CAN-8124/CAN-8224/CAN-8424
DeviceNet Slave Device

User Manual

Warranty

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

Warning

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

Copyright

Trademark

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

Table of Content

Chapter 1 Introduction

1.1 Overview

The CAN-8x24 series are the DeviceNet remote I/O units produced by ICP DAS for data acquisition and the control system that provides a wide range of capabilities, most specifically for real-time applications. A DeviceNet master operating via a DeviceNet protocol remotely controls the CAN-8124/ CAN-8224/CAN-8424 series products. In others words, the CAN-8124/ CAN-8224/CAN-8424 series can be a DeviceNet slave devices in the CAN on the DeviceNet network. Furthermore, the CAN-8124/CAN-8224/CAN-8424's main control units are small and middle size compact devices, but they offer many good features to the users.

The CAN-8124/CAN-8224/CAN-8424's main control units are based on the modular design which offers many good features to users and provides more flexibility in data acquisition and a control system for the DeviceNet network. In order to expand the I/O channel to make it more flexible, CAN-8124/CAN-8224/CAN-8424 support 1, 2 and 4 expansion slots respectively for user to expand their I/O channel numbers in various DeviceNet applications. Users can configure the i-87K or i-8000 IO series modules to fit their customized applications. The symbol 'x' in the CAN-8x24 represents how many expansion slots there are in the main control unit. Each expansion slot can have one i-87K or i-8000 series I/O module plugged into it. For example, a CAN-8124 has one expansion slot, and a CAN-8424 has 4 expansion slots. All of these main control units follow the DeviceNet specification Volume I, Release 2.0& Volume II, Release 2.0. In addition, ICP DAS also provides the CAN Slave Utility to create the EDS file dynamically and to configure DeviceNet slave devices. Therefore, users can easily apply the CAN-8124/CAN-8224/CAN-8424 embedded controller into the DeviceNet network. The general application architecture is demonstrated in figure 1-1:

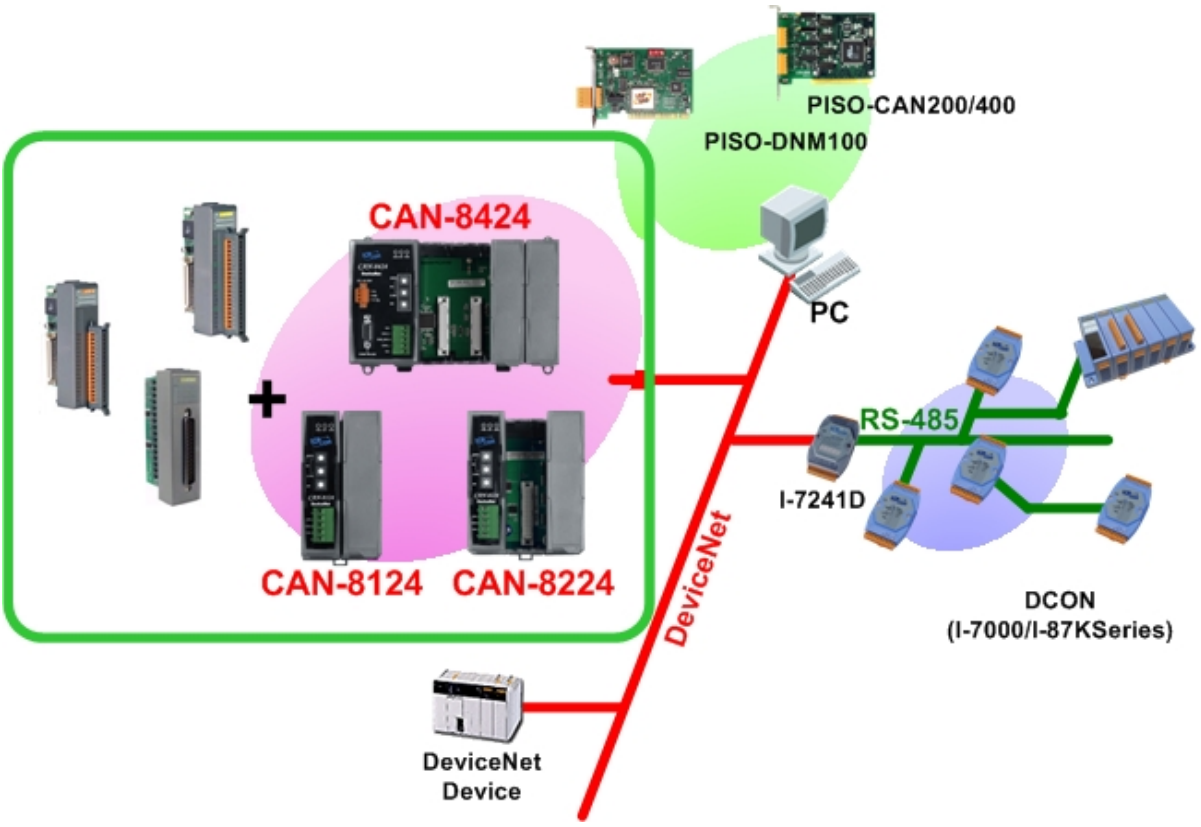


Figure 1-1 DeviceNet application

1.2 Hardware Features

- CPU:80186, 80MHz
- Philip SJA1000 CAN controller
- Philip 82C250 CAN transceiver
- SRAM:512K bytes
- Flash Memory:512K bytes
- EEPROM:2K bytes
- NVRAM: 32 bytes
- Real Time Clock
- Built-in Watchdog Timer
- 16-bit Timer
- PWR LED, NET LED, MOD LED
- Support 1/2/4 expansion I/O slots
- 2500 Vrms isolation on CAN side
- 120Ω terminal resister selected by jumper
- CAN bus interface: ISO/IS 11898-2, 5-pin screw terminal with on-board optical isolators protection.
- Power consumption:20W
- Power requirement: +10VDC to +30VDC (unregulated)
- Operating Temperature:-25°C to +75°C
- Storage Temperature:-30°C to +85°C
- Humidity:5%~95%

COM1 (CAN-8424 only)

- RS-232: TXD,RXD,RTS,CTS,GND
- Communication speed: 115200 bps.
- Configure tool connection

1.3 CAN-8124/CAN-8224/CAN-8424 DeviceNet Features

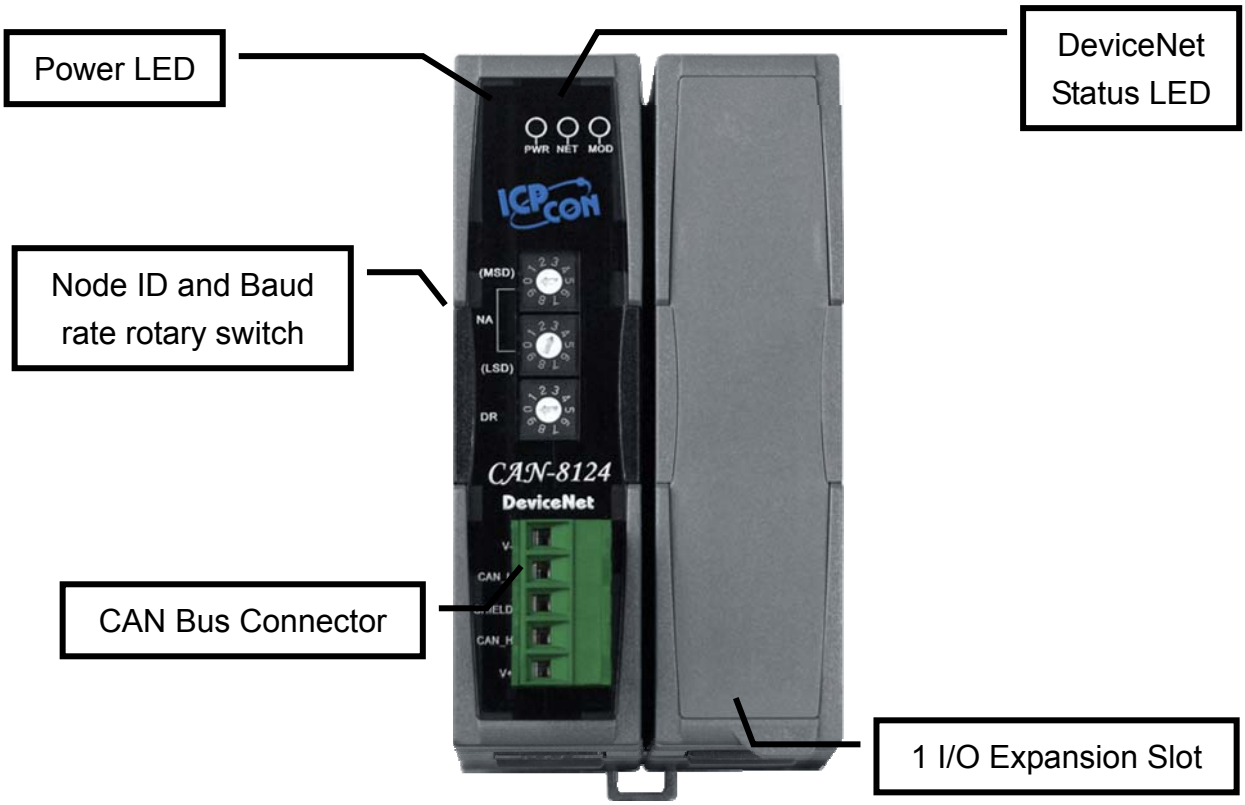
- Complies with DeviceNet specification Volume I, Release 2.0& Volume II, Release 2.0
- Group 2 Only Slave; (non UCMM-capable)
- Supports Predefined Master/slave Connection Set
- Supports Explicit message connection
- Supports Fragmented Explicit Message
- I/O operating modes: Polling, Bit-Strobe, Change of State/Cyclic
- Supports Fragmented IO (maximum 128 bytes output and/or 128 bytes input data)
- Dynamic Assembly Objects Mapping;
- Supports Device Heartbeat message
- Supports Device Shutdown message
- EDS file dynamically
- Support all standard DeviceNet data rate: 125K, 250K and 500K bps
- Data rate and Node Address (MAC ID) configured via rotary switch
- NET, MOD and Power Led directors

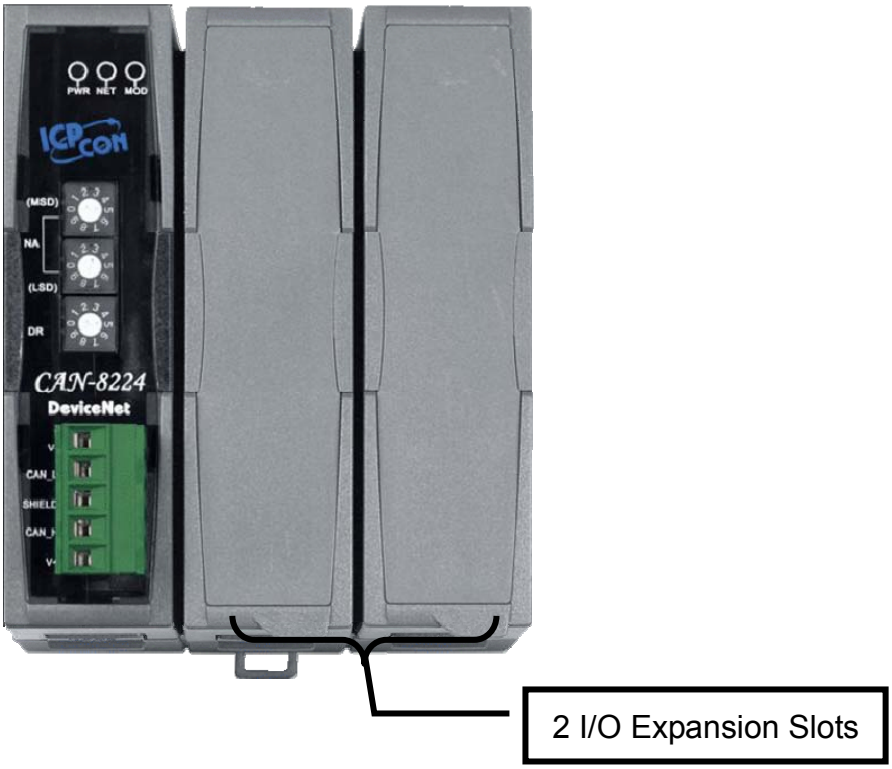
1.4 Utility Feature

- Support i-8k/i-87K modules
- Show i-8k/i-87K modules configuration
- Show Application and assembly objects configuration
- Support IO connection path setting
- Support EDS file creating

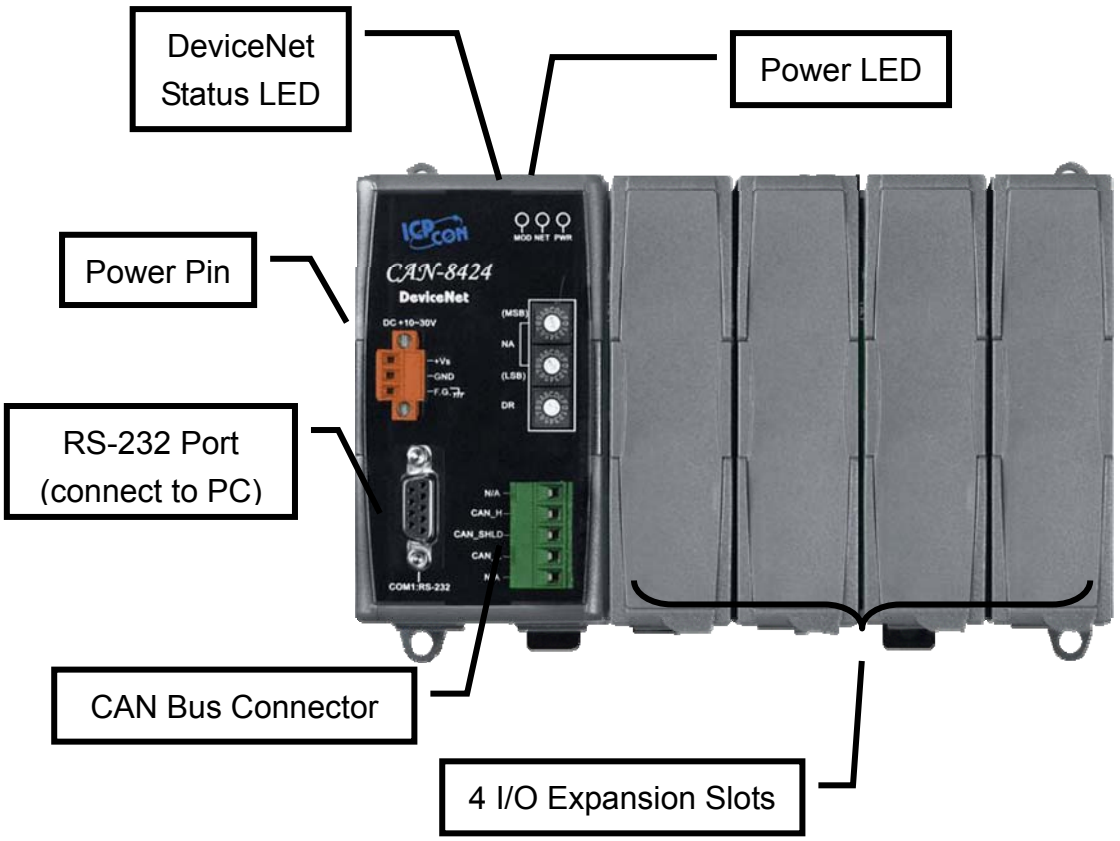
Chapter 2 Hardware Specification

2.1 CAN-8124/CAN-8224 Hardware Structure





2.2 CAN-8424 Hardware Structure



2.3 Wire Connection

In order to minimize the reflection effects on the CAN bus line, the CAN bus line has to be terminated at both ends by two terminal resistances as in the following figure. According to the ISO 11898-2 spec, each terminal resistance is set to 120Ω (or between 108Ω~132Ω). The length related resistance should have 70 mΩ/m. The user should check the resistances of their CAN bus, before they install a new CAN network as in figure 2-1.

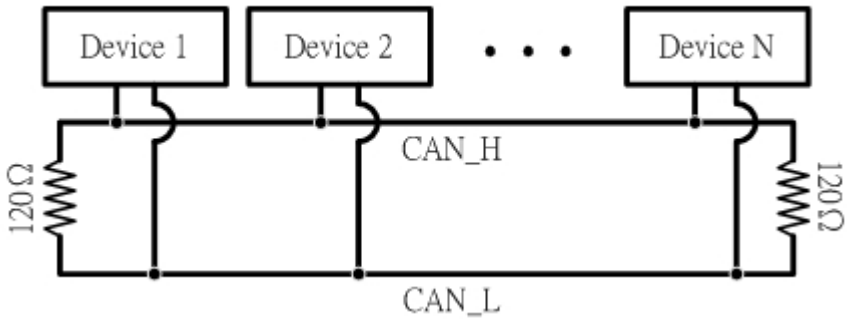


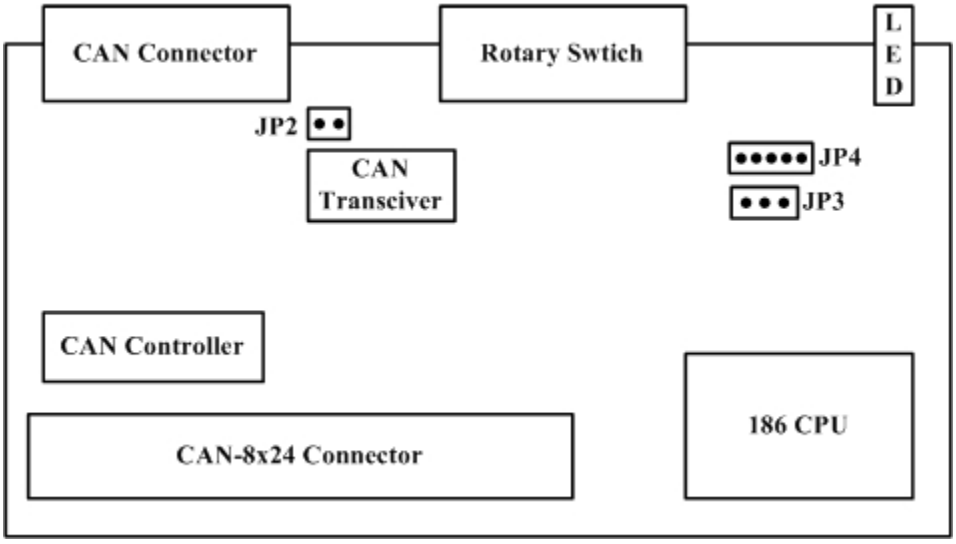
Figure 2-1 wire connections

Moreover, to minimize the voltage drop over long distances, the terminal resistance should be higher than the value defined by ISO 11898-2. Table 2-1 may be used as a reference.

Table 2-1 The relation between bus cable and length

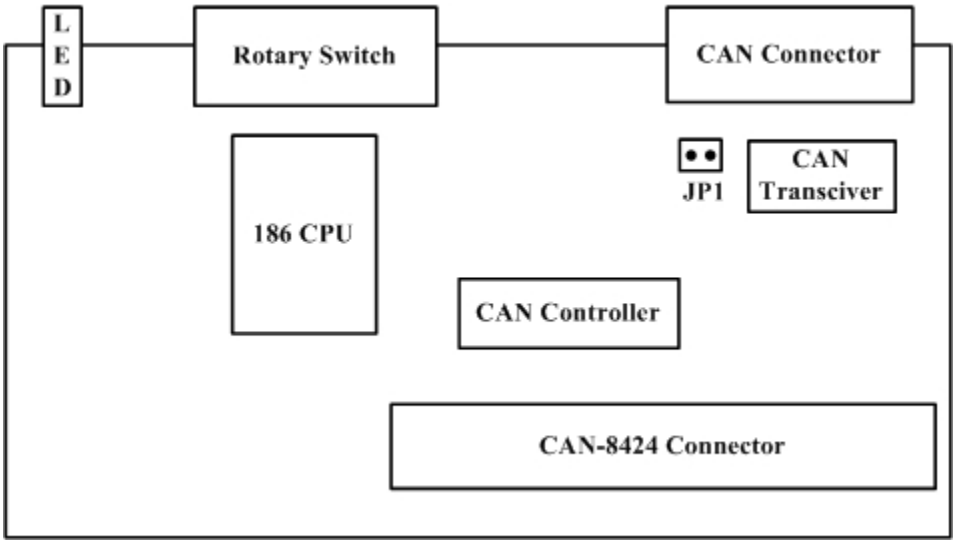
Bus Length (meter)	Bus Cable Parameters		Terminal Resistance (Ω)
	Length Related Resistance (mΩ/m)	Cross Section (Type)	
0~40	70	0.25(23AWG)~ 0.34mm^2(22AWG)	124 (0.1%)
40~300	< 60	0.34(22AWG)~ 0.6mm^2(20AWG)	127 (0.1%)
300~600	< 40	0.5~0.6mm^2 (20AWG)	150~300
600~1K	< 20	0.75~0.8.mm^2 (18AWG)	150~300

In the CAN-8124/CAN-8224/CAN-8424, the 120Ω terminal resistance is supplied. The JP2 for the CAN-8124/CAN-8224 is for terminal resistance. The JP2 position is shown in the following figure.

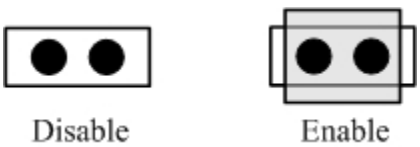


.

JP1 on CAN-8424 is used for adjusting terminal resistance, and its position is shown in the following figure.



The following connection status is presented for if the terminal resister is enable or disable.



The CAN bus baud rate has the high relationship with the bus length. Table 2-2 indicates the corresponding bus length for every kind of baud rate.

Table 2-2 Baud rate and bus lengths for the DeviceNet

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

Note: When the bus length is greater than 1000m, bridge or repeater devices may be needed.

The pin assignments for these CAN-8124/ CAN-8224 and CAN-8424 CAN bus connectors are shown in figure 2-2, figure 2-3, table 2-3 and table 2-4.

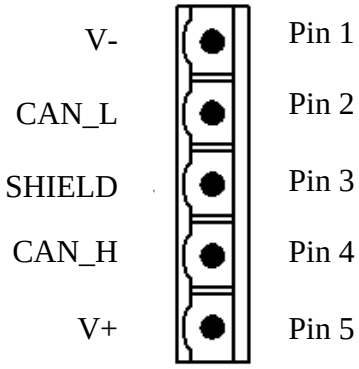


Figure 2-2 The connector pin assignments for the CAN-8124/CAN-8224

Table 2-3 Connector pins of CAN-8124/CAN-8224

Pin No.	Signal	Description
1	V-	Ground (0V)
2	CAN_L	CAN_L bus line (dominant low)
3	SHIELD	Optional CAN Shield
4	CAN_H	CAN_H bus line (dominant high)
5	V+	CAN external positive power supply (CAN-8124/CAN-8224 power)

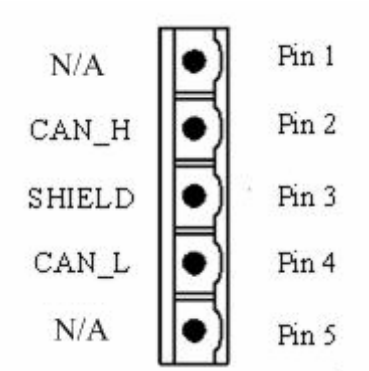


Figure 2-3 The connector pin assignments for the CAN-8424

Table 2-4 Connector pins of CAN-8424

Pin No.	Signal	Description
1	N/A	No use
2	CAN_H	CAN_H bus line (dominant high)
3	SHIELD	Optional CAN Shield
4	CAN_L	CAN_L bus line (dominant low)
5	N/A	No use

2.4 PWR LED

After connecting the CAN-8124/CAN-8224/CAN-8424 with electronic power (the range of input voltage is 10~30VDC), the PWR LED will be turned on. If the Power LED is off after giving the proper voltage, please check the power and load of power supply first. If the situation is not improved, please communicate your problem with your distributor in order to find the solution. The corresponding conditions are given in table 2-5.

Table 2-5 PWR led conditions

condition	status	indicates
Off	No power	No power supply
Solid red	Normal	Device is working

2.5 DeviceNet LED

CAN-8124/CAN-8224/CAN-8424 supplies 2 DeviceNet LED indicators. They are NET LED (Yellow), MOD LED (Green). The Indicators assist maintenance personnel in quickly identifying a problem unit. The LED test is to be performed when the device is powered–up. When the DeviceNet communication events occur, these indicators will be triggered to glitter with different conditions.

2.5.1 MOD LED

This LED provides the devices status. It indicates whether or not the device is operating properly. Table 2-6 shows the conditions for all MOD statuses.

Table 2-6 The MOD led conditions

condition	status	indicates
Off	Normal	
Solid	Critical fault	Device has unrecoverable fault;
Flashing	Non_critical fault	Device has recoverable fault; to recover: Reconfigure device Reset device Perform error recovery

2.5.2 NET LED

The NET LED indicates the current status of the DeviceNet communication link. Table 2-7 shows the conditions for all NET statuses.

Table 2-7 NET led conditions

condition	status	indicates
Off	Off line	Device is not online
Flashing	On line	Device is on line, but not communicating
Init solid	Link failed	(Critical) Device has detected an error that has rendered it incapable of communicating on the link; for example, detected a duplicate node address or network configuration error
Solid	On line, communicating	Device is online and communicating

2.6 NA and DR Rotary Switch

The CAN-8124/CAN-8224/CAN-8424 provide NA(node address) and DR(data rate) rotary switches to set the node ID and baud rate of the device. Refer to the figure 2-4.

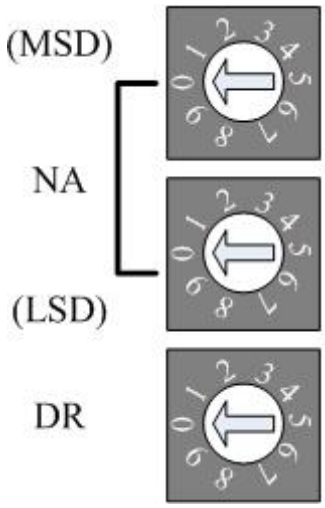


Figure 2-4 NA and DR rotary switches

The node address (MAC ID) and data rate (baud rate) to be used by the device for communication on the network are set via the upper rotary switches. The MSD means the most significant digit of the node address, and LSD represent the low significant digit of the node ID in the decimal format. The node address of the CAN-8124/CAN-8224/CAN-8424 is useless when the value exceeds 64 (decimal format) because the max node value in the DeviceNet is 63. For example, the node address in CAN-8124/CAN-8224/CAN-8424 is 32, if the MSD rotary switch is turned to 3 and the LSD rotary switch is turned to 2 as in figure 2-5.

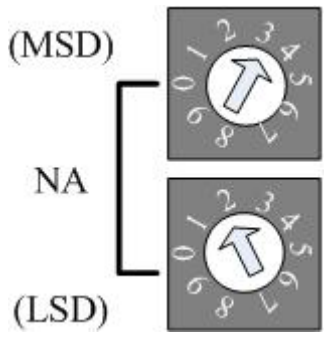


Figure 2-5 NA rotary switches

The lower rotary switch handles the CAN-8124/CAN-8224/CAN-8424 data rate (baud rate). The relationship between the rotary switch value and the practical baud rate is displayed in the following table.

Table 2-7

Rotary Switch Value	Baud rate (K BPS)
0	125
1	250
2	500

For example, the setting data rate is 125 Kbps as in figure 2-6.

Figure 2-6 NA rotary switches

Note: If the “DR” rotary switch for the CAN-8x24 is set to ‘9’, the CAN-8x24 will get into its initial mode. The DeviceNet firmware built in the CAN-8x24 will not be executed. Before users employ the utility tool to configure the CAN-8424, the initial mode is needed. And, because the CAN-8124/CAN-8224 has no RS-232 COM Port, it is necessary to run the utility tool in the off-line mode if users want to get the EDS file of users’ CAN-8124/CAN-8224.

Furthermore, when the CAN-8124/CAN-8224/CAN-8424 is started up, the DeviceNet firmware will check these rotary switches. Any illegal value on these rotary switches will cause the CAN-8124/CAN-8224/CAN-8424 to fail when booting-up.

Note: If users set the illegal values of the rotary switch, the MOD led would flash when powering up. In this condition, users must configure the legal values for the switches and reset the device, and then the device will work normally.

2.7 Module Support

CAN-8124/CAN-8224/CAN-8424 supports many kinds of DI, DO, AI and AO modules for the i-8000/i-87K series modules. When users want to use these modules on the DeviceNet network, they only need to plug these modules into the CAN-8124/CAN-8224/CAN-8424 I/O expansion slots. Then, the DeviceNet firmware built in the CAN-8124/CAN-8224/CAN-8424 will search through them automatically organizing them into their corresponding DeviceNet entries. For more information, please refer to the "CAN-8124/CAN-8224/CAN-8424 Support Module Table".

2.8 Application Flowchart

The procedure listed in figure 2-7 displayed how to use the CAN-8124/CAN-8224/ CAN-8424. Users can refer to the procedure to apply the CAN-8124/CAN-8224/CAN-8424 devices to the DeviceNet application.

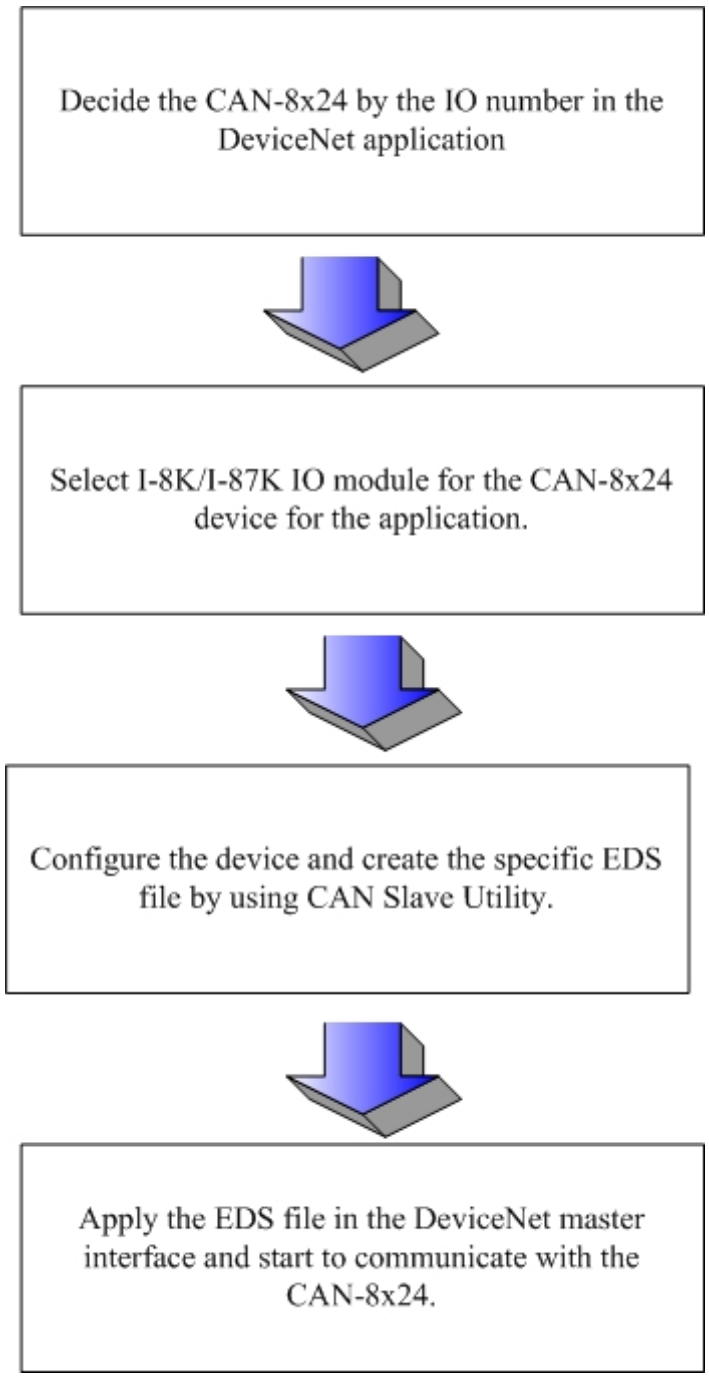


Figure 2-7 CAN-8124/CAN-8224/CAN-8424 application flow-chart

Chapter 3 DeviceNet System

3.1 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 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 of the kinds of network protocols based on the CAN bus which are mainly used for machine control in embedded networks, such as in textile machinery, printing machines, injection molding machinery, or packaging machines. DeviceNet is a low level network that provides connections between simple industrial devices (sensors, actuators) and higher level devices (controllers). It allows direct peer to peer data exchange between nodes in an organized and, if necessary, deterministic manner. The network management functions specified in DeviceNet simplifies project design, implementation and diagnosis by providing standard mechanisms for network start-up and error management. DeviceNet defines a connection-based scheme to facilitate all application communications. A DeviceNet connection provides a communication path between multiple endpoints. The endpoints of a connection are applications that need to share data. Figure 3.1 shows the DeviceNet layer in the control and information layers.

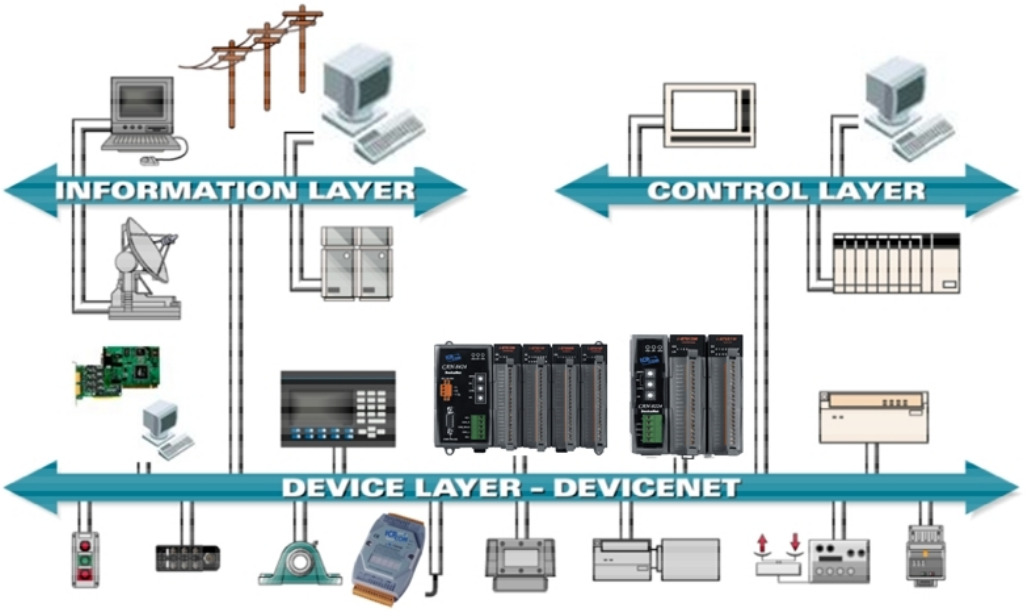


Figure 3.1 DeviceNet layer

DeviceNet Communication Protocol is based on the concept of connections. One must establish a connection with a device in order to exclude information with that device. To establish a connection, each gateway implements a Predefined Master/Slave Connection Set through the DeviceNet network. After establishing the explicit connections, the connection is then used to move information from one node to the other. Once IO connections have been established, I/O data may be moved among devices in the network. The 11-bit CAN identifier is used to identify the connection. DeviceNet defines four separate groups of 11-bit CAN identifiers: Group 1, Group 2, Group 3, and Group 4 described in Table 3.1. With respect to Connection Based Messages, the Connection ID is placed within the CAN Identifier Field. With this in mind, the below figure also describes the components for a DeviceNet Connection ID. Because of the arbitration scheme defined by CAN, Group 1 messages have a higher priority than group 2 messages and group 2 messages have higher priority than group 3 messages and so on. This prioritization must be taken into consideration when establishing connections.

Table 3.1 DeviceNet's Use of the CAN Identifier Field

IDENTIFIER BITS											IDENTITY USAGE	HEX RANGE
10	9	8	7	6	5	4	3	2	1	0		
0	Group 1 Message ID				Source MAC ID						Group 1 Messages	000 – 3ff
1	0	MAC ID						Group 2 Message ID			Group 2 Messages	400 – 5ff
1	1	Group 3 Message ID			Source MAC ID						Message Group 3	600-7bf
1	1	1	1	1	Group 4 Message ID						Group 4 Messages	7c0–7ef

CAN-8124/CAN-8224/CAN-8424 provides the Predefined Master/Slave Connection Set for users to establish connections. The Predefined Master/Slave Connection Set is a set of Connections that facilitate communications typically seen in a Master/Slave relationship. Many of the steps involved in the creation and configuration of an application-to-application connection have been removed within the Predefined Master/Slave Connection Set definition. This, in turn, presents the means by which a communication environment can be established using less network and device

resources. The CAN Identifier Fields associated with the Predefined Master/Slave Connection Set are shown in Table 3.2. The table defines the Identifiers that are to be used with all connection based messages involved in the Predefined Master/Slave Connection Set and, as such, it also illustrates the produced_connection_id and consumed_connection_id attributes associated with Predefined Master/Slave Connection Objects.

