FP-1100 USER MANUAL

Last revised: 12/29/00

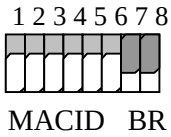
Table of Contents

Chapter 1
Hardware Configuration

The FP-1100 network module connects a DeviceNet network to FieldPoint I/O modules. On the DeviceNet network, the module acts as a slave UCMM device. It supports Poll, Change of State, and Cyclic connections of the Predefined Master/Slave I/O set.

Configuring the Network Module

The figure below shows the 8-position switch on the FP-1100 module. Switches 1-6 set the MAC ID, and switches 7-8 set the baud rate.



The MAC ID (Media Access Control Identifier) is a unique identifier assigned to the DeviceNet device to distinguish it from other DeviceNet nodes on the same network. It can range from 0 to 63. Switch 6 represents the LSB of the MACID and switch 1 represents the MSB of the address.

The FP-1100 module supports all three baud rates specified by the DeviceNet Specifications. The last two switches can be used to set the baud rate. Switch 8 represents the LSB of the baud rate value and switch 7 represents the MSB. The switch settings and the respective baud rates are shown in the table below.

Baud Rate Switch Settings

Switch Setting (S7 S8)	**Baud rate (Kbits/sec)**
0 0	125
0 1	250
1 0	500
1 1	Invalid setting

The baud rate and MACID switches are only read at power up or on soft reset. Changes made to the switches after the module has been configured are ignored.

LED Indicators

The FP-1100 module has three LED indicators: **POWER**, **MOD/NET STATUS**, and **I/O STATUS**. The green **POWER** LED is lit while the network module is powered up. This LED indicates that the power supply connected to the network module is acceptable, and that the network module is supplying power to the I/O modules.

The other two LEDs, **MOD/NET** and **I/O STATUS**, behave according to the DeviceNet Specifications. The **MOD/NET STATUS** LED serves as Combined Module/Network Status LED (Sec. 8-2.5 of the *DeviceNet Specifications* – vol. I). A summary of its behavior is listed below:

MOD/NET STATUS LED will

- flash green when no Explicit Messaging Connection (EM) is open
- become solid when at least one EM is open
- go back to flashing green if the EM is closed or timed-out
- flash red, irrespective of EM state, if there's a configuration error (major recoverable fault)
- become solid red if there's a major unrecoverable fault and will not respond to any EM or other message on the bus. Only a hard reset can clear this state.

The **I/O STATUS** LED conforms to the I/O Status LED mentioned in the DeviceNet Specifications (Sec. 8-2.7 of the *DeviceNet Specifications* – vol-I). A summary of its behavior is listed below:

The **I/O STATUS** LED is

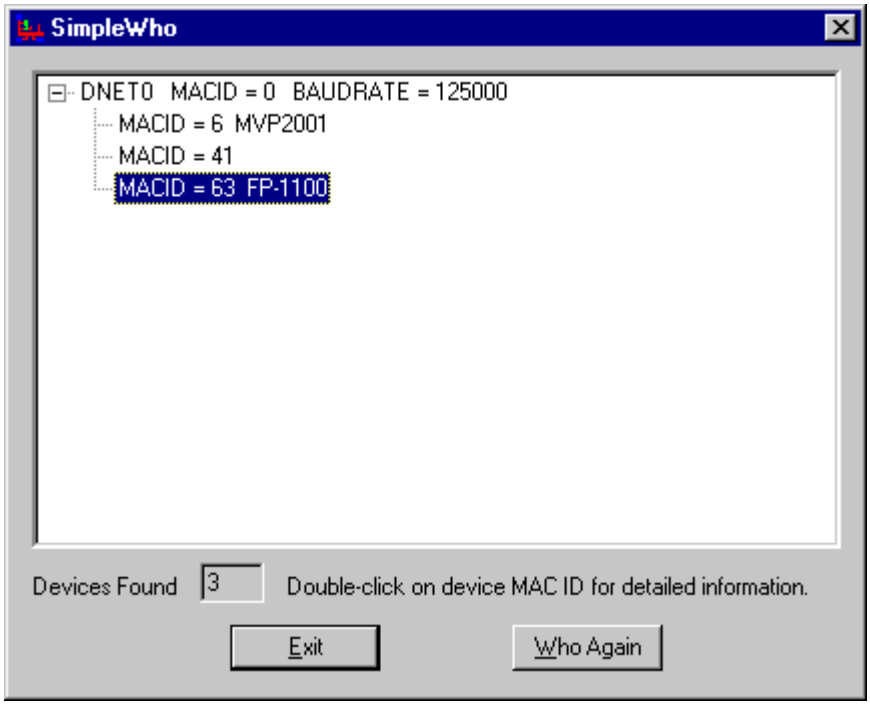
- off when no I/O connection is open
- solid green when at least one I/O connection is running
- flashing green when all open I/O connections are in Idle state
- flashing red when at least one I/O connection is in timeout state
- off at all other times (some exceptions apply)

Chapter 2
Configuring FP-1100 with SimpleWho

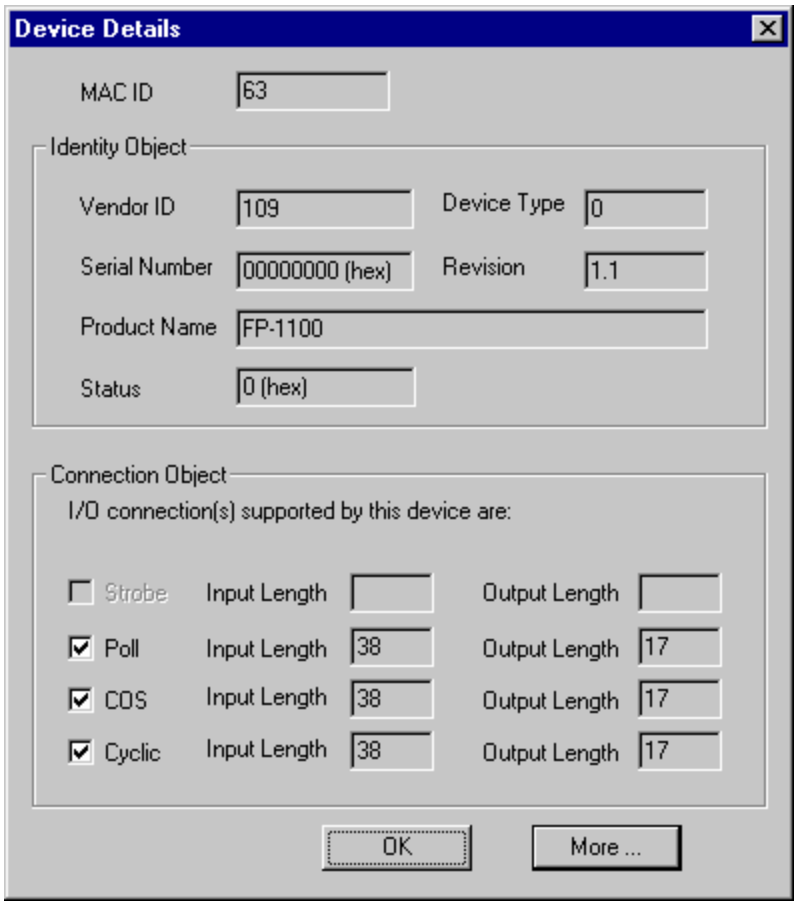
To configure the FieldPoint DeviceNet network module, run the SimpleWho utility that is shipped with FP-1100.

> NOTE *The SimpleWho utility requires that you have National Instruments DeviceNet hardware (NI-DNET board) and software installed in your machine. The NI-DNET software comes with a simplified version of SimpleWho. This chapter assumes that you know how to start the utility and find DeviceNet devices currently present on the network using SimpleWho shipped with FieldPoint.*

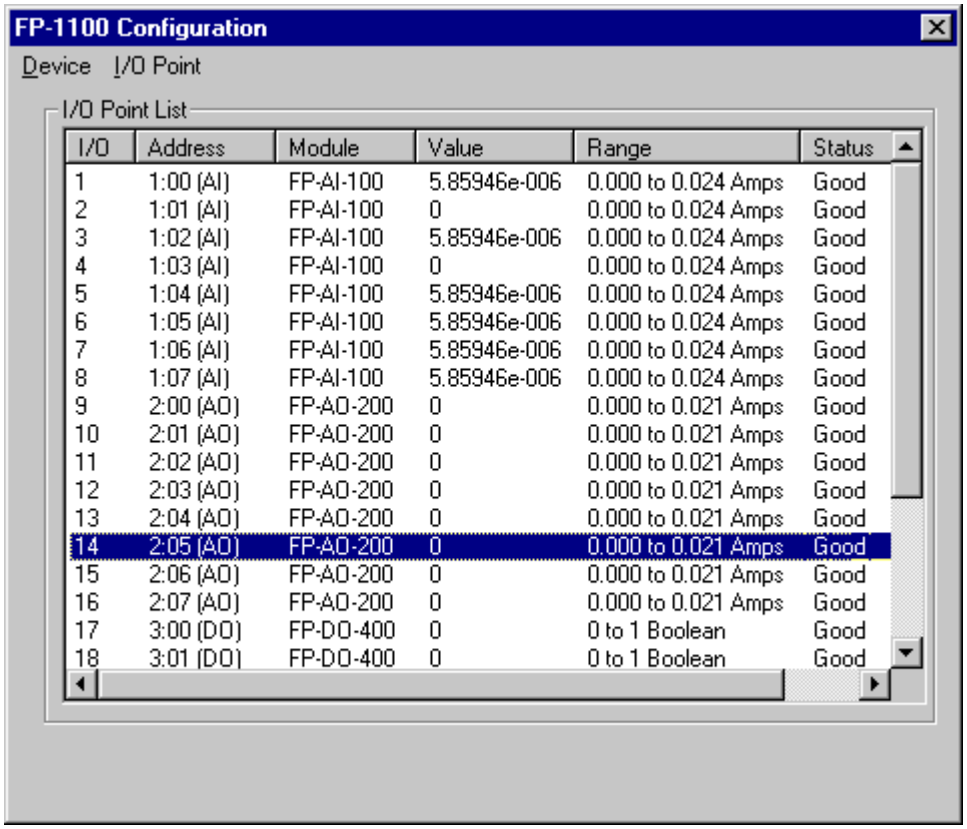
1. Follow the steps given in the *NI-DNET User Manual* to start SimpleWho and scan the network. A dialog box will pop up with a list of devices currently present on the network, as shown in the following illustration.



