

# I-7565-DNM

## USB / DeviceNet Master Converter

### User's Manual

#### **Warranty**

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

#### **Warning**

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

#### **Copyright**

Copyright 2009 by ICP DAS Co., LTD. All rights reserved worldwide.

#### **Trademark**

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

---

## Revision

| Version | Firmware Version | Date       | Author | Description   |
|---------|------------------|------------|--------|---|
| 1.9     | 1.9              | 2023/12/19 | Terry  | Update the installation steps   |
| 1.8     | 1.9              | 2021/10/04 | Johney | Add new function<br>1. I7565DNM_DisableKeepAliveMsg   |
| 1.7     | 1.6              | 2016/06/23 | Johney | *Update I7565DNM_TotalI7565DNMModule<br>*Add new functions<br>1. I7565DNM_PauseIOConnection<br>2. I7565DNM_ResumeIOConnection |
| 1.6     | 1.5              | 2015/09/15 | Johney | 1. Add C++ demo code  |
| 1.5     | 1.5              | 2013/10/01 | Johney | 1. Update the CAN pin description and CAN bus wire connection.  |
| 1.4     | 1.5              | 2013/06/04 | Johney | Add new function<br>1. I7565DNM_GetAttributeW<br>2. I7565DNM_SetAttributeW<br>3. I7565DNM_SendExplicitMSG_W                   |
| 1.3     | 1.3              | 2013/04/19 | Johney | 1.Update the supported OS.  |
| 1.2     | 1.2              | 2009 07/14 | Johney | Add new function<br>1. I7565DNM_ReadbackOutputData  |
| 1.1     | 1.1              | 2009 06/20 | Johney | Add new function<br>1. I7565DNM_SendExplicitMSG<br>2. I7565DNM_IsExplicitMSGRespOK<br>3. I7565DNM_GetExplicitMSGRespValue     |
| 1.0     | 1.0              | 2008 12/15 | Johney | This manual is for the I-7565-DNM module.   |

# Contents

|  |           |
|--|-----------|
| <b>REVISION.....</b>   | <b>2</b>  |
| 1.1 DEVICE <i>NET</i> INTRODUCTION .....   | 6         |
| 1.2 DEVICE <i>NET</i> APPLICATIONS.....  | 8         |
| 1.3 I-7565-DNM WITH VENDOR’S DEVICE <i>NET</i> SLAVES.....                       | 9         |
| 1.4 I-7565-DNM ARCHITECTURE.....   | 10        |
| 1.5 DEVICE <i>NET</i> MASTER CHARACTERISTICS .....                               | 11        |
| 1.6 I-7565-DNM FIRMWARE CHARACTERISTICS .....                                    | 14        |
| 1.7 HARDWARE & FIRMWARE FEATURES .....   | 16        |
| 1.8 BLOCK DIAGRAM.....   | 18        |
| 1.9 PRODUCT CHECK LIST .....   | 19        |
| 2.1 BOARD LAYOUT.....  | 20        |
| 2.2 JUMPER SELECTION.....  | 21        |
| 2.3 CONNECTOR PIN ASSIGNMENT.....  | 22        |
| 2.4 WIRE CONNECTION .....  | 23        |
| 2.5 INDICATOR LED.....   | 25        |
| 2.5.1 NS LED (Red) .....   | 25        |
| 2.5.2 RUN LED (Green) .....  | 26        |
| 2.5.3 MS LED (Yellow) .....  | 26        |
| 2.6 UPDATE FIRMWARE AND INIT/NORMAL SWITCH .....                                 | 27        |
| <b>DRIVER INSTALLATION AND SOFTWARE APPLICATION .....</b>                        | <b>28</b> |
| 3.1 DRIVER INSTALLATION OF THE I-7565-DNM .....                                  | 29        |
| 3.2 FLOW DIAGRAM FOR SEARCHING DEVICES.....                                      | 33        |
| 3.3 FLOW DIAGRAM FOR SLAVE CONFIGURATION .....                                   | 34        |
| 3.4 FLOW DIAGRAM FOR ON-LINE ADDING/REMOVING DEVICE.....                         | 35        |
| 3.5 FLOW DIAGRAM FOR “SETATTRIBUTE <i>W</i> ” AND “GETATTRIBUTE <i>W</i> ” ..... | 37        |
| 3.6 FLOW DIAGRAM FOR “SENEEXPLICITMSG_ <i>W</i> ” .....                          | 38        |
| 3.7 FLOW DIAGRAM FOR I/O CONNECTION.....   | 39        |
| 3.8 FLOW DIAGRAM FOR PAUSE AND RESUME I/O CONNECTION .....                       | 40        |
| <b>FUNCTION DESCRIPTION .....</b>  | <b>41</b> |
| 4.1 DLL FUNCTION DEFINITION AND DESCRIPTION .....                                | 42        |
| 4.2 FUNCTION RETURN CODE .....   | 46        |
| 4.3 FUNCTION DESCRIPTION .....   | 49        |
| 4.3.1 I7565DNM_TotalI7565DNMModule .....   | 49        |
| 4.3.2 I7565DNM_ActiveModule.....   | 50        |
| 4.3.3 I7565DNM_CloseModule .....   | 51        |

---

|                   |   |               |
|-------------------|---|---------------|
| 4.3.4             | I7565DNM_GetDLLVersion .....              | 52            |
| 4.3.5             | I7565DNM_GetFirmwareVersion.....          | 53            |
| 4.3.6             | I7565DNM_ResetFirmware .....              | 54            |
| 4.3.7             | I7565DNM_GetMasterMACID .....             | 55            |
| 4.3.8             | I7565DNM_SetMasterMACID .....             | 56            |
| 4.3.9             | I7565DNM_GetBaudRate .....                | 57            |
| 4.3.10            | I7565DNM_SetBaudRate.....                 | 58            |
| 4.3.11            | I7565DNM_GetMasterStatus .....            | 59            |
| 4.3.12            | I7565DNM_GetSlaveStatus .....             | 60            |
| 4.3.13            | I7565DNM_StartDevice.....                 | 61            |
| 4.3.14            | I7565DNM_StopDevice .....                 | 62            |
| 4.3.15            | I7565DNM_StartAllDevice.....              | 63            |
| 4.3.16            | I7565DNM_StopAllDevice.....               | 64            |
| 4.3.17            | I7565DNM_AddDevice .....                  | 65            |
| 4.3.18            | I7565DNM_RemoveDevice .....               | 66            |
| 4.3.19            | I7565DNM_AddIOConnection .....            | 67            |
| 4.3.20            | I7565DNM_RemoveIOConnection.....          | 69            |
| <del>4.3.21</del> | <del>I7565DNM_GetAttribute .....</del>    | <del>70</del> |
| 4.3.22            | I7565DNM_GetAttributeW.....               | 71            |
| 4.3.23            | I7565DNM_IsGetAttributeOK.....            | 74            |
| 4.3.24            | I7565DNM_GetAttributeValue.....           | 76            |
| <del>4.3.25</del> | <del>I7565DNM_SetAttribute .....</del>    | <del>78</del> |
| 4.3.26            | I7565DNM_SetAttributeW .....              | 79            |
| 4.3.27            | I7565DNM_IsSetAttributeOK .....           | 81            |
| 4.3.28            | I7565DNM_ClearAllConfig.....              | 83            |
| 4.3.29            | I7565DNM_SearchAllDevices.....            | 84            |
| 4.3.30            | I7565DNM_SearchSpecificDevice .....       | 85            |
| 4.3.31            | I7565DNM_IsSearchOK .....                 | 86            |
| 4.3.32            | I7565DNM_GetSearchedDevices .....         | 87            |
| 4.3.33            | I7565DNM_GetDeviceInfoFromScanList .....  | 88            |
| 4.3.34            | I7565DNM_GetScanList.....                 | 89            |
| 4.3.35            | I7565DNM_ImportEEPROM.....                | 90            |
| 4.3.36            | I7565DNM_ReadInputData .....              | 91            |
| 4.3.37            | I7565DNM_WriteOutputData.....             | 93            |
| <del>4.3.38</del> | <del>I7565DNM_SendExplicitMSG .....</del> | <del>95</del> |
| 4.3.39            | I7565DNM_SendExplicitMSG_W.....           | 96            |
| 4.3.40            | I7565DNM_IsExplicitMSGRespOK.....         | 97            |

---

---

|  |   |            |
|--|---|------------|
| 4.3.41                                   | I7565DNM_GetExplicitMSGRespValue .....                | 98         |
| 4.3.42                                   | I7565DNM_ReadbackOutputData .....                     | 99         |
| 4.3.43                                   | I7565DNM_PauseIOConnection.....                       | 100        |
| 4.3.44                                   | I7565DNM_ResumeIOConnection .....                     | 101        |
| 4.3.45                                   | I7565DNM_DisableKeepAliveMsg (Advanced Option).....   | 102        |
| <b>DEMO PROGRAMS FOR WINDOWS.....</b>    |   | <b>103</b> |
| 5.1                                      | <i>A BRIEF INTRODUCTION TO THE DEMO PROGRAMS.....</i> | 103        |
| 5.2                                      | <i>WIRE CONNECTION OF THE CAN BUS.....</i>            | 104        |
| 5.3                                      | <i>VC_DEMO1 INTRODUCTION .....</i>                    | 105        |
| 5.4                                      | <i>VC_DEMO2 INTRODUCTION .....</i>                    | 108        |
| 5.5                                      | <i>BCB_DEMO1 INTRODUCTION.....</i>                    | 110        |
| 5.6                                      | <i>BCB_DEMO2 INTRODUCTION.....</i>                    | 112        |
| <b>LABVIEW DRIVER INTRODUCTION .....</b> |   | <b>114</b> |
| 1.1                                      | <i>SOFTWARE INSTALLATION.....</i>                     | 114        |
| 1.2                                      | <i>FUNCTION DESCRIPTION.....</i>                      | 116        |
| 1.3                                      | <i>LABVIEW DEMO INTRODUCTION .....</i>                | 120        |

---

## 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.

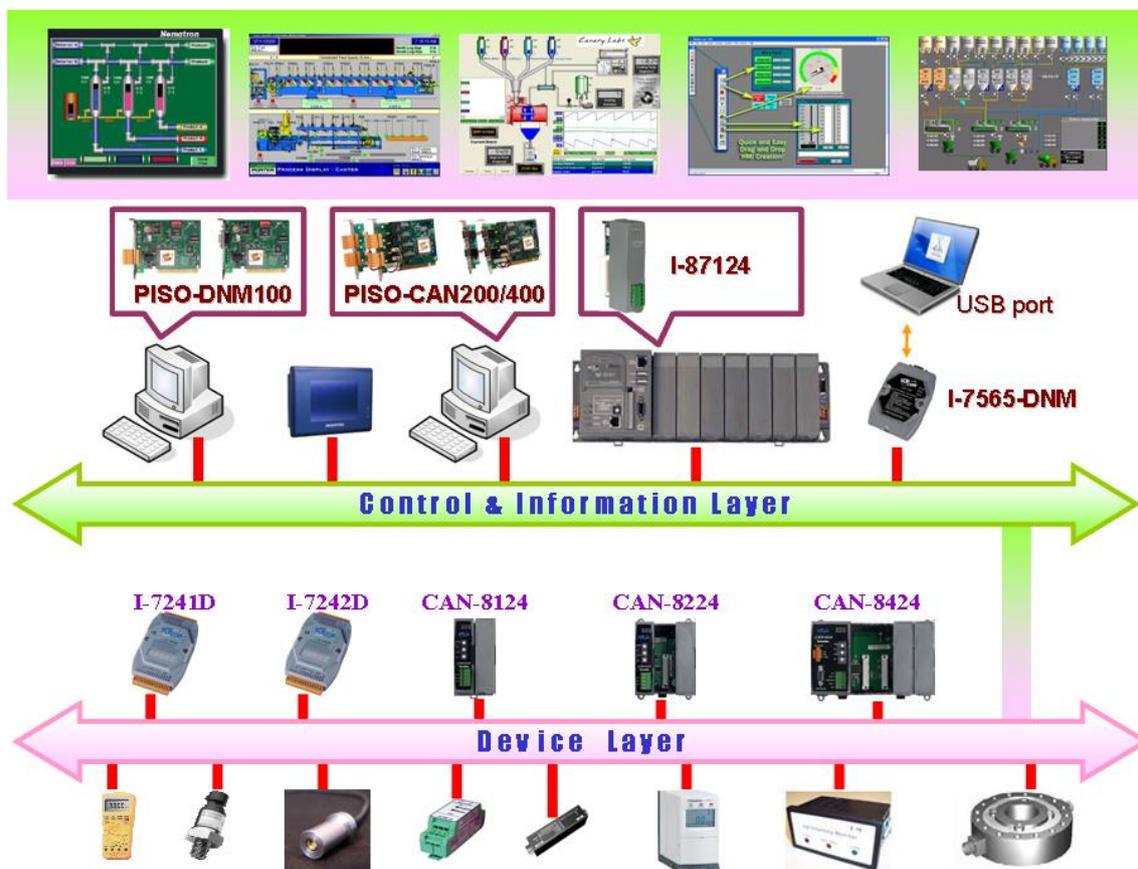


Figure 1.1 Example of the DeviceNet network

DeviceNet is a cost effective solution to one kind application of control

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.

| <b>Baud rate (bit/s)</b> | <b>Max. Bus length (m)</b> |
|--------------------------|----------------------------|
| 500 K                    | 100                        |
| 250 K                    | 250                        |
| 125 K                    | 500                        |

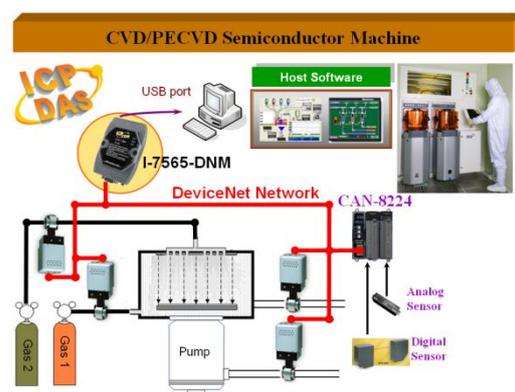
Table 1.1 The Baud rate and the Bus length

I-7565-DNM can represent an economic solution of DeviceNet application and be a DeviceNet master device on the DeviceNet network. I-7565-DNM 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, I-7565-DNM uses the new CAN controller Phillips SJA1000T and transceiver 82C250, which provide bus arbitration, error detection with auto correction and re-transmission function. It can be installed on almost any windows-based system, for example WinXP/Win7/Win10/Linux. 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 I-7565-DNM.

## 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](http://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
- T-shirt manufacturing



## 1.3 I-7565-DNM with Vendor's DeviceNet Slaves

We have communicated with the following DeviceNet slaves.

- Allen-Bradley PowerFlex series DeviceNet Inverters.
- BECKHOFF CX1500-B520 series DeviceNet I/O modules.
- BECKHOFF BK5250 series DeviceNet I/O modules.
- 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.
- OMRON DRT1-ID/ODxx series DeviceNet I/O modules.
- OMRON DRT2-MDxx series DeviceNet I/O modules.
- COSMOS PS-7 series DeviceNet gas detectors.
- CELERITY UNIT IFC-125 series DeviceNet devices.
- Allen-Bradley PowerFlex AC Drives / DC Drivers
- Allen-Bradley PowerFlex AC Drives with DriveLogix
- OMRON DRT2-ID08(-1)/MD16(-1)/OD08(-1)
- OMRON DRT2-ID16(-1)/OD16(-1)
- OMRON GRT1-DRT
- OMRON C200HW-DRT21
- 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
- Weidmueller SAI-AU M12 DN 16DI/8DO/AI/AO
- Weidmueller SAI-AU M12 DN GW
- ADVANCED ENERGY Apex RF generators and power-delivery
- SMC ITV series Electro-Pneumatic Regulator
- SMC Directional Control Valves
- SICK DME500 series Distance Sensor
- MTS Temposonics R-Series Position Sensors
- PFEIFFER VACUUM HiPace series turbo pumps

## 1.4 I-7565-DNM Architecture

The I-7565-DNM provides users to establish DeviceNet network rapidly by Master/Slave connection model. The I-7565-DNM is a high-performance DeviceNet master board with one CPU inside. This architecture of the I-7565-DNM almost doesn't cost CPU resource and really increases the work efficiency on DeviceNet network. Applying the I-7565-DNM, 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.

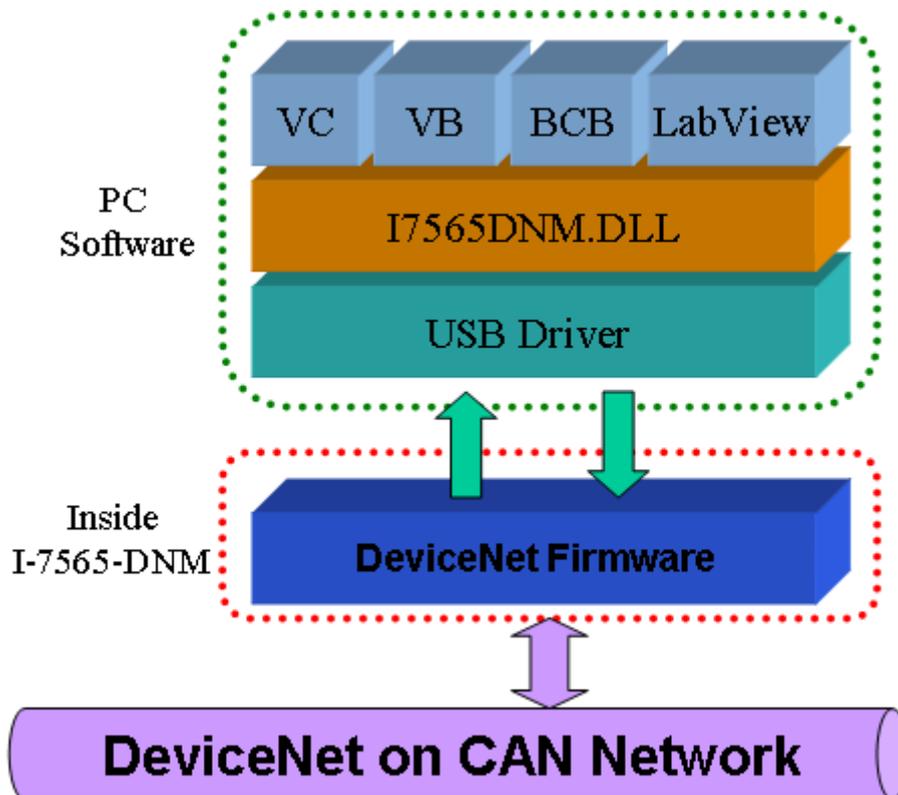


Figure 1.2 I-7565-DNM architecture.

