DeviceNet Slave Device

CAN-2000D Series

Communication User's Manual

Warranty

Without contrived damage, all products manufactured by ICP DAS are warranted in one year from the date of delivery to customers.

Warning

ICP DAS revises the manual at any time without notice. However, no responsibility is taken by ICP DAS unless infringement act imperils to patents of the third parties.

Copyright

Copyright © 2010 is reserved by ICP DAS.

Trademark

The brand name ICP DAS as a trademark is registered, and can be used by other authorized companies.

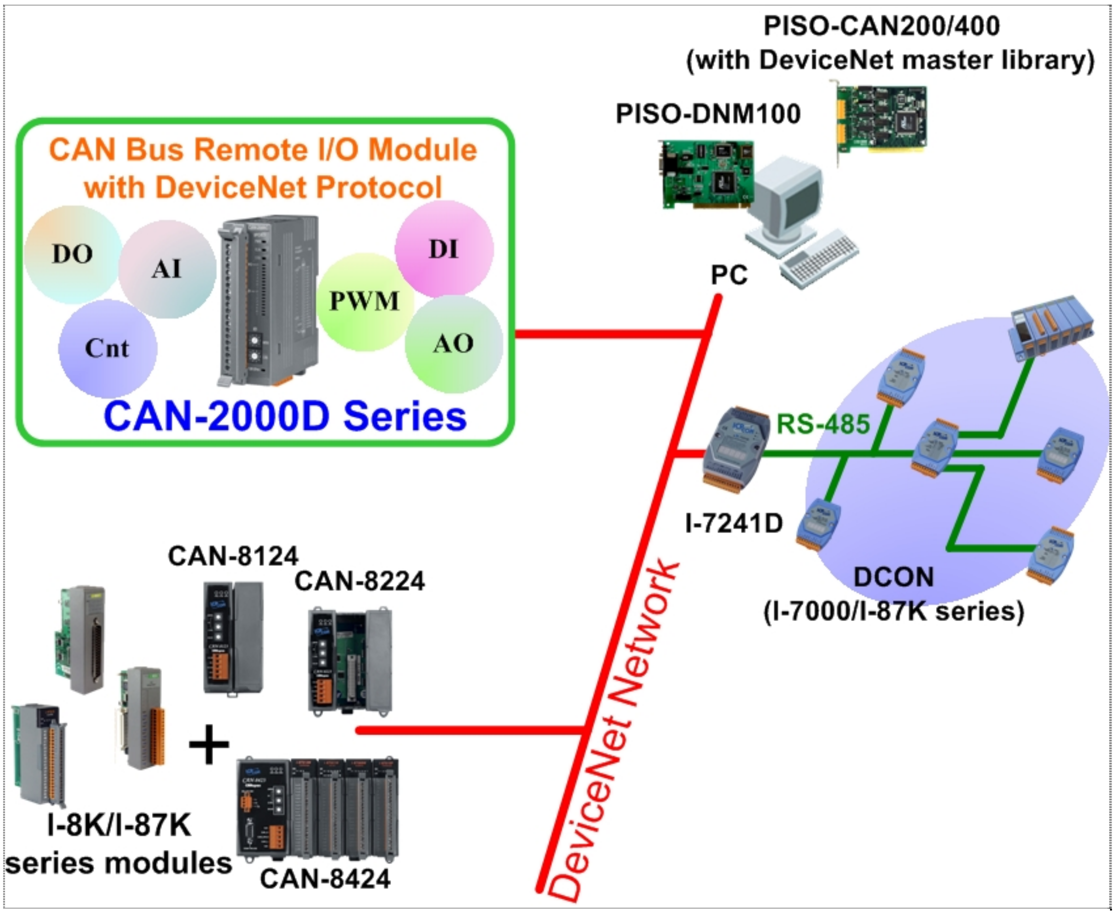
Contents

1 Introduction

1.1 Overview

DeviceNet is a CAN-based intelligent communication protocol and has been developed as a standard embedded network with a high flexible configuration. It provides standard communication methods for transmitting real-time data by “I/O Message” and configuration data by “Explicit Message”. Nowadays, DeviceNet is widely used in many applications, such as medical equipment, off-road vehicles, maritime electronics, public transportation, factory automation and so on.

The CAN-2000D series is suit to apply in the remote control systems. It is specially designed to the DeviceNet slave module. All of the CAN-2000D series follow the DeviceNet specification Volume I/II, Release 2.0, and supply standard DeviceNet communication methods, such as Polling, Bit Strobe I/O Message, Heartbeat, Shutdown Message and so forth. The general application architecture of the CAN-2000D series is as follows.



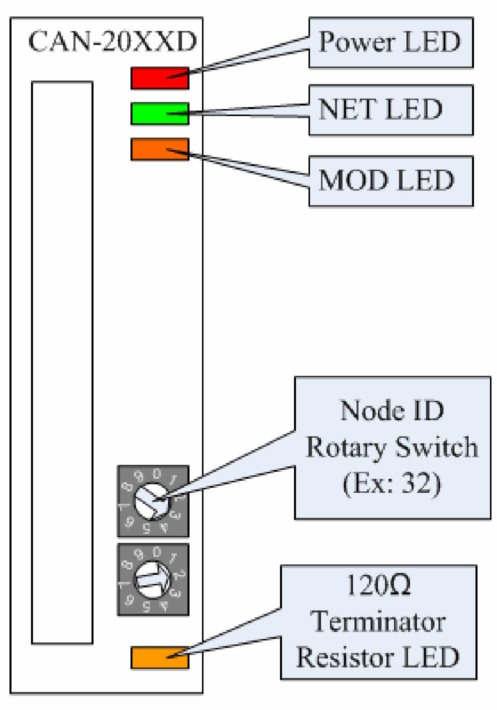
1.2 CAN-2000D General Hardware Specifications

- Built-in Watchdog
- PWR LED, NET LED, and MOD LED
- Terminator resister LED
- 120Ω terminator resister selected by DIP-switch
- CAN bus interface: ISO 11898-2, 5-pin screw terminal with removable terminal block.
- Unregulated from +10V_{DC} ~ +30V_{DC}.
- Operating Temperature:-25°C ~ +75°C
- Storage Temperature:-30°C ~ +80°C
- Humidity:10% ~ 90% RH, non-condensing

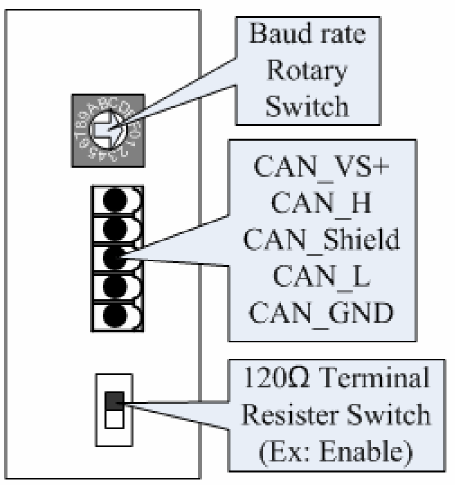
1.3 CAN-2000D Common Features

- DeviceNet general I/O slave devices.
- Comply with DeviceNet specification Volume I, Release 2.0 & Volume II, Release 2.0
- Group 2 Only Server (non UCMM-capable)
- Support Predefined Master/Slave Connection Set
- Connection supported:
  - 1 connection for Explicit Messaging
  - 1 connection for Polled I/O
  - 1 connection for Bit-Strobe I/O connection
- Support DeviceNet heartbeat and shutdown messages
- Provide EDS file for standard DeviceNet master interface.
- Support “Save” and “Load” command, can save I/O setting or restore I/O default setting.
- MAC ID selected by rotary switch from 0~63
- Baud rate selected by rotary switch from 0~2 (125 kbps, 250 kbps, 500 kbps).
- Status LED: PWR, NET, and MOD LED indicators

2 Hardware Specification



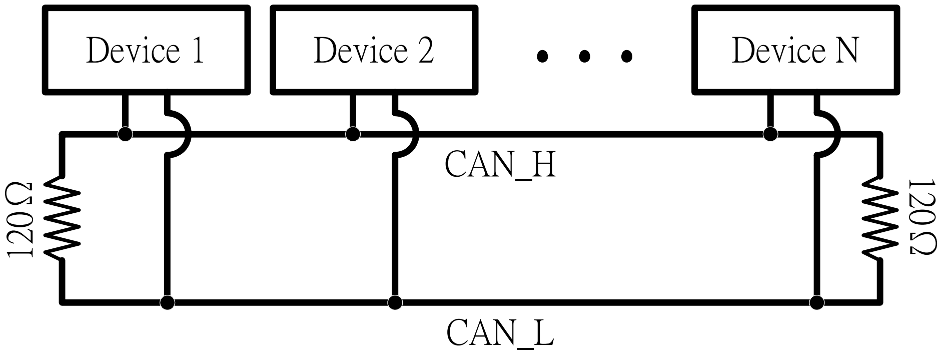
(Top View)



(Bottom View)

2.1 Wire Connection

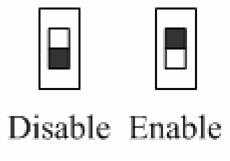
In order to minimize the reflection on the CAN bus line, the CAN bus line has to be terminated at both ends by two terminator resistors as shown in the following. According to the ISO 11898-2 spec, each terminator resistor is 120Ω (or other between 108Ω~132Ω). The length related resistance has to reach 70 mΩ/m. At this circumstance, users would better check the resistances of the CAN bus before installing a new CAN network.



Moreover, to minimize the voltage drop, value of the terminator resistor must be higher than the one defined in the ISO 11898-2. The following table is for users' reference.