2. Double click on the FP-1100 device. A Device Details dialog will pop up, showing the Identity Object and Connection Object attributes for the FP-1100 system, as shown in the following illustration. If you want to use the default configuration for the device, note the *Input Length* and *Output Length* values for the desired I/O connection and go to *Running a Simple Application* section. To change the default configuration, proceed to step 3.



3. To see a list of I/O point instances present in the device, click on the **Device Configuration** button. Note that this button is only enabled for FP-1100 modules. The "FP-1100 Configuration" dialog box containing a list of I/O points will pop up. The I/O points correspond to physical channels on the I/O modules starting from I/O module 1, channel 0. If the saved configuration is different from the actual physical configuration, the device boots up with the saved configuration and this list will consist of that configuration with **Status** of mismatching I/O Points set to *Bad*.



I/O	Address	Module	Value	Range	Status
1	1:00 (AI)	FP-AI-100	5.85946e-006	0.000 to 0.024 Amps	Good
2	1:01 (AI)	FP-AI-100	0	0.000 to 0.024 Amps	Good
3	1:02 (AI)	FP-AI-100	5.85946e-006	0.000 to 0.024 Amps	Good
4	1:03 (AI)	FP-AI-100	0	0.000 to 0.024 Amps	Good
5	1:04 (AI)	FP-AI-100	5.85946e-006	0.000 to 0.024 Amps	Good
6	1:05 (AI)	FP-AI-100	5.85946e-006	0.000 to 0.024 Amps	Good
7	1:06 (AI)	FP-AI-100	5.85946e-006	0.000 to 0.024 Amps	Good
8	1:07 (AI)	FP-AI-100	5.85946e-006	0.000 to 0.024 Amps	Good
9	2:00 (AO)	FP-AO-200	0	0.000 to 0.021 Amps	Good
10	2:01 (AO)	FP-AO-200	0	0.000 to 0.021 Amps	Good
11	2:02 (AO)	FP-AO-200	0	0.000 to 0.021 Amps	Good
12	2:03 (AO)	FP-AO-200	0	0.000 to 0.021 Amps	Good
13	2:04 (AO)	FP-AO-200	0	0.000 to 0.021 Amps	Good
14	2:05 (AO)	FP-AO-200	0	0.000 to 0.021 Amps	Good
15	2:06 (AO)	FP-AO-200	0	0.000 to 0.021 Amps	Good
16	2:07 (AO)	FP-AO-200	0	0.000 to 0.021 Amps	Good
17	3:00 (DO)	FP-DO-400	0	0 to 1 Boolean	Good
18	3:01 (DO)	FP-DO-400	0	0 to 1 Boolean	Good

Columns

- I/O – Instance Id of the I/O point starting from 1.
- Address – Physical location and type of this I/O point in *module:channel (type)* format. For example, 1:07 (AI) corresponds to channel 7 on module 1 which is of Analog Input type.
- Module – I/O module name, as shown on the module label.
- Value – Current channel value.
- Range – Configured range.
- Status – Current channel status (*Good/Bad*).

Menus

Device – items in this menu perform global operations.

Reset

Cycle Power – Reset to saved configuration.
Out of Box – Clear saved configuration and reset with factory defaults.

The reset item performs the Identity Object's reset service on the device with *reset_type* = 0 for Cycle Power and *reset_type* = 1 for Out of Box.

Save

to Device – Store configuration changes to non-volatile memory.

This option performs the I/O Point's **Save** service on the I/O Point Class instance. It should be requested after all the I/O configuration and assembly changes have been made.

to File – Save current configuration to file.

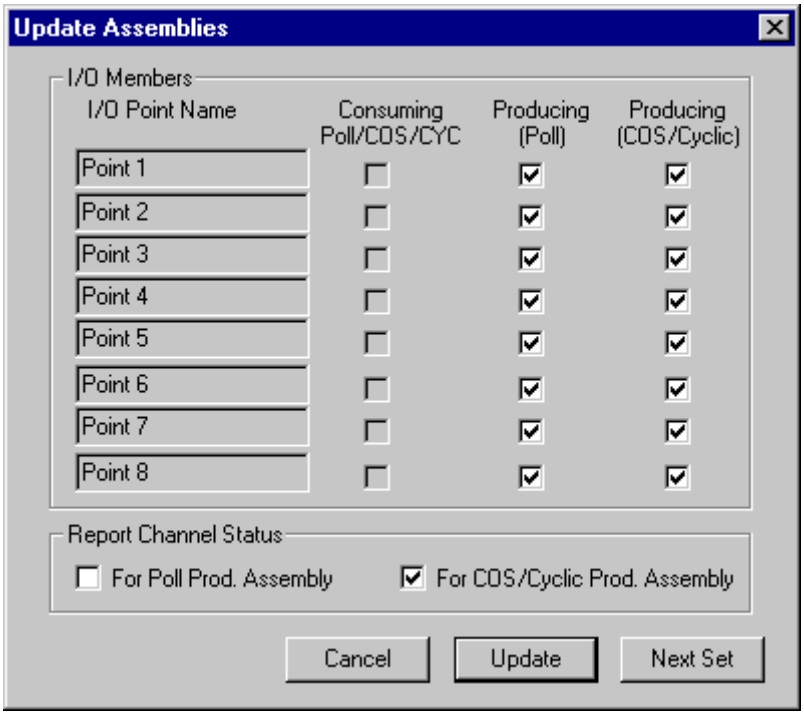
The *Save to File* option saves the current I/O Point configuration to a file (in "ini" file format) that can be used by other applications to determine range and data offset for a given point.

Update Firmware – Download new firmware image.

Modify Assembly – Change I/O connection assemblies.

This option changes the assembly list attribute for different I/O assemblies. On selecting this option the "Update Assemblies" dialog box will pop up, showing the current setting for the first eight I/O points. You can insert/remove an I/O point from the assembly list by checking/unchecking the corresponding checkbox. To go to the next set of I/O points, click on the **Next Set** button. The changes you have made so far will be saved locally and the next eight (or fewer) I/O points will be shown. To enable or disable the channel status reporting for a producing assembly, check/uncheck the corresponding box in the *Report Channel Status* section of the dialog box. When done modifying the assembly configuration, click on **Update** to commit your changes to the device. Note the items in *Report Channel Status* section apply to the assembly as a whole and the setting at the time of **Update** is the one actually sent to the device.

Select **Device>>Save>>to Device** to store the assembly changes in non-volatile memory.



Refresh

I/O Data – Update I/O value and status.
All – Reload I/O configuration and value from the device.

Exit

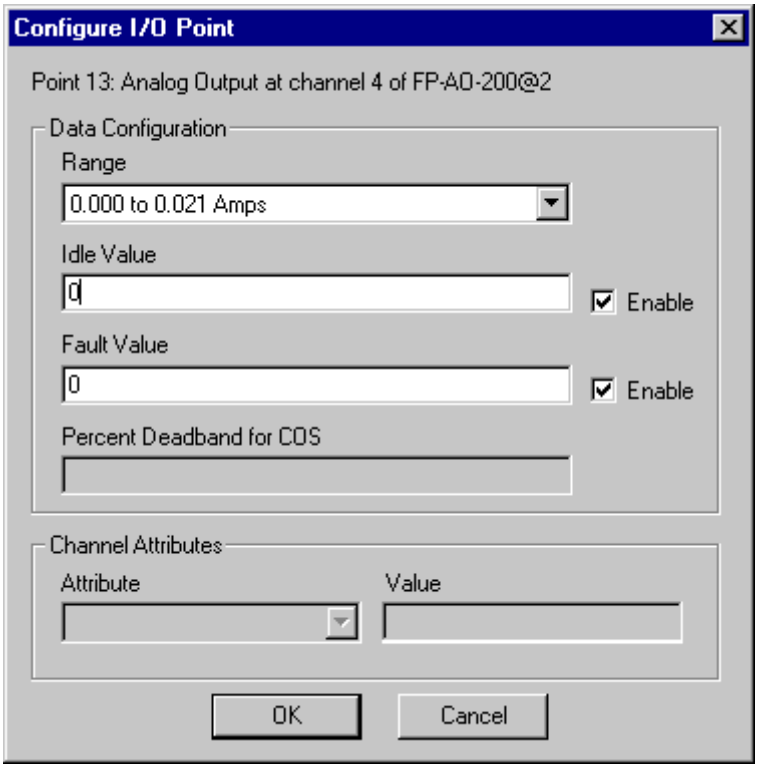
I/O Point – items in this menu perform object specific operations, and an I/O point must be highlighted before selecting the items from this menu.

Configure – Configure I/O point attribute(s), such as range, deadband, etc.

This option lets you configure the range and other attributes of an I/O point. On selecting this option the "Configure I/O Point" dialog box will pop up, showing the current setting for the selected I/O point. Depending on the I/O type (analog vs. discrete) and direction (input vs. output), some of the controls in this dialog box will be disabled. Modify the attributes as needed and then click on OK to send your changes to the device.

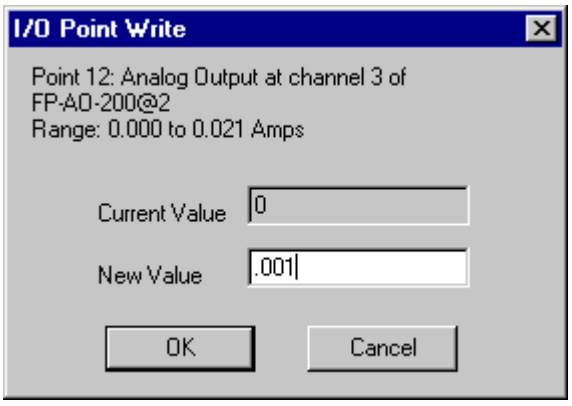
If you want to change multiple I/O points at the same time, check the "**Configure all I/O Points of the same type on this module**" button before pressing OK. This will change the configuration for all the I/O channels on this module that match the current point.

Select **Device>>Save>>to Device** to store the I/O point configuration changes in non-volatile memory. If you are planning to change several I/O points, make those changes and then perform the **Save** operation.



Write – Write I/O value.

This option lets you send a new value to an output point. On selecting this option, the "I/O Point Write" dialog box will pop up. Enter the new value and select OK. The new value will be written to the I/O Point and should be displayed in the "I/O Point List" shortly.