## 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.

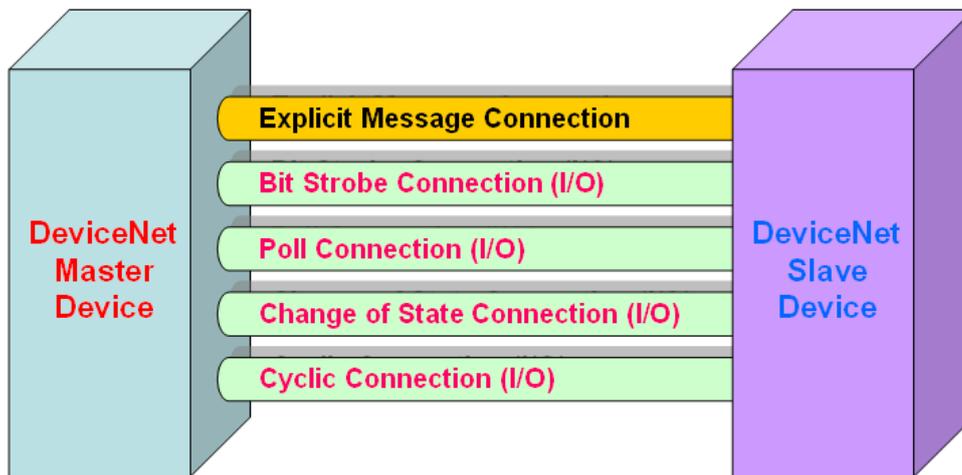


Figure 1.4 DeviceNet Messaging

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:

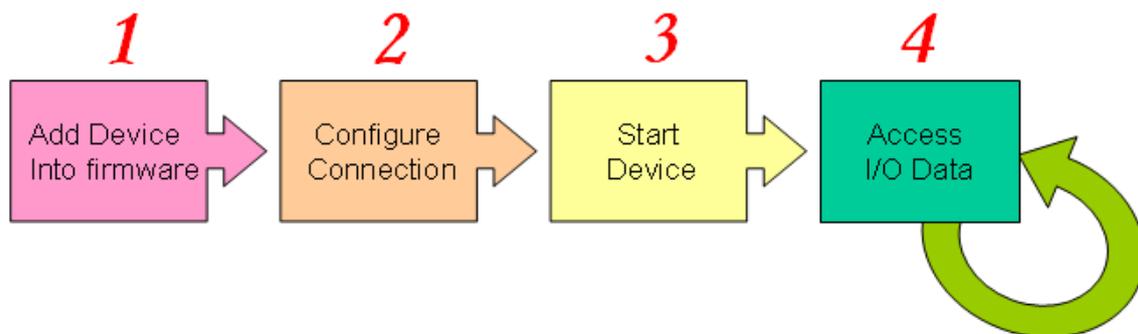


Figure 1.5 Four steps to establish connection

---

### 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 I-7565-DNM Firmware Characteristics

The I-7565-DNM is a high-performance DeviceNet master module. The firmware inside the board implements DeviceNet protocol automatically when the module 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.

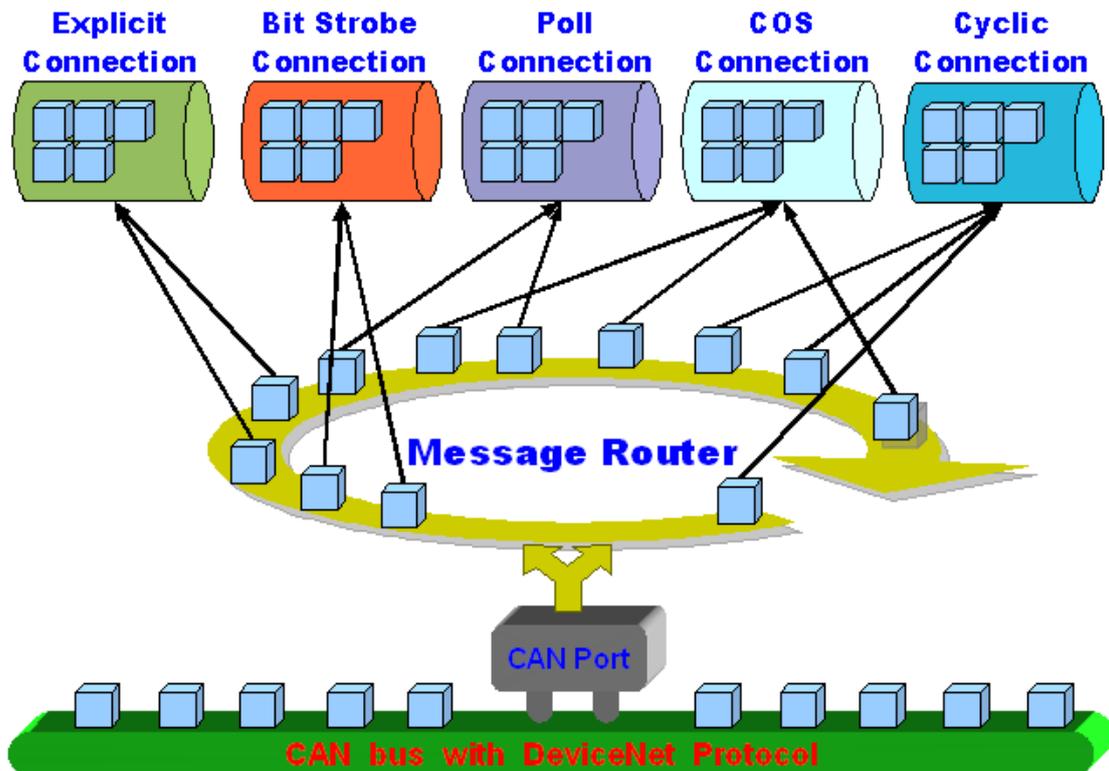


Figure 1.6 Message Router

The I-7565-DNM 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.

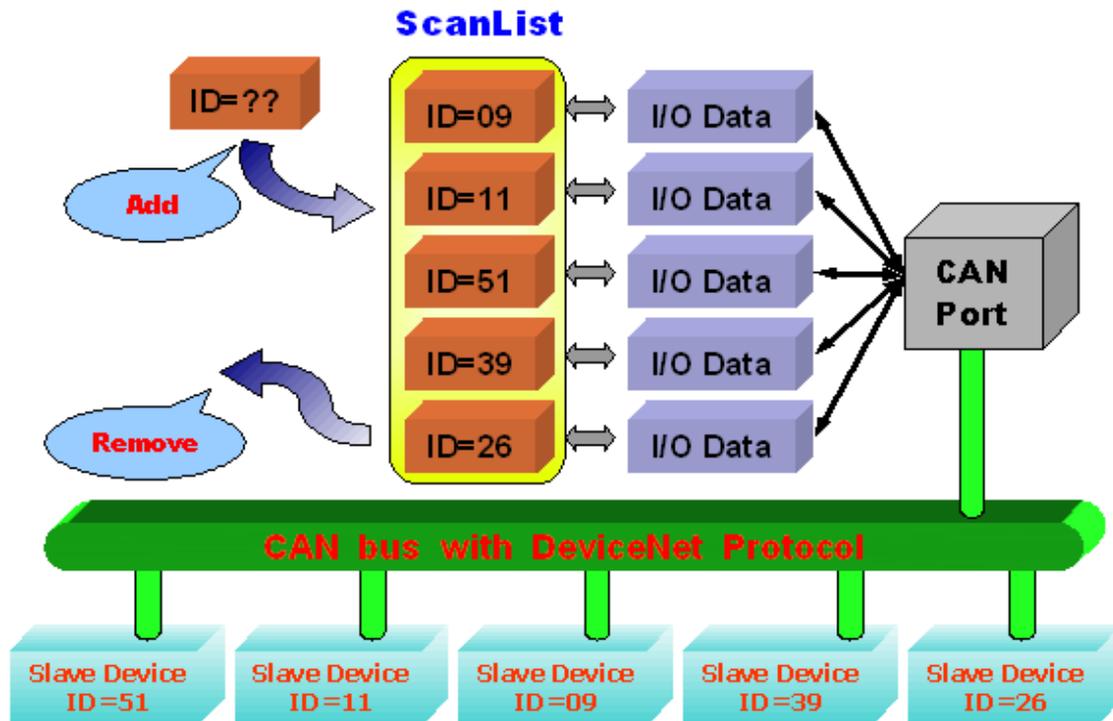


Figure 1.7 ScanList data structure

## 1.7 Hardware & Firmware Features

### Hardware Features

- USB interface connector: USB Type B.
- CAN controller: Philip SJA1000T.
- CAN transceiver: Philip 82C250.
- Signal support: CAN\_H, CAN\_L.
- Power requirements: USB(5V@200mA).
- Power consumption : 1W.
- CAN interface connector: 9-pin D-Sub male.
- 80186-80 MHz CPU
- 512K bytes SRAM.
- 16K bytes EEPROM.
- Built-in watchdog timer.
- It is powered by USB bus.
- 4 indicating LED (RUN, NS, MS and Power).
- Baud rate of USB is 921.6 Kbps.
- Jumper select 120Ω terminator resistor for CAN port.
- 2500Vrms photo-isolation protection on CAN bus.
- 3000Vrms galvanic DC/DC isolation on CAN side.
- Driver supported for WinXP/Win 7 /Win 10.
- Linux driver supported.
- Environmental:

Operating temp: -25 ~ +75°C

Storage temp: -30 ~ +80°C

Humidity: 5% ~ 95% non-condensing

Dimensions: 108mm x 72mm x 33mm (H x W x D)

### DeviceNet Firmware Features

- Programmable Master MAC ID.
  - Programmable transfer-rate 125K, 250K, 500K.
  - Each port support maximum nodes up to 64
-

- 
- Support Group 2 Only Server functions
  - Support UCMM functions
  - Predefined Master-Slave Connection Set
  - The maximum Fragment number is (Input/Output) up to 64
  - Support I/O Operation Mode: Poll, Bit-Strobe and Change Of State/Cyclic
  - Support Auto-Scan slave device function.
  - Support on-line adding and removing devices.
  - Support Auto-Reconnect when the connection is broken.
-

## 1.8 Block Diagram

The figure 1.8 shows the block diagram of the I-7565-DNM board.

1. USB Driver :

The USB port provides the communication channel between PC and I-7565-DNM.

2. EEPROM :

The EEPROM stores the configuration information. After restarting the PC, the configuration data will be loaded from 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.

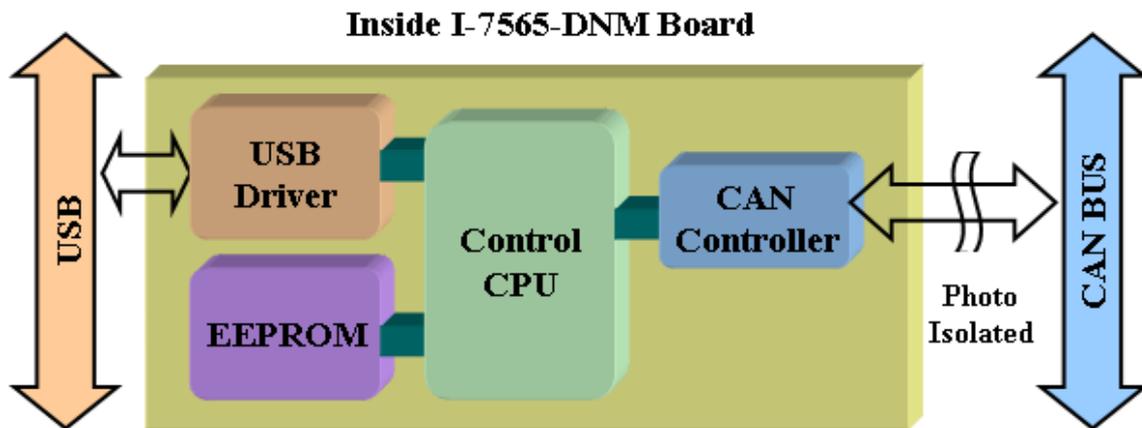


Figure 1.8 Block diagram of the I-7565-DNM

---

## 1.9 Product Check List

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

- I-7565-DNM module;
- Software CD ROM;
- Quick Start manual;

**It is recommended that users should read the quick start manual first. There shows how to get start quickly. All of the important information needed will be provided in this manual and website 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.

---

## Hardware Configuration

This section will describe the hardware settings of the I-7565-DNM. This information includes the wire connection and terminal resistance configuration for the CAN network.

### 2.1 Board Layout

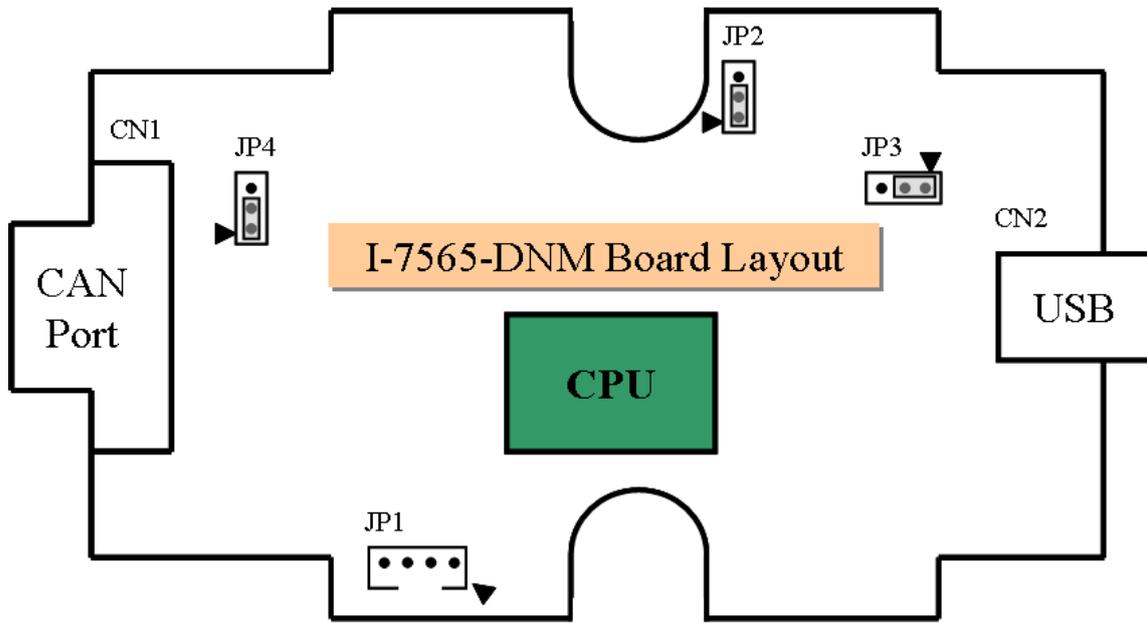


Figure2.1 I-7565-DNM Board LAYOUT

## 2.2 Jumper Selection

The following table shows the definition of jumpers. Users need to refer to this table to configure the I-7565-DNM hardware.

| Jumper | Description                                    | Status   |
|--------|--|--|
| JP1    | No use.  | None   |
| JP2    | No use.<br>(Don't change the default setting.) |   |
| JP3    | No use.<br>(Don't change the default setting.) |   |
| JP4    | CAN Port 120Ω terminal resistance.             | <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">  <p>Enable</p> </div> <div style="text-align: center;">  <p>Disable</p> </div> </div> |

Table 2.1 Jumper selections

## 2.3 Connector Pin Assignment

The I-7565-DNM 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:

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 2.2.

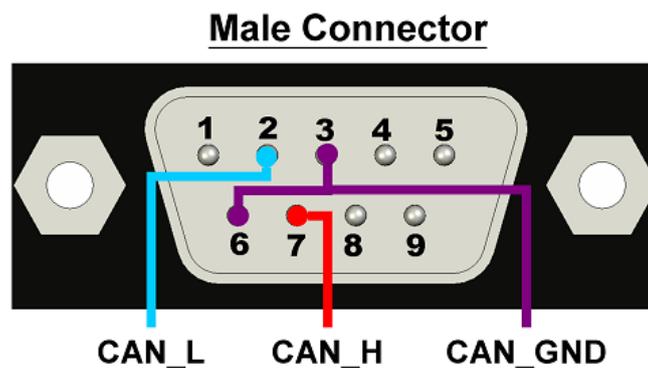


Figure2.5 9-pin D-sub male connector

| 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 2.2 Pin assignment of the 9-pin D-sub male connector

## 2.4 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  $120\Omega$  (or between  $108\Omega\sim 132\Omega$ ). The length related resistance should have  $70\text{ m}\Omega/\text{m}$ . Users should check the resistances of the CAN bus, before they install a new CAN network.

