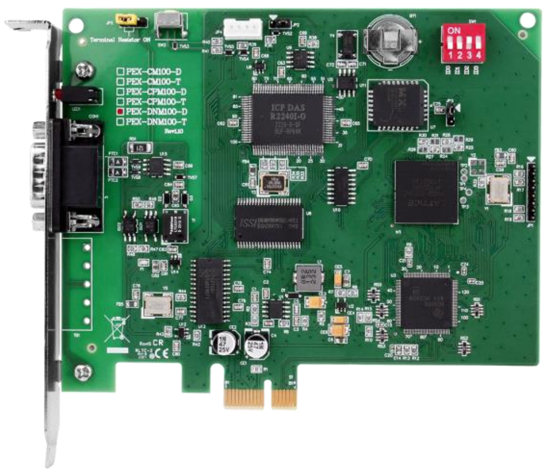
PISO-DNM100U-D, PISO-DNM100U-T(v2.1 or newer) PEX-DNM100-D, PEX-DNM100-T DeviceNet Master PCI/PCI-E Board

User's Manual

PISO-DNM100U (v2.1 or newer)


PEX-DNM100U

Warranty

Warning

Copyright

Trademark

Revision

Version	Firmware Version	Date	Author	Description
3.6	3.1	2023/ 04/10	Johney	1. Add PEX-DNM100 board 2. Rename to “DNM100 Boards” 3. Support PISO-DNM100U v2.1 or newer version.
3.5	3.1	2022/ 02/10	Johney	Add new function 1. DNM100_PauseIOConnection 2. DNM100_ResumeIOConnection 3. DNM100_DisableKeepAliveMsg
3.4	2.6	2019/ 01/10	Johney	1. Add Venders’ DeviceNet Slave. 2. Update Windows version.
3.3	2.5	2015/ 12/24	Johney	1. Add Venders’ DeviceNet Slave. 2. Update pin assignment.
3.2	2.5	2013/ 06/06	Johney	Add new function 1. DNM100_GetAttributeW 2. DNM100_SetAttributeW 3. DNM100_SendExplicitMSG_W
3.1	2.4	2011 01/10	Johney	Add new functions * DNM100_ReadInputArea * DNM100_WriteOutputArea * DNM100_ReadbackOutputArea
3.0	2.2	2009/ 12/25	Johney	Add the information of the PISO-DNM100U board.
2.02	2.1	2009 07/14	Johney	Add the function DNM100_ReadbackOutputData
2.01	2.0	2009 01/16	Johney	Correct the function name. GW7304_xxx --> DNM100_xxx
2.0	2.0	2008 12/15	Johney	Reduce the number of the API function from 66 to 36. The new firmware is more efficient.
1.0	1.0	2008 02/15	Johney	This manual is for the PISO-DNM100 board.

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.

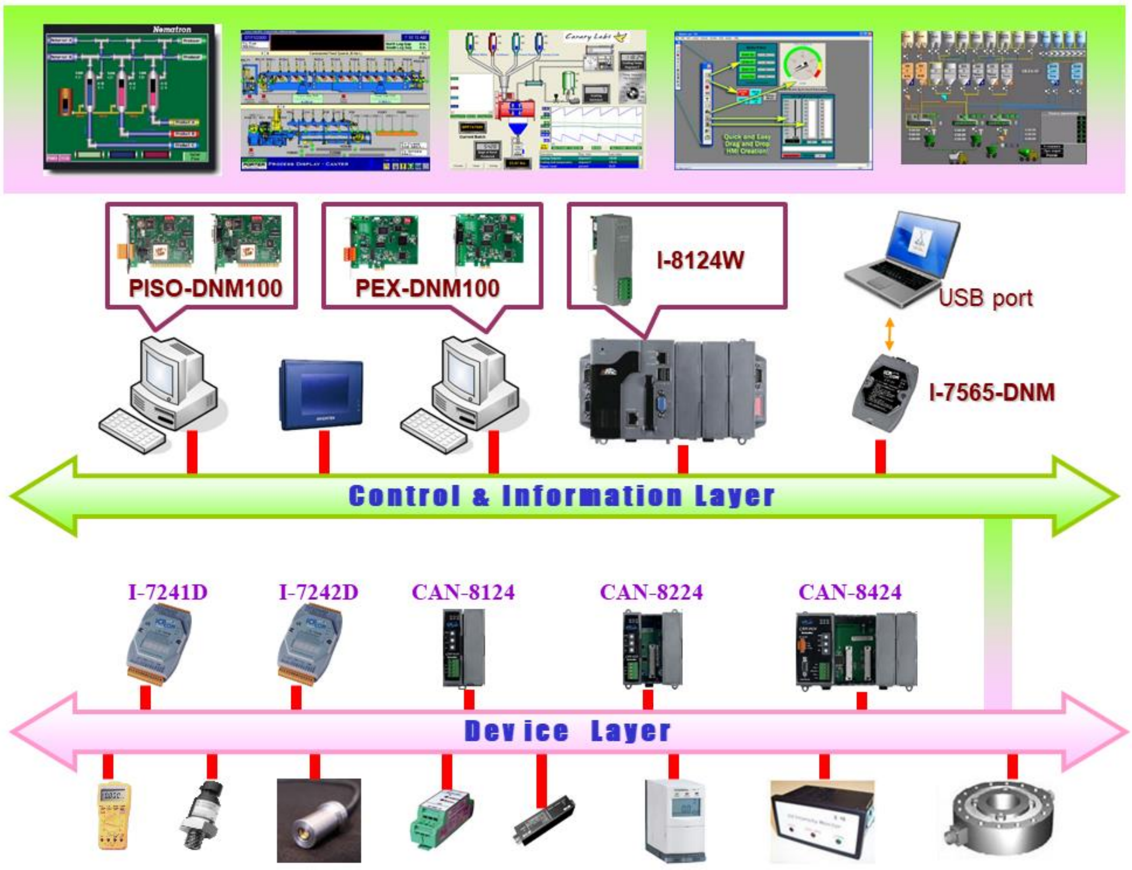


Figure 1.1 Example of the DeviceNet network

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 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). 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

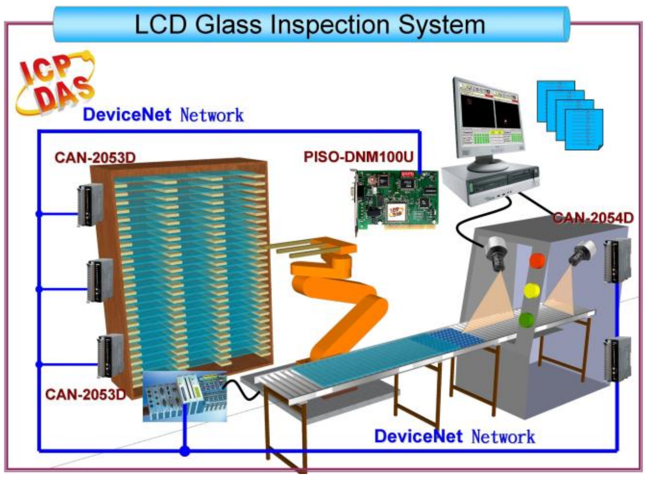
Table 1.1 The Baud rate and the Bus length

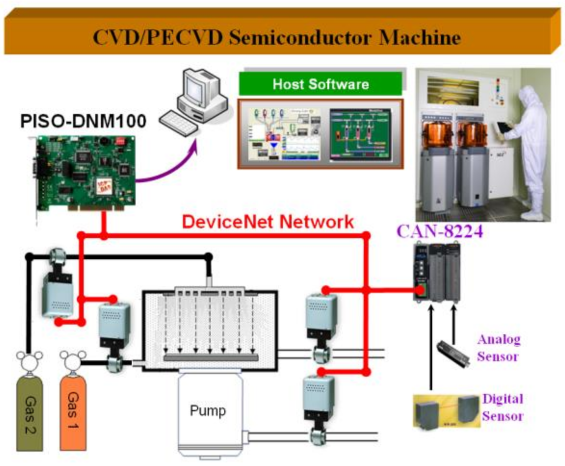
The PISO-DNM100U and PEX-DNM100 can represent an economic solution of DeviceNet application and be a DeviceNet master device on the DeviceNet network. The PISO-DNM100U and PEX-DNM100 supports Group 2 only Server and UCMM functions to communication with slave devices. It has an independent CAN bus communication port with the ability to cover a wide range of DeviceNet applications. Besides, the PISO-DNM100U and PEX-DNM100 uses the new CAN controller NXP SJA1000T, which provide bus arbitration, error detection with auto correction and re-transmission function. It can be installed on almost any windows-based system. It is popularly applied in the industrial automation, building automation, vehicle, marine, and embedded control network. Therefore, that is an easy way to develop the DeviceNet network with the PISO-DNM100U and PEX-DNM100.

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 DNM100 Series Board with Vendor's DeviceNet Slaves

We have communicated with the following DeviceNet slaves.

- Allen-Bradley(AB) **PowerFlex** series DeviceNet Inverters.
- Allen-Bradley(AB) **PowerFlex AC** Drives
- Allen-Bradley(AB) **PowerFlex AC** Drives with DriveLogix
- Allen-Bradley(AB) **PowerFlex DC** Drives
- Allen-Bradley(AB) 1769-ADN DeviceNet Adapter
- ADVANCED ENERGY Apex **RF generators** and power-delivery
- ABB DSQC 350A DeviceNet/Allen Bradley remote gateway
- BECKHOFF **CX1500-B520** series DeviceNet I/O modules.
- BECKHOFF **BK5250** series DeviceNet I/O modules.
- COSMOS **PS-7** series DeviceNet gas detectors.
- CELERITY **UNIT IFC-125** series DeviceNet devices.
- Festo CTEU-DN Bus nodes
- LINCOLN Power Wave AC/DC 1000 device
- MKS **683** series DeviceNet exhaust throttles.
- MKS **MFC (Mass Flow Controller)** series DeviceNet devices.
- MKS **DELTA-II FRC (Flow Ratio Controller)** series DeviceNet devices.
- MKS **DC Power Generator (OPT- xxx)** series DeviceNet devices.
- MKS Type T3B and T3P Valves With DeviceNet Interface
- MTS Temposonics R-Series sensor
- MKS MicroNode I/O series module
- MKS Baratron Type DMB Capacitance Manometer
- MTS **Temposonics R-Series** Position Sensors
- OMRON **DRT1-ID/ODxx** series DeviceNet I/O modules.
- OMRON **DRT2-MDxx** series DeviceNet I/O modules.

- OMRON **DRT2-ID08(-1)/MD16**
- OMRON **DRT2-OD08/MD16-1**
- OMRON **DRT2-OD08-1**
- OMRON **DRT2-ID16(-1)/OD16(-1)**
- OMRON **GRT1-DRT**
- OMRON **C200HW-DRT21**
- PFEIFFER VACUUM **HiPace series turbopumps**
- Swagelok **MS-VCM-D-6-0**, **MS-VCM-D-6-2** Digital DeviceNet Valve
- Swagelok **SS-PTX-D-G500-S4-K** Digital DeviceNet Pressure-Temperature Transducer
- Swagelok **SS-PTX-D-G500-SM-K** Digital DeviceNet Pressure-Temperature Transducer
- SMC **ITV series** Electro-Pneumatic Regulator
- SMC **Directional Control Valves**
- SICK **DME500 series** Distance Sensor
- Sevenstar CS200/CS220 Mass Flow Controller
- Weidmueller **SAI-AU M12 DN 16DI/8DO**
- Weidmueller **SAI-AU M12 DN GW 16DI**
- Weidmueller **SAI-AU M12 DN AI/AO/DI**

1.4 DNM100 Series Board Architecture

The PISO-DNM100U and PEX-DNM100 provides users to establish DeviceNet network rapidly by Master/Slave connection model. The PISO-DNM100U and PEX-DNM100 is a high-performance DeviceNet master board with one CPU inside. This architecture of the PISO-DNM100U and PEX-DNM100 almost doesn't cost CPU resource and really increases the work efficiency on DeviceNet network. Applying the PISO-DNM100U and PEX-DNM100, users don't need to take care of the detail of the DeviceNet protocol. The inside firmware implements the DeviceNet protocol to help users to establish the connection with DeviceNet slave devices easily. The illustration about the idea is shown as Figure 1.2.

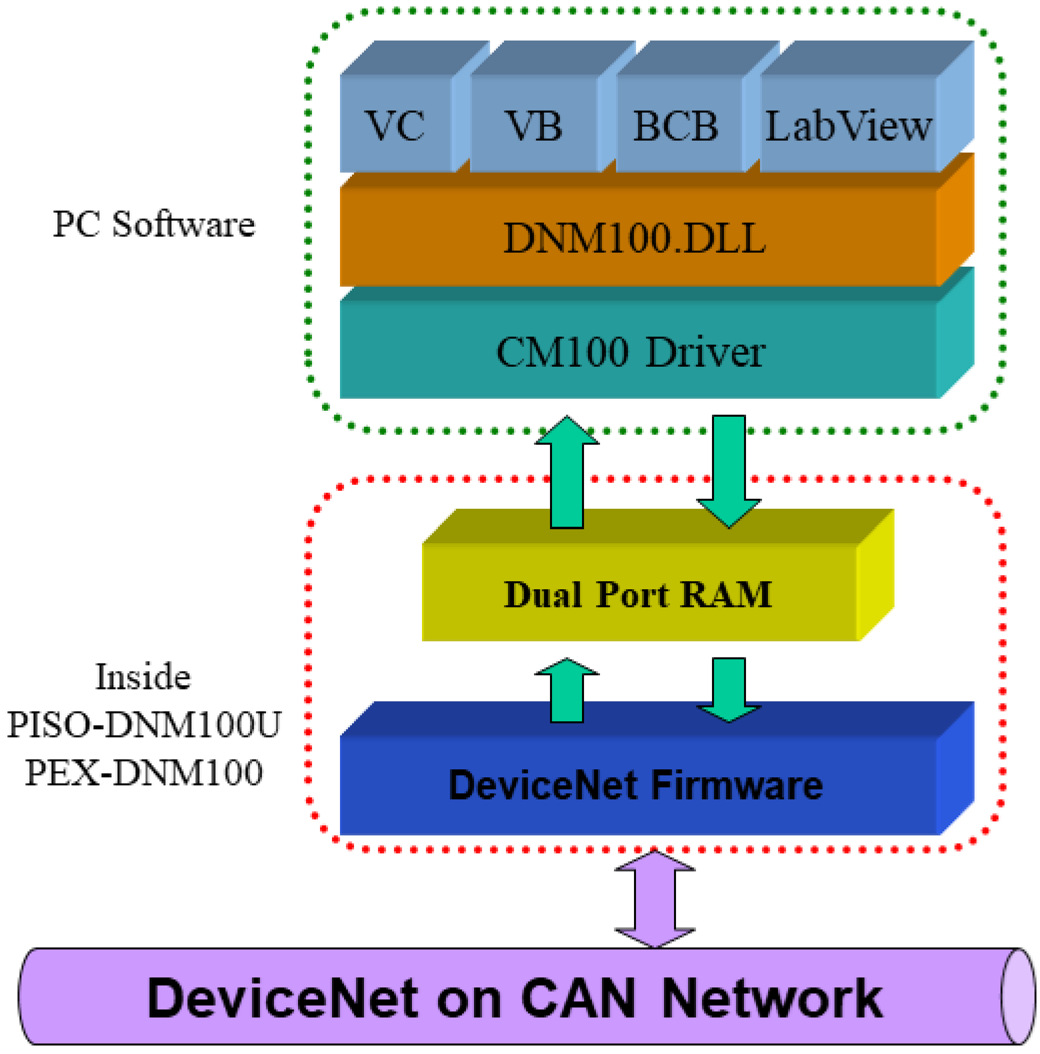


Figure 1.2 PISO-DNM100U and PEX-DNM100 illustration.

1.5 DeviceNet Master Characteristics

Using the API functions, users don't need to take care of the detail of the DeviceNet protocol. It can reduce the complexity of user's DeviceNet Master Software. The firmware 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.

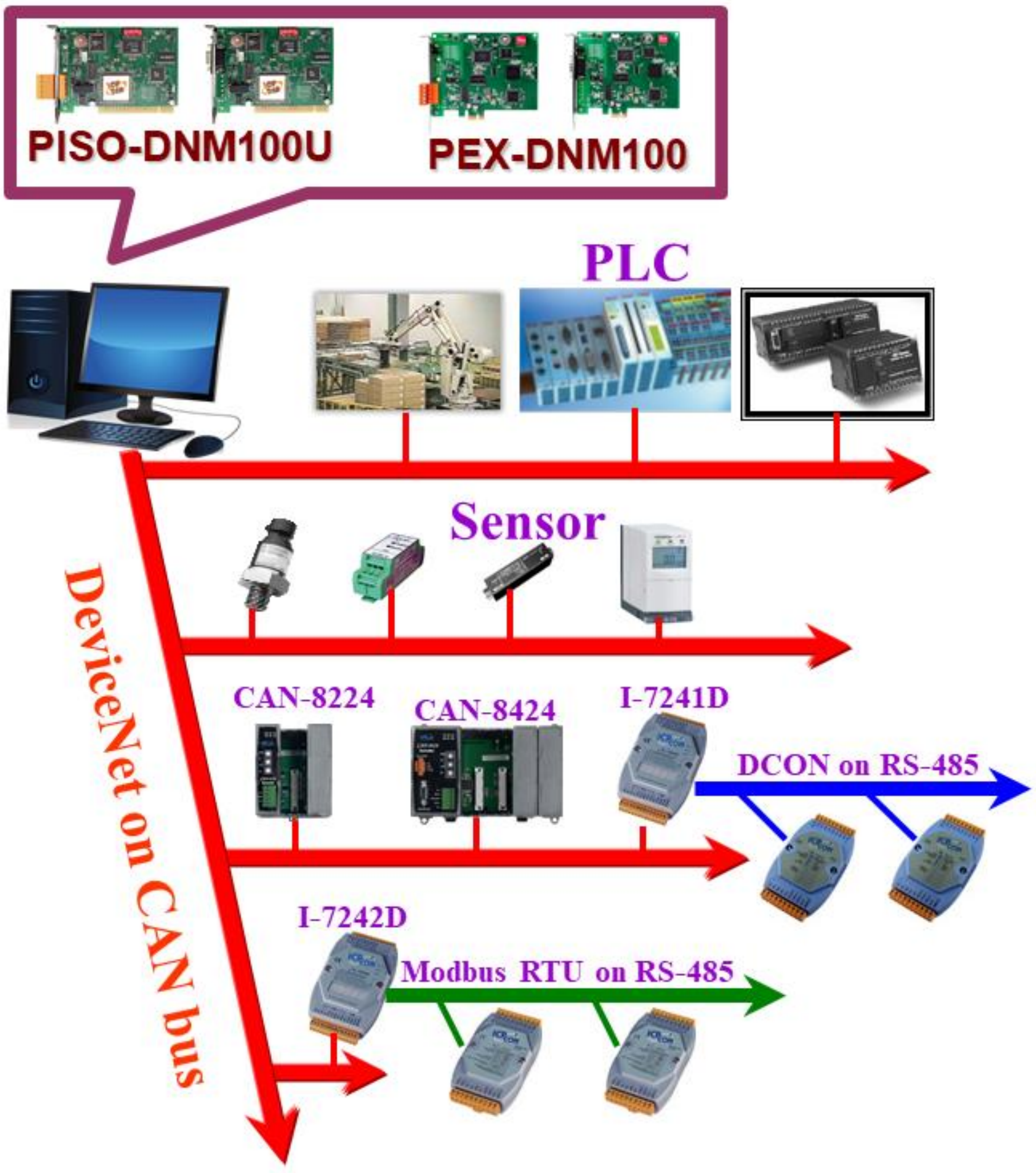


Figure 1.3 Application architecture

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 1.4.

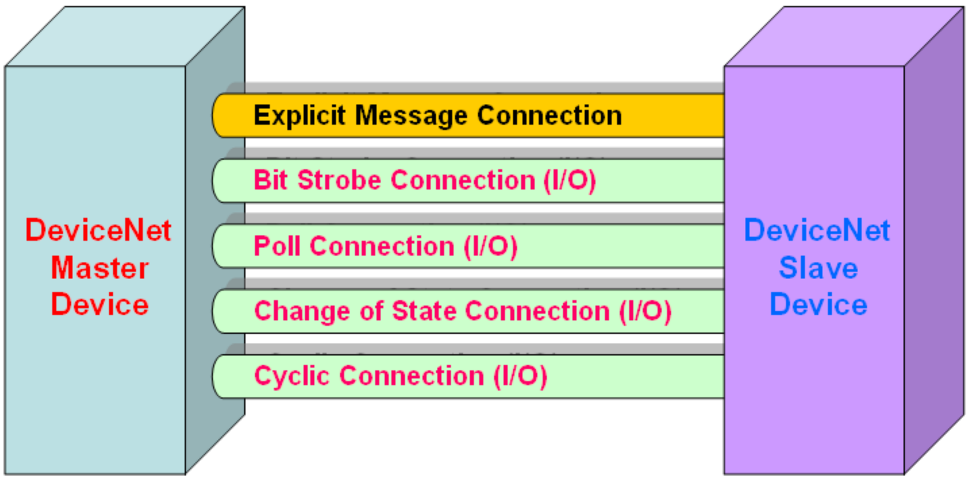


Figure 1.4 DeviceNet Messaging

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 1.5 demonstrates the basic process for the DeviceNet master communication. The every step function is described in below:

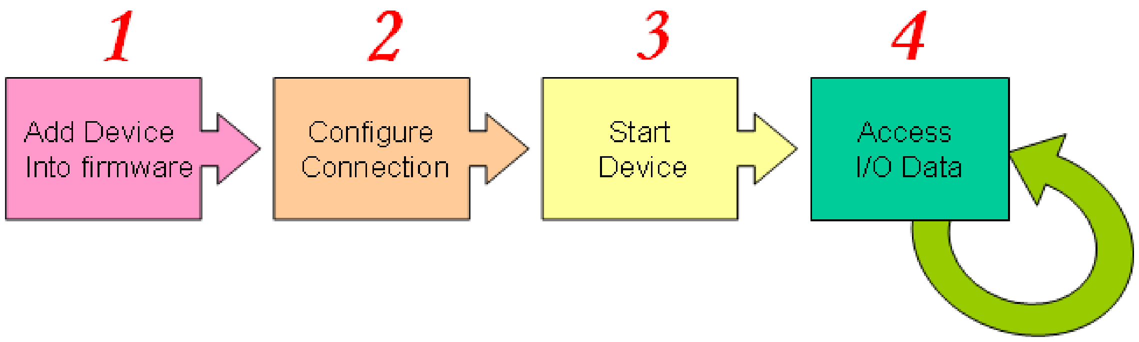


Figure 1.5 Four steps to establish connection

1. **Add device into firmware**

   You should provide the slave device’s MAC ID to add into firmware by using API function.

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 by using API function. The master will communicate with the slave device.

4. **Access I/O data**

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

After adding the device into the firmware, the master will wait for the I/O configuration information. Then users can create the I/O connections in the next step. Once I/O connections have been created and started, I/O data may be exchanged among devices in the DeviceNet network according to master device demand. Therefore, the master device can access I/O data of the slave devices by one of the four I/O connection methods. The API functions are not only easy to use but also providing a lot of the DeviceNet Master functions to retrieve and deliver the slave’s I/O data. For more information, please refer to functions description and demo programs in section 4.

1.6 DNM100 Series Board Firmware Characteristics

The PISO-DNM100U and PEX-DNM100 is a high-performance DeviceNet master board. The firmware inside the board implements DeviceNet protocol automatically when the board is active. The firmware always listens to the bus and receives the message at the same time. It works as shown in Figure 1.6.