NOTES

1. *If the FP-1100 module has an active I/O connection (I/O Status LED is on), then menu items (Device>>Update Firmware, Device>>Modify Assemblies, Device>>Save>>to Device, I/O Point>>Configure, and I/O Point>>Write) that change the device configuration may not work.*

2. *For more information on different objects and services mentioned in this section, refer to Chapter 3, FP-1100 DeviceNet Object.*

Running a Simple Application

If you have National Instruments DeviceNet (NI-DNET) hardware and software installed on your system, run the **SingleDevice** example that comes with the NI-DNET software. Use the MACID, Input Length, and Output Length you got from the SimpleWho utility to configure the I/O connection and then run the example. Refer to Appendix D, *Assembly Decoding*, to find the data offset for each channel. For more information on **SingleDevice** example, refer to the *NI-DNET User Manual*.

Chapter 3
FP-1100 DeviceNet Objects

This chapter describes the objects and the services supported by FP-1100.

Object Overview

A DeviceNet object is an abstract representation of a particular component within a DeviceNet device. Each object contains several attributes that describe externally visible characteristics or features of that object. Typically, attributes provide status information or govern the operation of an object. In addition, each object performs a number of procedures through one or more services that it supports.

A set of objects that all represent the same kind of system components is called a class. Objects within a class are called object instances. A class may include one or more instances of the object it represents. Each instance of a class has the same set of attributes, but has its own set of attribute values that distinguishes it from the other instances in the same class.

Object Addressing

A DeviceNet device contains several objects. These objects interact to provide basic product behavior. Each object instance of a class in the device has the same set of attributes, but has its own particular set of attribute values. To allow multiple object instances from the same class to reside in the device and be accessible over the network as separate components, DeviceNet defines a hierarchy of address identifiers. Together, these identifiers allow any attribute of any instance of any class to be uniquely identified over the DeviceNet network.

Media Access Control Identifier (MAC ID) This is a unique identifier assigned to the DeviceNet device to distinguish it from other DeviceNet nodes on the same network.

Class Identifier (Class ID) This is a unique identifier assigned to each object class that is accessible over the network. The Object Class may be referenced with this Class ID.

Instance Identifier (Instance ID) This is a unique identifier assigned to each object instance to distinguish it from all other instances of that class.

Attribute Identifier (Attribute ID) This is a unique identifier assigned to each attribute within the instance that identifies it among all other attributes of that object instance.

FP-1100 Objects

The FP-1100, like any other DeviceNet device, consists of a collection of objects. Each object is unique in its behavior and purpose. The table below lists all the DeviceNet objects supported by the FP-1100.

DeviceNet Objects Supported by FP-1100

Class ID	Object
01 hex	Identity
02 hex	Message Router
03 hex	DeviceNet
64 hex	I/O Point (*vendor specific*)
04 hex	Assembly
05 hex	Connection
2B hex	Acknowledge Handler

The following sections describe attributes and services supported by each of the above object classes. Any service supported at the class level can be invoked by requesting that service for the class instance (Instance ID 0) of that object.

Common Error Codes

The error codes given in the following table can be returned from any of the services listed in the following pages. For service specific codes, please refer to the error tables given in each service description section.

General Error Codes

Error Condition	General Error Response	General Error Code	Additional Error Code
The requested service is not supported/defined for the class instance of this object	Service not supported	08 hex	FF hex
The requested service is not supported for this object class	Service not supported	08 hex	FF hex
The requested service is not supported for the specified attribute Id	Service not supported	08 hex	Attribute ID
The attribute, specified by the attribute ID in the explicit message, can not be modified	Attribute not settable	0E hex	FF hex
The service request data is not enough to perform the specified operation (see the Service Request Data section given with each service description section to find the desired service data length).	Not enough data	13 hex	FF hex
The attribute Id does not refer to an attribute that is supported by this object	Attribute not supported	14 hex	FF hex
The service supplied more data than was expected	Too much data	15 hex	FF hex
Object instance *N*, specified in the instance field of the explicit message, does not exist	Object does not exist	16 hex	FF hex

Identity Object

The Identity Object provides identification and general information about a DeviceNet node. The FP-1100 unit contains one instance of the Identity Object.

The Identity Object is required to be present in all DeviceNet devices and its attributes and services are described in detail in Volume II of the DeviceNet Specifications. The information that is pertinent to FP-1100 is replicated here for quick reference.

Class Attributes

Attribute ID	Applicable Services	Name	Data Type	Default Value	Description of Attribute
1	Get	Revision	UINT	1	Revision of this object
2	Get	Max Instance	UINT	1	Maximum Instance number of the object in this class

Instance Attributes

Attribute ID	Applicable Services	Name	Data Type	Default Value	Description of Attribute
1	Get	Vendor ID	UINT	109	National Instruments vendor Id (see semantics)
2	Get	Device Type	UINT	1	General type of the product
3	Get	Product Code			
4	Get	Revision	Struct of:		Current version of FP-1100
		Major	USINT	1	
		Minor	USINT	1	
5	Get	Status	WORD		Status of the device (see semantics)
6	Get	Serial Number	UDINT		Serial Number of the device
7	Get	Product Name	SHORT_STRING		
8	Get	State	USINT		Present state of the device (see semantics)
9	Get	Configuration Consistency Value	UINT		Checksum of the last saved configuration of the device (see semantics)

Semantics

Vendor ID
The vendor ID is a unique identifier that is assigned by ODVA to each DeviceNet product manufacturer. The vendor ID for National Instruments is 109 (6D hex).

Device Type

Status

This attribute represents the current status of the entire device. The Status attribute is a WORD, with the following bit definitions (table replicated from DNET spec).

State

This attribute is an indication of the present state of the device. The possible states for the device are shown in the following table.

Device States

State	Description
00 hex	Nonexistent
01 hex	Device Self Testing
02 hex	Standby
03 hex	Operational
04 hex	Major Recoverable Fault
05 hex	Major Unrecoverable Fault

Configuration Consistency Value

The Configuration Consistency Value (CCV) keeps the cyclic redundancy checksum (CRC) of all configuration attributes that are stored in the non-volatile memory. If any of the non-volatile parameters are changed, the CCV value is recalculated to reflect the change.

Common Services

Service Code	Implemented For		Service Name	Service Description
	Class	Instance		
0x05	Yes	No	Reset	Invokes the reset service for the device
0x0E	Yes	Yes	Get Attribute Single	Returns the contents of the specified attribute

Reset Service

The reset service is used to reset the FP-1100 and all I/O modules attached to it. The behavior of the Reset service is the same for both class (Instance ID 0) and instance (Instance ID 1) levels of the Identity Object.

Request Service Data

Name	Data Type	Valid Values	Description
Reset Type	USINT	0	Emulates power cycle
		1	Returns to the out-of-box configuration and then emulates power cycle

Success Response Service Data
No service data is returned if the service is successful.

Possible Error Codes from the Reset Service

Error Condition	General Error Response	General Error Code	Additional Error Code
The reset type requested in the service is not supported by this object	Invalid Parameter	20 hex	01 hex

Get Attribute Single Service
This service returns the contents of the attribute specified in the service data.

Request Service Data

Name	Data Type	Valid Values	Description
Attribute ID	USINT	1 – n, where n is the maximum attribute Id supported by the object	Identity of the attribute to be read/returned

Success Response Service Data
The format of the response data is dictated by the data type of the attribute referred by the Attribute ID in the service request.

Possible Error Codes from the Get Attribute Single Service
Any error code listed in “General Error Codes“ table can be returned from this service. There are no service specific error codes for this service.

Message Router Object

The Message Router Object routes explicit messages to appropriate target objects. In addition, it provides a list of objects and connections supported by the device. The FP-1100 contains one instance of the Message Router Object.

Class Attributes

Attribute ID	Applicable Services	Name	Data Type	Default Value	Description of Attribute
1	Get	Revision	UINT	1	Revision of this object

Instance Attributes

Attribute ID	Applicable Services	Name	Data Type	Default Value	Description of Attribute
1	Get	Object List	Struct of		List of the objects supported by the FP-1100
		Number of Objects	UINT		Number of supported classes in the class codes array
		Class Codes	Array of UINT		List of supported class codes
2	Get	Number Available	UINT		Maximum number of connections supported
3	Get	Number Active	UINT		Number of connections currently active
4	Get	Active Connections	Array of UINT		List of the connection IDs of the currently active connections

Common Services

Service Code	Implemented For		Service Name	Service Description
	Class	Instance		
0x0E	Yes	Yes	Get Attribute Single	Returns the contents of the specified attribute

Get Attribute Single Service

This service returns the contents of the attribute specified in the service data.

Request Service Data

Name	Data Type	Valid Values	Description
Attribute ID	USINT	1 – 4	Identity of the attribute to be read/returned

Success Response Service Data
The format of the response data is dictated by the data type of the attribute referred by the Attribute ID in the service request. The size of the array for the array attributes can be determined from the attribute preceding the array attribute. For example, the size of the *Active Connections* array is equal to the value of the *Number Active* attribute.

Possible Error Codes from the Get Attribute Single Service
Any error code listed in "General Error Codes" table can be returned from this service. There are no service specific error codes for this service.

DeviceNet Object

The DeviceNet Object maintains configuration and status of physical attachments to the DeviceNet network. It also allocates and releases connection instances associated with the Predefined Master/Slave Connection Set. The FP-1100 contains one instance of the DeviceNet Object.

Class Attributes

Attribute ID	Applicable Services	Name	Data Type	Default Value	Description of Attribute
1	Get	Revision	UINT	1	Revision of this object

Instance Attributes

Attribute ID	Applicable Services	Name	Data Type	Default Value	Description of Attribute
1	Get	MAC ID	USINT	Dip switch (1-6)	Node Address (0-63)
2	Get	Baud Rate	USINT	Dip switch (7-8)	Baud rate of the device (0=>125K, 1=>250K, 2=>500K)
3	Get	BOI	BOOL		Bus-Off Interrupt
4	Get	Bus-Off Counter	USINT		Number of times CAN chip went to the bus-off state
5	Get	Allocation Information	Struct of:		Allocation information of the Predefined Master/Slave connection set
		Allocation Choice Byte	BYTE		
		Master's MAC ID	USINT		

Common Services

Service Code	Implemented For		Service Name	Service Description
	Class	Instance		
0x0E	Yes	Yes	Get Attribute Single	Returns the contents of the specified attribute
0x4B	Yes	No	Allocate Master/Slave connection set	
0x4C	Yes	No	Release Group 2 Identifier set	

Service Description

The allocation/release scheme used for the Predefined Master/Slave connection set is based on the *DeviceNet Specifications*, version 2.0. Refer to Chapter 5, Section 5-5.4.2 for details.