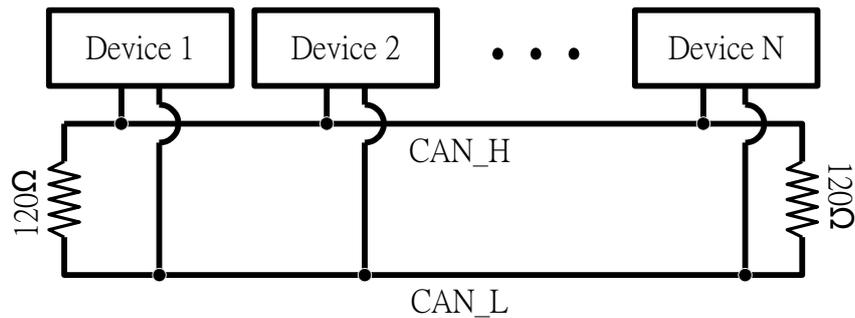


Figure 2.4 CAN bus network topology

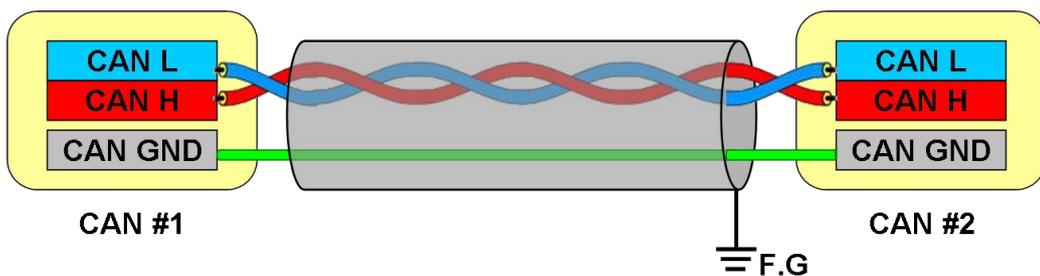


Figure 2.5 CAN bus wire connection without DC power

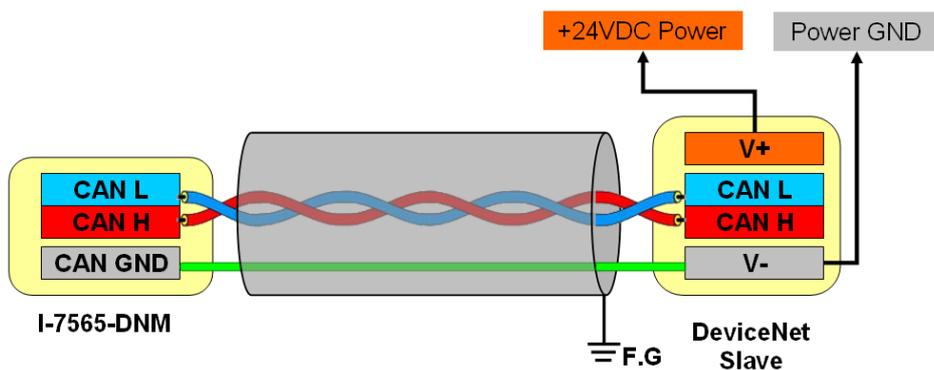


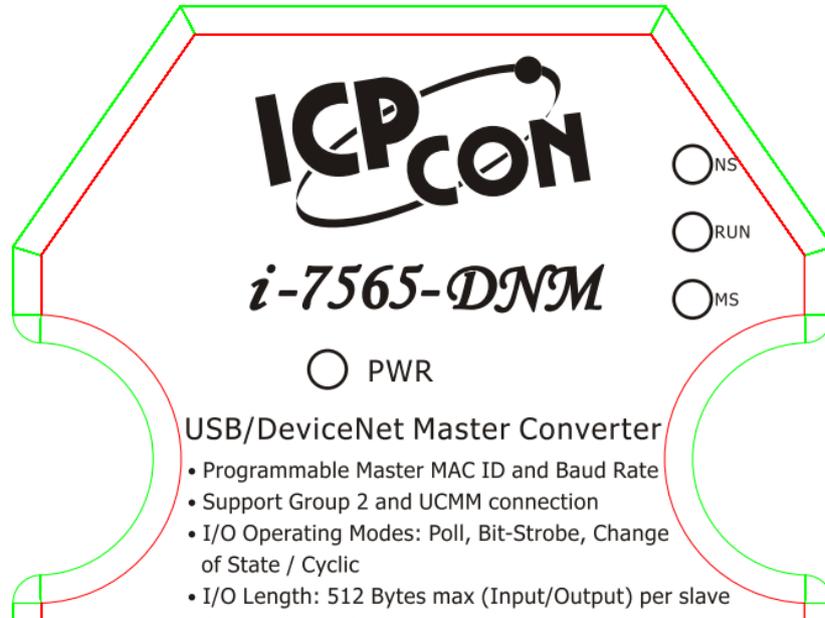
Figure 2.6 CAN bus wire connection with 24VDC power

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<br>(meter) | Bus Cable Parameters                            |   | Terminal<br>Resistance<br>( $\Omega$ ) |
|-----------------------|---|---|--|
|                       | Length Related<br>Resistance<br>(m $\Omega$ /m) | Cross Section<br>(Type)                     |  |
| 0~40                  | 70  | 0.25(23AWG)~<br>0.34mm <sup>2</sup> (22AWG) | 124 (0.1%)                             |
| 40~300                | < 60  | 0.34(22AWG)~<br>0.6mm <sup>2</sup> (20AWG)  | 127 (0.1%)                             |
| 300~600               | < 40  | 0.5~0.6mm <sup>2</sup><br>(20AWG)           | 150~300                                |
| 600~1K                | < 20  | 0.75~0.8mm <sup>2</sup><br>(18AWG)          | 150~300                                |

Table 2.4 Relationship between cable characteristics and terminal resistance

## 2.5 Indicator LED



### 2.5.1 NS LED (Red)

The [NS] 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 I-7565-DNM's MAC ID. There are two situations in [NS] LED.

(1). LED off:

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

(2). LED twinkle (Red) :

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

- The CAN connector doesn't connect to the slave devices.
- The power of the slave devices is off.
- The MAC ID collision between master and slave devices is occurring.

---

### **2.5.2 RUN LED (Green)**

The [RUN] LED indicates the I-7565-DNM's firmware status. There are three situations in [RUN] LED.

(1). LED off :

This indicates that there are some errors on the bus or in the I-7565-DNM module. The DeviceNet firmware is not running.

(2). LED twinkle (Green) :

This indicates that the CAN bus works fine. But there is no any slave devices configuration in the I-7565-DNM's EEPROM. The DeviceNet firmware is waiting for configuration.

(3). LED solid on (Green) :

This indicates that the DeviceNet firmware is running. The I-7565-DNM module is communicating with the slave devices.

### **2.5.3 MS LED (Yellow)**

The [MS] LED means Module Status. It indicates any slave devices which is disconnecting with the I-7565-DNM module. There are two situations in [MS] LED.

(1). LED off :

This shows that all of the slave devices are communicating with the I-7565-DNM normally.

(2). LED twinkle (Yellow) :

This shows that at least one slave device occurs communication errors. Maybe the configuration errors or slave devices errors happened.

---

## 2.6 Update firmware and Init/Normal Switch

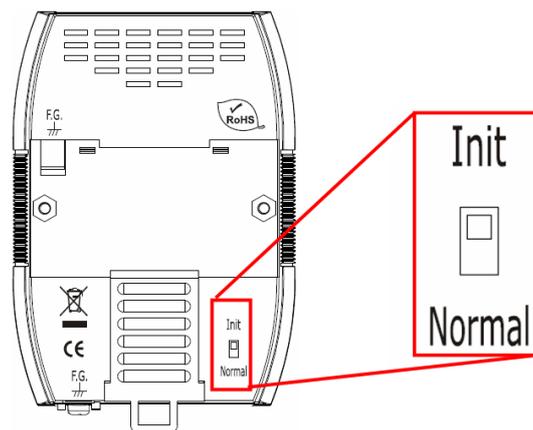
When the users want to download the new firmware into the I-7565-DNM, the users need to follow the steps as described below.

Step 1: Close all programs which are communicating with the I-7565-DNM.

Step 2: Unplug the USB wire of the I-7565-DNM from the USB port.

Step 3: The Init/Normal switch locates on the reverse side of the I-7565-DNM.

Switch it to the “Init” mode as shown below.



Step 4: Plug the USB wire of the I-7565-DNM into the USB port.

Step 5: The users would see those three indicators LED turn on one by one.

Step 6: Open the DNM\_Utility software located at C:\ICPDAS\DNM\_Utility

If you do not find DNM\_Utility in your PC, please install it at the path.

1. Fieldbus CD : \DeviceNet\Master\DNM\_Utility\

2. Website :

<https://www.icpdas.com/en/download/index.php?model=I-7565-DNM-G>

Step 7: Follow the process of updating firmware described in the manual of the DNM\_Utility.

## Driver Installation and Software Application

The DeviceNet DLL driver (I7565DNM.dll) collection of function calls for the I-7565-DNM module 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 I7565DNM.DLL driver to implement DeviceNet network application. And then the DeviceNet DLL driver will throughout the [UART.DLL] into the [SER2PL.SYS] to access the hardware system, as shown in the following Figure.

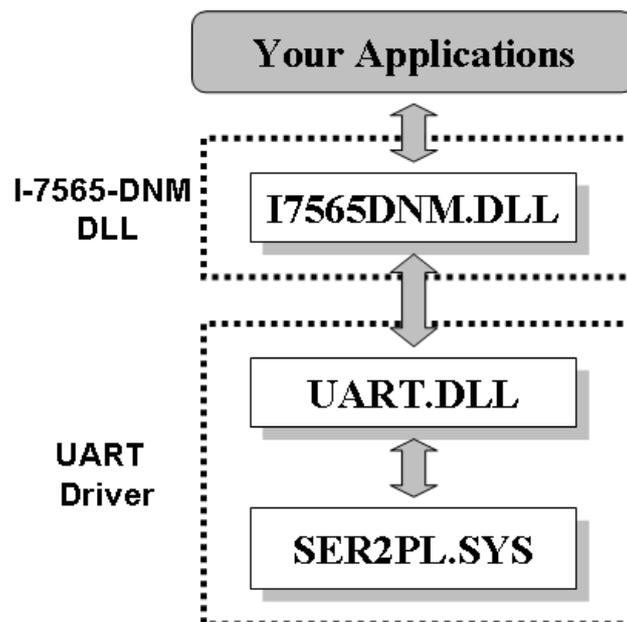


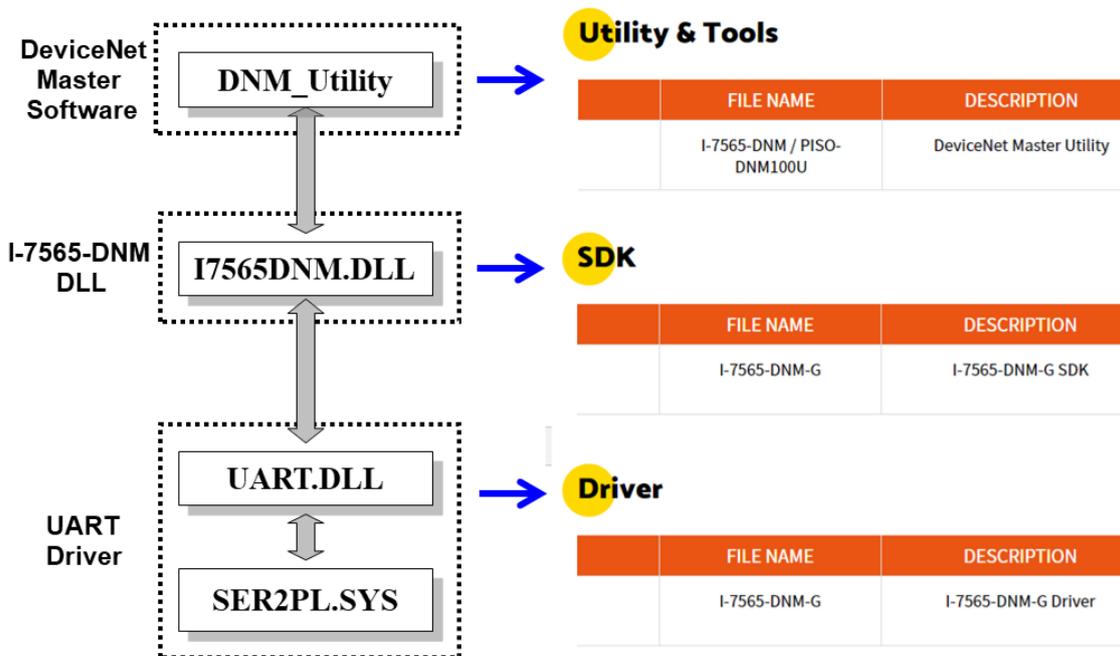
Figure 3.1 Software architecture in the Windows system

In the following sub-section, we show some flow diagrams to describe how to apply the DeviceNet protocol (I7565DNM.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.

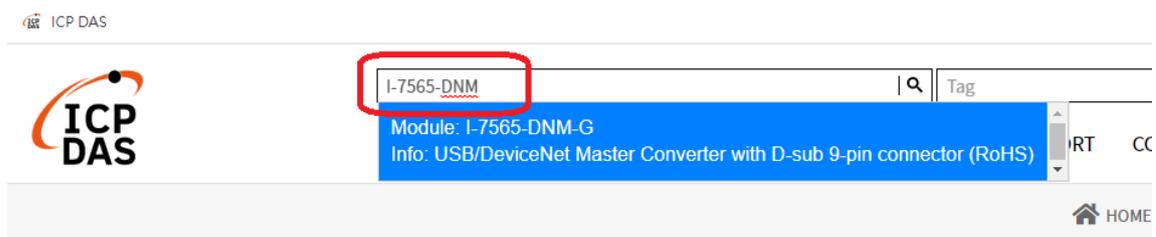
### 3.1 Driver Installation of the I-7565-DNM

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.

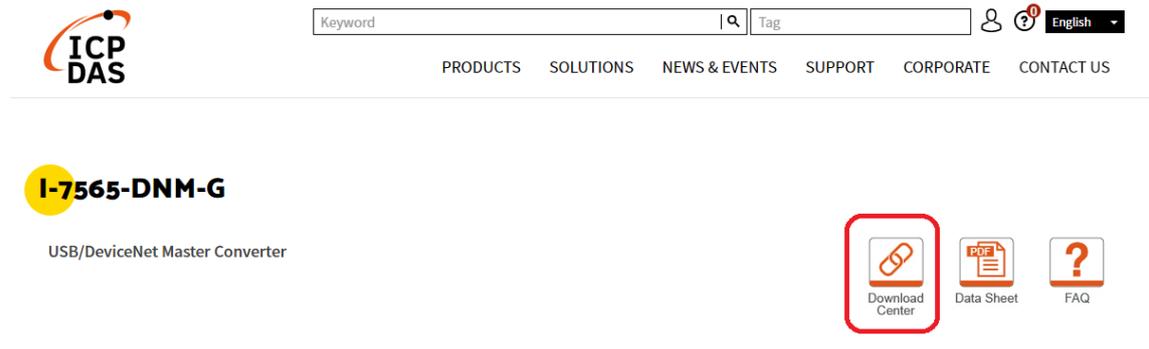
The driver of I-7565-DNM can be used in Windows environments. There are three installations. Here shows the illustration below.



Step 1: The users can visit the ICPDAS's website and search the "I-7565-DNM".

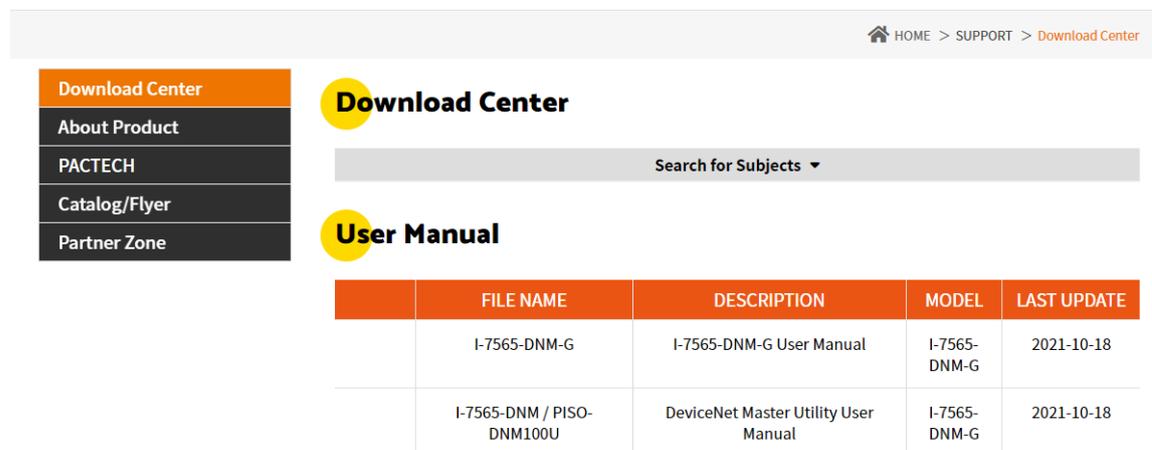


Step 2: Click the “Download Center” icon to visit the download page of the I-7565-DNM.



Step 3: Visit the “Download Center” page of the I-7565-DNM. The users also can visit the link below.

<https://www.icpdas.com/en/download/index.php?model=I-7565-DNM-G>



For these Windows operation systems, the recommended installation procedure is given as follows:

Step 1: Install the USB driver of the I-756x.

## Driver

|  | FILE NAME    | DESCRIPTION         | MODEL        | LAST UPDATE |
|--|--------------|---------------------|--------------|-------------|
|  | I-7565-DNM-G | I-7565-DNM-G Driver | I-7565-DNM-G | 2020-04-20  |

Step 2: Install the SDK of the I-7565-DNM which including the I7565DNM.dll and other development files.

### SDK

|  | FILE NAME    | DESCRIPTION      | MODEL        | LAST UPDATE |
|--|--------------|------------------|--------------|-------------|
|  | I-7565-DNM-G | I-7565-DNM-G SDK | I-7565-DNM-G | 2021-10-18  |

Step 3: Install the DNM\_Utility for all DeviceNet master products. DeviceNet Master Utility is a useful tool for users to configure and test the DeviceNet slave devices. The users can download the manual of the DNM\_Utility to read more information. After installing the software, the utility is installed in the path below.

### Utility & Tools

|  | FILE NAME                 | DESCRIPTION              | MODEL        | LAST UPDATE |
|--|---------------------------|--------------------------|--------------|-------------|
|  | I-7565-DNM / PISO-DNM100U | DeviceNet Master Utility | I-7565-DNM-G | 2021-10-18  |

Step 4: After installing those three installations, please restart your PC.

Then the installations 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.