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

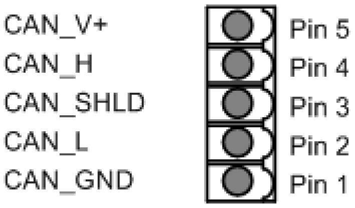
Inside the CAN-2000D series module, the 120Ω terminator resistor is supplied as a standard accessory. You can configure the SW1 to decide if the terminator resistor. The following figure indicates two conditions, "Disable (Up)" and "Enable (Down)". If you set the switch as the figure marked with "Enable", the terminator resistor is active. Meanwhile, the terminator resistor LED will on. Please refer to the figure at the start of section 2 for the position of terminator resistor LED and the switch of the terminator resistor.



The bus length determines the CAN bus baud rate. In the following the table provides users a relationship between the baud rate and the bus length.

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

The pin descriptions of the CAN bus connectors on the CAN-2000D are shown below.



Pin No.	Signal	Description
1	CAN_GND	Ground (0V)
2	CAN_L	CAN_L bus line (dominant low)
3	CAN_SHLD	Optional CAN Shield
4	CAN_H	CAN_H bus line (dominant high)
5	CAN_V+	CAN external positive supply

2.2 Power LED

The CAN-2000D series products need 10 to 30 V_{DC} power supplies. Under a normal connection, a good power supply and a correct voltage selection, as the unit is turned on, the PWR LED will light up in red. If it can't work, please check with local agents or resellers for more help.

2.3 DeviceNet Status LED

Each CAN-2000D module has two LED indicators. One is the MOD LED (orange) and another is the NET LED (green). The MOD LED and the NET LED information are presented in the DeviceNet specifications. When the DeviceNet communication carries out, these indicators will glitter. The following descriptions shows the meanings of the glittering signal as these indicators are being triggered.

2.3.1 NET LED

The NET LED indicates the current status of the DeviceNet communication link.

condition	status	indicates
Init Off	Off line	Device is not online
Off	Connection timeout	I/O connection timeout
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

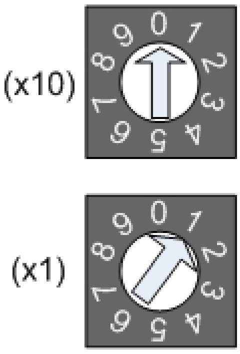
2.3.2 MOD LED

This LED provides the devices status. It indicates whether or not the device is operating properly.

condition	status	indicates
Off	Normal	
Solid	Critical fault	Device has unrecoverable fault.
Flashing	Non_critical fault	Device has recoverable fault to recover. If users want to fix the problem, reconfiguring device’s MAC ID or resetting device may work.

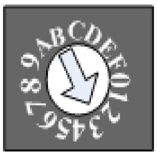
2.4 The Node ID & the Baud rate Rotary Switch

The rotary switches for node ID configure the node ID of CAN-2000D module. These two switches are for the tens digit and the units digit of node ID. The node ID value of this demo picture is 1.



Node ID rotary switch

The rotary switch for baud rate handles the CAN baud rate of the CAN-2000D module. The relationship between the rotary switch value and the practical baud rate is presented in the following table.



Baud rate rotary switch

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

2.5 Module Support

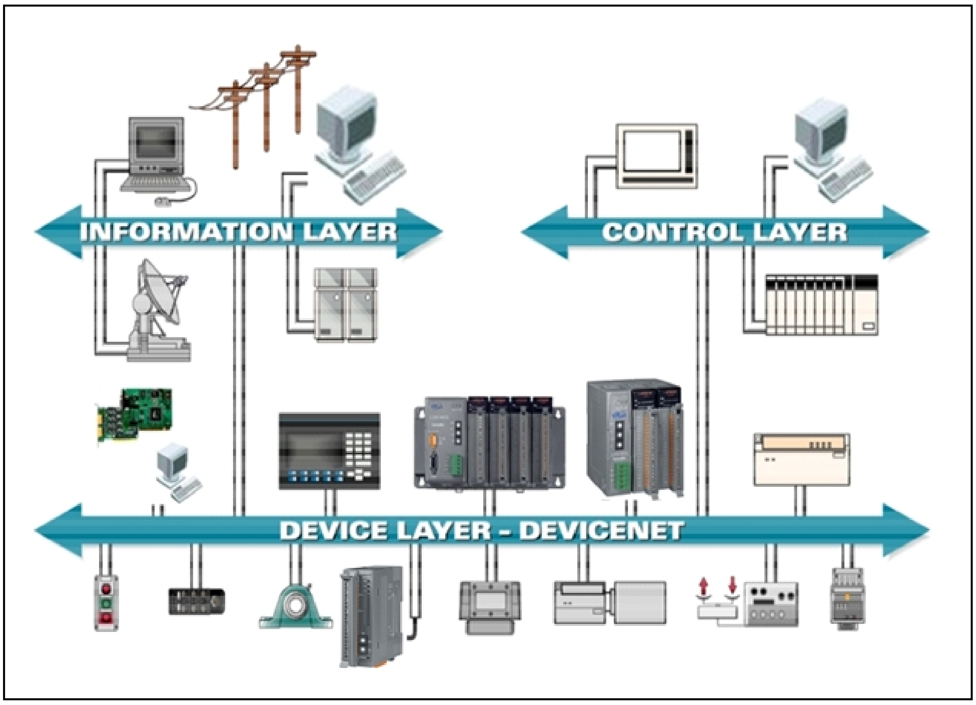
The CAN-2000D series modules include many kinds of DI, DO, AI and AO types series modules. Please refer to the web to get more detail information: http://www.icpdas.com/products/Remote_IO/can_bus/can-2000.htm

3 DeviceNet Application

The DeviceNet is a kind of network protocols evolving from the CAN bus, used on car control system in early days, and has been greatly used in various applications, such as vehicles, industrial machines, building automation, medical devices, maritime applications, restaurant appliances, laboratory equipment & research.

3.1 DeviceNet Introduction

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. The following figure shows the DeviceNet layer in the control and information layers.



The 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 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 the following table. 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.

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

DeviceNet's Use of the CAN Identifier Field

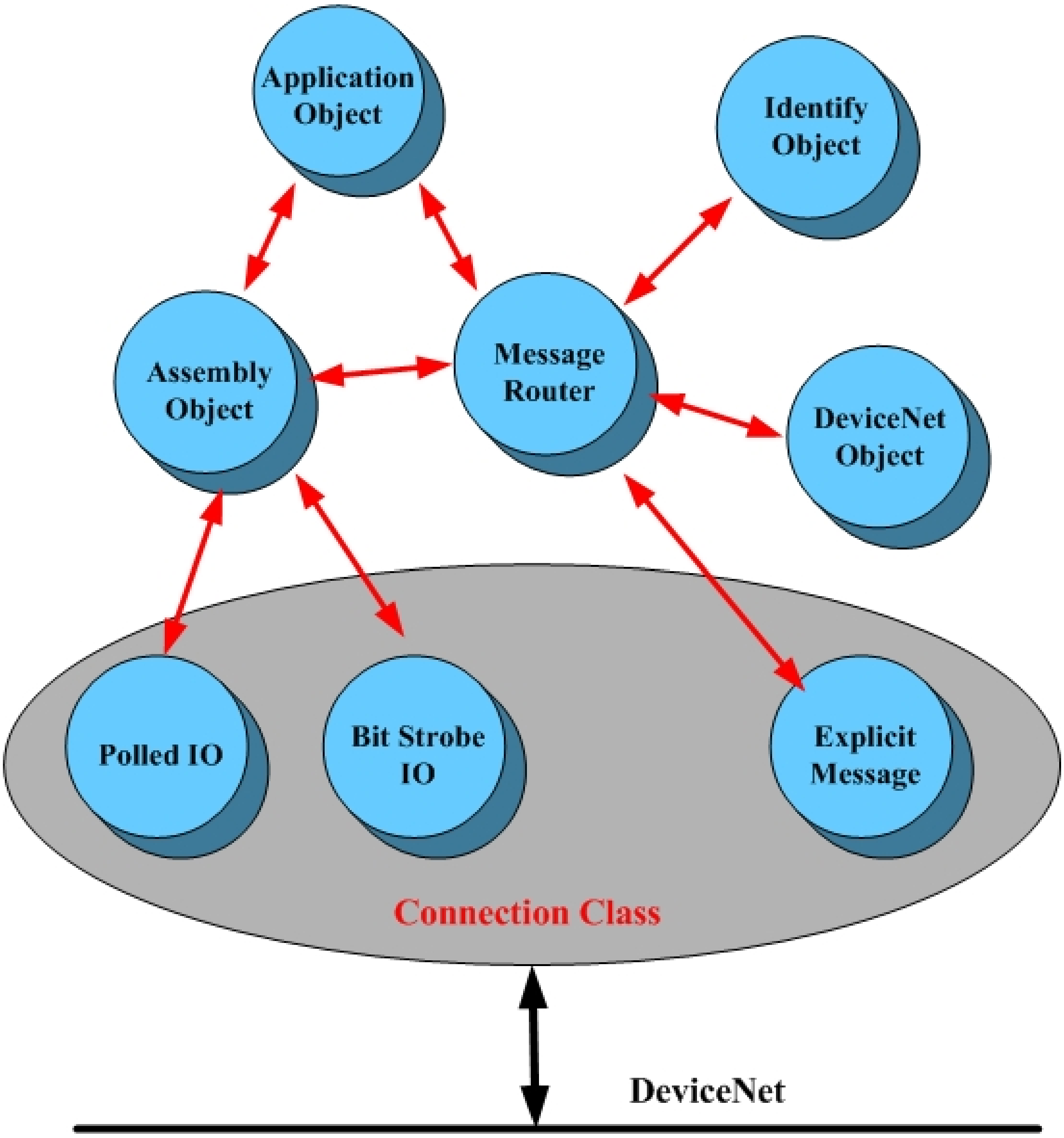
The CAN-2000D provides the Predefined Master/slave Connection Set for users to establish connections. The Predefined Master/Slave Connection Set is a set of Connections that facilitate communications typically seen in a Master/Slave relationship. Many of the steps involved in the creation and configuration 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 the following table. The table defines the Identifiers that are to be used with all connection based message 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 3.1 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. The following figure depicts the object model for CAN-2000D (Group 2 Only Server).