I/O Point Object

(Vendor Specific – Class ID = 0x64)

This I/O Point Object models the class of general I/O points that have similar behavior regardless of the actual I/O point's field signal characteristics. The I/O point can be an input, an output, or a combination point. The width of the point can also vary from a single bit (discrete digital point) to a multi-valued (analog, counter, etc) point. The FP-1100 uses the I/O Point Object as its main application object.

The number of I/O Point Object instances present in the FP-1100 at any given time depends on the number of I/O modules connected to the FP-1100 network module. For example, if there are 4 I/O modules connected, each with 8 physical channels, then the FP-1100 will create 32 instances of the I/O Point Object.

State Transition Diagram

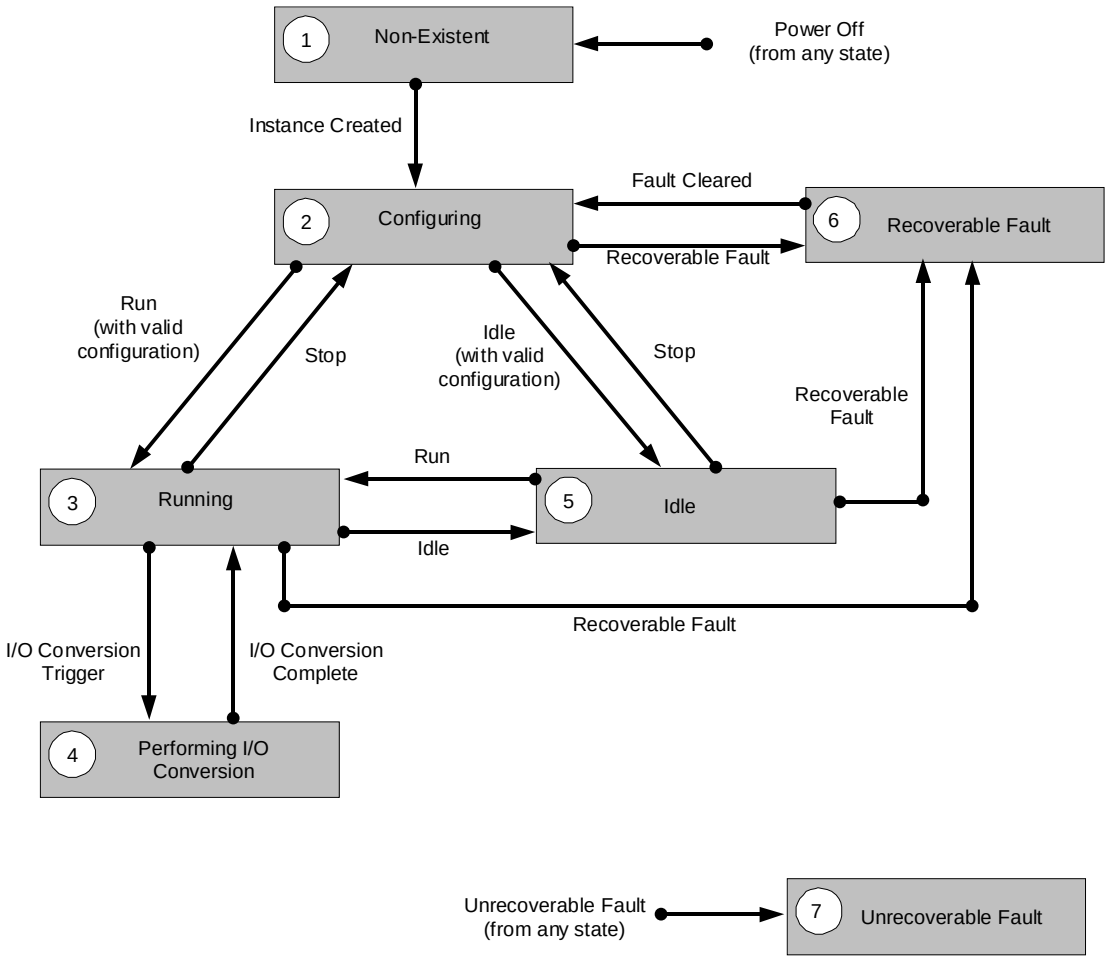


Figure 1: I/O Point's State Transition Diagram [2]

Class Attributes

Attribute ID	Applicable Services	Name	Data Type	Default Value	Description of Attribute
1	Get	Revision	UINT	1	Revision of this object
2	Get	Max Instance	UINT		Maximum Instance number of the object in this class

Instance Attributes

Attribute ID	Applicable Services	Name	Data Type	Default Value	Description of Attribute
3	Set Get	Value	Based on Current Range attribute (2 bytes)		Current value of the signal on the field terminals
4	Get	Status	BOOL		Status of the I/O Point (0=Good; 1=Fault)
5	Get	Fault Code	UINT		Code that describes the fault when Status is true
7	Get	State	USINT		Current state of the instance
8	Get	I/O Type	USINT		Type of I/O at the field
9	Get	Module Number	USINT		Physical hardware module containing the I/O point
10	Get	Channel Number	USINT		Physical hardware channel associated with the I/O point object instance
11	Get	Module Name	SHORT STRING		Module name associated with the I/O module
12	Set	COS Trigger Delta	USINT	0	Percent deadband for Analog Input Points
13	Set/Get Save/Restore	Idle Action	USINT	0	Action to be performed while in the Idle state
14	Set/Get Save/Restore	Idle Value	Based on Current Range attribute value (2 bytes)		Value an output will assume while in the Idle state when Idle Action is set to 1
15	Set/Get Save/Restore	Fault Action	USINT	0	Action to be performed while in the Fault state
16	Set/Get Save/Restore	Fault Value	Based on Current Range attribute value (2 bytes)		Value an output will assume while in the Fault state when Fault Action is set to 1

100	Get	Number Of Ranges	USINT		Number of ranges supported by the I/O Point
101	Get	Range Descriptor	ARRAY of *RANGE TYPE**		List of ranges supported by the I/O Point (see semantics for details)
102	Get Set Save/Restore	Current Range	*RANGE TYPE*		Current range setting of the I/O Point
103	Get	Number of Attributes	USINT		Number of channel specific attributes present in the I/O point
104	Get Get Member Set Member Save/Restore	Attribute Descriptor	ARRAY of STRUCT: {USINT Member Id *CH_ATTR TYPE** attrDesc}		List of attributes supported by the I/O Point (see semantics for details)
105	Get	Number of Commands	USINT		Number of channel specific commands present in the I/O point
106	Get Get Member Set Member Save/Restore	Command Descriptor	ARRAY of STRUCT: {USINT Member Id *CH_CMD TYPE** CmdDesc}		List of commands supported by the I/O Point (see semantics for details)
107	Get	Assembly Member Information	USINT: Assembly Mask ARRAY of UINT: MemberIds		
108	Get Member	List of Enumerated Attribute Settings' Names	USINT MemberId UINT32 AttributeId ARRAY OF STRUCT: {USINT EnumeratedSettingsId SHORTSTRING EnumeratedSettingsName}		
109	Get	Channel Information Structure Id	UINT		FieldPoint specific

* See semantics section for description of structure parameters

Semantics

Current Value

This attribute represents the current value on the field terminals of an I/O Point instance. The data type and range for this attribute is based on the Current Range (Attribute ID 20) configuration for the I/O instance.

The access rules of this attribute are dictated by the direction of the I/O point. If the I/O point is an output point, then both Set and Get operations are supported. For input points, only Get service is supported. The direction of the I/O point can be determined from the I/O type attribute.

The value of this attribute can be obtained by using an I/O connection (via an input assembly), or by calling **Get_Attribute_Single** service on the I/O point. For an output point, the value can be changed through an I/O connection (via an output assembly) or by calling a **Set_Attribute_Single** on this attribute while the I/O point is in the Configuration state. In any other I/O state, the Set service returns an error. For more information on the I/O assemblies, see the Assembly Object section, or the examples given later in this chapter.

Status

This attribute presents the current operational status of the I/O point. The status is returned as a Boolean value for efficient placement into I/O connection messages (via an input assembly).

If the instance is in the Configuring state (2), Idle state (3), Running state (4), or Performing I/O Conversion state (5) this attribute is False (0).

If the instance is in the Recoverable Fault state (6) or Unrecoverable Fault state (7), this attribute is True (1). When this attribute indicates a fault, attribute ID 5 (Fault Code) provides an error code that describes the fault.

Fault Code

When attribute ID 4 (Status) is True (1), this attribute provides an error code that describes the fault.

Fault codes are grouped using the fault categories for the Identity Object Status attribute (ID 5). The following working definitions are used for each fault category:

> ***Minor Recoverable Fault*** A fault has been detected which is considered *recoverable*. The fault does not prevent I/O conversion for the I/O Point. The signal on the field terminals is still converted to/from the Value attribute (ID 3). This fault does not cause a state change for the I/O Point.
>
> ***Minor Unrecoverable Fault*** A fault has been detected which is considered *unrecoverable*. The fault does not prevent I/O conversion for the I/O Point. The signal on the field terminals is still converted to/from the Value attribute (ID 3). This fault does not cause a state change for the I/O Point.

Major Recoverable Fault A fault has been detected which is considered *recoverable*. The fault prevents I/O conversion for the I/O Point. The signal on the field terminals is determined by the Fault Action attribute (ID 21). This fault causes the I/O Point to transition to the Recoverable Fault state.

Major Unrecoverable Fault A fault has been detected which is considered *unrecoverable*. The fault prevents I/O conversion for the I/O Point. If possible, the signal on the field terminals is determined by the Fault Action attribute (ID 21). This fault causes the I/O Point to transition to the Unrecoverable Fault state.

State

The State attribute describes the current state of the I/O point instance. The possible states of operation are illustrated in the State Transition Diagram (Figure 1). The value reported is the numerical equivalent of that shown on the State Transition Diagram.

I/O Type

This attribute specifies the type of physical I/O hardware associated with the I/O Point instance. Its value represents the fundamental characteristics of the signal at the field terminals.

Possible Values for I/O Type Attribute

I/O Type	Description
0	Discrete input
1	Discrete output
2	Analog input
3	Analog output
4	Counter Input
5	Counter Output

Module Number

This attribute identifies the physical hardware module containing the I/O point. The module number is the offset of the I/O module from the FP-1100 network module. For example, an I/O module placed next to the network module will have a Module Number of 1.