Note: The Master is the device that gathers and distributes I/O data for the process controller. Slaves are the devices from which the Master gathers I/O data and to which the Master distributes I/O data.

Table 3.2 DeviceNet Identifiers

IDENTIFIER BITS											IDENTITY USAGE	HEX RANGE
10	9	8	7	6	5	4	3	2	1	0		
0	Group 1 Message ID				Source MAC ID						Group 1 Messages	000 – 3ff
0	1	1	0	0	Source MAC ID						Slave's Multicast Poll Response	
0	1	1	0	1	Source MAC ID						Slave's I/O Change of State or Cyclic Message	
0	1	1	1	0	Source MAC ID						Slave's I/O Bit–Strobe Response Message	
0	1	1	1	1	Source MAC ID						Slave's I/O Poll Response or Change of State/Cyclic Acknowledge Message	
1	0	MAC ID						Group 2 Message ID			Group 2 Messages	400 – 5ff
1	0	Source MAC ID						0	0	0	Master's I/O Bit–Strobe Command Message	
1	0	Multicast MAC ID						0	0	1	Master's I/O Multicast Poll Command Message	
1	0	Destination MAC ID						0	1	0	Master's Change of State or Cyclic Acknowledge Message	
1	0	Source MAC ID						0	1	1	Slave's Explicit/ Unconnected Response Messages/ Device Heartbeat Message/ Device Shutdown Message	
1	0	Destination MAC ID						1	0	0	Master's Explicit Request Messages	
1	0	Destination MAC ID						1	0	1	Master's I/O Poll Command/Change of State/Cyclic Message	
1	0	Destination MAC ID						1	1	0	Group 2 Only Unconnected Explicit Request Messages (reserved)	
1	0	Destination MAC ID						1	1	1	Duplicate MAC ID Check Messages	

A device within a DeviceNet network is represented by the below object model. The object model provides a template for organizing and implementing

the Attributes (data), Services (methods or procedures) and behaviors of the components within a DeviceNet product. Figure 3.2 depicts the object model for CAN-8124/CAN-8224 (Group 2 server Only). The next section would explain these objects. More detailed information about Predefined Master/Slave Connection Set is described in the next section.

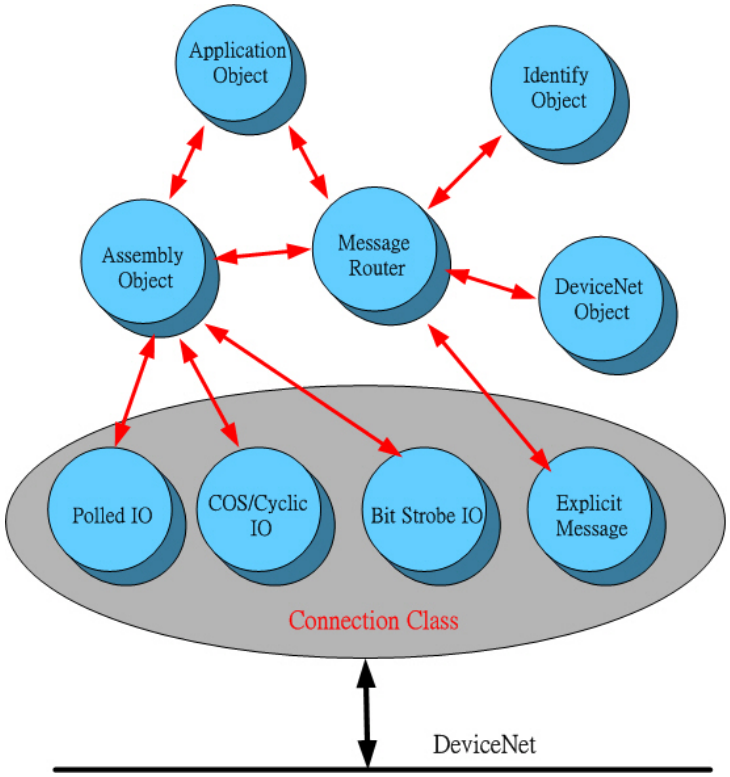


Figure 3.2 Object models of CAN-8124/CAN-8224/CAN-8424

3.2 Predefined Master Slave Connection Set

The CAN-8124/CAN-8224/CAN-8424 is a “Predefined Master Slave Connection Set” device. Users must understand these connection sets to know how to operate the device. The following section explains what the “Predefined Master Slave Connection Set” is.

When using the Predefined Master Slave Connection Set, DeviceNet allows devices with fewer resources to take part in DeviceNet network communication. For this reason a set of identifiers has been reserved within Message Group 2 to simplify the movement of I/O and configuration data typically seen in Master/Slave relationships. The steps which are necessary to create and configure a connection between devices have been removed within the Predefined Set. The Predefined Master Slave Connection Set allows for the establishing of a DeviceNet communication environment using less network and Device resources. The Predefined Set contains one Explicit Messaging Connection and allows for several different I/O connections which include a Bit Strobe Command/Response, Poll Command/Response, Change of State and Cyclic connections. The following types of messages are processed by a DeviceNet Slave

3.2.1 Explicit Messages

Explicit Request Messages are used to perform operations such as reading and writing attributes. Explicit response Messages indicate the results to attempt to service an Explicit Request massage. Within a Slave Explicit Request and Response Messages are received/transmitted by a single Connection Object. The architecture is as figure 3.3.

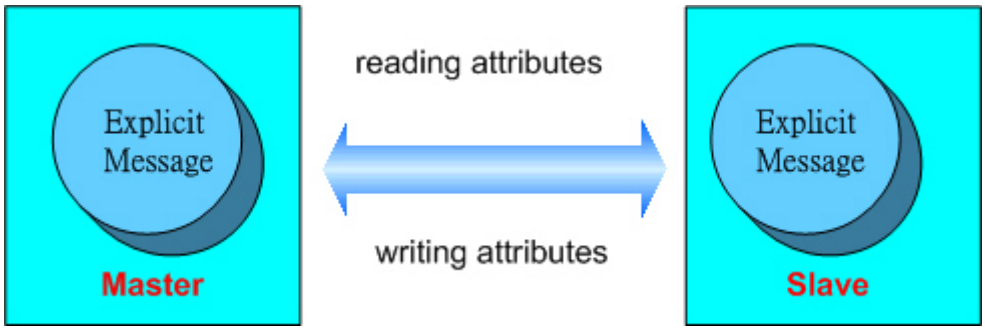


Figure 3.3 the architecture of Explicit message

3.2.2 I/O Bit Strobe Messages

The Bit-Strobe Command is an I/O message that is transmitted by the Master. A Bit-Strobe Command has multicast capabilities. Multiple Slaves can receive and react to the same Bit Strobe Command. The Bit-Strobe response is an I/O message that a Slave transmits back to the Master when the Bit-Strobe Command has been received. Within a Slave the two messages are received/ transmitted by a single Connection Object. The architecture is as in figure 3.4.

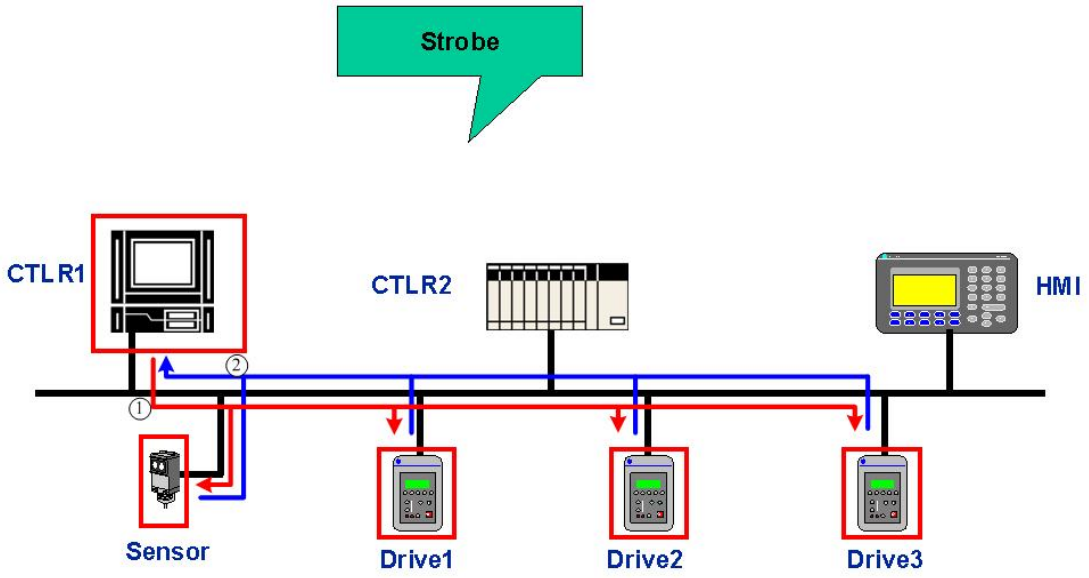


Figure 3.4 the architecture of IO bit strobe message

3.2.3 I/O Poll Messages

The Poll Command is a command that is transmitted by the Master. A Poll Command is directed towards a single, specific Slave (point-to-point connection). A Master must transmit a separate Poll command message for each one of its Slaves that is to be polled. The Poll-Response is an I/O message that the Slave transmits back to the Master when the Poll Command is received. Within a Slave the two messages are received or transmitted by a single Connection Object as in figure 3-5.

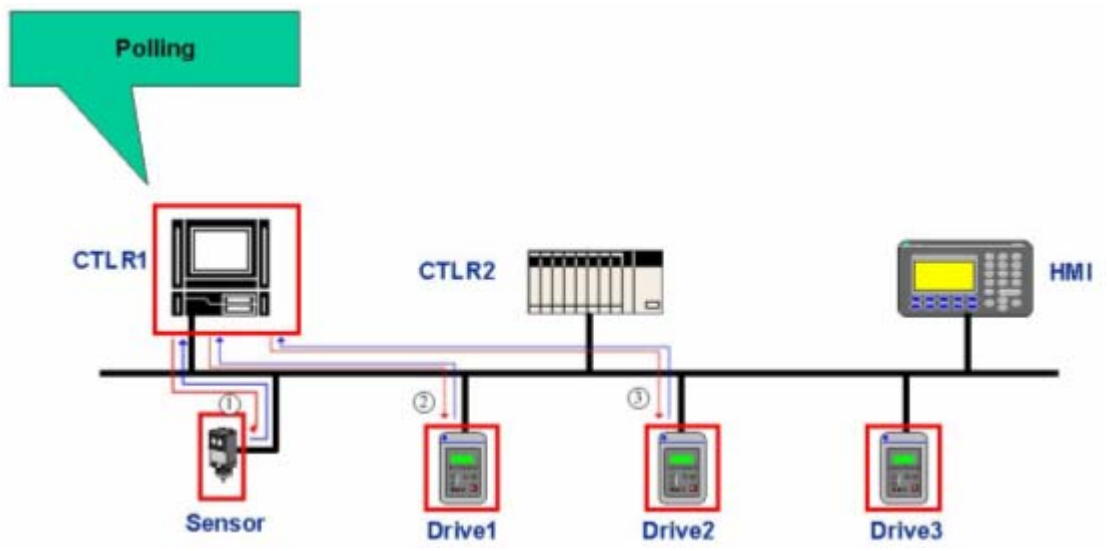


Figure 3-5 the architecture of poll IO message

3.2.4 I/O Change of State/Cyclic Messages

The Change of State Message is transmitted by either the Master or the Slave. A Change of Sate/Cyclic is directed towards a single specific node (point-to-point). An Acknowledge Message may be returned in response to this message. Within either the Master or the Slave the producing Change of State Message and consuming Acknowledge Messages are received or transmitted by one Connection Object. The consuming Change of State and producing Acknowledge Message are received or transmitted by a second Connection Object. The architecture for these messages can be seen in figure 3-6 and figure 3-7.

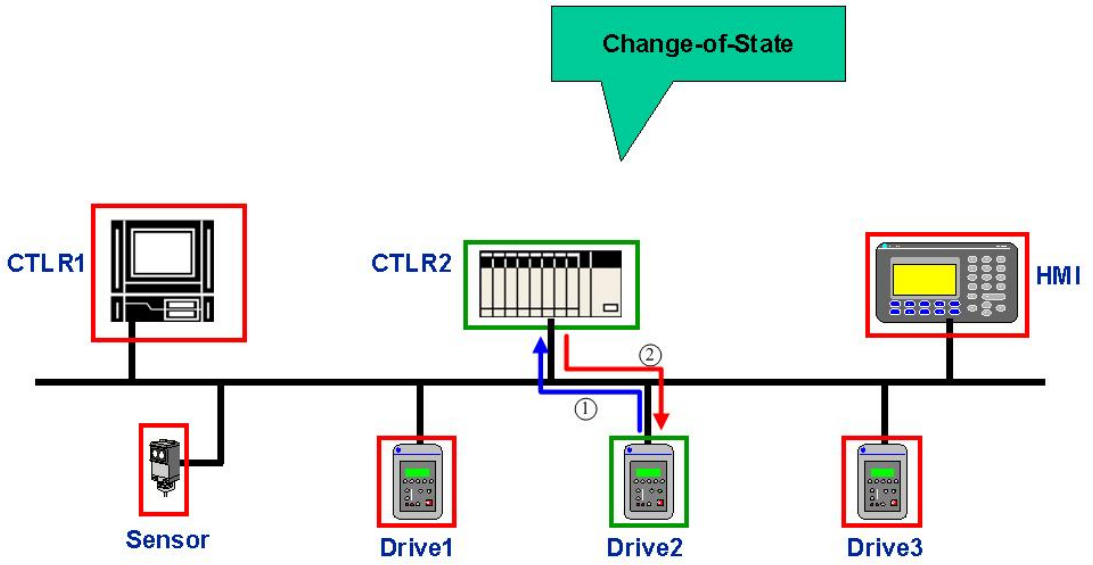


Figure 3-6 the architecture of the IO COS message

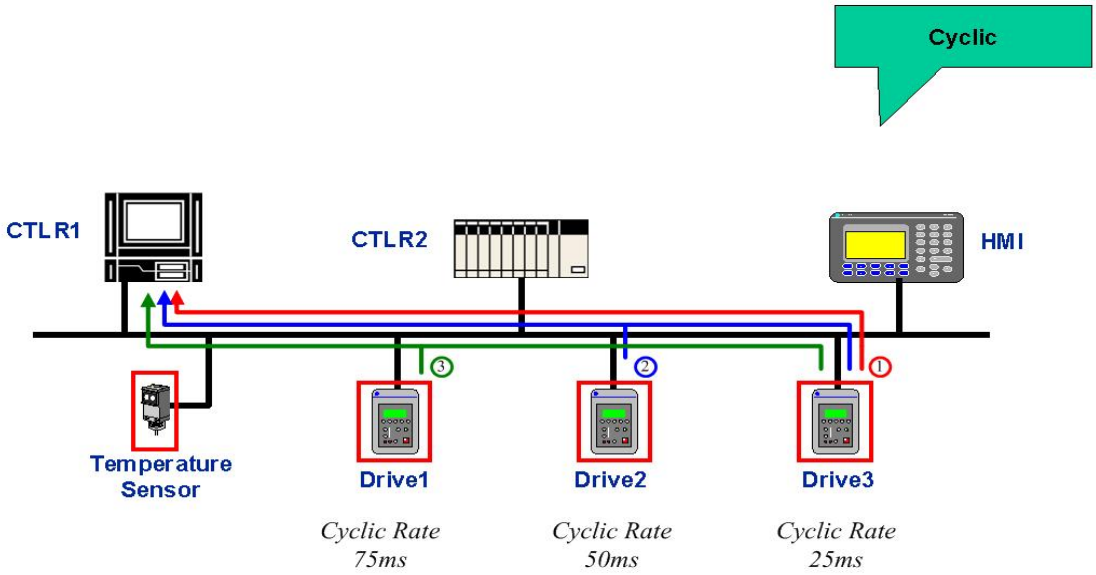


Figure 3-7 the architecture of the IO Cyclic message

3.3 EDS File

An Electronic Data Sheet is an external disk that contains information about configurable attributes for a device, including the object addresses of each parameter. The following figure shows the configuration of a device through the configuration tool that supports an EDS. The application objects in the device represent the destination addresses for the configuration data. These addresses are encoded in the EDS. ICP DAS provides users with the CAN Slave utility software they need to create suitable EDS files. The EDS file system architecture is presented in Figure 3-8.

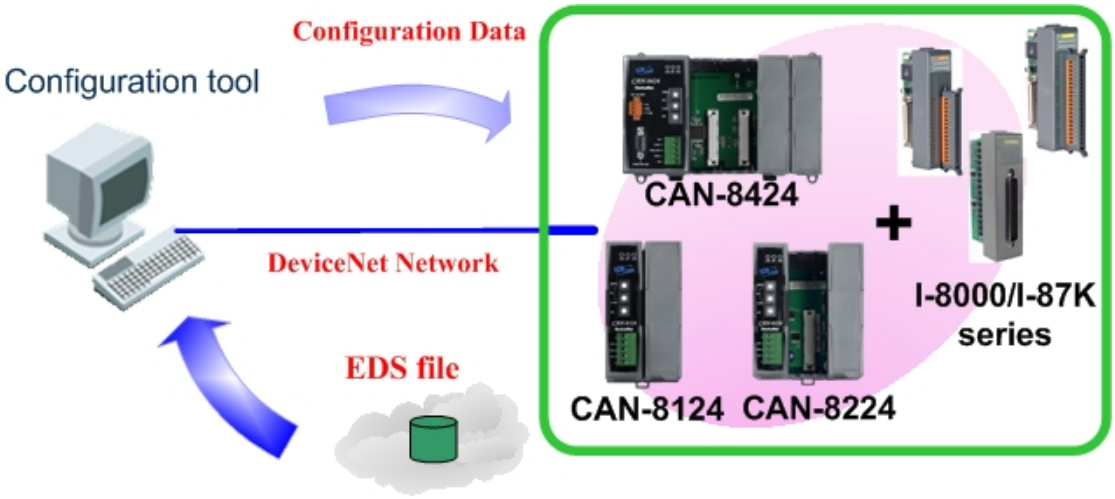


Figure 3-8 the architecture of EDS file

EDS provides information about the device's configuration data according to the following:

- context
- content
- format

The information in an EDS file allows configuration tools to provide informative screens that guide a user through the steps necessary to configure a device. ICP DAS provides CAN utilities, so that users can setup their own EDS file. You can use the EDS file in the DeviceNet Master to access DeviceNet Slave devices. The CAN utility is a very powerful tool for the DeviceNet network. Users can get more DeviceNet Slave information. The CAN utility can scan i-8K/i-87K IO modules connected to the CAN-8124/

CAN-8224. However, users can only use off-line mode to create EDS files of CAN-8124/CAN-8224 with the specific IO modules. It also provides a graph interface for users to be able to make up their own EDS file system. For more detailed information on this topic, please refer to the next section.

Chapter 4 DeviceNet Profile Area

This chapter is for users who want to get and understand more detailed information relating to the CAN-8124/CAN-8224/CAN-8424 device when using the DeviceNet protocol. This section documents the detailed functions for each object class that is implemented in the DeviceNet network.

4.1 DeviceNet Statement of Compliance

General Device Data

Conforms to DeviceNet Specification	Volume i-Release 1.1
Vendor Name	ICP DAS
Device Profile Name	ICPDAS-CAN-8x24
Production Revision	2.0.1

DeviceNet Physical Conformance Data

Network Power Consumption(Max)	Open-Hardwired
Isolated Physical Layer	Yes
LEDs Supported	Yes
MAC ID Setting	Switch
Default MAC ID	Switch
Communication Rate Setting	Switch
Predefined Master/Slave Connection Set	Group 2 Only Server
Fragmented Explicit Message implemented	yes

4.2 Identity Object (Class ID: 0x01)

This object provides the identification of and general information about the device.

Class Attribute

Attribute ID	Attribute name	Data Type	Method	Value
0x01	Revision	UINT	Get	0001
0x02	Max Instance	UINT	Get	1

Class Service

Service Code	Service name	Support
0x0E	Get_Attribute_Single	Yes

Instance Attribute

Attribute ID	Description	Method	DeviceNet Data Type	Value
1	Vendor UINT	Get	UINT	803
2	Product type UINT	Get	UINT	0x00
3	Product code	Get	UINT	2(CAN-8124/CAN-8224)/3(CAN-8424)
4	Revision vision Major Revision Minor Revision	Get	Struct of USINT USINT	2.01
5	Status	Get	WORD	0
6	Serial number	Get	UDINT	2(CAN-8124/CAN-8224)/3(CAN-8424)
7	Product name	Get	Short_String	ICPDAS-CAN-8x24
10	Heartbeat Interval	Get/Set	USINT	0(default)

Instance Service

Service Code	Service name	Support
0x0E	Get_Attribute_Single	Yes
0x10	Set_Attribute_Single	Yes
0x05	Reset	Yes

4.3 DeviceNet Object (Class ID:0x03)

The DeviceNet Object is used to provide the configuration and status of a physical attachment to DeviceNet.

Class attribute

Attribute ID	Attribute name	Data Type	Method	Value
0x01	revision	UINT	Get	2

Class service

Service Code	Service name	Support
0x0E	Get_Attribute_Single	Yes

Instance attribute

Attribute ID	Description	Method	DeviceNet Data Type	Value
1	MAC ID	Get	USINT	Range 0-63
2	Baud Rate	Get	USINT	Range 0-2
3	BOI	Get/Set	BOOL	0
4	Bus-off counter	Get/Set	USINT	0
5	Allocation information	Get/Set	STRUCT	
6	MAC ID Switch Changed	Get	BOOL	0=No Change 1=Change since last Reset or Power-up
7	Baud Rate Switch Changed	Get	BOOL	0= No Change 1= Change since last Reset or Power-up
8	MAC ID Switch Value	Get	USINT	Range 0-99
9	Baud Rate Switch Value	Get	USINT	Range 0-9

Instance Service

Service Code	Service name	Support
0x0E	Get_Attribute_Single	Yes
0x10	Set_Attribute_Single	Yes

4.4 Assembly Object (Class ID: 0x04)

The Assembly Object binds attributes of multiple objects, which allows data to or from each object to be sent or received over a single connection. Assembly objects can be used to bind input data or output data. The terms of ”input” and ”output” are defined from the network’s point of view. An input will produce data on the network and an output will consume data from the network.

Class attribute

Attribute ID	Attribute name	Data Type	Method	Value
0x01	Revision	UINT	Get	2
0x02	Max Instance	UINT	Get	By slot number

Class service

Service Code	Service name	Support
0x0E	Get_Attribute_Single	Yes

Instance attribute

Attribute ID	Description	Method	DeviceNet Data Type	Value
0x03	Data	Get/Set	Defined by users	By IO modules plugged in the device

Note: In CAN-8x24, the start-up number of assembly instance is 0x64.

Instance service

Service Code	Service name	Support
0x0E	Get_Attribute_Single	Yes
0x10	Set_Attribute_Single	Yes

4.5 Application Object (Class ID:0x64)

Application objects are the interfaces between an application and the DeviceNet Layer. The attributes of application Objects contain the data for the application, which are accessed and exchanged via DeviceNet. DeviceNet accesses application data by invoking read and write functions. These functions need to be provided by an Application Object. DeviceNet provides Get_Attribute_Single and Set_ Attribute_Single to read and write i-7K/i-87K IO modules.

Class attribute

Attribute ID	Attribute name	Data Type	Method	Value
0x01	Revision	UINT	Get	3
0x02	Max Instance	UINT	Get	By slot number

Class service

Service Code	Service name	Support
0x0E	Get_Attribute_Single	Yes

Instance attribute

Attribute ID	Description	Method	Data Type	Default Value
0x01	Module name	Get	WORD	0
0x02	Module Type	Get	CHAR	0
0x03	Configuration	Get	Depend on the number of channels	0
0x04	Total Channels	Get	CHAR	0
0x05	Total Length	Get	CHAR	0
0x06	Reserved	Get	CHAR	0
0x07	DO Length	Get	CHAR	0
0x08	AO Length	Get	CHAR	0
0x09	DI Length	Get	CHAR	0
0x0A	AI Length	Get	CHAR	0
0x0B	DO channel num	Get	CHAR	0
0x0C	AO channel num	Get	CHAR	0
0x0D	DI channel num	Get	CHAR	0
0x0E	AI channel num	Get	CHAR	0

Attribute ID	Description	Method	Data Type	Default Value
0x0F	Output Safe Value Flag	Get/Set	CHAR	0
0x10	Output Safe Value	Get/Set	Defined by module channel num	0
0x11	87K DI Counter Channel	Get/Set	Defined by module channel num	0
0x12	Clear 87K DI Counter Channel Value	Set	Defined by module channel num	0
0x13	87K DI Counter Value	Get	UINT	0
0x14	DO data	Set	Defined by module channel num	0
0x15	AO data	Set	Defined by module channel num	0
0x16	DI data	Get	Defined by module channel num	0
0x17	AI data	Get	Defined by module channel num	0
0x18	Clear Counter module	Set	CHAR	0
0x19	87K Counter module's Input Mode	Get/Set	CHAR	0

Application instance attribute interpreter:

Attribute name	Brief Description of Attribute
Module Name	The number corresponds to the modules name. For example: The plugged-in module is i-8055. The corresponding module name in DeviceNet is 8055.
Module Type	This attribute contains the module type of the plugged-in modules. 0- DO module 1- DI module

Attribute name	Brief Description of Attribute
	2- AO module 3- AI module 4- DO & DI module 5- DIO8 module 6- Counter & DO module 7- Counter
Configuration	This attribute includes the module configuration. In DI/DO module, the value is 0x40. Please refer to the module manual for more information relating to analog module configuration. Every channel in analog modules corresponds to the relating byte of the attribute.
Total Channels	The total channel attribute indicates the total channel number for the module.
Total Length	This attribute includes the total length of this module. In analog modules, the unit of length is 2 bytes. And 1 bit is corresponding to a channel in the digital modules.
DO Length	The attribute indicates the DO length of the module.
AO Length	The attribute indicates the AO length of the module.
DI Length	The attribute indicates the DI length of the module.
AI Length	The attribute indicates the AI length of the module.
DO channel num	The attribute indicates the DO channel number of the module.
AO channel num	The attribute indicates the AO channel number of the module.
DI channel num	The attribute indicates the DI channel number of the module.
AI channel num	The attribute indicates the AI channel number of the module.
Output safe value flag	The attribute indicates enable or disable the output safe value. 0: Disable; Others: Enable
Output safe value	The attribute indicates the output safe value of the module that you want to set.
87K DI Counter Channel	The attribute indicates which 87K DI counter channel is used
Clear 87K DI Counter Channel Value	The attribute indicates which 87K DI counter channel's value that you want to clear.
87K DI Counter Value	The attribute indicates which 87K DI counter channel's value that you want to get.

Attribute name	Brief Description of Attribute
DO data	The attribute indicates the DO data of the module.
AO data	The attribute indicates the AO data of the module.
DI data	The attribute indicates the DI data of the module.
AI data	The attribute indicates the AI data of the module.
Clear Counter module	Clear counter value of counter N. N: channel of counter module.
87K Counter module's Input Mode	The attribute indicates the 87K counter module's input mode that you want to set or get.

Instance service

Service Code	Service name	Support
0x0E	Get_Attribute_Single	Yes
0x10	Set_Attribute_Single	Yes

4.6 Connection Object (Class ID:0x05)

This section presents the externally visible characteristics of the Connection Objects associated with the Predefined Master/Slave Connection Set within Slave devices.

Note: In CAN-8424, the default IO connection path is its Assembly Object. Please use the CAN Slave Utility to set the IO connection path. For CAN-8124/CAN-8224, it is needed to set the connection path via the CAN bus via DeviceNet explicit message.

Connection Instance ID	Description
1	References the Explicit Messaging Connection into the Server
2	References the Poll I/O Connection
3	References the Bit–Strobe I/O Connection
4	References the Slave's Change of State or Cyclic I/O Connection

Class attribute

Attribute ID	Attribute name	Data Type	Method	Value
0x01	Revision	UINT	Get	1

Class service

Service Code	Service name	Support
0x0E	Get_Attribute_Single	Yes

4.6.1 Explicit connection

Instance (0x01) attribute

Attribute ID	Description	DeviceNet Data Type	Method	Default Value
0x01	state	USINT	Get	3
0x02	instance_type	USINT	Get	0
0x03	transportClass_trigger	BYTE	Get	0x83
0x04	produced_connection_id	UINT	Get	Table 3.2
0x05	consumed_connection_id	UINT	Get	Table 3.2
0x06	initial_comm_characteristics	BYTE	Get	0x21
0x07	produced_connection_size	UINT	Get	0x20
0x08	consumed_connection_size	UINT	Get	0x20
0x09	expected_packet_rate	UINT	Get	0x09c4
0x0C	watchdog_timeout_action	USINT	Get	1
0x0D	produced_connection_path_ length	UINT	Get	0
0x0E	produced_connection_path	EPATH	Get	Empty
0x0F	consumed_connection_path_length	UINT	Get	0
0x10	consumed_connection_path	EPATH	Get	Empty
0x11	production_inhibit_time	UINT	Get	0

Instance service

Service Code	Service name	Support
0x0E	Get_Attribute_Single	Yes
0x10	Set_Attribute_Single	Yes

4.6.2 Poll I/O connection

Instance (0x02) attribute

Attribute ID	Description	DeviceNet Data Type	Method	Default Value
0x01	state	USINT	Get	0x01
0x02	instance_type	USINT	Get	0x01
0x03	transportClass_trigger	BYTE	Get	0x83
0x04	produced_connection_id	UINT	Get	Table 3.2
0x05	consumed_connection_id	UINT	Get	Table 3.2
0x06	initial_comm_characteristics	BYTE	Get	0x01
0x07	produced_connection_size	UINT	Get	No specified default
0x08	consumed_connection_size	UINT	Get	No specified default
0x09	expected_packet_rate	UINT	Get	0
0x0C	watchdog_timeout_action	USINT	Get	0
0x0D	produced_connection_path_ length	UINT	Get	No specified default
0x0E	produced_connection_path	EPATH	Get	No specified default
0x0F	consumed_connection_path_length	UINT	Get	No specified default
0x10	consumed_connection_path	EPATH	Get	No specified default
0x11	production_inhibit_time	UINT	Get	0

Instance service

Service Code	Service name	Support
0x0E	Get_Attribute_Single	Yes
0x10	Set_Attribute_Single	Yes

4.6.3 Bit–Strobe I/O Connection

Instance (0x03) attribute

Attribute ID	Description	DeviceNet Data Type	Method	Default Value
0x01	state	USINT	Get	0x01
0x02	instance_type	USINT	Get	0x01
0x03	transportClass_trigger	BYTE	Get	0x83
0x04	produced_connection_id	UINT	Get	Table 3.2
0x05	consumed_connection_id	UINT	Get	Table 3.2
0x06	initial_comm_characteristics	BYTE	Get	0x02
0x07	produced_connection_size	UINT	Get	No specified default
0x08	consumed_connection_size	UINT	Get	0x08
0x09	expected_packet_rate	UINT	Get	0
0x0C	watchdog_timeout_action	USINT	Get	0
0x0D	produced_connection_path_ length	UINT	Get	No specified default
0x0E	produced_connection_path	EPATH	Get	No specified default
0x0F	consumed_connection_path_length	UINT	Get	No specified default
0x10	consumed_connection_path	EPATH	Get	No specified default
0x11	production_inhibit_time	UINT	Get	0

Instance service

Service Code	Service name	Support
0x0E	Get_Attribute_Single	Yes
0x10	Set_Attribute_Single	Yes

4.6.4 Change of State or Cyclic I/O Connection (Acknowledge)

Instance (0x04) attribute (Acknowledge)

Attribute ID	Description	DeviceNet Data Type	Method	Default Value
0x01	state	USINT	Get	0x01
0x02	instance_type	USINT	Get	0x01
0x03	transportClass_trigger	BYTE	Get	0x02
0x04	produced_connection_id	UINT	Get	Table 3.2
0x05	consumed_connection_id	UINT	Get	Table 3.2
0x06	initial_comm_characteristics	BYTE	Get	0x01
0x07	produced_connection_size	UINT	Get	No specified default
0x08	consumed_connection_size	UINT	Get	No specified default
0x09	expected_packet_rate	UINT	Get	0
0x0C	watchdog_timeout_action	USINT	Get	0
0x0D	produced_connection_path_ length	UINT	Get	No specified default
0x0E	produced_connection_path	EPATH	Get	No specified default
0x0F	consumed_connection_path_length	UINT	Get	No specified default
0x10	consumed_connection_path	EPATH	Get	20h 2Bh 24h 01h
0x11	production_inhibit_time	UINT	Get	0

Instance service

Service Code	Service name	Support
0x0E	Get_Attribute_Single	Yes
0x10	Set_Attribute_Single	Yes

4.6.5 Change of State or Cyclic I/O Connection (Unacknowledge)

Instance (0x04) attribute (Unacknowledge)

Attribute ID	Description	DeviceNet Data Type	Method	Default Value
0x01	state	USINT	Get	0x01
0x02	instance_type	USINT	Get	0x01
0x03	transportClass_trigger	BYTE	Get	0x00
0x04	produced_connection_id	UINT	Get	Table 3.2
0x05	consumed_connection_id	UINT	Get	0xFFFF
0x06	initial_comm_characteristics	BYTE	Get	0x0F
0x07	produced_connection_size	UINT	Get	No specified default
0x08	consumed_connection_size	UINT	Get	No specified default
0x09	expected_packet_rate	UINT	Get	0
0x0C	watchdog_timeout_action	USINT	Get	0
0x0D	produced_connection_path_ length	UINT	Get	No specified default
0x0E	produced_connection_path	EPATH	Get	No specified default
0x0F	consumed_connection_path_length	UINT	Get	0
0x10	consumed_connection_path	EPATH	Get	Empty
0x11	production_inhibit_time	UINT	Get	0

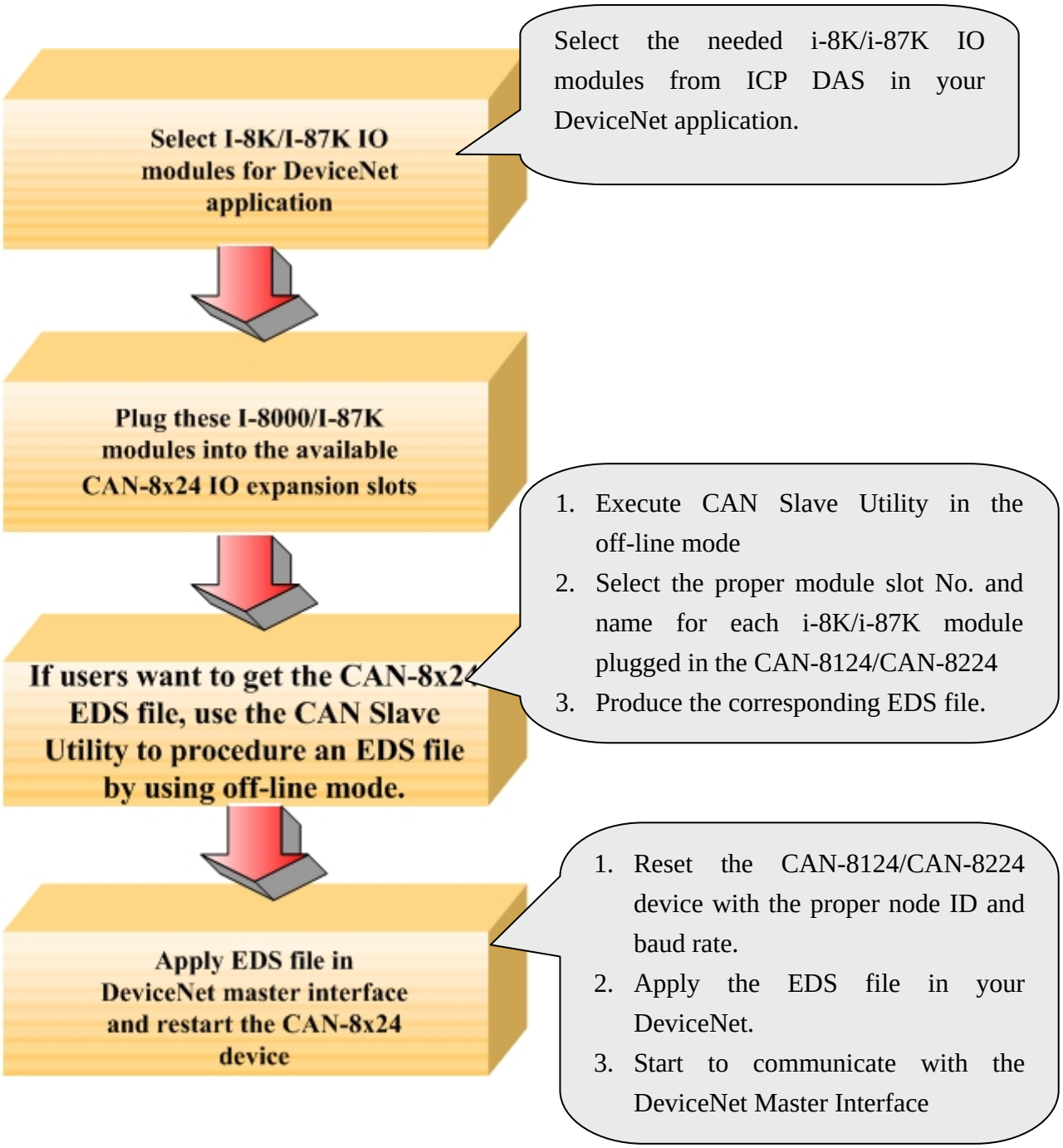
Instance service

Service Code	Service name	Support
0x0E	Get_Attribute_Single	Yes
0x10	Set_Attribute_Single	Yes