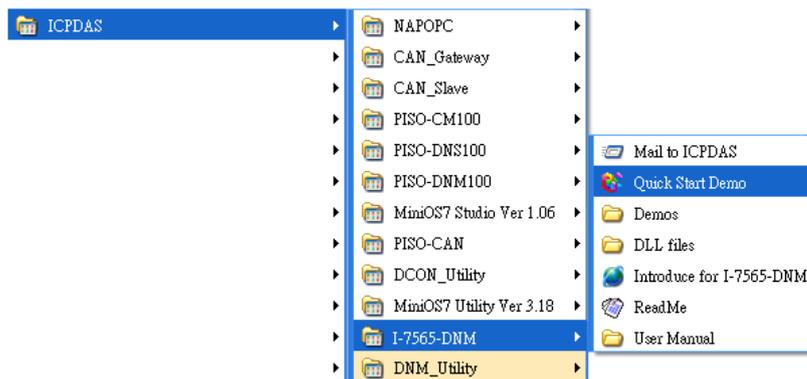
Windows XP – **WINDOWS\SYSTEM32\DRIVERS**

Windows 7/10 – **WINDOWS\SYSWOW64\DRIVERS**

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

**C:\ICPDAS\ I-7565-DNM \**

The program files picture is shown as follow.



### 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 with other slave devices is fine, the users 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.

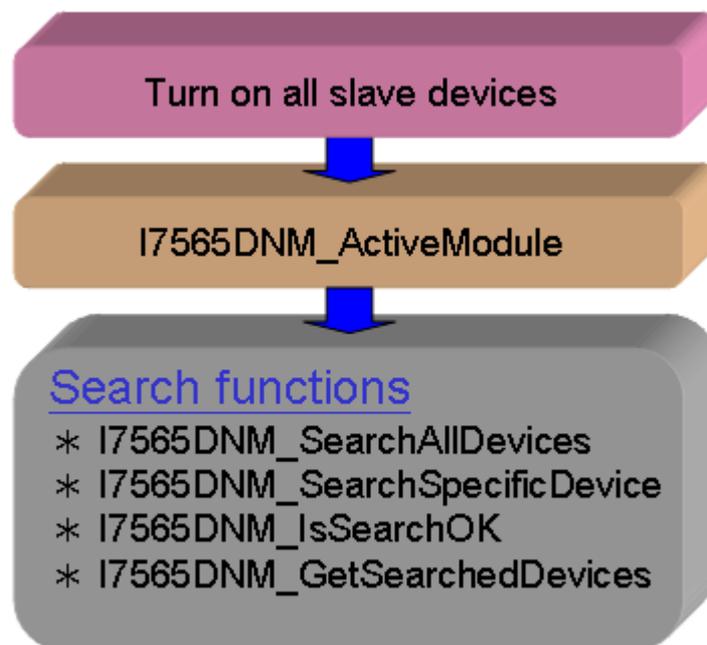


Figure 3.2 Searching Diagram

### 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 within the I-7565-DNM module. The firmware in the I-7565-DNM module 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.

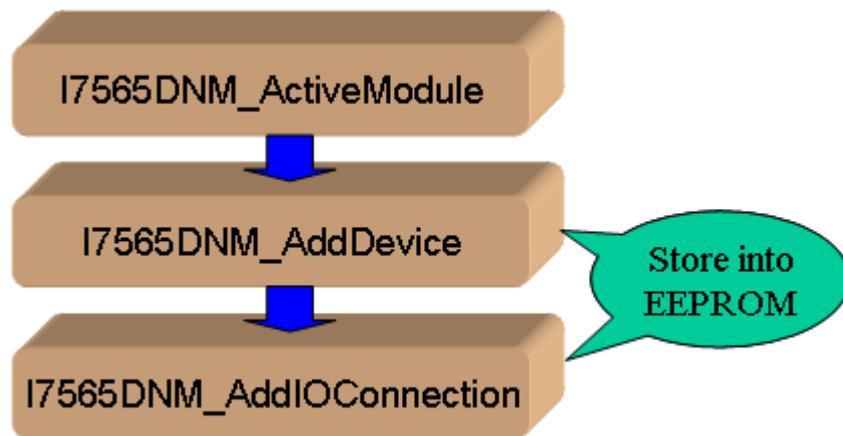


Figure 3.3 Slave Configuration Diagram

### 3.4 Flow Diagram for On-line Adding/Removing Device

The I-7565-DNM provides the on-line adding/removing slave device functions. The users need not to break the communication between original slaves device 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 :

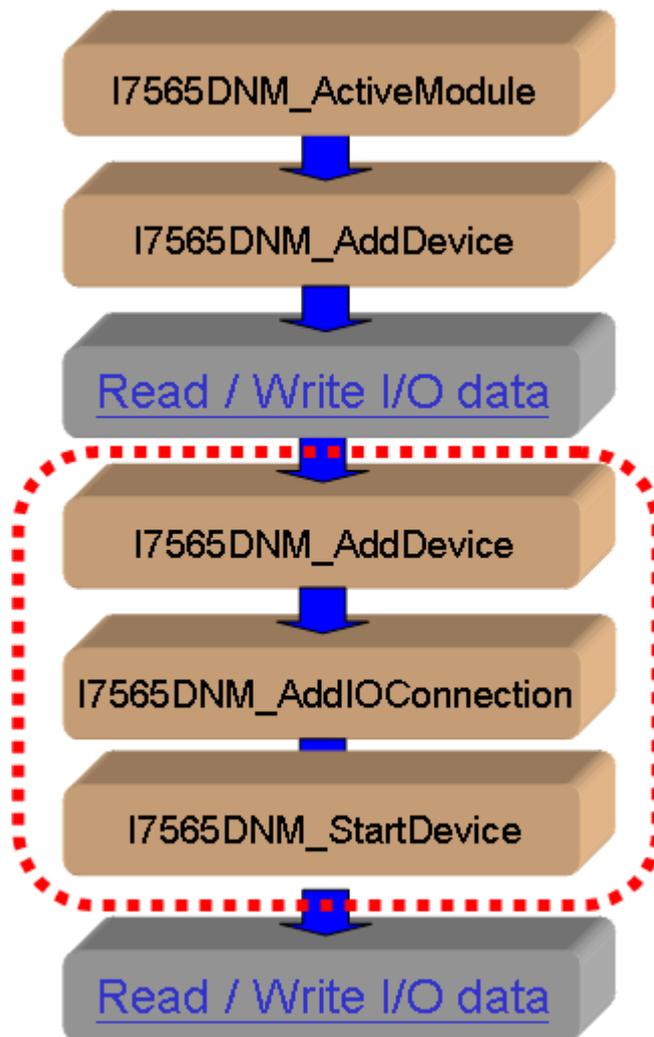


Figure 3.6 On-line Add Device Diagram

2. On-line Removing Devices :

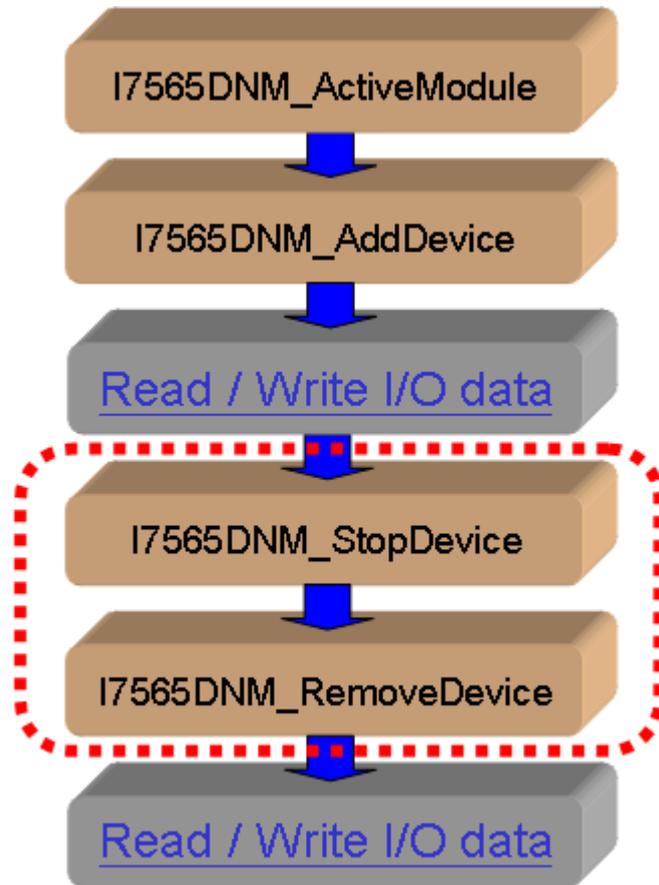


Figure 3.7 On-line Remove Device Diagram

### 3.5 Flow Diagram for “SetAttributeW” and “GetAttributeW”

The users can set or get DeviceNet device's property via DeviceNet network. The I-7565-DNM provides these functions to set or get the properties of the remote devices easily. The steps are shown in Figure 3.8.

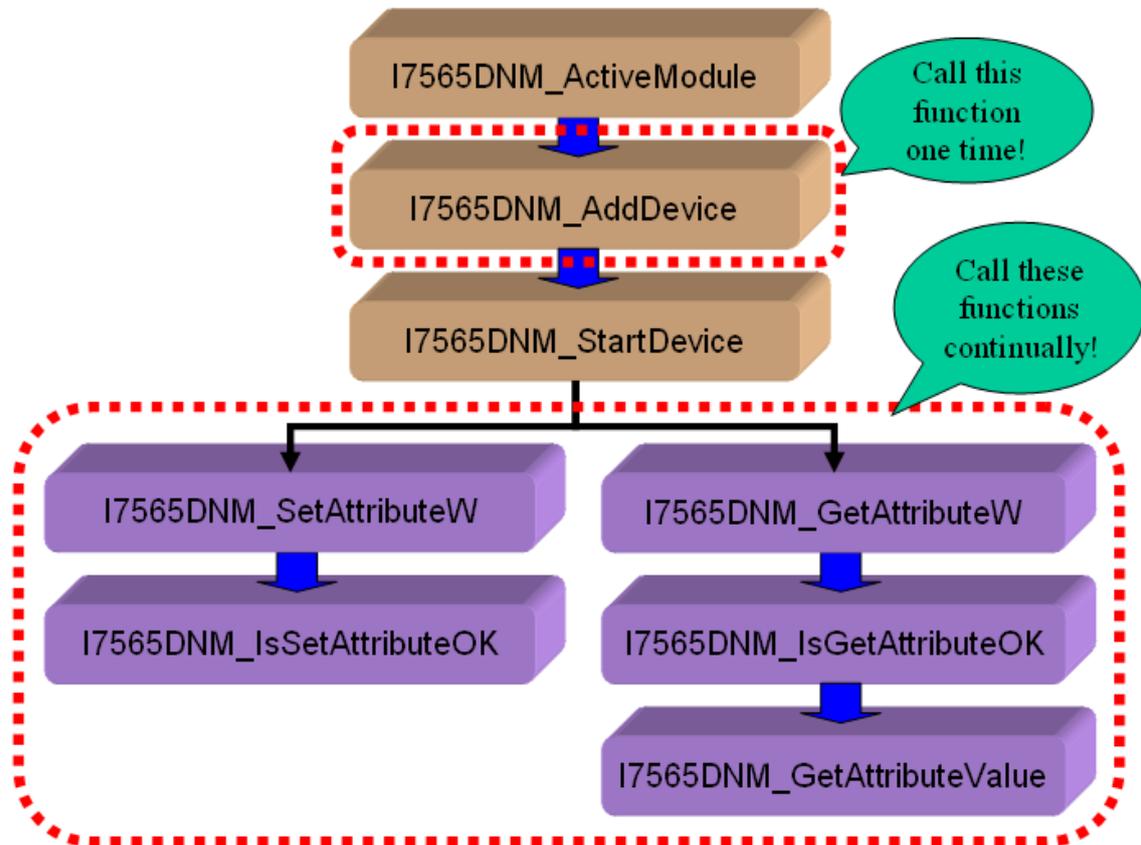


Figure 3.8 “SetAttributeW” and “GetAttributeW” Diagram

### 3.6 Flow Diagram for “SeneExplicitMSG\_W”

The users can send [Explicit Message] to the remote DeviceNet devices to set or get some parameters. The I-7565-DNM provides these functions to send the command and receive it replied message. The steps are shown in Figure 3.9.

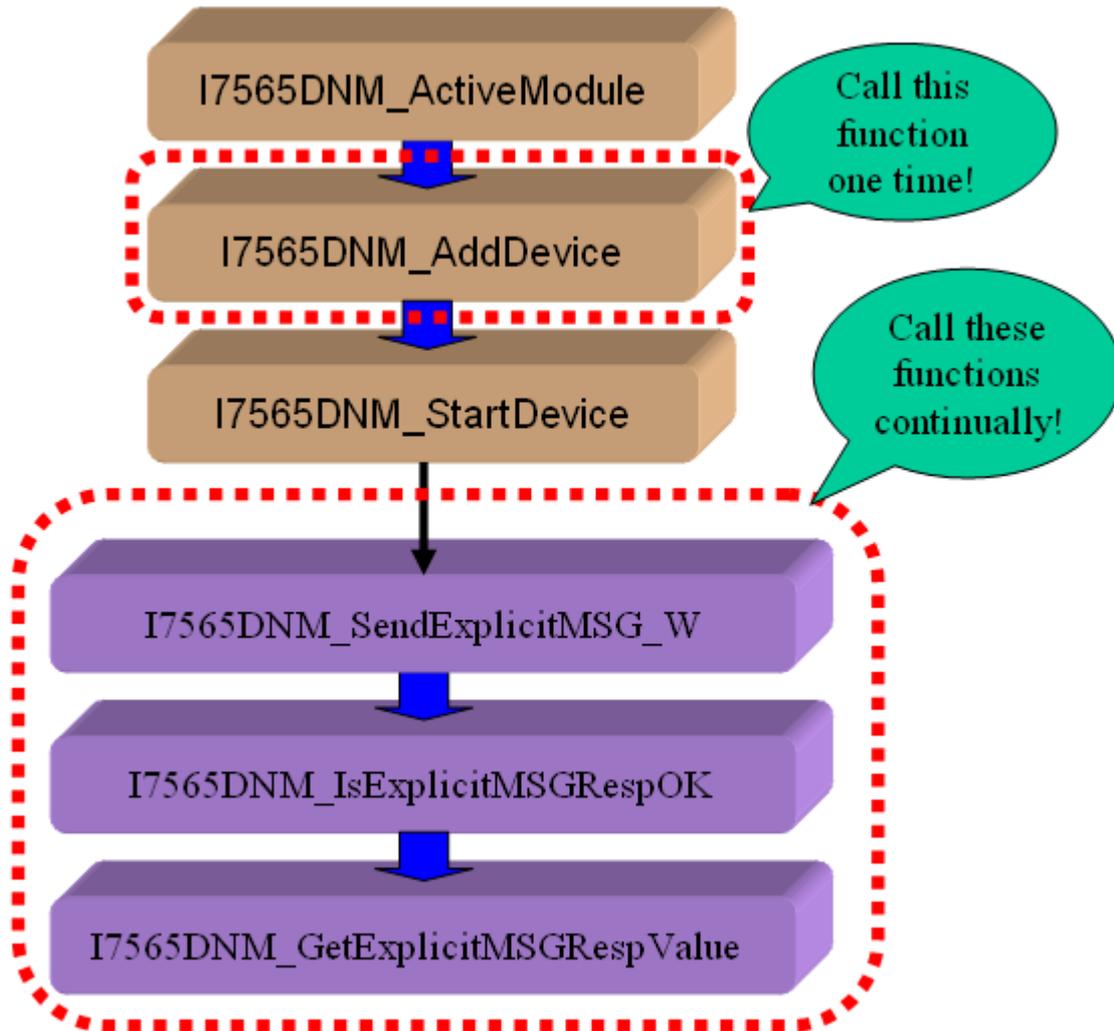


Figure 3.9 “SendExplicitMSG\_W” Diagram

### 3.7 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 I7565DNM\_AddIOConnection. Thirdly, the users can set the initial output value by calling I7565DNM\_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 3.10 shows the main steps to achieve this function. There are more functions described in chapter 4.

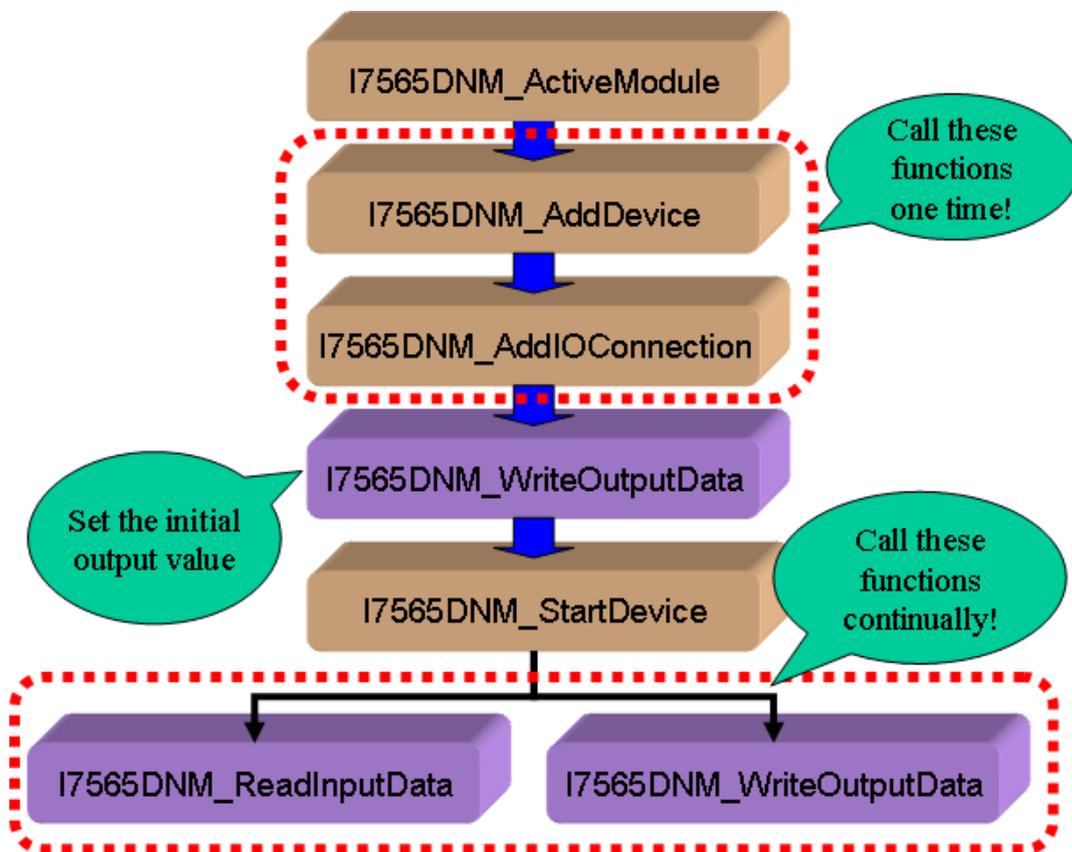


Figure 3.10 I/O Connection Diagram

Note: The Strobe connection doesn't support the output channel. The users can not use the I7565DNM\_WriteOutputData with Strobe connection.