Channel Number

This attribute identifies the physical channel of the I/O Point instance within the module. This number indicates a physical instance for the field terminals associated with this I/O Point instance (logical instance).

Module Name

This attribute identifies the module name of the I/O module. This name is the same as one shown on the physical I/O module containing this I/O point.

Idle Action

This attribute specifies the action to be performed by an output point while it's in the Idle state.

Idle Action Values

Idle Action	Description
0	Hold last value (default)
1	User specified value in Idle Value (Attribute ID 20)

Idle Value

This attribute specifies the value that an output point will assume while in the Idle state when Idle Action (attribute ID 13) is 1 (User value).

Fault Action

This attribute specifies the action to be performed while in the Recoverable Fault state (and if possible, the Unrecoverable Fault state). For output points, this attribute determines the value placed on the field terminals while in a Fault state.

Fault Action Values

Fault Action	Description
0	Hold last value (default)
1	User specified value in Fault Value (attribute ID 22)

Fault Value

This attribute specifies the value that an output point will assume while in the Recoverable Fault state (and if possible, the Unrecoverable Fault state) when attribute ID 15 (Fault Action) is 1 (User value).

COS Trigger Delta

This attribute specifies the amount by which the Value attribute must change before a Change Of State (COS) event is triggered. The Change Of State event is used to trigger message production for a COS I/O connection.

This attribute applies only to analog input points that are or may become members of a COS output assembly. For all other points, the value of this attribute is ignored. The special value of zero for the COS Trigger Delta means that any change in the Value attribute triggers a COS event.

Each time an I/O conversion is performed, the new input value is compared to the most recent value produced on the COS I/O connection. The production used for comparison can be due to a COS event or an EPR expiration, whichever occurred most recently. If the difference between the two values is greater than or equal to the COS Trigger Delta attribute, a new COS event is triggered, and the new value is produced on the COS I/O connection.

Number of Ranges

Each I/O point supports one or more range Ids depending on its type. This attribute returns the number of ranges supported by a particular I/O point. The length of the Range Descriptor is based on this number. For discrete points, this number is always 1. For analog points, this number can be anything between 1 and 8.

Range Descriptor

In most I/O Point instances, the underlying hardware module dictates a fixed set of valid range configurations. This attribute provides an array of descriptors for such range configurations. Each range descriptor documents a specific data type, engineering units, physical range, and logical range combination for use with the I/O Point instance. A Get on this attribute returns all the valid range settings for the I/O point.

The Range Descriptor may differ from one I/O point to the other within the same module. For example, if an I/O module supports both analog and discrete I/O points, the Range Descriptor for analog points will be different than the one for the discrete points.

Range Descriptor

This attribute provides an array of descriptors that are valid for the I/O point. Each element in the Range Descriptor array is represented in the following structure format.

```
STRUCT RANGE_TYPE
{
   USINT         RangeId;
   ENGUNIT       EngineeringUnits;
   USINT         PhysicalDataType;
   UINT32        LowPhysicalValue;   //4 bytes
   UINT32        HighPhysicalValue;  //4 bytes
   USINT         LogicalDataType;
   UINT          LowLogicalValue;    //2 bytes
   UINT          HighLogicalValue;   //2 bytes
 }
```

The physical range documents the range of physical values on the field terminals, and the logical range documents the corresponding range for the Value attribute (ID 3). Since the relationship between the physical and logical range is linear, the DeviceNet Value attribute can be converted to/from its real-world representation in a straightforward manner.

For an input/combination point, the following formula can be used to convert the Value attribute into its scaled physical representation:

```
Percent = (Value – LowLogValue) / (HighLogValue – LowLogValue);

ScaledValue = (Percent * (HighPhysValue – LowPhysValue)) +
LowPhysValue;
```

For an output point, the following formula can be used to convert the scaled physical representation into the Value attribute:

```
Percent = (ScaledValue – LowPhysValue) / (HighPhysValue –
LowPhysValue);
```

```
Value = (Percent * (HighLogValue – LowLogValue)) + LowLogValue;
```

RangeId The Range Id field in the Range Descriptor structure provides a unique identification number for a particular range. If an I/O point supports more than one range, then this is the number that is sent in the Set_Attribute_Single request service to change the Current Range attribute (see next section).

EngineeringUnits This field provides the engineering units for the physical signal on the field terminals.

The encoding for this field is specified in Appendix K, *Engineering Units*, of the DeviceNet Specification. For discrete input/output points, the engineering units are specified as Bit (0005 hex), physical and logical data type are both BOOL, and the physical and logical range are both 0-1. The voltage on the field terminals is determined by the actual power supply connected to the I/O hardware.

The following table lists the engineering units that may be returned from an I/O point instance.

Engineering Units for an I/O Point

Engineering Unit	Description
0x1001	count
0x1005	boolean
0x1007	percent
0x1200	degree Celsius
0x1201	degree Fahrenheit
0x1202	kelvin
0x1C02	milliamp
0x1D00	volts
0x1D01	millivolts
0x2800	Ohm
0x0810	counts per micro second

PhysicalDataType This field provides the DeviceNet data type for the physical range values (LowPhysicalValue and HighPhysicalValue). This data type is not used for actual representation of the point in DeviceNet objects and messages. The physical data type represents how the signal value is viewed in terms of its real-world values.

For example, analog values use a physical data type of REAL (CA hex), discrete values use BOOL (C1 hex), and other values such as counters use DINT (C4 hex).

LowPhysicalValue This field provides the lowest valid signal level for the field terminals. This signal level is not necessarily the lowest physical signal possible, but it is the physical signal that corresponds to the LowLogicalValue field. The PhysicalDataType field determines the data type of this field

For example, if the I/O Point were a 4-20 mA analog input, this field would often report the REAL value 4.0.

HighPhysicalValue This field provides the highest valid signal level for the field terminals. This signal level is not necessarily the highest physical signal possible, but it is the physical signal that corresponds to the HighLogicalValue field. The PhysicalDataType field determines the data type of this field.

For example, if the I/O Point were a 4-20 mA analog input, this field would often report the REAL value 20.0.

LogicalDataType This field provides the DeviceNet data type for the Value attribute (ID 3) and related attributes. This data type also applies to the logical range values (LowLogicalValue and HighLogicalValue). This data type is used for all representations of the point in DeviceNet objects and messages.

For example, although analog values are often thought of in terms of REAL numbers, most analog conversion hardware uses INT numbers, and therefore the Value attribute often uses the INT data type.

LowLogicalValue This field provides the lowest valid value for the Value attribute (ID 3) and related attributes. This field, in conjunction with the other range fields, describes the scaling necessary to properly interpret the Value attribute. The LogicalDataType field determines the data type of this field.

For example, if the I/O Point is a 4-20 mA analog input, this field might report the INT value 0000 hex.

HighLogicalValue This field provides the highest valid value for the Value attribute (ID 3) and related attributes. This field, in conjunction with the other range fields, describes the scaling necessary to properly interpret the Value attribute. The LogicalDataType field determines the data type of this field.

For example, if the I/O Point is a 4-20 mA analog input, this field might report the INT value 0FFF hex.

Please refer to Appendix B for a comprehensive list of ranges supported by each FieldPoint I/O module.

Current Range

This attribute contains the current range configuration of the I/O point instance. This attribute can be set to any element present in the Range Descriptor array for the I/O instance. A Get service on this attribute returns the current range descriptor in the same format as the Range Descriptor array element.

For I/O points that support multiple ranges, this attribute can be set to change the I/O range. For Set_Attribute_Single service, the service data only requires the Range Id of the desired range. If an I/O point supports only one value for Range Id, the **Set_Attribute_Single** will not change this attribute.

Number of Attributes

This attribute contains the number of elements in the Attribute Descriptor array. An Attribute Descriptor provides access to a configurable parameter in the physical I/O hardware that the I/O

point instance is representing. If an I/O point doesn't support any configurable parameter, this number is zero.

Attribute Descriptor

The Attribute Descriptor array contains the list of all the configurable attributes belonging to the I/O Point instance.

FieldPoint I/O modules support a wide variety of physical channels ranging from discrete outputs to temperature sensors. The *Attribute Descriptor* array allows a generic way to deal with this wide range of I/O types with the same I/O Point Object. Each element in this array contains a list of setting Ids that are valid for the configurable attribute for the physical channel that the I/O point instance is representing. Changing the setting Id over the DeviceNet network changes the value of that attribute in the I/O channel. For example, the thermocouple type of a channel can be changed from J to K by sending the appropriate setting Id to the I/O point representing the thermocouple channel.

An element in the Attribute Descriptor array can be accessed/changed by calling **Get_Member/Set_Member** service on the I/O point instance. The whole array is also accessible through **Get_Attribute_Single** service, as long as its length doesn't exceed the output buffer size. If the Number of Attributes is zero, the **Get_Attribute_Single** service will return an empty list.

Each element in the Attribute Descriptor array is represented in the following structure format:

```
STRUCT  ATTR_TYPE
{       UINT32                  AttrId
        SHORT_STRING            AttrName
        USINT                   Length
        USINT                   AttrType
        If (AttrType == ENUMERATED)
        {
                USINT           NumSetttings
                USINT           SettingIDs [NumSettings]
        }
        else
        If (AttrType == INTEGER)
        {
                UINT32          MinSetting
                UINT32          MaxSetting
        }

        UINT32                  DefaultSetting
        UINT32                  CurrentSetting
}
```

AttrId This field specifies a unique number to identify the configurable attribute. Any I/O point that supports this attribute will have the same *AttrId*. This number, along with the member Id and the new setting Id, is sent in the **Set_Member** service data field to change the attribute setting.

AttrName This field provides a textual description of the attribute.

AttrType The *AttrType* field defines the format of the next two fields of the structure. If the type is enumerated (0), then a list of enumerated settings is provided in the *SettingIDs* field. The value of the attribute can be set to any value that is present in the *SettingIDs* array. If the type is integer (1), then a minimum and a maximum value for the attribute are provided. In this case, the attribute can take any value within those bounds.

NumSettings If the *AttrType* is enumerated (0), then this field contains the number of enumerated settings the attribute supports. If the *AttrType* is integer (1), then this field doesn't exist.

SettingIDs If the *AttrType* is enumerated (0), then this array contains all the setting Ids that are valid for the attribute. The length of the array is equal to the *NumSettings* field. This field doesn't exist for integer type attributes.

MinSetting If the *AttrType* is integer (1), then this field replaces the *NumSettings* field of the structure. The value in this field represents the minimum setting the attribute can support.