Object models of CAN-2000D

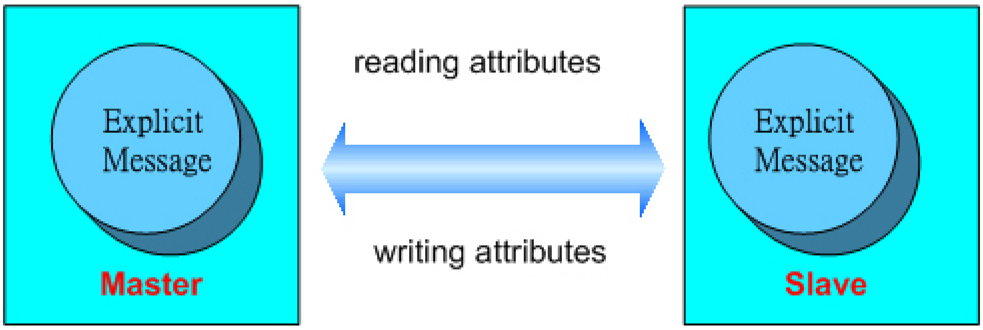
3.2 Predefined Master Slave Connection Set

The CAN-2000D provides "Predefined Master Slave Connection Set" device. Users must understand these connection set to know how to operate the device. The following section explains what the "Predefined Master Slave Connection Set" is.

With 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 the 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 several different I/O Connections which include a Bit Strobe Command/Response, Poll Command/Response. The following types of messages are processed by a DeviceNet slave.

3.2.1 Explicit Messages

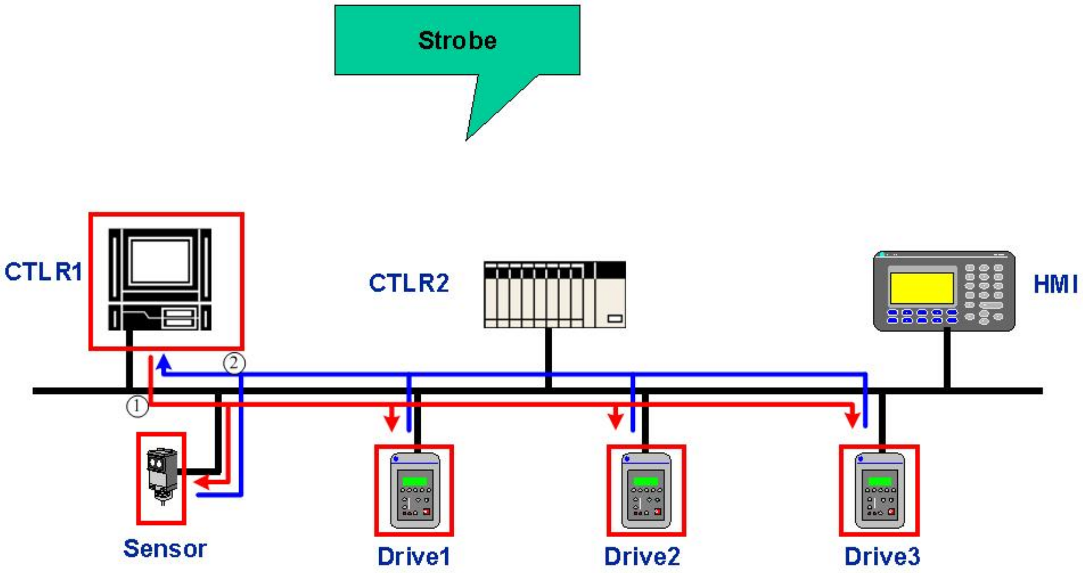
Explicit Request Messages are used to perform operations such as reading and writing attributes. Explicit response Messages indicate the results of the slaves answer to attempt to service an Explicit Request massage. Within a Slave Explicit Request and Response messages are received/transmitted by a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



The architecture of Explicit message

3.2.2 Bit Strobe I/O Messages

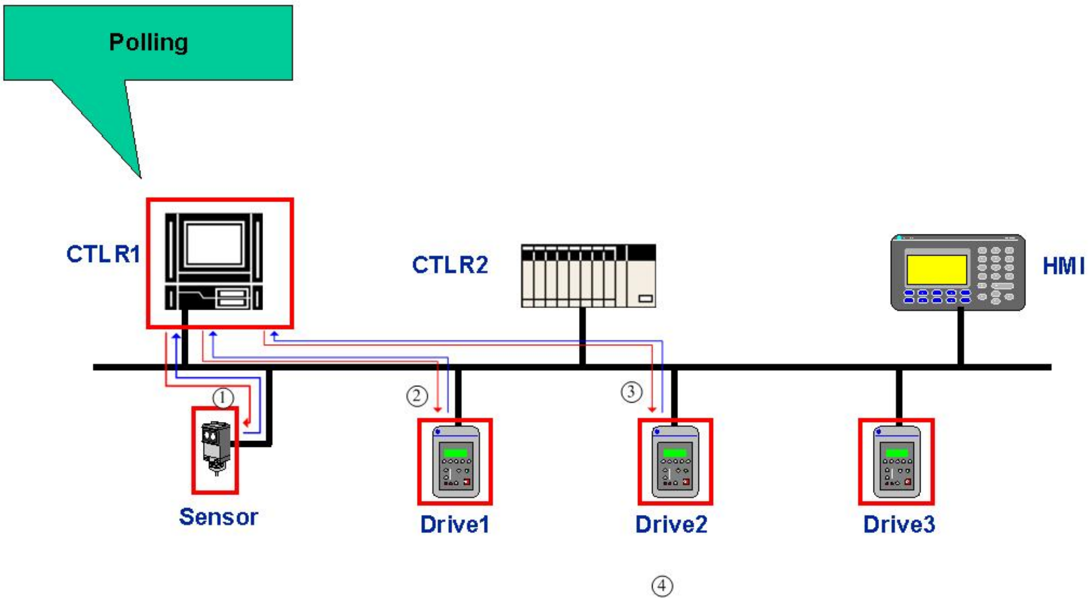
The Bit-Strobe Command is an I/O message that is transmitted by the Master. A Bit-Strobe Command has multicast capabilities. Multiple Slaves can receive and react to the same Bit Strobe Command. The Bit-Strobe response is an I/O message that a Slave transmits back to the Master when the Bit-Strobe Command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



The architecture of Bit Strobe I/O message

3.2.3 Poll I/O Messages

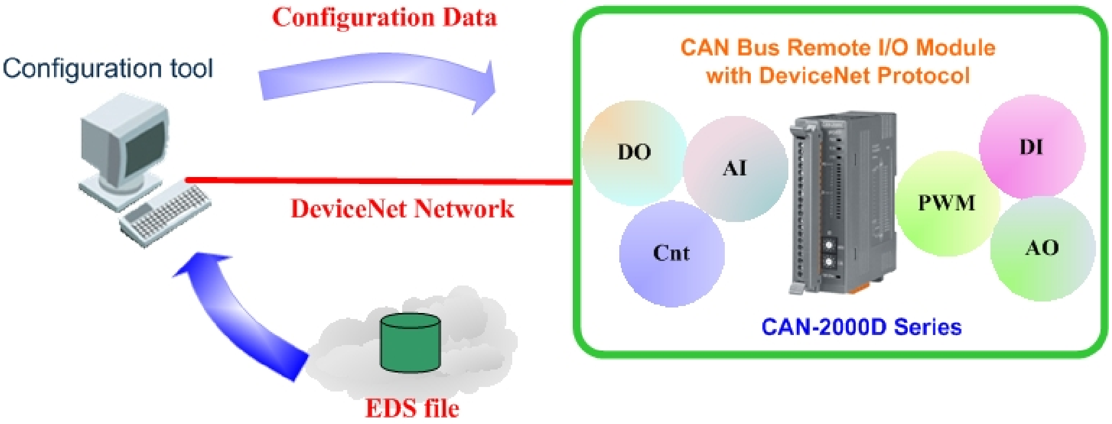
The Poll Command is a command that is transmitted by the Master. A Poll Command is directed towards a single, specific Slave (point-to-point connection). A Master must transmit a separate Poll command message for each one of its Slaves that will be polled. The Poll-Response is an I/O message that the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



The architecture of Poll I/O 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 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 CAN gateway utility software for users to create the suitable EDS file. The EDS file system architecture is as following figure.



Architecture of the EDS file

EDS provides information about the device's configuration data in terms of 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.

4 DeviceNet Profile Area

This section documents the detailed functions for each object class that is implemented in the DeviceNet system.

4.1 DeviceNet Statement of Compliance

General Device Data

Device Information	Description
Version Description of DeviceNet Specification	Volume I, Release 2.0 & Volume II, Release 2.0
Vendor Name	ICP DAS

DeviceNet Physical Conformance Data

Item	Description
DeviceNet status LED Support	Yes
MAC ID Setting	Switch (0 ~ 63)
Default MAC ID	1
Communication Baud Rate Setting	Switch (125, 250, 500 kbps)
Default Baud Rate	125 kbps
Predefined Master/Slave Connection Set	Group 2 Only Server

4.2 Identity Object (Class ID: 0x01)

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

Class Attribute (Instance ID=0)

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 (Instance ID=1)

Attribute ID	Description	Method	DeviceNet Data Type	Value
1	Vendor	Get	UINT	803
2	Product type	Get	UINT	0x00
3	Product code	Get	UINT	-
4	Major. Minor of firmware version	Get	Struct of USINT USINT	-
5	Status	Get	WORD	-
6	Serial number	Get	UDINT	-
7	Product name	Get	Short_String	CAN-20xxD
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

Note: Use the Instance Service 0x05 will reboot the device.