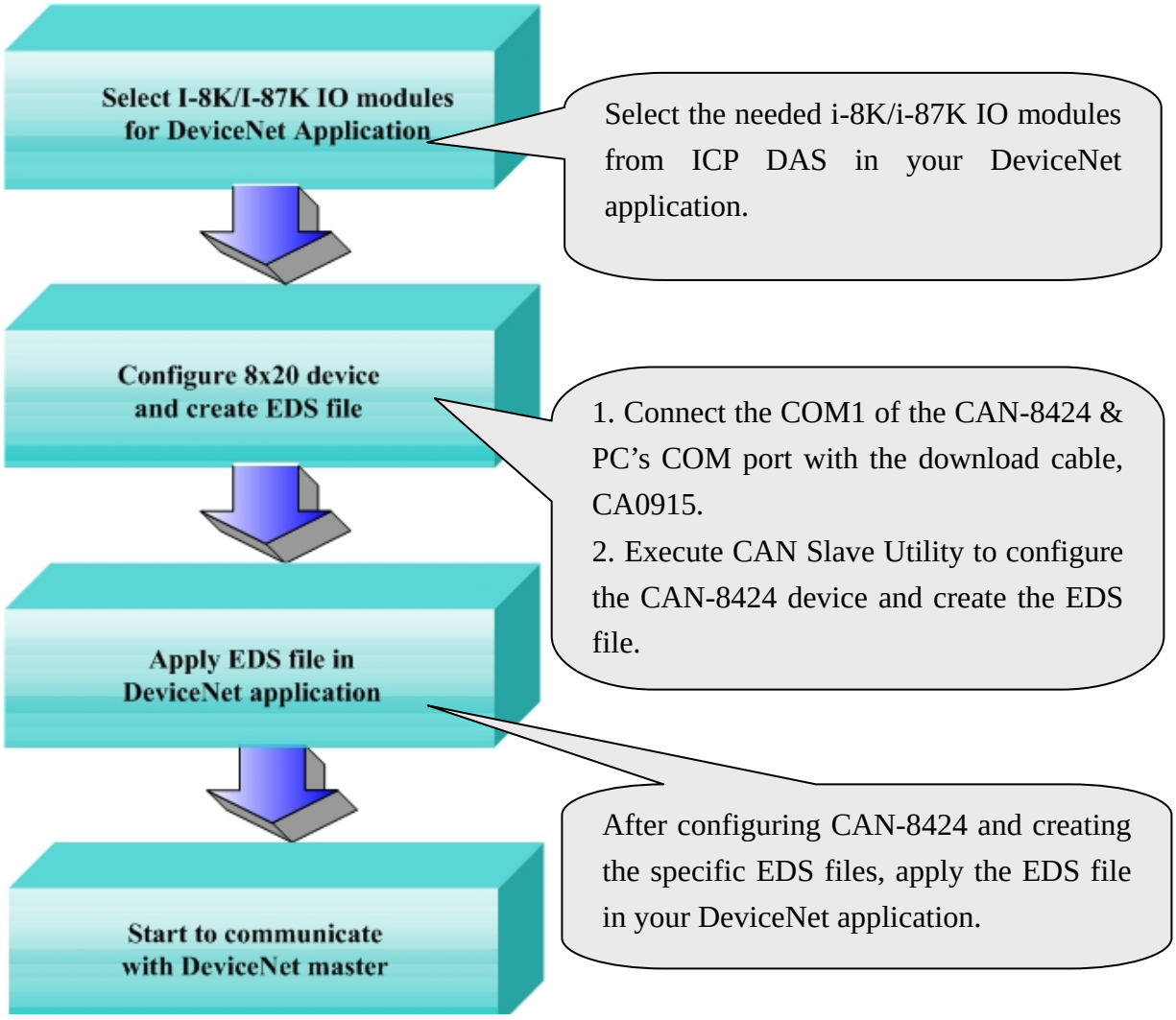
Note：The first assembly instance number in the CAN-8124/CAN-8224/ CAN-8424 is 0x64. The second is 0x65. The priority of Assembly instances follows the sequence of DO, AO, DI, AI and Counter. There are many examples for users to go through in order to understand Assembly objects. Please refer to the next section.

Chapter 5 Configuration & Getting Start

5.1 CAN-8124/CAN-8224 Configuration Flowchart



5.2 CAN-8424 Configuration Flowchart



5.3 CAN Slave Utility Overview

The CAN Slave Utility is designed for the CAN-8124/CAN-8224/ CAN-8424. It provides the following functions.

- Configures the input range of the i-8000 and i-87K AI/AO modules plugged in the CAN-8124/CAN-8224/CAN-8424.
- Scans the i-8000 or i-87K modules in the CAN-8424. Then, it creates EDS files to match the scan results in the on-line mode.
- Produces the EDS files by using the off-line method for CAN-8124/CAN-8224/ CAN-8424.
- Shows the important information which is useful in the DeviceNet network. Such as Assembly instances and Application instance attributes.

For the reason that all i-8K/i-87K AI/AO parameter configurations can be done by using explicit messages for the DeviceNet specification, the CAN-8124/CAN-8224/CAN-8424 can work directly without using the CAN Slave Utility if users don't need to use the CAN-8424 EDS file creation system in the on-line mode, That is to say that users can turn on the CAN-8124/CAN-8224 and directly apply it into the DeviceNet network by using default settings. Users can get the EDS file by using the CAN Slave Utility in the off-line mode. When the AI/AO channel's configurations are needed, use explicit message to modify the AI/AO parameter configurations. For more detailed information, please refer to section 7.2.

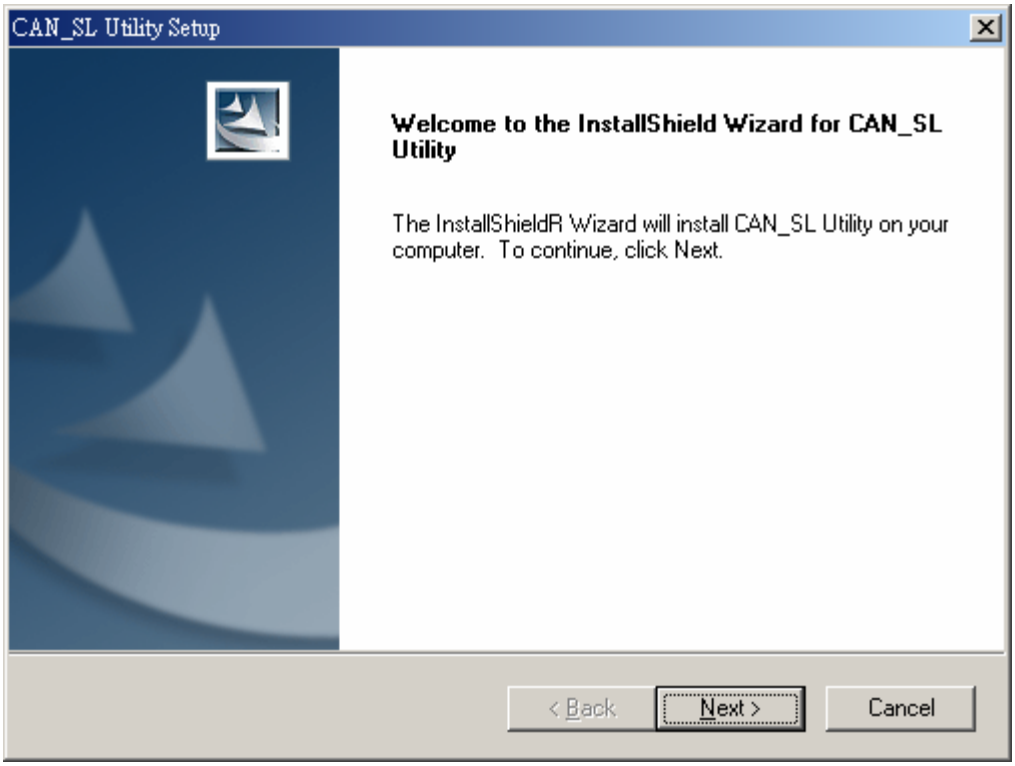
5.4 Configuration with the CAN Slave Utility

Installing the CAN Slave Utility

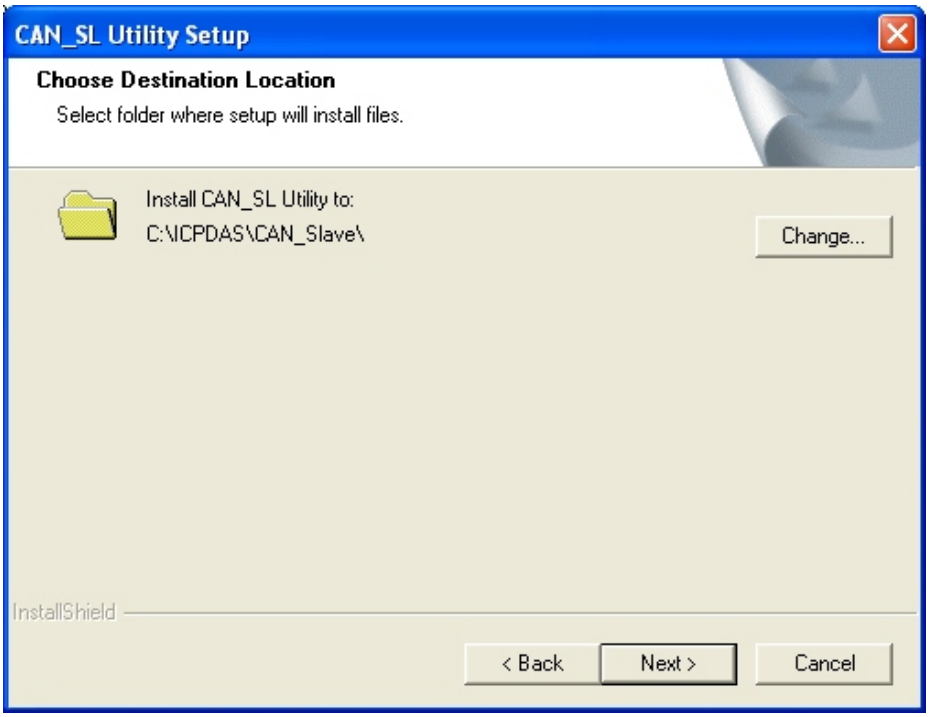
Step 1: Download the CAN Slave Utility setup file from the web site http://www.icpdas.com/download/can/Remote_IO.htm or CD-ROM disk following the path of “Fieldbus_CD:/DeviceNet/Slave/CAN-8x24/Utility/.

Step 2: Execute the setup.exe file to install the CAN Slave Utility.

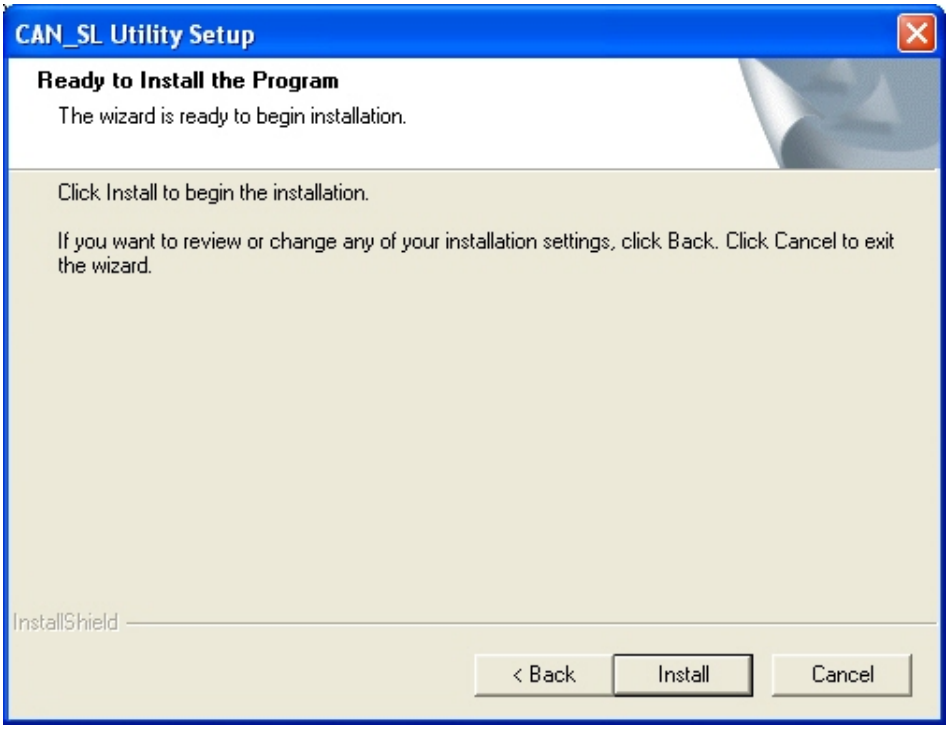
Step 3: A “Welcome” window will pop up to prompt the user to begin installation.



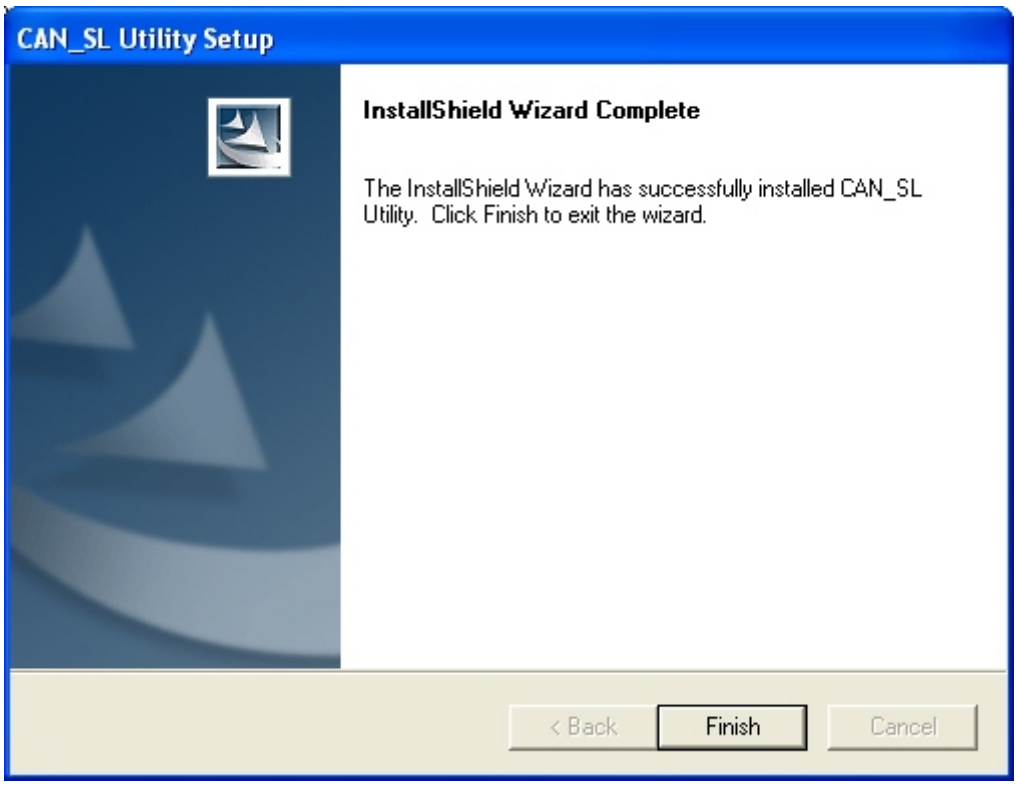
Step 4: Click the “Next” button and then a “Choose Destination Location” window will pop up for deciding the installation path.



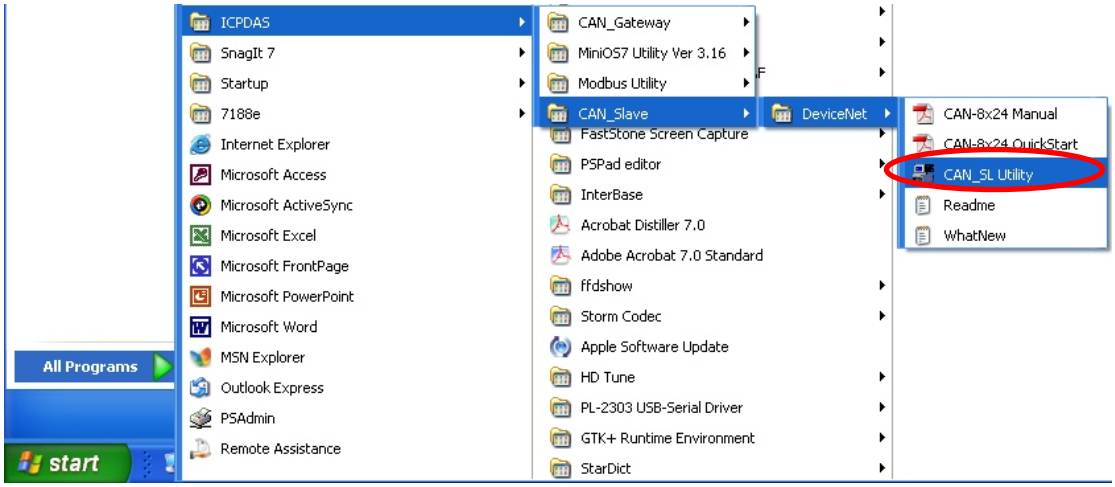
Step 5: Click the “Next” button. A “Ready to Install the Program” window will pop up. Here, press the “Install” button to install the Utility.



Step 6: Click the “Next” button and start to install the CAN Slave Utility to the system. After finishing this process, the following figure will be displayed prompting users to “Finish” the successful completion of the installation.



Step 7: After finishing the installation of the CAN Slave Utility, users can find the CAN_SL Utility as shown in the following screenshot.

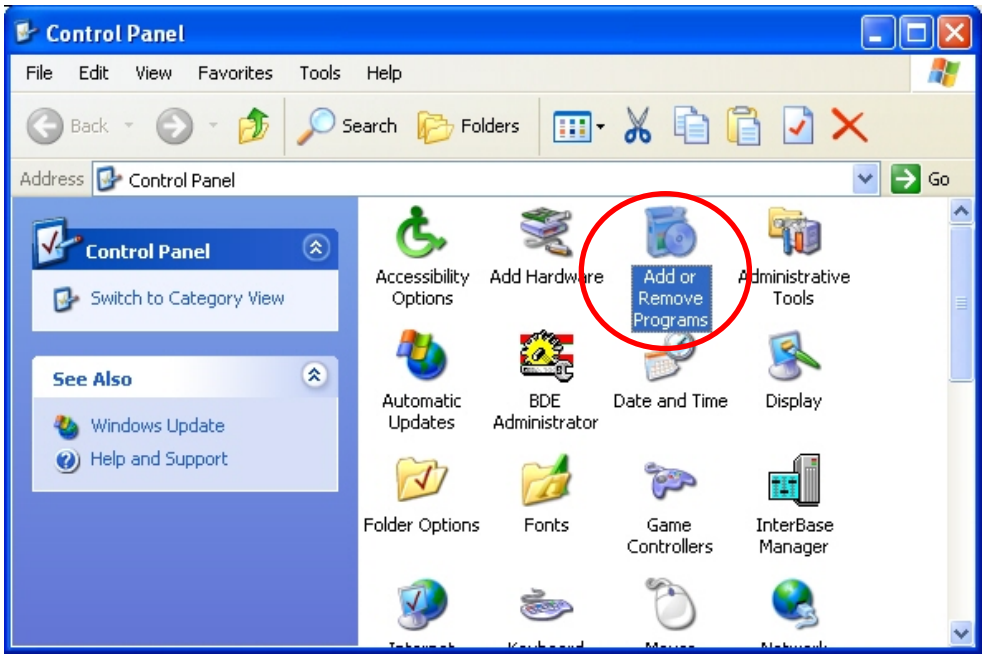


Uninstall CAN Slave Utility

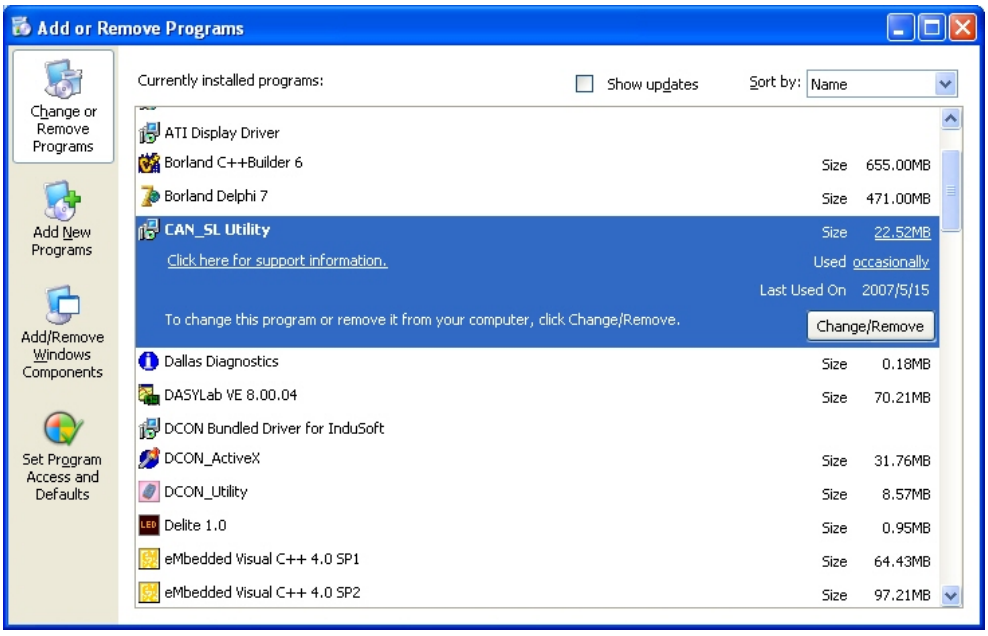
You can uninstall the CAN_SL Utility software by the following means described below.

Step 1: Click “Start” in the task bar, and then click the Settings/Control Panel.

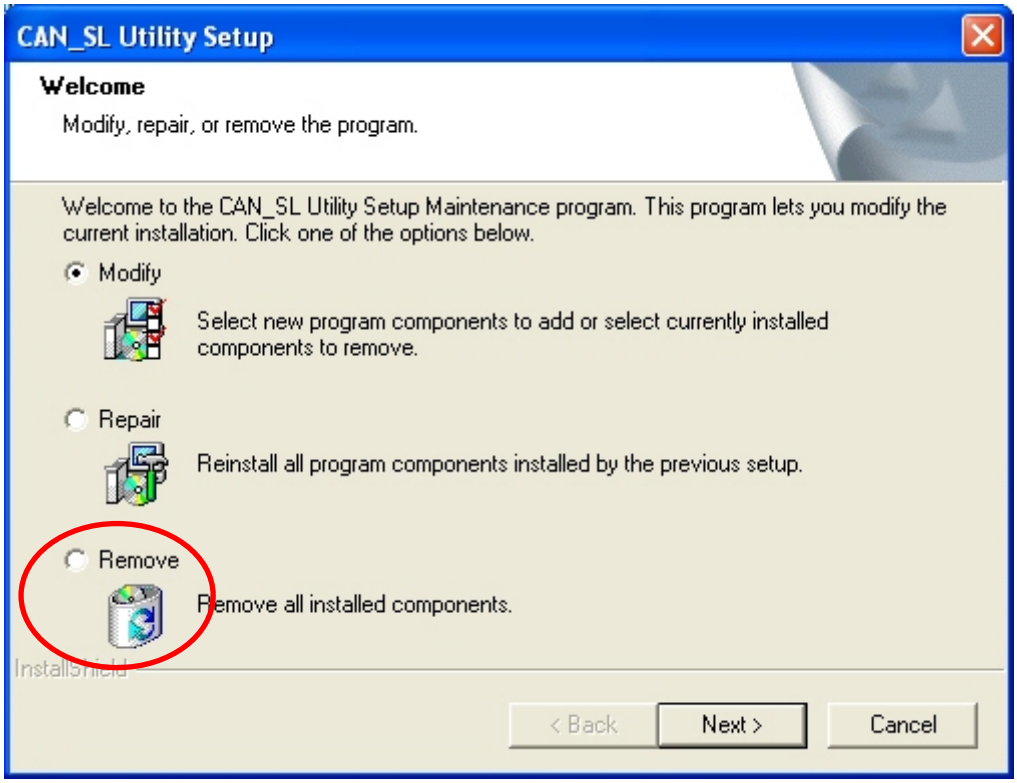
Step 2: Click the “Add/Remove Programs” button icon to open the dialog.



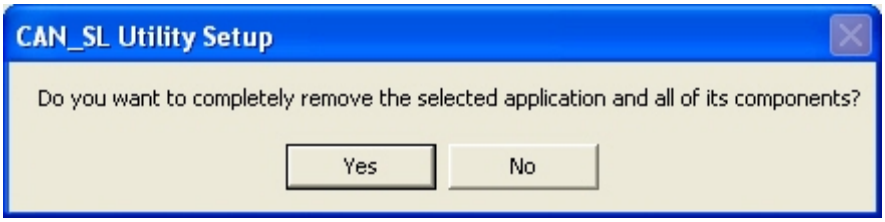
Step 3: Find out the CAN_SL Utility, and click the Change/Remove button.



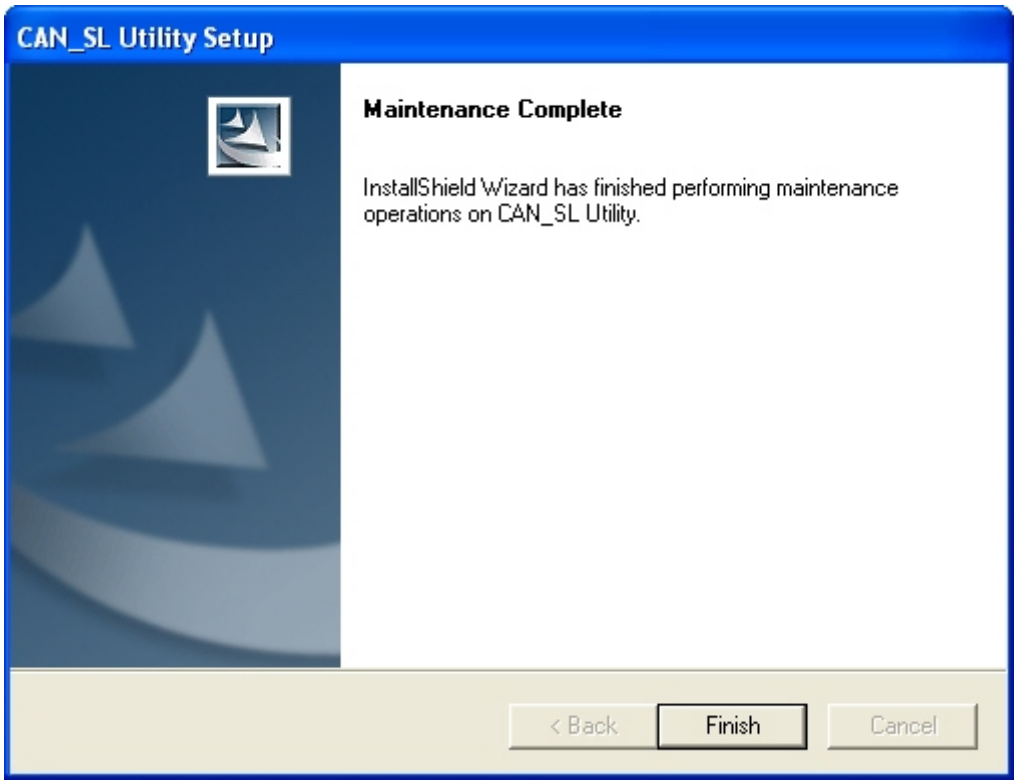
Step 4: Select the “Remove” option and press the “Next” button to remove the software.



Step 5: Click the “Yes” button to remove the software.

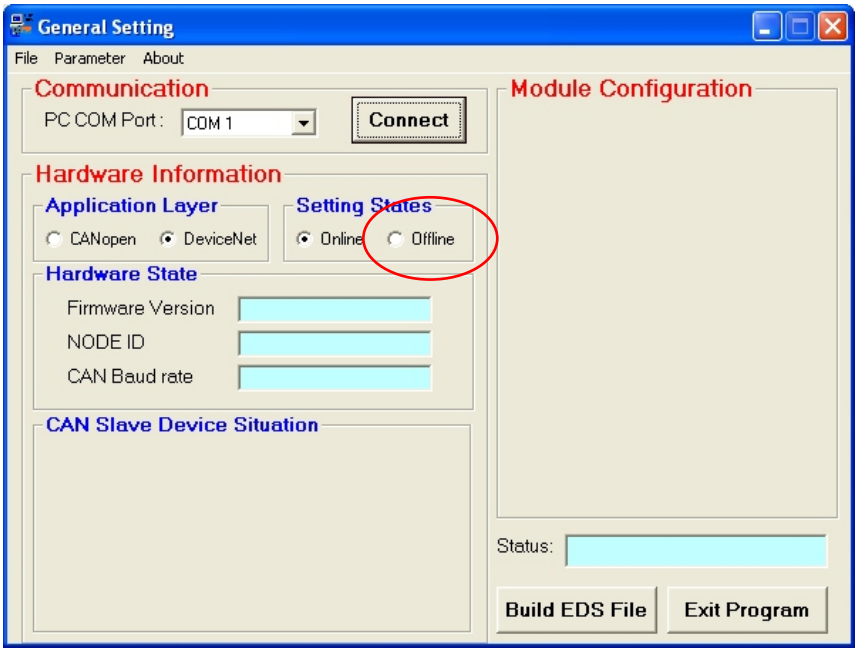


Step 6: Finally, click the “Finish” button to finish the uninstall process.

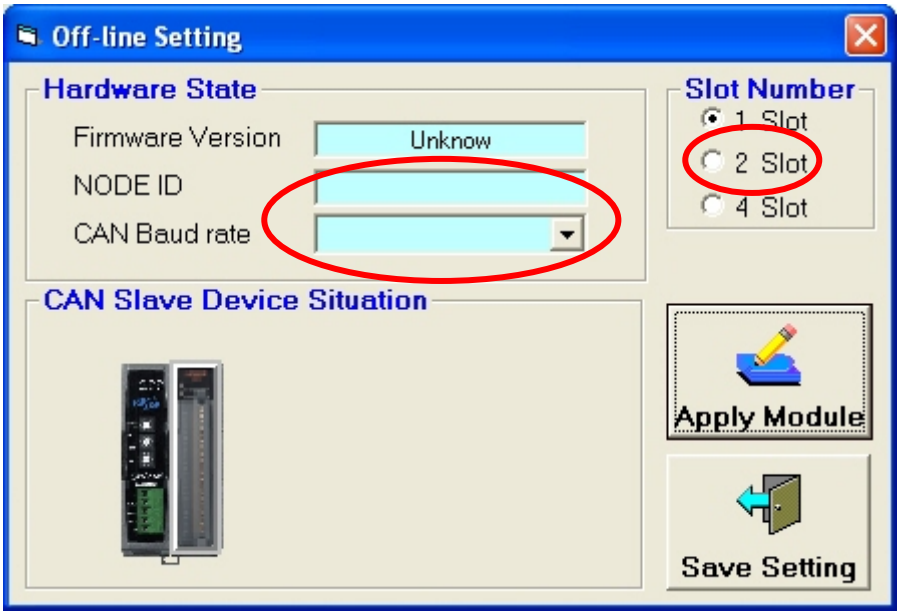


5.5 CAN-8124/CAN-8224 Configuration (Off-line mode)

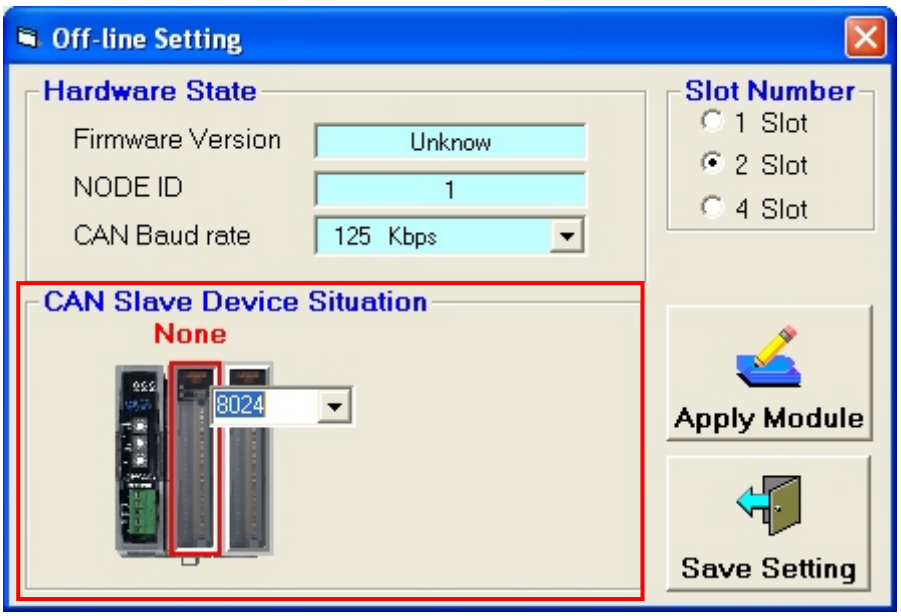
Step 1: Select “DeviceNet” from under the “Application Layer” option. Then, select “offline” from the “Setting status”.



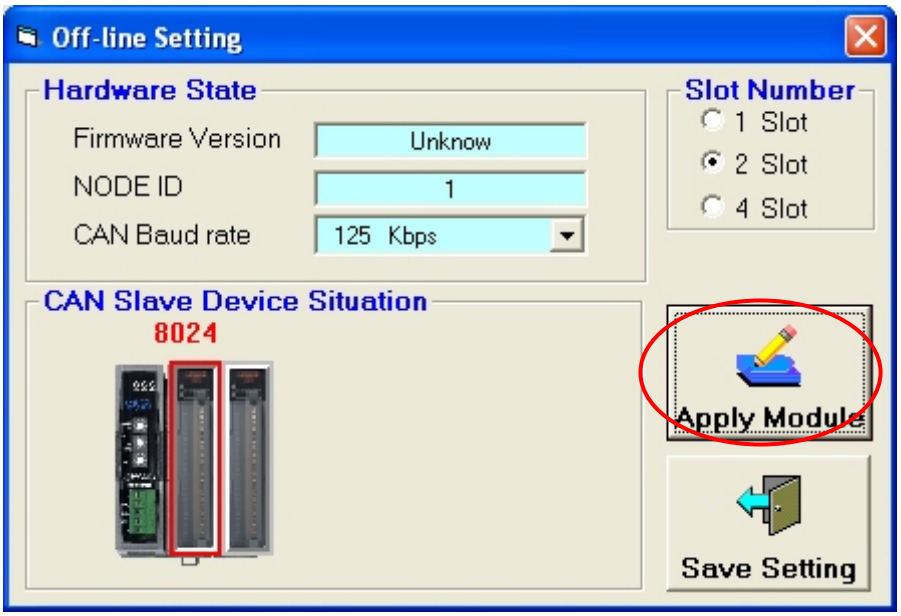
Step 2: If the CAN slave device is CAN-8224 with node id 1 and baud rate 125 Kbps, then users must set the file’s correct value for the CAN-8224 node id and baud rate in the “NODE ID” and “CAN Baud rate”. Then, select the “2 Slot” as the number of slots under “Slot Number”.



