ignore it. The screen shows a waiting dialog. It takes about 30 seconds to search the whole slave devices in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



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

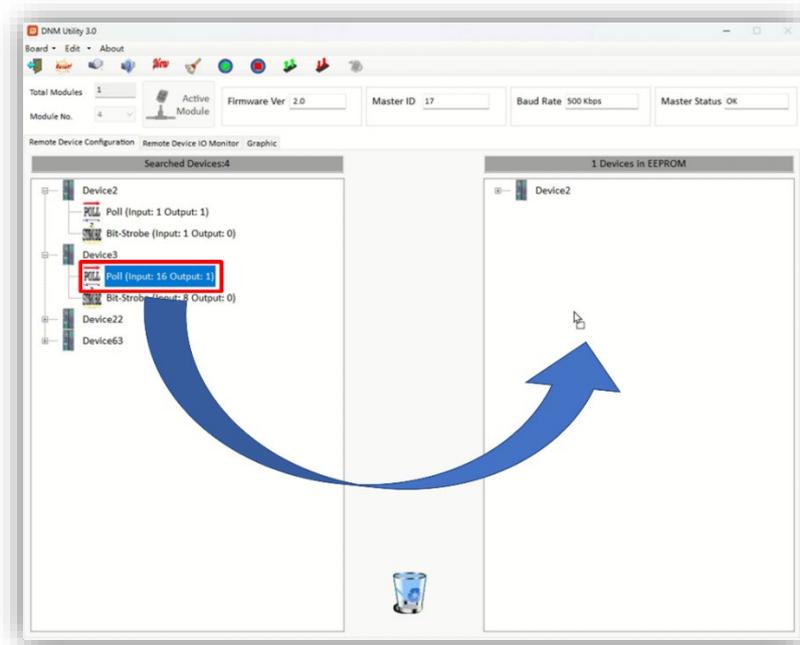


- The users can expand the device to find out more I/O connection information of those devices. The users can use this I/O information to develop your configuration in the EEPROM.

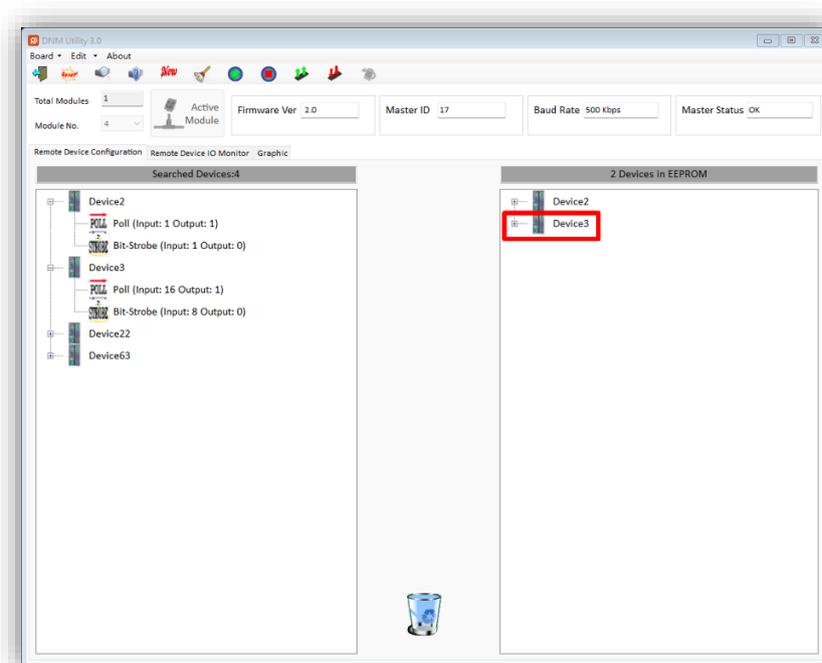


2.2.4 How to add I/O information into the EEPROM

1. Please activate your board or refer to section 2.2.2
2. Please search all devices or refer to section 2.2.3
3. Please select one of the I/O connection items in the “Searched Devices” field.
And drag the item into the “EEPROM” field.