MaxSetting If the *AttrType* is integer (1), then this field replaces the *SettingIDs* field of the structure. The value in this field represents the maximum setting the attribute can support.

DefaultSetting This field contains the default setting of the attribute. When the FieldPoint network powers up, the I/O point instance sets the physical channel's attribute setting to this default value.

CurrentSetting This field holds the actual setting of the attribute. This setting will be the same as the DefaultSetting, if **Set_Member** service has never been requested for the attribute.

Please refer to Appendix B for a comprehensive list of attributes supported by each FieldPoint I/O module.

Number of Commands

This attribute contains the number of elements in the Command Descriptor array. A Command Descriptor provides access to commands supported by the physical I/O hardware that the I/O point is representing. If an I/O point doesn't support any commands, this number is zero.

Command Descriptor

Some of the FieldPoint I/O channels support commands that allow the user to change their behavior. For example, a counter point can be cleared by sending it a reset command Id. The Command Descriptor array contains the list of all the commands that can be serviced by an I/O Point instance.

Just like an Attribute Descriptor array, a Command Descriptor array allows a generic way to perform commands on a wide range of I/O types with the same I/O Point Object. Each element in this array contains a list of command Ids that are supported by that I/O point. Sending the appropriate command Id to the I/O Point instance through **Set_Member** service results in the execution of the command by the corresponding physical I/O channel. If the I/O point doesn't support any commands, **Set_Member** service returns an error.

The format of the command descriptor structure is very much like the attribute descriptor structure, as shown below:

```
STRUCT CMD_TYPE
{       UINT32              CmdId
        SHORT_STRING        CmdName
        USINT               Length
        USINT               CmdType

        If (CmdType == ENUMERATED)
        {
                USINT       NumSetttings
                USINT       SettingIDs [NumSettings]
        }
        else
        If (CmdType == INTEGER)
        {
                UINT32      MinSetting
                UINT32      MaxSetting
        }

        UINT32              Reserved
}
```

CmdId This field specifies a unique number to identify the command. Any I/O point that supports this command will have the same *CmdId*. This number, along with the member Id and the new setting Id, is sent in the **Set_Member** service data field to when a new operation is requested.

CmdName This field provides a textual description of the command.

CmdType The *CmdType* field defines the format of the next two fields of the structure. If the type is enumerated (0), then a list of enumerated settings is provided in the *SettingIDs* field. Any value present in the *SettingIDs* array is a command supported by the I/O point instance. If the type is integer (1), then a minimum and a maximum value for the attribute are provided. In this case, the command can take any value within those bounds.

NumSettings If the *CmdType* is enumerated (0), then this field contains the number of enumerated settings the command supports. If the *CmdType* is integer (1), then this field doesn't exist.

SettingIDs If the *CmdType* is enumerated (0), then this array contains all the setting Ids that are valid commands. The length of the array is equal to the *NumSettings* field. This field doesn't exist for integer type commands.

MinSetting If the *CmdType* is integer (1), then this field replaces the *NumSettings* field of the structure. The value in this field represents the minimum setting the command can support.

MaxSetting If the *CmdType* is integer (1), then this field replaces the *SettingIDs* field of the structure. The value in this field represents the maximum setting the command can support.

Reserved Reserved for future use.

Please refer to Appendix B for a comprehensive list of commands supported by each FieldPoint I/O module.

Common Services

Service Code	Implemented For		Service Name	Service Description
	Class	Instance		
0x05	Yes	Yes	Reset	Invokes the reset service for the device
0x0E	Yes	Yes	Get Attribute Single	Returns the contents of the specified attribute
0x10	Yes	Yes	Set Attribute Single	Modifies the value of the specified attribute
0x15	Yes	Yes	Restore	Restores the specified instance or attribute to the last saved configuration
0x16	Yes	Yes	Save	Saves attribute(s) of the specified I/O instance(s) to non-volatile memory
0x1A	No	Yes	Get Member	Returns a member from the Range List, Attribute Descriptor, or the Command Descriptor arrays
0x1B	No	Yes	Set Member	Modifies a member in the Attribute Descriptor or the Command Descriptor

Reset Service

The reset service is used to reset one or more I/O Point objects. If invoked at the class level, all the I/O points will get reset. If invoked on an individual I/O Point, then only the specified I/O point instance will be reset.

Request Service Data

Name	Data Type	Valid Values	Description
Reset Type	USINT	1	Perform a hardware reset, then emulate as closely as possible cycling power for the I/O Point instance alone. If I/O point instance had some stored values, the I/O point will set its parameter to the saved configuration. The I/O point transitions to the Configuring state after the reset.
		2	Performs a hardware reset, then returns as closely as possible to the out-of-box configuration of the I/O Point instance. The I/O point goes to the Configuring state after the reset.
		3	Performs a hardware reset, then returns as closely as possible to the out-of-box configuration of the I/O Point instance. Also, clear any configuration stored for this I/O point. The I/O point goes to the Configuring state after the reset.

Success Response Data
No service data is returned if the service is successful.

Possible Error Codes from the Reset Service

Error Condition	General Error Response	General Error Code	Additional Error Code
Vendor specific error in reset	Vendor Specific error	1F hex	See Appendix C
The reset type requested in the service is not supported by this object	Invalid Attribute Value	09 hex	FF hex

Get Attribute Single Service
The Get Attribute Single service is supported for all attributes in the I/O point class.

Request Service Data

Name	Data Type	Valid Values	Description
Attribute ID	USINT	1 – n, where n is the maximum attribute Id supported by the object	Identity of the attribute to be read/returned

Success Response Data
If the service is successful, the value of the requested attribute is returned in the response data buffer. The length of the response buffer depends on the type of the attribute Id.

If an array attribute had been requested, then the number of the elements in that array can be determined from the attribute preceding the array attribute. For example, to know how many elements are present in the Attribute Descriptor array, first do a get on Number of Attributes and then call Get Attribute Single service on the Attribute Descriptor array Id.

To parse the array elements, refer to the structure format used by the corresponding array element.

Possible Error Codes from the Get Attribute Single service

Error Condition	General Error Response	General Error Code	Additional Error Code
Error in get	Vendor Specific error	1F hex	See Appendix C

Set Attribute Single Service

This service modifies the value of the attribute specified. The service can be applied to all the configurable attributes that are not represented as arrays of some structure type. For attributes of array data type, **Set Member** service is supported instead.

Request Service Data

Name	Data Type	Valid Values	Description
Attribute ID	USINT	1 – n, where n is the maximum attribute Id supported by the object	Identity of the attribute to be modified
Attribute Value	Attribute specific	Attribute specific	New value of the attribute. The size of this attribute depends on attribute's data type. For range setting, the Set service only requires the new range id in the data field

Success Response Data

No service data is returned if the service is successful.

Possible Error Codes from the Set Attribute Single service

Error Condition	General Error Response	General Error Code	Additional Error Code
The desired attribute value is invalid	Invalid attribute value	09 hex	01 hex
The desired range is not supported by this I/O point	Invalid attribute value	09 hex	02 hex
The actual attribute value is the same as the	Already in requested	0B hex	ff hex

desired attribute value	mode/state		
The object cannot perform a set operation in this state. The I/O point must be in its Configuring state to successfully set an attribute	Object state conflict	0C hex	= Current state

Restore Service

The Restore service restores the attributes to their last saved state. At the instance level, this service performs an internal restore of a specific I/O Point instance in the device. If invoked for the class, the Restore service restores all the I/O point objects to their last save configuration.

The Restore service can also be invoked for a particular attribute, in which case it will only restore the specified attribute of the I/O Point instance.

Request Service Data

The restore service doesn't require any service data if invoked at the class or instance level. If invoking this service at the attribute level, the attribute Id must be sent in the service data. Any attribute that supports **Set Attribute Single** or **Set Member** service and has been changed can be restored individually if its original value was saved before changing it.

Name	Data Type	Valid Values	Description
Attribute ID	USINT	1 – n, where n is the maximum attribute Id supported by the object	Identity of the attribute to be restored. **Only needed if the Restore service is requested to restore a particular attribute**

Success Response Service Data

No service data is returned if the service is successful.

Possible Error Codes from the Restore service

Error Condition	General Error Response	General Error Code	Additional Error Code
One (or more I/O objects if service requested for the class) could not be restored due to state conflict. The I/O point must be in it's Configuring state to successfully restore the saved attribute	Object state conflict	0C hex	= Current state
There is no saved data for the requested object	No stored attribute data	18 hex	FF hex
Vendor specific error in trying to restore the I/O class/point/attribute	Vendor specific error	1F hex	See Appendix C

Save Service

The Save service saves all the configurable I/O point attributes to the non-volatile memory for use during the next power-on sequence. This service should only be called if any of the attribute values have been changed with **Set Attribute Single** or **Set Member** service.

The Save service can be requested for an I/O Point attribute, instance, or the whole I/O point class. At the attribute level, the Save service will only save the current setting of the specified attribute to the non-volatile memory. If invoked for a particular I/O instance, all the attributes of that I/O instance will be saved. At the class level, the Save instance service will save all the I/O Points as well as the I/O assembly configuration. A save on the class is required before a save on an instance or attribute can be requested.

Request Service Data

The Save service doesn't require any service data if invoked at the class or instance level. If invoking this service at the attribute level, the attribute Id must be sent in the service data. Any attribute that supports **Set Attribute Single** or **Set Member** service can be saved individually.

Name	Data Type	Valid Values	Description
Attribute ID	USINT	1 – n, where n is the maximum attribute Id supported by the object	Identity of the attribute to be saved. Only needed if the Save service is requested to restore a particular attribute

Success Response Service Data

No service data is returned if the service is successful.

Possible Error Codes from the Save Service

Error Condition	General Error Response	General Error Code	Additional Error Code
Save on class never called	Privilege violation	0F hex	01 hex
The requested data was not saved due to a failure during the attempt	Store operation failure	19 hex	01 hex
The requested data was not saved due to an error in writing to the non-volatile memory	Store operation failure	19 hex	02 hex
Other errors	Vendor Specific error	1F hex	

Get Member Service

The Get Member Service returns a member at the specified member Id from the Attribute Descriptor or the Command Descriptor array.

Request Service Data

Name	Data Type	Valid Values	Description
Attribute ID	USINT	104 (Attribute Descriptor array) 106 (Command Descriptor array)	I/O points attribute Id
Member ID	WORD	1 – ***n*** (where ***n*** is the Number of Attributes in the Attribute Descriptor array or Number of Commands in the Command Descriptor array)	The member Id of the desired element in the array The most significant bit of this field must be equal to 0, since extended member protocol is not supported for this service.