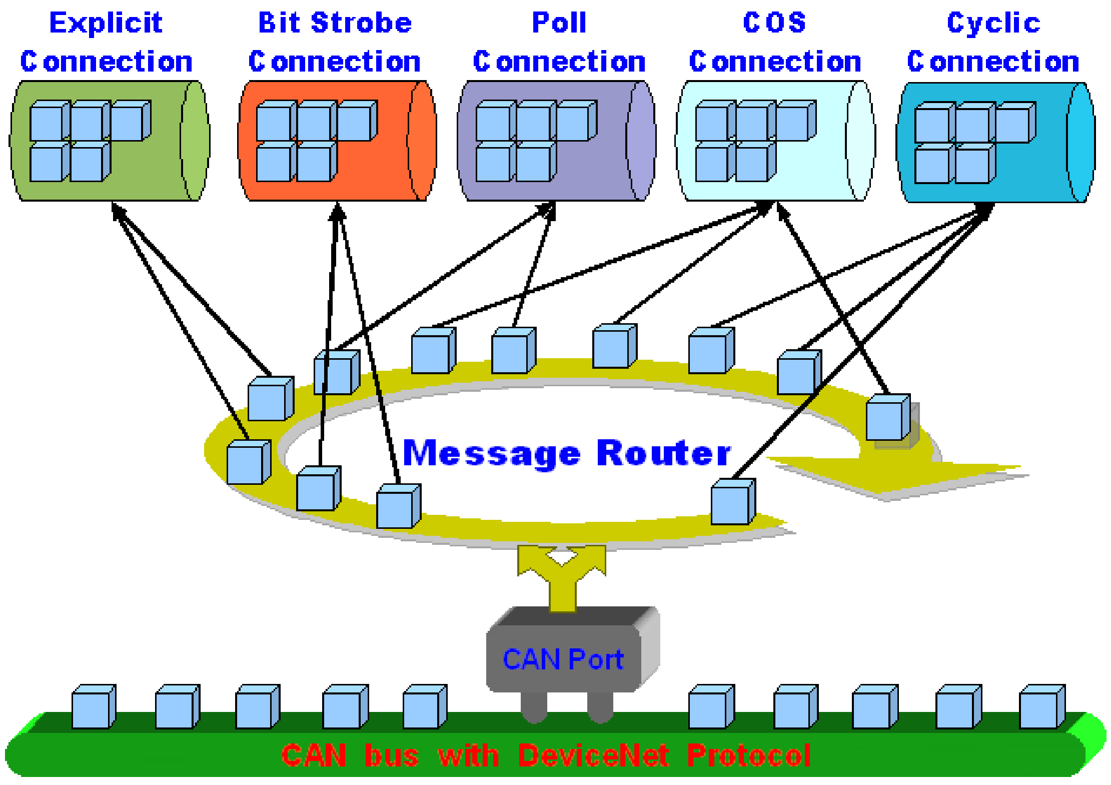


Figure 1.6 Message Router

The PISO-DNM100U and PEX-DNM100 firmware has a “ScanList” to store the remote slave devices information. After power off, the information still exists in the EEPROM. When the users turn on the PC next time, the “ScanList” will be loaded from EEPROM. The users can easily use the DLL functions to configure it, including adding devices or removing devices. It works as shown in Figure 1.7. There is more information about the library functions in chapter 4.

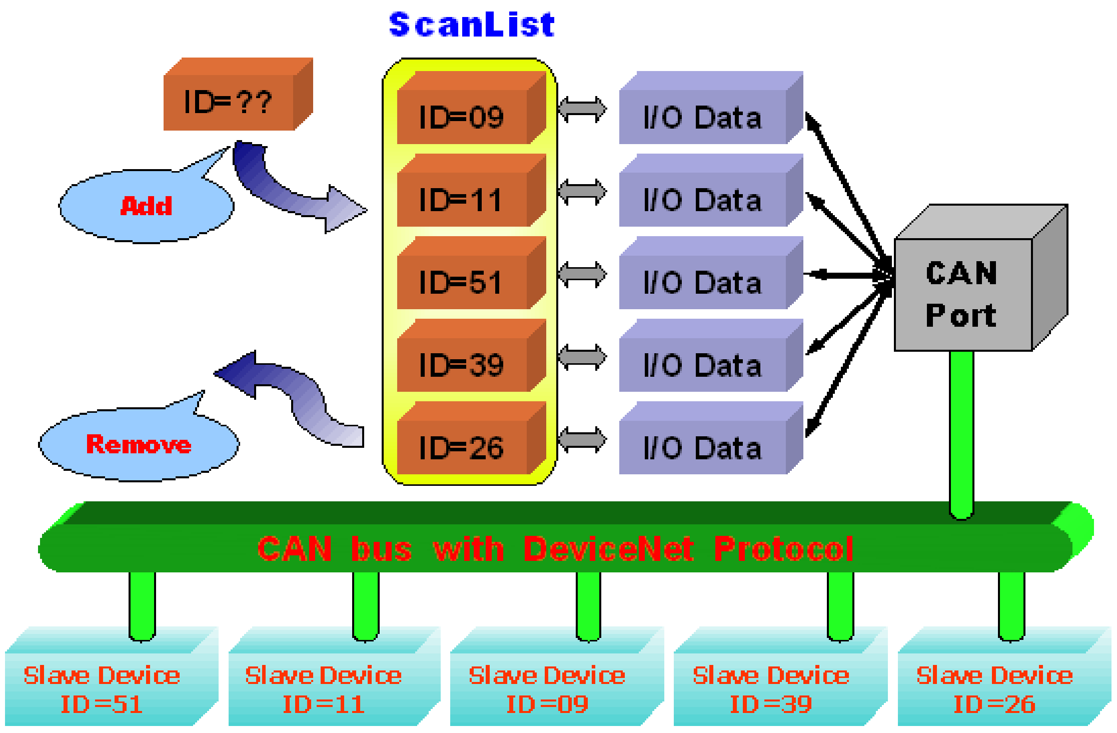


Figure 1.7 ScanList data structure

1.7 Features

Hardware Features

- PEX-DNM100-D/T.

  ＊ 2.5GHz PCI Express x1 plug and play technology.

- PISO-DNM100U-D/T

  ＊ Universal PCI card, supports both 5V and 3.3V PCI bus.

- Driver supported for Windows 7/8/10/11.
- High performance CPU with DeviceNet firmware inside.
- 8K bytes DPRAM inside.
- One CAN communication port.
- Compatible with CAN specification 2.0 parts A and B.
- Jumper select 120Ω terminator resistor for each port.
- 2 indicating LED (one for green and another for red).
- Direct memory mapping to the CAN controllers.
- 2500Vrms photo-isolation protection on CAN bus.
- 3000Vrms galvanic DC/DC isolation on CAN side.

DeviceNet Firmware Features

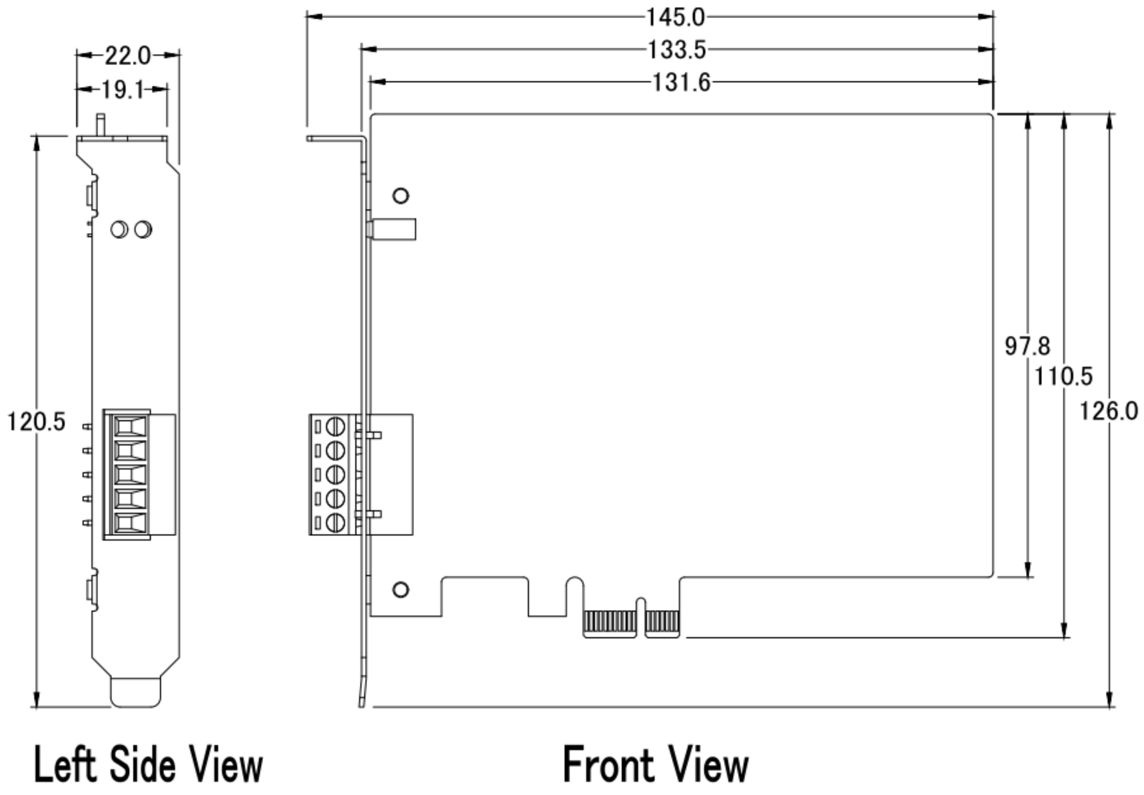
- Programmable Master MAC ID.
- Programmable transfer-rate 125K, 250K, 500K.
- Each port support maximum 63 slave nodes.
- Support Group 2 and UCMM connection
- I/O Length: 512 Bytes max (Input/Output) per slave
- Support I/O Operation: Poll, Bit-Strobe and Change Of State/Cyclic
- Support Auto-Search slave device function.
- Support Auto-detect Group 2 and UCMM device.
- Support Auto-Reconnect when the connection is broken.

1.8 Specifications

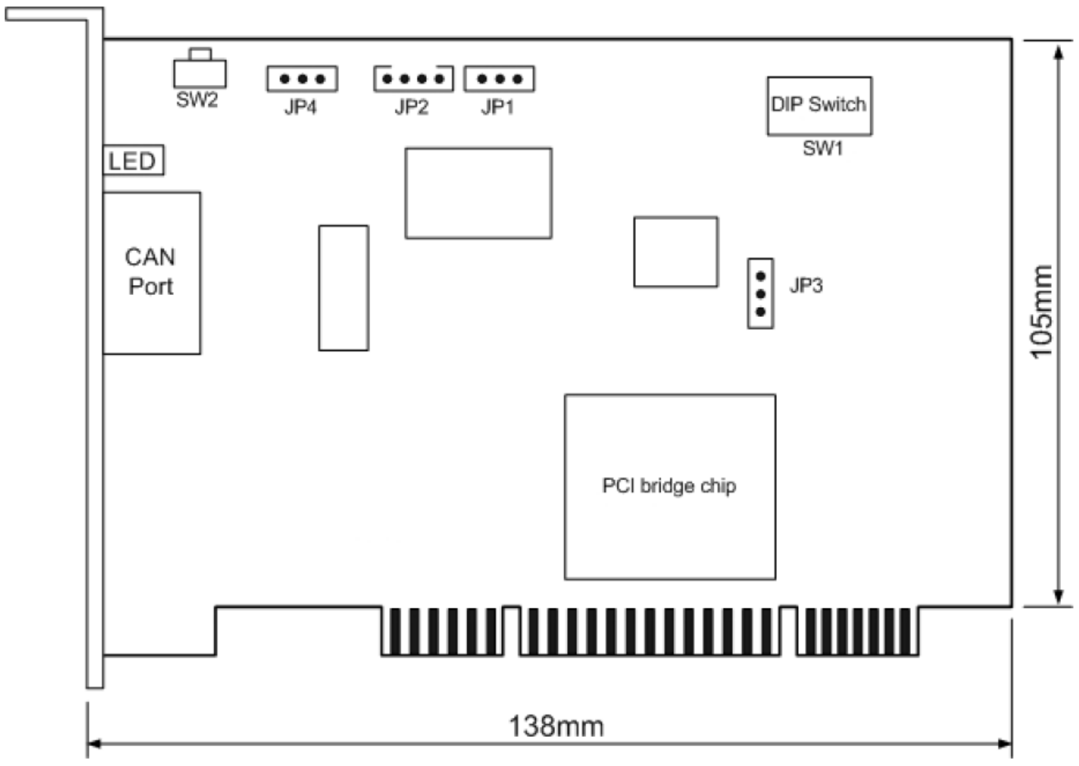
- CAN controller: NXP SJA1000T.
- CAN transceiver: NXP CAN series transceiver.
- Signal support: CAN_H, CAN_L.
- CAN controller frequency :16 MHz
- Connector: 5-pin screw terminal connector or 9-pin D-sub male connector.
- 2500Vrms photo-isolation protection on CAN bus.
- 3000Vrms galvanic DC/DC isolation on CAN side.
- Built-in high performance CPU
- 8K bytes DPRAM (1K bytes for system)
- 512 K bytes Flash memory (128K bytes for system)
- 512K bytes SRAM
- 2K EEPROM (256 bytes for system)
- 31 bytes NVRAM
- Power requirements:
  5V@400mA
- Environmental:

  Operating temp: -25~75℃
  Storage temp: -40~80℃
  Humidity: 5~95% non-condensing

1.9 Dimension



PEX-DNM100



PISO-DNM100U

1.10 Block Diagram

The figure 1.8 shows the block diagram of the PISO-DNM100U and PEX-DNM100 board.

1. DPRAM (Dual Port RAM) :
   The DPRAM is the memory buffer which provides the communication channel between PC and PISO-DNM100U or PEX-DNM100.

2. EEPROM :
   The EEPROM stores the configuration information. After restarting the PC, the configuration data will be loaded form the EEPROM automatically.

3. Control CPU :
   The CPU inside implementing the DeviceNet firmware.

4. CAN Controller :
   The CAN controller is used for sending and receiving the CAN messages. There is photo isolation between CAN controller and CAN bus.

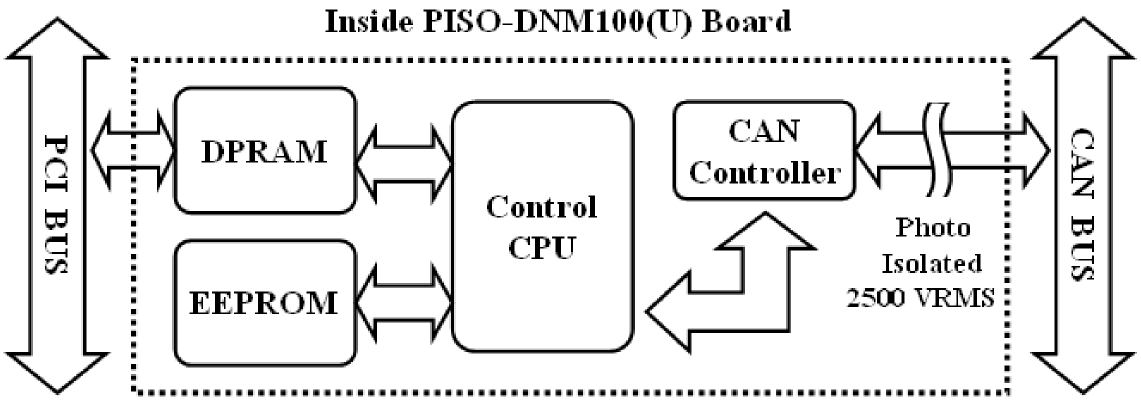


Figure 1.8 Block diagram of the PISO-DNM100U and PEX-DNM100

1.11 Product Check List

In addition to this manual, the package includes the following items:

- □ PISO-DNM100U or PEX-DNM100 board;
- □ Software CD ROM;
- □ Quick Start manual;
- □ Release Note

It is recommended that users should read the release note first. All of the important information needed will be provided in the release note as follows:

- □ Where you can find the software driver, utility and demo programs.
- □ How to install software & utility.
- □ Where is the diagnostic program?
- □ FAQ’s and answers.

Attention !

If any of these items are missing or damaged, please contact your local field agent. Keep aside the shipping materials and carton in case you want to ship or store the product in the future.

2. Hardware Configuration

This section will describe the hardware settings of the PISO-DNM100U and PEX-DNM100. This information includes the wire connection and terminal resistance configuration for the CAN network.

2.1 Board Layout

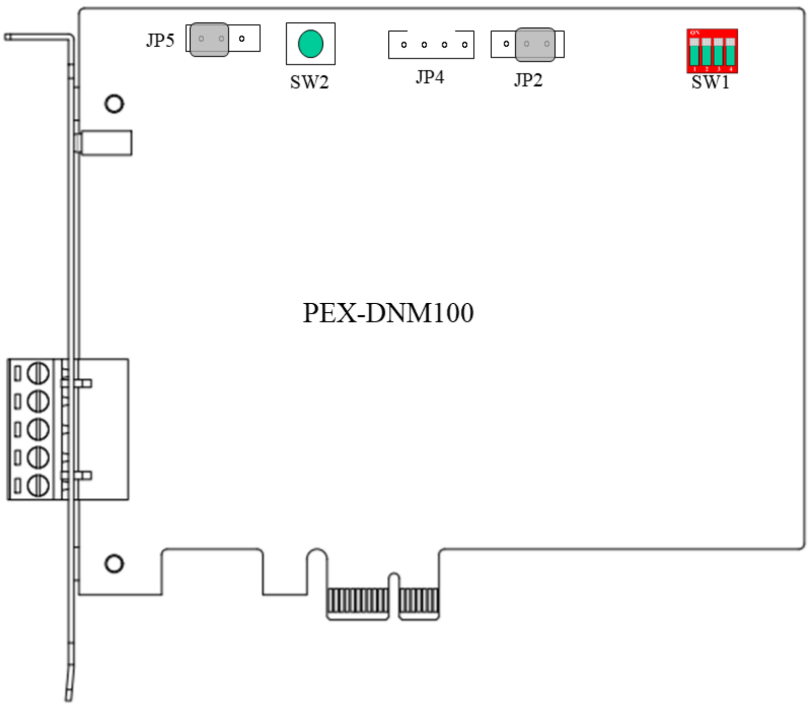


Figure2.1 PEX-DNM100 Board Layout

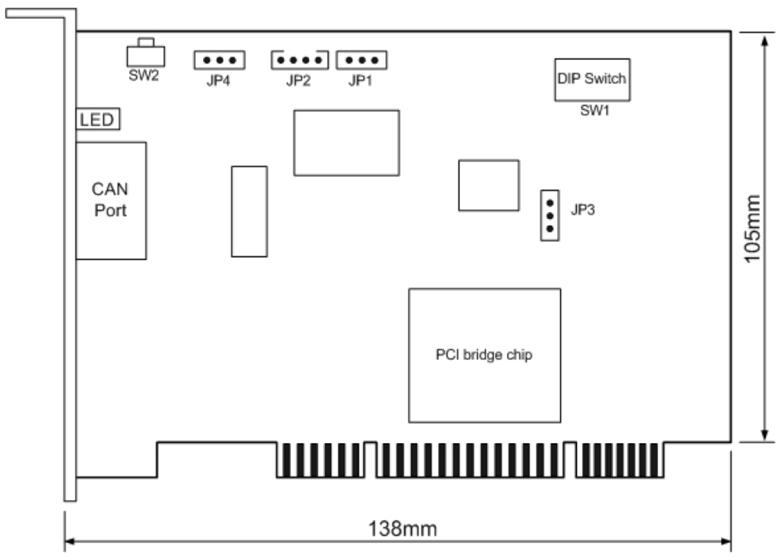


Figure 2.2 PISO-DNM100U Board Layout

Note: The PISO-DNM100U-T layout is similar with the PISO-DNM100U-D. The PEX-DNM100-T layout is similar with the PEX-DNM100-T. The only difference is the position of CAN port connector. The positions of jumper or DIP switch are the same. Therefore, users can also refer to the XXX-DNM100-**D** layout to configure the jumper or DIP switch if they use XXX-DNM100-**T**.

2.2 Jumper Selection

The following table shows the definition of jumpers or DIP switch. Users need to refer to this table to configure the PISO-DNM100U-D/T and PEX-DNM100 hardware.

PEX-DNM100-D / PEX-DNM100-T :

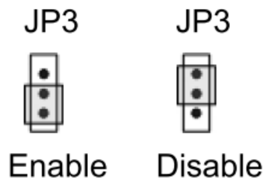
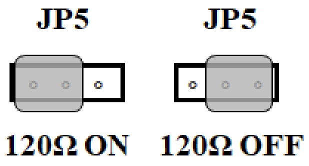
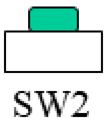
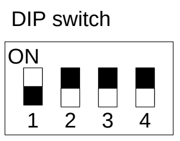
Jumper	Description	Status
JP3	The Flash protection jumper. If users would like to protect the data of the Flash. Enable this jumper. In the “Enable” status, the firmware can’t be updated from the utility.	
JP5	CAN Port 120Ω terminal resistance.	
SW2	The reset button for into the download mode. If users would like force the CAN board into download mode, enable this jumper to reset the firmware and let the PEX-DNM100 into the download mode.	
DIP switch (SW1)	DIP switch is used to set the PEX-DNM100 board No. The “1” is for bit0, the “2” is for bit1 and so on. For example, if the “1” of the switch is ON, the board No. is set to 1. The range of board No. is from 0 to 15. Be careful the following board No. for each board must be unique. ＊ PISO-CM100U-D/T, ＊ PEX-DNM100-D/T, ＊ PISO-DNM100U-D/T ＊ PISO-CPM100U-D/T	 This situation indicates the board No. 1.

Table 2.1 Jumper or DIP switch selections for PEX-DNM100

PISO-DNM100U-D / PISO-DNM100U-T :

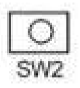
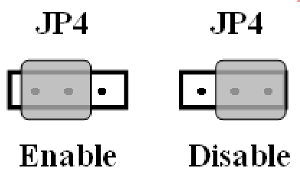
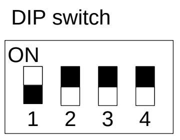
Jumper	Description	Status
JP1 JP2 JP3	None.	None.
SW2	Reset button for download error. If users want to update firmware but the process is fail, users can click this button to reset the PISO-DNM100U into download mode.	
JP4	CAN Port 120Ω terminal resistance.	
DIP switch	DIP switch is used to set the PISO-DNM100U board No. Switch1 is for bit0, switch2 is for bit1 and so forth. For example, if the left-hand-side switch (switch 1) is ON, the board No. is set to 1. The range of board No. is from 0 to 15. Be careful that the flolowing board No. for each board must be unique. ＊ PISO-CM100U-D/T, ＊ PEX-DNM100-D/T, ＊ PISO-DNM100U-D/T ＊ PISO-CPM100U-D/T	 This situation indicates the board No. 1.

Table 2.2 Jumper or DIP switch selections for PISO-DNM100U

2.3 Connector Pin Assignment

The PISO-DNM100U-T or PEX-DNM100-T is equipped with one **5-pin screw terminal connector** and the PISO-DNM100U-D or PEX-DNM100-D is equipped with one **9-pin D-sub male connector** for wire connection of the CAN bus. The connector's pin assignment is specified as follows:

2.3.1 5-pin screw terminal connector

The 5-pin screw terminal connector for the CAN bus is shown in Figure 2.4 and the details for the pin assignment are presented in Table 2.2.

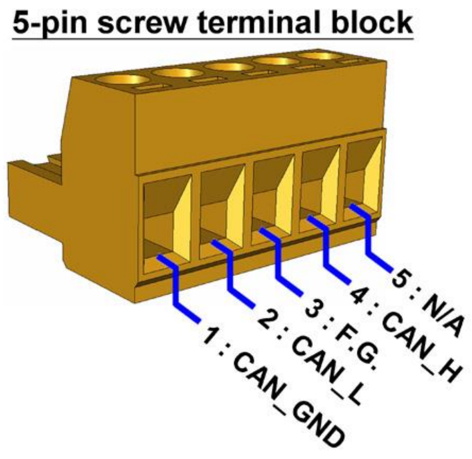


Figure2.4 5-pin screw terminal connector

Pin No.	Signal	Description
1	CAN_GND	CAN Ground
2	CAN_L	CAN_L bus line (dominant low)
3	F.G	Optional Frame Ground
4	CAN_H	CAN_H bus line (dominant high)
5	N/A	No use

Table 2.2: Pin assignment of 5-pin screw terminal connector

2.3.2 9-pin D-sub male connector

The 9-pin D-sub male connector of the CAN bus interface is shown in Figure 2.5 and the corresponding pin assignments are given in Table 2.3.