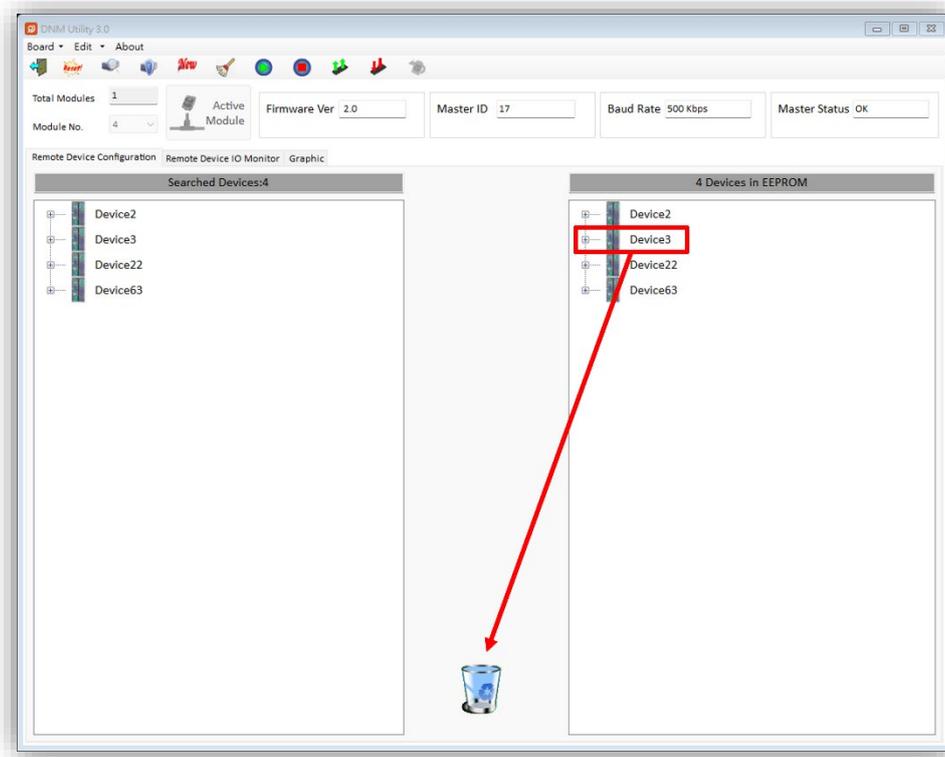


4. If the mission is successful, the users can find the selected item has been added into the “EEPROM” field.



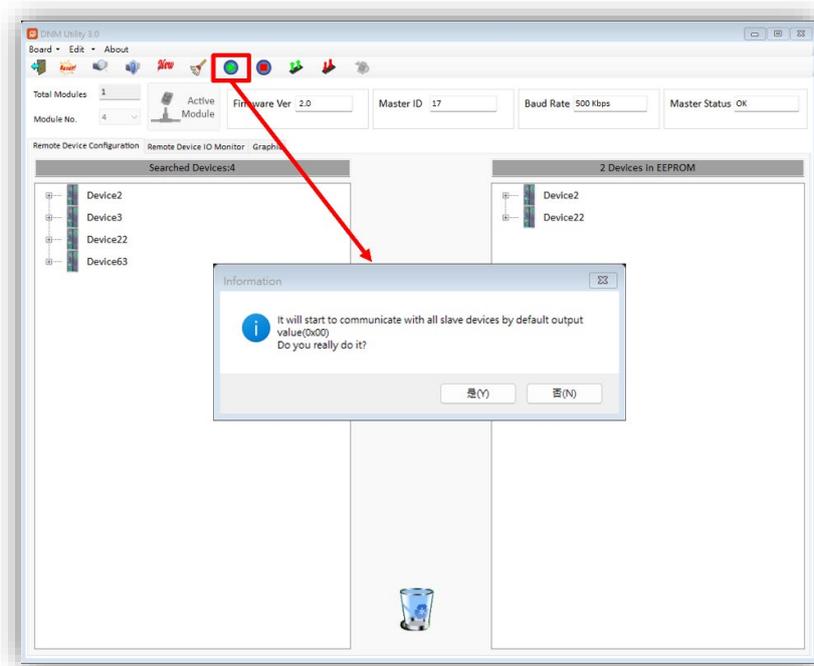
2.2.5 How to remove I/O information from the EEPROM

1. Please activate your board or refer to section 2.2.2
2. Please select one of the device items in the “EEPROM” field. And drag the item into the “trash can” image.