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 (Instance ID=0)

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 (Instance ID=1)

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	-
4	Bus-off counter	Get/Set	USINT	-
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 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.

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

Class attribute (Instance ID=0)

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

4.4.1 Explicit connection

Instance attribute (Instance ID=1)

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 3.1
0x05	consumed_connection_id	UINT	Get	Table 3.1
0x06	initial_comm_characteristics	BYTE	Get	0x21
0x07	produced_connection_size	UINT	Get	-
0x08	consumed_connection_size	UINT	Get	-
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

4.4.2 Poll I/O connection

Instance attribute (Instance ID=2)

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 3.1
0x05	consumed_connection_id	UINT	Get	Table 3.1
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

4.4.3 Bit Strobe I/O Connection

Instance attribute (Instance ID=3)

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 3.1
0x05	consumed_connection_id	UINT	Get	Table 3.1
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

5 DeviceNet Communication Set

5.1 Communication Set Introduction

The CAN-2000D series devices are “Group 2 Only Server” devices, and support the “Predefined Master/slave Connection Set”. To communicate with the devices, the process about how to establish a connection is important. The CAN Identifier Fields associated with the “Predefined Master/Slave Connection Set” for the DeviceNet are shown in the below table. The table defines the Identifiers that are used with all connection based messaging involved in the “Predefined Master/Slave Connection Set” for the CAN-2000D serials devices.

Table 5.1 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	

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 supported	14	None	FF
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.

1. **Build the connection with CAN-2000D by using 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. Then the Bit-Strobe I/O Connection Object State will become established.**

.

3. **There are two ways to access the IO channels of CAN-2000D. One method is by the way of an IO connection object. Another 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 when the DeviceNet master device doesn't use the CAN-2000D any more.**

5.2 Predefined Master/Slave Connection Set

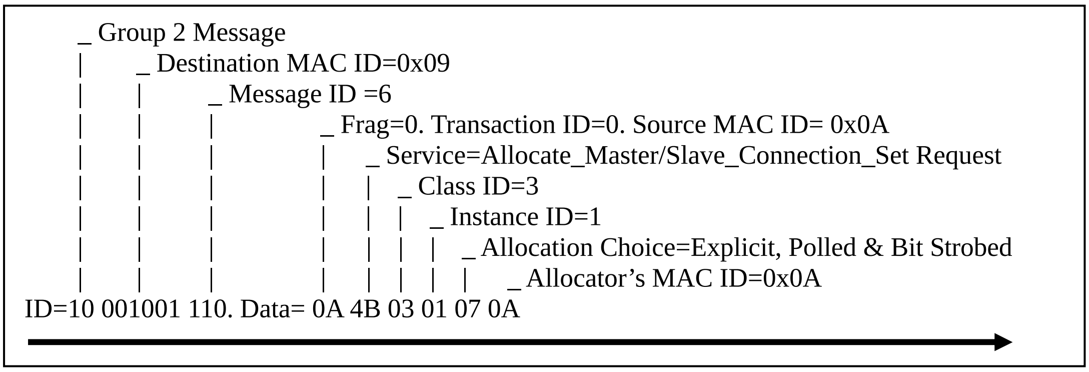
Master node sent a “Group 2 Only Unconnected Explicit Request” Message to request the use of the “Predefined Master/Slave Connection” set. This example is shows the user how to use it. In this demo, the Master establishes the Explicit Message, Poll IO and Bit-Strobe IO connections.

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

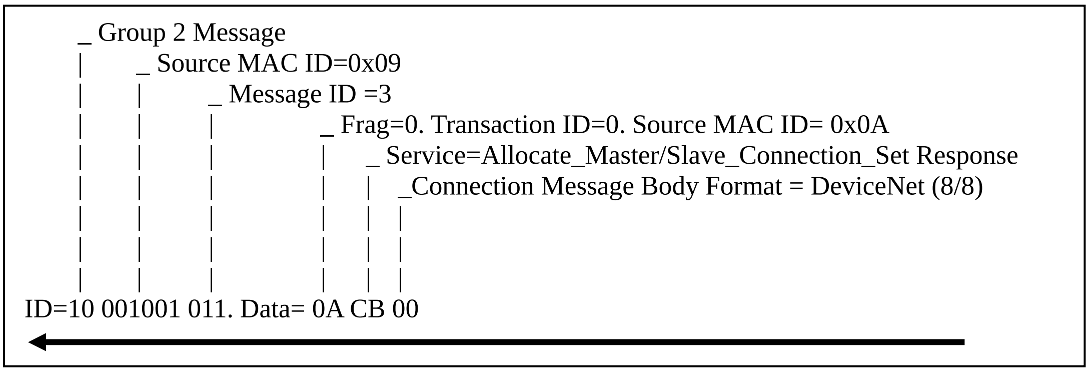
IDENTIFIER BITS											IDENTITY USAGE	HEX
10	9	8	7	6	5	4	3	2	1	0		RANGE
1	0	Source MAC ID						0	1	1	Slave’s Explicit/ Unconnected Response Messages	
1	0	Destination MAC ID						1	1	0	Group 2 Only Unconnected Explicit Request Messages	

Master node ID=0x0a, CAN-2000D series node ID=0x09

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
```

5.3 Explicit Messaging Connection Set

“Explicit Message Connection” is the basic connection between master/slave devices. After connecting with salve device, master can communicate and get/set slave attribute data via “Explicit Message Connection”. Here we use “Get/Set Attribute Single” services for example to apply the Application Object.

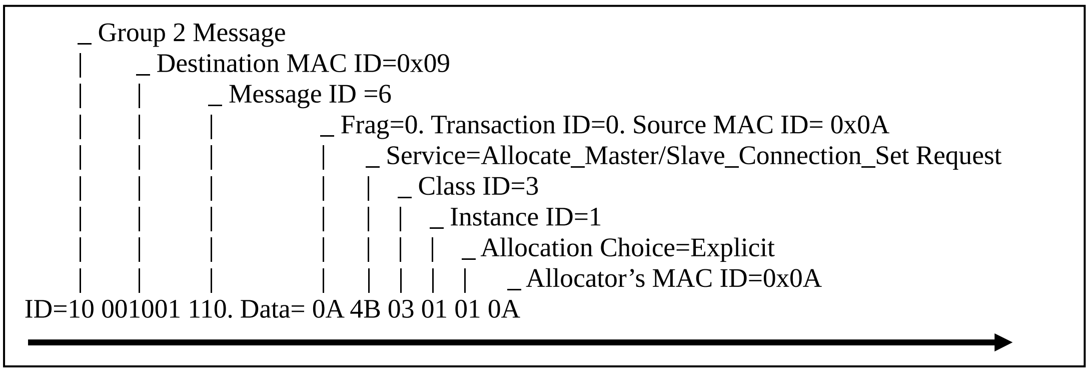
The figure below shows the Explicit Identifier Fields.

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

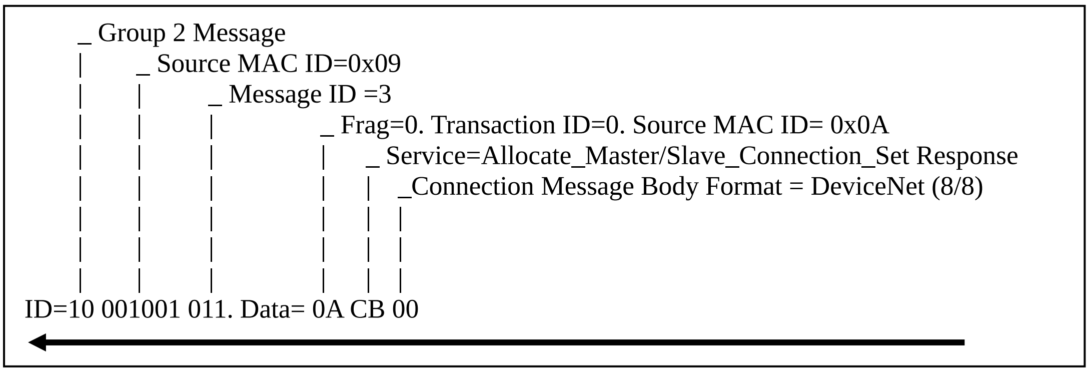
Master node ID=0x0A, CAN-2000D series node ID=0x09

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
   |     |      |          |    |   |   |   |    _ Allocator’s MAC ID=0x0A
ID=10 001001 110. Data= 0A 4B 03 01 01 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 Application Object (0x64) Instance (0x01) attribute (0x01)

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

```
     _ Group 2 Message
     |     _ Destination MAC ID=0x09
     |     |      _ Message ID =4
     |     |      |          _ Frag=0. Transaction ID=0. Source MAC ID= 0x0A
     |     |      |          |    _ Service=Set Attribute Request
     |     |      |          |    |  _ Class ID=0x64 (Application Object)
     |     |      |          |    |  |  _ Instance ID=1 (Application Object Instance 1)
     |     |      |          |    |  |  |  _ Attribute ID=1 (Application Object Attribute 1)
     |     |      |          |    |  |  |  |    _ Attribute Data= 0x01 (Set data 0x01)
ID=10 001001 100. Data= 0A 10 64 01 01 01
──────────────────────────────────────────────────────────────►
```