### 3.8 Flow Diagram for Pause and Resume I/O Connection

When communicating with the remote slave devices and need to pause the I/O connection a while, the users can use the “PauseIOConnection” function to pause 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/SetAttribute” and “SendExplicitMSG\_W” functions to configure some parameters when the I/O connection has been suspended. The user could use “ResumeIOConnection” function to re-connect the I/O connection which has been paused. The Figure 3.11 shows the main steps to achieve this function.

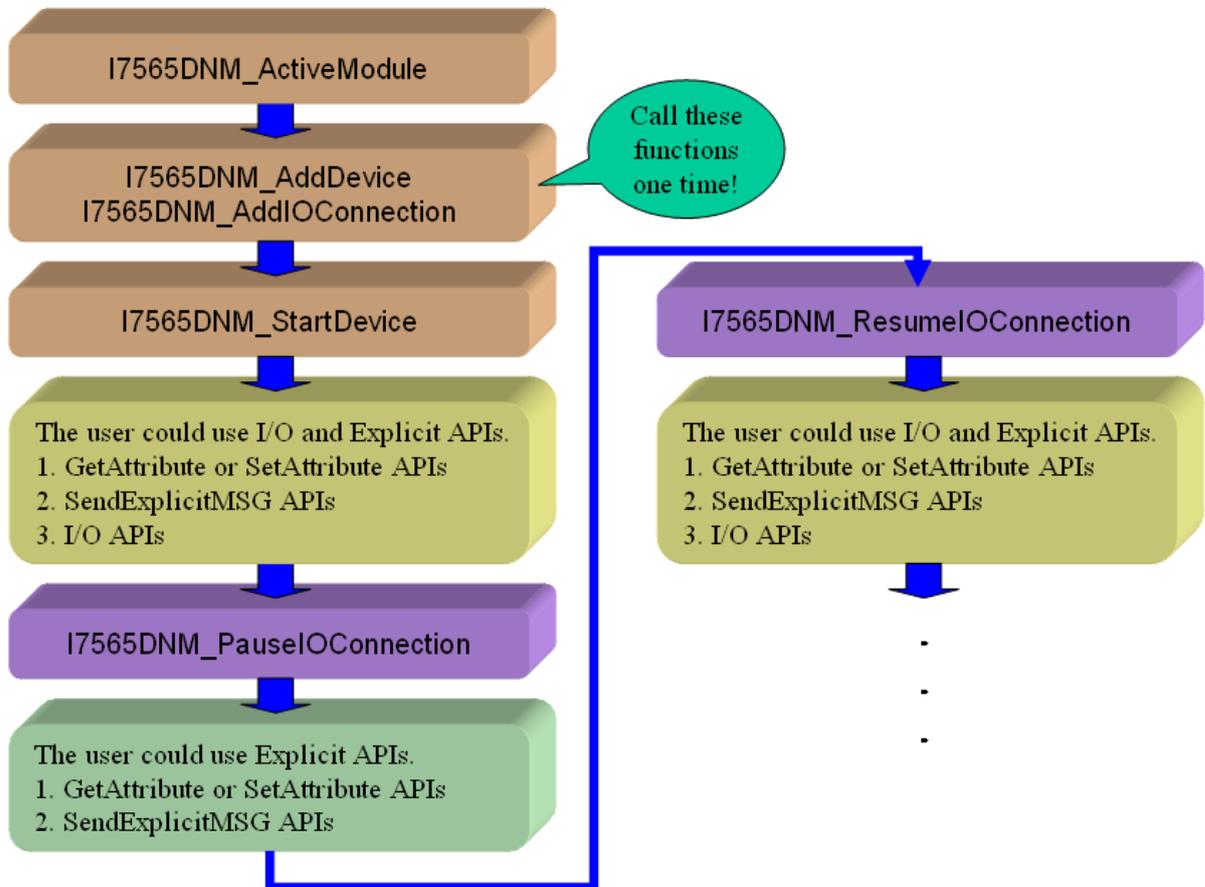


Figure 3.11 Pause and Resume I/O Connection Diagram

## Function description

All the functions of the I-7565-DNM can be separated into five groups. The idea is shown Figure 4.1. There is more detail description in CH 4.1.

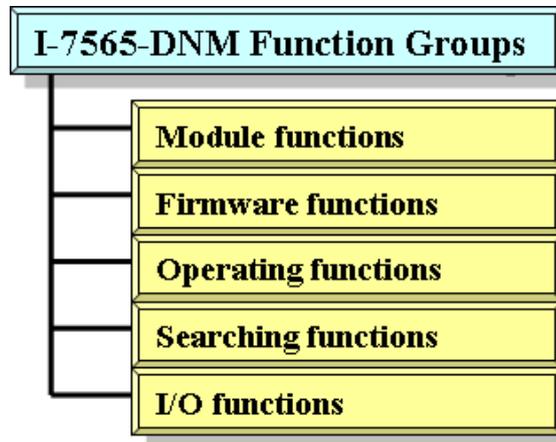


Figure 4.1 Five Function Groups

### **[Module Functions]**

These functions in this group help users to find I-7565-DNM modules or get module's information. The users can use these functions to configure or manage the modules 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 I-7565-DNM module.

### **[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 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 I7565DNM.DLL are listed in the following table and detail information for every function is presented in the next subsection. 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? |
| <b>[ input ]</b>  | Yes   | No  |
| <b>[ output ]</b> | No  | Yes   |

Table 4.1.1 Functions Table (Module Functions) 1/1

| No. | Function Name                | Description                             |
|-----|------------------------------|---|
| 1   | I7565DNM_TotalI7565DNMModule | Get total I-7565-DNM modules in the PC  |
| 2   | I7565DNM_ActiveModule        | Make I-7565-DNM module active           |
| 3   | I7565DNM_CloseModule         | Close the I-7565-DNM module             |
| 4   | I7565DNM_GetDLLVersion       | Get the DLL version of the I7565DNM.DLL |

Table 4.1.2 Functions Table (Firmware Functions) 1/1

| No. | Function Name               | Description  |
|-----|-----------------------------|--|
| 1   | I7565DNM_GetFirmwareVersion | Get the version of the firmware inside the I-7565-DNM module |
| 2   | I7565DNM_ResetFirmware      | Reset the firmware in the I-7565-DNM module                  |

Table 4.1.3 Functions Table (Operating Functions) 1/2

| No. | Function Name               | Description  |
|-----|-----------------------------|--|
| 1   | I7565DNM_SetMasterMACID     | Set the MAC ID of the I-7565-DNM module (DeviceNet Master's MAC ID)                            |
| 2   | I7565DNM_GetMasterMACID     | Get the MAC ID of the I-7565-DNM module (DeviceNet Master's MAC ID)                            |
| 3   | I7565DNM_GetBaudRate        | Get the baud rate of the CAN bus   |
| 4   | I7565DNM_SetBaudRate        | Set the baud rate of the CAN bus   |
| 5   | I7565DNM_GetMasterStatus    | Get the status of the I-7565-DNM module (DeviceNet Master's status) at present                 |
| 6   | I7565DNM_GetSlaveStatus     | Get the slave device's status.   |
| 7   | I7565DNM_StartDevice        | I-7565-DNM will start to communicate with the specific slave device                            |
| 8   | I7565DNM_StopDevice         | I-7565-DNM will stop to communicate with the specific slave device                             |
| 9   | I7565DNM_StartAllDevice     | I-7565-DNM will start to communicate with all slave devices                                    |
| 10  | I7565DNM_StopAllDevice      | I-7565-DNM will stop to communicate with all slave devices                                     |
| 11  | I7565DNM_AddDevice          | Add the specific slave device's information into the I-7565-DNM module (DeviceNet Master)      |
| 12  | I7565DNM_RemoveDevice       | Remove the specific slave device's information from the I-7565-DNM module (DeviceNet Master)   |
| 13  | I7565DNM_AddIOConnection    | Add I/O information of the specific slave device into the I-7565-DNM module (DeviceNet Master) |
| 14  | I7565DNM_RemoveIOConnection | Remove specific slave device's I/O information from the I-7565-DNM module (DeviceNet Master)   |

Table 4.1.4 Functions Table (Operating Functions) 2/2

| No. | Function Name                       | Description   |
|-----|-------------------------------------|---|
| 16  | <del>I7565DNM_GetAttribute</del>    | <del>Send the get attribute command to the slave device.</del>                                |
| 17  | I7565DNM_GetAttributeW              | Send the get attribute command to the slave device.   |
| 18  | I7565DNM_IsGetAttributeOK           | Check whether the slave has replied for the getting command or not.                           |
| 19  | I7565DNM_GetAttributeValue          | Get the attribute value of the I7565DNM_GetAttributeW   |
| 20  | <del>I7565DNM_SetAttribute</del>    | <del>Send the set attribute command to the slave device.</del>                                |
| 21  | I7565DNM_SetAttributeW              | Send the set attribute command to the slave device.   |
| 22  | I7565DNM_IsSetAttributeOK           | Check whether the slave has replied for the setting command or not.                           |
| 23  | I7565DNM_GetDeviceInfoFromScanList  | Get specific slave device's I/O information from the Scan List within the I-7565-DNM module.  |
| 24  | I7565DNM_GetScanList                | Get the I/O information of all slave devices from the Scan List within the I-7565-DNM module. |
| 25  | I7565DNM_ImportEEPROM               | Write the I/O information of all slave devices into the EEPROM within the I-7565-DNM module.  |
| 26  | I7565DNM_ClearAllConfig             | Clear all configurations in the EEPROM within the I-7565-DNM module.                          |
| 27  | <del>I7565DNM_SendExplicitMSG</del> | <del>Send the explicit request command.</del>   |
| 28  | I7565DNM_SendExplicitMSG_W          | Send the explicit request command.  |
| 29  | I7565DNM_IsExplicitMSGRespOK        | Check whether the I-7565-DNM has received the response message or not.                        |
| 30  | I7565DNM_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   | I7565DNM_SearchAllDevices     | I-7565-DNM will search the DeviceNet network to find out the I/O information of all slave devices.      |
| 2   | I7565DNM_SearchSpecificDevice | I-7565-DNM will search the DeviceNet network to find out the I/O information of specific slave devices. |
| 3   | I7565DNM_IsSearchOK           | Check whether the I-7565-DNM has searched completely or not.  |
| 4   | I7565DNM_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   | I7565DNM_ReadInputData      | Read the input data via I/O connection like Poll, Strobe, COS, Cyclic.   |
| 2   | I7565DNM_WriteOutputData    | Write the output data via I/O connection like Poll, COS, Cyclic. The Strobe doesn't support this operation.                                  |
| 3   | I7565DNM_ReadbackOutputData | Read back the output data via I/O connection like Poll, COS, Cyclic. The Strobe doesn't support this operation.                              |
| 4   | I7565DNM_PauseIOConnection  | To disconnect the I/O connection and keep the explicit connected. The user could set or get explicit message to configure the slave devices. |
| 5   | I7565DNM_ResumeIOConnection | To re-connect the paused I/O connection and keep the explicit connected. The user could read or write IO data of the slave devices.          |

## 4.2 Function Return Code

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

| Return Code | Error ID                  | Comment   |
|-------------|---------------------------|---|
| 0           | I7565DNM_NoError          | No error  |
| 10008       | I7565DNM_PortNotActive    | The USB port doesn't be activated.  |
| 10015       | I7565DNM_PortNoResp       | The USB port replied nothing.   |
| 10025       | I7565DNM_PortInUse        | The USB port is used by another program.  |
| 10027       | I7565DNM_ReStartPort      | The module has been re-plugged. Please restart your application.                        |
| 5000        | DNMXS_UnKnowError         | The DeviceNet has some unknown errors.  |
| 1000        | DNMXS_BoardNotActive      | The I-7565-DNM 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 I-7565-DNM 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 I-7565-DNM 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 4.2.2 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 Bit-Strobe 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 I-7565-DNM and all slave devices has been suspended. |

Table 4.2.3 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 I-7565-DNM 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 I7565DNM\_TotalI7565DNMModule

- **Description:**

The function will open the whole COM ports in the user's PC and try to find out where the module is. It can get the count of total I-7565-DNM modules in the user's PC.

- **Syntax:**

```
DWORD I7565DNM_Total I7565DNMModule (BYTE *TotalModules ,
                                       BYTE *PortList)
```

- **Parameter:**

**TotalModules:** [output] The amount of total modules.

**PortList:** [output] The list of all USB port in each modules.

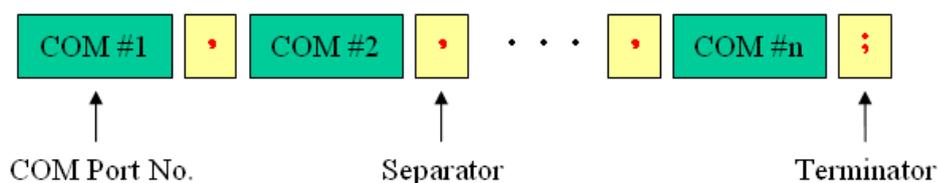
- **Return:**

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

- **Advanced Option:**

In some conditions, some COM ports in user's PC should not be opened. The user could provide a text file which should be named as "ExclusiveCOMPort.txt" to avoid this API to open those COM ports. The user could edit the content by common text editor.

Text file content



Example: The exclusive COM ports are 3, 8, 19. Here is the text content.

➔ 3,8,19;

---

### 4.3.2 I7565DNM\_ActiveModule

- **Description:**

The function is used to activate the I-7565-DNM module. It must be called once before using the other functions of I-7565-DNM APIs.

- **Syntax:**

DWORD I7565DNM\_ActiveModule (BYTE cPort)

- **Parameter:**

**cPort:** [input] The USB port number.

- **Return:**

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

---

### 4.3.3 I7565DNM\_CloseModule

- **Description:**

The function is used to stop and close the USB driver. This method must be called once before exiting the user's application program.

- **Syntax:**

DWORD I7565DNM\_CloseModule (BYTE cPort)

- **Parameter:**

**cPort:** [input] The USB port number.

- **Return:**

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

---

---

#### 4.3.4 I7565DNM\_GetDLLVersion

- **Description:**

The function can obtain the version information of I7565DNM.DLL.

- **Syntax:**

DWORD I7565DNM\_GetDLLVersion (void)

- **Parameter:**

None

- **Return:**

The DLL version information. For example: If 100(hex) is return, it means DLL version is 1.00. If 123(hex) is return, it means DLL version is 1.23.

---

---

#### 4.3.5 I7565DNM\_GetFirmwareVersion

- **Description:**

The function can obtain the version information of the firmware inside the I-7565-DNM module.

- **Syntax:**

DWORD I7565DNM\_GetFirmwareVersion (BYTE cPort)

- **Parameter:**

**cPort:** [input] The USB port number.

- **Return:**

The firmware version information. For example: If 100(hex) is return, it means firmware version is 1.00. If 123(hex) is return, it means firmware version is 1.23.

- **Error Return:**

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

---

---

#### 4.3.6 I7565DNM\_ResetFirmware

- **Description:**

The function is used to reset the I-7565-DNM 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 I7565DNM\_ResetFirmware (BYTE cPort)

- **Parameter:**

**cPort:** [input] The USB port number.

- **Return:**

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

---

### 4.3.7 I7565DNM\_GetMasterMACID

- **Description:**

The function can get the MAC ID of the DeviceNet master (I-7565-DNM).

- **Syntax:**

DWORD I7565DNM\_GetMasterMACID (BYTE cPort)

- **Parameter:**

**cPort:** [input] The USB port number.

- **Return:**

If return value is upper than 63, please refer to the chapter 4.2 for the function return code.

---

#### 4.3.8 I7565DNM\_SetMasterMACID

- **Description:**

The function can set the MAC ID of the DeviceNet master (I-7565-DNM). After calling this function, the users must call I7565DNM\_ResetFirmware to make the change enabled. It will save the information in the EEPROM in the I-7565-DNM.

- **Syntax:**

```
DWORD I7565DNM_SetMasterMACID (BYTE cPort,  
                                BYTE MasterMACID)
```

- **Parameter:**

**cPort:** [input] The USB port number.

**MasterMACID:** [input] The new MAC ID of the master. (0 ~ 63)

- **Return:**

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

---

#### 4.3.9 I7565DNM\_GetBaudRate

- **Description:**

This function can help you to get the DeviceNet baud rate information of I-7565-DNM.

- **Syntax:**

```
DWORD I7565DNM_GetBaudRate (BYTE cPort)
```

- **Parameter:**

**cPort:** [input] The USB port number.

- **Return:**

The CAN bus baud rate information in the I-7565-DNM.

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.

---

#### 4.3.10 I7565DNM\_SetBaudRate

- **Description:**

This function can set the DeviceNet baud rate of the I-7565-DNM. After calling this function, you must call I7565DNM\_ResetFirmware to reset the firmware to make change enabled.

- **Syntax:**

DWORD I7565DNM\_SetBaudRate (BYTE cPort, BYTE BaudRate)

- **Parameter:**

**cPort:** [input] The USB port number.

**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.

---

---

#### 4.3.11 I7565DNM\_GetMasterStatus

- **Description:**

The function is used to obtain the firmware status inside the I-7565-DNM. The users can call this function to make sure that the DeviceNet master is online successfully.