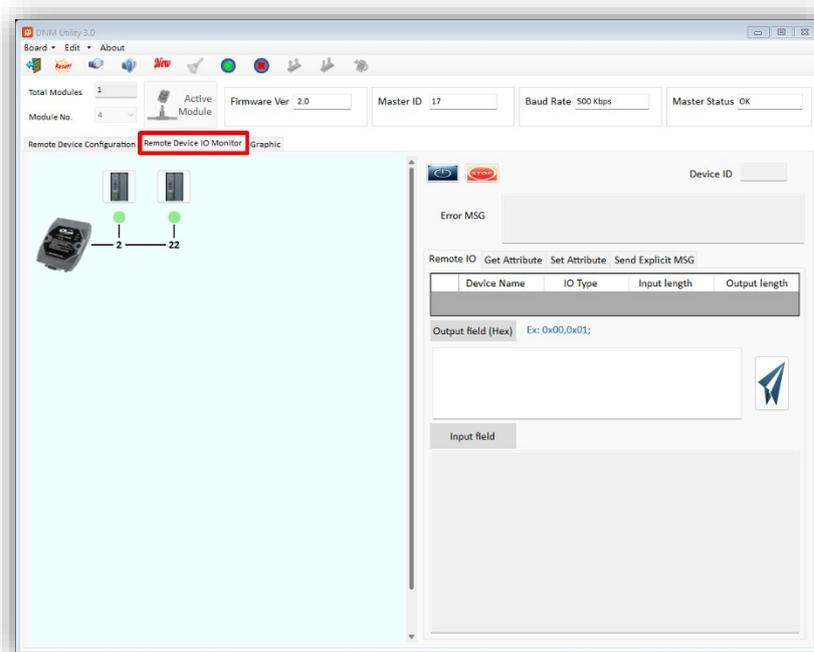


2.2.6 How to read/write the I/O data form the slave device

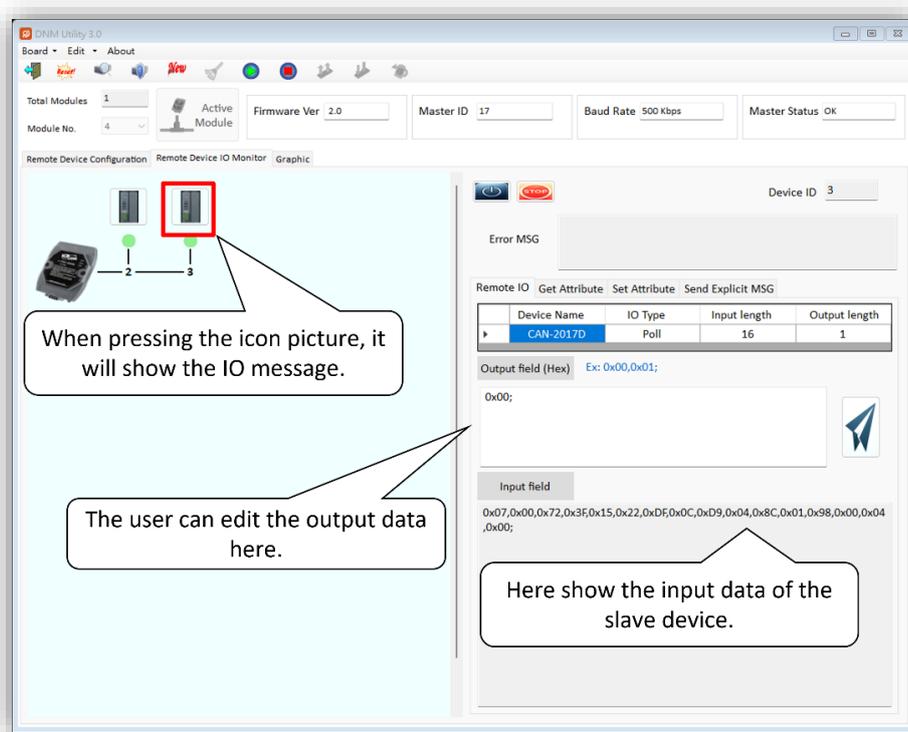
1. If the users have no I/O configuration in the EEPROM, please refer to section 2.2.4 to add at least one I/O configuration.
2. Please press “Start all Device” button to communicate with all slave devices. The information message would pop-up. In this example, we can ignore it.



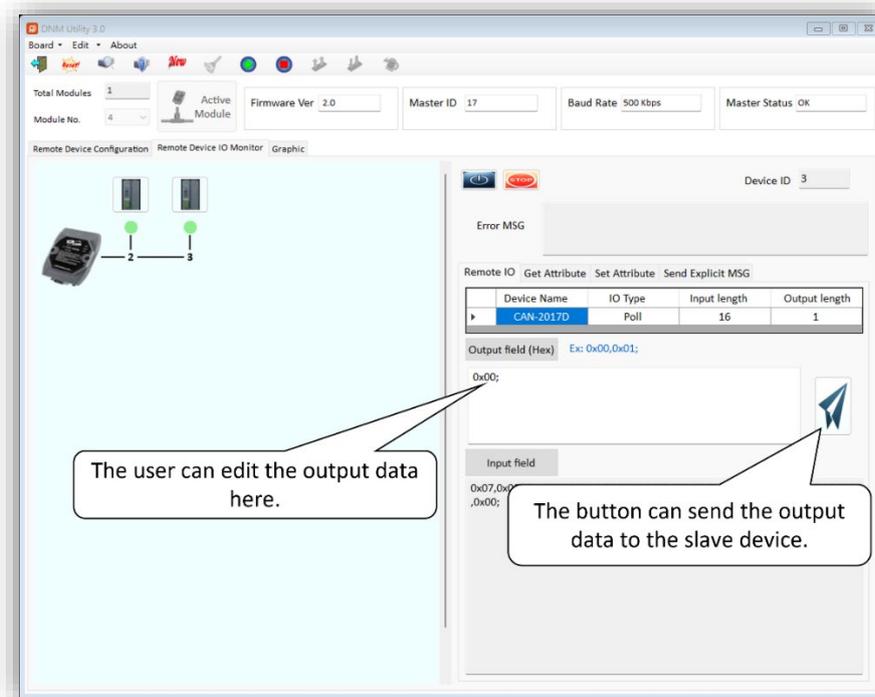
3. The users can click “Remote Device I/O Monitor” page to view the I/O data of the slave devices.