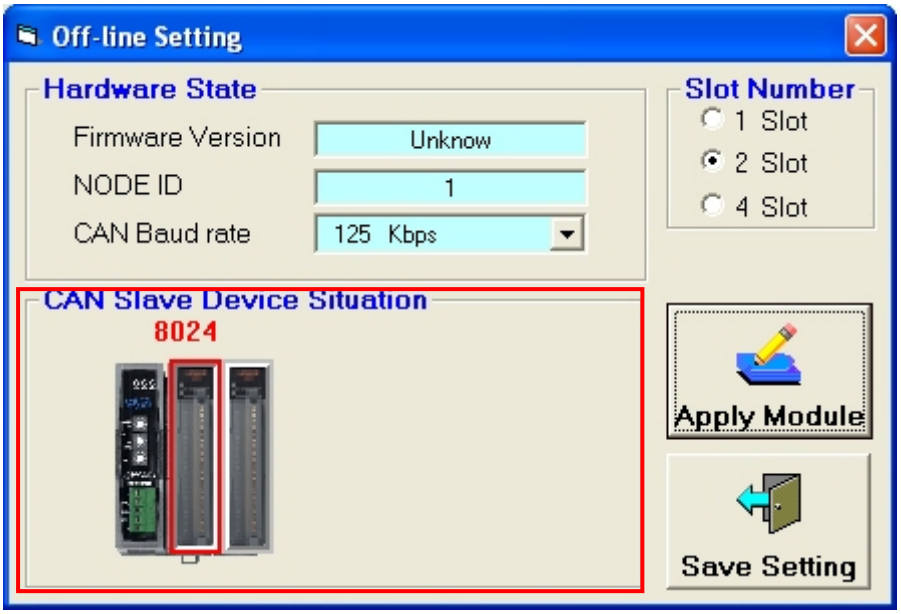
Step 3: Click one of the slot module icons as shown in the “CAN Slave Device Situation” frame below. A list box will pop up. Select the correct slot module that is plugged into the CAN-8224.



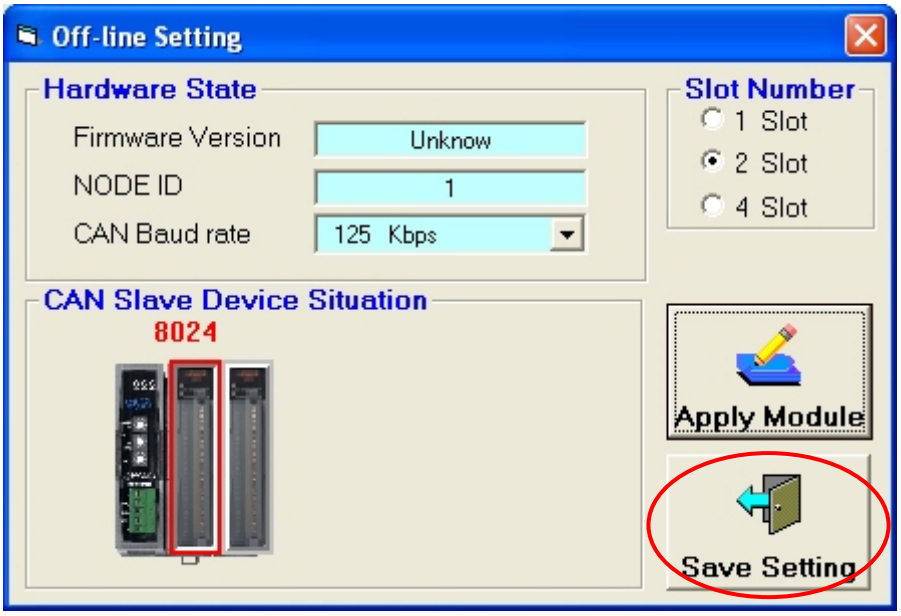
Step 4: If the i-8024 and i-8042 modules are plugged into slot 0 and slot 1 respectively, then, select 8024 from the list box, and click the “Apply Module” button to save this configuration.



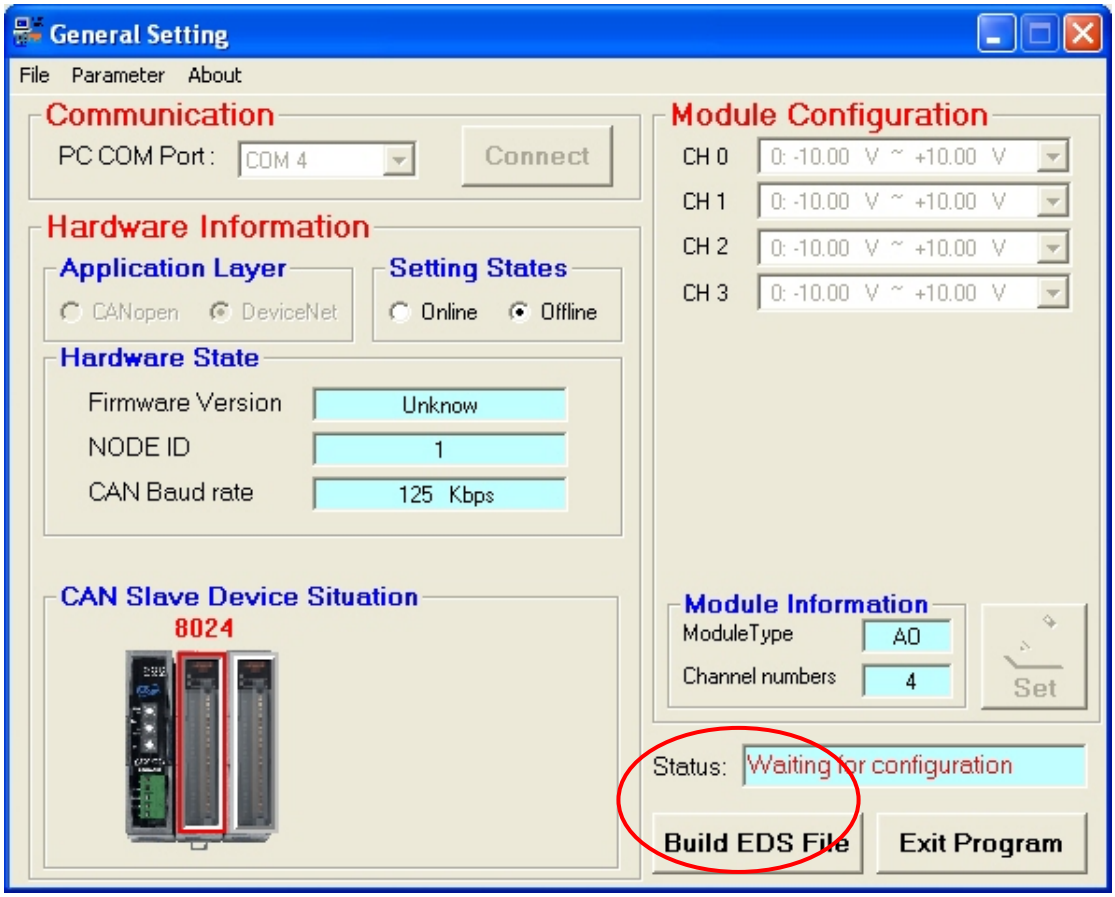
Step 5: Once completing this configuration, users can move the mouse point to the slot module in the “CAN Slave Device Situation” frame. If the configuration has been successful, users will see the correct module name on the top of the slot module.



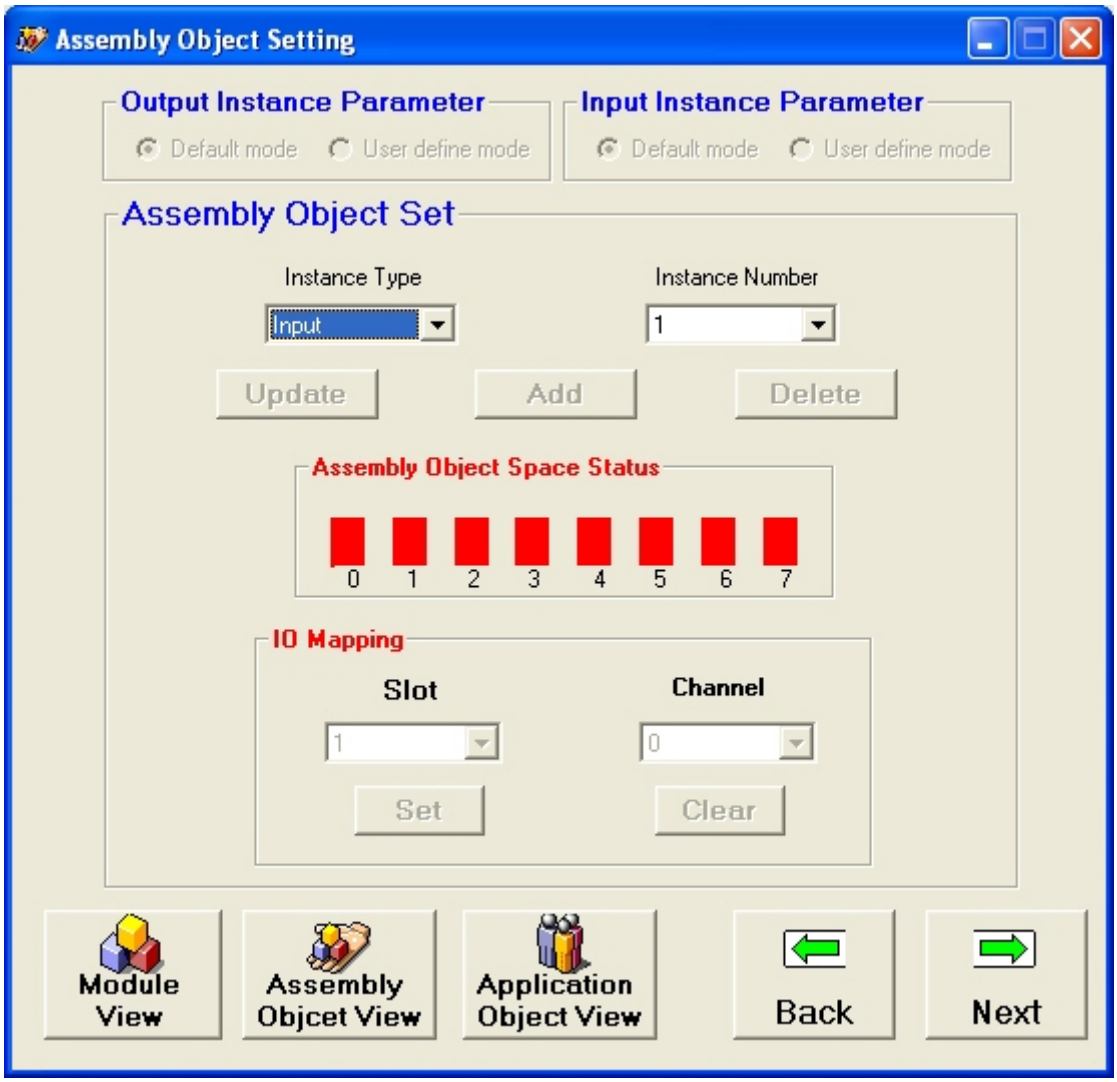
Step 6: Repeat steps 4~6 to configure slot 2 with the i-8042 module. Then, click the “Save Setting” button to finish the off-line parameter settings.



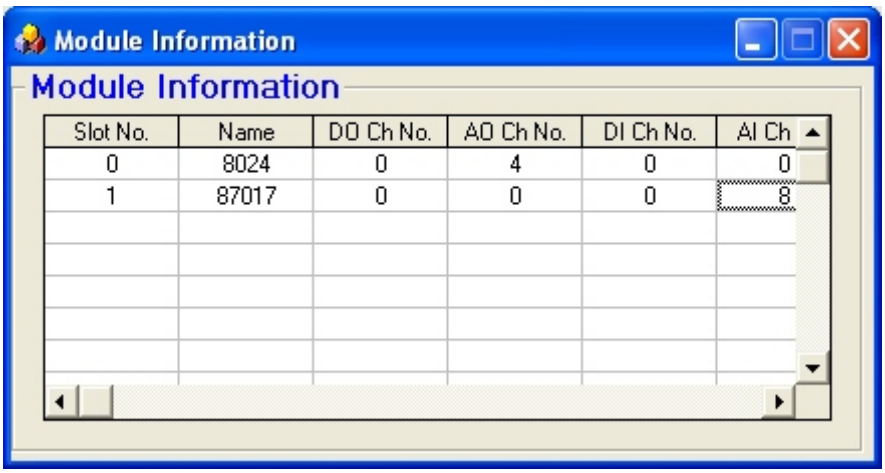
Step 7: Users can perform their parameter settings in the “General Setting” window. Also, users can check the default settings for each slot module by clicking on the module’s icon. Alternatively users can move the mouse pointer over the slot module to show the modules name and information displayed in the “Module Information” frame. Then, click the “Build EDS file” button to go to the next step.



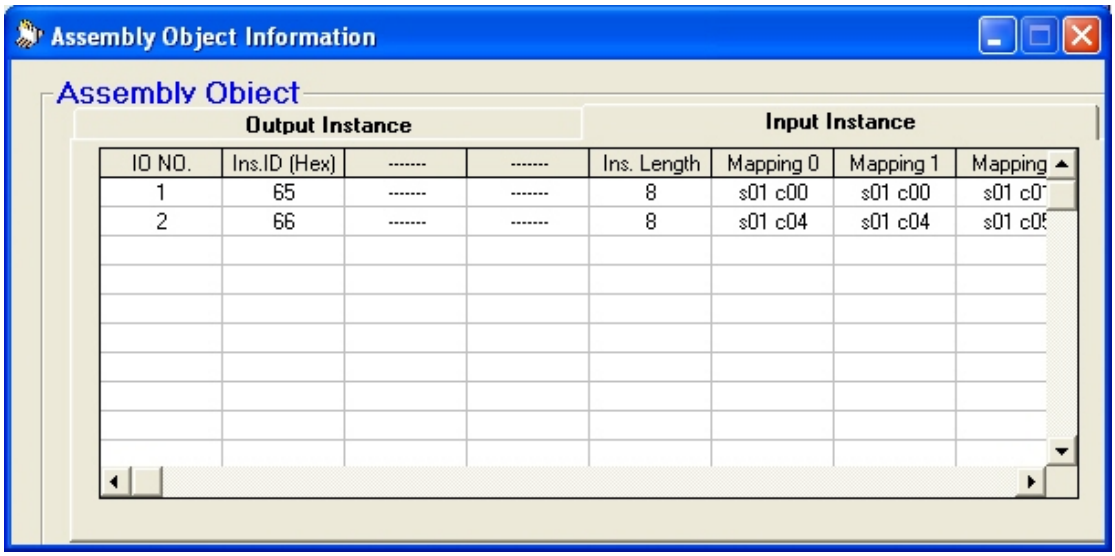
Step 8: In this step users can begin building the specific EDS files for their CAN-8124/CAN-8224. In this dialog, users can know the information of IO modules, assembly components and application instances. Due to the off-line mode, the output and input instance parameters are not able to be set.



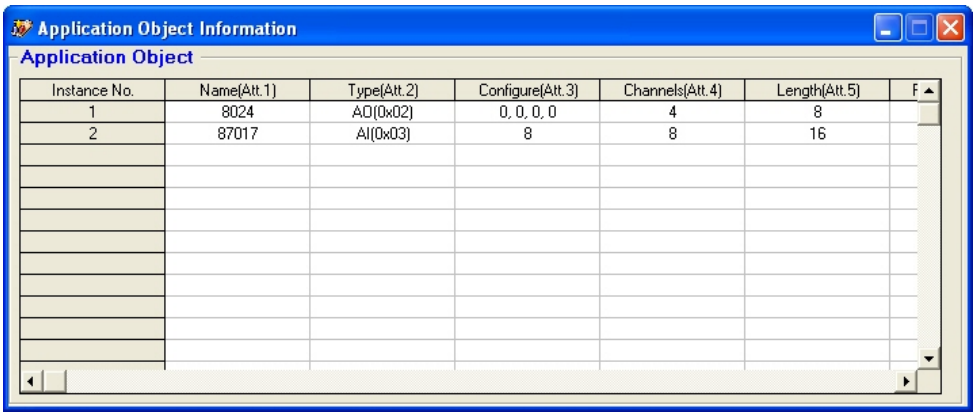
By clicking the “Module View” button, the following figure will be displayed.



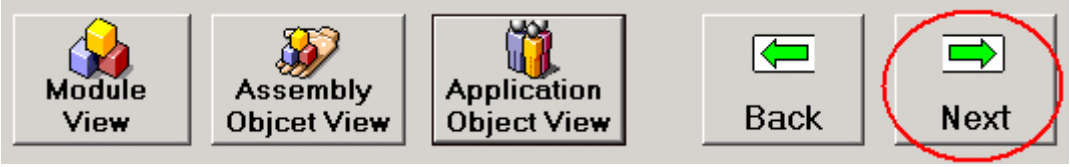
Clicking on the “Assembly Object View” button, the following figure will be displayed.



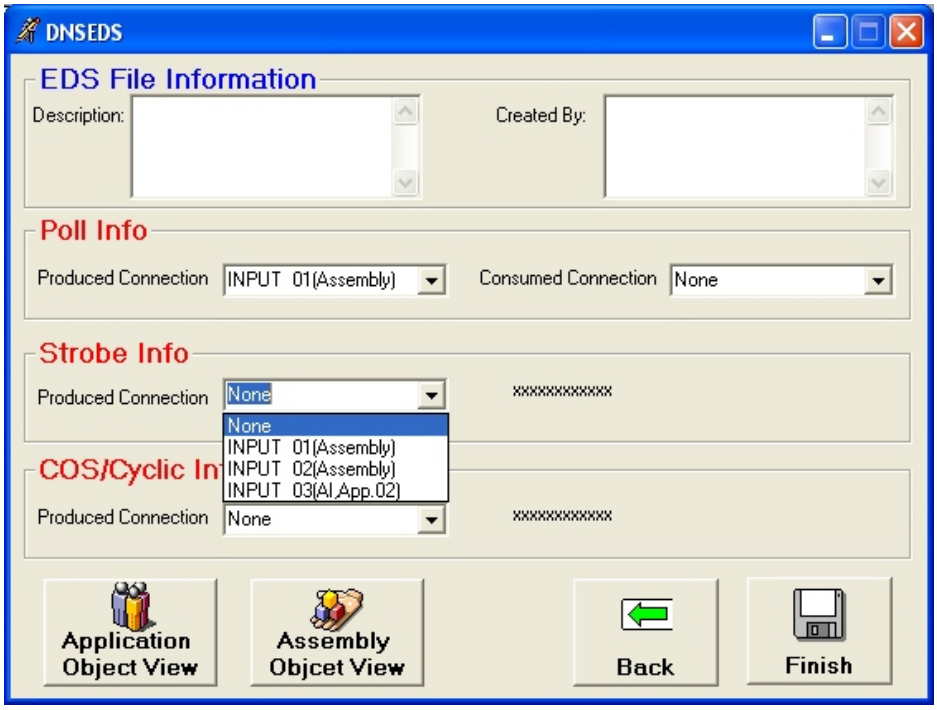
Clicking the “Application Object View” button and the following figure will pop up.



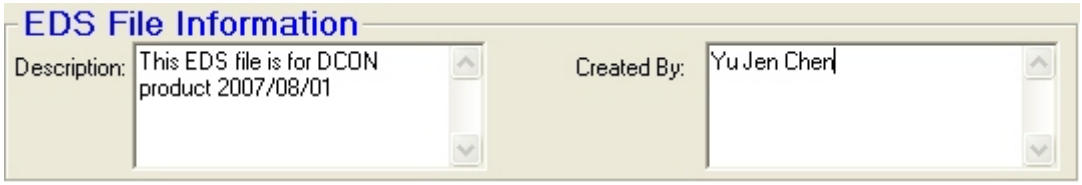
Step 9: Press the “NEXT” button to go to the next step. Otherwise you can press the “Back” button to go back to the above step.



Step 10: The next step shows the settable connection path. The connection path in the CAN-8124/CAN-8224 device is default path. Users can refer to the settable path to set these IO connection paths. The application and assembly object instance can also be shown in this figure.



Step 11: Setting the EDS file's information and giving it a description in the description box provided as can be seen in the following figure.



Step 12: Click the "Finish" button to complete the CAN-8124/CAN-8224 configuration and the system will create the EDS file for users as in the following figure.

Step 13: When this process is finished, the main window will pop-up. Then select "Exit program" to exit the program.

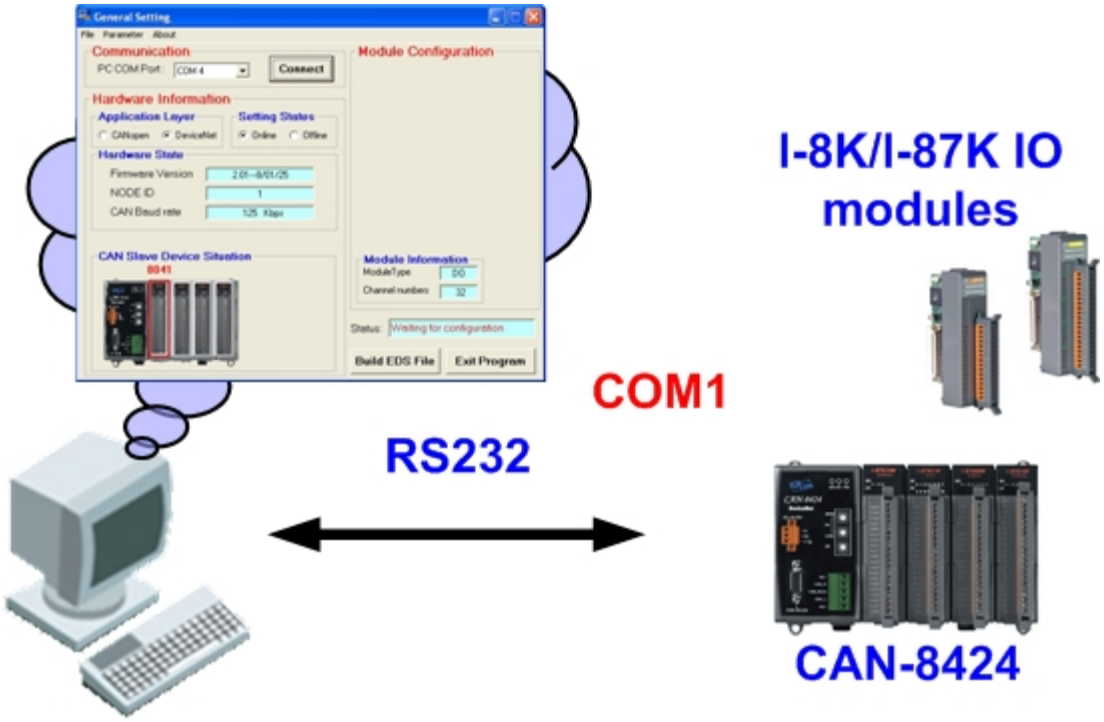
You can find the EDS file for the specific CAN-8124/CAN-8224. The file name is ICPDNS1.eds. "1" represents the Node ID of the device. Therefore, users can apply the EDS file in the DeviceNet application as the following figure.

```
$ ICPDAS-DNS Electronic Data Sheet
$ Version 1.0
$ File Description Section :
$
$ Created by :
$=============================================================================================
$                 Controller Information: 2 slot  DeviceNet Slave Production
$
$                   slot     ModuleName      Module Channel     Module Type
$                  ------------------------------------------------------------
$                     0          8024           04 channel         Type=AO
$                     1          87017          08 channel         Type=AI
$
$==================== Application Object Information (Class ID: 0x64) ======================
$ Instance No. Att.1    2       3     4     5    6    7    8    9   10   11   12   13   14
$  Inst. Num   Name    Type    Conf  Chan  Len  Res  DOL  AOL  DIL  AIL  DON  AON  DIN  AIN
$---------------------------------------------------------------------------------------------
$     01       8024   AO(0x02) ****   04   08   ***  00   08   00   00   00   04   00   00
$     02       87017  AI(0x03) ****   08   16   ***  00   00   00   16   00   00   00   08
$
$=================================== Assembly Object Information (Class ID: 0x04) ================================
$ Ins ID(H) Ins.Len    Mapping 0       1           2           3           4           5           6           7
$-------------------------------------------------------------------------------------------------------------------
$ Out Ins:
$   64         8        s00 c00     s00 c00     s00 c01     s00 c01     s00 c02     s00 c02     s00 c03     s00 c03
$-------------------------------------------------------------------------------------------------------------------
$ In  Ins:
$   65         8        s01 c00     s01 c00     s01 c01     s01 c01     s01 c02     s01 c02     s01 c03     s01 c03
$   66         8        s01 c04     s01 c04     s01 c05     s01 c05     s01 c06     s01 c06     s01 c07     s01 c07
$
$===================================================================================================================
[File]
```

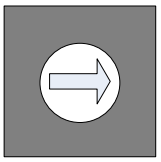
Note: There is also some device information in the EDS file. Users can also see the information form the EDS file.

5.6 CAN-8424 Configuration (On-line mode)

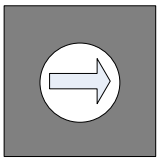
Before using the CAN Slave utility in the On-line mode with the CAN-8424, please make sure that you have connected the COM1 port of the CAN-8424 with the available COM port on your PC. The architecture is displayed in the following figure. In this demo, the CAN-8424 will be used, and slot modules, i-8041, i-8040, i-8024 and i-8017H are plugged into slots 0, 1, 2 and 3 respectively.



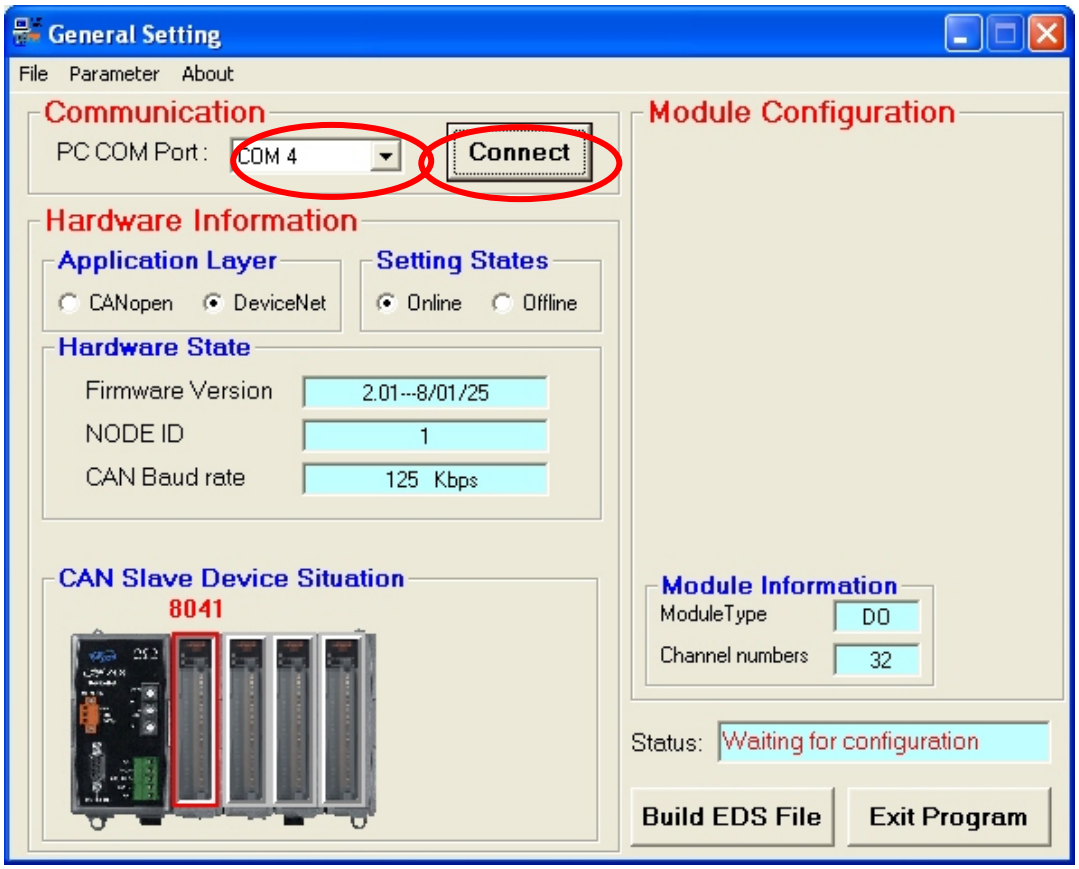
Step 1: Turn off the CAN-8424. Set the “DR” rotary switch for the CAN-8424 to 9. Then turn on the CAN-8424.



Step 2: Use the “DR” rotary switch again to set the baud rate for the CAN-8424. Here, we use the baud rate at 125 Kbps for the demo. Therefore, set the “DR” rotary switch to 0.

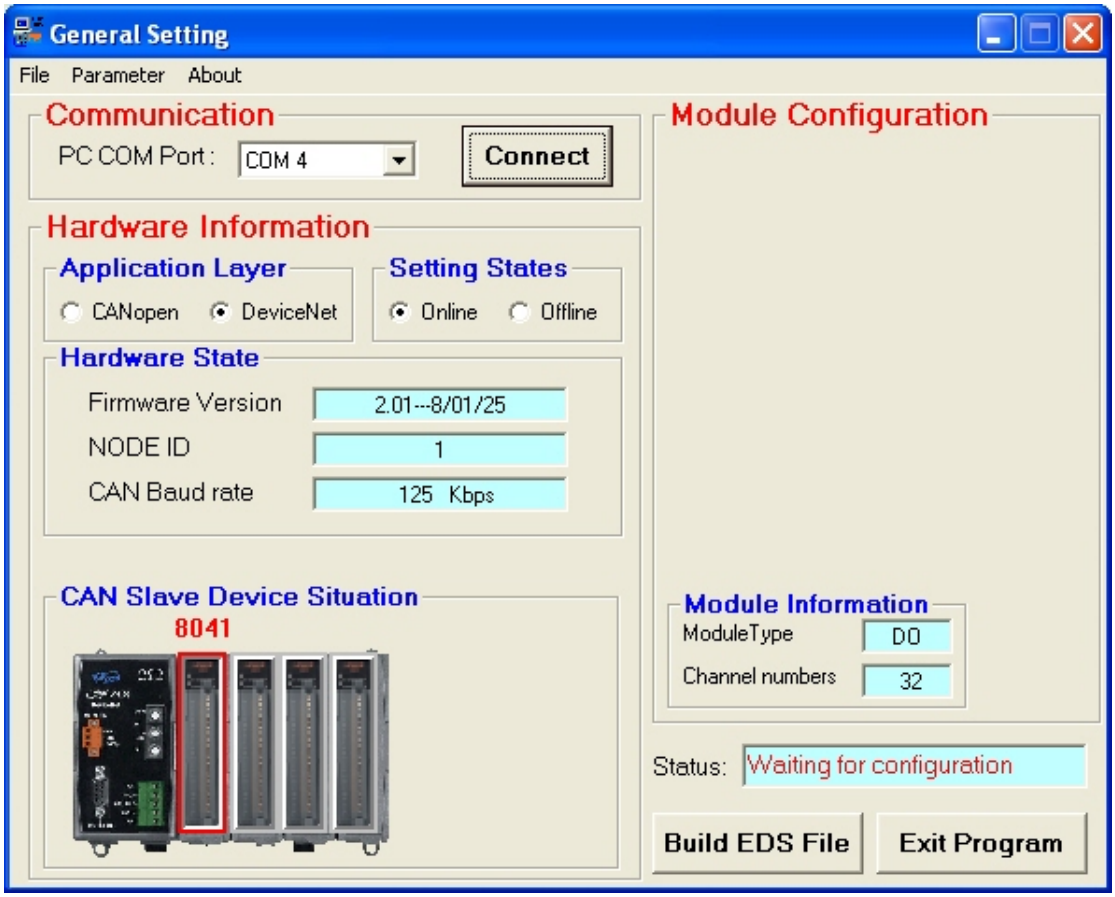


Step 3: Execute the CAN_SL.exe file, and the figure will be displayed. Select a PC COM port to connect the CAN-8424. We used the PC COM 6 port for this demo. Click the “Connect” button to store the information into the CAN-8424. Then press the “Connect” button to connect to the CAN-8424. The utility will scan the IO modules plugged into the CAN-8424 automatically.



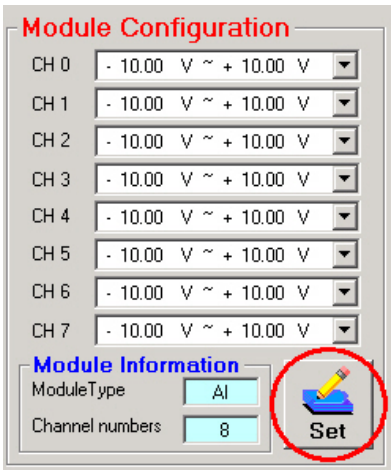
DR

Step 4: After scanning the IO modules, there will be some information about the firmware version, MAC ID and baud rate as shown in the below windows.

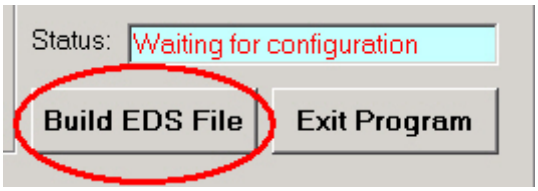


Step 5: In the “CAN Slave Device situation” section, press the specific modules and their module configuration information will be revealed in the right upper window. Furthermore users can set their specific module configuration.

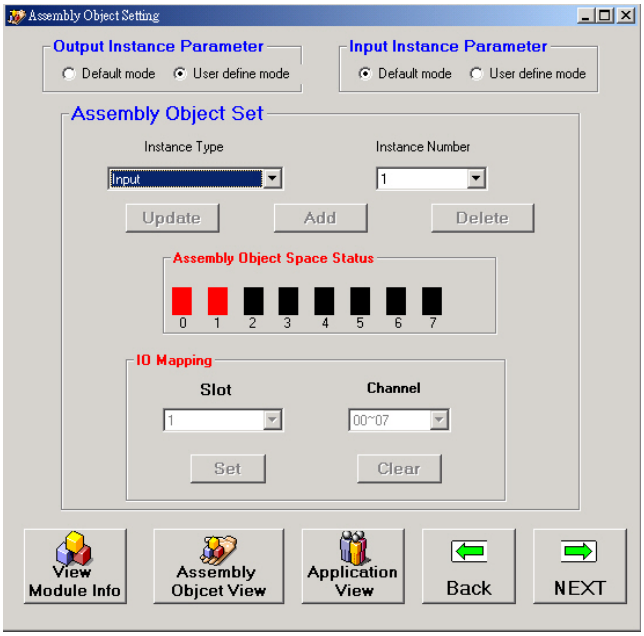
Step 6: After selecting the needed configuration, press the “Set” button to set the module’s configuration as in this specific one.



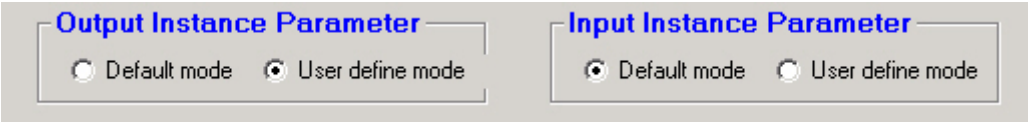
Step 7: After finishing the process of configuration, press the “Build EDS file” to button to go to the next step to start to build the specific EDS file for your CAN-8424.