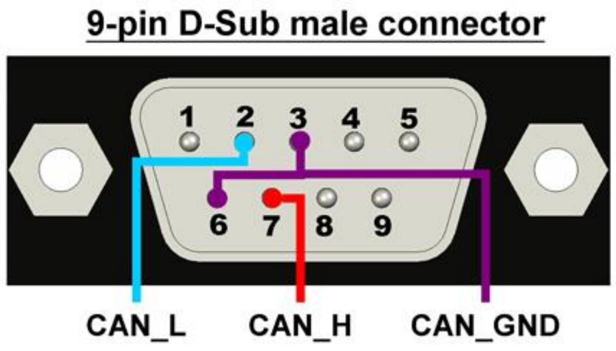


Figure2.5 9-pin D-sub male connector

Pin No.	Signal	Description
1	N/A	No use
2	CAN_L	CAN_L bus line (dominant low)
3	CAN_GND	CAN Ground
4	N/A	No use
5	N/A	No use
6	CAN_GND	CAN Ground
7	CAN_H	CAN_H bus line (dominant high)
8	N/A	No use
9	N/A	No use

Table 2.3 Pin assignment of the 9-pin D-sub male connector

2.3.3 Wire connection

In order to minimize the reflection effects on the CAN bus line, the CAN bus line has to be terminated at both ends by two terminal resistances as in the following figure. According to the ISO 11898-2 spec, each terminal resistance is 120Ω (or between 108Ω~132Ω). The length related resistance should have 70 mΩ/m. Users should check the resistances of the CAN bus, before they install a new CAN network.

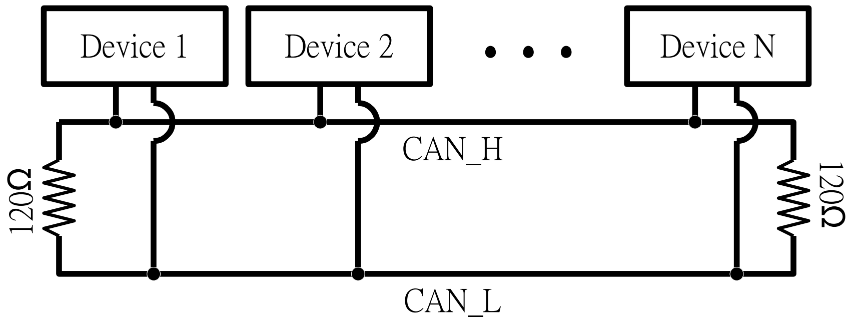


Figure 2.4 CAN bus network topology

Moreover, to minimize the voltage drop over long distances, the terminal resistance should be higher than the value defined in the ISO 11898-2. The following table can be used as a good reference.

Bus Length (meter)	Bus Cable Parameters		Terminal Resistance (Ω)
	Length Related Resistance (mΩ/m)	Cross Section (Type)	
0~40	70	0.25(23AWG)~ 0.34mm^2(22AWG)	124 (0.1%)
40~300	< 60	0.34(22AWG)~ 0.6mm^2(20AWG)	127 (0.1%)
300~600	< 40	0.5~0.6mm^2 (20AWG)	150~300
600~1K	< 20	0.75~0.8mm^2 (18AWG)	150~300

Table 2.4 Relationship between cable characteristics and terminal resistance

2.4 Indicator LED

2.4.1 Green LED

The [Green] LED indicates the firmware status in the PISO-DNM100U and PEX-DNM100. There are 3 situations in [Green] LED.

(1). LED off:

This indicates that there are some errors on the bus or in the firmware. The DeviceNet firmware is not running.

(2). LED twinkle:

This indicates that the CAN bus works fine. But there is no any slave devices configuration in the EEPROM of the PISO-DNM100U and PEX-DNM100. The DeviceNet firmware is waiting for configuration.

(3). LED on:

This indicates that the DeviceNet firmware is running. The PISO-DNM100U and PEX-DNM100 is communicating with the slave devices.

2.4.2 Red LED

The [Red] LED means Network Status. It indicates that there are errors on the bus or there is any slave device's MAC ID collides with the DNM100's MAC ID. There are two situations in [Red] LED.

(1). LED off:

This indicates that there is no error on the bus and about the MAC ID.

(2). LED twinkle:

This indicates that there are errors on the bus which maybe the situations as shown bellow:

(a) The CAN connector doesn't connect to the slave devices.
(b) The power of the slave devices is off.
(c) The MAC ID collision between master and slave devices is occurring.

2.5 Hardware Installation

When users want to use the PISO-DNM100U-D/T or PEX-DNM100-D/T, the hardware installation needs to be finished as following steps.

1. Shutdown your personal computer.
2. Configure the DIP switch and JP4 of your PISO-DNM100U-D/T for board No. and terminal resistance. Configure the DIP switch and JP5 of your PEX-DNM100-D/T for board No. and terminal resistance.More detail information could be found on the Figure 2.1/2.2 and Table 2.1/2.2.
3. Check JP3 status of the PISO-DNM100U-D/T and PEX-DNM100. If necessary, enable it to protect the firmware.
4. Find an empty PCI slot for your PISO-DNM100U-D/T or an empty PCI express x1 slot for your PEX-DNM100-D/T on the motherboard of the personal computer. Plug the configured PISO-DNM100U-D/T or PEX-DNM100 into this empty slot. See Figure 2.5 or 2.6.
5. Plug your CAN bus cable(s) into the 5-pin screw terminal connector or the 9-pin D-sub connector.

When the steps described above is completed, turn on the personal computer.

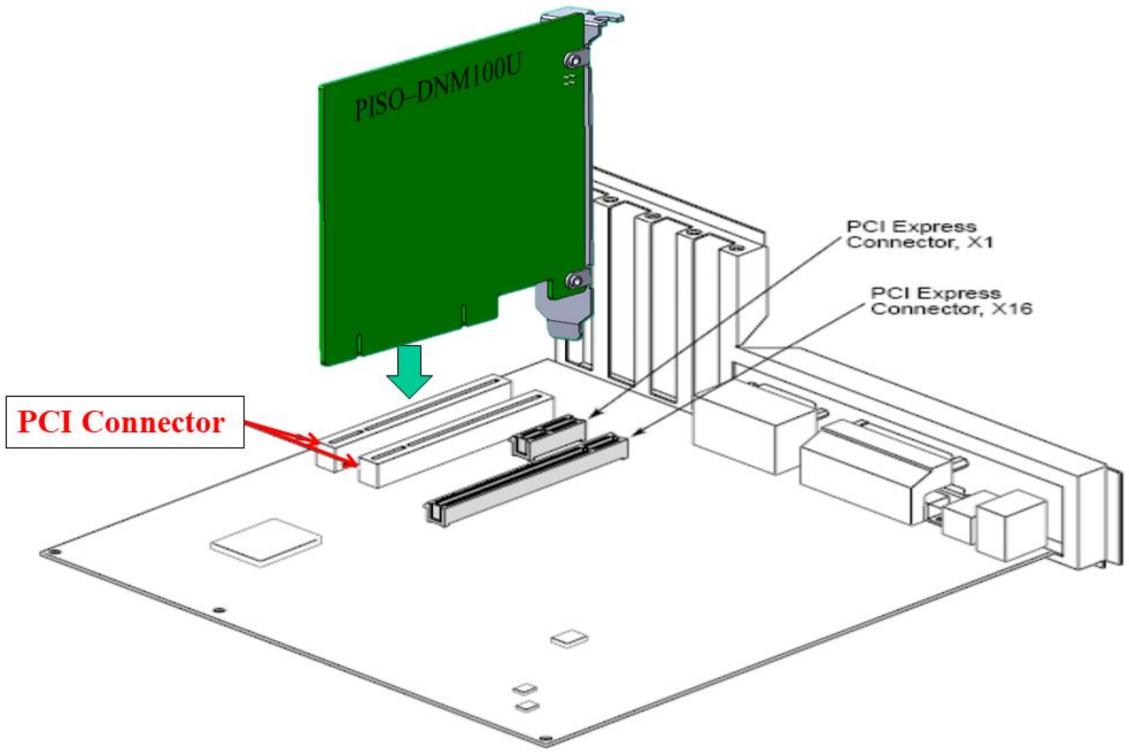


Figure 2.5 PISO-DNM100U installation

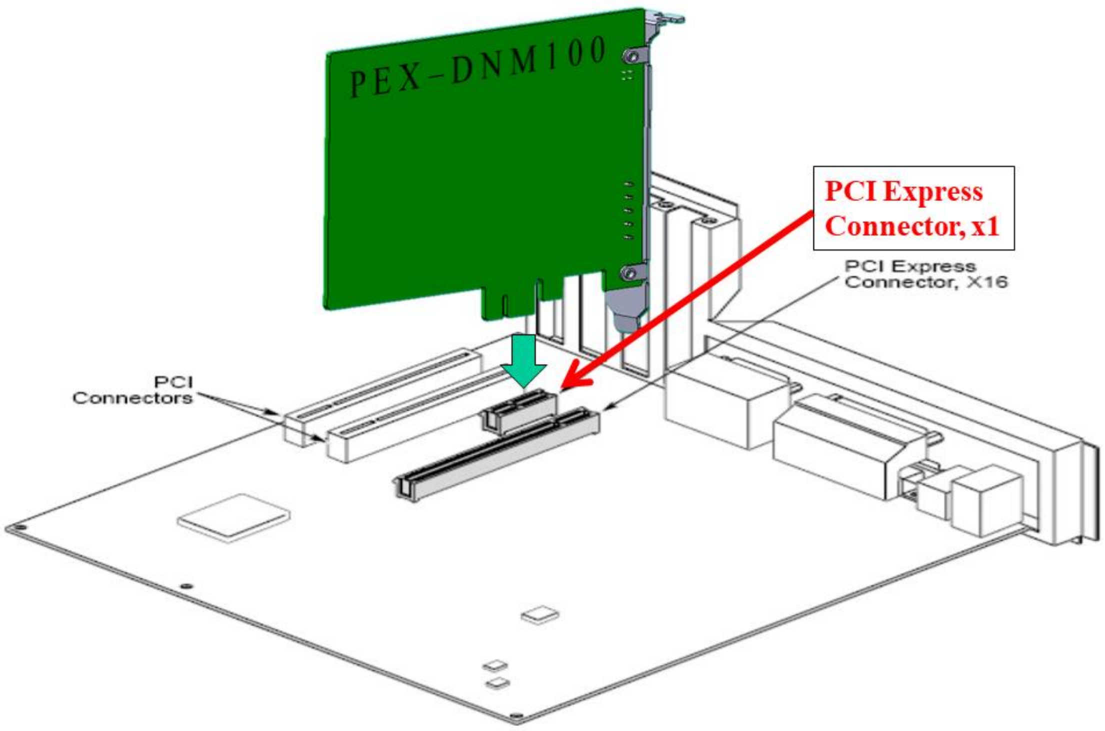


Figure 2.6　PEX-DNM100 installation

3. Driver Installation and Software Application

The DeviceNet DLL driver (DNM100.dll) collection of function calls for the PISO-DNM100U and PEX-DNM100 board used in Windows systems. The application structure is presented in the following figure. The user's DeviceNet application programs can be developed by the following designated tools: VB, Delphi and Borland C++ Builder…etc. In these tools, the application program can call the DNM100.DLL driver to implement DeviceNet network application. And then the DeviceNet DLL driver will throughout the CM100.dll into the KP_CM100.sys to access the hardware system, as shown in the following Figure.

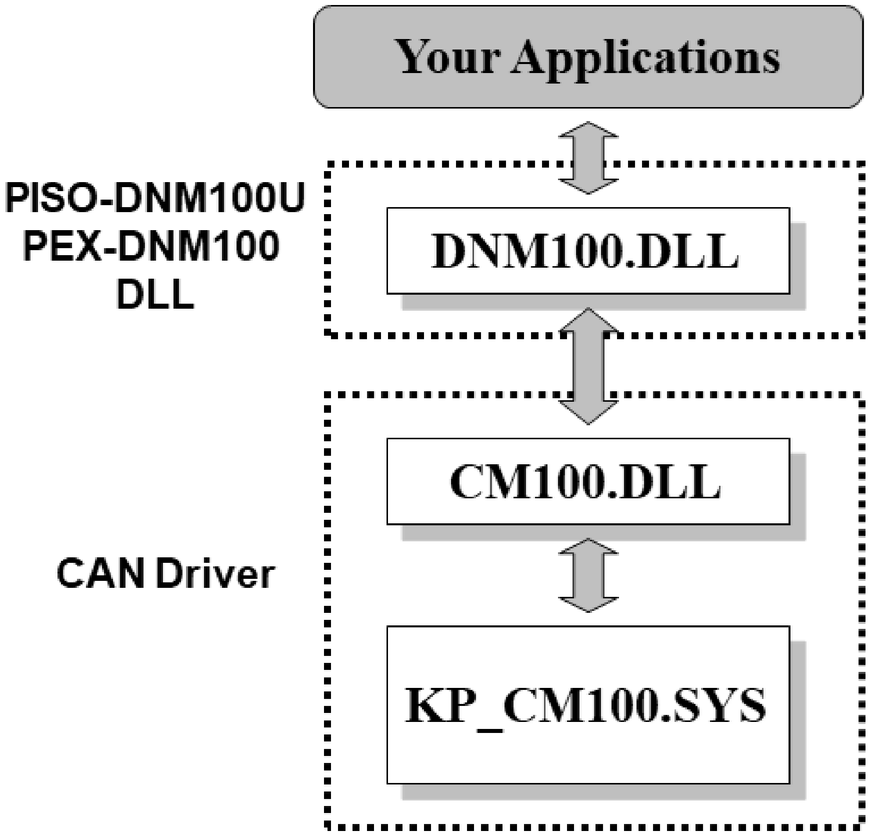


Figure 3.1 Software architecture in the Windows system

In the following subsection, we show some flow diagrams to describe how to apply the DeviceNet protocol (DNM100.DLL) to build a master device. Section 3.2 ~ 3.10 show the flow diagram for users to understand easily. Note that users need to follow the operation principle of the DeviceNet protocol correctly and easily to communicate with the remote nodes by these connection methods.

3.1 Driver Installation of the DNM100 Series Boards

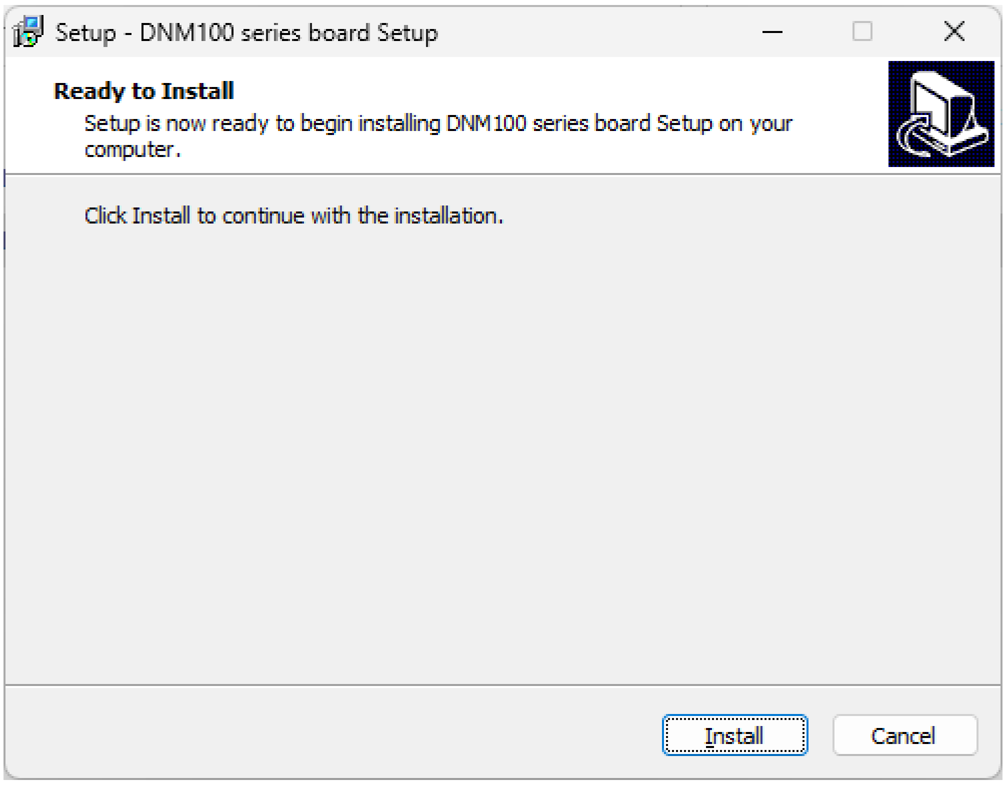
The software Installation for DeviceNet application is demonstrated as the following descriptions. After finishing the procedure, the driver, demos, manual and utility can be in your PC. For the advance application, users can refer to the basic demo programs to develop the customized DeviceNet master application.

The PEX-DNM100 and PISO-DNM100 use the same driver in the Windows environments. For these Windows operation systems, the recommended installation procedure is given as follows:

Step 1: Insert the companion CD into the CD-ROM driver or download setup file from the website of the PISO-DNM100U or PEX-DNM100. You will find the setup file which named “DNM100 board Setup for Winxp_7_8_10 vxxx.exe”

DNM100_Boards_Win_7_8_10_V100.exe

Step 2: After clicking the file, the setup process will start.

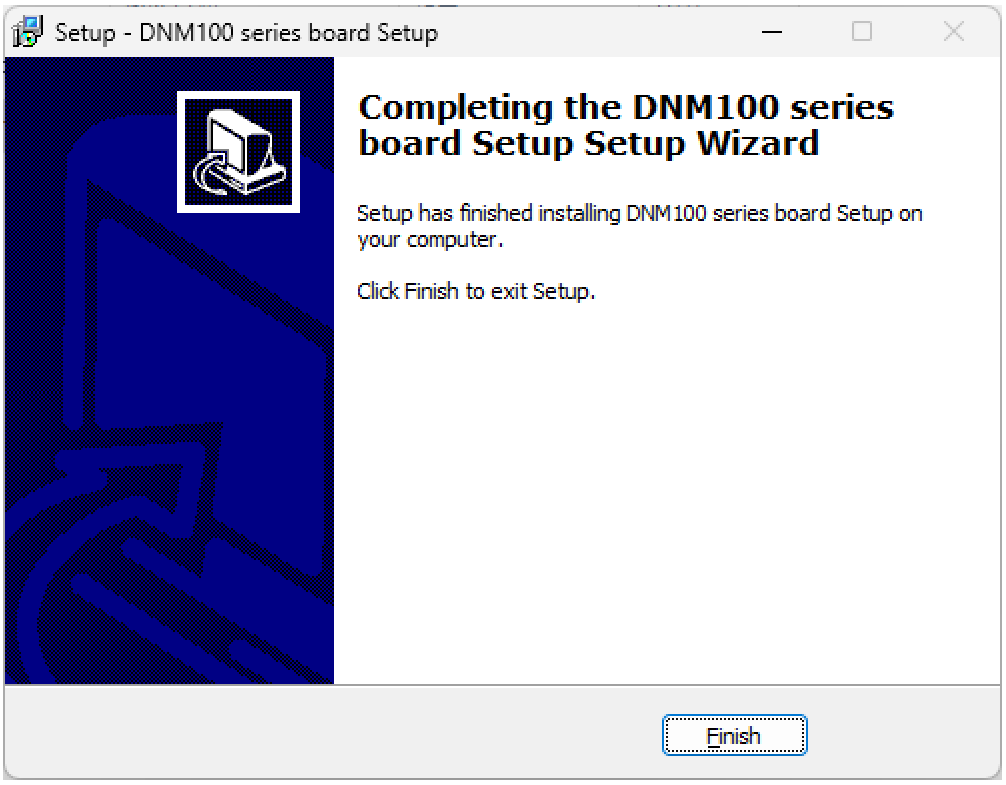

Setup - DNM100 series board Setup

Ready to Install

Setup is now ready to begin installing DNM100 series board Setup on your computer.

Click Install to continue with the installation.

Step 3: After finish the process, the dialog will be the following picture.


Setup - DNM100 series board Setup

Completing the DNM100 series board Setup Setup Wizard

Setup has finished installing DNM100 series board Setup on your computer.

Click Finish to exit Setup.

Step 4: Please restart your PC to make the windows system ready.

Then the setup software would copy the related material to the indicated directory and register the driver on your computer. The driver target directory is different according to the different systems as follows.

Windows 32-bit version – **WINDOWS\SysWOW64\DRIVERS**

Windows 64-bit version – **WINDOWS\SYSTEM32\DRIVERS**

The other data and resource is copied to the following directory:

C:\ICPDAS\DNM100_Boards\

Note：DeviceNet Master Utility is a useful tool for users to configure and test the DeviceNet slave devices before developing user's application. You can find the software in the【Download Center】of the website below. https://www.icpdas.com/en/download/show.php?num=1962&model=PISO-DNM100U-D. After installing the software, the utility is installed in the path below.

C:\ICPDAS\DNM_Utility\DNM_Utility.exe

Please refer to the manual of utility to know the detail.

3.2 Flow Diagram for Searching Devices

Before developing the DeviceNet applications, users should diagnose the connection between the slave devices. First, the users can search the slave devices in the network by using the searching functions. If the connection between the master and other slave devices is fine, the uses can find the information of the corresponding slave devices. When the users have no idea to communicate with the slave devices, users can follow these steps shown in figure 3.2. The following functions can help users to get the DeviceNet information of the slave devices. The users can find out the problem of the slave devices by using these functions. The detail information about those functions is in the next chapter.

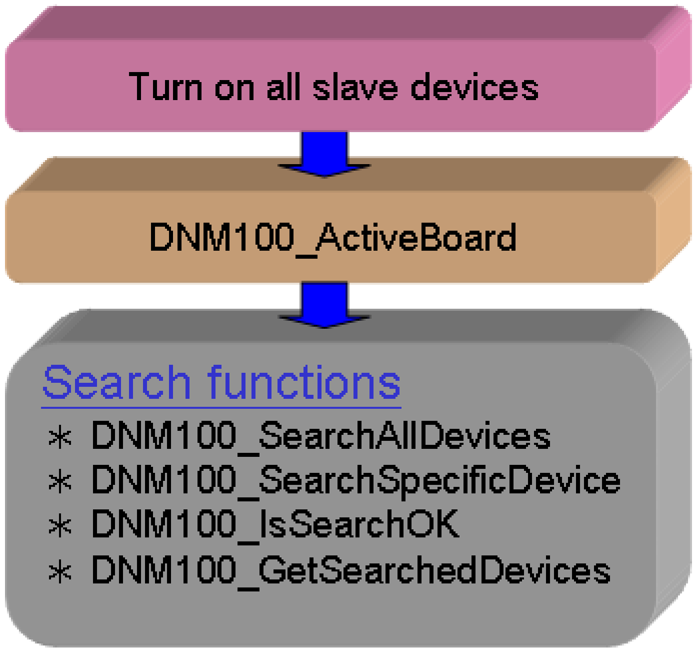

Turn on all slave devices
DNM100_ActiveBoard
Search functions
* DNM100_SearchAllDevices
* DNM100_SearchSpecificDevice
* DNM100_IsSearchOK
* DNM100_GetSearchedDevices

Figure 3.2 Searching Diagram

3.3 Flow Diagram for Slave Configuration

After getting the DeviceNet I/O information of the slave devices, users should save the parameters into the EEPROM in PISO-DNM100U or PEX-DNM100. The EEPROM will store the configuration data. The firmware in PISO-DNM100U or PEX-DNM100 will load the previous configuration from the EEPROM in the next boot-up. When the devices in the DeviceNet network are changed, the users must set the configuration data to fit the application. The configuration diagram is shown in Figure 3.3. There is more information about those functions in the next chapter.