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

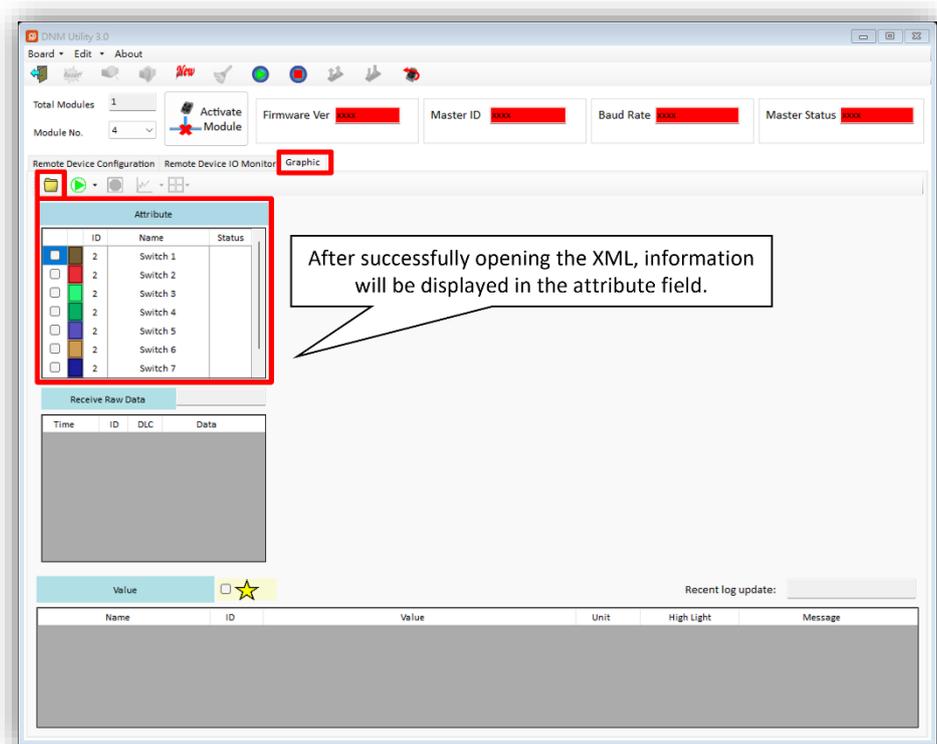


- The users can press "Write output data" button to send the output data to the slave device.



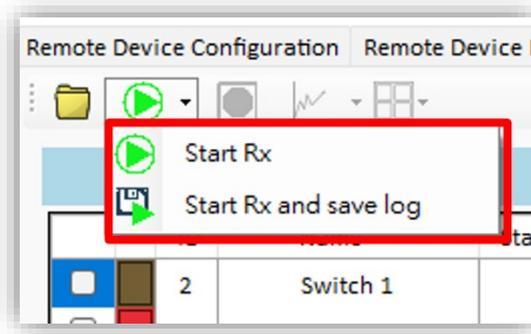
2.2.7 How to import XML file

1. 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.

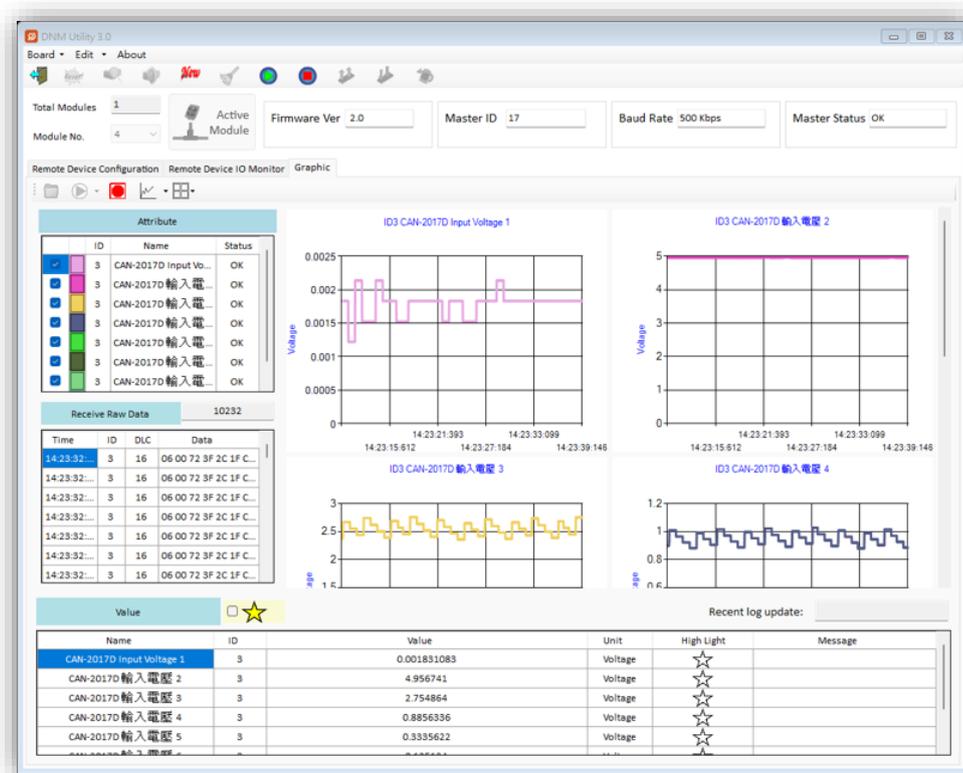


2.2.8 How to Start Rx after importing XML

1. Selecting “Start Rx” will directly initiate communication with the slave devices. Selecting “Start Rx and save log” will open a file dialog to choose the storage location for the log file and then begin Rx.
2. The log file is in CSV format, and its contents include the following fields: Time, Device ID, Attribute Name, Value, and Raw Data.

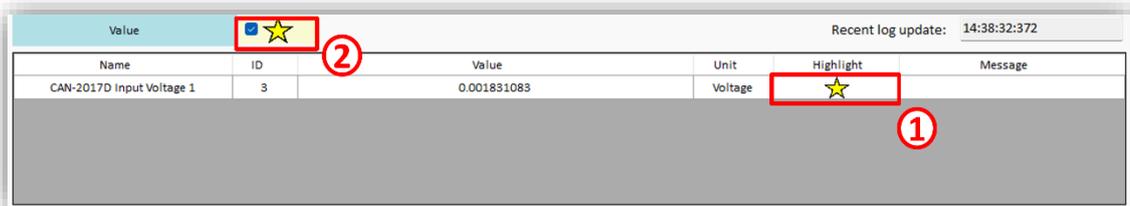


3. When "Start Rx" is in progress, the screen will display all the receive data and converted information.



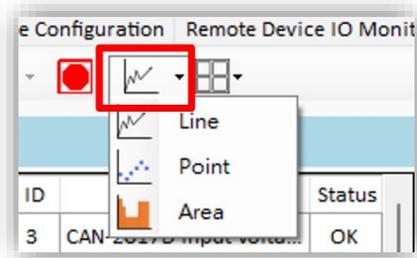
2.2.9 How to use highlight star

1. 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.