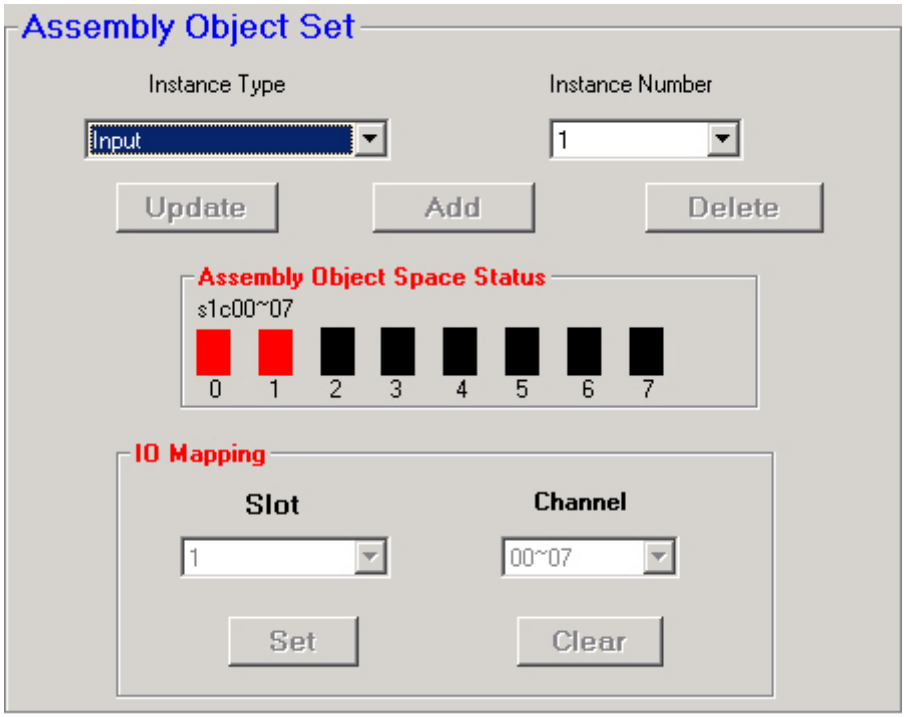
Step 8: The dialog will be displayed as follows. In this dialog, users can define their needed assembly objects.



Step 9: Set the output instance and input instance to either default or user defined mode.

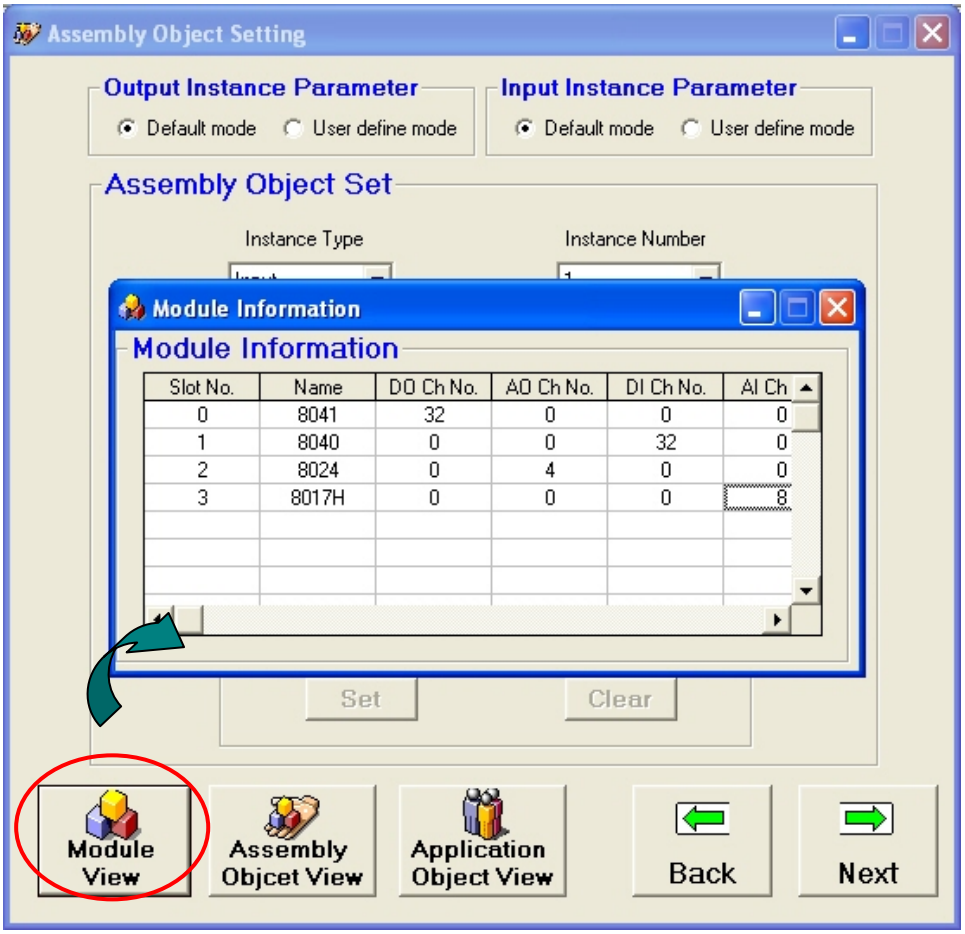


The dialog will be shown by the assembly mapping information. If users set it to the “user-defined mode”, users can add, delete or update instance mapping in the dialog. In addition, the usable IO information will be shown in the IO Mapping frame. Users can set/clear the specific channel IO of the slot to the assembly instance. In the default mode, the system just shows the default Assembly instance information. Users cannot modify the assembly instance components.

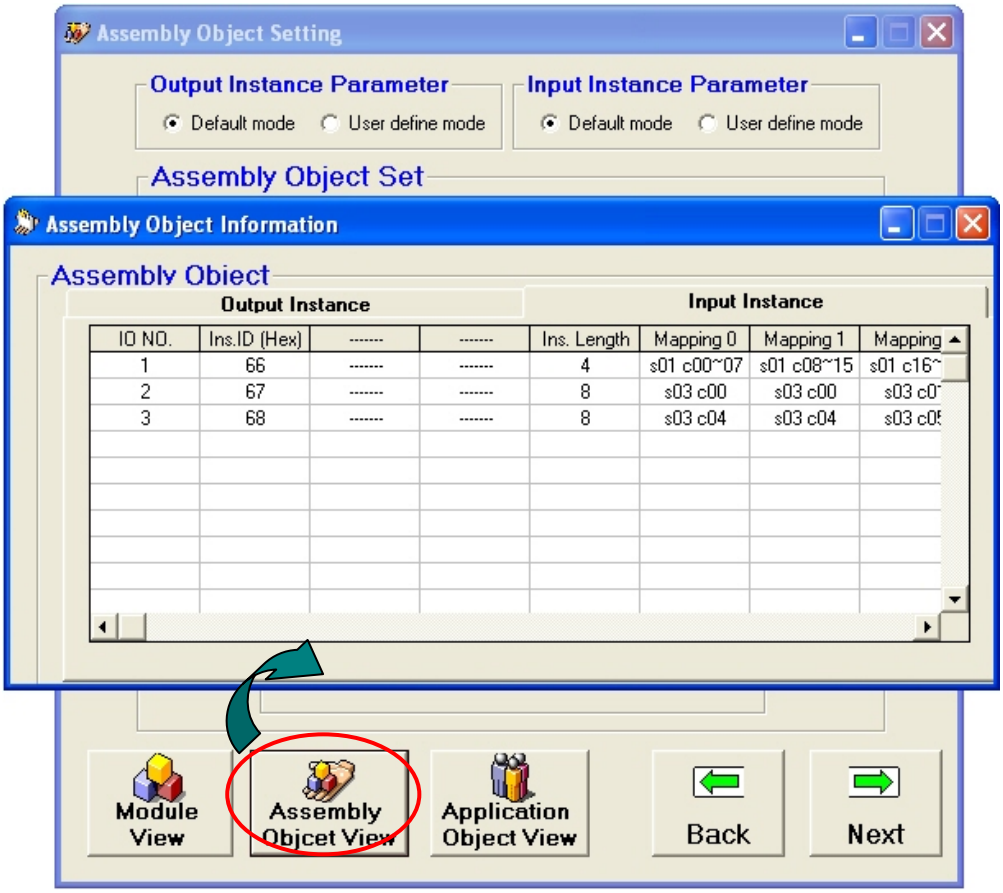


Step 10: In addition, the utility also provides module, assembly and application object information by pressing the “Module view”, “Assembly Object View” and “Application View” buttons.

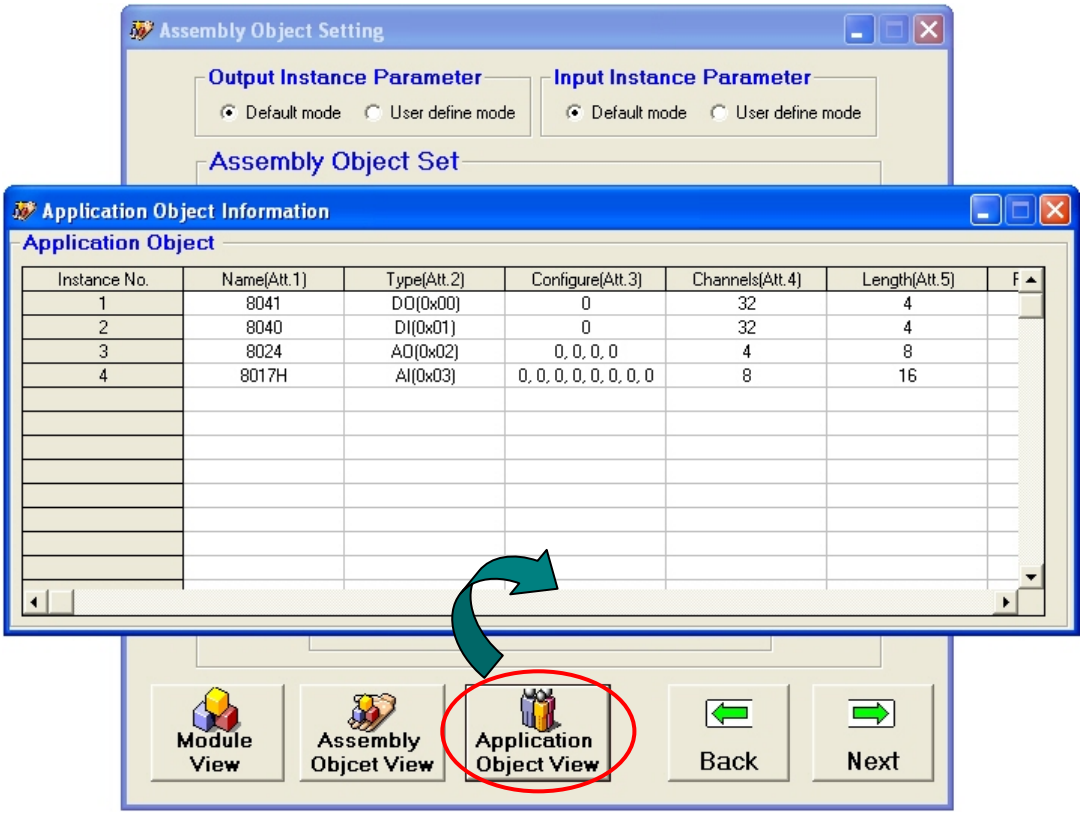
Step 11: If you press the “Module View” button, the “Module Information” window will pop-up. Users can get the IO module plugged into the CAN-8424 from this dialog.



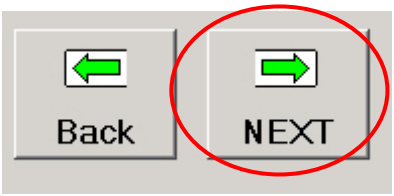
Step 12: By pressing the “Assembly Object View” button, the “Assembly Object Information” window will pop-up. Users can get every assembly object information in the CAN-8424 from this dialog.



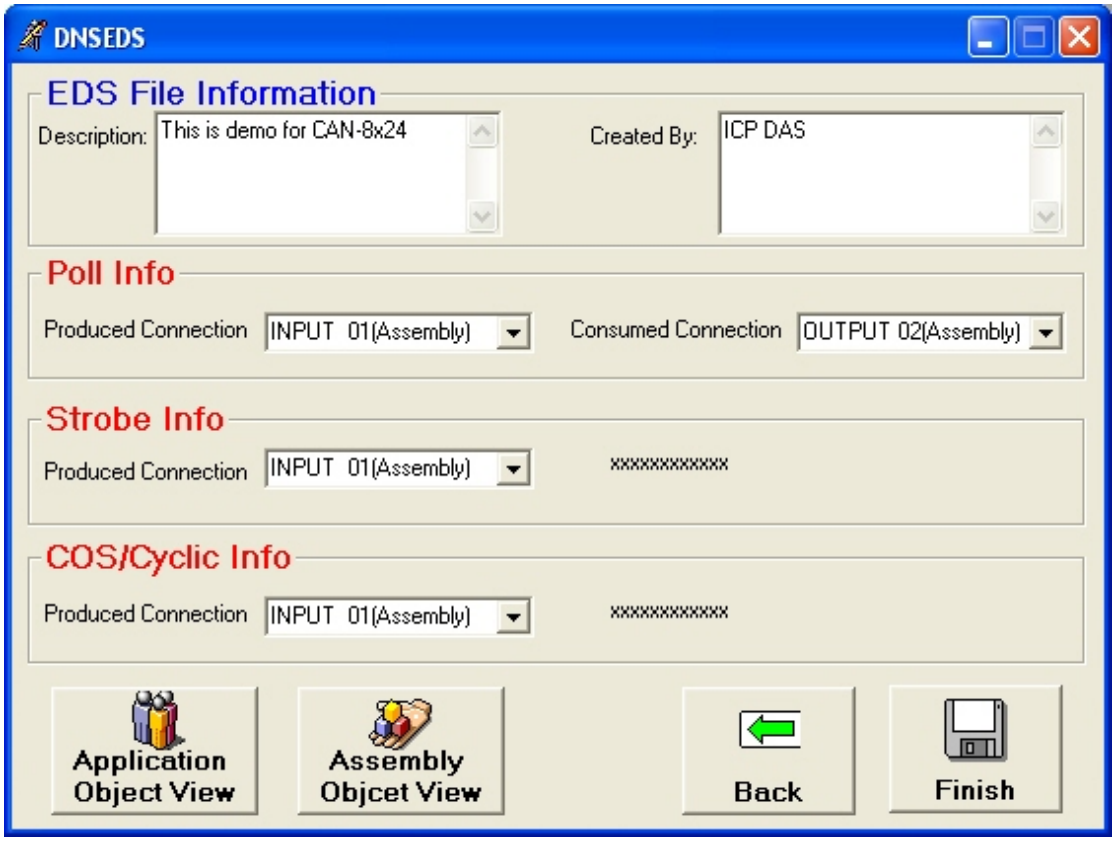
Step 13: When pressing the “Application Object View” button, the “Application Object Information” window will pop-up. Users can get every application object’s information in the CAN-8424 from this dialog.



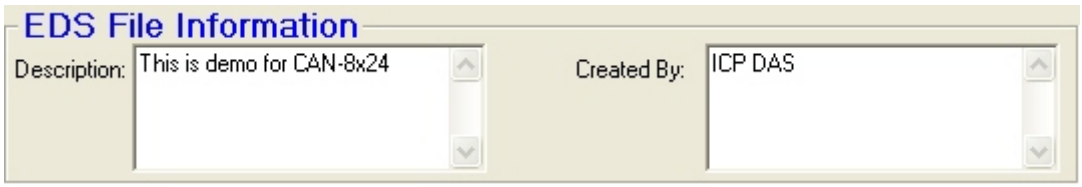
Step 14: After creating the assembly object, press the “NEXT” button to go to the next step. Alternatively you can press the “Back” button to go back to the above step.



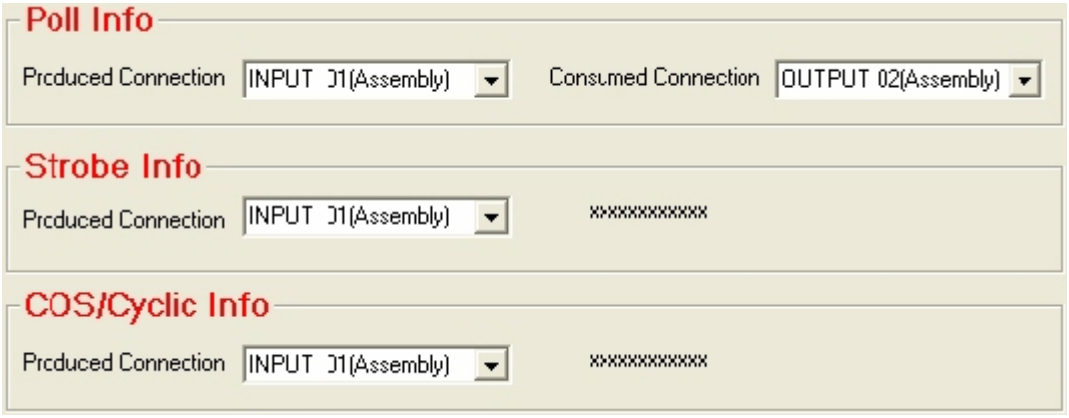
Step 15: Upon completion of the above steps, the following window will pop-up. Users can set their EDS files information from here.



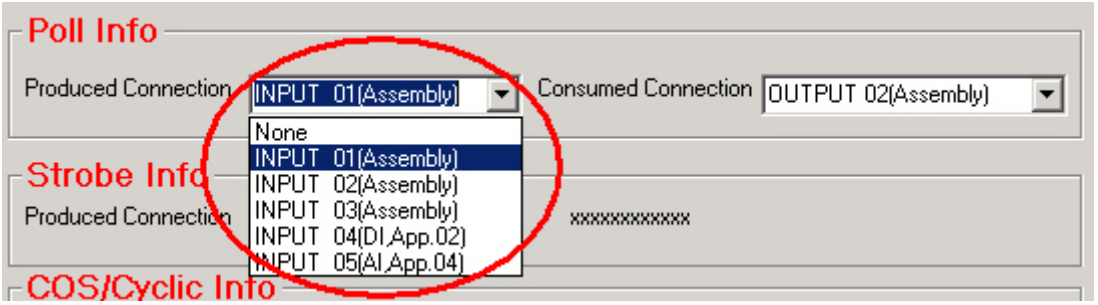
Step 16: Users can set the specified information they want in the below fields. This information will be stored in the EDS file.



Step 17: Set the Polling/Bit Strobe/COS/Cyclic IO connection path for the CAN-8424.



The utility will list all useable paths in these fields. These selections include assembly and application objects.



Step 18: Press the “Finish” button to complete the CAN-8424 configuration and the system will create the EDS file for the CAN-8424. Otherwise press the “Back” button to go back to the above step. Users can then find the EDS file in the execute file path.

Chapter 6 Components of Assembly Objects

6.1 Components in the Assembly object

The Assembly Object binds attributes of multiple objects, which allows data to or from each object to be sent or received over a single connection. The CAN-8x24 provides many assembly objects for users. The number of assembly objects is decided by the user-defined or default settings. However, the number of assembly objects is only a default setting in the CAN-8124/CAN-8224. Every IO slot modules represents an application object instance. The CAN-8x24 device would arrange the application instances in order by slot address. The assembly object instances consist of these application object attributes. Moreover the default assembly instances are the groups relating to DO/AO/DI/AI signals. However, in the CAN-8x24, users can define the specific assembly object instances they need by using the CAN Slave Utility. The sketch map is as figure 6-1. The CAN-8x24 supports a maximum of 16 assembly instances.

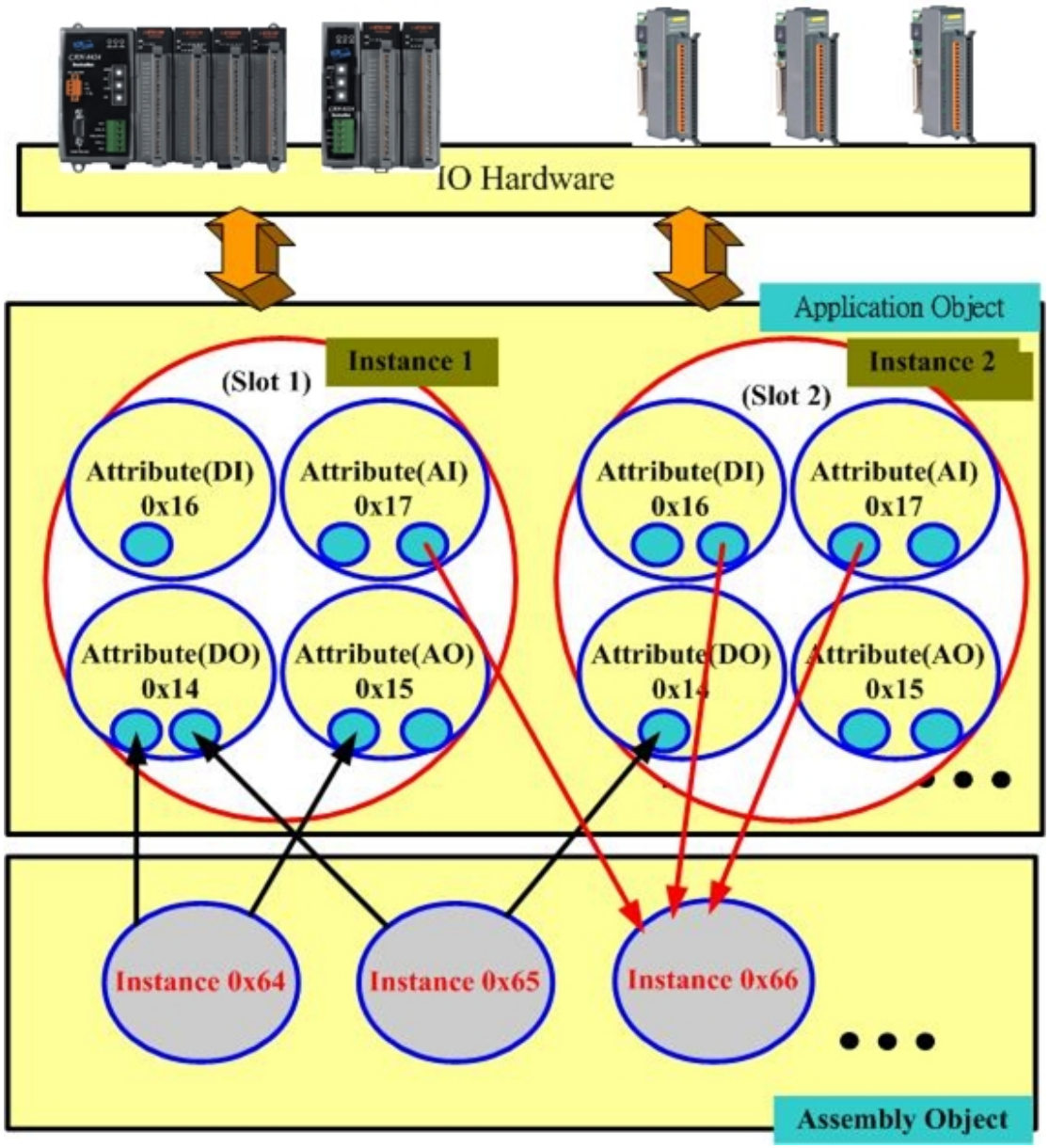


Figure 6-1 DeviceNet object sketch map

Note: The max number in the assembly instance is 16.

6.2 CAN-8424 Assembly Example

There are many IO examples related to the CAN-8424 in this section. These demos should help users to understand the CAN-8424 to a suitable degree.

Example1:

In this demo, apply the i-87017 (slot 0), i-8024 (slot 1), i-8057 (slot 2) and i-8053 (slot 3) to the CAN-8424 as follows.

The CAN-8424 will arrange the application objects for the DeviceNet according to the following table.

Slot Address	Application Instance ID	Module name	DO Length(Byte)	AO Length(Byte)	DI Length(Byte)	AI Length(Byte)
0	0x01	87017	0	0	0	16
1	0x02	8024	0	8	0	0
2	0x03	8057	2	0	0	0
3	0x04	8053	0	0	2	0

Because there are four slots in the CAN-8424, the application instance number is 4. The corresponding Instance attribute is as follows:

Application Instance 1

Attribute ID	Description	Method	Data Type	Value
0x01	Module name	Get	WORD	87017
0x02	Module Type	Get	CHAR	3 (AI type)
0x03	Configuration	Get/Set	Depend on the number of module channel	1st byte (1st channel) 2nd byte (2nd channel) 3rd byte (3rd channel) 4th byte (4th channel) 5th byte (5th channel) 6th byte (6th channel) 7th byte (7th channel) 8th byte (8th channel)
0x04	Total Channels	Get	CHAR	8
0x05	Total Length	Get	CHAR	16
0x06	Reserved	Get	CHAR	0
0x07	DO Length	Get	CHAR	0
0x08	AO Length	Get	CHAR	0
0x09	DI Length	Get	CHAR	0
0x0A	AI Length	Get	CHAR	16
0x0B	DO channel num	Get	CHAR	0
0x0C	AO channel num	Get	CHAR	0
0x0D	DI channel num	Get	CHAR	0
0x0E	AI channel num	Get	CHAR	8
0x0F	Enable/Disable output safe value	Get/Set	CHAR	0
0x10	Output safe value	Get/Set	Defined by module channel num	0
0x11	DI counter channel	Get/Set	CHAR	0
0x12	Clear DI counter value	Set	CHAR	0
0x13	DI counter value	Get	INTEGER	0
0x14	DO data	Set	Defined by module channel num	0
0x15	AO data	Set	Defined by module channel	0

			num	
0x16	DI data	Get	Defined by module channel num	0
0x17	AI data	Get	Defined by module channel num	1st and 2nd bytes (1st channel) 3rd and 4th bytes (2nd channel) 5th and 6th bytes (3rd channel) 7th and 8th bytes (4th channel) 9th and 10th bytes (5th channel) 11th and 12th bytes (6th channel) 13th and 14th bytes (7th channel) 15th and 16th bytes (8th channel)
0x18	Clear Counter module	Set	CHAR	0
0x19	Counter module's Input Mode	Get/Set	CHAR	0

Application Instance 2

Attribute ID	Description	Method	Data Type	Value
0x01	Module name	Get	WORD	8024
0x02	Module Type	Get	CHAR	2 (AO type)
0x03	Configuration	Get/Set	Depend on the number of module channel	1st byte (1st channel) 2nd byte (2nd channel) 3rd byte (3rd channel) 4th byte (4th channel)
0x04	Total Channels	Get	CHAR	4
0x05	Total Length	Get	CHAR	8
0x06	Reserved	Get	CHAR	0
0x07	DO Length	Get	CHAR	0
0x08	AO Length	Get	CHAR	0
0x09	DI Length	Get	CHAR	0
0x0A	AI Length	Get	CHAR	8
0x0B	DO channel num	Get	CHAR	0
0x0C	AO channel	Get	CHAR	0

	num			
0x0D	DI channel num	Get	CHAR	0
0x0E	AI channel num	Get	CHAR	4
0x0F	Enable/Disable output safe value	Get/Set	CHAR	0
0x10	Output safe value	Get/Set	Defined by module channel num	0
0x11	DI counter channel	Get/Set	CHAR	0
0x12	Clear DI counter value	Set	CHAR	0
0x13	DI counter value	Get	WORD	0
0x14	DO data	Set	Defined by module channel num	0
0x15	AO data	Set	Defined by module channel num	1st and 2nd bytes (1st channel) 3rd and 4th bytes (2nd channel) 5th and 6th bytes (3rd channel) 7th and 8th bytes (4th channel)
0x16	DI data	Get	Defined by module channel num	0
0x17	AI data	Get	Defined by module channel num	0
0x18	Clear Counter module	Set	CHAR	0
0x19	Counter module's Input Mode	Get/Set	CHAR	0

Application Instance 3

Attribute ID	Description	Method	Data Type	Value
0x01	Module name	Get	WORD	8057
0x02	Module Type	Get	CHAR	0 (DO type)
0x03	Configuration	Get/Set	Depend on the number of module channel	0x40
0x04	Total Channels	Get	CHAR	16
0x05	Total Length	Get	CHAR	2
0x06	Reserved	Get	CHAR	0
0x07	DO Length	Get	CHAR	2
0x08	AO Length	Get	CHAR	0
0x09	DI Length	Get	CHAR	0
0x0A	AI Length	Get	CHAR	0
0x0B	DO channel num	Get	CHAR	16
0x0C	AO channel num	Get	CHAR	0
0x0D	DI channel num	Get	CHAR	0
0x0E	AI channel num	Get	CHAR	0
0x0F	Enable/Disable output safe value	Get/Set	CHAR	0
0x10	Output safe value	Get/Set	Defined by module channel num	0
0x11	DI counter channel	Get/Set	CHAR	0
0x12	Clear DI counter value	Set	CHAR	0
0x13	DI counter value	Get	WORD	0
0x14	DO data	Set	Defined by module channel num	1st bit of 1st byte (1st channel) 2nd bit of 1st byte (2nd channel) 3rd bit of 1st byte (3rd channel)

				4th bit of 1st byte (4th channel) 5th bit of 1st byte (5th channel) 6th bit of 1st byte (6th channel) 7th bit of 1st byte (7th channel) 8th bit of 1st byte (8th channel) 1st bit of 2nd byte (9th channel) 2nd bit of 2nd byte (10th channel) 3rd bit of 2nd byte (11th channel) 4th bit of 2nd byte (12th channel) 5th bit of 2nd byte (13th channel) 6th bit of 2nd byte (14th channel) 7th bit of 2nd byte (15th channel) 8th bit of 2nd byte (16th channel)
0x15	AO data	Set	Defined by module channel num	0
0x16	DI data	Get	Defined by module channel num	0
0x17	AI data	Get	Defined by module channel num	0
0x18	Clear Counter module	Set	CHAR	0
0x19	Counter module's Input Mode	Get/Set	CHAR	0

Application Instance 4

Attribute ID	Description	Method	Data Type	Value
0x01	Module name	Get	WORD	8053
0x02	Module Type	Get	CHAR	2 (DI type)
0x03	Configuration	Get/Set	Depend on the number of module channel	0x40
0x04	Total Channels	Get	CHAR	16
0x05	Total Length	Get	CHAR	2
0x06	Reserved	Get	CHAR	0
0x07	DO Length	Get	CHAR	0
0x08	AO Length	Get	CHAR	0
0x09	DI Length	Get	CHAR	2
0x0A	AI Length	Get	CHAR	0
0x0B	DO channel num	Get	CHAR	0
0x0C	AO channel num	Get	CHAR	0
0x0D	DI channel num	Get	CHAR	16
0x0E	AI channel num	Get	CHAR	0
0x0F	Enable/Disable output safe value	Get/Set	CHAR	0
0x10	Output safe value	Get/Set	Defined by module channel num	0
0x11	DI counter channel	Get/Set	CHAR	0
0x12	Clear DI counter value	Set	CHAR	0
0x13	DI counter value	Get	WORD	0
0x14	DO data	Set	Defined by module channel num	0
0x15	AO data	Set	Defined by module channel num	0
0x16	DI data	Get	Defined by	1st bit of 1st byte (1st

			module channel num	channel) 2nd bit of 1st byte (2nd channel) 3rd bit of 1st byte (3rd channel) 4th bit of 1st byte (4th channel) 5th bit of 1st byte (5th channel) 6th bit of 1st byte (6th channel) 7th bit of 1st byte (7th channel) 8th bit of 1st byte (8th channel) 1st bit of 2nd byte (9th channel) 2nd bit of 2nd byte (10th channel) 3rd bit of 2nd byte (11th channel) 4th bit of 2nd byte (12th channel) 5th bit of 2nd byte (13th channel) 6th bit of 2nd byte (14th channel) 7th bit of 2nd byte (15th channel) 8th bit of 2nd byte (16th channel)
0x17	AI data	Get	Defined by module channel num	0
0x18	Clear Counter module	Set	CHAR	0
0x19	Counter module's Input Mode	Get/Set	CHAR	0

Refer to the application object instances. The CAN-8424 will define the default assembly object instances according to the following table.

Assembly Object Instance ID(Hex)	Data Length(Byte)	Component modules
0x64	DO: 2	i-8053 (ch0~ch15)
0x65	AO: 8	i-8024 (ch0~ch3)
0x66	DI: 2	i-8057 (ch0~ch15)
0x67	AI: 8	i-87017 (ch0~ch3)
0x68	AI: 8	i-87017 (ch4~ch7)

If the default assembly instances are not adaptive, the user-defined assembly object can be created by applying the CAN Slave Utility. The system will show the unit of IO modules in the utility software. Therefore, users can arrange the unit to the specific assembly instance. In analog modules, the unit of length is 2 bytes and 1 byte is in the digital module.

Slot Address	Application Instance ID	Module name	Useable DO unit	Useable AO unit	Useable DI unit	Useable AI unit
0	0x01	87017	0	0	0	8
1	0x02	8024	0	4	0	0
2	0x03	8057	2	0	0	0
3	0x04	8053	0	0	2	0

Users can assign assembly instances as either input or output instances. However there must be DO or AO units in output instances. Plus there must be DI or AI units in input instances. For example:

Assembly Object Instance ID (Hex)	Instance Type	Data Length(Byte)	Component modules
0x64	Output	8	1st byte DO (1st byte of i-8053) 2nd byte DO (2nd byte of i-8053) 3rd byte AO (1st CH of i-8024) 4th byte AO (1st CH of i-8024) 5th byte AO (2nd CH of i-8024) 6th byte AO (2nd CH of i-8024) 7th byte AO (3rd CH of i-8024) 8th byte AO (3rd CH of i-8024)
0x65	Output	2	1st byte AO (4th CH of i-8024) 2nd byte AO (4th CH of i-8024)
0x66	Input	8	1st byte DI (1st byte of i-8057) 2nd byte DI (2nd byte of i-8057) 3rd byte AI (1st CH of i-87017) 4th byte AI (1st CH of i-87017) 5th byte AI (2nd CH of i-87017) 6th byte AI (2nd CH of i-87017) 7th byte AI (3rd CH of i-87017) 8th byte AI (3rd CH of i-87017)
0x67	Input	8	1st byte AI (4th CH of i-87017) 2nd byte AI (4th CH of i-87017) 3rd byte AI (5th CH of i-87017) 4th byte AI (5th CH of i-87017) 5th byte AI (6th CH of i-87017) 6th byte AI (6th CH of i-87017) 7th byte AI (7th CH of i-87017) 8th byte AI (7th CH of i-87017)
0x68	Input	2	1st byte AI (8th CH of i-87017) 2nd byte AI (8th CH of i-87017)

Example2:

If users plug i-8057 (slot 0), i-8064 (slot 1), i-8042 (slot 2) and i-8053 (slot 3) into the CAN-8424, the CAN-8424 will arrange the application objects for the DeviceNet according to the following table.