```
     _ Group 2 Message
     |     _ Source MAC ID=0x09
     |     |      _ Message ID =3
     |     |      |          _ Frag=0. Transaction ID=0. Source MAC ID= 0x0A
     |     |      |          |    _ Service= Set Attribute Response (No Error)
     |     |      |          |    |
     |     |      |          |    |
     |     |      |          |    |
     |     |      |          |    |
ID=10 001001 011. Data= 0A 90
◄──────────────────────────────────────────────────────────
```

3. Apply the Master's Explicit Request Messages to get the Application Object (0x64) Instance (0x01) attribute (0x01)

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

```
     _ Group 2 Message
     |     _ Destination MAC ID=0x09
     |     |      _ Message ID =4
     |     |      |          _ Frag=0. Transaction ID=0. Source MAC ID= 0x0A
     |     |      |          |    _ Service=Get Attribute Request
     |     |      |          |    |  _ Class ID=0x64 (Application Object)
     |     |      |          |    |  |  _ Instance ID=1 (Application Object Instance 1)
     |     |      |          |    |  |  |  _ Attribute ID=1 (Application Object Attribute 1)
     |     |      |          |    |  |  |  |
ID=10 001001 100. Data= 0A 0E 64 01 01
──────────────────────────────────────────────────────────────►
```

```
     _ Group 2 Message
     |     _ Source MAC ID=0x09
     |     |      _ Message ID =3
     |     |      |          _ Frag=0. Transaction ID=0. Source MAC ID= 0x0A
     |     |      |          |    _ Service= Set Attribute Response (No Error)
     |     |      |          |    |
     |     |      |          |    |  _ Attribute Data= 0x01
     |     |      |          |    |  |
     |     |      |          |    |  |
ID=10 001001 011. Data= 0A 8E 01
◄──────────────────────────────────────────────────────────
```

5.4 Polling I/O Message Set

"Poll Command and Poll Response" message can 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.

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	

Master node ID=0x0a, CAN-2000D series node ID=0x09

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 of the Poll I/O connection. Then the Poll I/O Connection Object State will become established.**

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

```
     _ Group 2 Message
     |     _ Destination MAC ID=0x09
     |     |      _ Message ID =4
     |     |      |         _ Frag=0. Transaction ID=0. Source MAC ID= 0x0A
     |     |      |         |   _ Service=Set Attribute Request
     |     |      |         |   |  _ Class ID=5
     |     |      |         |   |  |  _ Instance ID=2    ( Poll IO connection Instance ID )
     |     |      |         |   |  |  |  _ Attribute ID=9
     |     |      |         |   |  |  |  |    _ Attribute Data= 0x0E0A
ID=10 001001 100. Data= 0A 10 05 02 09 0A 0E
───────────────────────────────────────────────────────────►
```

```
     _ Group 2 Message
     |     _ Source MAC ID=0x09
     |     |      _ Message ID =3
     |     |      |         _ Frag=0. Transaction ID=0. Source MAC ID= 0x0A
     |     |      |         |   _ Service= Set Attribute Response
     |     |      |         |   |  _Response Attribute Data=0x0E10
     |     |      |         |   |  |
     |     |      |         |   |  |
     |     |      |         |   |  |
ID=10 001001 011. Data= 0A 90 10 0E
◄───────────────────────────────────────────────────────
```

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

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

```
     _ Group 2 Message
     |     _ Destination MAC ID=0x09
     |     |      _ Message ID =5
     |     |      |
     |     |      |           _ Poll Output Data
     |     |      |           |
ID=10 001001 101. Data= FF FF

───────────────────────────────────────────────────────────►
```

```
     _ Group 1 Message
     |  _ Message ID =F
     |  |      _ Source MAC ID=0x09
     |  |      |              _ Poll Response Data
     |  |      |              |
ID= 0 1111 001001. Data= FF DF
   ◄───────────────────────────────────────────────────────
```

5.5 Bit Strobe I/O Message Set

“Bit-Strobe Command and Bit-Strobe 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	

Master node ID=0x0a, CAN-2000D series node ID=0x09

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

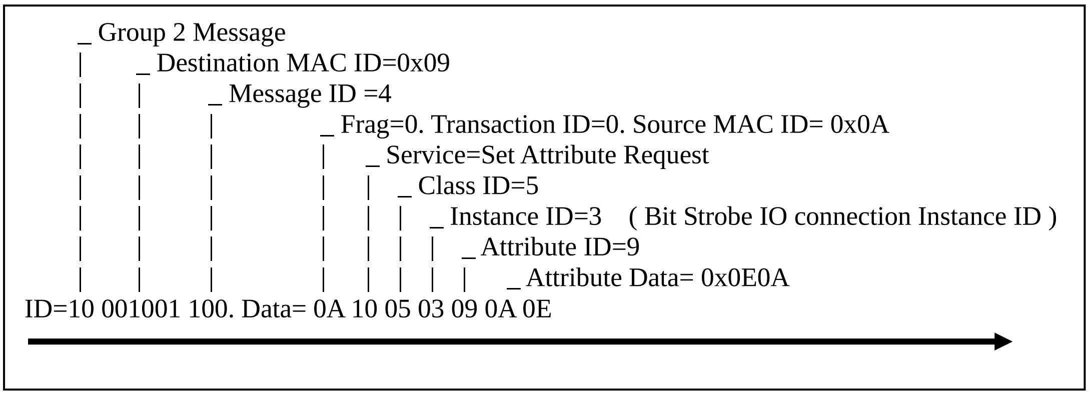
Master (MAC ID =0x0A)   Slave (MAC ID =0x09)

```
       _ 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
 ──────────────────────────────────────────────────────────────────────────►
```

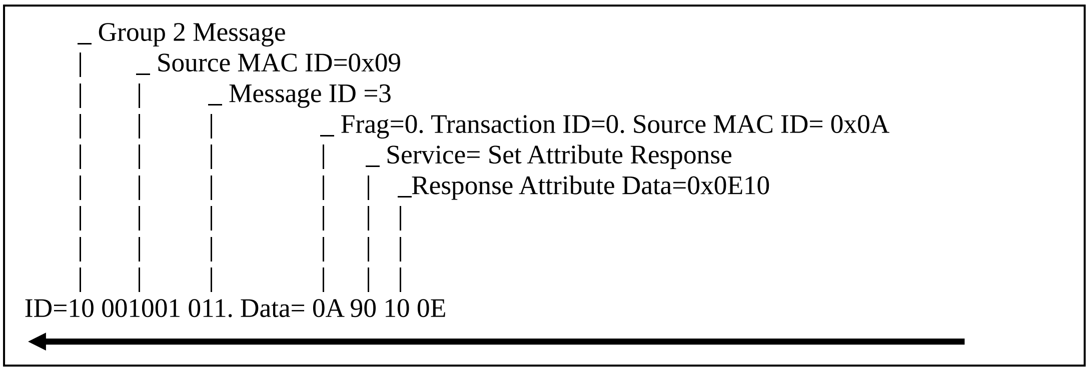
```
       _ Group 2 Message
       |      _ Source MAC ID=0x09
       |      |        _ Message ID =3
       |      |        |             _ Frag=0. Transaction ID=0. Source MAC ID= 0x0A
       |      |        |             |     _ Service=Allocate_Master/Slave_Connection_Set Response
       |      |        |             |     |   _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 Bit-Strobe I/O connection. Then the Bit-Strobe I/O Connection Object State will become established.**

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



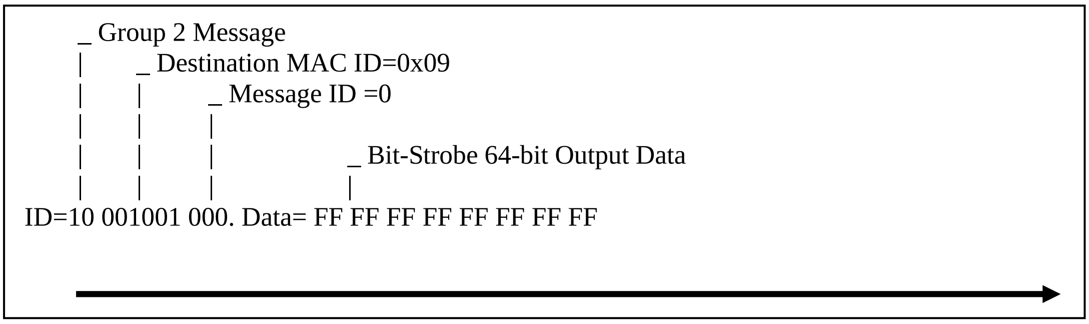
```
     _ Group 2 Message
     |      _ Destination MAC ID=0x09
     |      |       _ Message ID =4
     |      |       |            _ Frag=0. Transaction ID=0. Source MAC ID= 0x0A
     |      |       |            |     _ Service=Set Attribute Request
     |      |       |            |     |   _ Class ID=5
     |      |       |            |     |   |   _ Instance ID=3    ( Bit Strobe IO connection Instance ID )
     |      |       |            |     |   |   |   _ Attribute ID=9
     |      |       |            |     |   |   |   |     _ Attribute Data= 0x0E0A
ID=10 001001 100. Data= 0A 10 05 03 09 0A 0E
```