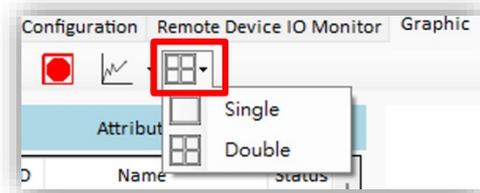


2.2.10 How to change chart display format

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



and



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



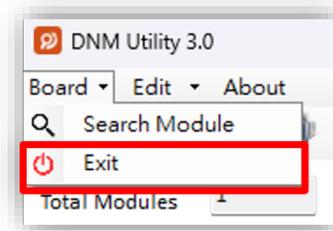
or

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.

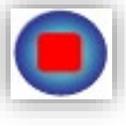
2.3.8 Start All Device



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



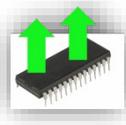
This button would disconnect the communication with all slave devices which have configured in the EEPROM. All remote slave devices will change to the "off-line" state.

2.3.10 Clear the EEPROM



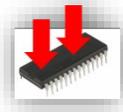
This button will clear all the slave devices information stored in the EEPROM.

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.

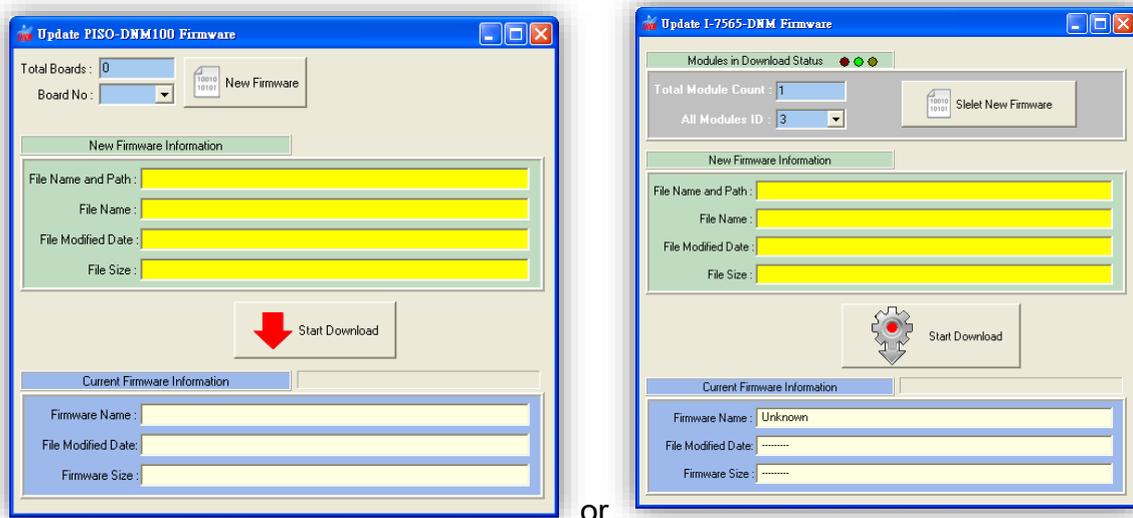


Figure 2.3.1 Update Firmware

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 figure 2.3.2.

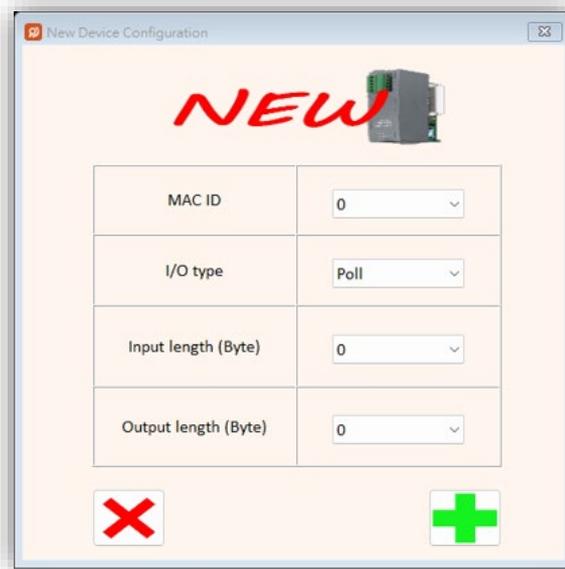


Figure 2.3.2 Add New Device by user-defined

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 Figure 2.3.3.