- **Syntax:**

DWORD I7565DNM\_GetMasterStatus (BYTE cPort)

- **Parameter:**

**cPort:** [input] The USB port number.

- **Return:**

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

---

---

#### 4.3.12 I7565DNM\_GetSlaveStatus

- **Description:**

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

- **Syntax:**

DWORD I7565DNM\_GetSlaveStatus (BYTE cPort, BYTE DesMACID)

- **Parameter:**

**cPort:** [input] The USB port number.

**DesMACID:** [input] The remote slave's MAC ID. (0~63)

- **Return:**

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

---

---

### 4.3.13 I7565DNM\_StartDevice

- **Description:**

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

- **Syntax:**

DWORD I7565DNM\_StartDevice (BYTE cPort, BYTE DesMACID)

- **Parameter:**

**cPort:** [input] The USB port number.

**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 I7565DNM\_StopDevice

- **Description:**

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

- **Syntax:**

DWORD I7565DNM\_StopDevice (BYTE cPort, BYTE DesMACID)

- **Parameter:**

**cPort:** [input] The USB port number.

**DestMACID:** [input] The remote slave device's MAC ID (0~63)

- **Return:**

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

---

#### 4.3.15 I7565DNM\_StartAllDevice

- **Description:**

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

- **Syntax:**

DWORD I7565DNM\_StartAllDevice (BYTE cPort)

- **Parameter:**

**cPort:** [input] The USB port number.

- **Return:**

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

---

---

#### 4.3.16 I7565DNM\_StopAllDevice

- **Description:**

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

- **Syntax:**

DWORD I7565DNM\_StopAllDevice (BYTE cPort)

- **Parameter:**

**cPort:** [input] The USB port number.

- **Return:**

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

---

### 4.3.17 I7565DNM\_AddDevice

- **Description:**

This function can add the slave devices into the ScanList of the I-7565-DNM 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 I7565DNM\_AddDevice (BYTE cPort, BYTE DesMACID,  
WORD Explicit\_EPR)

- **Parameter:**

**cPort:** [input] The USB port number.

**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.

- **C++ demo code:**

```
WORD Ret, Slave_EPR;
```

```
BYTE ActivedModuleNo, Slave_MACID;
```

```
ActivatedModuleNo = 2;
```

```
Slave_MACID = 12;
```

```
//AddDevice for the first time
```

```
Ret = I7565DNM_AddDevice(ActivatedModuleNo, Slave_MACID, 1000);
```

```
if(Ret != 0) return Ret;
```

---

#### 4.3.18 I7565DNM\_RemoveDevice

- **Description:**

This function is used for removing the specified slave device from the ScanList in the I-7565-DNM. And the information of the device in EEPROM is erased at the same time.

- **Syntax:**

DWORD I7565DNM\_RemoveDevice (BYTE cPort, BYTE DesMACID)

- **Parameter:**

**cPort:** [input] The USB port number.

**DestMACID:** [input] The remote slave device's MAC ID (0~63)

- **Return:**

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

---

### 4.3.19 I7565DNM\_AddIOConnection

- **Description:**

This method is used to configure the I/O connection of the specific MAC ID device. The I-7565-DNM 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 I-7565-DNM.

- **Syntax:**

DWORD I7565DNM\_AddIOConnection (BYTE cPort, BYTE DesMACID, BYTE ConType, WORD DeviceInputLen, WORD DeviceOutputLen, WORD EPR)

- **Parameter:**

**cPort:** [input] The USB port number.

**DestMACID:** [input] The remote slave device's MAC ID (0~63)

**ConType:** [input] The remote slave device's I/O 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.

---

- **C++ demo code:**

```
WORD Ret, Slave_EPR, InputLen, OutputLen;  
BYTE ActivatedModuleNo, Slave_MACID, ConType;
```

```
ActivatedModuleNo = 2;  
Slave_MACID = 12;  
Slave_EPR = 200; //polling rate = 200ms  
InputLen = 5; //the input byte of the slave device.  
OutputLen = 7; //the output byte of the slave device.  
ConType = ConType_Poll;
```

```
//AddDevice for the first time
```

```
Ret = I7565DNM_AddDevice(ActivatedModuleNo, Slave_MACID, 1000);  
if(Ret != 0) return Ret;
```

```
//AddIOConnection for the first time
```

```
Ret = I7565DNM_AddIOConnection(ActivatedModuleNo, Slave_MACID,  
                                ConType, InputLen, OutputLen, Slave_EPR);  
if(Ret != 0) return Ret;
```

```
...
```

---

#### 4.3.20 I7565DNM\_RemoveIOConnection

- **Description:**

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

- **Syntax:**

DWORD I7565DNM\_RemoveIOConnection (BYTE cPort,  
BYTE DesMACID,  
BYTE ConType)

- **Parameter:**

**cPort:** [input] The USB port number.

**DestMACID:** [input] The remote slave device's MAC ID (0~63)

**ConType:** [input] The remote slave device's I/O 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.

---

### ~~4.3.21 I7565DNM\_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 "I7565DNM\_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 "I7565DNM\_GetAttributeW".~~

● ~~Syntax:~~

~~DWORD I7565DNM\_GetAttribute (BYTE cPort, BYTE DestMACID,  
BYTE ClassID, BYTE InstanceID,  
BYTE AttributeID)~~

● ~~Parameter:~~

~~**cPort:** [input] The USB port number.~~

~~**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.~~

### 4.3.22 I7565DNM\_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 first execute "I7565DNM\_IsGetAttributeOK" and then proceed to run "I7565DNM\_GetAttributeValue" to obtain the response message returned from the remote slave device.

This function could totally complain with the old function which has the same name without the "W". The user could use this function instead of the "I7565DNM\_GetAttribute".

- **Syntax:**

```
DWORD I7565DNM_GetAttributeW (BYTE cPort, BYTE DesMACID,
                               WORD ClassID, WORD InstanceID,
                               BYTE AttributeID)
```

- **Parameter:**

**cPort:** [input] The USB port number.

**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.

- **C++ demo code:**

```
WORD Ret, ClassID, InstanceID, AttributeID, Len;
BYTE ActivedModuleNo, Slave_MACID, Data[512];
CString Str_Hex, Str_ASCII, temp;
```

```
ActivedModuleNo = 2;
```

```
Slave_MACID = 12;
```

```
//Get the name of the slave device, CID = 1, Inst. ID = 1, Attr. ID = 7
ClassID = 1;
InstanceID = 1;
AttributeID = 7;

Ret = I7565DNM_GetAttributeW(ActivatedModuleNo, Slave_MACID,
                             ClassID, InstanceID, AttributeID);
if(Ret != 0) return Ret;

Sleep(1000); //busy waiting for the slave.
Ret = I7565DNM_IsGetAttributeOK (ActivatedModuleNo, Slave_MACID);
if(Ret != 0) Sleep(1000); //busy waiting for the slave again.

Ret = I7565DNM_GetAttributeValue(ActivatedModuleNo, Slave_MACID,
                                 &Len, Data);

if(Ret != 0) return Ret;

Str_Hex = "";
Str_ASCII = "";
for(int i=0;i<Len;i++)
{
    temp.printf("%X ",Data[i]);
    Str_Hex += temp;
    temp.printf("%C ",Data[i]);
    Str_ASCII += temp;
}
AfxMessageBox(Str_Hex);
AfxMessageBox(Str_ASCII);
```

---



### 4.3.23 I7565DNM\_IsGetAttributeOK

- **Description:**

This function is used to check whether the I-7565-DNM has received the response message or not. After checking the response message, you should execute the “I7565DNM\_GetAttributeValue” to get the response message returned from remote slave device.

- **Syntax:**

DWORD I7565DNM\_IsGetAttributeOK (BYTE cPort, BYTE DesMACID)

- **Parameter:**

**cPort:** [input] The USB port number.

**DestMACID:** [input] The remote slave device’s MAC ID (0~63)

- **Return:**

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

- **C++ demo code:**

```
WORD Ret, ClassID, InstanceID, AttributeID, Len;
BYTE ActivatedModuleNo, Slave_MACID, Data[512];
CString Str_Hex, Str_ASCII, temp;
```

```
ActivatedModuleNo = 2;
```

```
Slave_MACID = 12;
```

```
//Get the name of the slave device, CID = 1, Inst. ID = 1, Attr. ID = 7
```

```
ClassID = 1;
```

```
InstanceID = 1;
```

```
AttributeID = 7;
```

```
Ret = I7565DNM_GetAttributeW(ActivatedModuleNo, Slave_MACID,
                             ClassID, InstanceID, AttributeID);
```

```
if(Ret != 0) return Ret;
```

```
Sleep(1000); //busy waiting for the slave.
```

```
Ret = I7565DNM_IsGetAttributeOK (ActivatedModuleNo, Slave_MACID);  
if(Ret != 0) Sleep(1000); //busy waiting for the slave again.
```

```
Ret = I7565DNM_GetAttributeValue(ActivatedModuleNo, Slave_MACID,  
                                &Len, Data);
```

```
if(Ret != 0) return Ret;
```

```
Str_Hex = "";
```

```
Str_ASCII = "";
```

```
for(int i=0;i<Len;i++)
```

```
{
```

```
    temp.printf("%X ",Data[i]);
```

```
    Str_Hex += temp;
```

```
    temp.printf("%C ",Data[i]);
```

```
    Str_ASCII += temp;
```

```
}
```

```
AfxMessageBox(Str_Hex);
```

```
AfxMessageBox(Str_ASCII);
```



```
if(Ret != 0) return Ret;
```

```
Sleep(1000); //busy waiting for the slave.
```

```
Ret = I7565DNM_IsGetAttributeOK (ActivatedModuleNo, Slave_MACID);
```

```
if(Ret != 0) Sleep(1000); //busy waiting for the slave again.
```

```
Ret = I7565DNM_GetAttributeValue(ActivatedModuleNo, Slave_MACID,  
                                &Len, Data);
```

```
if(Ret != 0) return Ret;
```

```
Str_Hex = "";
```

```
Str_ASCII = "";
```

```
for(int i=0;i<Len;i++)
```

```
{
```

```
    temp.printf("%X ",Data[i]);
```

```
    Str_Hex += temp;
```

```
    temp.printf("%C ",Data[i]);
```

```
    Str_ASCII += temp;
```

```
}
```

```
AfxMessageBox(Str_Hex);
```

```
AfxMessageBox(Str_ASCII);
```

---

### 4.3.25 ~~I7565DNM\_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 "I7565DNM\_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 "I7565DNM\_SetAttributeW".~~

● ~~Syntax:~~

```
DWORD I7565DNM_SetAttribute (BYTE cPort, BYTE DestMACID,  
                             BYTE ClassID, BYTE InstanceID,  
                             BYTE AttributeID, WORD DataLen,  
                             BYTE *DATA)
```

● ~~Parameter:~~

~~**cPort:** [input] The USB port number.~~

~~**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.~~

### 4.3.26 I7565DNM\_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 "I7565DNM\_IsSetAttributeOK" to check the response message returned from the remote slave device.

This function could totally complain with the old function which has the same name without the "W". The user could use this function instead of the "I7565DNM\_SetAttribute".

- **Syntax:**

```
DWORD I7565DNM_SetAttributeW (BYTE cPort, BYTE DesMACID,
                               WORD ClassID, WORD InstanceID,
                               BYTE AttributeID, WORD DataLen,
                               BYTE *DATA)
```

- **Parameter:**

**cPort:** [input] The USB port number.

**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.

- **C++ demo code:**

```
WORD Ret, ClassID, InstanceID, AttributeID, Len;
BYTE ActivatedModuleNo, Slave_MACID, Data[512];
```

```
ActivatedModuleNo = 2;
```

```
Slave_MACID = 12;
```

```
//Set the EPR of the slave device, CID = 5, Inst. ID = 1, Attr. ID = 9
ClassID = 5;
InstanceID = 1;
AttributeID = 9;
```

```
//Set the EPR = 2500(0x9C4)
Data[0] = 0xC4;
Data[1] = 0x09;
Len = 2;
```

```
Ret = I7565DNM_SetAttributeW(ActivatedModuleNo, Slave_MACID,
                             ClassID, InstanceID, AttributeID, Len, Data);
if(Ret != 0) return Ret;
```

```
Sleep(1000); //busy waiting for the slave.
Ret = I7565DNM_IsSetAttributeOK(ActivatedModuleNo, Slave_MACID);
if(Ret != 0) Sleep(1000); //busy waiting for the slave again.
```

---

### 4.3.27 I7565DNM\_IsSetAttributeOK

- **Description:**

This function is used to get the response value after executing the “I7565DNM\_SetAttributeW” function.

- **Syntax:**

DWORD I7565DNM\_IsSetAttributeOK (BYTE cPort, BYTE DesMACID)

- **Parameter:**

**cPort:** [input] The USB port number.

**DestMACID:** [input] The remote slave device’s MAC ID (0~63)

- **Return:**

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

- **C++ demo code:**

```
WORD Ret, ClassID, InstanceID, AttributeID, Len;
BYTE ActivatedModuleNo, Slave_MACID, Data[512];
```

```
ActivatedModuleNo = 2;
```

```
Slave_MACID = 12;
```

```
//Set the EPR of the slave device, CID = 5, Inst. ID = 1, Attr. ID = 9
```

```
ClassID = 5;
```

```
InstanceID = 1;
```

```
AttributeID = 9;
```

```
//Set the EPR = 2500(0x9C4)
```

```
Data[0] = 0xC4;
```

```
Data[1] = 0x09;
```

```
Len = 2;
```

```
Ret = I7565DNM_SetAttributeW(ActivatedModuleNo, Slave_MACID,
                             ClassID, InstanceID, AttributeID, Len, Data);
```

```
if(Ret != 0) return Ret;
```

```
Sleep(1000); //busy waiting for the slave.  
Ret = I7565DNM_IsSetAttributeOK(ActivatedModuleNo, Slave_MACID);  
if(Ret != 0) Sleep(1000); //busy waiting for the slave again.
```

---

#### 4.3.28 I7565DNM\_ClearAllConfig

- **Description:**

This function will clear all configurations in the EEPROM of the I-7565-DNM.

- **Syntax:**

DWORD I7565DNM\_ClearAllConfig (BYTE cPort)

- **Parameter:**

**cPort:** [input] The USB port number.

- **Return:**

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

---

---

#### 4.3.29 I7565DNM\_SearchAllDevices

- **Description:**

This function is used to retrieve all devices in DeviceNet network. This function makes the I-7565-DNM to start the searching process. The users need to check whether the process is complete or not by calling the “I7565DNM\_IsSearchOK”. After completing the search process, the users could call the “I7565DNM\_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 I7565DNM\_SearchAllDevices (BYTE cPort)

- **Parameter:**

**cPort:** [input] The USB port number.

- **Return:**

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

---

### 4.3.30 I7565DNM\_SearchSpecificDevice

- **Description:**

This function is used to retrieve some devices which specified by the users. This function makes the I-7565-DNM to start the searching process. The users need to check whether the process is complete or not by calling the “I7565DNM\_IsSearchOK”. After completing the search process, the users could call the “I7565DNM\_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 I7565DNM_SearchSpecificDevice (BYTE cPort,  
                                     WORD ListCount,  
                                     BYTE *DesMACIDList)
```

- **Parameter:**

**cPort:** [input] The USB port number.

**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.

---

---

#### 4.3.31 I7565DNM\_IsSearchOK

- **Description:**

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

- **Syntax:**

DWORD I7565DNM\_IsSearchOK (BYTE cPort)

- **Parameter:**

**cPort:** [input] The USB port number.

- **Return:**

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

---

### 4.3.32 I7565DNM\_GetSearchedDevices

- **Description:**

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

- **Syntax:**

```
DWORD I7565DNM_GetSearchedDevices (BYTE cPort,  
                                     WORD *TotalDevices,  
                                     BYTE *DesMACID,  
                                     BYTE *Type,  
                                     WORD *DeviceInputLen,  
                                     WORD *DeviceOutputLen)
```

- **Parameter:**

**cPort:** [input] The USB port number.

**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.

---

### 4.3.33 I7565DNM\_GetDeviceInfoFromScanList

- **Description:**

This function will get the ScanList data of certain device in the I-7565-DNM.

- **Syntax:**

```
DWORD I7565DNM_GetDeviceInfoFromScanList  
    (BYTE cPort, BYTE DesMACID, WORD *ListCount,  
     BYTE *ConnectionTypeList, WORD *InputDataLenList,  
     WORD *OutputDataLenList,WORD *EPRList)
```

- **Parameter:**

**cPort:** [input] The USB port number.

**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.

---

#### 4.3.34 I7565DNM\_GetScanList

- **Description:**

This function will get all the ScanList data in the I-7565-DNM.

- **Syntax:**

```
DWORD I7565DNM_GetScanList (BYTE cPort, WORD *TotalDevices,  
                             BYTE *DesMACIDList,  
                             BYTE *ConnectionTypeList,  
                             WORD *InputDataLenList,  
                             WORD *OutputDataLenList,  
                             WORD *EPR_List)
```

- **Parameter:**

**cPort:** [input] The USB port number.

**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.

---

### 4.3.35 I7565DNM\_ImportEEPROM

- **Description:**

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

- **Syntax:**

```
DWORD I7565DNM_ImportEEPROM (BYTE cPort,  
                               WORD ListCount,  
                               BYTE *DesMACIDList,  
                               BYTE *ConnectionTypeList,  
                               WORD *InputDataLenList,  
                               WORD *OutputDataLenList,  
                               WORD *EPR_List)
```

- **Parameter:**

**cPort:** [input] The USB port number.

**ListCount:** [input] The data count of all the information.

**DestMACIDList:** [input] The MAC ID of all the slave devices.

**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.

---

### 4.3.36 I7565DNM\_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 I7565DNM_ReadInputData (BYTE cPort, BYTE DesMACID,
                               BYTE ConType, WORD *IOLen,
                               BYTE *IODATA)
```

- **Parameter:**

**cPort:** [input] The USB port number.

**DestMACID:** [input] The remote slave device's MAC ID (0~63)

**ConType:** [input] The connection type of the remote slave.

- 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.