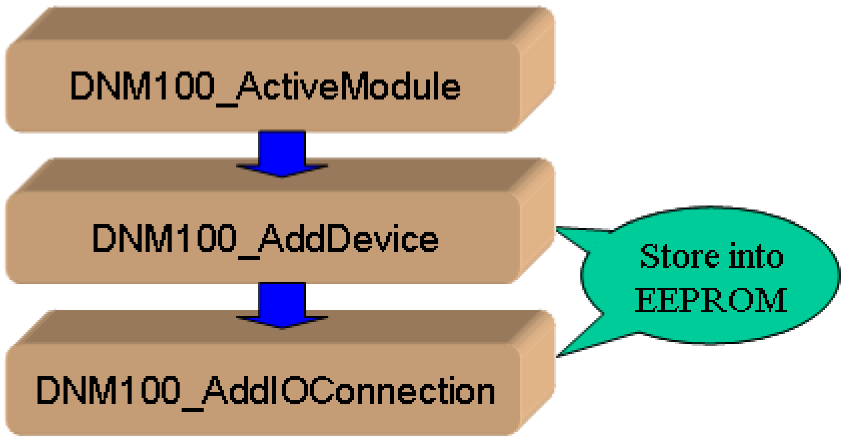


Figure 3.3 Slave Configuration Diagram

3.4 Flow Diagram for On-line Adding/Removing Device

The PISO-DNM100U or PEX-DNM100 provides the on-line adding/removing slave device functions. The users need not to break the communication between original slave devices when adding or removing the slave devices. The users can follow the steps to achieve this function. The steps are shown in Figure 3.6 and Figure 3.7.

1. On-line Adding Devices :

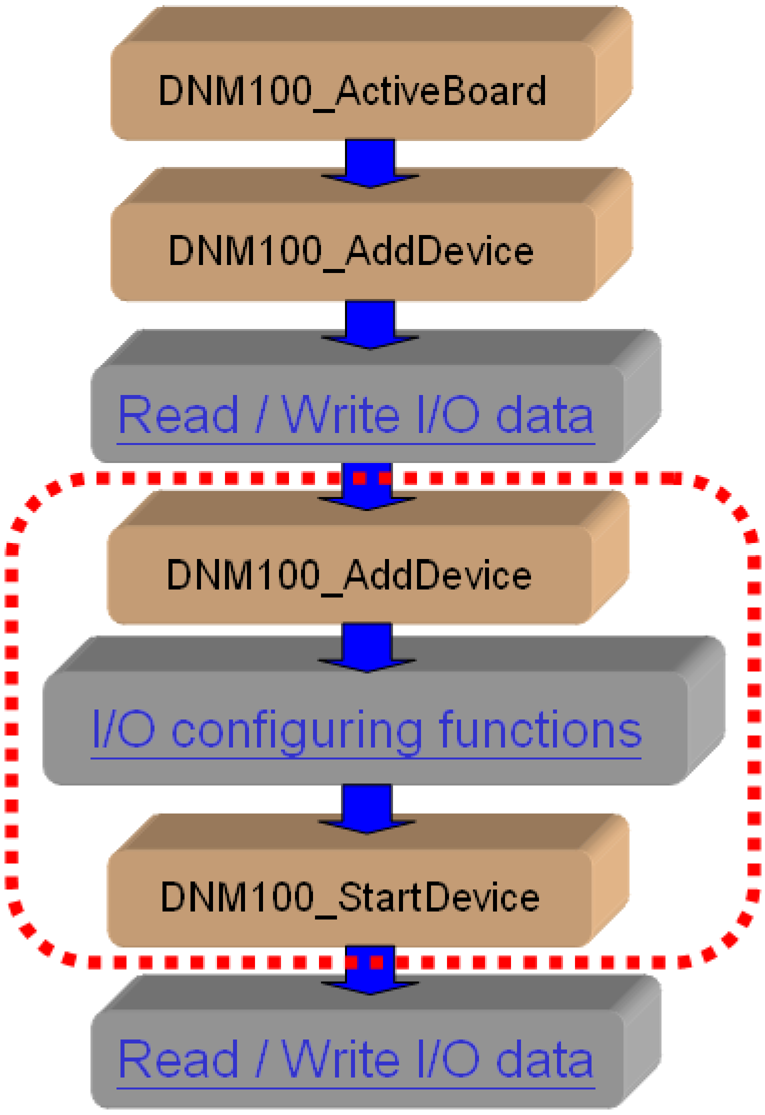


Figure 3.6 On-line Add Device Diagram

2. On-line Removing Devices :

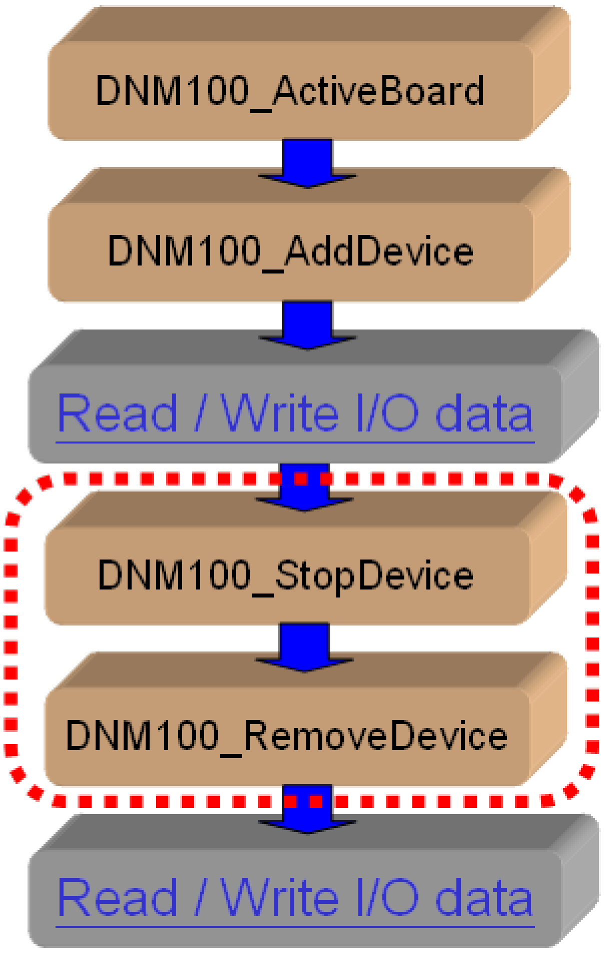


Figure 3.7 On-line Remove Device Diagram

3.5 Flow Diagram for “SetAttribute” and “GetAttribute”

The users can set or get DeviceNet device's property via DeviceNet network. The PISO-DNM100U or PEX-DNM100 provides these functions to set or get the properties of the remote devices easily. The steps are shown in Figure 3.8.

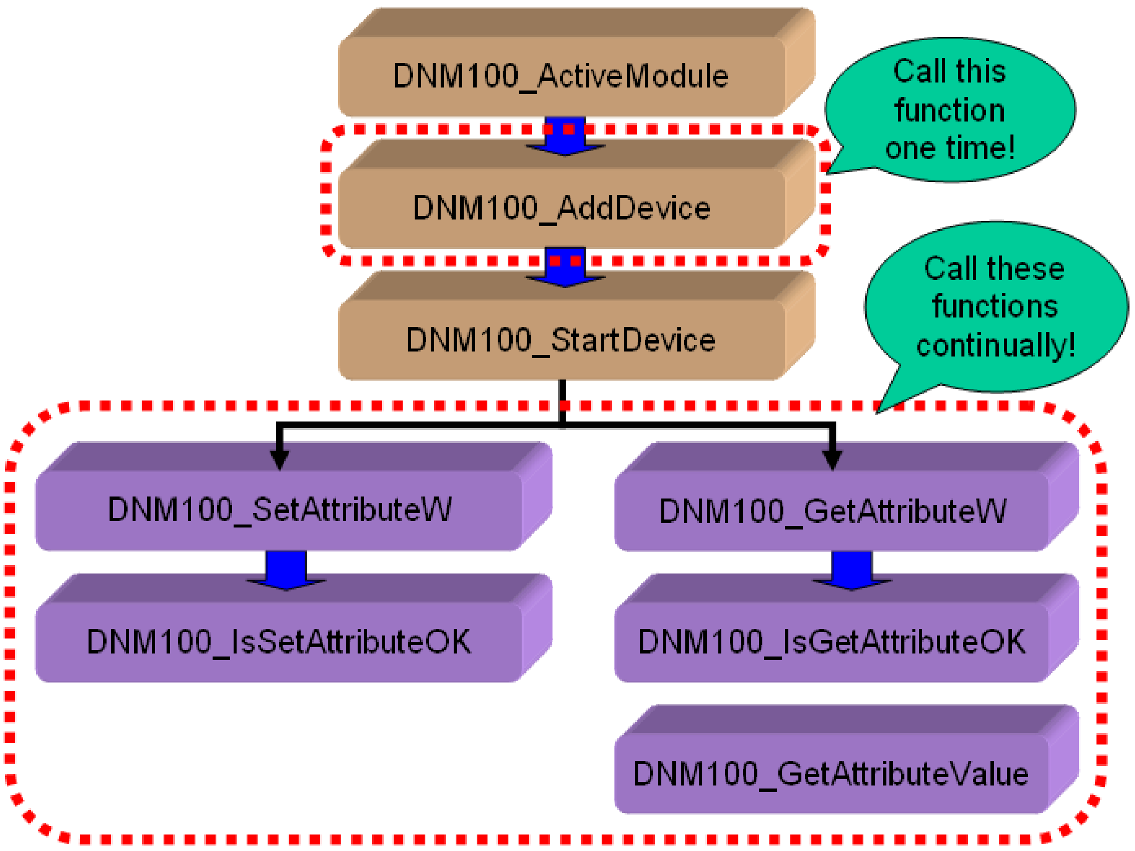


Figure 3.8 “SetAttribute” and “GetAttribute” Diagram

3.6 Flow Diagram for I/O Connection

The users can read or write device's I/O data via the DeviceNet I/O connections like Poll, Strobe, COS and Cyclic connection. There are four important steps to read and write the I/O data easily. Firstly, the users should know the device's I/O input length (in Byte) and output length (in Byte). Secondly, the users should set these two parameters by calling DNM100_AddIOConnection. Thirdly, the users can set the initial output value by calling DNM100_WriteOutputData before starting the specific slave device. If the users do not initialize the output value, the firmware default output value is 0. Fourthly, the users can start communicating with device to read or write I/O data. If the specific slave device doesn't have any output channel, the firmware will start communicating with the device automatically. The Figure 3.9 shows the main steps to achieve this function. There are more functions described in chapter 4.

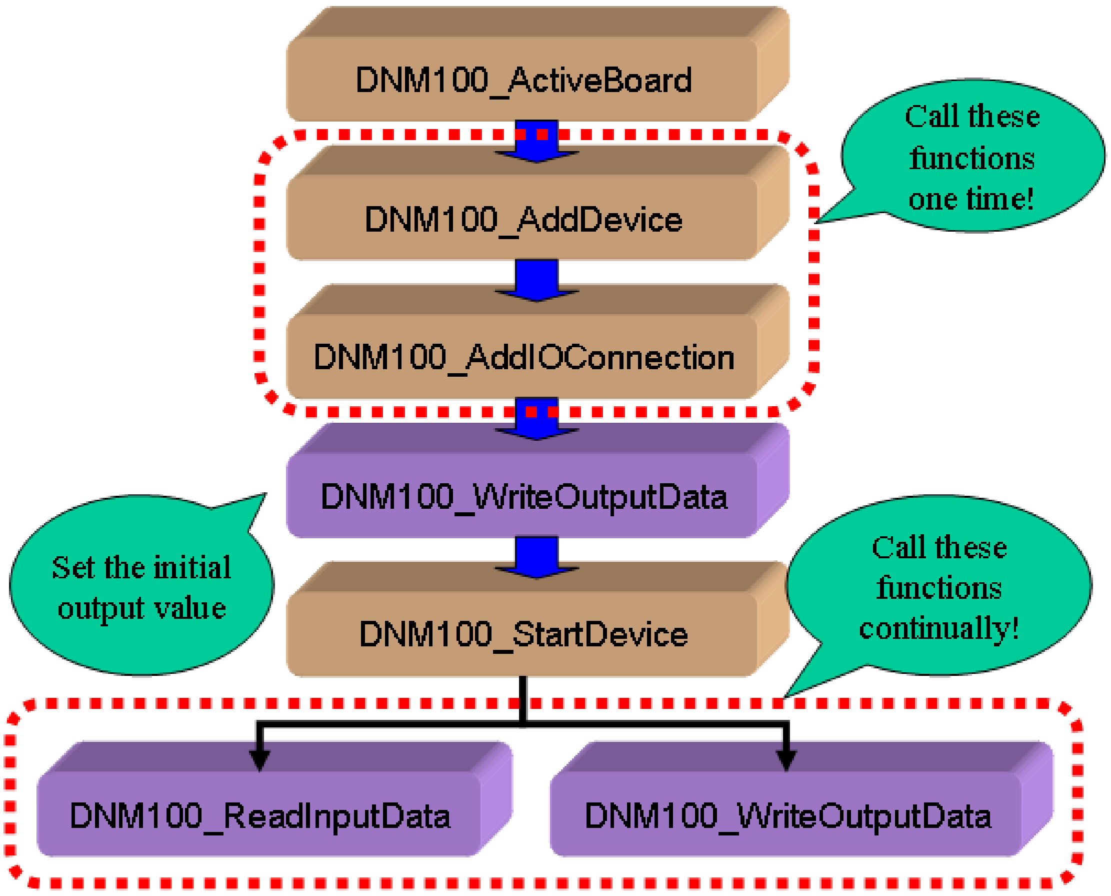


Figure 3.9 I/O Connection Diagram

Note: The Strobe connection doesn't support the output channel. The users can not use the DNM100_WriteOutputData with Strobe connection.

3.7 Input and Output I/O Data Area (Advanced Option)

Here exist two memory areas, “Remote Input Area” and “Remote Output Area”. The input data of all DeviceNet slaves would be stored in the “Remote Input Area”, and the output data of them would be in the “Remote Output Area”. Please refer to the Figure 3.10.

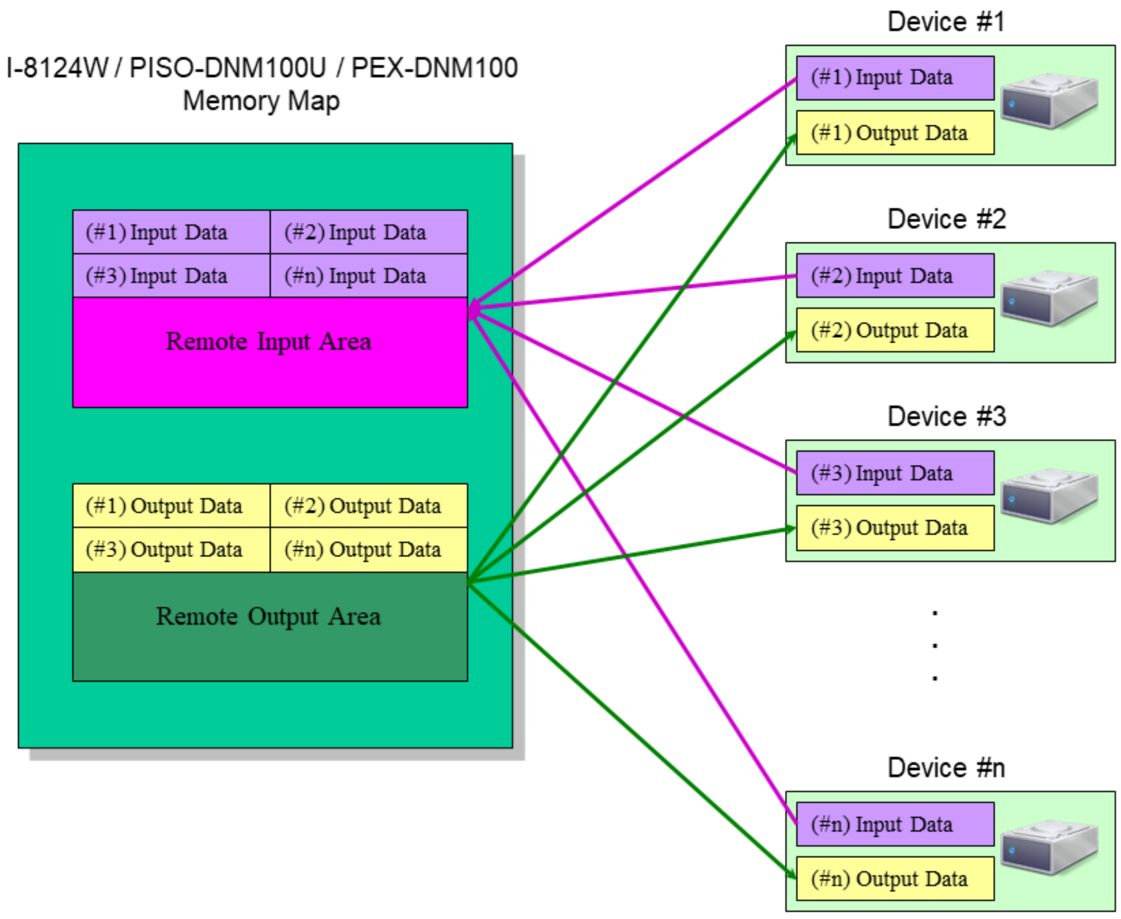


Figure 3.10 The Memory Mapping of the PISO-DNM100U or PEX-DNM100

Users can read a bulk data from “Remote Input Area” in the PISO-DNM100U or PEX-DNM100. This bulk data contains multiple devices’ input statuses. If one of the input status of the remote DeviceNet slave changes, the corresponding data located in the “Remote Input Area” would change immediately. Oppositely, the “Remote Output Area” contains multiple devices’ output data. Users may change the output value of a certain device by changing the corresponding data located in the “Remote Output Area”.

There is another important thing. Uses need to know what the arrangement of those data. The data of the slave with the smallest DeviceNet MAC ID would be located in the most front of the Remote Input/Output Area. The data of the salve with the following MAC ID would be located in the following section, and so on. The data of the salve with the largest MAC ID would be located in the last section of the Remote Input/Output Area. Here shows three examples about the arrangement rule of the PISO-DNM100U or PEX-DNM100.

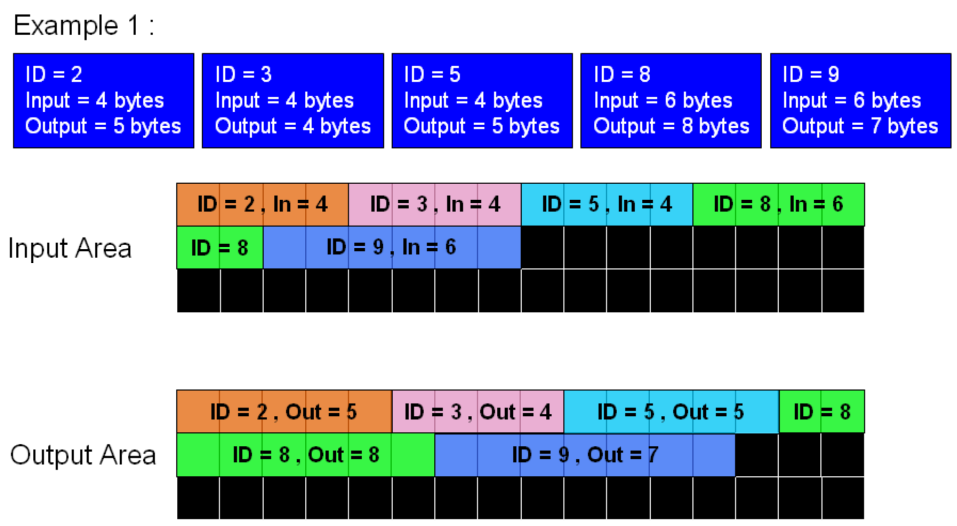
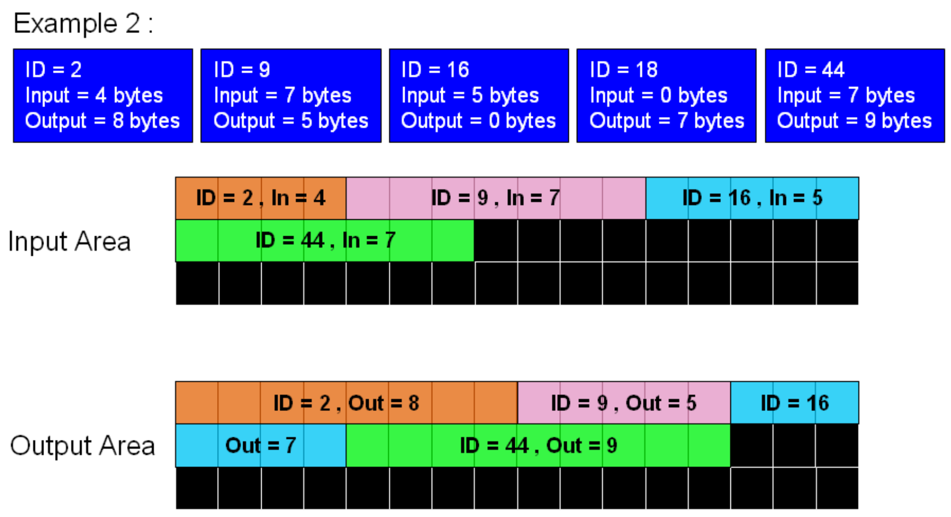


Figure 3.11 The example of the DNM100 borad memory mapping



Example 2 :
ID = 2 Input = 4 bytes Output = 8 bytes
ID = 9 Input = 7 bytes Output = 5 bytes
ID = 16 Input = 5 bytes Output = 0 bytes
ID = 18 Input = 0 bytes Output = 7 bytes
ID = 44 Input = 7 bytes Output = 9 bytes
Input Area
ID = 2 , In = 4
ID = 9 , In = 7
ID = 16 , In = 5
ID = 44 , In = 7
Output Area
ID = 2 , Out = 8
ID = 9 , Out = 5
ID = 16
Out = 7
ID = 44 , Out = 9

Figure 3.12 The example of the DNM100 board memory mapping

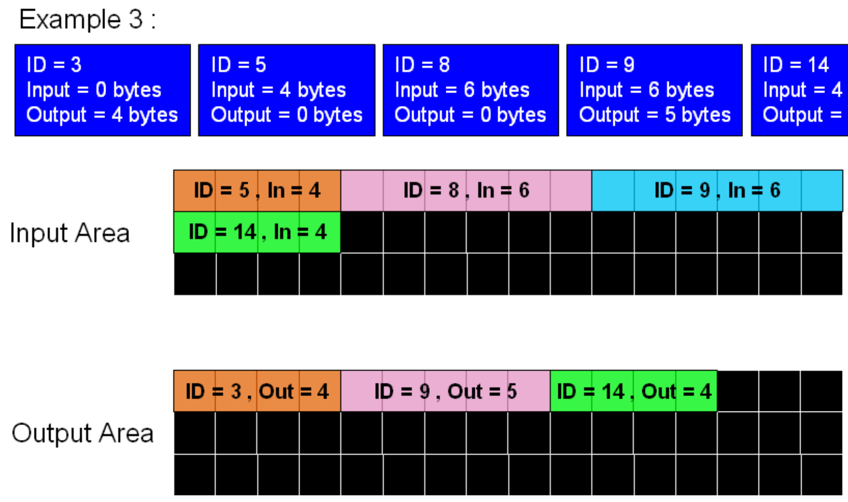


Figure 3.13 The example of the DNM100 board memory mapping

User can read data from the "Remote Input Area" or write data to the "Remote Output Area". There are three functions to access these two areas.

1.DNM100_ReadInputArea
Call this function to get a bulk data from the "Remote Input Area". Please refer to the section 4.3.41 for more information.

2.DNM100_WriteOutputArea
Call this function to set a bulk data to the "Remote Output Area". Please refer to the section 4.3.42 for more information.

3.DNM100_ReadbackOutputArea
Call this function to get a bulk data from "Remote Output Area". Please refer to the section 4.3.43 for more information.

Note:

If users add/remove any slaves' information into/from the PISO-DNM100U or PEX-DNM100, they need to reset firmware to update the modification of the data arrangement of the "Remote Input Area" and "Remote Output Area". In other word, if users have called these functions below, they need to call DNM100_ResetFirmware to make the modification active.