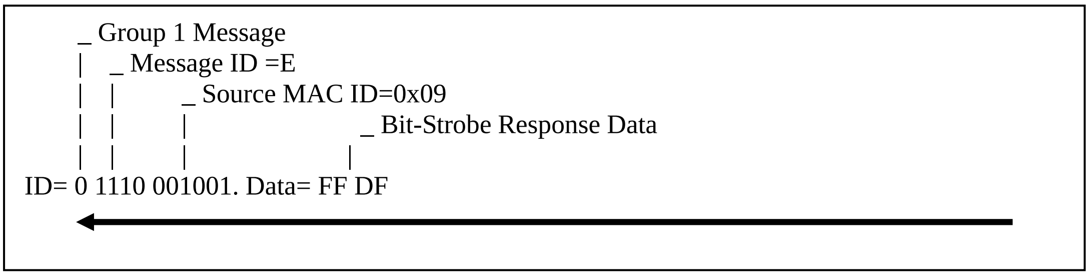
```
     _ Group 2 Message
     |      _ Source MAC ID=0x09
     |      |       _ Message ID =3
     |      |       |            _ Frag=0. Transaction ID=0. Source MAC ID= 0x0A
     |      |       |            |     _ Service= Set Attribute Response
     |      |       |            |     |   _Response Attribute Data=0x0E10
     |      |       |            |     |   |
     |      |       |            |     |   |
     |      |       |            |     |   |
ID=10 001001 011. Data= 0A 90 10 0E
```

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

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



```
     _ Group 2 Message
     |      _ Destination MAC ID=0x09
     |      |       _ Message ID =0
     |      |       |
     |      |       |               _ Bit-Strobe 64-bit Output Data
     |      |       |               |
ID=10 001001 000. Data= FF FF FF FF FF FF FF FF
```



```
     _ Group 1 Message
     |   _ Message ID =E
     |   |       _ Source MAC ID=0x09
     |   |       |                _ Bit-Strobe Response Data
     |   |       |                |
ID= 0 1110 001001. Data= FF DF
```

5.6 Reset Service

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

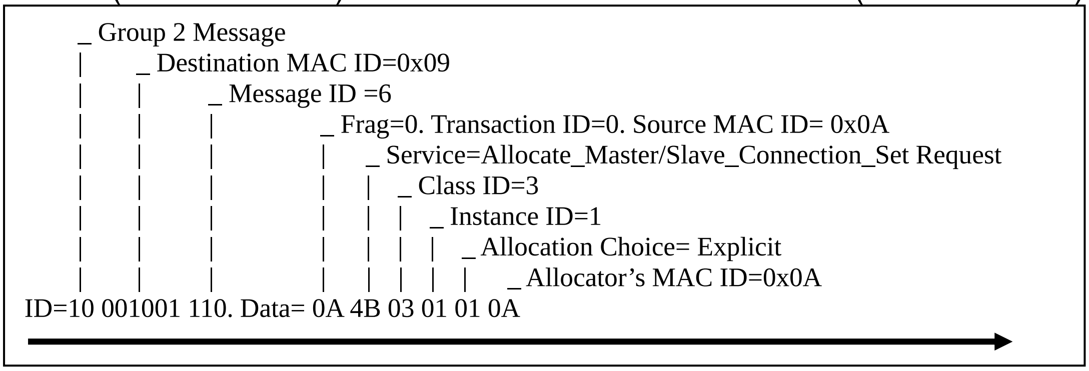
The parameter type for the reset common service has the following bit specifications:

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.

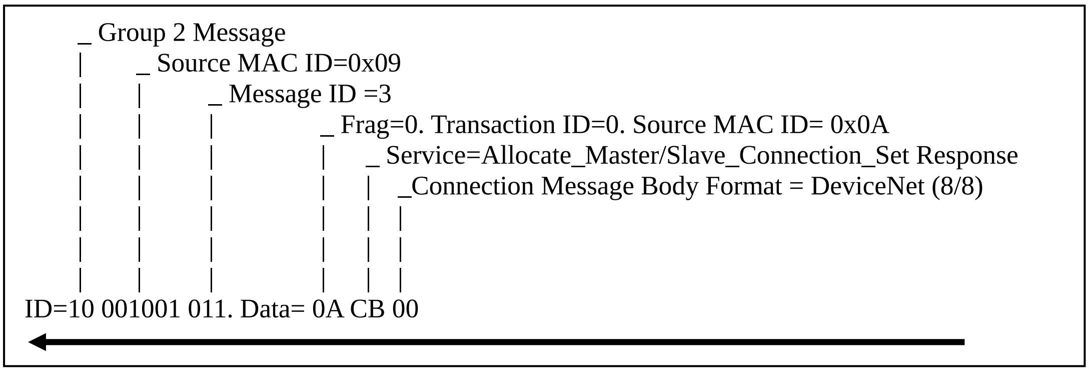
Master node ID=0x0a, CAN-2000D series node ID=0x09

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

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



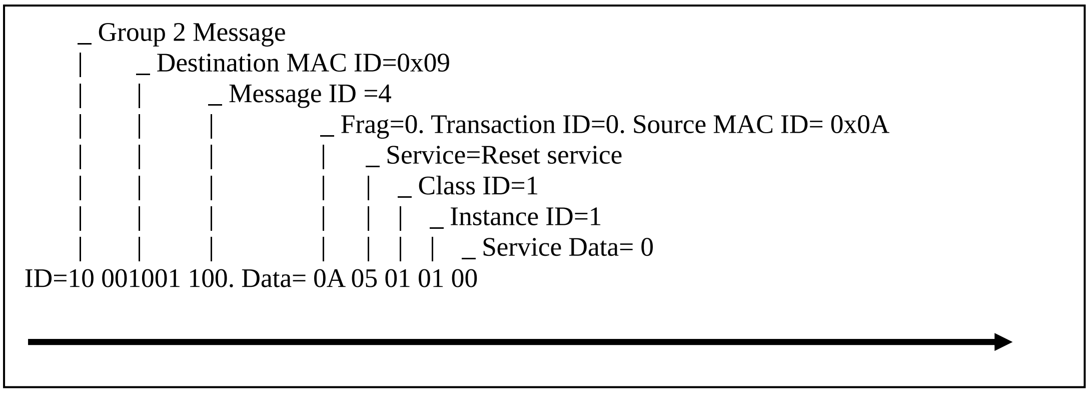
```
      _ Group 2 Message
      |       _ Destination MAC ID=0x09
      |       |         _ Message ID =6
      |       |         |              _ Frag=0. Transaction ID=0. Source MAC ID= 0x0A
      |       |         |              |     _ Service=Allocate_Master/Slave_Connection_Set Request
     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



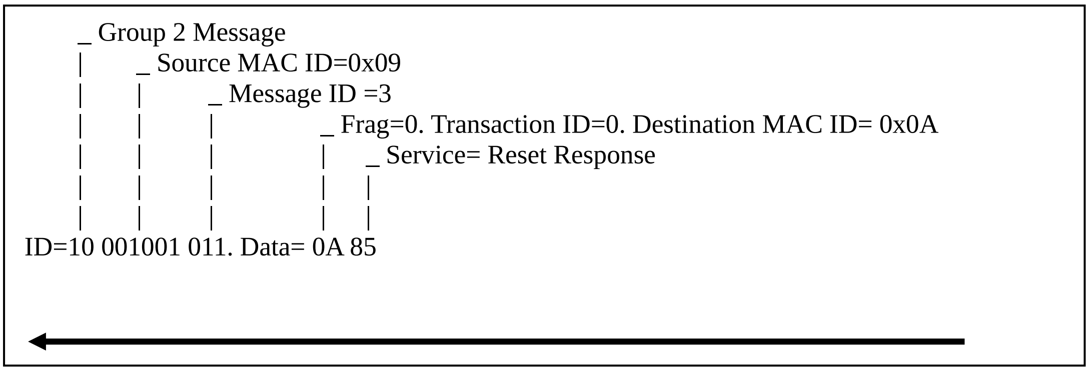
```
      _ Group 2 Message
      |       _ Source MAC ID=0x09
      |       |         _ Message ID =3
      |       |         |              _ Frag=0. Transaction ID=0. Source MAC ID= 0x0A
      |       |         |              |     _ Service=Allocate_Master/Slave_Connection_Set Response
      |       |         |              |     |   _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 Identify object. The service ID (0x05) is the reset service.

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



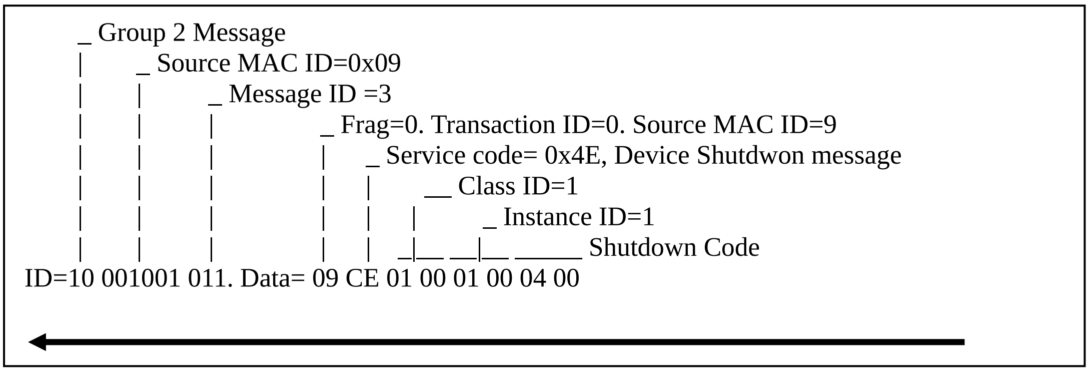
```
      _ Group 2 Message
      |     _ Destination MAC ID=0x09
      |     |      _ Message ID =4
      |     |      |          _ Frag=0. Transaction ID=0. Source MAC ID= 0x0A
      |     |      |          |    _ Service=Reset service
      |     |      |          |    |   _ Class ID=1
      |     |      |          |    |   |   _ Instance ID=1
      |     |      |          |    |   |   |   _ Service Data= 0
ID=10 001001 100. Data= 0A 05 01 01 00
```



```
      _ Group 2 Message
      |     _ Source MAC ID=0x09
      |     |      _ Message ID =3
      |     |      |          _ Frag=0. Transaction ID=0. Destination MAC ID= 0x0A
      |     |      |          |    _ Service= Reset Response
      |     |      |          |    |
      |     |      |          |    |
ID=10 001001 011. Data= 0A 85
```