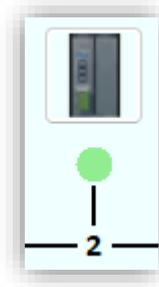


Figure 2.3.3 Green color LED

2.3.16 Stop Device

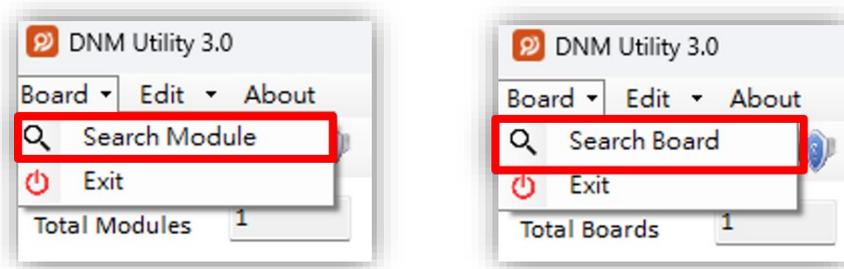


This button can stop to communicate with the device which the users have selected. If the function is successful, the LED image will turn red as shown in Figure 2.3.4.



Figure 2.3.4 Red color LED

2.3.17 Search Board

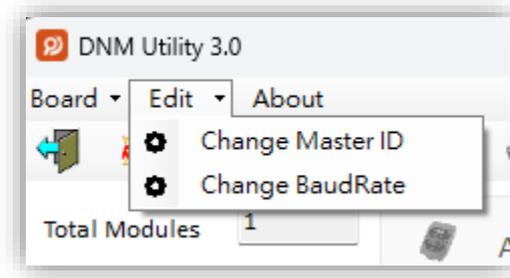


or

"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 figure 2.3.5.

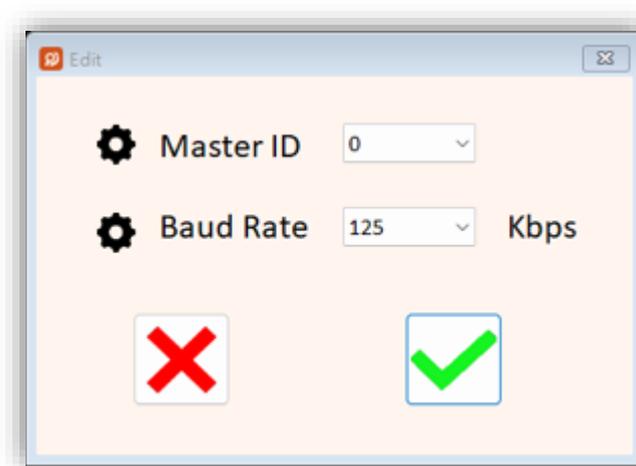


Figure 2.3.5 Change MAC ID and Baud Rate Dialog

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

3. XML 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:

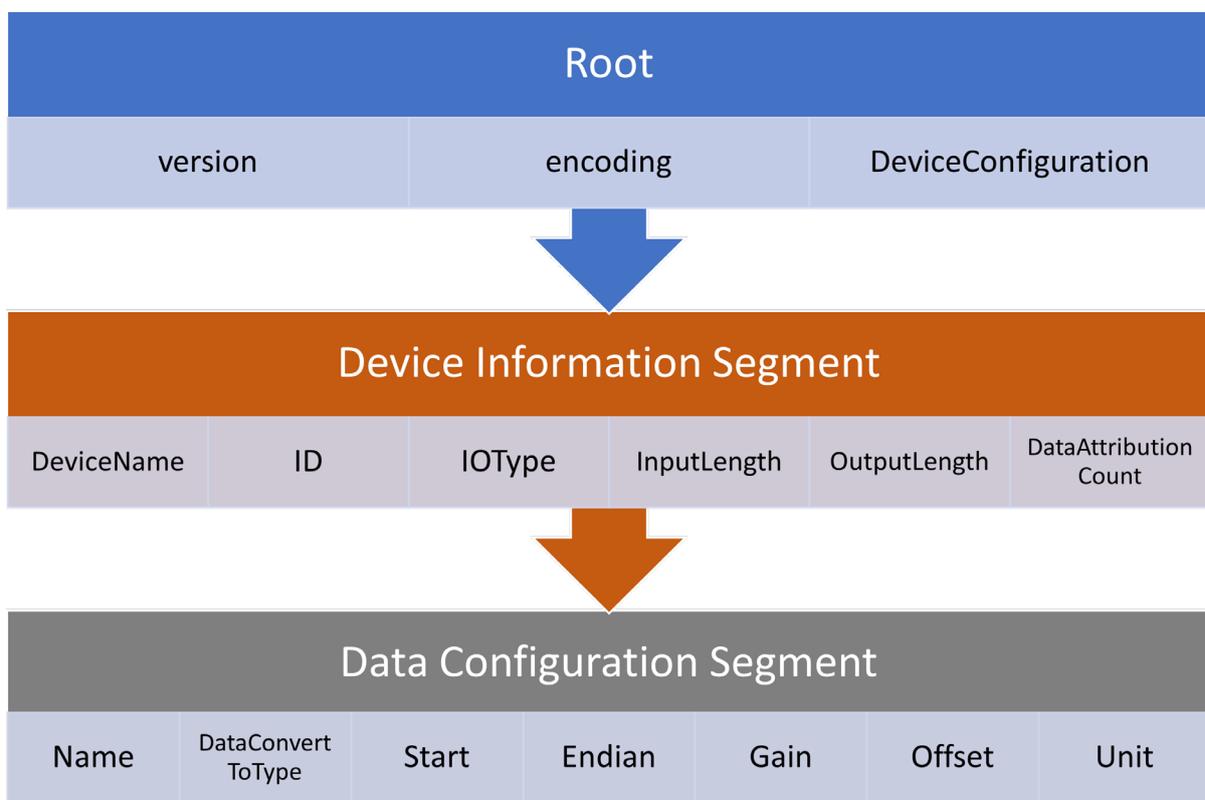


Figure 3.1.1 XML flowchart

3.1 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 Device Information Segment: <Device></Device>

The content within <Device> includes the basic information of the device. Using CAN-2017D as an example. Figure 3.2.1 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 3.2.1 Device information code

3.2.1 Required and Optional Parameters

Table 3.2.1 provides all the required or optional parameters of <device> field.