1. DNM100_AddDevice
2. DNM100_RemoveDevice
3. DNM100_AddIOConnection
4. DNM100_RemoveIOConnection

4. Function description

All the functions of the PISO-DNM100U or PEX-DNM100 can be separated into five groups. The idea is shown Figure 4.1. There is more detail description in CH 4.1.

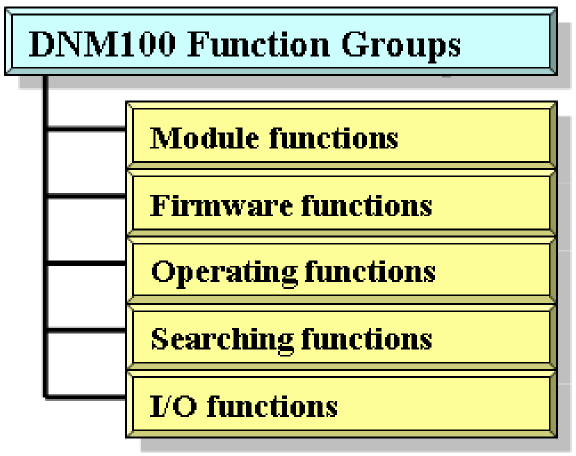


Figure 4.1 Five Function Groups

[Board Functions]

These functions in this group help users to find DNM100 boards or get board's information. The users can use these functions to configure or manage the boards in the PC.

[Firmware Functions]

These functions in this group help users to operate the firmware or get the status of the firmware inside the PISO-DNM100U or PEX-DNM100.

[Operating Functions]

These operating functions are the important operation of the DeviceNet master. They help users to configure the whole network.

[Searching Functions]

These searching functions can help user to debug the network, including the wire connection, the slave device's setting, and etc. When building the DeviceNet network, the user can use these functions to make sure that the network or the slave devices are fine.

[I/O Functions]

These functions help user to read or to write the I/O data from or to the remote slave devices.

4.1 DLL Function Definition and Description

All the functions provided in the DNM100.DLL are listed in the following table and detail information for every function is presented in the next sub-section. However, in order to make the descriptions more simply and clear, the attributes for the both the input and output parameter functions are given as **[input]** and **[output]** respectively, as shown in the following table.

Keyword	Set parameter by user before calling this function?	Get the data from this parameter after calling this function?
[input]	Yes	No
[output]	No	Yes

Table 4.1.1 Functions Table (Board Functions) 1/1

No.	Function Name	Description
1	DNM100_GetBoardInf	Get the driver information of the PISO-DNM100U or PEX-DNM100
2	DNM100_TotalDNM100Board	Get total PISO-DNM100U or PEX-DNM100 boards in the PC
2	DNM100_ActiveBoard	Enable the driver of the PISO-DNM100U or PEX-DNM100
3	DNM100_CloseBoard	Close driver of the PISO-DNM100U or PEX-DNM100
4	DNM100_GetDLLVersion	Get the DLL version of the DNM100.DLL

Table 4.1.2 Functions Table (Firmware Functions) 1/1

No.	Function Name	Description
1	DNM100_GetFirmwareVersion	Get the version of the firmware inside the PISO-DNM100U or PEX-DNM100 board
2	DNM100_ResetFirmware	Reset the firmware in the PISO-DNM100U or PEX-DNM100 board

Table 4.1.3 Functions Table (Operating Functions) 1/2

No.	Function Name	Description
1	DNM100_SetMasterMACID	Set the MAC ID of the DNM100 board (DeviceNet Master's MAC ID)
2	DNM100_GetMasterMACID	Get the MAC ID of the DNM100 board (DeviceNet Master's MAC ID)
3	DNM100_GetBaudRate	Get the baud rate of the CAN bus
4	DNM100_SetBaudRate	Set the baud rate of the CAN bus
5	DNM100_GetMasterStatus	Get the status of the DNM100 board (DeviceNet Master's status)
6	DNM100_GetSlaveStatus	Get the slave device's status.
7	DNM100_StartDevice	The DNM100 board will start to communicate with the specific slave device
8	DNM100_StopDevice	The DNM100 board will stop to communicate with the specific slave device
9	DNM100_StartAllDevice	The DNM100 board will start to communicate with all slave devices
10	DNM100_StopAllDevice	The DNM100 board will stop to communicate with all slave devices
11	DNM100_AddDevice	Add the specific slave device's information into the DNM100 board (DeviceNet Master)
12	DNM100_RemoveDevice	Remove the specific slave device's information from the DNM100 board (DeviceNet Master)
13	DNM100_AddIOConnection	Add I/O information of the specific slave device into the DNM100 board (DeviceNet Master)
14	DNM100_RemoveIOConnection	Remove specific slave device's I/O information from the DNM100 board (DeviceNet Master)

Table 4.1.4 Functions Table (Operating Functions) 2/2

No.	Function Name	Description
16	~~DNM100_GetAttribute~~	~~Send the get attribute command to the slave device.~~
17	DNM100_GetAttributeW	Send the get attribute command to the slave device.
18	DNM100_IsGetAttributeOK	Check whether the slave has replied for the getting command or not.
19	DNM100_GetAttributeValue	Get the attribute value of the DNM100_GetAttributeW
20	~~DNM100_SetAttribute~~	~~Send the set attribute command to the slave device.~~
21	DNM100_SetAttributeW	Send the set attribute command to the slave device.
22	DNM100_IsSetAttributeOK	Check whether the slave has replied for the setting command or not.
23	DNM100_GetDeviceInfoFromScanList	Get specific slave device's I/O information form the Scan List within the DNM100 board.
24	DNM100_GetScanList	Get the I/O information of all slave devices form the Scan List within the DNM100 board.
25	DNM100_ImportEEPROM	Write the I/O information of all slave devices into the EEPROM within the DNM100 board.
26	DNM100_ClearAllConfig	Clear all configurations in the EEPROM within the DNM100 board.
27	~~DNM100_SendExplicitMSG~~	~~Send the explicit request command.~~
28	DNM100_SendExplicitMSG_W	Send the explicit request command.
29	DNM100_IsExplicitMSGRespOK	Check whether the DNM100 board has received the response message or not.
30	DNM100_GetExplicitMSGRespValue	Get the attribute value of the specific device's instance.

Table 4.1.5 Functions Table (Searching Functions) 1/1

No.	Function Name	Description
1	DNM100_SearchAllDevices	The DNM100 board will search the DeviceNet network to find out the I/O information of all slave devices.
2	DNM100_SearchSpecificDevice	The DNM100 board will search the DeviceNet network to find out the I/O information of specific slave devices.
3	DNM100_IsSearchOK	Check whether the DNM100 board has searched completely or not.
4	DNM100_GetSearchedDevices	Get the result of the searching command and retrieve the slave’s I/O information.

Table 4.1.6 Functions Table (I/O Functions) 1/1

No.	Function Name	Description
1	DNM100_ReadInputData	Read the input data via I/O connection like Poll, Strobe, COS, Cyclic.
2	DNM100_WriteOutputData	Write the output data via I/O connection like Poll, COS, Cyclic. The Strobe doesn’t support this operation.
3	DNM100_ReadbackOutputData	Read back the output data via I/O connection like Poll, COS, Cyclic. The Strobe doesn’t support this operation.
4	DNM100_ReadInputArea	Read the bulk data from the input area memory.
5	DNM100_WriteOutputArea	Write the bulk data to the output area memory.
6	DNM100_ReadbackOutputArea	Read back the bulk data from the output area memory.

4.2 Function Return Code

Table 4.2.1 Interpretation of the return code (Hardware Error) 1/1

Return Code	Error ID	Comment
0	DNM100_NoError	No error
10001	DNM100_DriverError	Kernel driver is not opened.
10002	DNM100_ActiveBoardError	This board can not be activated.
10003	DNM100_BoardNumberError	The Board number exceeds the total board numbers.
10004	DNM100_PortNumberError	The Port number is not correct.
10007	DNM100_InitError	The DNM100 board replies error.
10021	DNM100_SoftBufferIsEmpty	No CAN messages in the buffer.
10022	DNM100_SoftBufferIsFull	The software buffer is overflow.
10023	DNM100_TimeOut	The DNM100 board has no response.
10024	DNM100_SetCyclicMsgFailure	The cyclic messages are over 5 counts. This is special function for CAN.
10025	DNM100_DpramOverRange	The command length is over 512 bytes.
10026	DNM100_NoDpramCmd	There is no command in DPRAM.
10027	DNM100_ModeError	This board can't be changed to firmware mode.
10030	DNM100_NoFileInside	There is no firmware in DNM100 board.
10031	DNM100_DownloadFailure	The download firmware process is failure.
10032	DNM100_EEPROMDamage	The EEPROM is out of order.
10033	DNM100_NotEnoughSpace	The firmware is too large to put it into the DNM100 board
10034	DNM100_StillDownloading	The firmware is downloading.
10035	DNM100_BoardModeError	The firmware mode is error.
10036	DNM100_CardTypeError	The firmware is not for the DNM100 board.

Table 4.2.2 Interpretation of the return code (General Error) 1/1

Return Code	Error ID	Comment
5000	DNMXS_UnKnowError	The DeviceNet has some unknown errors.
1000	DNMXS_BoardNotActive	The DNM100 board has not been activated.
1001	DNMXS_OnlineError	The master MAC ID collides with other slave device in the DeviceNet network.
1002	DNMXS_CANBusError	The CAN port can't send message. Please check the baud rate or the port of the CAN bus.
1003	DNMXS_Booting	The DNM100 board is still booting.
1050	DNMXS_MACIDError	The MAC ID is exceed the range(0 ~ 63)
1051	DNMXS_BaudRateError	The baud rate is exceed the range(0 ~ 2)
1052	DNMXS_ConnectionTypeError	The connection type is exceed the range (0 ~ 4)
1053	DNMXS_DuplicMasterMACID	The MAC ID is the same with the master's ID.
1054	DNMXS_EEPROMError	The EEPROM is out of order.
1055	DNMXS_NowScanning	The DNM100 board is searching the slave.
1056	DNMXS_ScanListError	The Scan List has some errors.
1057	DNMXS_DeviceExist	The information of the slave device already exists.
1058	DNMXS_DeviceNotExist	The information of the slave device doesn't exist.
1059	DNMXS_MapTableError	The MapTable has some errors.

Table 4.2.3 Interpretation of the return code (I/O Error) 1/1

Return Code	MapTable Error	Comment
1100	DNMXS_ExplicitNotAllocate	The Explicit connection is not established.
1101	DNMXS_PollNotAllocate	The Poll connection is not established.
1102	DNMXS_BitStrobeNotAllocate	The Strobe connection is not established.
1103	DNMXS_COSNotAllocate	The COS connection is not established.
1104	DNMXS_CyclicNotAllocate	The Cyclic connection is not established.
1105	DNMXS_PollAlreadyExist	The Poll connection has been established.
1106	DNMXS_BitStrobeAlreadyExist	The Poll connection has been established.
1107	DNMXS_COSAlreadyExist	The COS connection has been established.
1108	DNMXS_CyclicAlreadyExist	The Cyclic connection has been established.
1109	DNMXS_CommunicationPause	The communication between DNM100 board and all slave devices has been suspended.

Table 4.2.5 Interpretation of the return code (Slave Error) 1/1

Return Code	DeviceNet Error	Comment
1150	DNMXS_SlaveNoResp	The slave has no any response.
1151	DNMXS_WaitForSlaveResp	The DNM100 board is waiting for the response form the slave device.
1152	DNMXS_SlaveRespError	The slave replied some errors.
1153	DNMXS_OutputDataLenError	The output length of the I/O connection doesn't match the device's output length.
1154	DNMXS_InputDataLenError	The input length of the I/O connection doesn't match the device's input length.

4.3 Function Description

4.3.1 DNM100_GetBoardInfo

- **Description:**

This function is used to obtain the driver information of PISO-DNM100U or PEX-DNM100 board.

- **Syntax:**

```
DWORD DNM100_GetBoardInf (BYTE BoardNo, DWORD *dwVID,
                          DWORD *dwDID, DWORD *dwSVID,
                          DWORD *dwSDID, DWORD *dwSAuxID,
                          DWORD *dwIrqNo)
```

- **Parameter:**

BoardNo: [input] The DNM100 board number (0~15)
dwVID: [output] The address of a variable which is used to receive the vendor ID.
dwDID: [output] The address of a variable used to receive device ID.
dwSVID: [output] The address of a variable applied to receive sub-vendor ID.
dwSDID: [output] The address of a variable applied to receive sub-device ID.
dwSAuxID: [output] The address of a variable used to receive sub-auxiliary ID.
dwIrqNo: [output] The address of a variable used to receive logical interrupt number.

- **Return:**

Please refer to the chapter 4.2 for the function return code.

4.3.2 DNM100_TotalDNM100Board

- **Description:**

  The function can get the count of total PISO-DNM100U and PEX-DNM100 boards in the user's PC.

- **Syntax:**

```
DWORD DNM100_TotalDNM100board (BYTE *TotalBoards ,
                                                    BYTE *BoardIDList)
```

- **Parameter:**

  TotalBoards: [output] The amount of total DNM100 boards.
  BoardIDList: [output] The list of all DIP-Switch No. in each boards.

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.3 DNM100_ActiveBoard

4.3.3 DNM100_ActiveBoard

- **Description:**

  The function is used to activate the PISO-DNM100U or PEX-DNM100 board. It must be called once before using the other functions of DNM100 board APIs.

- **Syntax:**

  DWORD DNM100_ActiveBoard (BYTE BoardNo)

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.4 DNM100_CloseBoard

4.3.4 DNM100_CloseBoard

- **Description:**

  The function is used to stop and close the kernel driver and release the device resource from computer device resource. This method must be called once before exiting the user’s application program.

- **Syntax:**

  DWORD DNM100_CloseBoard (BYTE BoardNo)

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.5 DNM100_GetDLLVersion

- **Description:**

  The function can obtain the version information of DNM100.DLL.

- **Syntax:**

  DWORD DNM100_GetDLLVersion (void)

- **Parameter:**

  None

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.6 DNM100_GetFirmwareVersion

- **Description:**

  The function can obtain the version information of the firmware inside the DNM100 board.

- **Syntax:**

  DWORD DNM100_GetFirmwareVersion (BYTE BoardNo)

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)

- **Return:**

  The firmware version information. For example: If 100(hex) is return, it means firmware version is 1.00.

- **Error Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.7 DNM100_ResetFirmware

4.3.7 DNM100_ResetFirmware

- **Description:**

  The function is used to reset the PISO-DNM100U or PEX-DNM100 firmware. When the users have changed the baud rate of CAN bus or changed the Master's MAC ID, the function must be called to make the change enable. After calling this function, the users should wait for 1 or 2 seconds to make the firmware boot up completely.

- **Syntax:**

  DWORD DNM100_ResetFirmware (BYTE BoardNo)

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.8 DNM100_GetMasterMACID

4.3.8 DNM100_GetMasterMACID

- **Description:**

  The function can get the MAC ID of the DeviceNet master (PISO-DNM100U or PEX-DNM100).

- **Syntax:**

  DWORD DNM100_GetMasterMACID (BYTE BoardNo)

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.9 DNM100_SetMasterMACID

4.3.9 DNM100_SetMasterMACID

- **Description:**

  The function can set the MAC ID of the DeviceNet master (PISO-DNM100U or PEX-DNM100). After calling this function, the users must call DNM100_ResetFirmware to make the change enabled. It will save the information in the EEPROM in the DNM100 board.

- **Syntax:**

```
    DWORD DNM100_SetMasterMACID (BYTE BoardNo,
                                 BYTE MasterMACID)
```

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)
  MasterMACID: [input] The new MAC ID of the master. (0 ~ 63)

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.10 DNM100_GetBaudRate

- **Description:**

  This function can help you to get the DeviceNet baud rate information of DNM100 board.

- **Syntax:**

  DWORD DNM100_GetBaudRate (BYTE BoardNo)

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)

- **Return:**

  The CAN bus baud rate information in the DNM100 board.
  If the value is 0, the baud rate is 125Kbps.
  If the value is 1, the baud rate is 250Kbps.
  If the value is 2, the baud rate is 500Kbps.

- **Error Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.11 DNM100_SetBaudRate

- **Description:**

This function can set the DeviceNet baud rate of the DNM100 board. After calling this function, you must call DNM100_ResetFirmware to reset the firmware to make change enabled.

- **Syntax:**

```
DWORD DNM100_SetBaudRate (BYTE BoardNo,BYTE BaudRate)
```

- **Parameter:**

BoardNo: [input] The DNM100 board number (0~15)
BaudRate: [input] The new baud rate value.
0 : 125K bps
1 : 250K bps
2 : 500K bps

- **Return:**

Please refer to the chapter 4.2 for the function return code.

4.3.12 DNM100_GetMasterStatus

- **Description:**

  The function is used to obtain the firmware status inside the PISO-DNM100U or PEX-DNM100. The users can call this function to make sure that the DeviceNet master is online successfully.

- **Syntax:**

  DWORD DNM100_GetMasterStatus (BYTE BoardNo)

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.13 DNM100_GetSlaveStatus

- **Description:**

  This function is to get the remote slave device's communication status.

- **Syntax:**

```
DWORD DNM100_GetSlaveStatus (BYTE BoardNo,
                             BYTE DesMACID)
```

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)
  DesMACID: [input] The remote slave's MAC ID. (0~63)

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.14 DNM100_StartDevice

- **Description:**

  This function is used to start to communicate with the specific device that the users applying to.

- **Syntax:**

  DWORD DNM100_StartDevice (BYTE BoardNo, BYTE DesMACID)

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)
  DesMACID: [input] The remote slave's MAC ID. (0~63)

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.15 DNM100_StopDevice

- **Description:**

  This function is used to stop to communicate with the destination device that the users appointed to.

- **Syntax:**

  DWORD DNM100_StopDevice (BYTE BoardNo, BYTE DesMACID)

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)
  DestMACID: [input] The remote slave device's MAC ID (0~63)

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.16 DNM100_StartAllDevice

- **Description:**

  This function is used to start to communicate with all slave devices in ScanList.

- **Syntax:**

  DWORD DNM100_StartAllDevice (BYTE BoardNo)

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.17 DNM100_StopAllDevice

- **Description:**

  This function is used to stop to communicate with all destination devices in ScanList.

- **Syntax:**

  DWORD DNM100_StopAllDevice (BYTE BoardNo)

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.18 DNM100_AddDevice

- **Description:**

  This function can add the slave devices into the ScanList of the DNM100 board and save the information into the EEPROM. Before communicating with any slave devices, the users should call this function to add these devices.

- **Syntax:**

```
DWORD DNM100_AddDevice (BYTE BoardNo, BYTE DesMACID,
                                        WORD Explicit_EPR)
```

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)
  DestMACID: [input] The remote slave device's MAC ID (0~63)
  Explicit_EPR: [input] The Expected Packet Rate. (Usually is 2500).

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.19 DNM100_RemoveDevice

4.3.19 DNM100_RemoveDevice

- **Description:**

  This function is used for removing the specified slave device from the ScanList in the DNM100 board. And the information of the device in EEPROM is erased at the same time.

- **Syntax:**

  DWORD DNM100_RemoveDevice (BYTE BoardNo, BYTE DesMACID)

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)
  DestMACID: [input] The remote slave device's MAC ID (0~63)

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.20 DNM100_AddIOConnection

- **Description:**

This method is used to configure the I/O connection of the specific MAC ID device. The DNM100 board can get/set the data via the connection, which connects to the specific slave, according to the produced / consumed connection path of this slave device. This configuration data will be saved into EEPROM within the DNM100 board.

- **Syntax:**

```
DWORD DNM100_AddIOConnection (BYTE BoardNo, BYTE DesMACID,
                               BYTE ConType,
                               WORD DeviceInputLen,
                               WORD DeviceOutputLen,
                               WORD EPR)
```

- **Parameter:**

BoardNo: [input] The DNM100 board number (0~15)
DestMACID: [input] The remote slave device's MAC ID (0~63)
ConType: [input] The remote slave device's I/O connection type
0 : Explicit connection type
1 : Poll connection type
2 : Bit-Strobe connection type
3 : COS connection type
4 : Cyclic connection type
DeviceInputLen: [input] The remote slave device's input length. (Byte)
DeviceOutputLen: [input] The remote slave device's output length. (Byte)
EPR: [input] The expected packet rate. (mSec)

- **Return:**

Please refer to the chapter 4.2 for the function return code.

4.3.21 DNM100_RemoveIOConnection

4.3.21 DNM100_RemoveIOConnection

- **Description:**

  The function is used to remove the I/O connection configuration.

- **Syntax:**

```
DWORD DNM100_RemoveIOConnection (BYTE BoardNo,
                                 BYTE DesMACID,
                                 BYTE ConType)
```

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)
  DestMACID: [input] The remote slave device's MAC ID (0~63)
  ConType: [input] The remote slave device's I/O connection type
  0 : Explicit connection type
  1 : Poll connection type
  2 : Bit-Strobe connection type
  3 : COS connection type
  4 : Cyclic connection type

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

~~4.3.22 DNM100_GetAttribute~~

~~4.3.22 DNM100_GetAttribute~~

- ~~**Description:**~~

  ~~This function is used to send the request command to retrieve the attribute value of the specific device's instance. Before calling this function, you must start the device. After calling this function, you should execute the "DNM100_GetAttributeValue" to get the response message returned from remote slave device.~~

  This old function will be removed in the future. Please use the new function which is "DNM100_GetAttributeW".

- ~~**Syntax:**~~

```
~~DWORD DNM100_GetAttribute (BYTE BoardNo, BYTE DesMACID,~~
                                          ~~BYTE ClassID, BYTE InstanceID,~~
                                          ~~BYTE AttributeID)~~
```

- ~~**Parameter:**~~

  ~~**BoardNo**: [input] PISO-DNM100(U) board number (0~15)~~
  ~~**DestMACID:** [input] The remote slave device's MAC ID (0~63)~~
  ~~**ClassID:** [input] The remote slave device's ClassID(~~BYTE~~)~~
  ~~**InstanceID:** [input] The remote slave device's InstanceID(~~BYTE~~)~~
  ~~**AttributeID:** [input] The remote slave device's AttributeID~~

- ~~**Return:**~~

  ~~Please refer to the chapter 4.2 for the function return code.~~

4.3.23 DNM100_GetAttributeW

- **Description:**

  This function is used to send the request command to retrieve the attribute value of the specific device's instance. Before calling this function, you must start the device. After calling this function, you should execute the "DNM100_GetAttributeValue" to get the response message returned from remote slave device.

- **Syntax:**

```
DWORD DNM100_GetAttributeW (BYTE BoardNo, BYTE DesMACID,
                            WORD ClassID, WORD InstanceID,
                            BYTE AttributeID)
```

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)
  DestMACID: [input] The remote slave device's MAC ID (0~63)
  ClassID: [input] The remote slave device's ClassID(WORD)
  InstanceID: [input] The remote slave device's InstanceID(WORD)
  AttributeID: [input] The remote slave device's AttributeID

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.24 DNM100_IsGetAttributeOK

- **Description:**

  This function is used to check whether the DNM100 board has received the response message or not. After checking the response message, you should execute the “DNM100_GetAttributeValue” to get the response message returned from remote slave device.

- **Syntax:**

  DWORD DNM100_IsGetAttributeOK (BYTE BoardNo, BYTE DesMACID)

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)
  DestMACID: [input] The remote slave device’s MAC ID (0~63)

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.25 DNM100_GetAttributeValue

4.3.25 DNM100_GetAttributeValue

- **Description:**

This function is used to get the attribute value of the specific device's instance from the remote slave device. Before calling this function, the users should call DNM100_GetAttributeW to send request command first.

- **Syntax:**

DWORD DNM100_GetAttributeValue (BYTE BoardNo, BYTE DesMACID,
WORD *DataLen, BYTE *DATA)

- **Parameter:**

BoardNo: [input] The DNM100 board number (0~15)
DestMACID: [input] The remote slave device's MAC ID (0~63)
DataLen: [output] The length of the attribute value (in byte).
DATA: [output] The attribute value that returned from the slave device.

- **Return:**

Please refer to the chapter 4.2 for the function return code.