Success Response Service Data

Name	Data Type	Description
Member ID	WORD	The member Id of the desired element in the array The most significant bit of this field must be equal to 0, since extended member protocol is not supported for this service.
Member Data	STRUCT of CMD_TYPE or ATTR_TYPE	If the member is an element of the Attribute Descriptor array, then the member data is in the Attribute Descriptor format. If the member is an element of the Command Descriptor array, then the member data is in the CMD_TYPE format. In either case, the parsing of the structure has to be done based on the type (AttrType or CmdType) of the structure.

Possible Error Codes from the Get Member Service

Error Condition	General Error Response	General Error Code	Additional Error Code
Get operation failed internally	Vendor Specific error	1F hex	See Appendix C
Requested member ID (either 0 or greater than array size) does not exist	Invalid Member ID	28 hex	01 hex
Extended protocol not supported for this member service	Invalid Member ID	28 hex	02 hex

Set Member Service

The Set Member service is used to change the setting Id of an element in the Attribute Descriptor array or to send a new command Id to a Command Descriptor array element.

The Set Member service requires the attribute/command Id and the new setting Id along with the member Id of the Descriptor array element in the service data field. The extended member service protocol is not supported for the I/O Point Set Member service.

Request Service Data

Name	Data Type	Valid Values	Description
Attribute Id	USINT	104 (Attribute Descriptor array) 106 (Command Descriptor array)	I/O points attribute Id
Member ID	WORD	1 – ***n*** (where ***n*** is the Number of Attributes in the Attribute Descriptor array and Number of Commands in the Command Descriptor array)	The member Id of the element to be set. The most significant bit of this field must be equal to 0, since extended member protocol is not supported for this service.
Element Id	UINT32		The element Id is the unique number for that element. For the Attribute Descriptor, this parameter is the same as the ***AttrId*** and for the Command Descriptor, this is equal to the ***CmdId***.
New Setting	UINT32		The desired setting for the command or the attribute. For an enumerated descriptor, this value has to match one of the values in the ***SettingIDs*** list. For an integer descriptor type, this value can be any number between the Min and Max setting.

Ideally a **Get Member** service should be requested before the **Set Member** service is invoked, to obtain the attribute or command Id and a valid setting Id. This information is also accessible through **Get Attribute Single** service on the descriptor array. The member Ids are given to the array members according to placement in the array. For example, the first member in the array will have Member Id of 1 and the third element will have a Member Id of 3.

Success Response Service Data

No service data is returned if the service is successful.

Possible Error Codes from the Set Member Service

Error Condition	General Error Response	General Error Code	Additional Error Code
The desired attribute or command setting does not match any of the elements in the ***SettingIDs*** list	Invalid attribute value	09 hex	01 hex

The desired attribute or command setting is smaller than the ***MinSetting*** allowed by the I/O channel	Invalid attribute value	09 hex	02 hex
The desired attribute or command setting is greater than the ***MaxSetting*** allowed by the I/O channel	Invalid attribute value	09 hex	03 hex
Internal failure of the set operation	Vendor Specific error	1F hex	See Appendix C
Requested member ID (either 0 or greater than array size) does not exist	Invalid Member ID	28 hex	01 hex
Extended protocol not supported for this member service	Invalid Member ID	28 hex	02 hex

Connection Object

(Class ID = 0x05)

The Connection Class allocates and manages the internal resources associated with I/O and Explicit Messaging Connections. An I/O connection provides dedicated, special purpose communication paths between a producing application and one or more consuming applications. The Explicit Messaging Connections provide generic, multi-purpose communication paths between two devices. Explicit Messages provide the typical request/response oriented network communications, whereas I/O connections move application specific I/O data across the network.

The FP-1100 is a UCMM (Unconnected Message Manager) capable device that also supports the Poll, Change of State (COS), Cyclic, and Explicit Messaging objects of the Predefined Master/Slave Connection Set.

The attributes, services, and behavior of the Connection Class are defined in the DeviceNet Specification. The following tables list the attributes and services supported by the FP-1100. For details on the semantics and behavior of these attributes, refer to Volume 1, Chapter 5, of the *DeviceNet Specifications*.

Class Attributes

Attribute ID	Applicable Services	Name	Data Type	Default Value	Description of Attribute
1	Get	Revision	UINT	1	Revision of this object

Instance Attributes

Attribute ID	Applicable Services	Name	Data Type	Default Value	Description of Attribute
1	Get	State	USING		State of the object
2	Get	Device Type	UINT	1	Indicates either I/O or Explicit

					Messaging Connection
3	Set/Get	Transport Class Trigger	USINT		Defines server or client behavior
4	Get	Produced Connection ID	UINT		Arbitration Id of the CAN message transmitted by the connection
5	Get	Consumed Connection ID	UINT		Arbitration Id of the CAN message received by this connection
6	Set/Get	Initial Communication Characteristics	USINT		Defines the message groups across which production and consumptions associated with this connection occur
7	Get	Produced Connection Size	UINT		Maximum number of bytes transmitted across this connection
8	Get	Consumed Connection Size	UINT		Maximum number of bytes received across this connection
9	Set/Get Reset	Expected Packet Rate	UINT		Defines timing associated with this connection
10-11	Reserved				
12	Set/Get	Watchdog Timeout Action	USINT		Defines how to handle Inactivity/Watchdog timeouts
13	Get	Produced Connection Path Length	UINT		Number of bytes in the produced connection path attribute
14	Get	Produced Connection Path	EPATH		Specifies the Application Object whose data is to be produced by this Connection Object
15	Get	Consumed Connection Path Length	UINT		Number of bytes in the consumed connection path attribute
16	Get	Consumed Connection Path	EPATH		Specifies the Application Object whose data is to be consumed by this Connection Object
17	Set/Get	Production Inhibit Time	UINT		Defines minimum time between new data production

Common Services

Service Code	Implemented For		Service Name	Service Description
	Class	Instance		
0x05	Yes	No	Reset	Invokes the reset service for the device
0x0E	Yes	Yes	Get Attribute Single	Returns the contents of the specified attribute

0x10	No	Yes	Set Attribute Singe	Modified the value of the attribute specified
0x09	Yes	Yes	Delete	Deletes a connection object and releases all associated resources

Assembly Object

The Assembly Object binds attributes of multiple objects, allowing data to or from each object to be sent over a single I/O connection. For the FP-1100, the Assembly Object is used to bind I/O point's Value and Status attributes in the input and output assemblies that are consumed/produced over the I/O connections.

The FP-1100 creates three Assembly Object instances on power-up. These instances are linked to one of the I/O Connection Objects (COS, cyclic, or poll) and remain active for the life of the connection. The Assembly Object's member list is pseudo-dynamic; that is, the members can be added or deleted when the assembly instance is inactive (corresponding I/O connection is not in the Established state), but a new assembly instance cannot be created. The Assembly object's instances are in the vendor specific range (0x64 – 0x66), where instance 0x64 corresponds to the consuming I/O assembly for I/O connection instances 2 and 4, instance 0x65 points to the producing assembly for the I/O connection instance 2 (Poll), and instance 0x66 points to the producing assembly for the I/O connection instance 4 (COS/Cyclic).

The following tables list the attributes and services supported by the Assembly Objects that reside in the FP-1100. For more information on these parameters, refer to *DeviceNet Specifications* (Volume II, Chapter 6).

State Transition Diagram

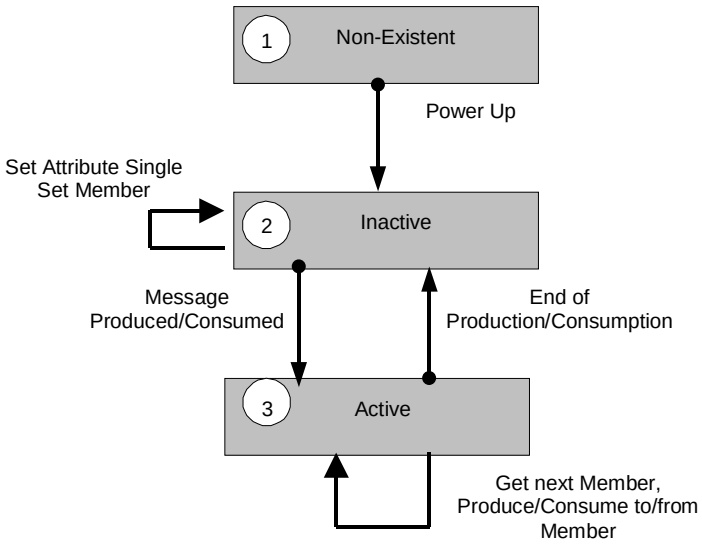


Figure 2: State Transition Diagram for Assembly Object [1]

Class Attributes

Attribute ID	Applicable Services	Name	Data Type	Default Value	Description of Attribute
1	Get	Revision	UINT	2.0	Revision of this object
2	Get	Max Instance	UINT	3	Maximum Instance number of the object in this class

Instance Attributes

Attribute ID	Applicable Services	Name	Data Type	Default Value	Description of Attribute
1	Get	Number of Members in List	UINT		Current number of members present in the Member List
2	Set Member Get Member Get	Member List	Array of Struct :		
		Member Data Description	UINT		Size of member data in bits
		Member Path Size	UINT	6	Size of member path (in bytes)
		Member Path	EPATH		
3	Get	Data	ARRAY of BYTE		Data values of the members
100 (64 hex)	Set	Include Channel status	BOOL	TRUE	Include channel status for producing assembly

Semantics

See *DeviceNet Specifications* (vol II) for description of attributes 1 to 3.

Include I/O Status

This attribute controls the inclusion of channel status information in the producing assemblies. If this attribute is set to true (default), the last [I/O Point Max Instance/8] bytes of the assembly represent channel status for each I/O channel. The LSB of the first byte in this sequence points to

the *Status* attribute of I/O instance 1 and the MSB of the last byte points to the *Status* attribute of the last I/O instance.

To remove the channel status information from the assembly, set this attribute to FALSE via a **Set_Attribute_Single** request. This attribute is only applicable to producing assembly instances (0x65 & 0x66). A **Set_Attribute_Single** request for the consuming instance (0x64) will return an error.