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

Table 3.2.1 <Device> field used parameters

3.2.2 Accept Value for Device Information Field

Table 3.2.2 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 3.2.2 Accept value for <Device> field

Based on the field information above, Figure 3.2.2 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.

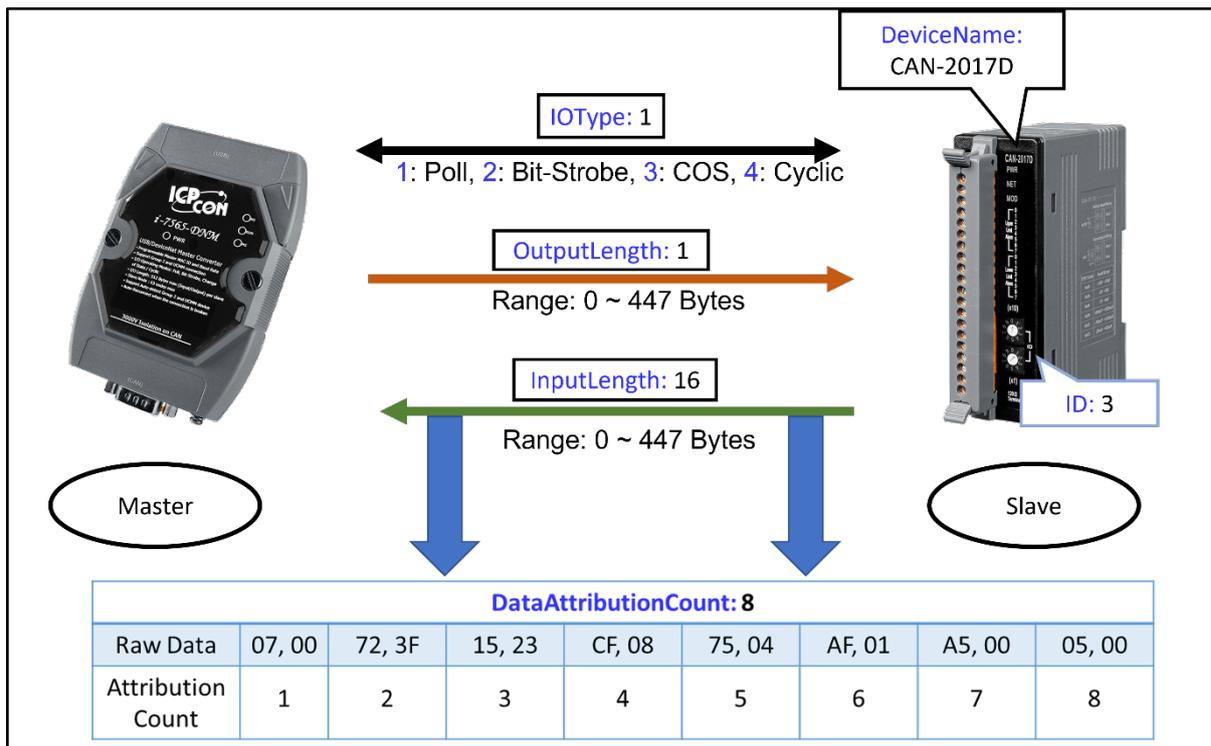


Figure 3.2.2 Device information diagram

3.3 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. Figure 3.3.1 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>0.0003051804379339284</Gain>
    <Offset>0.0</Offset>
    <Unit>Voltage</Unit>
  </Attribute>
</DataAttributes>

```

Figure 3.3.1 Attribute segment code

3.3.1 Required and Optional Parameters

Table 3.3.1 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 3.3.1 Attribute field used parameters

3.3.2 Accept Value for Data Attribute Field

Table 3.3.2 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 3.3.2 DataConvertToType and Endian field accept value