~~4.3.26 DNM100_SetAttribute~~

~~4.3.26 DNM100_SetAttribute~~

- ~~**Description:**~~

~~The method is used to set the attribute of the specific device's instance. Before calling this function, you must start the device. After calling this function, you should execute the "DNM100_IsSetAttributeOK" to check the response message returned from the remote slave device.~~

This old function will be removed in the future. Please use the new function which is "DNM100_SetAttributeW".

- ~~**Syntax:**~~

~~DWORD DNM100_SetAttribute (BYTE BoardNo, BYTE DesMACID,~~
~~BYTE ClassID, BYTE InstanceID,~~
~~BYTE AttributeID, WORD DataLen,~~
~~BYTE *DATA)~~

- ~~**Parameter:**~~

~~**BoardNo**: [input] PISO-DNM100(U) board number (0~15)~~
~~**DestMACID:** [input] The remote slave device's MAC ID (0~63)~~
~~**ClassID:** [input] The remote slave device's ClassID(BYTE)~~
~~**InstanceID:** [input] The remote slave device's InstanceID(BYTE)~~
~~**AttributeID:** [input] The remote slave device's AttributeID~~
~~**DataLen:** [input] The length of the attribute value (in byte).~~
~~**DATA:** [input] The attribute value that the users want to send.~~

- ~~**Return:**~~

~~Please refer to the chapter 4.2 for the function return code.~~

4.3.27 DNM100_SetAttributeW

- **Description:**

  The method is used to set the attribute of the specific device's instance. Before calling this function, you must start the device. After calling this function, you should execute the "DNM100_IsSetAttributeOK" to check the response message returned from the remote slave device.

- **Syntax:**

```
DWORD DNM100_SetAttributeW (BYTE BoardNo, BYTE DesMACID,
                            WORD ClassID, WORD InstanceID,
                            BYTE AttributeID, WORD DataLen,
                            BYTE *DATA)
```

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)
  DestMACID: [input] The remote slave device's MAC ID (0~63)
  ClassID: [input] The remote slave device's ClassID(WORD)
  InstanceID: [input] The remote slave device's InstanceID(WORD)
  AttributeID: [input] The remote slave device's AttributeID
  DataLen: [input] The length of the attribute value (in byte).
  DATA: [input] The attribute value that the users want to send.

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.28 DNM100_IsSetAttributeOK

4.3.28 DNM100_IsSetAttributeOK

- **Description:**

  This function is used to get the response value after executing the "DNM100_SetAttributeW" function.

- **Syntax:**

  DWORD DNM100_IsSetAttributeOK (BYTE BoardNo, BYTE DesMACID)

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)
  DestMACID: [input] The remote slave device's MAC ID (0~63)

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.29 DNM100_ClearAllConfig

4.3.29 DNM100_ClearAllConfig

- **Description:**

  This function will clear all configurations in the EEPROM of the DNM100 board.

- **Syntax:**

  DWORD DNM100_ClearAllConfig (BYTE BoardNo)

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.30 DNM100_SearchAllDevices

4.3.30 DNM100_SearchAllDevices

- **Description:**

  This function is used to retrieve all devices in DeviceNet network. This function makes the DNM100 board to start the searching process. The users need to check whether the process is complete or not by calling the “DNM100_IsSearchOK”. After completing the search process, the users could call the “DNM100_GetSearchedDevices“ to get the searched devices. Attention! This function will terminate all communications with remote devices. This function is usually used for developing or debugging applications.

- **Syntax:**

  DWORD DNM100_SearchAllDevices (BYTE BoardNo)

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.31 DNM100_SearchSpecificDevice

4.3.31 DNM100_SearchSpecificDevice

- **Description:**

This function is used to retrieve some devices which specified by the users. This function makes the DNM100 board to start the searching process. The users need to check whether the process is complete or not by calling the “DNM100_IsSearchOK”. After completing the search process, the users could call the “DNM100_GetSearchedDevices“ to get the searched devices. Attention! This function will terminate all communications with remote devices. This function is usually used for developing or debugging applications.

- **Syntax:**

```
DWORD DNM100_SearchSpecificDevice (BYTE BoardNo,
                                   WORD ListCount,
                                   BYTE *DesMACIDList)
```

- **Parameter:**

BoardNo: [input] The DNM100 board number (0~15)
ListCount: [input] The amount of the slave’s ID.
DestMACIDList: [input] The list of all slave’s MAC ID.

- **Return:**

Please refer to the chapter 4.2 for the function return code.

4.3.32 DNM100_IsSearchOK

4.3.32 DNM100_IsSearchOK

- **Description:**

  This function will check whether the searching process has finished or not.

- **Syntax:**

  DWORD DNM100_IsSearchOK (BYTE BoardNo)

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.33 DNM100_GetSearchedDevices

4.3.33 DNM100_GetSearchedDevices

- **Description:**

This function will get the device which have been searched in the network

- **Syntax:**

```
DWORD DNM100_GetSearchedDevices (BYTE BoardNo,
                                  WORD *TotalDevices,
                                  BYTE *DesMACID,
                                  BYTE *Type,
                                  WORD *DeviceInputLen,
                                  WORD *DeviceOutputLen)
```

- **Parameter:**

BoardNo: [input] The DNM100 board number (0~15)
TotalDevices: [output] The amount of all slave device which are found.
DesMACID: [output] The list of slave's MAC ID which are found.
Type: [output] The list of slave's connection type which are found.

0 : Explicit connection type
1 : Poll connection type
2 : Bit-Strobe connection type
3 : COS connection type
4 : Cyclic connection type

DeviceInputLen: [output] The list of slave's input length which are found.
DeviceOutputLen: [output] The list of slave's output length which are found.

- **Return:**

Please refer to the chapter 4.2 for the function return code.

4.3.34 DNM100_GetDeviceInfoFromScanList

4.3.34 DNM100_GetDeviceInfoFromScanList

- **Description:**

  This function will get the ScanList data of certain device in the DNM100 board.

- **Syntax:**

```
DWORD DNM100_GetDeviceInfoFromScanList
                    (BYTE BoardNo, BYTE DesMACID, WORD *ListCount,
                     BYTE *ConnectionTypeList, WORD *InputDataLenList,
                     WORD *OutputDataLenList,WORD *EPRList)
```

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)
  DesMACID: [input] The MAC ID number.
  ListCount: [output] The amount of all information items.
  ConnectionTypeList: [output] The list of slave's connection type.
  0 : Explicit connection type
  1 : Poll connection type
  2 : Bit-Strobe connection type
  3 : COS connection type
  4 : Cyclic connection type

  InputDataLenList: [output] The list of slave's input length.
  OutputDataLenList: [output] The list of slave's output length.
  EPRList: [output] The list of slave's expected packet rate.

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.35 DNM100_GetScanList

4.3.35 DNM100_GetScanList

- **Description:**

This function will get all the ScanList data in the DNM100 board.

- **Syntax:**

```
DWORD DNM100_GetScanList (BYTE BoardNo, WORD *TotalDevices,
                          BYTE *DesMACIDList,
                          BYTE *ConnectionTypeList,
                          WORD *InputDataLenList,
                          WORD *OutputDataLenList,
                          WORD *EPR_List)
```

- **Parameter:**

BoardNo: [input] The DNM100 board number (0~15)
TotalDevices: [output] The data count of all the information.
DestMACIDList: [output] The MAC ID of all the slave devices in the ScanList.
ConnectionTypeList: [output] The connection type of all the slave devices in the ScanList.
0 : Explicit connection type
1 : Poll connection type
2 : Bit-Strobe connection type
3 : COS connection type
4 : Cyclic connection type

InputDataLenList: [output] The input data length of all the slave devices in the ScanList.
OutputDataLenList: [output] The output data length of all the slave devices in the ScanList.
EPR_List: [output] The EPR value of all the slave devices in the ScanList.

- **Return:**

Please refer to the chapter 4.2 for the function return code.

4.3.36 DNM100_ImportEEPROM

4.3.36 DNM100_ImportEEPROM

- **Description:**

This function will write all specific devices' information in the ScanList to the EEPROM.

- **Syntax:**

```
DWORD DNM100_ImportEEPROM (BYTE BoardNo,
                           WORD ListCount,
                           BYTE *ConnectionTypeList,
                           WORD *InputDataLenList,
                           WORD *OutputDataLenList,
                           WORD *EPR_List)
```

- **Parameter:**

BoardNo: [input] The DNM100 board number (0~15)
ListCount: [input] The data count of all the information.
ConnectionTypeList: [input] The connection type of all slave devices.
0 : Explicit connection type
1 : Poll connection type
2 : Bit-Strobe connection type
3 : COS connection type
4 : Cyclic connection type

InputDataLenList: [input] The input data length of all slave devices.
OutputDataLenList: [input] The output data length of all slave devices.
EPR_List: [input] The EPR value of all slave devices.

- **Return:**

Please refer to the chapter 4.2 for the function return code.

4.3.37 DNM100_ReadInputData

4.3.37 DNM100_ReadInputData

- **Description:**

  This function is to get the data according with the produced connection path of the specific MAC ID device via the I/O connection.

- **Syntax:**

```
DWORD DNM100_ReadInputData (BYTE BoardNo, BYTE DesMACID,
                            BYTE ConType, WORD *IOLen,
                            BYTE *IODATA)
```

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)
  DestMACID: [input] The remote slave device’s MAC ID (0~63)
  ConType: [input] The connection type of the remote slave.
  0 : Explicit connection type
  1 : Poll connection type
  2 : Bit-Strobe connection type
  3 : COS connection type
  4 : Cyclic connection type

  IOLen: [output] The length of the I/O data (In byte).
  IODATA: [output] The remote I/O data.

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.38 DNM100_WriteOutputData

4.3.38 DNM100_WriteOutputData

- **Description:**

The function will set the data according with the consumed connection path of the specific MAC ID device via the I/O connection.

- **Syntax:**

```
DWORD DNM100_WriteOutputData (BYTE BoardNo, BYTE DesMACID,
                              BYTE ConType, WORD IOLen,
                              BYTE *IODATA)
```

- **Parameter:**

BoardNo: [input] The DNM100 board number (0~15)
DestMACID: [input] The remote slave device's MAC ID (0~63)
ConType: [input] The connection type of the remote slave.

0 : Explicit connection type
1 : Poll connection type
2 : Bit-Strobe connection type
3 : COS connection type
4 : Cyclic connection type

IOLen: [Input] The length of the I/O data (In byte).
IODATA: [Input] The remote I/O data.

- **Return:**

Please refer to the chapter 4.2 for the function return code.

~~4.3.39 DNM100_SendExplicitMSG~~

- ~~**Description:**~~

~~This function is used to send the explicit request command to retrieve or configure the attribute value of the specific device's instance. Before calling this function, you must start the device. After calling this function, you should execute the "DNM100_GetExplicitMSGRespValue" to get the response message returned from remote slave device.~~

This old function will be removed in the future. Please use the new function which is "DNM100_SendExplicitMSG_W".

- ~~**Syntax:**~~

```
~~DWORD DNM100_SendExplicitMSG (BYTE BoardNo, BYTE DesMACID,~~
                                    ~~BYTE ServiceID, BYTE ClassID,~~
                                    ~~BYTE InstanceID,~~
                                    ~~WORD DataLen, BYTE *DATA)~~
```

- ~~**Parameter:**~~

~~**BoardNo**: [input] PISO-DNM100(U) board number (0~15)~~
~~**DestMACID:** [input] The remote slave device's MAC ID (0~63)~~
~~**ServiceID:** [input] The remote slave device's ServiceID.~~
~~**ClassID:** [input] The remote slave device's ClassID(BYTE).~~
~~**InstanceID:** [input] The remote slave device's InstanceID(BYTE).~~
~~**DataLen:** [input] The length of the attribute value (in byte).~~
~~**DATA:** [input] The attribute value that the users want to send.~~

- ~~**Return:**~~

~~Please refer to the chapter 4.2 for the function return code.~~

4.3.40 DNM100_SendExplicitMSG_W

- **Description:**

This function is used to send the explicit request command to retrieve or configure the attribute value of the specific device's instance. Before calling this function, you must start the device. After calling this function, you should execute the "DNM100_GetExplicitMSGRespValue" to get the response message returned from remote slave device.

- **Syntax:**

```
DWORD DNM100_SendExplicitMSG_W (BYTE BoardNo, BYTE DesMACID,
                                BYTE ServiceID, WORD ClassID,
                                WORD InstanceID,
                                WORD DataLen, BYTE *DATA)
```

- **Parameter:**

BoardNo: [input] The DNM100 board number (0~15)
DestMACID: [input] The remote slave device's MAC ID (0~63)
ServiceID: [input] The remote slave device's ServiceID.
ClassID: [input] The remote slave device's ClassID(WORD).
InstanceID: [input] The remote slave device's InstanceID(WORD).
DataLen: [input] The length of the attribute value (in byte).
DATA: [input] The attribute value that the users want to send.

- **Return:**

Please refer to the chapter 4.2 for the function return code.

4.3.41 DNM100_IsExplicitMSGRespOK

- **Description:**

This function is used to check whether the DNM100 board has received the response message or not. After checking the response message, you should execute the “DNM100_GetExplicitMSGRespValue” to get the response message returned from remote slave device.

- **Syntax:**

```
DWORD DNM100_IsExplicitMSGRespOK (BYTE BoardNo,
                                   BYTE DesMACID)
```

- **Parameter:**

BoardNo: [input] The DNM100 board number (0~15)
DestMACID: [input] The remote slave device’s MAC ID (0~63)

- **Return:**

Please refer to the chapter 4.2 for the function return code.

4.3.42 DNM100_GetExplicitMSGRespValue

- **Description:**

  This function is used to get the attribute value of the specific device's instance from the remote slave device. Before calling this function, the users should call DNM100_SendExplicitMSG_W to send request command first.

- **Syntax:**

```
DWORD DNM100_GetExplicitMSGRespValue (BYTE BoardNo,
                                      BYTE DesMACID,
                                      WORD *DataLen ,
                                      BYTE *DATA)
```

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)
  DestMACID: [input] The remote slave device's MAC ID (0~63)
  DataLen: [output] The length of the attribute value (in byte).
  DATA: [output] The attribute value that returned from the slave device.

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.43 DNM100_ReadbackOutputData

- **Description:**

  The function will read the data according with the consumed connection path of the specific MAC ID device via the I/O connection.

- **Syntax:**

```
DWORD DNM100_ReadbackOutputData (BYTE BoardNo,
                                  BYTE DesMACID,
                                  BYTE ConType,
                                  WORD *IOLen,
                                  BYTE *IODATA)
```

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15)
  DestMACID: [input] The remote slave device's MAC ID (0~63)
  ConType: [input] The connection type of the remote slave.
  0 : Explicit connection type
  1 : Poll connection type
  2 : Bit-Strobe connection type
  3 : COS connection type
  4 : Cyclic connection type

  IOLen: [output] The length of the I/O data (In byte).
  IODATA: [output] The remote I/O data.

- **Return:**

  Please refer to the chapter 4.2 for the function return code.

4.3.44 DNM100_ReadInputArea (Advanced Option)

- **Description:**

  This function read the bulk data from the input area memory. Users can use this function to read all the input data quickly. It provides effective method to read the bulk data in one command. If the users need to read the input data of all slave devices, they need to call the "DNM100_ReadInputData" for each slave device in general. By calling the advanced function "DNM100_ReadInputArea", the whole input data would be read once in this function call. Please refer to the section 3.7 for more description.

- **Syntax:**

```
DWORD DNM100_ReadInputArea (BYTE BoardNo,
                            WORD Offset,
                            WORD DataLen,
                            BYTE *DataArray)
```

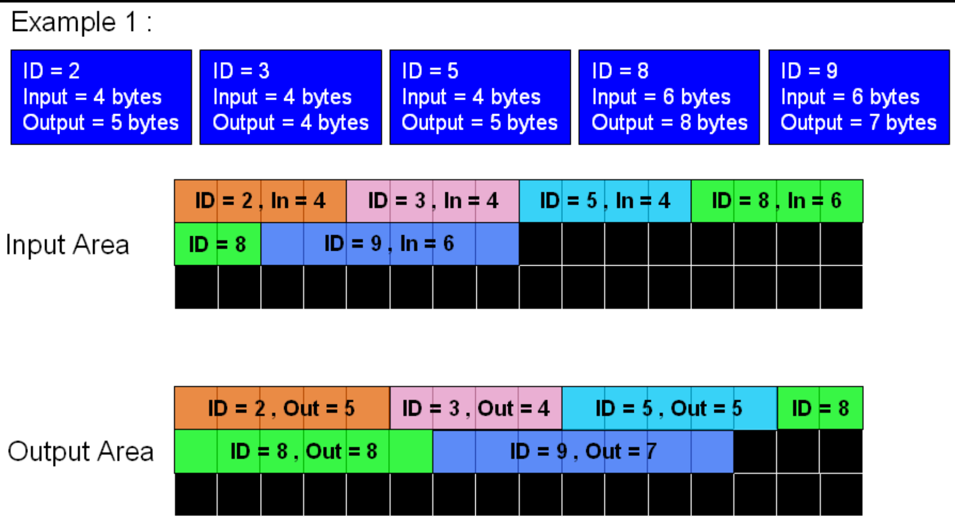
- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15).
  Offset: [input] The offset of the input area memory.
  DataLen: [input] The length in byte which the users want to read.
  DataArray: [output] The data pointer of the obtained data from the input area memory.

- **Return:**

  Please refer to the section 4.2 for the return code.

- **Example:**



```
BYTE BoardNo = 0; //Assume that the DNM100's board No. is 0.
WORD Offset = 0;
WORD DataLen = 0;
BYTE DataArray[512] = {0};

//Read whole input data of all slave devices.
Offset = 0; //Read the data from the beginning of the input area.
DataLen = 4 + 4 + 4 + 6 + 6; //The sum of all slave's input length.
DNM100_ReadInputArea (BoardNo, Offset, DataLen, DataArray);
DataArray = [The whole input data of all slave devices];


//Read input data of the slave device which ID = 5.
Offset = 4 + 4; //Read the data from the beginning of the ID = 5.
DataLen = 4; //The input length of the ID = 5.
DNM100_ReadInputArea (BoardNo, Offset, DataLen, DataArray);
DataArray = [The input data of the slave device which ID = 5];
```

4.3.45 DNM100_WriteOutputArea (Advanced Option)

- **Description:**

This function can write the bulk data to the output area memory. The DeviceNet slave will change the output status of the DeviceNet slaves according to the data located in the output area memory. Users can use this function to write a large amount of the output data. It provides effective method to change bulk data in one command. If the users need to write the output data of all slave devices, they need to call the "DNM100_WriteOutputData" for each slave device in general. By calling the advanced function "DNM100_WriteOutputArea", the whole output data would be wrote once in this function call. Please refer to the section 3.7 for more description.

- **Syntax:**

```
DWORD DNM100_WriteOutputArea (BYTE BoardNo,
                              WORD Offset,
                              WORD DataLen,
                              BYTE *DataArray)
```

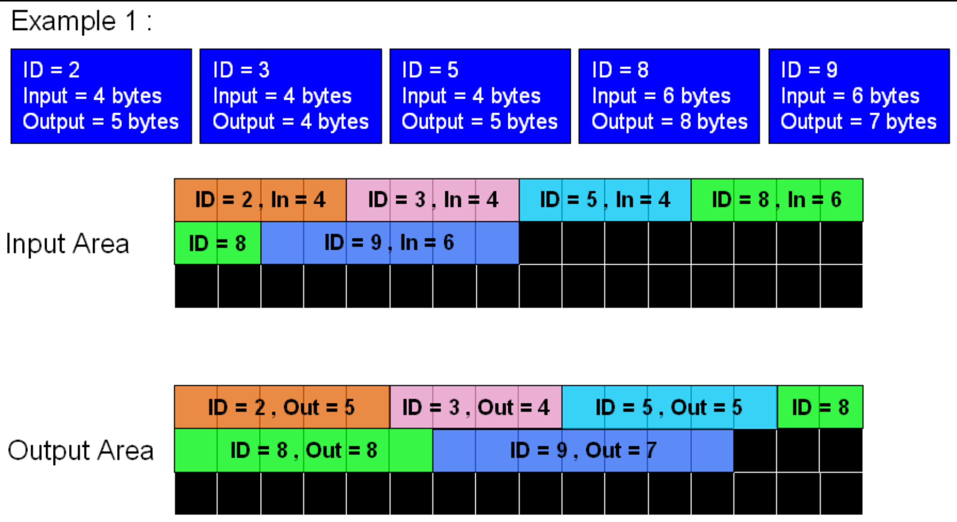
- **Parameter:**

BoardNo: [input] The DNM100 board number (0~15).
Offset: [input] The offset of the input area memory.
DataLen: [input] The length in byte which the users want to write.
DataArray: [input] The data pointer pointed the data written to the output data area.

- **Return:**

Please refer to the section 4.2 for the return code.

- **Example:**



```
BYTE BoardNo = 0; //Assume that the DNM100's board No. is 0.
WORD Offset = 0;
WORD DataLen = 0;
BYTE DataArray[512] = {0};

//Write the whole output data of all slave devices.
Offset = 0; //Write the data from the beginning of the output area.
DataLen = 5 + 4 + 5 + 8 + 7; //The sum of all slave's output length.
DataArray = [The whole output data];
DNM100_WriteOutputArea (BoardNo, Offset, DataLen, DataArray);


//Write the output data of the slave device which ID = 5.
Offset = 5 + 4; //Write the data from the beginning of the ID = 5.
DataLen = 5; //The output length of the ID = 5.
DataArray = [The output data of the ID = 5];
DNM100_WriteOutputArea (BoardNo, Offset, DataLen, DataArray);
```

4.3.46 DNM100_ReadbackOutputArea (Advanced Option)

- **Description:**

This function reads the bulk data from the output area memory. It dose not change the output data located in the output area memory. Users can use this function to read back a large amount of the output data, but the data may be different with the real output statuses of the DeviceNet slaves. It provides effective method to read bulk data in one command. If the users need to read back the output data of all slave devices, they need to call the “DNM100_ReadbackOutputData” for each slave device in general. By calling the advanced function “DNM100_ReadbackOutputArea”, the whole output data would be wrote once in this function call. Please refer to the section 3.7 for more description.

- **Syntax:**

```
DWORD DNM100_ReadbackOutputArea (BYTE BoardNo,
                                 WORD Offset,
                                 WORD DataLen,
                                 BYTE *DataArray)
```

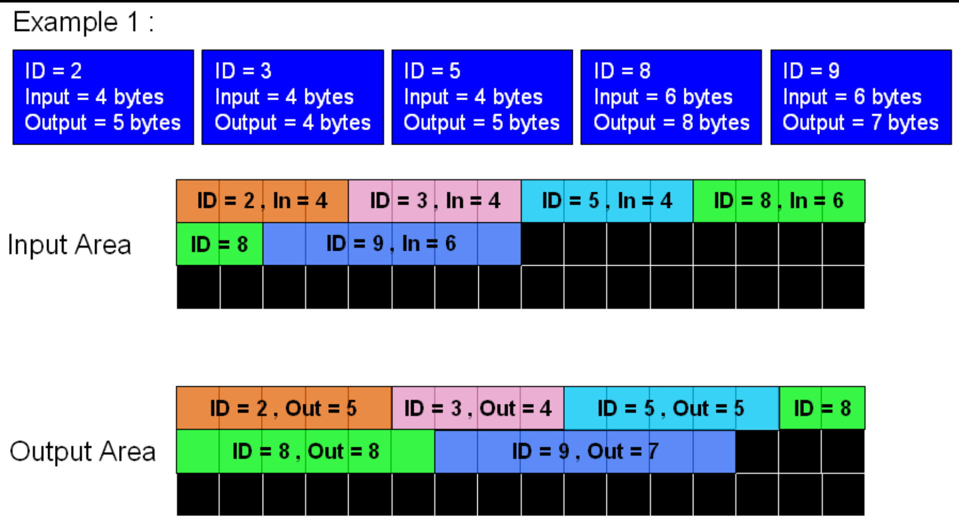
- **Parameter:**

BoardNo: [input] The DNM100 board number (0~15).
Offset: [input] The offset of the output area memory.
DataLen: [input] The length in byte which the users want to read.
DataArray: [output] The data pointer pointed the observed data form the output area memory.

- **Return:**

Please refer to the section 4.2 for the return code.