- **C++ demo code:**

```
WORD Ret, IOLen, i;
BYTE ActivedModuleNo, Slave_MACID, ConType, IOData[512];
CString temp;
```

```
ActivatedModuleNo = 2;
Slave_MACID = 12;
ConType = ConType_Poll;
```

```
Ret = I7565DNM_ReadInputData(ActiveBoardNo, Slave_MACID,  
                             ConType_Poll, &IOLen, IOData);  
  
if(Ret == I7565DNM_NoError)  
{  
    Str.Format("");  
    for(i = 0;i < IOLen;i++)  
    {  
        temp.Format("0x%X, ",IOData[i]);  
        Str += temp;  
    }  
}  
else  
    Str.Format("Return : Error! %d",Ret);  
AfxMessageBox(Str);
```

---

### 4.3.37 I7565DNM\_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 I7565DNM\_WriteOutputData (BYTE cPort, BYTE DesMACID, BYTE ConType, WORD IOLen, BYTE \*IOWDATA)

- **Parameter:**

**cPort:** [input] The USB port number.

**DestMACID:** [input] The remote slave device's MAC ID (0~63)

**ConType:** [input] The connection type of the remote slave.

- 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).

**IOWDATA:** [Input] The remote I/O data.

- **Return:**

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

- **C++ demo code:**

```
WORD Ret, IOLen, i;
BYTE ActivatedModuleNo, Slave_MACID, ConType, IOData[512];
CString temp;

ActivatedModuleNo = 2;
Slave_MACID = 12;
ConType = ConType_Poll;
IOData[0] = 0xFF;
```

```
IOData[1] = 0xAB;  
IOData[2] = 0xCC;  
IOLen = 3;
```

```
Ret = I7565DNM_WriteOutputData(ActivatedModuleNo, Slave_MACID,  
                               ConType_Poll, IOLen, IOData);  
if(Ret != I7565DNM_NoError) return Ret;
```

---

### 4.3.38 ~~I7565DNM\_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 "I7565DNM\_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 "I7565DNM\_SendExplicitMSG\_W".~~

#### ● ~~Syntax:~~

```
DWORD I7565DNM_SendExplicitMSG (BYTE cPort, BYTE DestMACID,
                                BYTE ServiceID, BYTE ClassID,
                                BYTE InstanceID,
                                WORD DataLen, BYTE *DATA)
```

#### ● ~~Parameter:~~

~~**cPort:** [input] The USB port number.~~

~~**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.~~

### 4.3.39 I7565DNM\_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. Before executing "I7565DNM\_GetExplicitMSGRespValue," you should first execute "I7565DNM\_IsExplicitMSGRespOK" to ensure the correctness of the response message from the remote slave device.

This function could totally complain with the old function which has the same name without the "\_W". The user could use this function instead of the "I7565DNM\_SendExplicitMSG".

- **Syntax:**

DWORD I7565DNM\_SendExplicitMSG\_W (BYTE cPort, BYTE DesMACID, BYTE ServiceID, WORD ClassID, WORD InstanceID, WORD DataLen, BYTE \*DATA)

- **Parameter:**

**cPort:** [input] The USB port number.

**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.

#### 4.3.40 I7565DNM\_IsExplicitMSGRespOK

- **Description:**

This function is used to check whether the I-7565-DNM has received the response message or not. After checking the response message, you should execute the “I7565DNM\_GetExplicitMSGRespValue” to get the response message returned from remote slave device.

- **Syntax:**

DWORD I7565DNM\_IsExplicitMSGRespOK (BYTE cPort,  
BYTE DesMACID)

- **Parameter:**

**cPort:** [input] The USB port number.

**DestMACID:** [input] The remote slave device’s MAC ID (0~63)

- **Return:**

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

---

#### 4.3.41 I7565DNM\_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 "I7565DNM\_SendExplicitMSG\_W" and "I7565DNM\_IsExplicitMSGRespOK" to send request command first.

- **Syntax:**

DWORD I7565DNM\_GetExplicitMSGRespValue (BYTE cPort,  
BYTE DesMACID,  
WORD \*DataLen ,  
BYTE \*DATA)

- **Parameter:**

**cPort:** [input] The USB port number.

**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.

---

#### 4.3.42 I7565DNM\_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 I7565DNM_ReadbackOutputData (BYTE cPort,  
                                     BYTE DesMACID,  
                                     BYTE ConType,  
                                     WORD *IOLen,  
                                     BYTE *IODATA)
```

- **Parameter:**

**cPort:** [input] The USB port number.

**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.

---

#### 4.3.43 I7565DNM\_PauseIOConnection

- **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/SetAttribute” and “SendExplicitMSG\_W” functions to configure some parameters when the I/O connection has been suspended.

- **Syntax:**

```
DWORD I7565DNM_PauseIOConnection (BYTE cPort,  
                                   BYTE DesMACID)
```

- **Parameter:**

**cPort:** [input] The USB port number.

**DestMACID:** [input] The remote slave device's MAC ID (0~63)

- **Return:**

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

---

#### 4.3.44 I7565DNM\_ResumeIOConnection

- **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/SetAttribute” functions.

- **Syntax:**