3.3.3 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. Figure 3.3.2 illustrates the relationship between the “Start” field of each attribute and the byte offset when “DataConvertToType” is set to Int16.

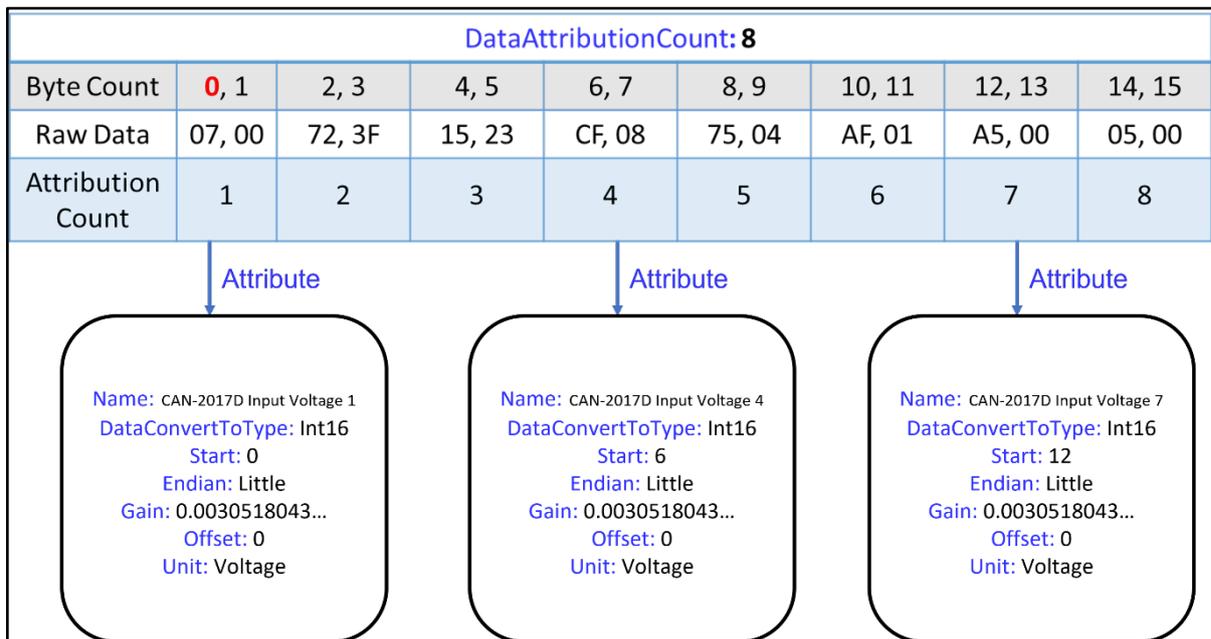


Figure 3.3.2 CAN-2017D data attributes

3.3.4 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. Figure 3.3.3 shows the input format for "Start" when "DataConvertToType" is set to bool.

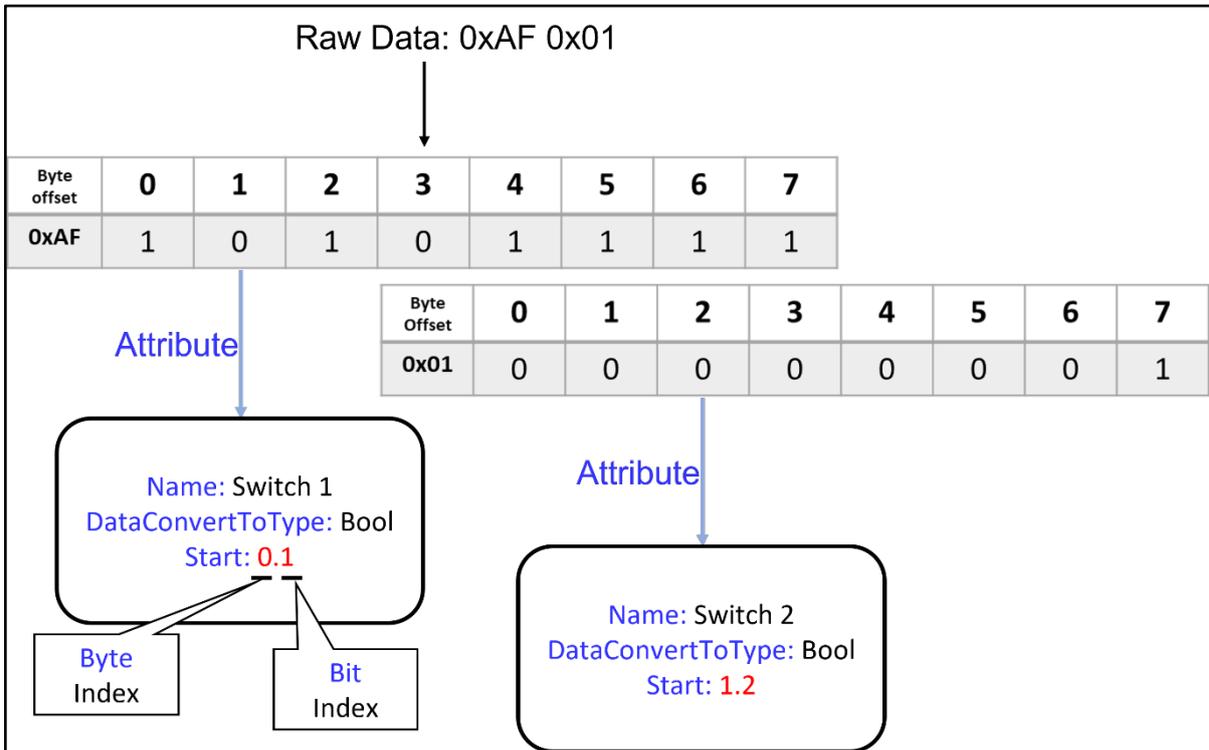


Figure 3.3.3 While DataConvertToType is bool data attributes

Notice! Byte Index and Bit Index start from 0.

3.3.5 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.

$$\underline{\text{Physical value} = \text{Gain} * \text{Input data value} + \text{Offset}}$$

Using CAN-2017D as an example. Figure 3.3.4 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>0.0003051804379339284</Gain>
      <Offset>0.0</Offset>
      <Unit>Voltage</Unit>
    </Attribute>
  </DataAttributes>
</DeviceConfiguration>
```

Figure 3.3.4 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>).

3.4 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 $\pm 10V$.

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”.