- **Example:**



```
BYTE BoardNo = 0; //Assume that the DNM100's board No. is 0.
WORD Offset = 0;
WORD DataLen = 0;
BYTE DataArray[512] = {0};

//Read back the whole output data of all slave devices.
Offset = 0; //Read back the data from the beginning of the output area.
DataLen = 5 + 4 + 5 + 8 + 7; //The sum of all slave's output length.
DNM100_ReadbackOutputArea (cSlot, Offset, DataLen, DataArray);
DataArray = [The whole output data];


//Read back the output data of the slave device which ID = 5.
Offset = 5 + 4; //Read back the data from the beginning of the ID = 5.
DataLen = 5; //The output length of the ID = 5.
DNM100_WriteOutputArea (cSlot, Offset, DataLen, DataArray);
DataArray = [The output data of the ID = 5];
```

4.3.47 DNM100_PauseIOConnection (Advanced Option)

- **Description:**

  The function will disconnect the I/O connection with the remote slave. When communicating with the remote slave devices and need to pause the I/O connection a while, the users can use this function to disconnect the I/O connection which has been established. When the I/O connection has been suspended, the [Explicit Connection] will still exist and the read/write I/O functions will not change the I/O data of the slave devices. The user could use “Get/SetAttributeW” and “SendExplicitMSG_W” functions to configure some parameters when the I/O connection has been suspended.

- **Syntax:**

```
DWORD DNM100_PauseIOConnection (BYTE BoardNo,
                                                      BYTE DesMACID)
```

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15).
  DestMACID: [input] The remote slave device’s MAC ID (0~63)

- **Return:**

  Please refer to the section 4.2 for the return code.

4.3.48 DNM100_ResumeIOConnection (Advanced Option)

- **Description:**

  The function will re-connect the I/O connection which has been paused. After connecting the I/O connection, the users could use the “read/write I/O” functions and “Get/SetAttributeW” functions.

- **Syntax:**

```
DWORD DNM100_ResumeIOConnection (BYTE BoardNo,
                                 BYTE DesMACID)
```

- **Parameter:**

  BoardNo: [input] The DNM100 board number (0~15).
  DestMACID: [input] The remote slave device’s MAC ID (0~63)

- **Return:**

  Please refer to the section 4.2 for the return code.

4.3.49 DNM100_DisableKeepAliveMsg (Advanced Option)

- **Description:**

The DNM100 board will read periodically certain explicit attribute to keep the explicit connection alive. This function can disable the reading process. For some slave devices, the keep explicit connection is not necessary. The users can call this function after the DNM100_ActiveBoard(). This disable status will NOT keep in PISO-DNM100U board. The users need to call this for every boot-up.

- **Syntax:**

```
DWORD DNM100_DisableKeepAliveMsg (BYTE BoardNo,
                                  BYTE DesMACID)
```

- **Parameter:**

BoardNo: [input] The DNM100 board number (0~15).
DestMACID: [input] The remote slave device’s MAC ID (0~63)

- **Return:**

Please refer to the section 4.2 for the return code.

5. Demo Programs for Windows

All of demo programs will not work normally if DNM100 driver would not be installed correctly. During the installation process of the driver, the install-shields will register the correct kernel driver to the operation system and copy the DLL driver and demo programs to the correct position based on the driver software package you have selected. After completing the driver installation, the related demo programs, development library and declaration header files for different development environments are installed in the system as follows.

The DNM100's root directory is C:\ICPDAS\DNM100_Boards

|--\DLL	The DLL driver for the user's application
|--\Driver	The window driver of the DNM100 board
|--\Manual	The user manual of the DNM100 board
|--\Demo	Demo program
|--\Demo\BCB 6	Demos for Borland C++ Builder 6
|--\Demo\VC++ 6	Demos for Visual C++ 6

5.1 A brief introduction to the demo programs

VC_Demo1 : Demonstrate the basic functions to communicate with the remote slave device. The demo program will lead you step by step to complete the setting and communication.

VC_Demo2 : Demonstrate the scan function to scan all the remote slave devices in the same DeviceNet network. The demo program will show you all the slave devices and their I/O connection type.

BCB_Demo1 : Demonstrate the scan function and add/remove function to configure the information of the remote slave device.

BCB_Demo2 : Demonstrate the I/O functions to access the I/O data of the remote slave device. The demo program will show you the input value of the remote device and let you send out the data to the output pins of the remote slave devices.

5.2 Wire Connection of the CAN bus

Before starting the demos, the users should have at least one slave device. Here show the users how to connect the master and slave devices by CAN bus. The slave devices should be connected to form the serial type which is shown as Figure 5.1

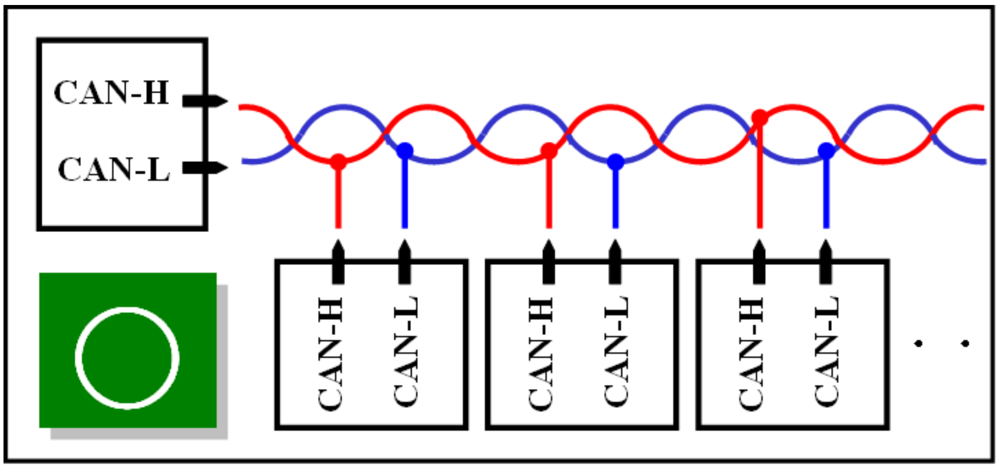


Figure 5.1 Correct wire connection

The following wire connection is **wrong** which is shown as Figure 5.2

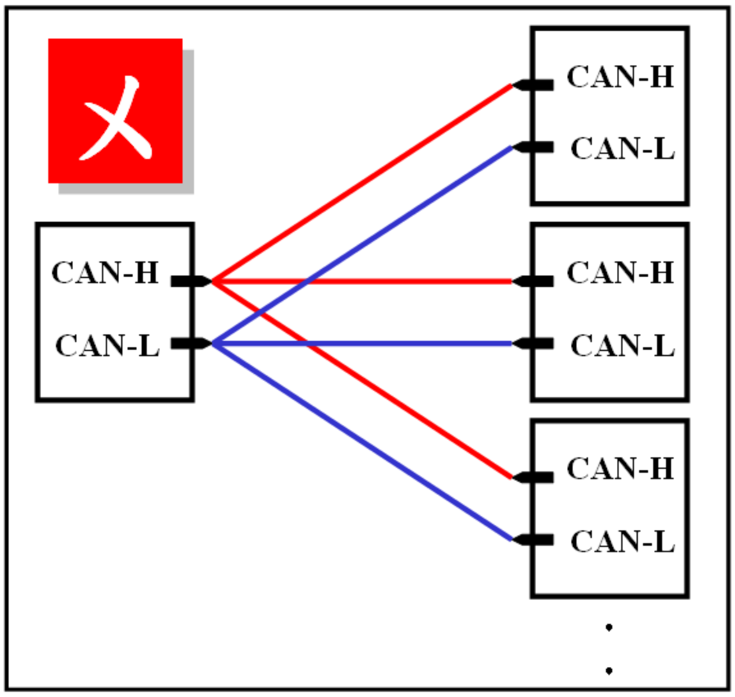
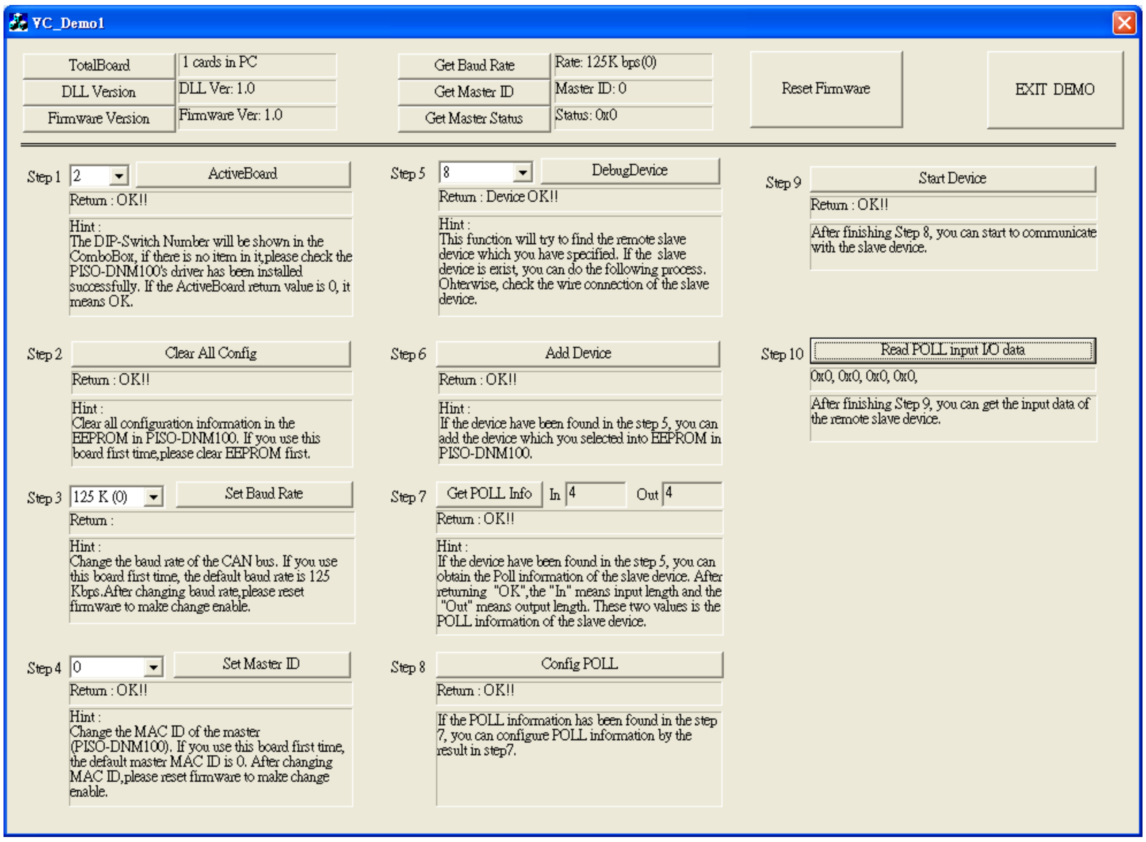


Figure 5.2 Wrong wire connection

5.3 VC_Demo1 Introduction

VC_Demo1 is the example used for starting the DeviceNet communication. The screen shoot is shown as Figure 5.3. This demo program is designed to communicate with slave devices step by step. This program will read the input value of the slave device when the POLL connection has been established. Before exercising this demo, the users should have at least one DeviceNet slave device which has input channels (AI or DI) and finish the wire connection between the Master and slave device. (See Figure 5.1)



1 cards in PC
DLL Ver: 1.0
Firmware Ver: 1.0

Rate: 125K bps(0)
Master ID: 0
Status: 0x0

Step 1 2
Return : OK!!
Hint :
The DIP-Switch Number will be shown in the ComboBox, if there is no item in it,please check the PISO-DNM100's driver has been installed successfully. If the ActiveBoard return value is 0, it means OK.

Step 2
Return : OK!!
Hint :
Clear all configuration information in the EEPROM in PISO-DNM100. If you use this board first time,please clear EEPROM first.

Step 3 125 K (0)
Return :
Hint :
Change the baud rate of the CAN bus. If you use this board first time, the default baud rate is 125 Kbps.After changing baud rate,please reset firmware to make change enable.

Step 4 0
Return : OK!!
Hint :
Change the MAC ID of the master (PISO-DNM100). If you use this board first time, the default master MAC ID is 0. After changing MAC ID,please reset firmware to make change enable.

Step 5 8
Return : Device OK!!
Hint :
This function will try to find the remote slave device which you have specified. If the slave device is exist, you can do the following process. Ohterwise, check the wire connection of the slave device.

Step 6
Return : OK!!
Hint :
If the device have been found in the step 5, you can add the device which you selected into EEPROM in PISO-DNM100.

Step 7 In 4 Out 4
Return : OK!!
Hint :
If the device have been found in the step 5, you can obtain the Poll information of the slave device. After returning "OK",the "In" means input length and the "Out" means output length. These two values is the POLL information of the slave device.

Step 8
Return : OK!!
If the POLL information has been found in the step 7, you can configure POLL information by the result in step7.

Step 9
Return : OK!!
After finishing Step 8, you can start to communicate with the slave device.

Step 10
0x0, 0x0, 0x0, 0x0,
After finishing Step 9, you can get the input data of the remote slave device.

Figure 5.3 the screenshoot of VC_Demo1

After running the program, the users will see the “TotalBoards” information on the left and up corner of the screen. This function determinates how many DNM100 board in your PC automatically. If the program doesn’t find any board, the users should check that the windows driver has been installed successfully. Otherwise, if it has found at lease one board, the users can continue exercising the demo program.

Step 1 : ActiveBoard

Before performing other buttons, the "ActiveBoard" button should be clicked firstly. The DIP-Switch on the board means the ID of this board. The users should make sure that every board's ID in your PC is unique. The dropdown list will show the board's ID which the users have selected. After clicking the button, the return code will be 0. Otherwise, please check the windows driver has been installed successfully.

Step 2 : Clear All Config

To avoid unknown configuration in the DNM100 board, the users can click this button to clear all configuration in the board.

Step 3 : Set Baud Rate

The default baud rate of DeviceNet is 125Kbps. If the users want to change the value, they can select the correct value then click the button. After changing the baud rate, the users should reset the firmware in the DNM100 board by clicking the "ResetFirmware" button. It needs to wait for 1 or 2 seconds to the next step.

Step 4 : Set Master ID

The default Master's MAC ID is 0. If the users want to change the value, you can select the correct value then click the button. After changing the Master's ID, the uses should reset the firmware in the DNM100 board by clicking the "ResetFirmware" button. It needs to wait for 1 or 2 seconds to the next step.

Step 5 : Debug Device

Before performing this function, the users should set the MAC ID of the slave device and turn on it. In the demo, the users can select the MAC ID of the slave device then click "Debug Device" button. This function will try to find the appointed remote slave device waiting for 5 seconds, if the slave device exists in the network, the return value will be 0. Otherwise, the device doesn't exist without response. The users can go to next step until the problem of the slave device has been solved.

Step 6 : Add Device

Before performing this function, the users should use step 5 to find an exist device. This function can add the device's information found at step 5 into

EEPROM of the DNM100 board. If it is successful, the return value will be 0.

Step 7 : Get POLL Info

This function is to obtain the device’s POLL information. After the slave device responses the data, it would show in the “In” and “Out”. “In” means the input length of the slave device. “Out” means the output length of the slave device. If the slave device responses successfully, the return value will be 0.

Step 8 : Config POLL

If the step 7 is successful, the users can perform “Config POLL” button. This function is to add the device’s POLL information into EEPROM in the DNM100 board. If it is successful, the return value will be 0.

Step 9 : Start Device

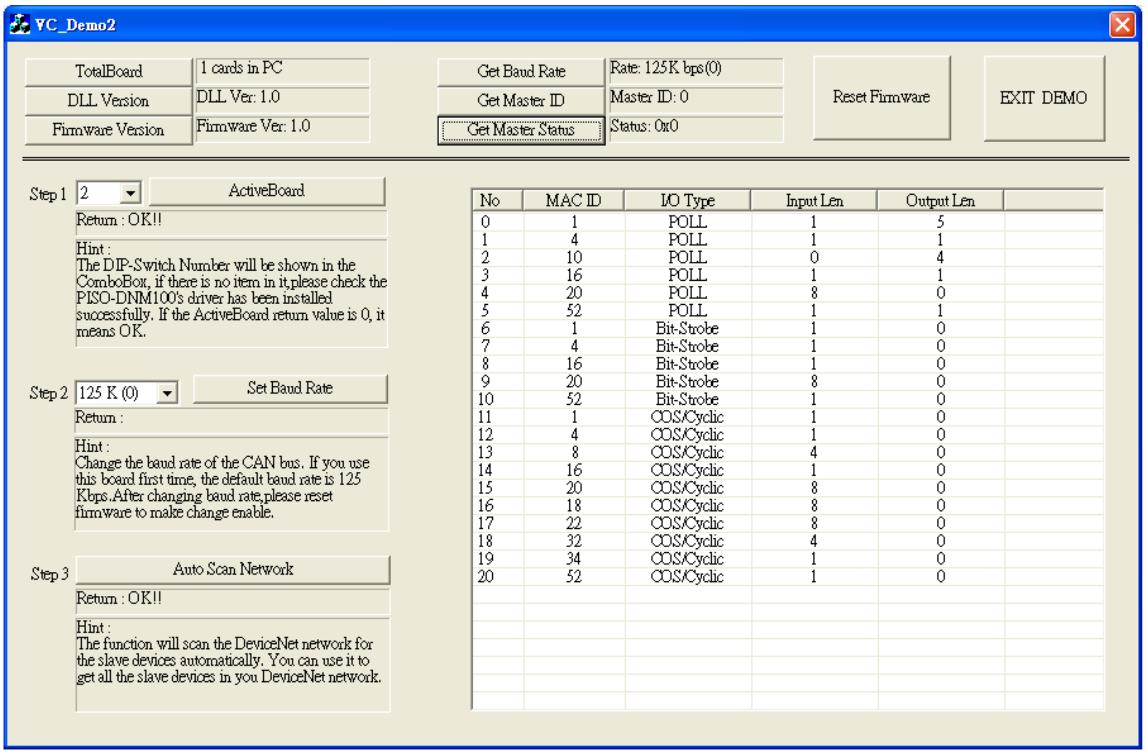
If the step 8 is successful, the users can perform “Start Device” button. This function would communicate with the slave device which the users have configured in the previous steps. If it is successful, the return value will be 0.

Step 10 : Read POLL Input I/O Data

If the master is communicating with the slave device successfully, the users can read the input I/O data from the slave device in this step. This function would obtain the input I/O data and show them in byte (8-bits).

5.4 VC_Demo2 Introduction

VC_Demo2 is the example used for scanning the DeviceNet slave devices in the network. The screen shoot is shown as Figure 5.4. This demo program is designed to operate the master step by step. This program will show the information of all the slave devices in the network. Before exercising this demo, the users should have at least one DeviceNet slave device and finish the wire connection between the Master and slave device. (See Figure 5.1)



1 cards in PC
DLL Ver: 1.0
Firmware Ver: 1.0
Rate: 125K bps(0)
Master ID: 0
Status: 0x0

Step 1 2
Return : OK!!
Hint :
The DIP-Switch Number will be shown in the ComboBox, if there is no item in it,please check the PISO-DNM100's driver has been installed successfully. If the ActiveBoard return value is 0, it means OK.

Step 2 125 K (0)
Return :
Hint :
Change the baud rate of the CAN bus. If you use this board first time, the default baud rate is 125 Kbps.After changing baud rate,please reset firmware to make change enable.

Step 3
Return : OK!!
Hint :
The function will scan the DeviceNet network for the slave devices automatically. You can use it to get all the slave devices in you DeviceNet network.

No	MAC ID	I/O Type	Input Len	Output Len
0	1	POLL	1	5
1	4	POLL	1	1
2	10	POLL	0	4
3	16	POLL	1	1
4	20	POLL	8	0
5	52	POLL	1	1
6	1	Bit-Strobe	1	0
7	4	Bit-Strobe	1	0
8	16	Bit-Strobe	1	0
9	20	Bit-Strobe	8	0
10	52	Bit-Strobe	1	0
11	1	COS/Cyclic	1	0
12	4	COS/Cyclic	1	0
13	8	COS/Cyclic	4	0
14	16	COS/Cyclic	1	0
15	20	COS/Cyclic	8	0
16	18	COS/Cyclic	8	0
17	22	COS/Cyclic	8	0
18	32	COS/Cyclic	4	0
19	34	COS/Cyclic	1	0
20	52	COS/Cyclic	1	0

Figure 5.4 The screenshoot of VC_Demo2

After running the program, the users will see the “TotalBoards” information on the left and up corner of the screen. This function determines how many DNM100 board in your PC automatically. If the program doesn’t find any board, the users should check that the windows driver has been installed successfully. Otherwise, if it has found at lease one board, the users can continue exercising the demo program.

Step 1 : ActiveModule

Before performing other buttons, the “ActiveBoard” button should be clicked firstly. The DIP-Switch on the board means the ID of this board. The users should make sure that every board’s ID in your PC is unique. The dropdown list will show the board’s ID which the users have selected. After clicking the button, the return code will be 0. Otherwise, please check the windows driver has been installed successfully.

Step 2 : Set Baud Rate

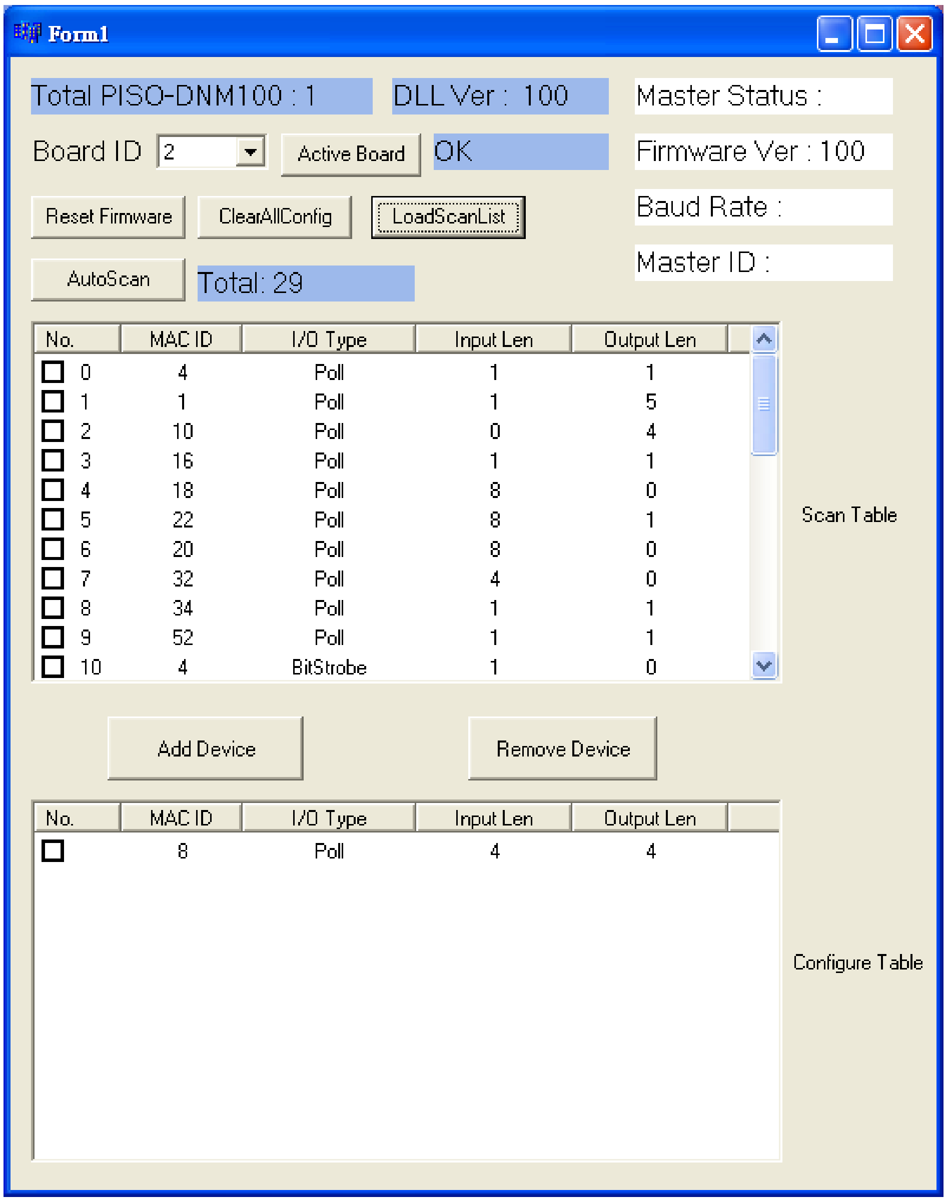
The default baud rate of DeviceNet is 125Kbps. If the users want to change the value, they can select the correct value then click the button. After changing the baud rate, the users should reset the firmware in the DNM100 board by clicking the “ResetFirmware” button. It needs to wait for 1 or 2 seconds to the next step.