Slot Address	Application Instance ID	Module name	DO Length(Byte)	AO Length(Byte)	DI Length(Byte)	AI Length(Byte)
0	0x01	8057	2	0	0	0
1	0x02	8064	1	0	0	0
2	0x03	8042	2	0	2	0
3	0x04	8053	0	0	2	0

Since there are four slots in CAN-8424, the application instance number is 4. The corresponding instance attributes are as follows:

Application Instance 1

Attribute ID	Description	Method	Data Type	Value
0x01	Module name	Get	WORD	8057
0x02	Module Type	Get	CHAR	0 (DO type)
0x03	Configuration	Get/Set	Depend on the number of module channel	0x40
0x04	Total Channels	Get	CHAR	16
0x05	Total Length	Get	CHAR	2
0x06	Reserved	Get	CHAR	0

0x07	DO Length	Get	CHAR	2
0x08	AO Length	Get	CHAR	0
0x09	DI Length	Get	CHAR	0
0x0A	AI Length	Get	CHAR	0
0x0B	DO channel num	Get	CHAR	16
0x0C	AO channel num	Get	CHAR	0
0x0D	DI channel num	Get	CHAR	0
0x0E	AI channel num	Get	CHAR	0
0x0F	Enable/Disable output safe value	Get/Set	CHAR	0
0x10	Output safe value	Get/Set	Defined by module channel num	0
0x11	DI counter channel	Get/Set	CHAR	0
0x12	Clear DI counter value	Set	CHAR	0
0x13	DI counter value	Get	WORD	0
0x14	DO data	Set	Defined by module channel num	1st bit of 1st byte (1st channel) 2nd bit of 1st byte (2nd channel) 3rd bit of 1st byte (3rd channel) 4th bit of 1st byte (4th channel) 5th bit of 1st byte (5th channel) 6th bit of 1st byte (6th channel) 7th bit of 1st byte (7th channel) 8th bit of 1st byte (8th channel) 1st bit of 2nd byte (9th channel) 2nd bit of 2nd byte (10th channel) 3rd bit of 2nd byte (11th channel) 4th bit of 2nd byte (12th channel) 5th bit of 2nd byte (13th channel) 6th bit of 2nd byte (14th

				channel) 7th bit of 2nd byte (15th channel) 8th bit of 2nd byte (16th channel)
0x15	AO data	Set	Defined by module channel num	0
0x16	DI data	Get	Defined by module channel num	0
0x17	AI data	Get	Defined by module channel num	0
0x18	Clear Counter module	Set	CHAR	0
0x19	Counter module's Input Mode	Get/Set	CHAR	0

Application Instance 2

Attribute ID	Description	Method	Data Type	Value
0x01	Module name	Get	WORD	8064
0x02	Module Type	Get	CHAR	0 (DO type)
0x03	Configuration	Get/Set	Depend on the number of module channel	0x40
0x04	Total Channels	Get	CHAR	8
0x05	Total Length	Get	CHAR	1
0x06	Reserved	Get	CHAR	0
0x07	DO Length	Get	CHAR	1
0x08	AO Length	Get	CHAR	0
0x09	DI Length	Get	CHAR	0
0x0A	AI Length	Get	CHAR	0
0x0B	DO channel num	Get	CHAR	8
0x0C	AO channel num	Get	CHAR	0
0x0D	DI channel num	Get	CHAR	0

0x0E	AI channel num	Get	CHAR	0
0x0F	Enable/Disable output safe value	Get/Set	CHAR	0
0x10	Output safe value	Get/Set	Defined by module channel num	0
0x11	DI counter channel	Get/Set	CHAR	0
0x12	Clear DI counter value	Set	CHAR	0
0x13	DI counter value	Get	WORD	0
0x14	DO data	Set	Defined by module channel num	1st bit of 1st byte (1st channel) 2nd bit of 1st byte (2nd channel) 3rd bit of 1st byte (3rd channel) 4th bit of 1st byte (4th channel) 5th bit of 1st byte (5th channel) 6th bit of 1st byte (6th channel) 7th bit of 1st byte (7th channel) 8th bit of 1st byte (8th channel)
0x15	AO data	Set	Defined by module channel num	0
0x16	DI data	Get	Defined by module channel num	0
0x17	AI data	Get	Defined by module channel num	0
0x18	Clear Counter module	Set	CHAR	0
0x19	Counter module's Input Mode	Get/Set	CHAR	0

Application Instance 3

Attribute ID	Description	Method	Data Type	Value
0x01	Module name	Get	WORD	8042
0x02	Module Type	Get	CHAR	4 (DO_AND_DI_TYPE)
0x03	Configuration	Get/Set	Depend on the number of module channel	0x40
0x04	Total Channels	Get	CHAR	32
0x05	Total Length	Get	CHAR	4
0x06	Reserved	Get	CHAR	0
0x07	DO Length	Get	CHAR	2
0x08	AO Length	Get	CHAR	0
0x09	DI Length	Get	CHAR	2
0x0A	AI Length	Get	CHAR	0
0x0B	DO channel num	Get	CHAR	16
0x0C	AO channel num	Get	CHAR	0
0x0D	DI channel num	Get	CHAR	16
0x0E	AI channel num	Get	CHAR	0
0x0F	Enable/Disable output safe value	Get/Set	CHAR	0
0x10	Output safe value	Get/Set	Defined by module channel num	0
0x11	DI counter channel	Get/Set	CHAR	0
0x12	Clear DI counter value	Set	CHAR	0
0x13	DI counter value	Get	WORD	0
0x14	DO data	Set	Defined by module channel num	1st bit of 1st byte (1st channel) 2nd bit of 1st byte (2nd channel)

				3rd bit of 1st byte (3rd channel) 4th bit of 1st byte (4th channel) 5th bit of 1st byte (5th channel) 6th bit of 1st byte (6th channel) 7th bit of 1st byte (7th channel) 8th bit of 1st byte (8th channel) 1st bit of 2nd byte (9th channel) 2nd bit of 2nd byte (10th channel) 3rd bit of 2nd byte (11th channel) 4th bit of 2nd byte (12th channel) 5th bit of 2nd byte (13th channel) 6th bit of 2nd byte (14th channel) 7th bit of 2nd byte (15th channel) 8th bit of 2nd byte (16th channel)
0x15	AO data	Set	Defined by module channel num	0
0x16	DI data	Get	Defined by module channel num	1st bit of 1st byte (1st channel) 2nd bit of 1st byte (2nd channel) 3rd bit of 1st byte (3rd channel) 4th bit of 1st byte (4th channel) 5th bit of 1st byte (5th channel) 6th bit of 1st byte (6th channel) 7th bit of 1st byte (7th channel) 8th bit of 1st byte (8th channel) 1st bit of 2nd byte (9th channel) 2nd bit of 2nd byte (10th channel) 3rd bit of 2nd byte (11th channel) 4th bit of 2nd byte (12th channel) 5th bit of 2nd byte (13th channel) 6th bit of 2nd byte (14th

				channel) 7th bit of 2nd byte (15th channel) 8th bit of 2nd byte (16th channel)
0x17	AI data	Get	Defined by module channel num	0
0x18	Clear Counter module	Set	CHAR	0
0x19	Counter module's Input Mode	Get/Set	CHAR	0

Application Instance 4

Attribute ID	Description	Method	Data Type	Value
0x01	Module name	Get	WORD	8053
0x02	Module Type	Get	CHAR	1 (DI type)
0x03	Configuration	Get/Set	Depend on the number of module channel	0x40
0x04	Total Channels	Get	CHAR	16
0x05	Total Length	Get	CHAR	2
0x06	Reserved	Get	CHAR	0
0x07	DO Length	Get	CHAR	0
0x08	AO Length	Get	CHAR	0
0x09	DI Length	Get	CHAR	2
0x0A	AI Length	Get	CHAR	0
0x0B	DO channel num	Get	CHAR	0
0x0C	AO channel num	Get	CHAR	0
0x0D	DI channel num	Get	CHAR	16
0x0E	AI channel num	Get	CHAR	0
0x0F	Enable/Disable output safe value	Get/Set	CHAR	0
0x10	Output safe value	Get/Set	Defined by module channel num	0
0x11	DI counter	Get/Set	CHAR	0

	channel			
0x12	Clear DI counter value	Set	CHAR	0
0x13	DI counter value	Get	WORD	0
0x14	DO data	Set	Defined by module channel num	0
0x15	AO data	Set	Defined by module channel num	0
0x16	DI data	Get	Defined by module channel num	1st bit of 1st byte (1st channel) 2nd bit of 1st byte (2nd channel) 3rd bit of 1st byte (3rd channel) 4th bit of 1st byte (4th channel) 5th bit of 1st byte (5th channel) 6th bit of 1st byte (6th channel) 7th bit of 1st byte (7th channel) 8th bit of 1st byte (8th channel) 1st bit of 2nd byte (9th channel) 2nd bit of 2nd byte (10th channel) 3rd bit of 2nd byte (11th channel) 4th bit of 2nd byte (12th channel) 5th bit of 2nd byte (13th channel) 6th bit of 2nd byte (14th channel) 7th bit of 2nd byte (15th channel) 8th bit of 2nd byte (16th channel)
0x17	AI data	Get	Defined by module channel num	0
0x18	Clear Counter module	Set	CHAR	0
0x19	Counter module's Input Mode	Get/Set	CHAR	0

Refer to the application object instances. The CAN-8x24 will define the default assembly object instances according to the following table.

Assembly Object Instance ID (Hex)	Instance Type	Data Length(Byte)	Component modules
0x64	Output	DO: 5	i-8053/ i-8064/ i-8042
0x65	Input	DI: 4	i-8042/i-8057

If the default assembly instances are not adaptive, a user-defined assembly object can be created. The system will show the unit of IO modules in the CAN Slave Utility software. Therefore, users can arrange the unit to the specific assembly instances. In analog modules, the unit of length is 2 bytes where 1 byte is in the digital modules.

Users can assign assembly instances as either input or output instances. However there must be DO or AO units in output instances. Plus there must be DI or AI units in input instances. For example:

Assembly Object Instance ID (Hex)	Instance Type	Data Length(Byte)	Component modules
0x64	Output	2	1st byte DO (1st byte of i-8053), 2nd byte DO (1st byte of i-8064)
0x65	Output	3	1st byte DO (2nd byte of i-8053), 2nd byte DO (1st byte of i-8042), 3rd byte DO (2nd byte of i-8042)
0x66	Input	4	1st byte DI (2nd byte of i-8042), 2nd byte DI (1st byte of i-8042), 3rd byte DI (2nd byte of i-8057) 4th byte DI (2nd byte of i-8057)

6.3 CAN-8124/CAN-8224 Assembly Example

There are few IO examples related to the CAN-8124/CAN-8224 in this section. These demos should help users to understand the CAN-8124/ CAN-8224 to a suitable degree.

Example1: AI/AO modules demo

In this demo, apply the i-87017 (slot 0), i-8024 (slot 1) into the CAN-8224 as follows.

The CAN-8224 would arrange the application objects for the DeviceNet according to the following table.

Slot Address	Application Instance ID	Module name	DO Length(Byte)	AO Length(Byte)	DI Length(Byte)	AI Length(Byte)
0	0x01	87017	0	0	0	16
1	0x02	8024	0	8	0	0

Application Instance 1

Attribute ID	Description	Method	Data Type	Value
0x01	Module name	Get	WORD	87017
0x02	Module Type	Get	CHAR	3 (AI type)
0x03	Configuration	Get/Set	Depend on the number of module channel	1st byte (1st channel) 2nd byte (2nd channel) 3rd byte (3rd channel) 4th byte (4th channel) 5th byte (5th channel) 6th byte (6th channel) 7th byte (7th channel) 8th byte (8th channel)
0x04	Total Channels	Get	CHAR	8
0x05	Total Length	Get	CHAR	16
0x06	Reserved	Get	CHAR	0
0x07	DO Length	Get	CHAR	0
0x08	AO Length	Get	CHAR	0
0x09	DI Length	Get	CHAR	0
0x0A	AI Length	Get	CHAR	16
0x0B	DO channel num	Get	CHAR	0
0x0C	AO channel num	Get	CHAR	0
0x0D	DI channel num	Get	CHAR	0
0x0E	AI channel num	Get	CHAR	8
0x0F	Enable/Disable output safe value	Get/Set	CHAR	0
0x10	Output safe value	Get/Set	Defined by module channel num	0
0x11	DI counter channel	Get/Set	CHAR	0
0x12	Clear DI counter value	Set	CHAR	0
0x13	DI counter value	Get	WORD	0
0x14	DO data	Set	Defined by module channel	0

			num	
0x15	AO data	Set	Defined by module channel num	0
0x16	DI data	Get	Defined by module channel num	0
0x17	AI data	Get	Defined by module channel num	1st and 2nd bytes (1st channel) 3rd and 4th bytes (2nd channel) 5th and 6th bytes (3rd channel) 7th and 8th bytes (4th channel) 9th and 10th bytes (5th channel) 11th and 12th bytes (6th channel) 13th and 14th bytes (7th channel) 15th and 16th bytes (8th channel)
0x18	Clear Counter module	Set	CHAR	0
0x19	Counter module's Input Mode	Get/Set	CHAR	0

Application Instance 2

Attribute ID	Description	Method	Data Type	Value
0x01	Module name	Get	WORD	8024
0x02	Module Type	Get	CHAR	2 (AO type)
0x03	Configuration	Get/Set	Depend on the number of module channel	1st byte (1st channel) 2nd byte (2nd channel) 3rd byte (3rd channel) 4th byte (4th channel)
0x04	Total Channels	Get	CHAR	4
0x05	Total Length	Get	CHAR	8
0x06	Reserved	Get	CHAR	0
0x07	DO Length	Get	CHAR	0
0x08	AO Length	Get	CHAR	8
0x09	DI Length	Get	CHAR	0

0x0A	AI Length	Get	CHAR	0
0x0B	DO channel num	Get	CHAR	0
0x0C	AO channel num	Get	CHAR	4
0x0D	DI channel num	Get	CHAR	0
0x0E	AI channel num	Get	CHAR	0
0x0F	Enable/Disable output safe value	Get/Set	CHAR	0
0x10	Output safe value	Get/Set	Defined by module channel num	0
0x11	DI counter channel	Get/Set	CHAR	0
0x12	Clear DI counter value	Set	CHAR	0
0x13	DI counter value	Get	WORD	0
0x14	DO data	Set	Defined by module channel num	0
0x15	AO data	Set	Defined by module channel num	1st and 2nd bytes (1st channel) 3rd and 4th bytes (2nd channel) 5th and 6th bytes (3rd channel) 7th and 8th bytes (4th channel)
0x16	DI data	Get	Defined by module channel num	0
0x17	AI data	Get	Defined by module channel num	0
0x18	Clear Counter module	Set	CHAR	0
0x19	Counter	Get/Set	CHAR	0

	module's Input Mode			

Refer to the application object instances. The CAN-8224 will define the default assembly object instances according to the following table.

Assembly Object Instance ID(Hex)	Data Length(Byte)	Component modules
0x64	AO: 8	i-8024
0x65	AI: 8	i-87017
0x66	AI: 8	i-87017

Example 2: DI/DO module demo

In this demo, apply the i-8042 (slot 0) into the CAN-8124 as follows.

Slot Address	Application Instance ID	Module name	DO Length(Byte)	AO Length(Byte)	DI Length(Byte)	AI Length(Byte)
2	0x01	8042	2	0	2	0

Application Instance 1

Attribute ID	Description	Method	Data Type	Value
0x01	Module name	Get	WORD	8042
0x02	Module Type	Get	CHAR	4 (DO_AND_DI_TYPE)
0x03	Configuration	Get/Set	Depend on the number of module channel	0x40
0x04	Total Channels	Get	CHAR	32
0x05	Total Length	Get	CHAR	4
0x06	Reserved	Get	CHAR	0
0x07	DO Length	Get	CHAR	2
0x08	AO Length	Get	CHAR	0
0x09	DI Length	Get	CHAR	2
0x0A	AI Length	Get	CHAR	0
0x0B	DO channel	Get	CHAR	16

	num			
0x0C	AO channel num	Get	CHAR	0
0x0D	DI channel num	Get	CHAR	16
0x0E	AI channel num	Get	CHAR	0
0x0F	Enable/Disable output safe value	Get/Set	CHAR	0
0x10	Output safe value	Get/Set	Defined by module channel num	0
0x11	DI counter channel	Get/Set	CHAR	0
0x12	Clear DI counter value	Set	CHAR	0
0x13	DI counter value	Get	WORD	0
0x14	DO data	Set	Defined by module channel num	1st bit of 1st byte (1st channel) 2nd bit of 1st byte (2nd channel) 3rd bit of 1st byte (3rd channel) 4th bit of 1st byte (4th channel) 5th bit of 1st byte (5th channel) 6th bit of 1st byte (6th channel) 7th bit of 1st byte (7th channel) 8th bit of 1st byte (8th channel) 1st bit of 2nd byte (9th channel) 2nd bit of 2nd byte (10th channel) 3rd bit of 2nd byte (11th channel) 4th bit of 2nd byte (12th channel) 5th bit of 2nd byte (13th channel) 6th bit of 2nd byte (14th channel) 7th bit of 2nd byte (15th channel)

				8th bit of 2nd byte (16th channel)
0x15	AO data	Set	Defined by module channel num	0
0x16	DI data	Get	Defined by module channel num	1st bit of 1st byte (1st channel) 2nd bit of 1st byte (2nd channel) 3rd bit of 1st byte (3rd channel) 4th bit of 1st byte (4th channel) 5th bit of 1st byte (5th channel) 6th bit of 1st byte (6th channel) 7th bit of 1st byte (7th channel) 8th bit of 1st byte (8th channel) 1st bit of 2nd byte (9th channel) 2nd bit of 2nd byte (10th channel) 3rd bit of 2nd byte (11th channel) 4th bit of 2nd byte (12th channel) 5th bit of 2nd byte (13th channel) 6th bit of 2nd byte (14th channel) 7th bit of 2nd byte (15th channel) 8th bit of 2nd byte (16th channel)
0x17	AI data	Get	Defined by module channel num	0
0x18	Clear Counter module	Set	CHAR	0
0x19	Counter module's Input Mode	Get/Set	CHAR	0

Refer to the application object instances. The CAN-8124 will define the default

assembly object instances according to the following table.

Assembly Object Instance ID (Hex)	Instance Type	Data Length(Byte)	Component modules
0x64	Output	DO: 2	i-8042
0x65	Input	DI: 2	i-8042

Chapter 7 DeviceNet Communication Set

7.1 DeviceNet Communication Set Introduction

The CAN-8124/CAN-8224/CAN-8424 is a Group 2 Only Slave device, and supports the “Predefined Master/slave Connection Set”. To communicate with the device, the process about how to establish a connection is important. In addition, we also show examples about how to access IO modules.

The CAN Identifier Fields associated with the Predefined Master/Slave Connection Set for the DeviceNet are shown in the below table. Table defines the Identifiers that are to be used with all connection based messaging involved in the Predefined Master/Slave Connection Set for the CAN-8124/CAN-8224/CAN-8424.

IDENTIFIER BITS											IDENTITY USAGE	HEX
10	9	8	7	6	5	4	3	2	1	0		RANGE
0	Group 1 Message ID				Source MAC ID						Group 1 Messages	000 – 3ff
0	1	1	0	1	Source MAC ID						Slave’s I/O Change of State or Cyclic Message	
0	1	1	1	0	Source MAC ID						Slave’s I/O Bit–Strobe Response Message	
0	1	1	1	1	Source MAC ID						Slave’s I/O Poll Response or Change of State/Cyclic Acknowledge Message	
1	0	MAC ID						Group 2 Message ID			Group 2 Messages	400 – 5ff
1	0	Source MAC ID						0	0	0	Master’s I/O Bit–Strobe Command Message	
1	0	Destination MAC ID						0	1	0	Master’s Change of State or Cyclic Acknowledge Message	
1	0	Source MAC ID						0	1	1	Slave’s Explicit/ Unconnected Response Messages/ Device Heartbeat Message/ Device Shutdown Message	
1	0	Destination MAC ID						1	0	0	Master’s Explicit Request Messages	
1	0	Destination MAC ID						1	0	1	Master’s I/O Poll Command/Change of State/Cyclic Message	
1	0	Destination MAC ID						1	1	0	Group 2 Only Unconnected Explicit Request Messages (reserved)	

1	0	Destination MAC ID				1	1	1	Duplicate MAC ID Check Messages	
1	1	1	1	1	Group 4 Message ID				Group 4 Messages	000 – 3ff
1	1	1	1	1	2C				Communication Faulted Response Message	
1	1	1	1	1	2D				Communication Faulted Request Message	

The following table lists the Error Codes that may be present in the General Error Code field of an Error Response message.

Error Condition	General Error code(Hex)	Additional Error Condition	Additional Error Code(Hex)
Resource unavailable	02	Invalid allocation choice	02
		Invalid Unconnected request	03
		Poll After COS_CYCLIC	04
Service not support	08	None	FF
Invalid attribute value	9	None	FF
Already in requested mode/state	0B	None	FF
Object state conflict	0C	Class specific error	01
Attribute not settable	0E	None	FF
Privilege violation	0F	None	FF
Device state conflict	10	None	FF
Reply data too large	11	None	FF
Not enough data	13	None	FF
Attribute not	14	None	FF

supported			
Too much data	15	None	FF
Object does not exist	16	None	FF
FRAGMENTATION EQ	17	None	FF
Invalid parameter	20	None	FF

The following steps may be useful to those users who would like to implement their own DeviceNet applications using the command set.

1. **Request the use of the Predefined Master/Slave Connection Set.**

2. **Apply the Master's Explicit Request Messages to set an expected_packet_rate attribute for the IO connection to establish an I/O Connection Object State.**

.

3. **There are two ways to access the IO modules. The first method is by way of an IO connection object. The other is by using an explicit message to set/get the IO attribute for the application object.**

4. **Release the use of the Predefined Master/Slave Connection Set.**

7.2 Examples of the DeviceNet communication set

7.2.1 Requests the use of the Predefined Master/Slave Connection Set

An unconnected explicit messaging request sent by the Master node via a destination node's Group 2 Only Unconnected Explicit Request Message to request the use of the Predefined Master/Slave Connection set. This example is shows the user how to use it. In this demo, the Master establishes the Explicit Message, Poll IO and Bit-Strobe IO connections.

The figure below shows the Group 2 Only Unconnected Explicit connection Identifier Fields.

IDENTIFIER BITS											IDENTITY USAGE	HEX RANGE
10	9	8	7	6	5	4	3	2	1	0		
1	0	Source MAC ID						0	1	1	Slave's Explicit/ Unconnected Response Messages	
1	0	Destination MAC ID						1	1	0	Group 2 Only Unconnected Explicit Request Messages	

Note: CAN-8424: Node ID=0x09, Master node ID=0x0a

Master (MAC ID =0x0A) Slave (MAC ID =0x09)

```
_ Group 2 Message
|   _ Destination MAC ID=0x09
|   |   _ Message ID =6
|   |   |   _ Frag=0. Transaction ID=0. Source MAC ID= 0x0A
|   |   |   |   _ Service=Allocate_Master/Slave_Connection_Set Request
|   |   |   |   |   _ Class ID=3
|   |   |   |   |   |   _ Instance ID=1
|   |   |   |   |   |   |   _ Allocation Choice=Explicit, Polled & Bit Strobed
|   |   |   |   |   |   |   |   _ Allocator's MAC ID=0x0A
ID=10 001001 110. Data= 0A 4B 03 01 07 0A
```

```
_ Group 2 Message
|   _ Source MAC ID=0x09
|   |   _ Message ID =3
|   |   |   _ Frag=0. Transaction ID=0. Source MAC ID= 0x0A
|   |   |   |   _ Service=Allocate_Master/Slave_Connection_Set Response
|   |   |   |   |   _Connection Message Body Format = DeviceNet (8/8)
|   |   |   |   |   |
|   |   |   |   |   |
|   |   |   |   |   |
ID=10 001001 011. Data= 0A CB 00
```

7.2.2 How to apply the Poll IO connection

Poll Command and Response message move any amount of I/O data between a Master and its Polled Slaves. This example is revealed how to apply the Poll IO connection in the DeviceNet application.

Note: CAN-8x24: Node ID=0x09, Master node ID=0x0A

The figure below shows the Poll I/O connection Identifier Fields.

IDENTIFIER BITS											IDENTITY USAGE	HEX
10	9	8	7	6	5	4	3	2	1	0		RANGE
1	0	Destination MAC ID						1	0	1	Master's I/O Poll Command/Change of State/Cyclic Message	
1	0	Source MAC ID						0	1	1	Slave's Explicit/ Unconnected Response Messages	
1	0	Destination MAC ID						1	1	0	Group 2 Only Unconnected Explicit Request Messages	
1	0	Destination MAC ID						1	0	0	Master's Explicit Request Messages	
0	1	1	1	1	Source MAC ID						Slave's I/O Poll Response Message	

1. Requests the use of the Predefined Master/Slave Connection set

Master (MAC ID =0x0A) Slave (MAC ID =0x09)

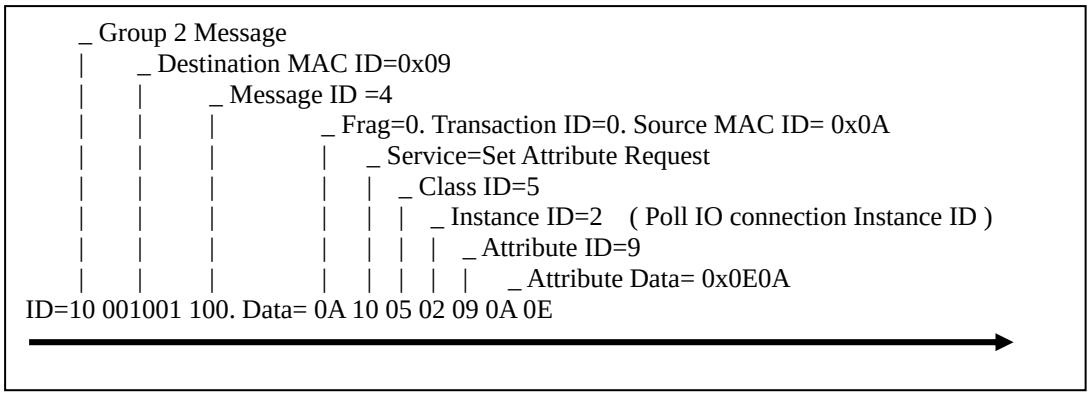
```
_ Group 2 Message
|   _ Destination MAC ID=0x09
|   |   _ Message ID =6
|   |   |   _ Frag=0. Transaction ID=0. Source MAC ID= 0x0A
|   |   |   |   _ Service=Allocate_Master/Slave_Connection_Set Request
|   |   |   |   |   _ Class ID=3
|   |   |   |   |   |   _ Instance ID=1
|   |   |   |   |   |   |   _ Allocation Choice=Explicit, Polled & Bit Strobed
|   |   |   |   |   |   |   |   _ Allocator's MAC ID=0x0A
ID=10 001001 110. Data= 0A 4B 03 01 07 0A
```

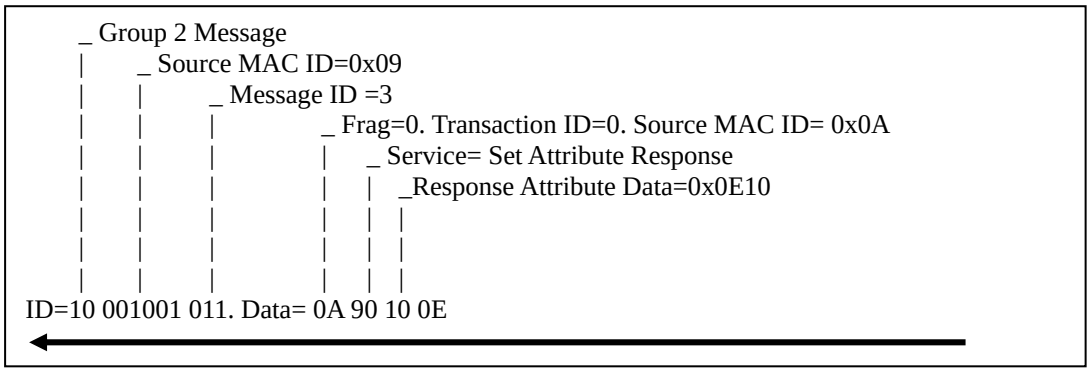
```
_ Group 2 Message
|   _ Source MAC ID=0x09
|   |   _ Message ID =3
|   |   |   _ Frag=0. Transaction ID=0. Source MAC ID= 0x0A
|   |   |   |   _ Service=Allocate_Master/Slave_Connection_Set Response
|   |   |   |   |   _Connection Message Body Format = DeviceNet (8/8)
|   |   |   |   |   |
|   |   |   |   |   |
|   |   |   |   |   |
ID=10 001001 011. Data= 0A CB 00
```

2. Apply the Master's Explicit Request Messages to set the

expected_packet_rate attributes so that the IO connection can establish the I/O Connection Object State.

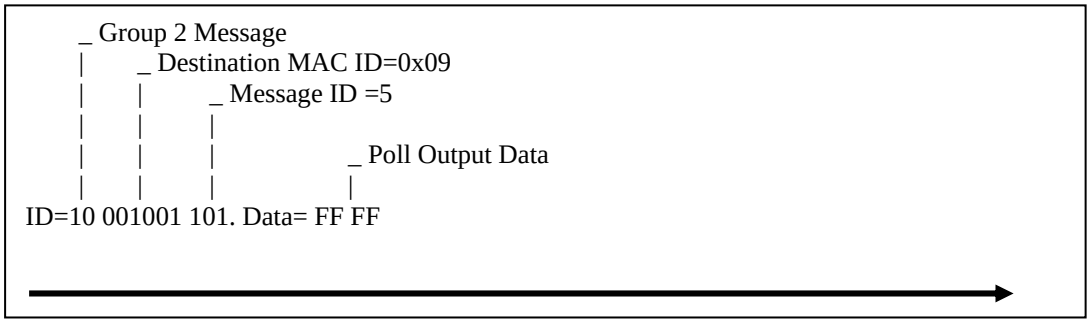
Master (MAC ID =0x0A) Slave (MAC ID =0x09)

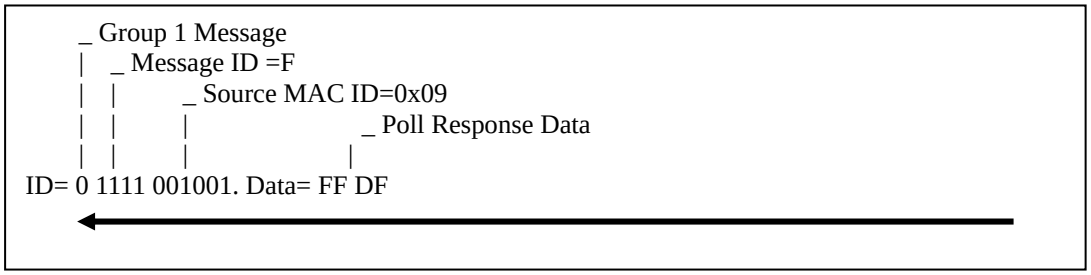




3. Apply the Poll I/O connection to access the IO modules

Master (MAC ID =0x0A) Slave (MAC ID =0x09)





7.2.3 The Bit-Strobe IO connection example

Bit-Strobe Command and Response messages rapidly move small amounts of I/O data between a Master and its Bit-Strobe Slaves.

The figure below shows Bit-Strobe I/O connection Identifier Fields.

IDENTIFIER BITS											IDENTITY USAGE	HEX
10	9	8	7	6	5	4	3	2	1	0		RANGE
0	1	1	1	0	Source MAC ID						Slave's I/O Bit–Strobe Response Message	
1	0	Source MAC ID						0	0	0	Master's I/O Bit–Strobe Command Message	
1	0	Source MAC ID						0	1	1	Slave's Explicit/ Unconnected Response Messages	
1	0	Destination MAC ID						1	1	0	Group 2 Only Unconnected Explicit Request Messages	
1	0	Destination MAC ID						1	0	0	Master's Explicit Request Messages	

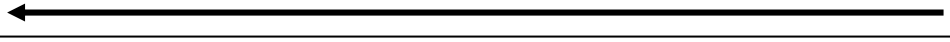
Note: CAN-8x24: Node ID=0x09, Master node ID=0x0A

1. Requests the use of the Predefined Master/Slave Connection set

Master (MAC ID =0x0A) Slave (MAC ID =0x09)

```
      _ Group 2 Message
      |     _ Destination MAC ID=0x09
      |     |     _ Message ID =6
      |     |     |           _ Frag=0. Transaction ID=0. Source MAC ID= 0x0A
      |     |     |           |     _ Service=Allocate_Master/Slave_Connection_Set Request
      |     |     |           |     |   _ Class ID=3
      |     |     |           |     |   |   _ Instance ID=1
      |     |     |           |     |   |   |   _ Allocation Choice=Explicit, Polled & Bit Strobed
      |     |     |           |     |   |   |   |     _ Allocator's MAC ID=0x0A
ID=10 001001 110. Data= 0A 4B 03 01 07 0A
```

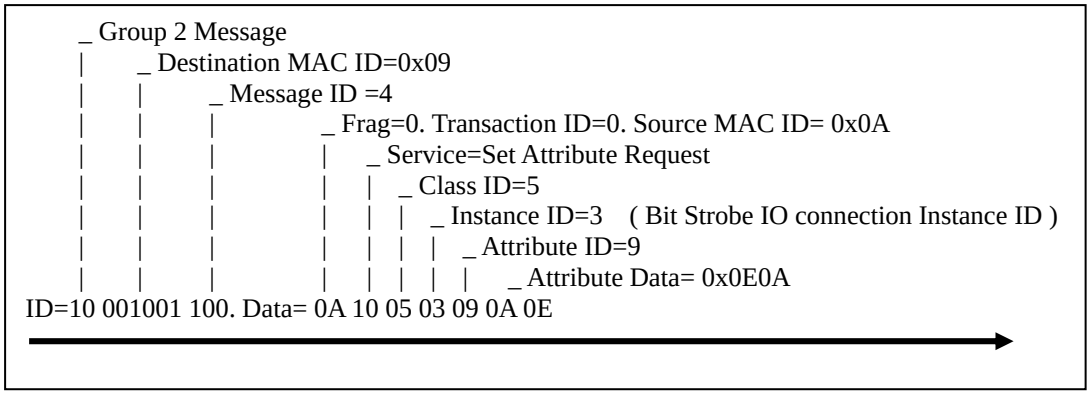
```
      _ Group 2 Message
      |     _ Source MAC ID=0x09
      |     |     _ Message ID =3
      |     |     |           _ Frag=0. Transaction ID=0. Source MAC ID= 0x0A
      |     |     |           |     _ Service=Allocate_Master/Slave_Connection_Set Response
      |     |     |           |     |   _Connection Message Body Format = DeviceNet (8/8)
      |     |     |           |     |   |
      |     |     |           |     |   |
      |     |     |           |     |   |
ID=10 001001 011. Data= 0A CB 00
```

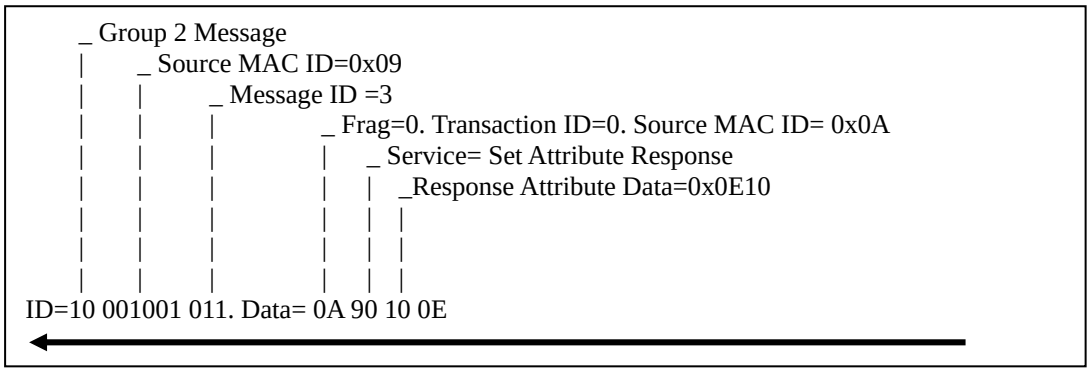


2. Apply Master's Explicit Request Messages to set

expected_packet_rate attribute of IO connection to make I/O Connection Object State established.

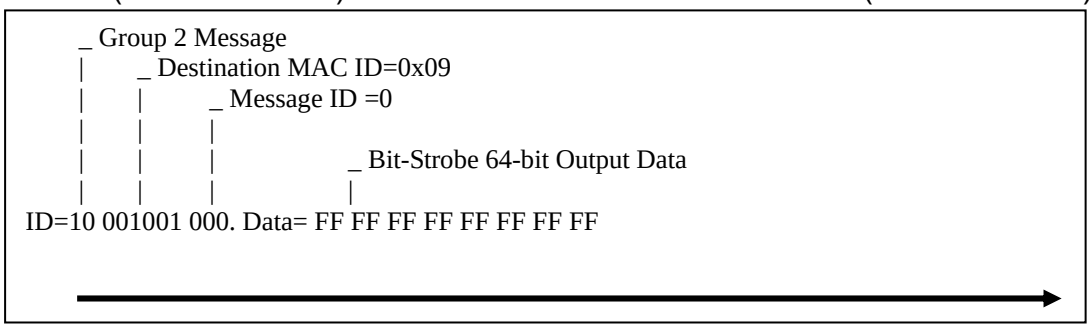
Master (MAC ID =0x0A) Slave (MAC ID =0x09)

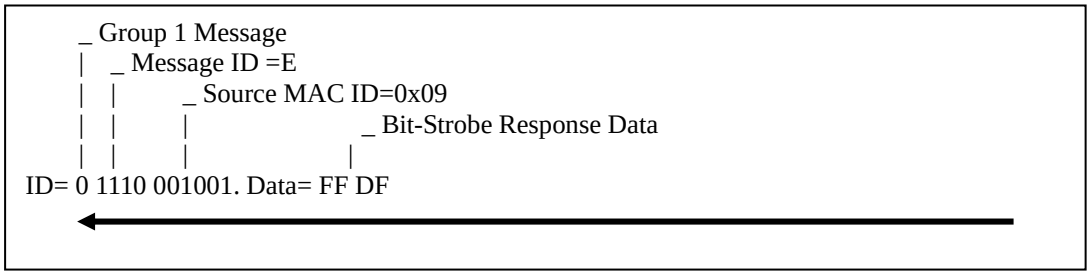




3. Apply Bit-Strobe I/O connection to access IO modules

Master (MAC ID =0x0A) Slave (MAC ID =0x09)





7.2.4 Change of State/Cyclic IO with Acknowledge connections

The Change of State/Cyclic connection sets use connection instance 2 (the polled connection instance) for master to slave data production and slave to master acknowledgment. Connection instance 4 is used for slave to master data production and master to slave acknowledgment. If a device does not support the poll, and has no support for output data, connection instance 2 does not need to be instantiated.