```
DWORD I7565DNM_ResumeIOConnection (BYTE cPort,  
                                     BYTE DesMACID)
```

- **Parameter:**

**cPort:** [input] The USB port number.

**DestMACID:** [input] The remote slave device’s MAC ID (0~63)

- **Return:**

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

---

---

#### 4.3.45 I7565DNM\_DisableKeepAliveMsg (Advanced Option)

- **Description:**

- The I-7565-DNM 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 I7565DNM\_ActiveModule(). This disable status will NOT keep in I-7565-DNM. The users need to call this for every boot-up.

- **Syntax:**

DWORD I7565DNM\_DisableKeepAliveMsg (BYTE cPort,  
BYTE DesMACID)

- **Parameter:**

**cPort:** [input] The USB port number.

**DestMACID:** [input] The remote slave device's MAC ID (0~63)

- **Return:**

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

---

---

## ***Demo Programs for Windows***

All of demo programs will not work normally if I-7565-DNM 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 (Win98,Me,NT,win2000,XP). 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 I-7565-DNM's root directory is C:\ICPDAS\I-7565-DNM

|                |   |
|----------------|---|
| --\DLL         | The DLL driver for the user's application |
| --\Driver      | The window driver of the I-7565-DNM       |
| --\Manual      | The user manual of the I-7565-DNM         |
| --\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

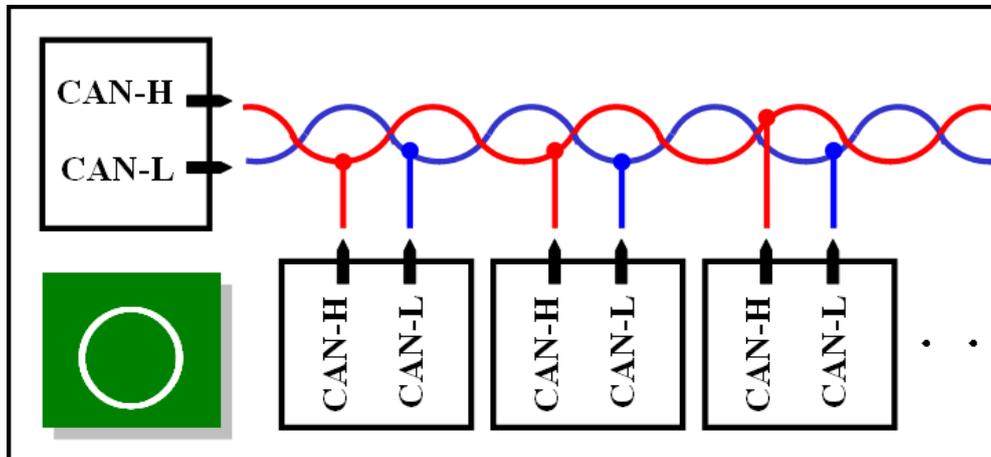


Figure 5.1 Correct wire connection

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

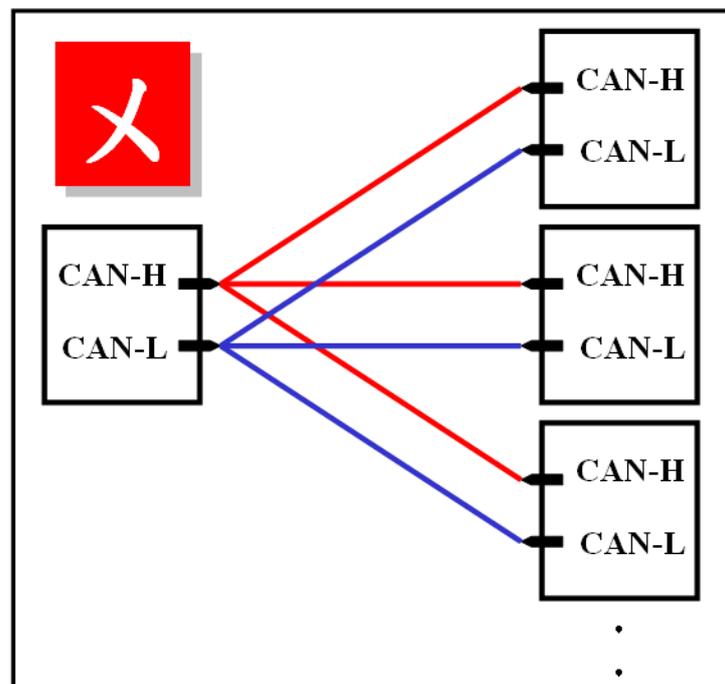


Figure 5.2 Wrong wire connection

## 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)

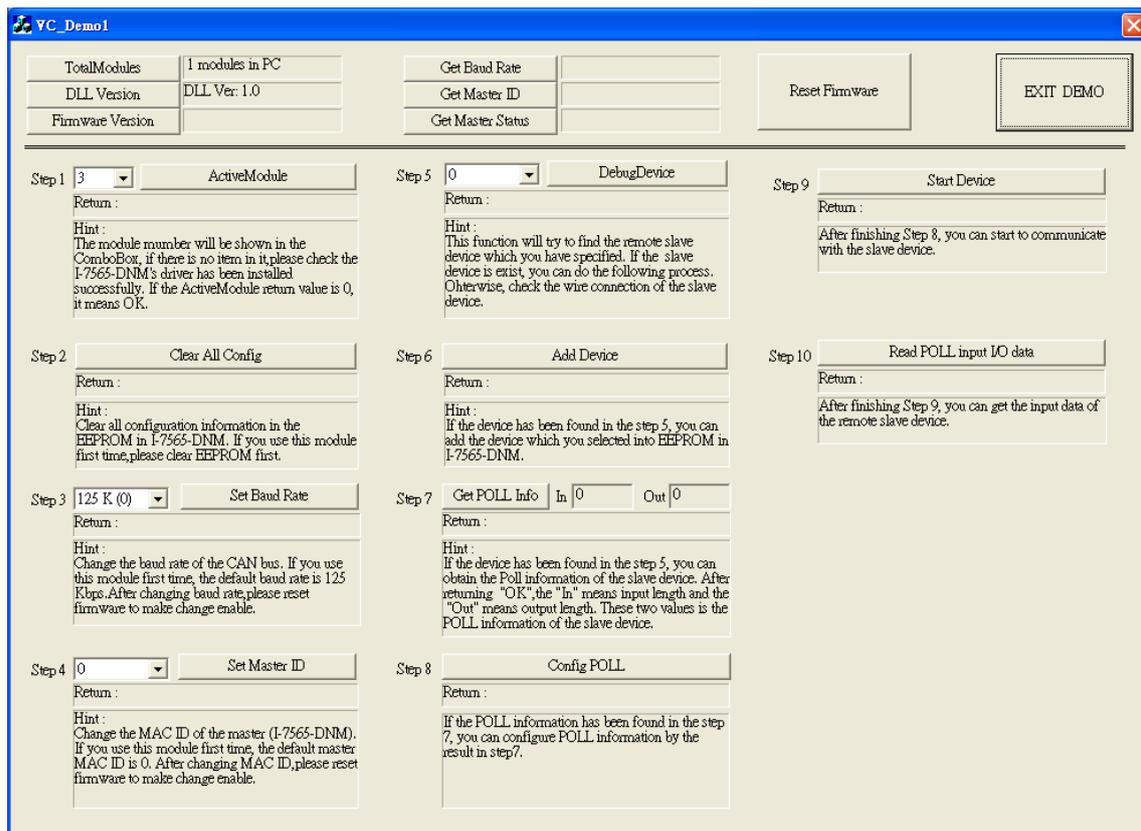


Figure 5.3 the screen shoot of VC\_Demo1

After running the program, the users will see the “TotalModules” information on the left and up corner of the screen. This function determinates how many I-7565-DNM in your PC automatically. If the program doesn’t find any module, the users should check that the windows driver has been installed successfully or the USB wire connection between the PC and the I-7565-DNM module. Otherwise, if it has found at least one module, the users can continue exercising the demo program.

---

### Step 1 : ActiveModule

Before performing other buttons, the “ActiveMoudle” button should be clicked firstly. The module number means the COM port number in the PC. The drop-down list would show the module’s number which has been plugged in the PC. 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 I-7565-DNM, the users can click this button to clear all configuration in the module.

### 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 I-7565-DNM 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 I-7565-DNM 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 I-7565-DNM. 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 I-7565-DNM. 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)

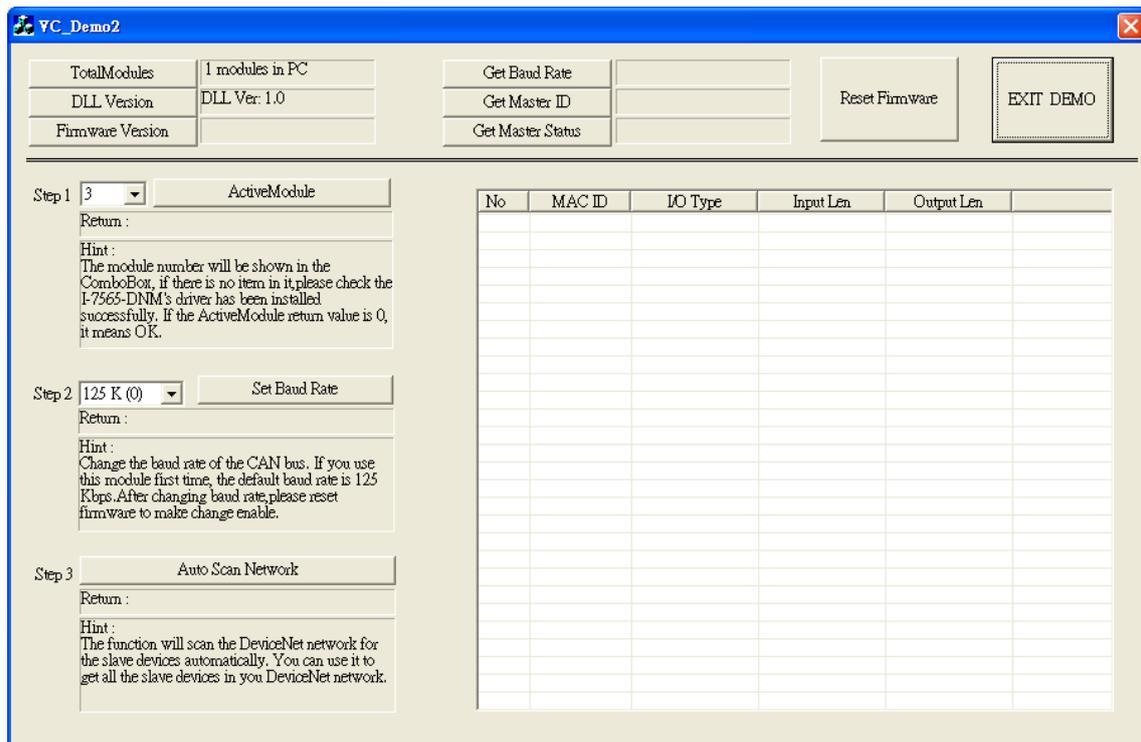


Figure 5.4 The screen shoot of VC\_Demo2

After running the program, the users will see the “TotalModules” information on the left and up corner of the screen. This function determinates how many I-7565-DNM in your PC automatically. If it doesn’t find any board, the users should check that the windows driver has been installed successfully or the USB wire connection between the PC and the I-7565-DNM module. Otherwise, if it has found at least one module, the users can continue exercising the demo program.

---

### Step 1 : ActiveModule

Before performing other buttons, the “ActiveMoudle” button should be clicked firstly. The module number means the COM port number in the PC. The drop-down list would show the module’s number which has been plugged in the PC. 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 I-7565-DNM by clicking the “ResetFirmware” button. It needs to wait for 1 or 2 seconds to the next step.

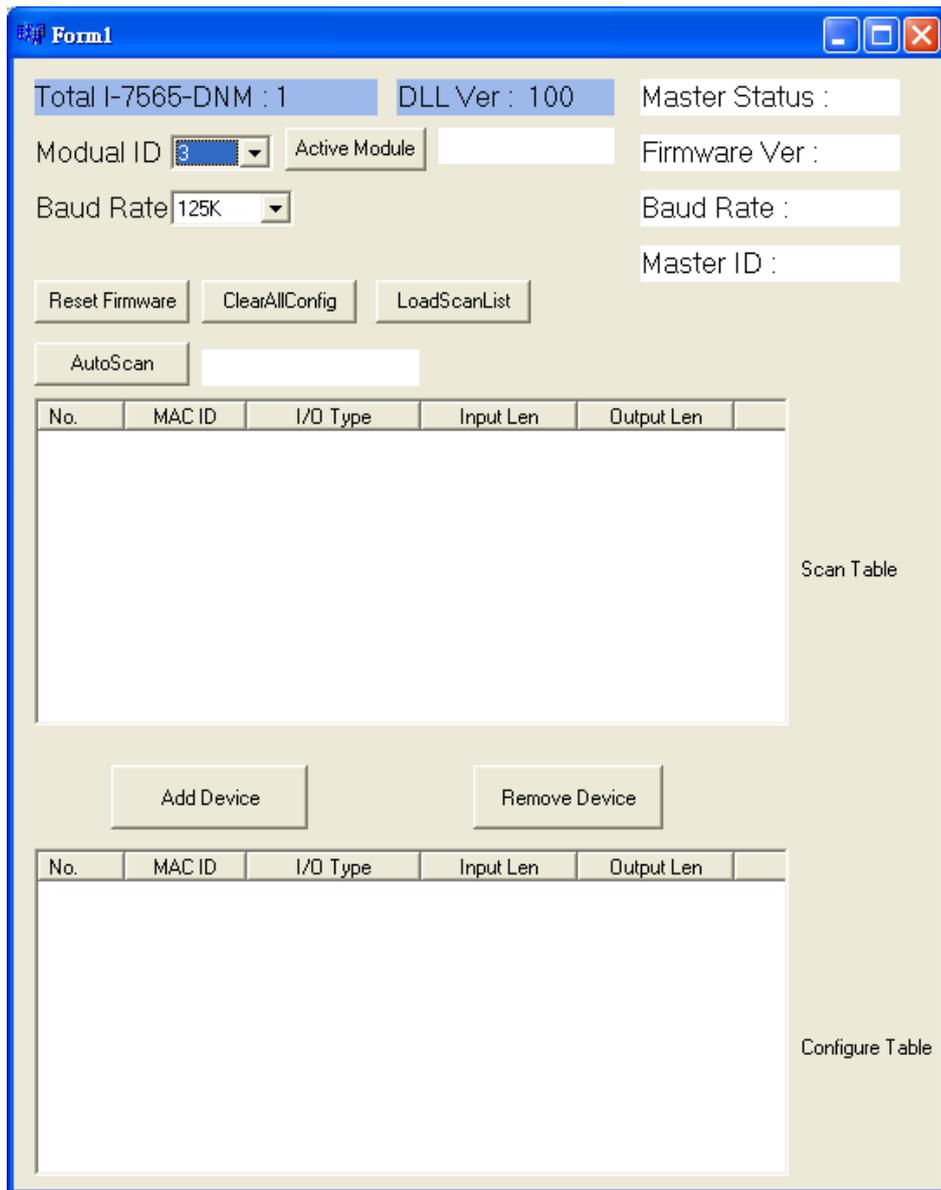
### Step 3 : Auto Scan Network

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.



Form1

Total I-7565-DNM : 1    DLL Ver : 100    Master Status :

Modual ID 3    Active Module    Firmware Ver :

Baud Rate 125K    Baud Rate :

Master ID :

Reset Firmware    ClearAllConfig    LoadScanList

AutoScan

| No.        | MAC ID | I/O Type | Input Len | Output Len |
|------------|--------|----------|-----------|------------|
| Scan Table |        |          |           |            |

Add Device    Remove Device

| No.             | MAC ID | I/O Type | Input Len | Output Len |
|-----------------|--------|----------|-----------|------------|
| Configure Table |        |          |           |            |

Figure 5.5 The screen shoot of BCB\_Demo1

After running the program, the users would see the “Total I-7565-DNM : x” information on the left and up corner of the screen. This function determinates how many I-7565-DNM in your PC automatically. If the program doesn’t find any module, the users should check that the windows driver has been installed successfully or the USB wire connection between the PC and the I-7565-DNM module. Otherwise, if it has found at least one module, the users can continue exercising the demo program.

#### Step 1 : Active Module

Before performing other buttons, the “ActiveMoudle” button should be clicked firstly. The module number means the COM port number in the PC. The drop-down list would show the module’s number which has been plugged in the PC. 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 I-7565-DNM. 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 I-7565-DNM. 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.

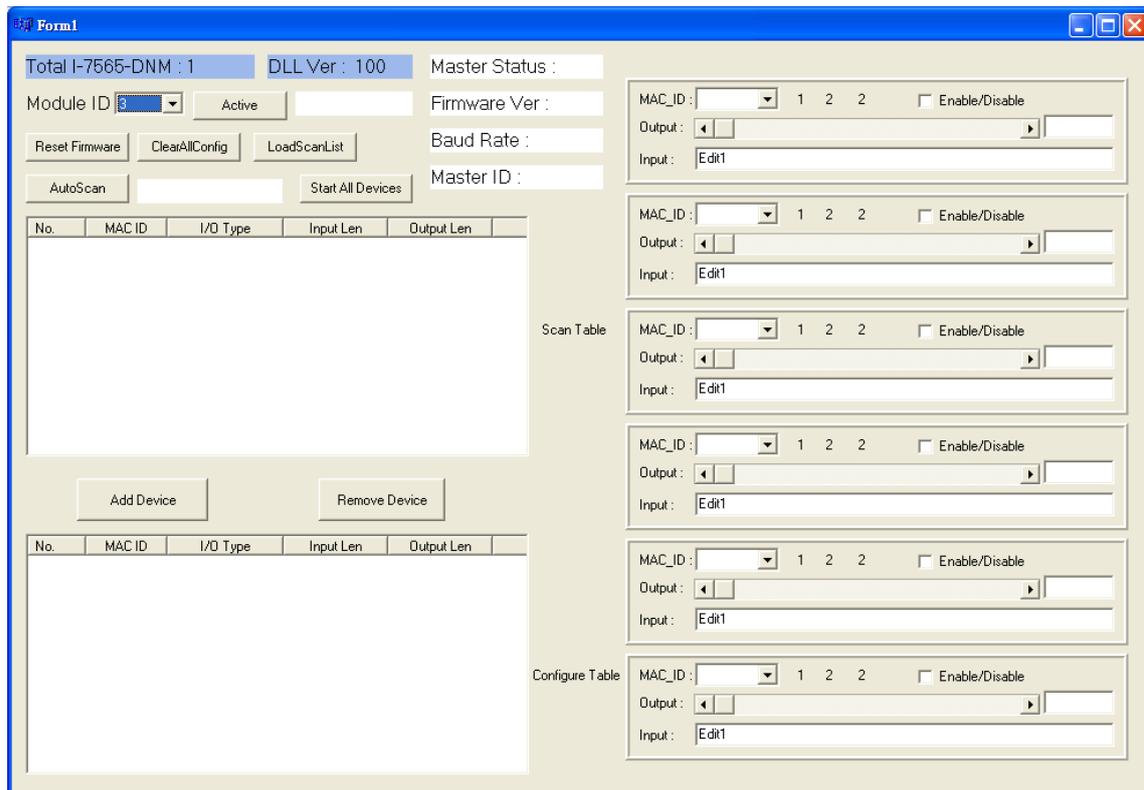


Figure 5.6 The screen shoot of BCB\_Demo2

---

If the users want to know how to configure the slave device information into EEPROM in the I-7565-DNM, please refer to section 5.5.

If “Active Module” 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.

---

# LabVIEW Driver Introduction

## 1.1 Software Installation

The I-7565-DNM LabVIEW 8.x driver is the I-7565-DNM function reference for the DeviceNet master device used on LabVIEW 8.x environment on Windows 2000/XP. Before users use this driver to develop the machine control system, the I-7565-DNM driver must be installed first, because the I-7565-DNM LabVIEW 8.x driver needs to call the function of it. The driver architecture is shown as Figure 6.1.

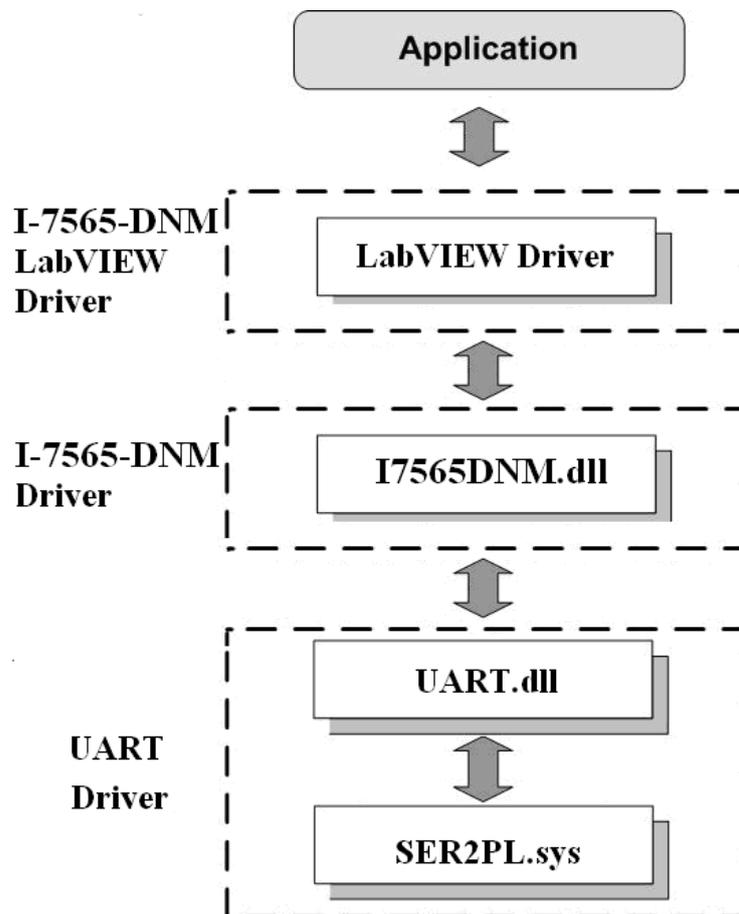


Figure 6.1 Driver concept of I-7565-DNM LabVIEW driver

After completing the LabVIEW driver installation, the directory of I-7565-DNM LabVIEW Driver is C:\ICPDASI-7565-DNM\LabVIEW.

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

I-7565-DNM LabVIEW Function palette is showed as below.

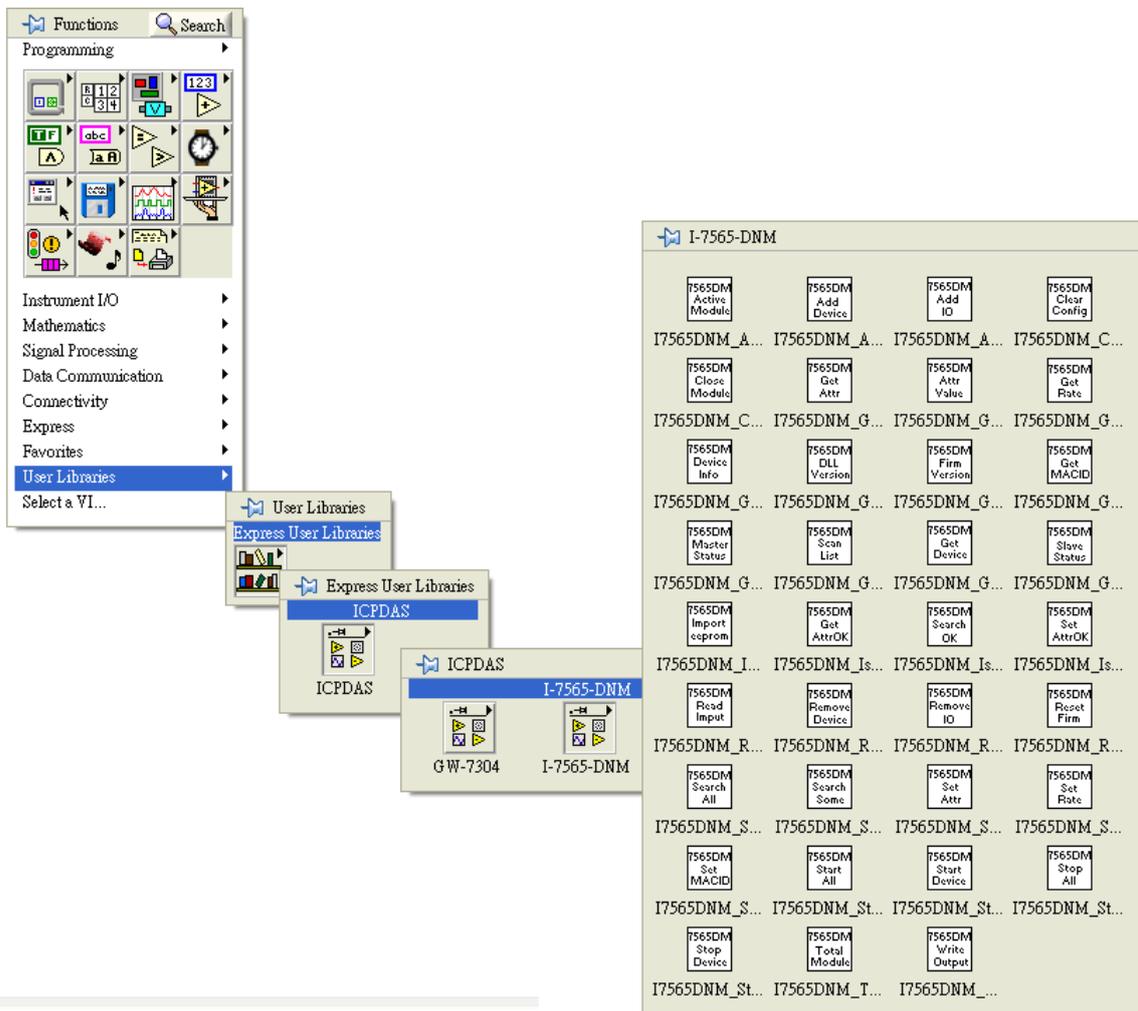


Figure 6.2 I-7565-DNM LabVIEW Function palette

## 1.2 Function description

Every function provides each VI for user to use in the LabVIEW environment. All the functions provided by the I7565DNM.dll are listed in the following table.

Table 6.2.1 Module Functions

| No. | VI<br>ICON   | Function Name                | Description                             |
|-----|--|------------------------------|---|
| 1   |   | I7565DNM_TotalI7565DNMModule | Get total I-7565-DNM modules in the PC  |
| 2   |   | I7565DNM_ActiveModule        | Make I-7565-DNM module active           |
| 3   |   | I7565DNM_CloseModule         | Close the I-7565-DNM module             |
| 4   |  | I7565DNM_GetDLLVersion       | Get the DLL version of the I7565DNM.DLL |

Table 6.2.2 Firmware Functions

| No. | VI<br>ICON  | Function Name               | Description  |
|-----|---|-----------------------------|--|
| 1   |  | I7565DNM_GetFirmwareVersion | Get the version of the firmware inside the I-7565-DNM module |
| 2   |  | I7565DNM_ResetFirmware      | Reset the firmware in the I-7565-DNM module                  |

Table 6.2.3 Operating Functions 1/3

| No. | VI<br>ICON  | Function Name           | Description   |
|-----|---|-------------------------|---|
| 1   |  | I7565DNM_SetMasterMACID | Set the MAC ID of the I-7565-DNM module (DeviceNet Master's MAC ID) |
| 2   |  | I7565DNM_GetMasterMACID | Get the MAC ID of the I-7565-DNM module (DeviceNet Master's MAC ID) |
| 3   |  | I7565DNM_GetBaudRate    | Get the baud rate of the CAN bus                                    |

Table 6.2.4 Operating Functions 2/3

| No. | VI<br>ICON  | Function Name               | Description  |
|-----|---|-----------------------------|--|
| 4   |    | I7565DNM_SetBaudRate        | Set the baud rate of the CAN bus   |
| 5   |    | I7565DNM_GetMasterStatus    | Get the status of the I-7565-DNM module (DeviceNet Master's status) at present                 |
| 6   |    | I7565DNM_GetSlaveStatus     | Get the slave device's status.   |
| 7   |    | I7565DNM_StartDevice        | I-7565-DNM will start to communicate with the specific slave device                            |
| 8   |    | I7565DNM_StopDevice         | I-7565-DNM will stop to communicate with the specific slave device                             |
| 9   |  | I7565DNM_StartAllDevice     | I-7565-DNM will start to communicate with all slave devices                                    |
| 10  |  | I7565DNM_StopAllDevice      | I-7565-DNM will stop to communicate with all slave devices                                     |
| 11  |  | I7565DNM_AddDevice          | Add the specific slave device's information into the I-7565-DNM module (DeviceNet Master)      |
| 12  |  | I7565DNM_RemoveDevice       | Remove the specific slave device's information from the I-7565-DNM module (DeviceNet Master)   |
| 13  |  | I7565DNM_AddIOConnection    | Add I/O information of the specific slave device into the I-7565-DNM module (DeviceNet Master) |
| 14  |  | I7565DNM_RemoveIOConnection | Remove specific slave device's I/O information from the I-7565-DNM module (DeviceNet Master)   |
| 15  |  | I7565DNM_GetAttribute       | Send the get attribute command to the slave device.  |
| 16  |  | I7565DNM_IsGetAttributeOK   | Check whether the slave has replied for the getting command or not.                            |

Table 6.2.5 Operating Functions 3/3

| No. | VI<br>ICON  | Function Name                       | Description   |
|-----|---|-------------------------------------|---|
| 17  |    | I7565DNM_GetAttributeValue          | Get the attribute value of the I7565DNM_GetAttribute  |
| 18  |    | I7565DNM_SetAttribute               | Send the set attribute command to the slave device.   |
| 19  |    | I7565DNM_IsSetAttributeOK           | Check whether the slave has replied for the setting command or not.                           |
| 20  |    | I7565DNM_GetDeviceInfoFrom ScanList | Get specific slave device's I/O information form the Scan List within the I-7565-DNM module.  |
| 21  |  | I7565DNM_GetScanList                | Get the I/O information of all slave devices form the Scan List within the I-7565-DNM module. |
| 22  |  | I7565DNM_ImportEEPROM               | Write the I/O information of all slave devices into the EEPROM within the I-7565-DNM module.  |
| 23  |  | I7565DNM_ClearAllConfig             | Clear all configurations in the EEPROM within the I-7565-DNM module.                          |

Table 6.2.6 Searching Functions

| No. | VI<br>ICON  | Function Name                 | Description   |
|-----|---|-------------------------------|---|
| 1   |  | I7565DNM_SearchAllDevices     | I-7565-DNM will search the DeviceNet network to find out the I/O information of all slave devices.      |
| 2   |  | I7565DNM_SearchSpecificDevice | I-7565-DNM will search the DeviceNet network to find out the I/O information of specific slave devices. |
| 3   |  | I7565DNM_IsSearchOK           | Check whether the I-7565-DNM has searched completely or not.  |
| 4   |  | I7565DNM_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<br>ICON  | Function Name            | Description   |
|-----|---|--------------------------|---|
| 1   |  | I7565DNM_ReadInputData   | Read the input data via I/O connection like Poll, Strobe, COS, Cyclic.                                      |
| 2   |  | I7565DNM_WriteOutputData | Write the output data via I/O connection like Poll, COS, Cyclic. The Strobe doesn't support this operation. |

## 1.3 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.



Figure 6.3.1 The screen shoot of LabVIEW Demo

---

----- [End] -----

---

---