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

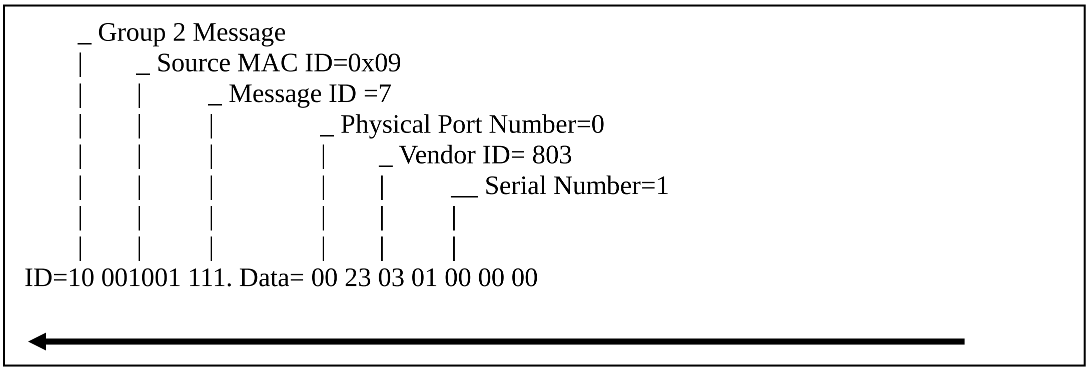
Master (MAC ID =0x0A)   Slave (MAC ID =0x09)



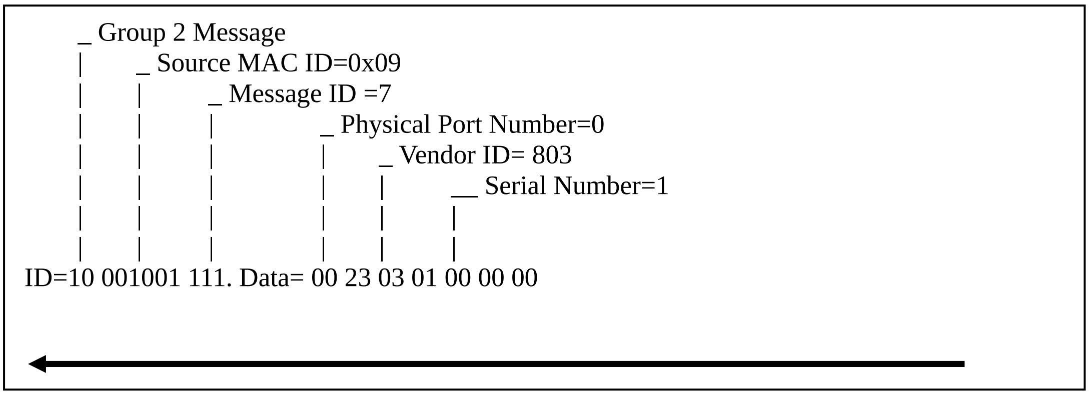
```
      _ Group 2 Message
      |     _ Source MAC ID=0x09
      |     |      _ Message ID =3
      |     |      |          _ Frag=0. Transaction ID=0. Source MAC ID=9
      |     |      |          |    _ Service code= 0x4E, Device Shutdwon message
      |     |      |          |    |      __ Class ID=1
      |     |      |          |    |    |       _ Instance ID=1
      |     |      |          |    |   _|__ __|__ _____ Shutdown Code
ID=10 001001 011. Data= 09 CE 01 00 01 00 04 00
```

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)



```
      _ Group 2 Message
      |     _ Source MAC ID=0x09
      |     |      _ Message ID =7
      |     |      |          _ Physical Port Number=0
      |     |      |          |     _ Vendor ID= 803
      |     |      |          |     |      __ Serial Number=1
      |     |      |          |     |      |
      |     |      |          |     |      |
ID=10 001001 111. Data= 00 23 03 01 00 00 00
```



```
      _ Group 2 Message
      |     _ Source MAC ID=0x09
      |     |      _ Message ID =7
      |     |      |          _ Physical Port Number=0
      |     |      |          |     _ Vendor ID= 803
      |     |      |          |     |      __ Serial Number=1
      |     |      |          |     |      |
      |     |      |          |     |      |
ID=10 001001 111. Data= 00 23 03 01 00 00 00
```

5.7 Device Heartbeat

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

Master node ID=0x0a, CAN-2000D series node ID=0x09

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

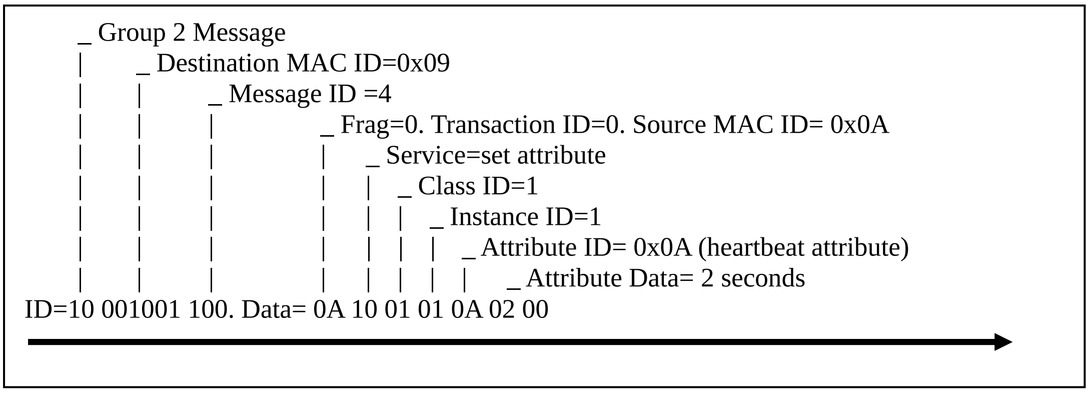
Master (MAC ID =0x0A)   Slave (MAC ID =0x09)

```
      _ 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
      |      |        |             |      |   |   |   |     _ Allocator's MAC ID=0x0A
ID=10 001001 110. Data= 0A 4B 03 01 01 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 heartbeat interval value.

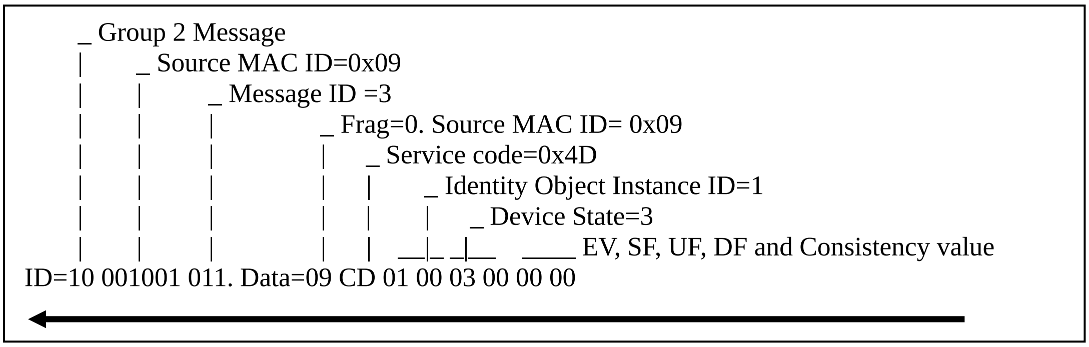
Master (MAC ID =0x0A)   Slave (MAC ID =0x09)



```
      _ Group 2 Message
      |     _ Destination MAC ID=0x09
      |     |      _ Message ID =4
      |     |      |         _ Frag=0. Transaction ID=0. Source MAC ID= 0x0A
      |     |      |         |    _ Service=set attribute
      |     |      |         |    |  _ Class ID=1
      |     |      |         |    |  |  _ Instance ID=1
      |     |      |         |    |  |  |  _ Attribute ID= 0x0A (heartbeat attribute)
      |     |      |         |    |  |  |  |    _ Attribute Data= 2 seconds
ID=10 001001 100. Data= 0A 10 01 01 0A 02 00
```

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

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



```
      _ Group 2 Message
      |     _ Source MAC ID=0x09
      |     |      _ Message ID =3
      |     |      |         _ Frag=0. Source MAC ID= 0x09
      |     |      |         |    _ Service code=0x4D
      |     |      |         |    |    _ Identity Object Instance ID=1
      |     |      |         |    |    |    _ Device State=3
      |     |      |         |    |  __|_ _|__   ____ EV, SF, UF, DF and Consistency value
ID=10 001001 011. Data=09 CD 01 00 03 00 00 00
```

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

5.8 Fragmentation Message

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**

Fragmentation of an **I/O message** is performed in an **Unacknowledged** fashion. Unacknowledged fragmentation consists of the back-to-back transmission of the fragment (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 acknowledge from the receiving modules.

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)

```
     _ Group 2 Message
     |      _ Destination MAC ID=0x09
     |      |        _ Message ID =5
     |      |        |
     |      |        |              _ Fragment Type= First Fragment, Fragment Count=0
     |      |        |             |     ---------------------- 1st portion of the I/O data
ID=10 001001 101. Data= 00 01 02 03 04 05 06 07
```