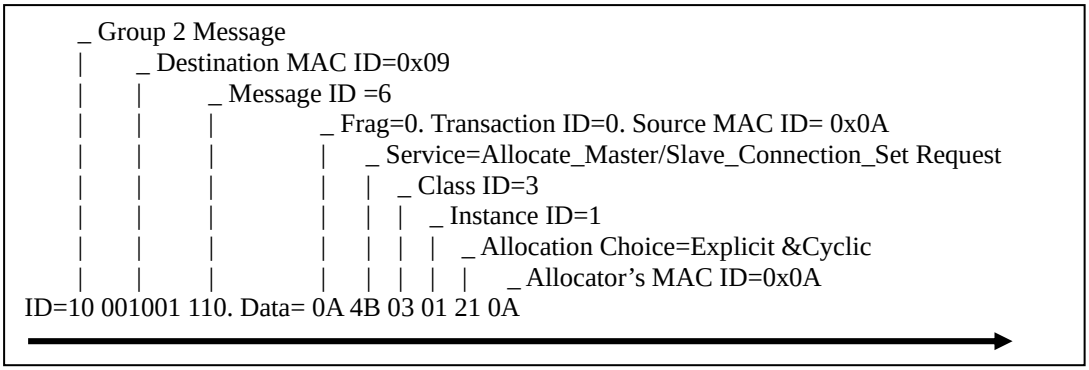
The figure below shows COS/Cyclic I/O connection Identifier Fields.

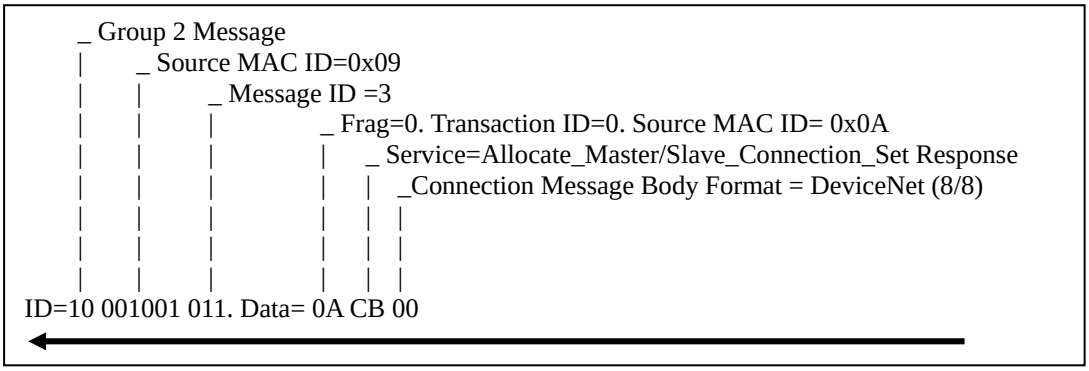
IDENTIFIER BITS											IDENTITY USAGE	HEX
10	9	8	7	6	5	4	3	2	1	0		RANGE
0	1	1	0	1	Source MAC ID						Slave's I/O Change of State or Cyclic Message	
1	0	Destination MAC ID						0	1	0	Master's Change of State or Cyclic Acknowledge Message	
1	0	Source MAC ID						0	1	1	Slave's Explicit/ Unconnected Response Messages	
1	0	Destination MAC ID						1	1	0	Group 2 Only Unconnected Explicit Request Messages	
1	0	Destination MAC ID						1	0	0	Master's Explicit Request Messages	

Note: CAN-8x24: Node ID=0x09, Master node ID=0x0A

1. Requests the use of the Predefined Master/Slave Connection set

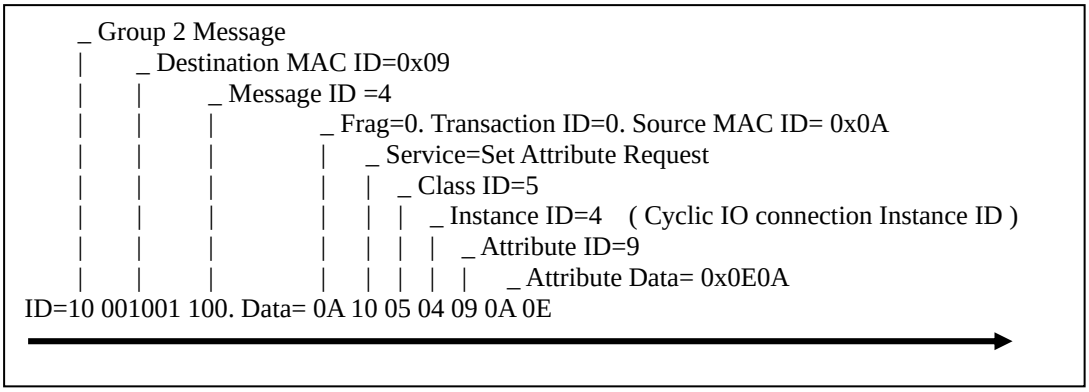
Master (MAC ID =0x0A) Slave (MAC ID =0x09)

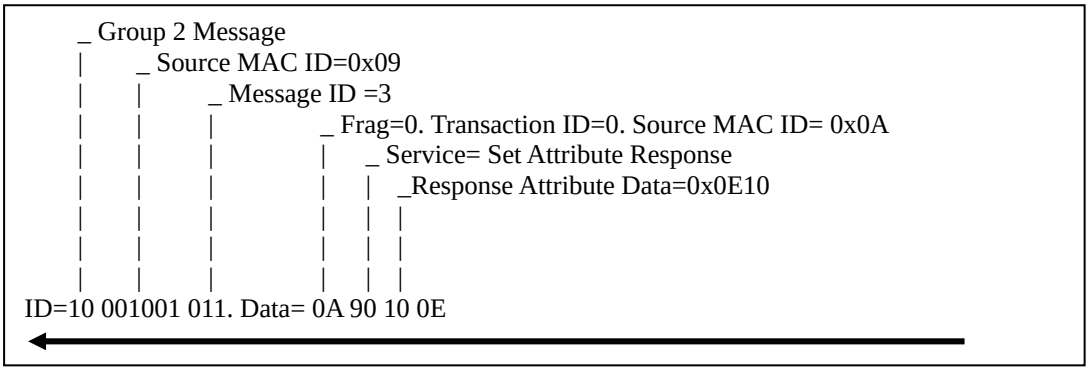




2. Apply the Master's Explicit Request Messages to set an expected_packet_rate attribute for the IO connection to establish an I/O Connection Object State.

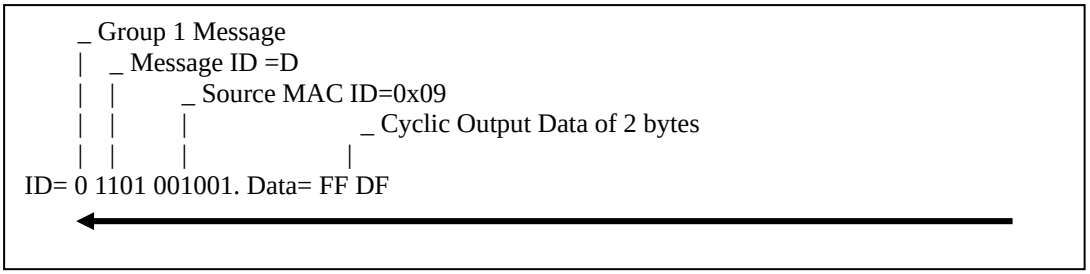
Master (MAC ID =0x0A) Slave (MAC ID =0x09)

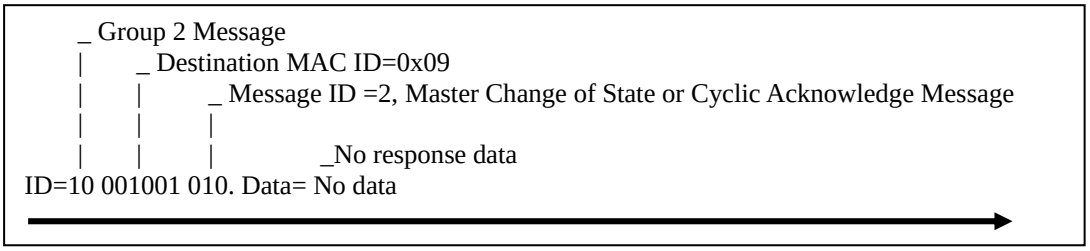




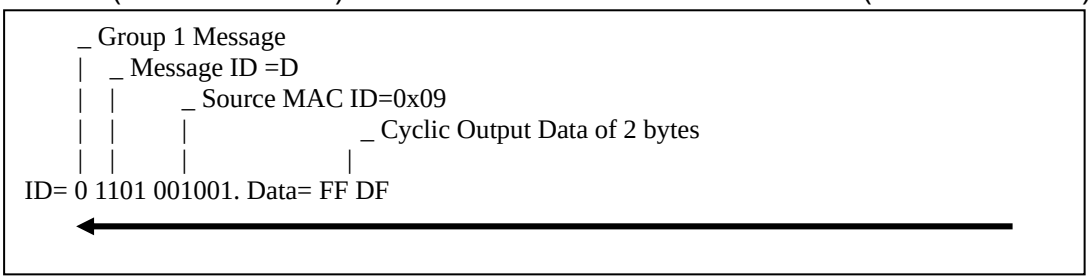
3. The Cyclic I/O connection starts to transfer the IO data

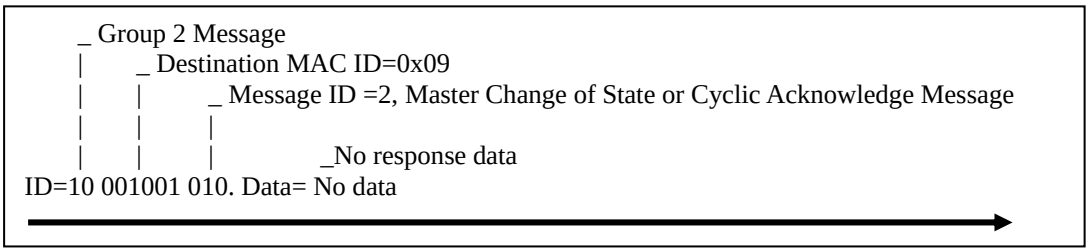
Master (MAC ID =0x0A) Slave (MAC ID =0x09)





Master (MAC ID =0x0A) Slave (MAC ID =0x09)





7.2.5 Change of State/Cyclic IO without Acknowledge connections

This example shows how to apply the COS/Cyclic IO without an acknowledge connection.

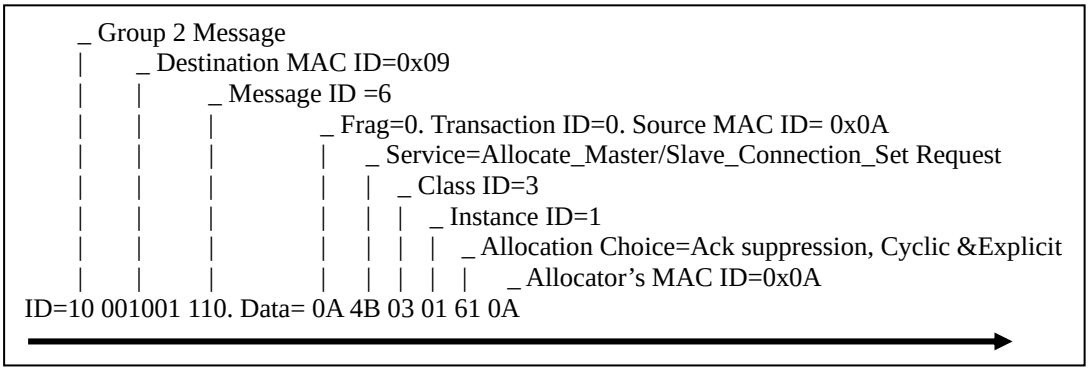
The figure below shows the COS/Cyclic I/O connection Identifier Fields.

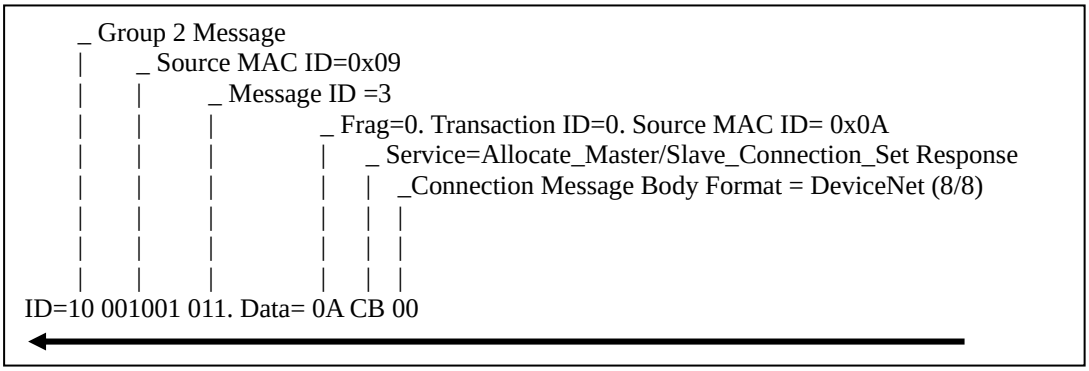
IDENTIFIER BITS											IDENTITY USAGE	HEX RANGE
10	9	8	7	6	5	4	3	2	1	0		
0	1	1	0	1	Source MAC ID						Slave's I/O Change of State or Cyclic Message	
1	0	Source MAC ID						0	1	1	Slave's Explicit/ Unconnected Response Messages	
1	0	Destination MAC ID						1	1	0	Group 2 Only Unconnected Explicit Request Messages	
1	0	Destination MAC ID						1	0	0	Master's Explicit Request Messages	

Note: CAN-8x24: Node ID=0x09, Master node ID=0x0A

1. Requests the use of the Predefined Master/Slave Connection set

Master (MAC ID =0x0A) Slave (MAC ID =0x09)

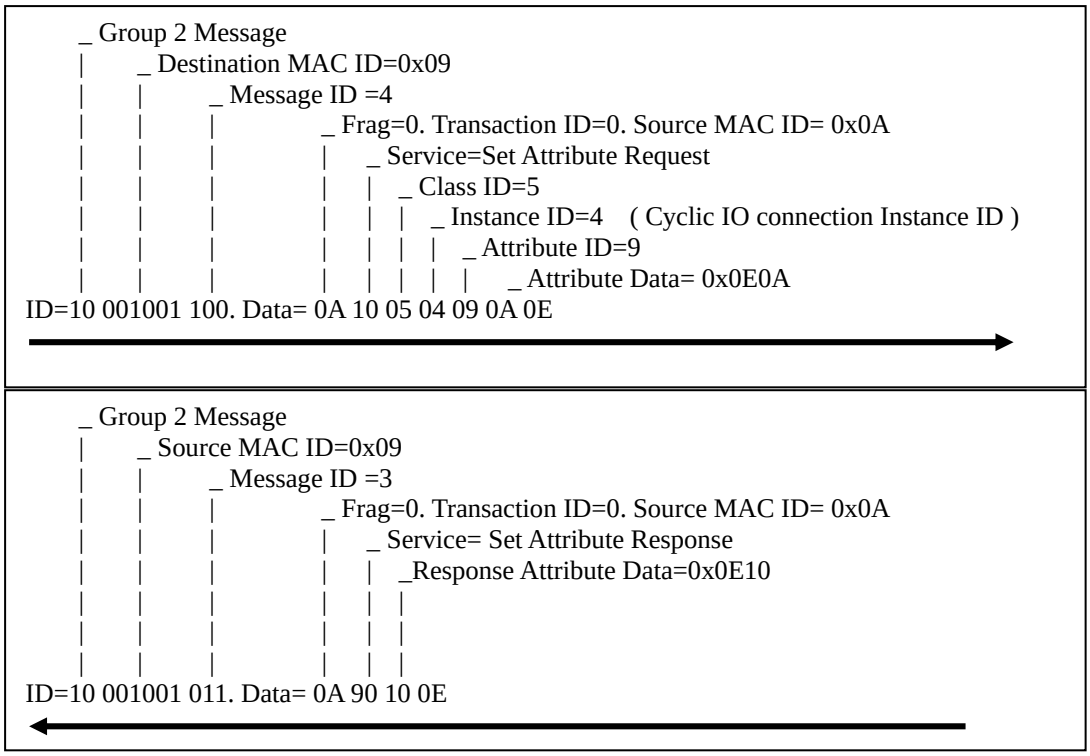




2. Apply the Master's Explicit Request Messages to set an

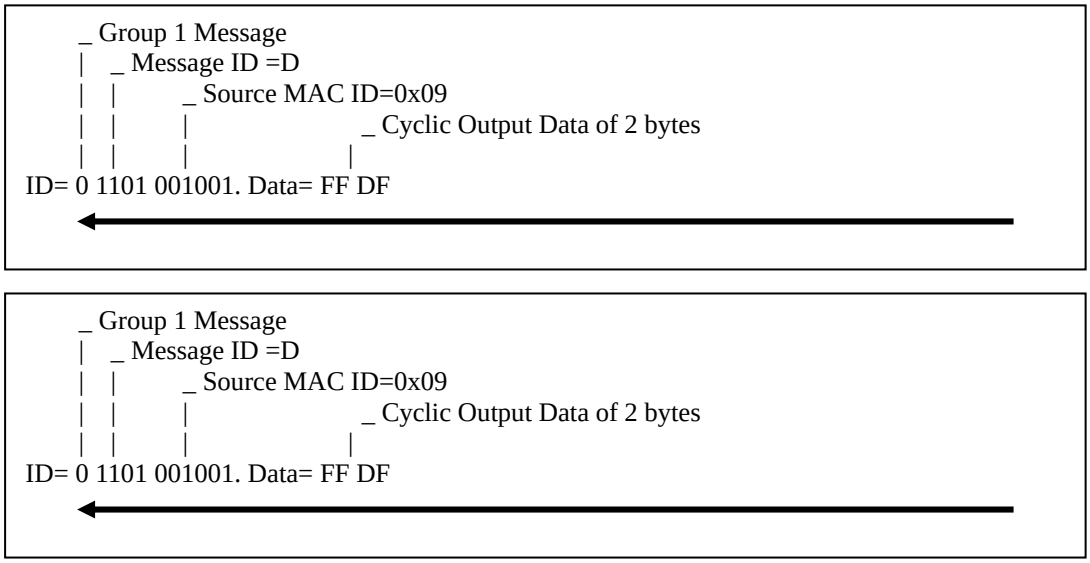
expected_packet_rate attribute for the IO connection to establish an I/O Connection Object State.

Master (MAC ID =0x0A) Slave (MAC ID =0x09)



3. The slave device starts to transmit data cyclically.

Master (MAC ID =0x0A) Slave (MAC ID =0x09)



7.2.6 Reset Service

This service can reset the device. If users want to reset the device, they can apply this service to the device

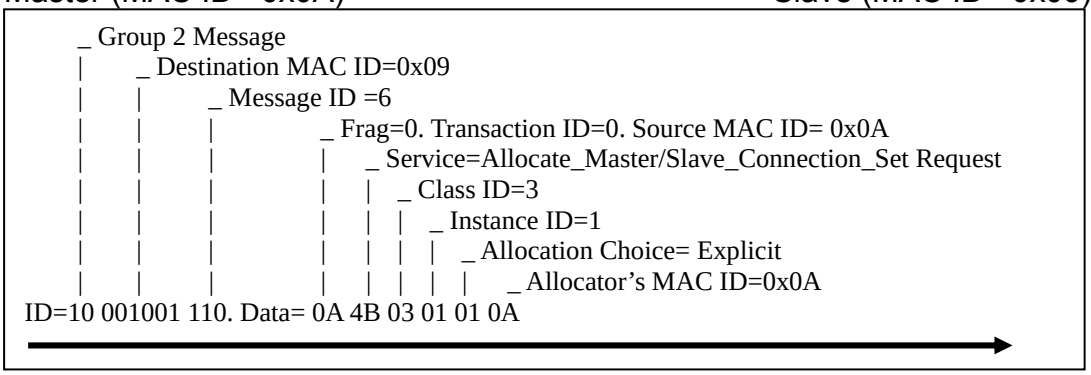
The parameter type for the reset common service has the following bit specifications:

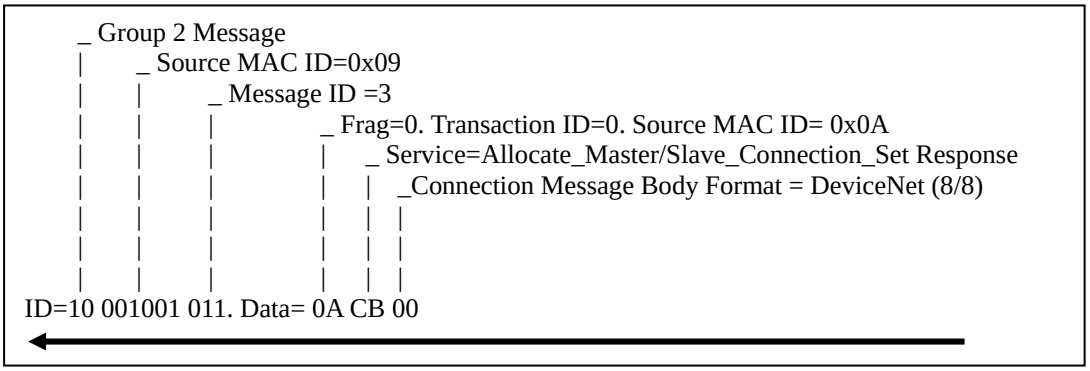
Value	Type of Reset
0	Emulate as closely as possible cycling power on the item the Identity object represents.
1	Return as closely as possible to the out-of-box configuration, then emulate the cycling power as closely as possible.

Note: CAN-8x24: Node ID=0x09, Master node ID=0x0A

1. Requests the use of the Predefined Master/Slave Connection set

Master (MAC ID =0x0A) Slave (MAC ID =0x09)

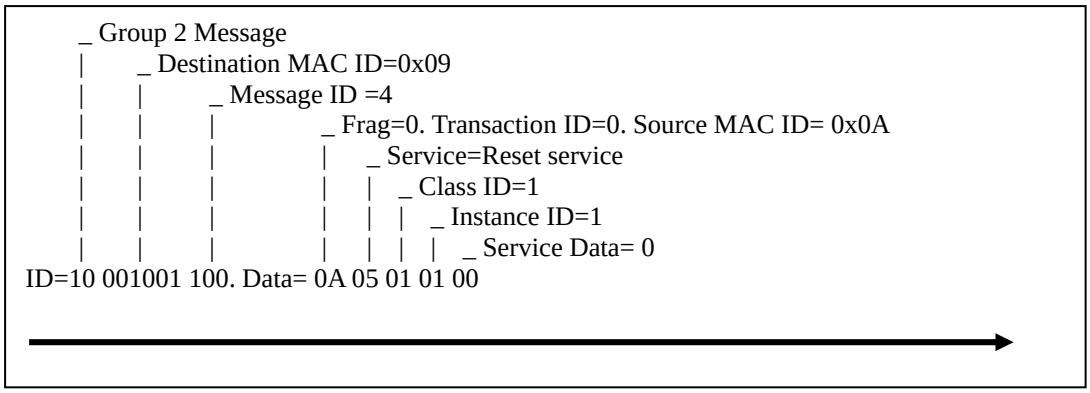


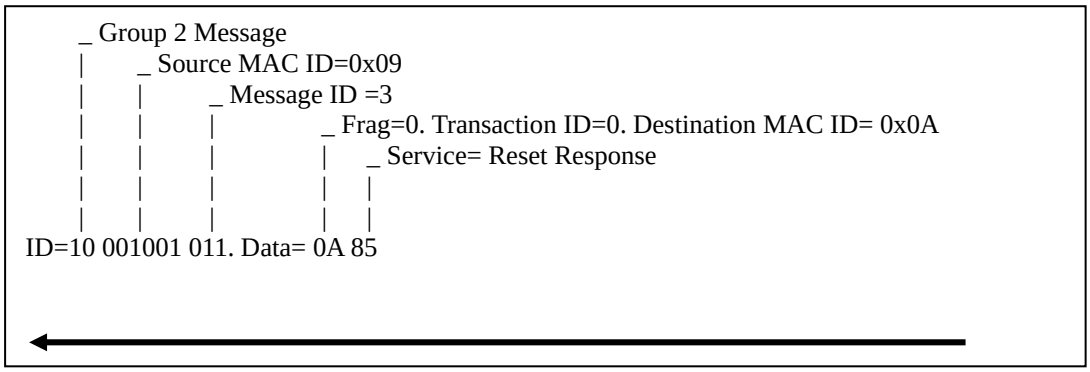


2. Apply the Master's Explicit Request Messages to set the Identify

object. The service ID(0x05) is the reset service.

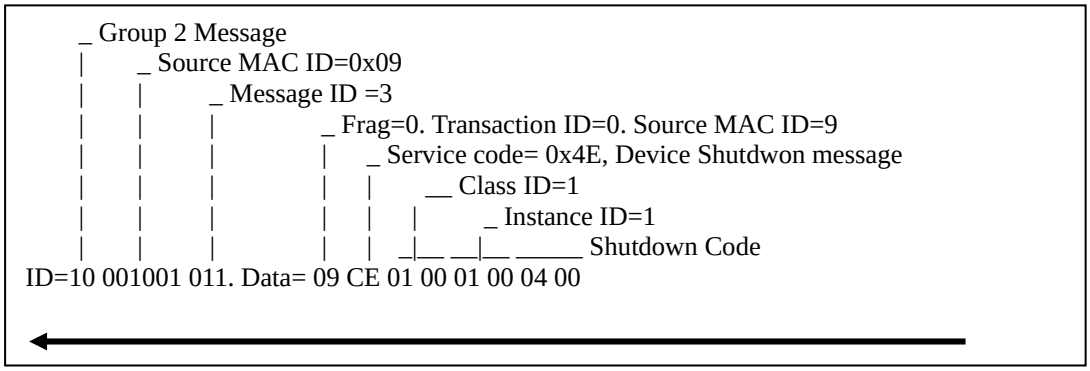
Master (MAC ID =0x0A) Slave (MAC ID =0x09)





Then the slave sends the shutdown message to the CAN bus.

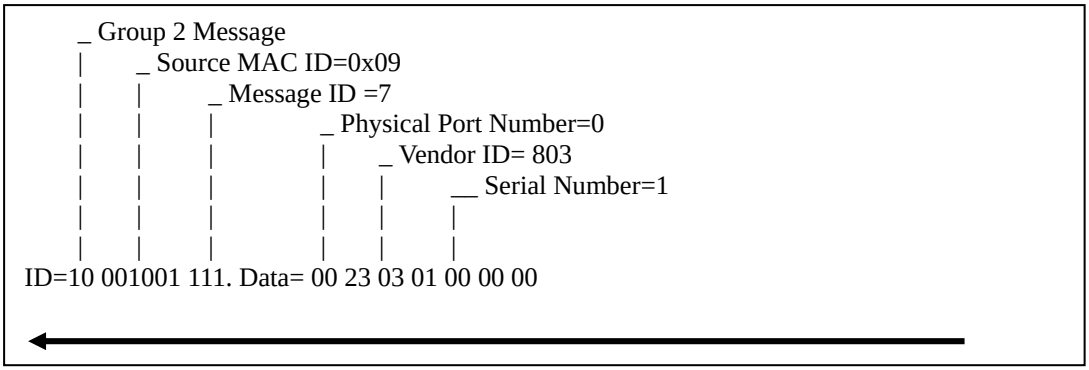
Master (MAC ID =0x0A) Slave (MAC ID =0x09)

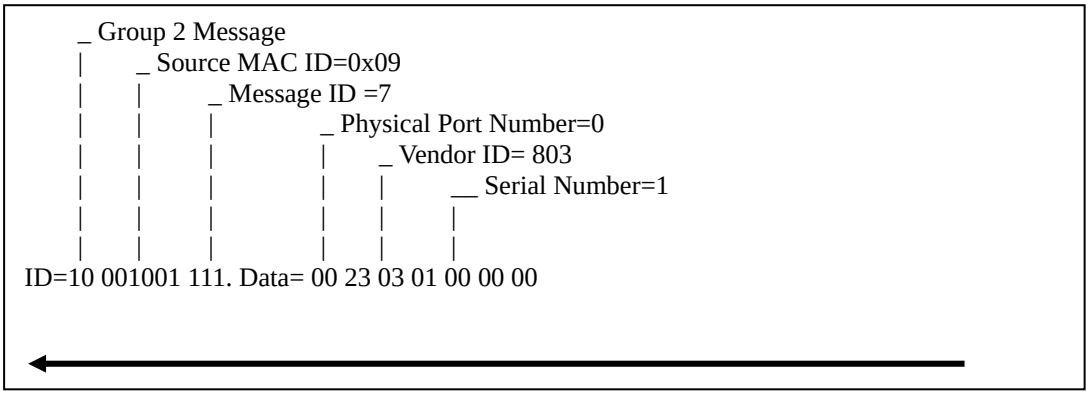


3. After the device sends out the shutdown message, it will reset and

send Duplicated ID messages.

Master (MAC ID =0x0A) Slave (MAC ID =0x09)





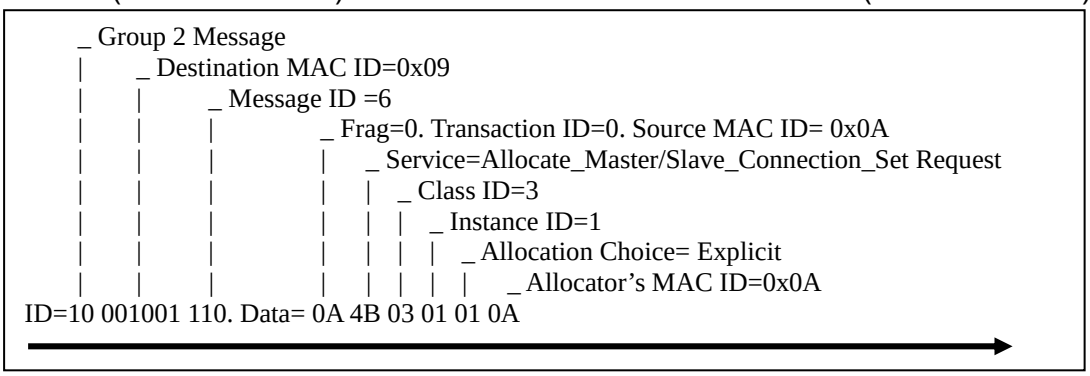
7.2.7 Device Heartbeat

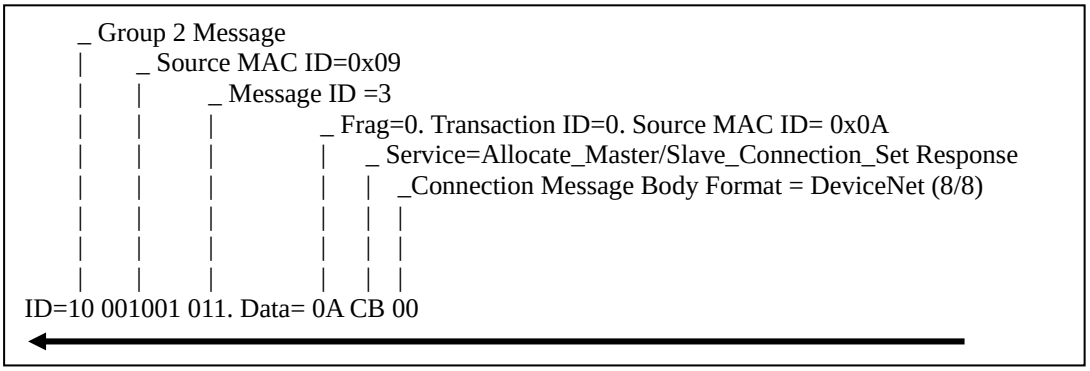
This message broadcasts the current state of the device. This message is transmitted by a group 2 only server as an Unconnected Response Message (Message Group 2, Message ID 3).

Note: CAN-8x24: Node ID=0x09, Master node ID=0x0A

1. Requests the use of the Predefined Master/Slave Connection set

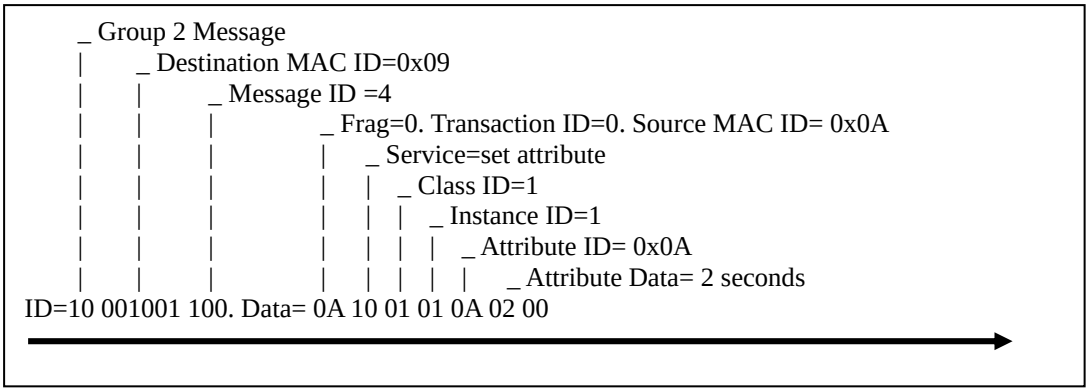
Master (MAC ID =0x0A) Slave (MAC ID =0x09)





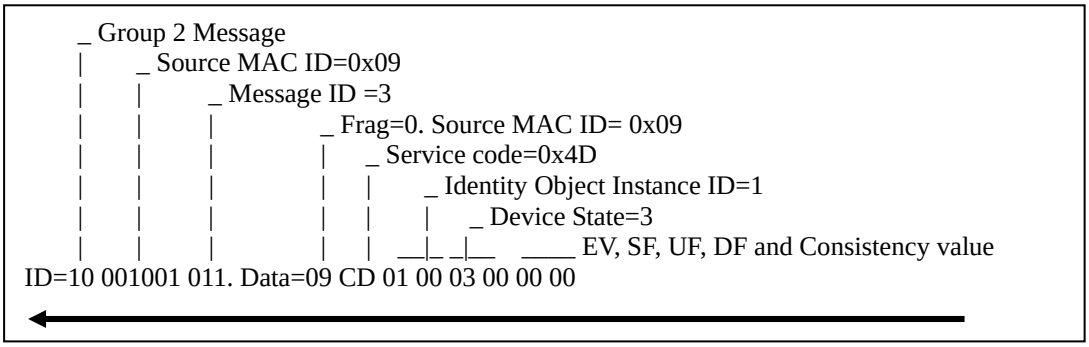
2. Apply the Master's Explicit Request Messages to set the heartbeat interval value.

Master (MAC ID =0x0A) Slave (MAC ID =0x09)



Then slave (MAC ID =0x09) would send the heartbeat message in every 2 seconds.

Master (MAC ID =0x0A) Slave (MAC ID =0x09)



Note: If users want to cancel the heartbeat message, please set 0 into the heartbeat interval attribute value in the Identity object instance.

7.2.8 Fragmentation example

There are 2 kinds of fragmentation messages in the DeviceNet. One is acknowledged fragmentation for explicit message. The other is the unacknowledged fragmentation for IO messages. If the length of the message data is greater than 8 bytes, this message must be fragmented to be sent.

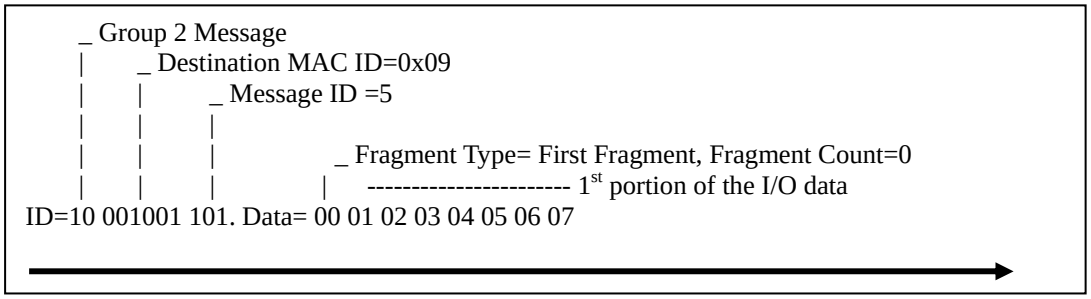
- **Unacknowledged Fragmentation example**

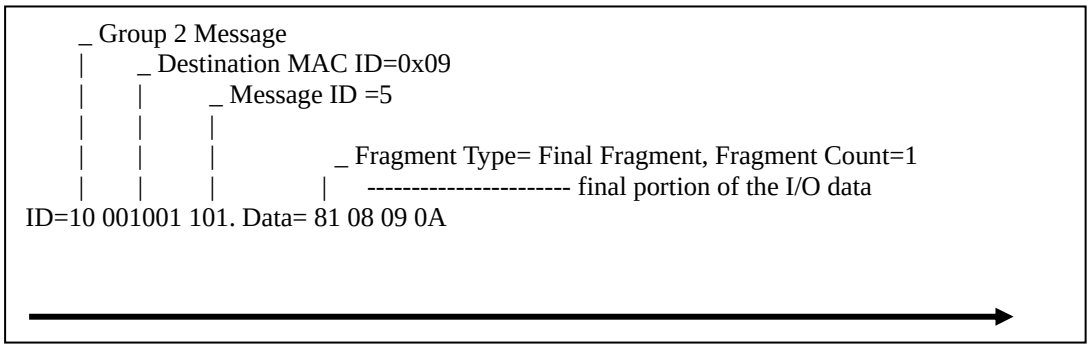
This example relates to when fragmentation of an I/O message is performed in an unacknowledged fashion. Unacknowledged fragmentation consists of the back-back transmission of the fragments from the transmitting module. The receiving module(s) returns no acknowledgments (other than the CAN-provided Ack) on a per-fragment basis. The connection simply invokes the Link Producer's Send service as necessary to move the message without waiting for any specific acknowledgment from the receiving module(s).