Step 3 : Auto Scan Network

Before performing this function, the users should set the MAC ID and the baud rate of the slave device and turn on it. In the demo, the user can click “Auto Scan Network” button to obtain all the I/O information of all slave devices in the network. The scan procedure needs about 30 seconds. The users will see the entire slave device in the network and their I/O information in the list table.

5.5 BCB_Demo1 Introduction

BCB_Demo1 is the example used for scanning the DeviceNet slave devices in the network. The screen shoot is shown as Figure 5.5. This program will show the information of all the slave devices. This demo is similar to VC_Demo2. Before exercising this demo, the users should have at least one DeviceNet slave device and finish the wire connection between the Master and slave device. (See Figure 5.1) The users can configure the slave device by the scanning information. This demo can be a tool to add or remove the configuration of the slave device.



Total PISO-DNM100 : 1

DLL Ver : 100

Master Status :

Board ID 2

OK

Firmware Ver : 100

Baud Rate :

Master ID :

Total: 29

No.	MAC ID	I/O Type	Input Len	Output Len
☐ 0	4	Poll	1	1
☐ 1	1	Poll	1	5
☐ 2	10	Poll	0	4
☐ 3	16	Poll	1	1
☐ 4	18	Poll	8	0
☐ 5	22	Poll	8	1
☐ 6	20	Poll	8	0
☐ 7	32	Poll	4	0
☐ 8	34	Poll	1	1
☐ 9	52	Poll	1	1
☐ 10	4	BitStrobe	1	0

No.	MAC ID	I/O Type	Input Len	Output Len
☐	8	Poll	4	4

Figure 5.5 The screenshoot of BCB_Demo1

After running the program, the users would see the “Total PISO-DNM100(U) : x” information on the left and up corner of the screen. This function determinates how many DNM100 board in your PC automatically. If the program doesn’t find any board, the users should check that the windows driver has been installed successfully. Otherwise, if it has found at lease one module, the users can continue exercising the demo program.

Step 1 : Active Module

Before performing other buttons, the “ActiveBoard” button should be clicked firstly. The DIP-Switch on the board means the ID of this board. The users should make sure that every board’s ID in your PC is unique. The dropdown list will show the board’s ID which the users have selected. After clicking the button, the return code will be 0. Otherwise, please check the windows driver has been installed successfully.

Step 2 : Auto Scan

Before performing this function, the users should set the MAC ID and the baud rate of the slave device and turn on it. In the demo, the user can click “Auto Scan Network” button to obtain all the I/O information of all slave devices in the network. The scan procedure needs about 30 seconds. The users will see the entire slave device in the network and their I/O information in the “Scan Table”.

Step 3 : Add Device

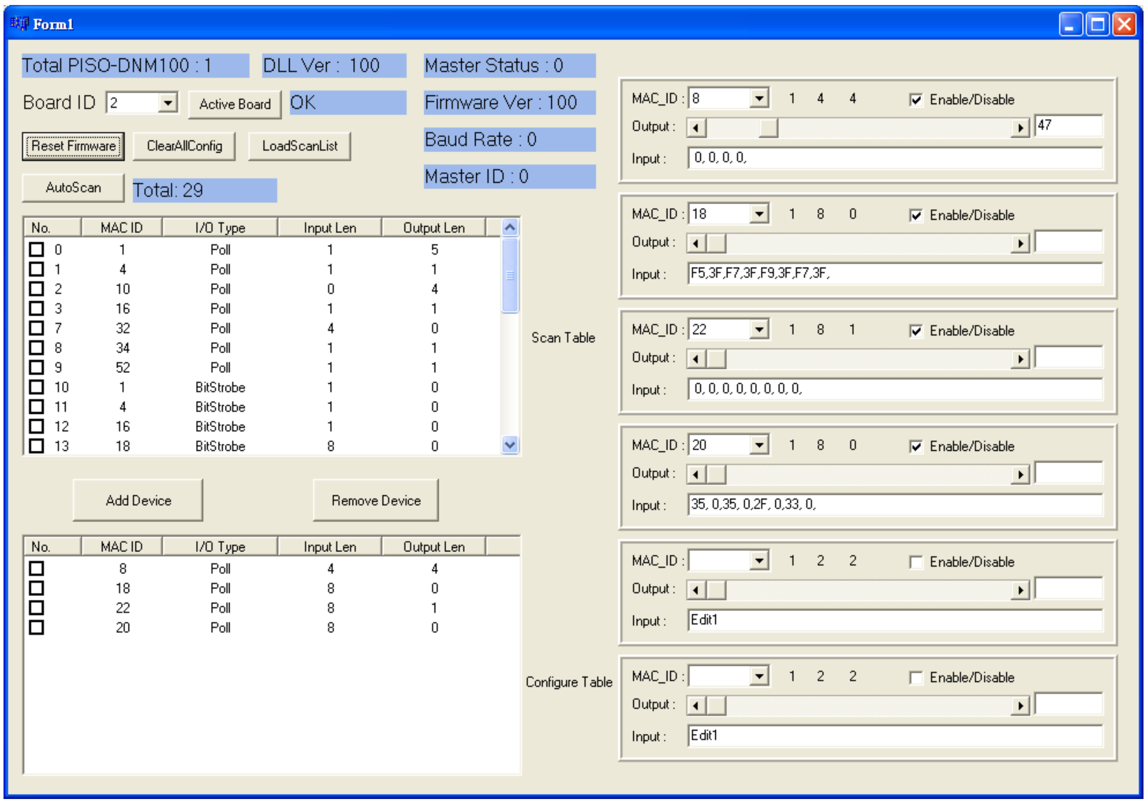
This function is to add the device’s information into EEPROM in the DNM100 board. The users can check the item which you want to add. After checking what you want, push “Add Device” button to add the information into the EEPROM. If it is successful, the items which you selected will be shown in the “Configure Table”.

Step 4 : LoadScanList

This function is to obtain the device’s information from EEPROM in the DNM100 board. The users can check the information in the EEPROM. After performing the function, the information will be shown in the “Configure Table”.

5.6 *BCB_Demo2 Introduction*

BCB_Demo2 is the extension of the BCB_Demo1. The screen shoot is shown as Figure 5.6. This program can read the input data and write the output data in every second. This demo is similar to BCB_Demo1. We just introduce the extension part. Before exercising this demo, the users should have at least one DeviceNet slave device and finish the wire connection between the Master and slave device. (See Figure 5.1) The users can configure the slave device by the scanning information. This demo can be a tool to add or remove the configuration of the slave device. Additionally, the users can operate the I/O data form the remote slave devices.



Total PISO-DNM100 : 1
DLL Ver : 100
Master Status : 0
Board ID 2
OK
Firmware Ver : 100
Baud Rate : 0
Master ID : 0
Total: 29

Scan Table

No.	MAC ID	I/O Type	Input Len	Output Len
☐ 0	1	Poll	1	5
☐ 1	4	Poll	1	1
☐ 2	10	Poll	0	4
☐ 3	16	Poll	1	1
☐ 7	32	Poll	4	0
☐ 8	34	Poll	1	1
☐ 9	52	Poll	1	1
☐ 10	1	BitStrobe	1	0
☐ 11	4	BitStrobe	1	0
☐ 12	16	BitStrobe	1	0
☐ 13	18	BitStrobe	8	0

Configure Table

No.	MAC ID	I/O Type	Input Len	Output Len
☐	8	Poll	4	4
☐	18	Poll	8	0
☐	22	Poll	8	1
☐	20	Poll	8	0

MAC_ID :				Enable/Disable	Output :	Input :
8	1	4	4	✓	47	0, 0, 0, 0,
18	1	8	0	✓		F5,3F,F7,3F,F9,3F,F7,3F,
22	1	8	1	✓		0, 0, 0, 0, 0, 0, 0, 0,
20	1	8	0	✓		35, 0,35, 0,2F, 0,33, 0,
	1	2	2	☐		Edit1
	1	2	2	☐		Edit1

Figure 5.6 The screen shoot of BCB_Demo2

If the users want to know how to configure the slave device information into EEPROM in the DNM100 board, please refer to section 5.5.

If "Active Board" is OK, the users can click "LoadScanList" button. The configuration information will be shown in "Configure Table". At the same time, the MAC IDs also are shown on the right side of the screen. The users can check the "Enable/Disable" box to enable or disable the read and write the I/O data. The "Output" scroll bar presents the output value which will be written to the output channel of the slave device. If the scroll value is 0x23, every byte of the slave device's output is 0x23. The users can find out the change of the output channel easily. The "Input" field presents the input value from the input channel of the slave device.

6. LabVIEW Driver Introduction

6.1 Software Installation

The DNM100 LabVIEW 8.x driver is the DNM100 board function reference for the DeviceNet master device used on LabVIEW 8.x environment on Windows. Before users use this driver to develop the machine control system, the DNM100 driver must be installed first, because the DNM100 LabVIEW 8.x driver needs to call the function of it. The driver architecture is shown as Figure 6.1.

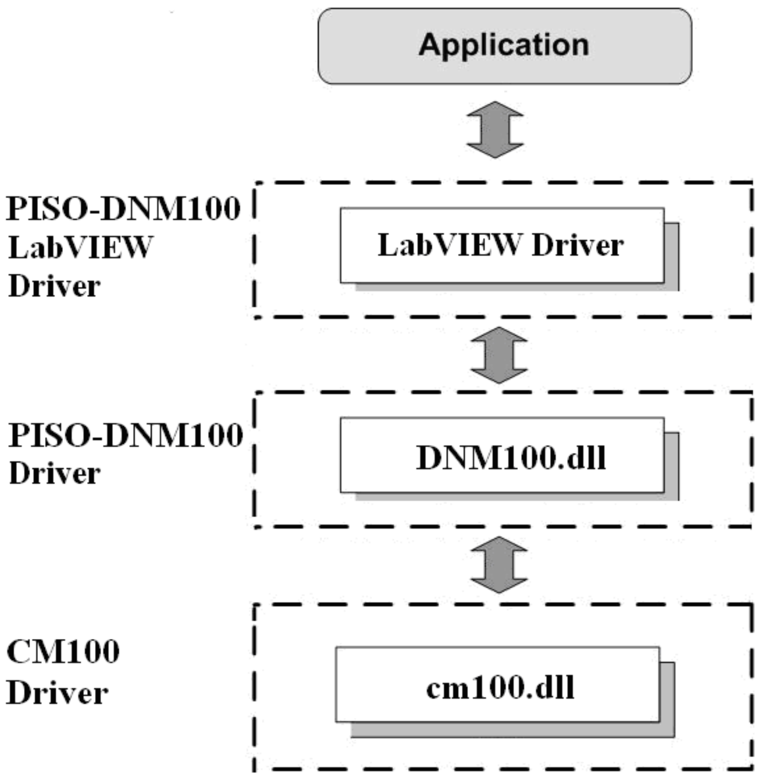


Figure 6.1 Driver concept of DNM100 LabVIEW driver

After completing the LabVIEW driver installation, the directory of DNM100 LabVIEW driver is C:\ICPDAS\PISO-DNM100\LabVIEW.

|--\Driver	LabVIEW driver
|--\Demo	LabVIEW demo program

DNM100 LabVIEW Function palette is showed as below.

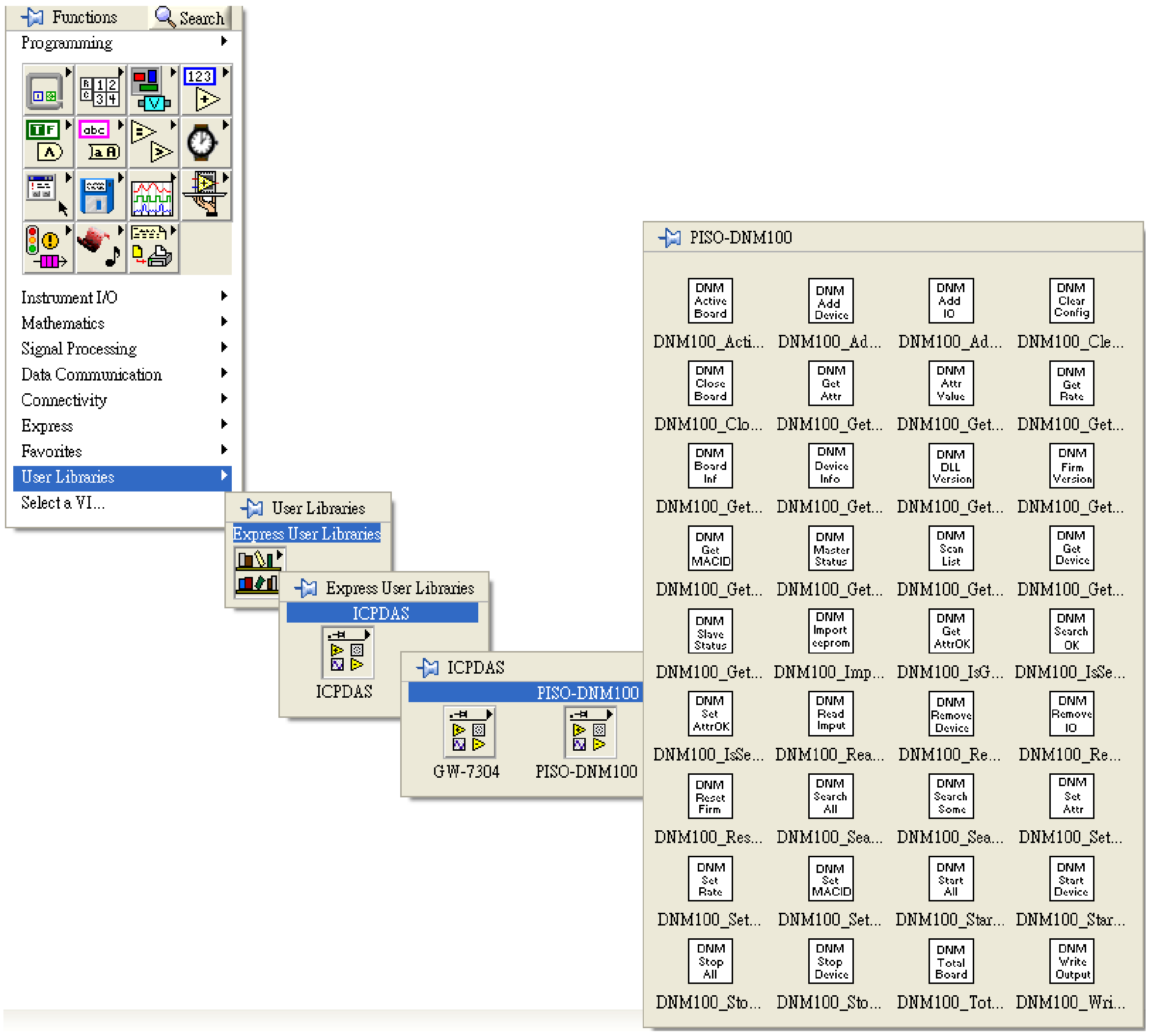


Figure 6.2 DNM100 LabVIEW Function palette

6.2 Function description

Every function provides each VI for user to use in the LabVIEW environment. All the functions provided by the DNM100.dll are listed in the following table.

Table 6.2.1 Board Functions

No.	VI ICON	Function Name	Description
1	DNM Board Inf	DNM100_GetBoardInf	Get the driver information of the DNM100 board
2	DNM Total Board	DNM100_TotalDNM100Board	Get total DNM100 boards in the PC
3	DNM Active Board	DNM100_ActiveBoard	Enable the driver of the DNM100 baord
4	DNM Close Board	DNM100_CloseBoard	Close driver of the DNM100 board
5	DNM DLL Version	DNM100_GetDLLVersion	Get the DLL version of the DNM100.DLL

Table 6.2.2 Firmware Functions

No.	VI ICON	Function Name	Description
1	DNM Firm Version	DNM100_GetFirmwareVersion	Get the version of the firmware inside the DNM100 board
2	DNM Reset Firm	DNM100_ResetFirmware	Reset the firmware in the DNM100 board

Table 6.2.3 Operating Functions 1/3

No.	VI ICON	Function Name	Description
1	DNM Set MACID	DNM100_SetMasterMACID	Set the MAC ID of the DNM100 board (DeviceNet Master's MAC ID)
2	DNM Get MACID	DNM100_GetMasterMACID	Get the MAC ID of the DNM100 board (DeviceNet Master's MAC ID)
3	DNM Get Rate	DNM100_GetBaudRate	Get the baud rate of the CAN bus

Table 6.2.4 Operating Functions 2/3

No.	VI ICON	Function Name	Description
17	DNM Attr Value	DNM100_GetAttributeValue	Get the attribute value of the DNM100_GetAttribute
18	DNM Set Attr	DNM100_SetAttribute	Send the set attribute command to the slave device.
19	DNM Set AttrOK	DNM100_IsSetAttributeOK	Check whether the slave has replied for the setting command or not.
20	DNM Device Info	DNM100_GetDeviceInfoFromScanList	Get specific slave device's I/O information form the Scan List within the DNM100 board.
21	DNM Scan List	DNM100_GetScanList	Get the I/O information of all slave devices form the Scan List within the DNM100 board.
22	DNM Import eeprom	DNM100_ImportEEPROM	Write the I/O information of all slave devices into the EEPROM within the DNM100 board.
23	DNM Clear Config	DNM100_ClearAllConfig	Clear all configurations in the EEPROM within the DNM100 board.

Table 6.2.5 Operating Functions 3/3

No.	VI ICON	Function Name	Description
4	DNM Set Rate	DNM100_SetBaudRate	Set the baud rate of the CAN bus
5	DNM Master Status	DNM100_GetMasterStatus	Get the status of the DNM100 board (DeviceNet Master's status) at present
6	DNM Slave Status	DNM100_GetSlaveStatus	Get the slave device's status.
7	DNM Start Device	DNM100_StartDevice	The DNM100 board will start to communicate with the specific slave device
8	DNM Stop Device	DNM100_StopDevice	The DNM100 board will stop to communicate with the specific slave device
9	DNM Start All	DNM100_StartAllDevice	The DNM100 board will start to communicate with all slave devices
10	DNM Stop All	DNM100_StopAllDevice	The DNM100 board will stop to communicate with all slave devices
11	DNM Add Device	DNM100_AddDevice	Add the specific slave device's information into the DNM100 board (DeviceNet Master)
12	DNM Remove Device	DNM100_RemoveDevice	Remove the specific slave device's information from the DNM100 board (DeviceNet Master)
13	DNM Add IO	DNM100_AddIOConnection	Add I/O information of the specific slave device into the DNM100 board (DeviceNet Master)
14	DNM Remove IO	DNM100_RemoveIOConnection	Remove specific slave device's I/O information from the DNM100 board (DeviceNet Master)
15	DNM Get Attr	DNM100_GetAttribute	Send the get attribute command to the slave device.
16	DNM Get AttrOK	DNM100_IsGetAttributeOK	Check whether the slave has replied for the getting command or not.

Table 6.2.6 Searching Functions

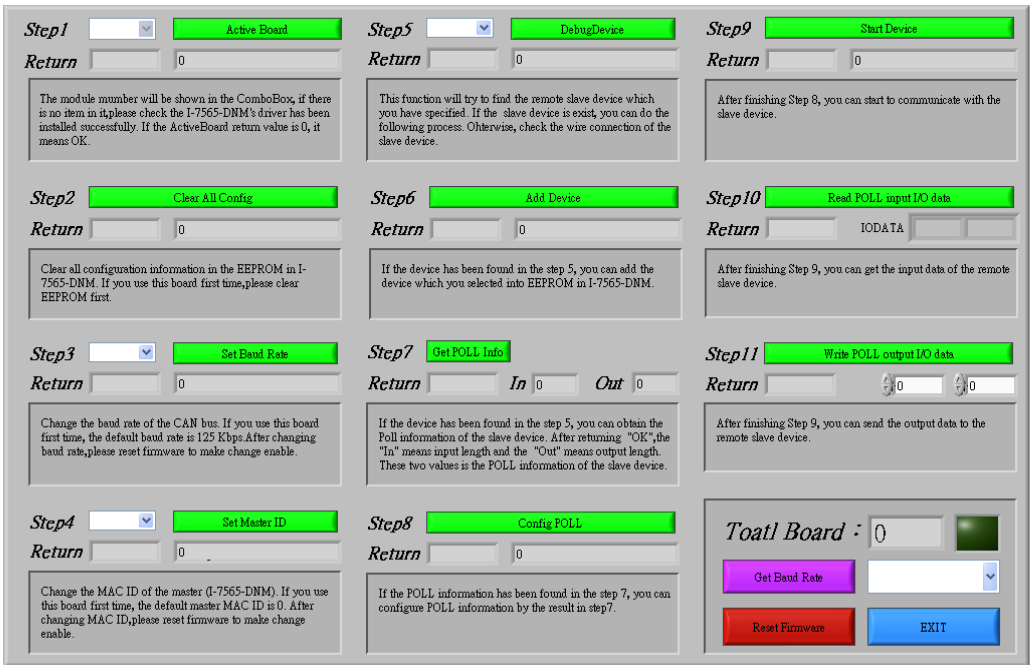
No.	VI ICON	Function Name	Description
1	DNM Search All	DNM100_SearchAllDevices	The DNM100 board will search the DeviceNet network to find out the I/O information of all slave devices.
2	DNM Search Some	DNM100_SearchSpecificDevice	The DNM100 board will search the DeviceNet network to find out the I/O information of specific slave devices.
3	DNM Search OK	DNM100_IsSearchOK	Check whether the DNM100 board has searched completely or not.
4	DNM Get Device	DNM100_GetSearchedDevices	Get the result of the searching command and retrieve the slave's I/O information.

Table 6.2.7 I/O Functions

No.	VI ICON	Function Name	Description
1	DNM Read Imput	DNM100_ReadInputData	Read the input data via I/O connection like Poll, Strobe, COS, Cyclic.
2	DNM Write Output	DNM100_WriteOutputData	Write the output data via I/O connection like Poll, COS, Cyclic. The Strobe doesn't support this operation.

6.3 LabVIEW Demo Introduction

The LabVIEW Demo is similar to the VC_Demo1. The screen shoot is shown as Figure 6.3.1. This demo program is designed to communicate with slave device step by step. All operating steps and detail descriptions can see in the section 5.3.



Step1: The module mumber will be shown in the ComboBox, if there is no item in it,please check the I-7565-DNM's driver has been installed successfully. If the ActiveBoard return value is 0, it means OK.

Step2: Clear all configuration information in the EEPROM in I-7565-DNM. If you use this board first time,please clear EEPROM first.

Step3: Change the baud rate of the CAN bus. If you use this board first time, the default baud rate is 125 Kbps.After changing baud rate,please reset firmware to make change enable.

Step4: Change the MAC ID of the master (I-7565-DNM). If you use this board first time, the default master MAC ID is 0. After changing MAC ID,please reset firmware to make change enable.

Step5: This function will try to find the remote slave device which you have specified. If the slave device is exist, you can do the following process. Ohterwise, check the wire connection of the slave device.

Step6: If the device has been found in the step 5, you can add the device which you selected into EEPROM in I-7565-DNM.

Step7: If the device has been found in the step 5, you can obtain the Poll information of the slave device. After returning "OK",the "In" means input length and the "Out" means output length. These two values is the POLL information of the slave device.

Step8: If the POLL information has been found in the step 7, you can configure POLL information by the result in step7.

Step9: After finishing Step 8, you can start to communicate with the slave device.

Step10: After finishing Step 9, you can get the input data of the remote slave device.

Step11: After finishing Step 9, you can send the output data to the remote slave device.

Figure 6.3.1 The screenshoot of LabVIEW Demo