```
     _ Group 2 Message
     |      _ Destination MAC ID=0x09
     |      |        _ Message ID =5
     |      |        |
     |      |        |              _ Fragment Type= Final Fragment, Fragment Count=1
     |      |        |             |     ---------------------- final portion of the I/O data
ID=10 001001 101. Data= 81 08 09 0A
```

- **Acknowledge Fragmentation**

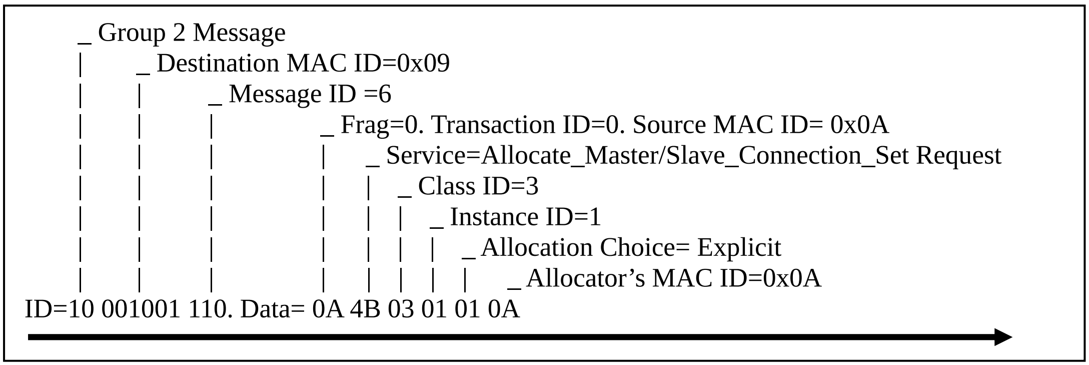
Fragmentation of an **Explicit Message** is performed in an **Acknowledged** fashion. Acknowledged fragmentation consists of the transmission of a fragment from the transmitting module followed by the transmission of an acknowledgment by the receiving module. The receiving module acknowledges the reception 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.

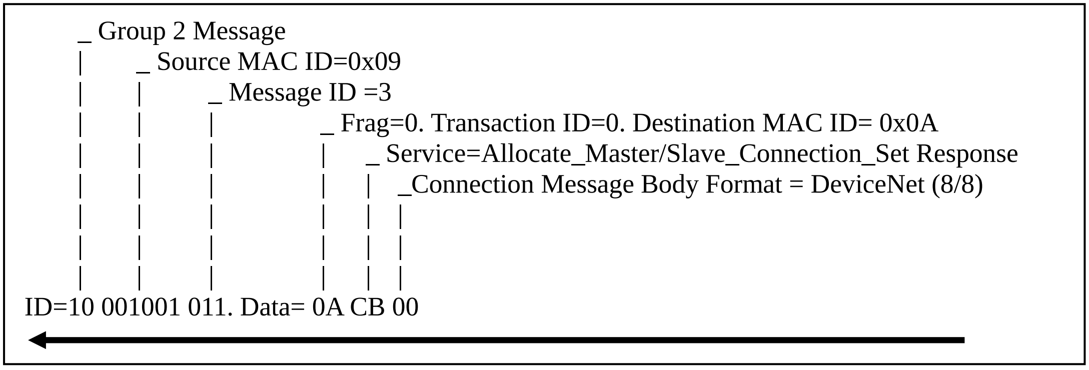
Note: The slave device node is 0x09, and the master node ID is 0x0A

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

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



```
     _ 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
     |     |     |         |    |   |   |   |    _ Allocator's MAC ID=0x0A
ID=10 001001 110. Data= 0A 4B 03 01 01 0A
```

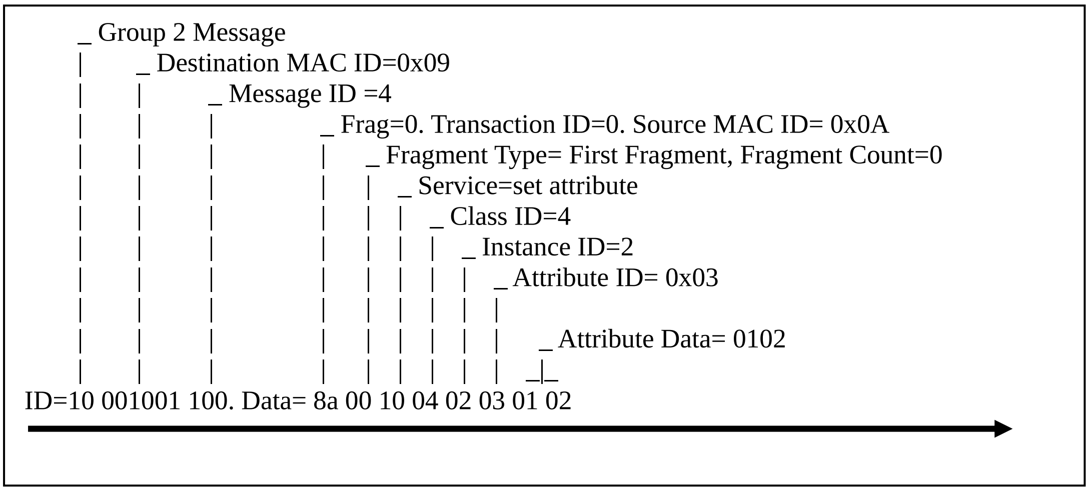


```
     _ Group 2 Message
     |     _ Source MAC ID=0x09
     |     |     _ Message ID =3
     |     |     |         _ Frag=0. Transaction ID=0. Destination MAC ID= 0x0A
     |     |     |         |    _ Service=Allocate_Master/Slave_Connection_Set Response
     |     |     |         |    |   _Connection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

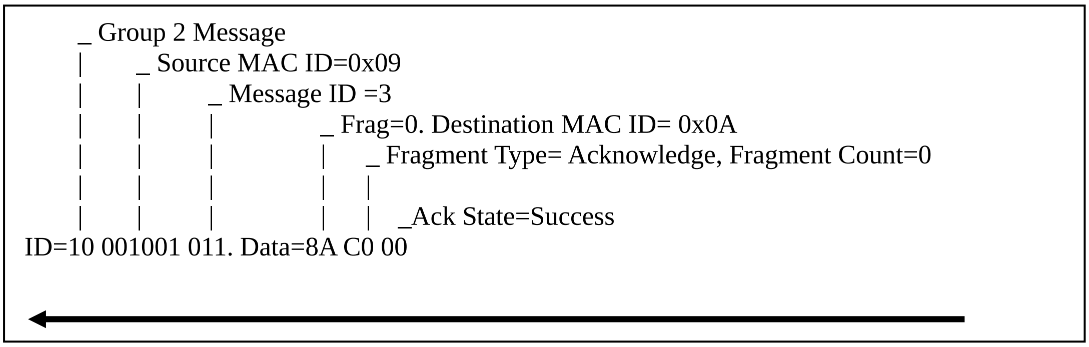
2. Apply the Master’s Explicit Request Messages to set the Assembly object, Class ID (0x04), Instance ID (0x02), Attribute ID (0x03).

Service (0x10) = set attribute service.
Data = 0102030405060708090A.

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



```
     _ Group 2 Message
     |     _ Destination MAC ID=0x09
     |     |      _ Message ID =4
     |     |      |          _ Frag=0. Transaction ID=0. Source MAC ID= 0x0A
     |     |      |          |    _ Fragment Type= First Fragment, Fragment Count=0
     |     |      |          |    |  _ Service=set attribute
     |     |      |          |    |  |  _ Class ID=4
     |     |      |          |    |  |  |  _ Instance ID=2
     |     |      |          |    |  |  |  |  _ Attribute ID= 0x03
     |     |      |          |    |  |  |  |  |
     |     |      |          |    |  |  |  |  |   _ Attribute Data= 0102
     |     |      |          |    |  |  |  |  |  _|_
ID=10 001001 100. Data= 8a 00 10 04 02 03 01 02
```



```
     _ Group 2 Message
     |     _ Source MAC ID=0x09
     |     |      _ Message ID =3
     |     |      |          _ Frag=0. Destination MAC ID= 0x0A
     |     |      |          |    _ Fragment Type= Acknowledge, Fragment Count=0
     |     |      |          |    |
     |     |      |          |    |  _Ack State=Success
ID=10 001001 011. Data=8A C0 00
```

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

```
     _ Group 2 Message
     |     _ Destination MAC ID=0x09
     |     |      _ Message ID =4
     |     |      |          _ Frag=0. Transaction ID=0. Source MAC ID= 0x0A
     |     |      |          |    _ Fragment Type= First Fragment, Fragment Count=1
     |     |      |          |    |     _ Attribute Data= 03 04 05 06 07 08
     |     |      |          |    |   __|___________
ID=10 001001 100. Data= 8a 41 03 04 05 06 07 08

──────────────────────────────────────────────────────────────►
```

```
     _ Group 2 Message
     |     _ Source MAC ID=0x09
     |     |      _ Message ID =3
     |     |      |          _ Frag=1. Destination MAC ID= 0x0A
     |     |      |          |    _ Fragment Type= Acknowledge, Fragment Count=1
     |     |      |          |    |
     |     |      |          |    |   _Ack State=Success
ID=10 001001 011. Data=8A C1 00

◄──────────────────────────────────────────────────────────
```

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

```
     _ Group 2 Message
     |     _ Destination MAC ID=0x09
     |     |      _ Message ID =4
     |     |      |          _ Frag=1. Transaction ID=0. Source MAC ID= 0x0A
     |     |      |          |    _ Fragment Type= Final Fragment, Fragment Count=2
     |     |      |          |    |     _ Attribute Data= 09 0A
     |     |      |          |    |   __|_
ID=10 001001 100. Data= 8a 82 09 0A

──────────────────────────────────────────────────────────────►
```

```
     _ Group 2 Message
     |     _ Source MAC ID=0x09
     |     |      _ Message ID =3
     |     |      |          _ Frag=1. Destination MAC ID= 0x0A
     |     |      |          |    _ Fragment Type= Acknowledge, Fragment Count=2
     |     |      |          |    |
     |     |      |          |    |   _Ack State=Success
ID=10 001001 011. Data=8A C2 00

◄──────────────────────────────────────────────────────────
```