In this demo, the polling consumed size is 10 bytes. The master must send the fragmented messages. Data=0102030405060708090A. Assume that an I/O Connection has been established.

Note: The slave device node is 0x09, and the master node ID is 0x0A

Master (MAC ID =0x0A) Slave (MAC ID =0x09)





- **Acknowledge Fragmentation**

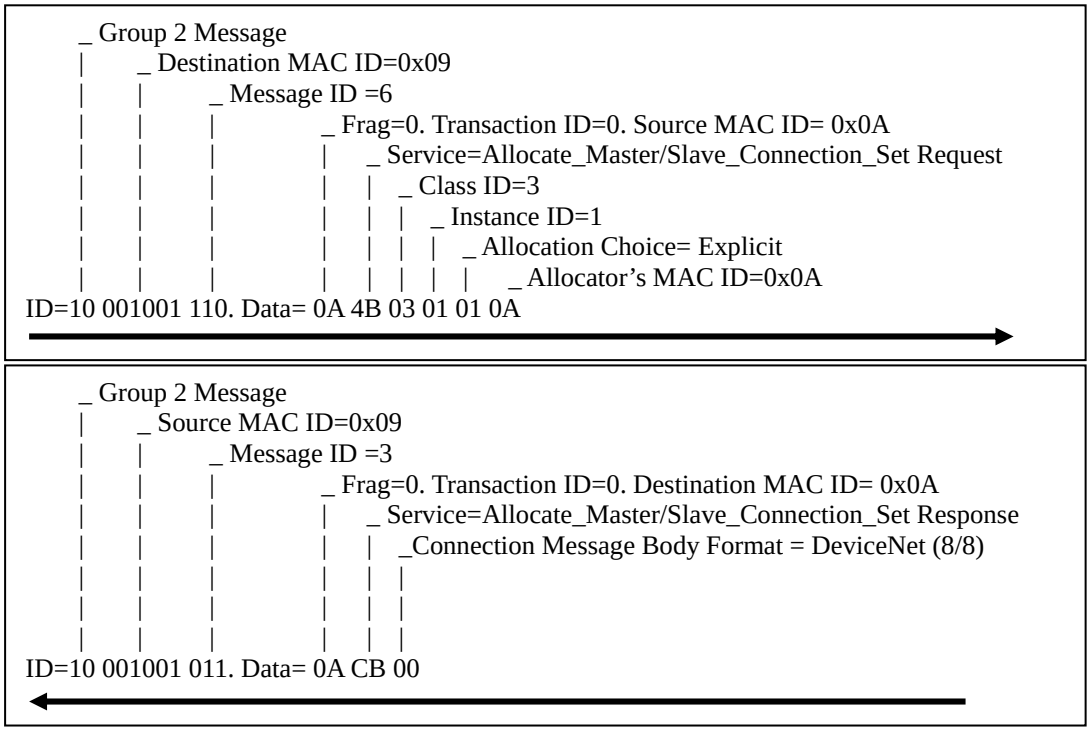
This example relates to when fragmentation of an Explicit Message is performed in an Acknowledged fashion. Acknowledged fragmentation consists of the transmission of a fragment from the transmitting module followed by the transmission of an acknowledgment by the receiving module. The receiving module acknowledges the reception of each fragment. This provides a degree of flow control. The assumption is that larger bodies of information may be moved across Explicit Messaging Connections (e.g. Upload/Download functions) and, as such, a degree of flow control is necessary.

In this demo, assume that attribute data=0102030405060708090A. The assembly instance ID=4, attribute=3.

Note: The slave device node is 0x09, and the master node ID is 0x0A

1. Requests the use of the Predefined Master/Slave Connection set

Master (MAC ID =0x0A) Slave (MAC ID =0x09)

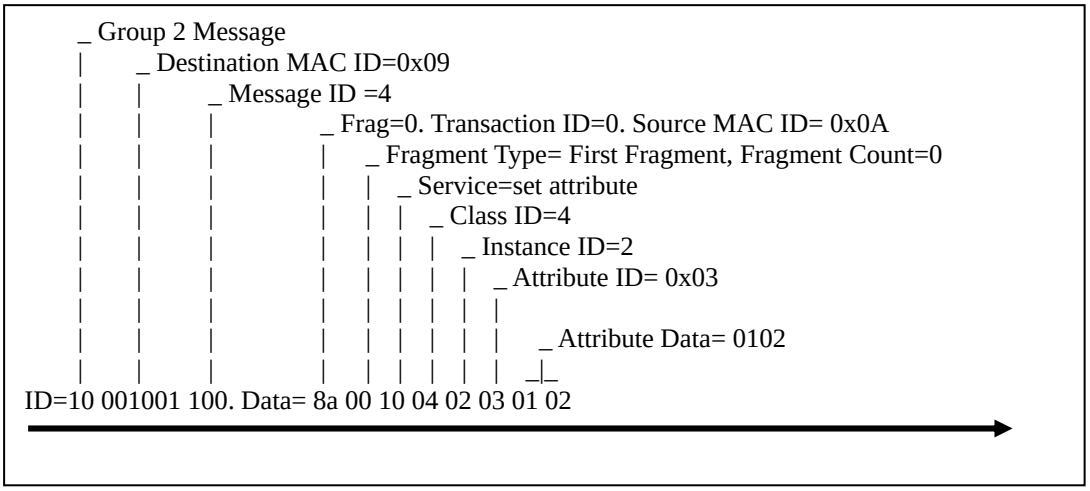


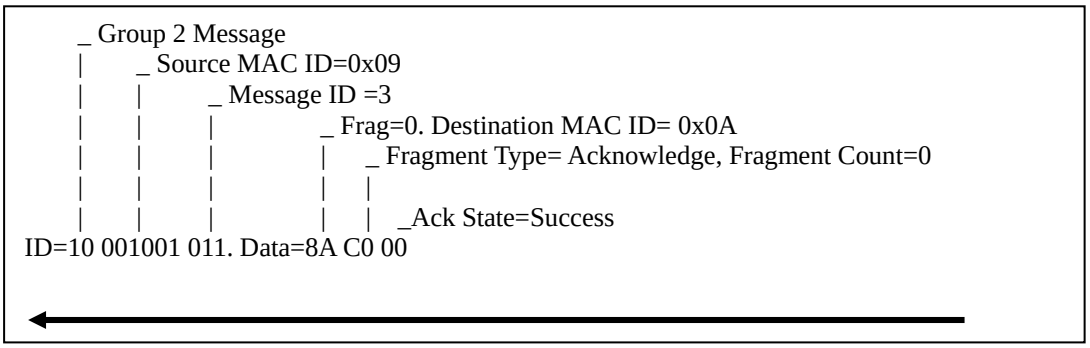
2. Apply the Master's Explicit Request Messages to set the Assembly object

Service(0x10) = set attribute service.

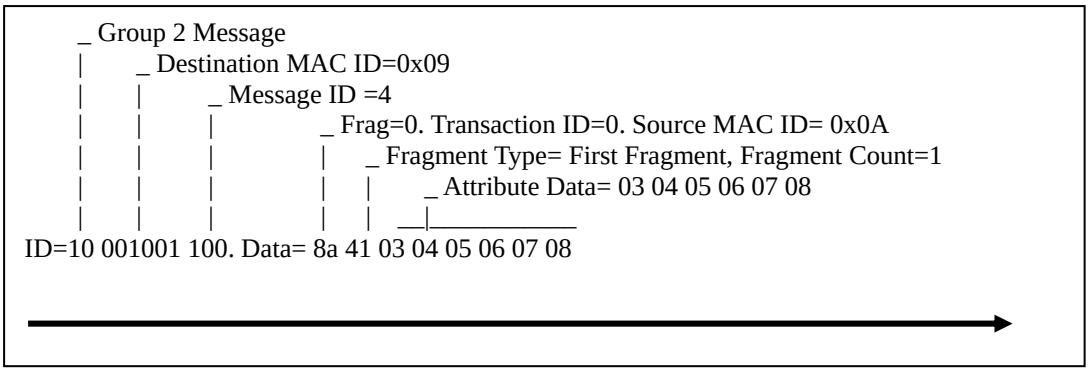
Data = 01020304050607080 90A.

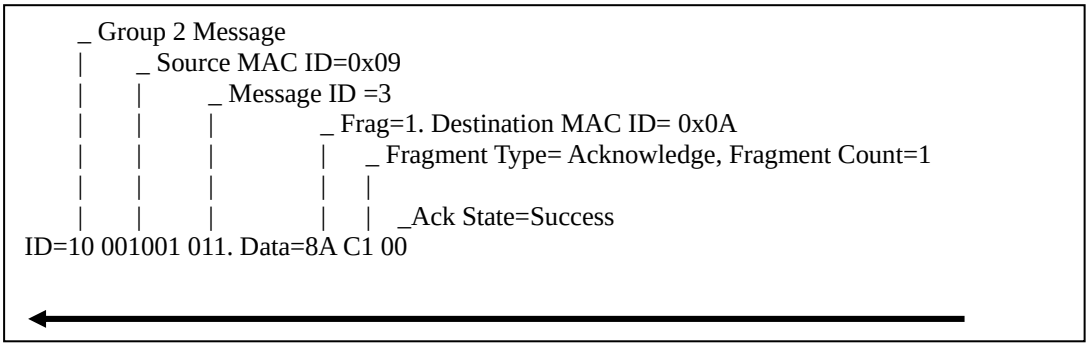
Master (MAC ID =0x0A) Slave (MAC ID =0x09)



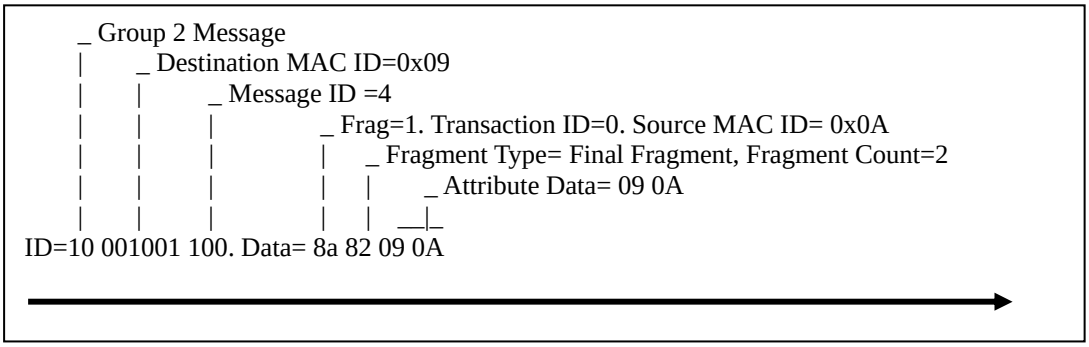


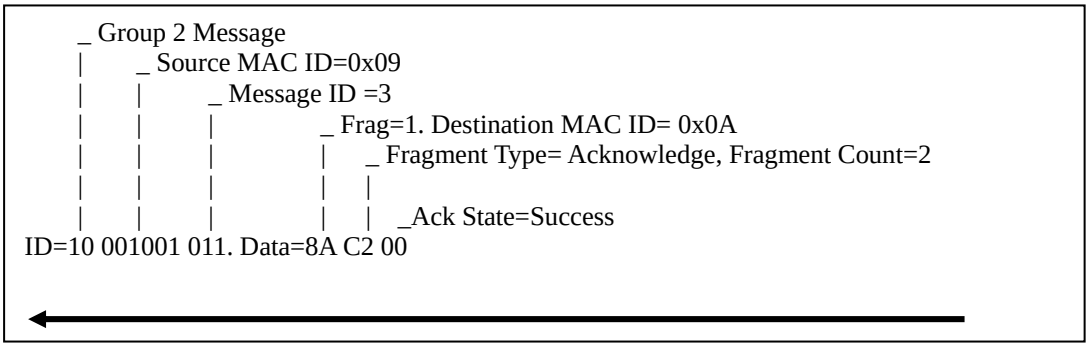
Master (MAC ID =0x0A) Slave (MAC ID =0x09)





Master (MAC ID =0x0A) Slave (MAC ID =0x09)





Chapter 8 Interpreting Analog Module Data

8.1 Analog Input Module Data transfer

Since the CAN-8x24 only supports the hex format, all of the AI channels need to transfer to the hex format when storing into this object. The transformation equation is shown below.

$$FloatValue = \left(\frac{HexValue - H\min}{H\max - H\min} \right) * \left(F\max - F\min \right) + F\min$$

The FloatValue is the result after transformation. The HexValue is the value which wants to be transferred. The Hmax and Hmin is the maximum and minimum values of the 2's complement hex range. The Fmax and Fmin is the maximum and minimum value of the float range. Users can find out the Hmax, Hmin, Fmax, and Fmin, values from in the appendix B. For example, the input range for the module i-7017 is set to -10V ~ +10V. According to the table in the appendix B, we can find out the range for the hex format as 0x8000 (+32767) ~ 0x7FFF (-32768). Therefore, if the value received from the AI channel of the i-7017 is 0x1234, the AI value with a float format can be calculated as follows.

$$\left(\frac{4660 - \left(-32768 \right)}{32767 - \left(-32768 \right)} \right) * \left(10V - \left(-10V \right) \right) + \left(-10V \right) \approx 1.422V$$

Incidentally, any AI value which is bigger than the maximum value of the input range will be set to the maximum value of the input range automatically. Furthermore, the AI value which is smaller than the minimum value of the input range is also set to the minimum value of the input range automatically.

8.2 Analog Output Module Data transfer

For the reason that the CAN-8x24 doesn't support the float format, user needs to transfer the AO value from the float format to the hex format. It is similar with the AI situation. The transformation equation is as follows.

$$HexValue = \left(\frac{FloatValue - F\min}{F\max - F\min} \right) * (H\max - H\min) + H\min$$

The HexValue is the result after transformation. The FloatValue is the value which wants to be transferred. The Fmax and Fmin are the maximum and minimum values of the float range. The Hmax and Hmin are the maximum and minimum values of the 2's complement hex range. User can find out the Fmax, Fmin, Hmax, and Hmin values from in the appendix B.

Chapter 9 Troubleshooting

The following section describes some typical problem scenarios and how to diagnose these problems.

9.1 Problem: Unable to Communicate with the Device

If you are unable to communicate with the device, please make sure that you have specified an appropriate MAC ID. The MSD means the most significant digit of the node address, and LSD represents the low significant digit of the node ID in the decimal format. Please check these settings and try again. Figure 9-1 shows the NA rotary switch.

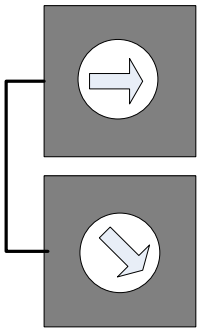


Figure 9-1 NA rotary switch

(MSB)

NA

9.2 Problem: All of the LEDs are off

It is necessary to provide device power to the device. Please check the connect pins on the CAN-8124/CAN-8224/CAN-8424. Figure 9-2 and 9-3 show the CAN-8124/CAN-8224/CAN-8424 power pin assignments.

Figure 9-2 CAN-8424 power pin assignments

Figure 9-3 CAN-8124 power pin assignments

9.3 Problem: MOD LED is Flashing

Check the setting values on the NA and DR rotary switches. If the values are invalid, the MOD led will be flashing.

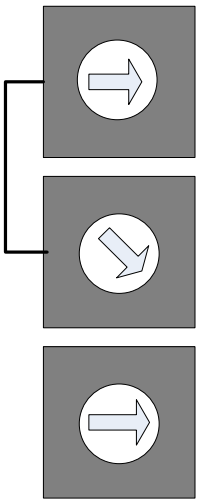


Figure 9-4 NA and DR rotary switches

(MSB)

9.4 Problem: NET LED is Solid when power-up

The device detects a duplicate node address. Please change the NA rotary switch to another value and try again.

NA

(LSB)

DR

9.5 Problem: How can I start to use the ICP DAS DeviceNet products?

ICP DAS provides many kinds of DeviceNet products. Users will need some basic knowledge for DeviceNet before they begin applying it. The easy way to use any DeviceNet products from ICP DAS is to study the manual. This manual provides the important DeviceNet knowledge for users. It is enough to use ICP DAS's DeviceNet products.

9.6 Problem: Why can I not to communicate any IO message with the device?

CAN-8x24 device provides the Assembly Object path as the default IO connection path when the distributors send it to you. Before you use the new device, please configure your device with the CAN Slave Utility. For more details relating to this, please refer to chapter 5. The critical point is about application and Assembly Objects because these objects are not vendor objects. Users must understand the relationship between the Application and Assembly Objects.

9.7 Problem: How to get IO data from CAN-8x24?

We support the i-8053, i-8064, i-8042 and i-8057 modules plugged-in CAN-8424 as is depicted in the following figure.

The CAN-8424 will take the information of these modules as the attribute data of application instance as in the following table.

Slot Address	Application Instance ID	Module name	DO Length(Byte)	AO Length(Byte)	DI Length(Byte)	AI Length(Byte)
0	0x01	8053	0	0	2	0
1	0x02	8064	1	0	0	0
2	0x03	8042	2	0	2	0
3	0x04	8057	2	0	0	0

Refer to the application object instances. The CAN-8424 will define the default assembly object instances according to the following table.

Assembly Object Instance ID (Hex)	Instance Type	Data Length(Byte)	Component modules
0x64	Output	DO: 5	i-8053/ i-8064/ i-8042
0x65	Input	DI: 4	i-8042/i-8057

If the default assembly instances are not adaptive, a user-defined assembly

object can be created. The system will show the unit of IO modules in the CAN Slave Utility software. Therefore, users can arrange the unit to the specific assembly instances. In analog modules, the unit of length is 2 bytes where 1 byte is in the digital modules.

Users can assign assembly instances as either input or output instances. However there must be DO or AO units in output instances. Plus there must be DI or AI units in input instances. For example:

Assembly Object Instance ID (Hex)	Instance Type	Data Length(Byte)	Component modules
0x64	Output	2	1st byte DO (1st byte of i-8053), 2nd byte DO (1st byte of i-8064)
0x65	Output	3	1st byte DO (2nd byte of i-8053), 2nd byte DO (1st byte of i-8042), 3rd byte DO (2nd byte of i-8042)
0x66	Input	4	1st byte DI (2nd byte of i-8042), 2nd byte DI (1st byte of i-8042), 3rd byte DI (2nd byte of i-8057) 4th byte DI (2nd byte of i-8057)

There are two ways to get IO data:

1. Users can apply the explicit connection to get the IO attribute of application instance. We can get the DI value for the i-8042 by getting the 22^{tth} attribute for the 3rd application instance. Also, to output the DO data on the i-8053 by setting the 20th attribute data for the first application instance.

2. Apply the IO connection to get/set the modules IO data. Plus the IO connection path can be either an application or assembly instances.

Appendix A: Analog I/O Transformation Table

In order to look up your required information, we have separated the transformation table into several parts according to the slot module names. They are given below.

i-87K Range Code	i-8K Range Code
i-87013/87015	i-8024
i-87017/87017R	i-8017H
i-87017W-A5	i-8080
i-87017RC	
i-87018/87018R/87018Z	
i-87019R	
i-87022	
i-87024	
i-87026	
i-87082	

i-87013/i-87015

RTD Type Definition

Type Code	RTD Type	Data Format	Max Value	Min Value
20 (Default)	Platinum 100 a = 0.00385 -100 to 100 degree Celsius	Engineer Unit	+100.00	-100.00
		% of FSR	+100.00	-100.00
		2's complement HEX	7FFF	8000
		Ohm	+138.50	+060.25
21	Platinum 100 a = 0.00385 0 to 100 degree Celsius	Engineer Unit	+100.00	+000.00
		% of FSR	+100.00	+000.00
		2's complement HEX	7FFF	0000
		Ohm	+138.50	+100.00
22	Platinum 100 a = 0.00385 0 to 200 degree Celsius	Engineer Unit	+200.00	+000.00
		% of FSR	+100.00	+000.00
		2's complement HEX	7FFF	0000
		Ohm	+175.84	+100.00
23	Platinum 100 a = 0.00385 0 to 600 degree Celsius	Engineer Unit	+600.00	+000.00
		% of FSR	+100.00	+000.00
		2's complement HEX	7FFF	0000
		Ohm	+313.59	+100.00
24	Platinum 100 a = 0.003916 -100 to 100 degree Celsius	Engineer Unit	+100.00	-100.00
		% of FSR	+100.00	-100.00
		2's complement HEX	7FFF	8000
		Ohm	+139.16	+059.58
25	Platinum 100 a = 0.003916 0 to 100 degree Celsius	Engineer Unit	+100.00	+000.00
		% of FSR	+100.00	+000.00
		2's complement HEX	7FFF	0000
		Ohm	+139.16	+100.00
26	Platinum 100 a = 0.003916	Engineer Unit	+200.00	+000.00
		% of FSR	+100.00	+000.00

	0 to 200 degree Celsius	2's complement HEX	7FFF	0000
		Ohm	+177.13	+100.00
27	Platinum 100 a = 0.003916 0 to 600 degree Celsius	Engineer Unit	+600.00	+000.00
		% of FSR	+100.00	+000.00
		2's complement HEX	7FFF	0000
		Ohm	+317.28	+100.00
28	Nickel 120 -80 to 100 degree Celsius	Engineer Unit	+100.00	-080.00
		% of FSR	+100.00	-080.00
		2's complement HEX	7FFF	999A
		Ohm	+200.64	+120.60
29	Nickel 120 0 to 100 degree Celsius	Engineer Unit	+100.00	+000.00
		% of FSR	+100.00	+000.00
		2's complement HEX	7FFF	0000
		Ohm	+200.64	+120.60
2A	Platinum 1000 a = 0.00385 -200 to 600 degree Celsius	Engineer Unit ()	+600.00	-200.00
		% of FSR	+100.00	-033.33
		2's complement HEX	7FFF	D556
		Ohm	+3137.1	+0185.2
2B[*1]	Cu 100 a = 0.00421 -20 to 150 degree Celsius	Engineer Unit ()	+150.00	-020.00
		% of FSR	+100.00	-013.33
		2's complement HEX	7FFF	EEEF
		Ohm	+163.17	+091.56
2C[*1]	Cu 100 a = 0.00421 0 to 200 degree Celsius	Engineer Unit ()	+200.00	-000.00
		% of FSR	+100.00	-000.00
		2's complement HEX	7FFF	0000
		Ohm	+167.75	+090.34
2D[*1]	Cu 1000 a = 0.00421 -20 to 150 degree Celsius	Engineer Unit ()	+150.00	-020.00
		% of FSR	+100.00	-013.33
		2's complement HEX	7FFF	EEEF
		Ohm	+1631.7	+0915.6

2E*2	Pt 100 a = 0.00385 -200 to +200 degree Celsius	Engineer Unit ()	+200.00	-200.00
		% of FSR	+100.00	-100.00
		2's complement HEX	7FFF	8000
		Ohm	+175.84	+018.49
2F*2	Pt 100 a = 0.003916 -200 to +200 degree Celsius	Engineer Unit ()	+200.00	-200.00
		% of FSR	+100.00	-100.00
		2's complement HEX	7FFF	8000
		Ohm	+177.14	+017.14
80*2	Pt 100 a = 0.00385 -200 to +600 degree Celsius	Engineer Unit ()	+600.00	-200.00
		% of FSR	+100.00	-033.33
		2's complement HEX	7FFF	D556
		Ohm	+313.59	+018.49
81*2	Pt 100 a = 0.003916 -200 to +600 degree Celsius	Engineer Unit ()	+600.00	-200.00
		% of FSR	+100.00	-033.33
		2's complement HEX	7FFF	D556
		Ohm	+317.28	+017.14

Note :
* 1: Type 2B, 2C and 2D are only available with i-87015.
* 2: Type 2E, 2F, 80 and 81 are only available with the i-87015 firmware version A1.10 and later, i-87013 firmware version B1.3 and later.

Appendix B

RTD Input Type Over/Under Range Response		
For i-87013		
Data Format	Over Range	Under Range
Engineer Unit	+9999	-0000
% of FSR	+9999	-0000
2's Complement HEX	7FFF	8000
For i-87015		
Data Format	Over Range	Under Range
Engineer Unit	+9999.9	-9999.9
% of FSR	+999.99	-999.99
2's Complement HEX	7FFF	8000

Appendix B

i-87017/87017R

Type 08 to 0D Definition					
Type Code	Input Range	Data Format	Full Scale	Zero	Negative Full Scale
08 (Default)	-10V to +10V	Engineer Unit	+10.000	+00.000	-10.000
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000
09	-5V to +5V	Engineer Unit	+5.0000	+0.0000	-5.0000
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000
0A	-1V to +1V	Engineer Unit	+1.0000	+0.0000	-1.0000
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000
0B	-500mV to +500mV	Engineer Unit	+500.00	+000.00	-500.00
		% of FSR	+100.00	+000.00	-100.00

		2's Complement HEX	7FFF	0000	8000
0C	-150mV to +150mV	Engineer Unit	+150.00	+000.00	-150.00
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000
0D*1	-20mA to +20mA	Engineer Unit	+20.000	+00.0000	-20.000
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000

Note:
*1: When i-87017 and i-87017R are connecting to a current source set to OD type code, an optional external 125 Ohms resistor is required.

Appendix B

i-87017W-A5

Type 1B to 1C Definition

Type Code	Input Range	Data Format	Full Scale	Zero	Negative Full Scale
1B (Default)	-150V to +150V	Engineer Unit	+150.00	+000.0000	-150.00
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000
1C	-50V to +50V	Engineer Unit	+50.000	+00.0000	-50.000
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000

Appendix B

i-87017RC

Type 07 to 1A Definition

Type Code	Input Range	Data Format	Full Scale	Zero	Negative Full Scale
07 (Default)	-4mA to +20mA	Engineer Unit	+04.000	+00.0000	+20.000
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000
0D	-20mA to +20mA	Engineer Unit	+20.000	+00.0000	-20.000
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000
1A	+0A to +20mA	Engineer Unit	+00.000	+00.0000	+20.000
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000

Note:

i-87017RC has built-in 125 Ohms resistors for each channel. When connecting to a current source, no add any external resistors required.

Appendix B

i-87018/87018R/87018Z

Type 00 to 16 and Thermocouple Type Definition

Type Code	Input Range	Data Format	Full Scale	Zero	Negative Full Scale
00	-15mV to +15mV	Engineer Unit	+15.000	+00.000	-15.000
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000
01	-50mV to +50mV	Engineer Unit	+50.000	+00.000	-50.000
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000
02	-100mV to +100mV	Engineer Unit	+100.00	+000.00	-100.00
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000
03	-500mV to +500mV	Engineer Unit	+500.00	+000.00	-500.00
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000
04	-1V to +1V	Engineer Unit	+1.0000	+0.0000	-1.0000
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000
05 (Default)	-2.5V to +2.5V	Engineer Unit	+2.5000	+0.0000	-2.5000
		% of FSR	+100.00	+000.00	-100.00
		2's Complement	7FFF	0000	8000

		HEX			
06	-20mA to +20mA with 125 ohms resistor	Engineer Unit	+20.000	+00.0000	-20.000
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000
0E	J Type	Engineer Unit	+760.00	+00.000	-210.00
		% of FSR	+100.00	+000.00	-027.63
		2's Complement HEX	7FFF	0000	DCA2
0F	K Type	Engineer Unit	+1372.0	+00.000	-0270.0
		% of FSR	+100.00	+000.00	-019.68
		2's Complement HEX	7FFF	0000	E6D0
10	T Type	Engineer Unit	+400.00	+000.00	-270.00
		% of FSR	+100.00	+000.00	-067.50
		2's Complement HEX	7FFF	0000	A99A
11	E Type	Engineer Unit	+1000.0	+000.00	-0270.0
		% of FSR	+100.00	+000.00	-027.00
		2's Complement HEX	7FFF	0000	DD71
12	R Type	Engineer Unit	+1768.0	+00.000	+0000.0
		% of FSR	+100.00	+000.00	+000.00
		2's Complement HEX	7FFF	0000	0000
13	S Type	Engineer Unit	+1768.0	+00.000	+0000.0

		% of FSR	+100.00	+000.00	+000.00
		2's Complement HEX	7FFF	0000	0000
14	B Type	Engineer Unit	+1820.0	+000.00	+0000.0
		% of FSR	+100.00	+000.00	+000.00
		2's Complement HEX	7FFF	0000	0000
15	N Type	Engineer Unit	+1300.0	+000.00	-0270.0
		% of FSR	+100.00	+000.00	-020.77
		2's Complement HEX	7FFF	0000	E56B
16	C Type	Engineer Unit	+2320.0	+00.000	+0000.0
		% of FSR	+100.00	+000.00	+000.00
		2's Complement HEX	7FFF	0000	0000
17	L Type	Engineer Unit	+800.00	+00.000	-200.00
		% of FSR	+100.00	+000.00	-025.00
		2's Complement HEX	7FFF	0000	E000
18	M Type	Engineer Unit	+100.00	+000.00	-200.00
		% of FSR	+050.00	+000.00	-100.00
		2's Complement HEX	4000	0000	8000
19	L Type DIN43710	Engineer Unit	+900.00	+000.00	-200.00
		% of FSR	+100.00	+000.00	-022.22
		2's Complement	7FFF	0000	E38F

		HEX			

Appendix B

i-87019R

Type 00 to 19 Definition

Type Code	Input Range	Data Format	Full Scale	Zero	Negative Full Scale
00	-15mV to +15mV	Engineer Unit	+15.000	+00.000	-15.000
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000
01	-50mV to +50mV	Engineer Unit	+50.000	+00.000	-50.000
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000
02	-100mV to +100mV	Engineer Unit	+100.00	+000.00	-100.00
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000
03	-500mV to +500mV	Engineer Unit	+500.00	+000.00	-500.00
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000
04	-1V to +1V	Engineer Unit	+1.0000	+0.0000	-1.0000
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000

05	-2.5V to +2.5V	Engineer Unit	+2.5000	+0.0000	-2.5000
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000
06	-20mA to +20mA with 125 ohms resistor	Engineer Unit	+20.000	+00.0000	-20.000
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000
08 (Default)	-10V to +10V	Engineer Unit	+10.000	+00.000	-10.000
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000
09	-5V to +5V	Engineer Unit	+5.0000	+0.0000	-5.0000
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000
0A	-1V to +1V	Engineer Unit	+1.0000	+0.0000	-1.0000
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000
0B	-500mV to +500mV	Engineer Unit	+500.00	+000.00	-500.00
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000
0C	-150mV to +150mV	Engineer Unit	+150.00	+000.00	-150.00
		% of FSR	+100.00	+000.00	-100.00

		2's Complement HEX	7FFF	0000	8000
0D	-20mA to +20mA with 125 ohms resistor	Engineer Unit	+20.000	+00.0000	-20.000
		% of FSR	+100.00	+000.00	-100.00
		2's Complement HEX	7FFF	0000	8000
0E	J Type	Engineer Unit	+760.00	+00.000	-210.00
		% of FSR	+100.00	+000.00	-027.63
		2's Complement HEX	7FFF	0000	DCA2
0F	K Type	Engineer Unit	+1372.0	+00.000	-0270.0
		% of FSR	+100.00	+000.00	-019.68
		2's Complement HEX	7FFF	0000	E6D0
10	T Type	Engineer Unit	+400.00	+000.00	-270.00
		% of FSR	+100.00	+000.00	-067.50
		2's Complement HEX	7FFF	0000	A99A
11	E Type	Engineer Unit	+1000.0	+000.00	-0270.0
		% of FSR	+100.00	+000.00	-027.00
		2's Complement HEX	7FFF	0000	DD71
12	R Type	Engineer Unit	+1768.0	+00.000	+0000.0
		% of FSR	+100.00	+000.00	+000.00
		2's Complement HEX	7FFF	0000	0000
13	S Type	Engineer Unit	+1768.0	+00.000	+0000.0

		% of FSR	+100.00	+000.00	+000.00
		2's Complement HEX	7FFF	0000	0000
14	B Type	Engineer Unit	+1820.0	+000.00	+0000.0
		% of FSR	+100.00	+000.00	+000.00
		2's Complement HEX	7FFF	0000	0000
15	N Type	Engineer Unit	+1300.0	+000.00	-0270.0
		% of FSR	+100.00	+000.00	-020.77
		2's Complement HEX	7FFF	0000	E56B
16	C Type	Engineer Unit	+2320.0	+00.000	+0000.0
		% of FSR	+100.00	+000.00	+000.00
		2's Complement HEX	7FFF	0000	0000
17	L Type	Engineer Unit	+800.00	+00.000	-200.00
		% of FSR	+100.00	+000.00	-025.00
		2's Complement HEX	7FFF	0000	E000
18	M Type	Engineer Unit	+100.00	+000.00	-200.00
		% of FSR	+050.00	+000.00	-100.00
		2's Complement HEX	4000	0000	8000
19	L Type DIN43710	Engineer Unit	+900.00	+000.00	-200.00
		% of FSR	+100.00	+000.00	-022.22
		2's Complement	7FFF	0000	E38F

		HEX			

Appendix B

i-87022

Type Code	Output Range	Data Format	Max Value	Min Value
0	0 to 20mA	Engineer Unit	20.000	00.000
		% of FSR	+100.00	+000.00
		Hexadecimal	FFF	000
1	4 to 20mA	Engineer Unit	20.000	04.000
		% of FSR	+100.00	+000.00
		Hexadecimal	FFF	000
2 (Default)	0 to 10V	Engineer Unit	10.000	00.000
		% of FSR	+100.00	+000.00
		Hexadecimal	FFF	000

Appendix B

i-87024

Type Code	Output Range	Data Format	Max Value	Min Value
30	0 to 20mA	Engineer Unit	+20.000	+00.000
		Hexadecimal	7FFF	0000
31	4 to 20mA	Engineer Unit	+20.000	+04.000
		Hexadecimal	7FFF	0000
32	0 to 10V	Engineer Unit	+10.000	+00.000
		Hexadecimal	7FFF	0000
33 (Default)	-10 to 10V	Engineer Unit	+10.000	-10.000
		Hexadecimal	7FFF	8000
34	0 to 5V	Engineer Unit	+05.000	+00.000
		Hexadecimal	7FFF	0000
35	-5 to 5V	Engineer Unit	+05.000	-05.000
		Hexadecimal	7FFF	8000

Appendix B

i-87026

Type Code	Output Range	Data Format	Max Value	Min Value
0	0 to 20mA	Engineer Unit	20.000	00.000
		% of FSR	+100.00	+000.00
		Hexadecimal	FFFF	0000
1	4 to 20mA	Engineer Unit	20.000	04.000
		% of FSR	+100.00	+000.00
		Hexadecimal	FFFF	0000
2 (Default)	0 to 10V	Engineer Unit	10.000	00.000
		% of FSR	+100.00	+000.00
		Hexadecimal	FFFF	0000

Appendix B

i-87082

Type Code	Description
50 (Default)	Counter Mode
51	Frequency Measurement

Appendix B

i-8024

Type Code	Output Range	Data Format	Max Value	Min Value
0 (Default)	-10 to +10V	Engineer Unit	10.000	-10.000
		Hexadecimal	7FFF	8000
1	0 to +20mA	Engineer Unit	20.000	00.000
		Hexadecimal	7FFF	0000

Appendix B

i-8017H

Type Code	Input Range	Data Format	Max Value	Min Value
0 (Default)	-10 to +10V	Engineer Unit	+10.000	-10.000
		Hexadecimal	1FFF	2000
1	-5 to +5V	Engineer Unit	+5.000	-5.000
		Hexadecimal	1FFF	2000
2	-2.5 to +2.5V	Engineer Unit	+2.500	-2.500
		Hexadecimal	1FFF	2000
3	-1.25 to +1.25V	Engineer Unit	+1.250	-1.250
		Hexadecimal	1FFF	2000
4	-20.0 to +20.0mA	Engineer Unit	+20.0	-20.0
		Hexadecimal	1FFF	2000

Appendix B

i-8080

Type Code	Operation Mode
0	Dir/Pulse counting mode
1	Up/Down counting mode
2	Frequency mode
3 (Default)	Up counting mode

Appendix B