Common Services

Service Code	Implemented For		Service Name	Service Description
	Class	Instance		
0x0E	Yes	Yes	Get Attribute Single	Returns the contents of the specified attribute
0x10	No	Yes	Set Attribute Single	Modifies an attribute value
0x18	No	Yes	Insert Member	Adds a member to the Assembly Object's member list
0x19	No	Yes	Remove Member	Removes a member from the Assembly Object's member list
0x1A	No	Yes	Get Member	Returns a member from the member list

Get Attribute Service

This service returns the contents of the attribute specified in the service data.

For the Member List attribute, the Get service is only supported if the array fits the explicit message buffer. If the list is too long, a Reply Data Too Large (11 hex) error will be returned. In that case, a Get Member service can be used to get all the members from the array list. In either case, a **Get_Attribute_Single** service should be called on attribute 1 (Number of Member in the Member List), before calling a get on the Member List attribute.

Request Service Data

Name	Data Type	Valid Values	Description
Attribute ID	USINT	1, 2, 3, 100	Identity of the attribute to be read/returned

Success Response Service Data

The format of the response data is dictated by the data type of the attribute referred by the Attribute ID in the service request.

Possible Error Codes from the Get Attribute Single Service

Any error code listed in the "General Error Codes" table can be returned from this service. There are no service specific error codes for this service.

Set Attribute Service

The Set Attribute Service is only supported for the Include I/O Status attribute, that enables or disables inclusion of channel status in the output assembly.

Request Service Data

Name	Data Type	Valid Values	Description
Attribute ID	USINT	100	Identity of the attribute to be read/returned
Service Data	USINT	0 or 1	New attribute value

Possible Error Codes from the Get Attribute Single Service

Any error code listed in the "General Error Codes" table can be returned from this service. There are no service specific error codes for this service.

Insert Member Service

This service inserts new member(s) in the member list of the Assembly Object. The service adds the new member(s) at the specified Member ID of the member list. The Member IDs of members following the specified Member ID will change. For output assemblies (producing assemblies), the Member ID **1** is reserved and cannot be changed or deleted by the user.

If the Member ID is greater than the last Member ID in the member list, the new member is inserted at the end of the list.

Request Service Data

Name	Data Type	Valid Values	Description
Attribute ID	USINT	2 (Member List)	Attribute Id of the member list
Member ID/ EX	WORD	1 – n, for input assembly 2 – n, for output assembly n is the maximum number of members the list can have	The lower 15 bits identify the Member ID of the new member. The most significant bit (bit 15 - EX) defines the basic (0) or extended (1) message format. The extended format is not supported for this service.
Member Data	EPATH	Class ID = I/O_PT_CLASS_ID (always) Instance ID = 1–144 Attribute ID = 3 (value) or 4 (status)	The class, instance, and attribute Ids of the new members

Success Response Service Data

Name	Data Type	Description
Member ID	WORD	The member ID of the member serviced

Possible Error Codes from the Insert Member Service

Error Condition	General Error Response	General Error Code	Additional Error Code
The member list has reached its capacity. No more members can be inserted, without removing some other member(s)	Resource unavailable	02 hex	01 hex
The requested member is already present in the list (the member ID of the member may be different from the specified member ID).	Already in requested mode/state	0B hex	01 hex
The requested service cannot be performed in this state. The Assembly Object has to be in its inactive state (that is corresponding I/O connection is not Established) to modify the member list.	Object state conflict	0C hex	01 hex
Requested member ID (0) is invalid.	Invalid Member ID	28 hex	01 hex
The requested member ID (1 or 2) is reserved for this instance of the Assembly Object.	Invalid Member ID	28 hex	03 hex
Extended protocol not supported	Invalid Member ID	28 hex	04 hex

Delete Member Service

This service deletes member(s) from the member list. The Member IDs of the member(s) that follow the removed member(s) will change.

Request Service Data

Name	Data Type	Valid Values	Description
Attribute ID	USINT	2 (Member List)	Attribute Id of the member list
Member ID/ EX	WORD	1 – n, for consuming assembly 2 – n, for producing assembly n is the maximum number of members the list can have	The lower 15 bits identify the Member ID of the new member. The most significant bit (bit 15 - EX) defines the basic (0) or extended (1) message format. The extended format is not supported for this service.

Success Response Service Data

Name	Data Type	Description
Member ID	WORD	The member ID of the member serviced. For extended protocol, this is the member ID of the first member serviced.

Possible Error Codes from the Delete Member Service

Error Condition	General Error Response	General Error Code	Additional Error Code
The member list is empty	Already in requested mode/state	0B hex	01 hex
The requested service cannot be performed in this state. The Assembly Object has to be in its inactive state (that is corresponding I/O connection is not Established) to modify the member list	Object state conflict	0C hex	01 hex
Requested member ID (0) is invalid	Invalid Member ID	28 hex	01 hex
Requested member ID points to a Filler and cannot be removed	Invalid Member ID	28 hex	02 hex
The requested member ID is reserved for status information for this instance of the Assembly Object	Invalid Member ID	28 hex	03 hex
Extended protocol not supported	Invalid Member ID	28 hex	04 hex

Get Member Service

This service gets member data for the specified Member ID from the member list.

Request Service Data

Name	Data Type	Valid Values	Description
Attribute ID	USINT	2 (Member List)	Attribute Id of the member list
Member ID/ EX	WORD	1 – n n is the maximum number of members the list can have	The lower 15 bits identify the Member ID of the new member The most significant bit (bit 15 - EX) defines the basic (0) or extended (1) message format. The extended protocol is not supported

Success Response Service Data

Name	Data Type	Description

Member ID	WORD	The Member ID of the member serviced. If the requested member ID was greater than the largest existing Member ID, it returns the member with the highest Member ID For extended protocol, this is the member ID of the first member serviced
Member Data at Member ID	AsmMem Struct: UINT UINT EPATH	The member data of the specified member

Possible Error Codes from the Get Member Service

Error Condition	General Error Response	General Error Code	Additional Error Code
The member list is empty	Resource unavailable	02 hex	FF hex
Requested member ID (0) is invalid	Invalid Member ID	28 hex	FF hex
Extended protocol not supported	Invalid Member ID	28 hex	04 hex

NOTE *The assembly configuration is saved to the flash anytime a save service is requested for the I/O point Class. If the user changes the member list for the assembly instance that has a copy saved in the flash, then a restore service for the I/O point class will restore the old assembly configuration. The Save/Restore service on an I/O Point instance or attribute does not save/restore the assembly configuration.*

Acknowledge Handler Object

(Class ID = 0x2B)

The FP-1100 contains one instance of the Acknowledge Handler Object. It is used to manage the reception of message acknowledgements. This object communicates with change-of-state (COS) or Cyclic connection object. The Acknowledge Handler Object notifies the producing application of acknowledge reception, acknowledge timeouts, and production retry limit.

Class Attributes

Attribute ID	Access Rule	Name	Data Type	Default Value	Description of Attribute
1	Get	Revision	UINT	1	Revision of this object

Instance Attributes

Attribute ID	Applicable Services	Name	Data Type	Default Value	Description of Attribute
1	Set/Get	Acknowledge Timer	UINT	16 ms	Time to wait for acknowledge before resending
2	Set/Get	Retry Limit	USINT	1	Number of acknowledge timeouts to wait before informing the producing application of a Retry Limit reached event
3	Get	COS Producing Connection Instance	UINT		Connection instance which contains the path of the producing I/O application object which will be notified of Acknowledge Handler events

Semantics
See *DeviceNet Specifications* (vol II).

Common Services

Service Code	Implemented For		Service Name	Service Description
	Class	Instance		
0x0E	Yes	Yes	Get Attribute Single	Returns the contents of the specified attribute
0x10	No	Yes	Set Attribute Single	Modifies the value of the attribute

Get Attribute Single Service
This service returns the contents of the attribute specified in the service data.

Request Service Data

Name	Data Type	Valid Values	Description
Attribute ID	USINT	1 – 3	Identity of the attribute to be read/returned

Success Response Service Data
The format of the response data is dictated by the data type of the attribute referred by the Attribute ID in the service request.

Possible Error Codes from the Get Attribute Single Service
Any error code listed in the “General Error Codes” table can be returned from this service. There are no service specific error codes for this service.

Set Attribute Single Service

This service modifies the value of the specified attribute. The service can be applied to attributes 1 and 2. To modify an Acknowledge Handler Object attribute, the COS or Cyclic I/O connection must be in the Configuring state.

Request Service Data

Name	Data Type	Valid Values	Description
Attribute ID	USINT	1 – 2	Identity of the attribute to be modified
Attribute Value	Attribute specific	Attribute specific	New value of the attribute

Success Response Data

Name	Data Type	Description
Value set	Attribute specific	Actual value of the attribute. If the specified Attribute ID in the service request referred to the Acknowledge Timer attribute, then the value might be modified from the desired value due to timer resolution.

Possible Error Codes from the Set Attribute Single service

Error Condition	General Error Response	General Error Code	Additional Error Code
The desired attribute value is invalid	Invalid attribute value	09 hex	01 hex
The actual attribute value is the same as the desired attribute value	Already in requested mode/state	0B hex	ff hex
The object cannot perform a set operation in this state.	Object state conflict	0C hex	01 hex

Appendix A
LabVIEW Interface for FP-1100

The FP-1100 software includes LabVIEW VIs, based on NI-DNET software, to simplify application development for FP-1100 systems.

OpenExpMsgComm VI

This VI opens an Explicit Messaging connection with FP-1100 module and returns it's object handle. It can also open and start the network interface if it's not already open, and will pass out the interface handle. If this VI opens the interface, then the user is responsible to stop and close that interface when it's no longer needed.

I/O Point VIs

The I/O Point VIs can be grouped into two categories: Service VIs and Conversion VIs. The Service VIs send/receive messages over the DeviceNet network and may change the I/O point data. The Conversion VIs are used to process data that is sent to/received from the Service VIs.

All the Service VIs expect an Explicit Messaging (EM) connection has been opened with the FP-1100 module, and the interface has started. They take in the *ObjHandle In* for the EM connection, and the *I/O Point instance* (0 if sending the service to I/O Point class). They also take optional parameter *Num Points* (default 1), if the service is desired for more than one I/O Point. If the VIs output any service related data, it is usually in the form or an array, with the first element corresponding to the *I/O Point Instance* specified in the input, and the array length equals the *Num Points* input. All services, except for **Get Attribute Single** and **Get Member**, will only be successful if there are no I/O connections open (refer to I/O Point's State Transition Diagram in Chapter 3 for more details).

A list of Service VIs is given in the following table. A detailed description of these services and attributes is given in Chapter 3 of this manual.

Service VIs

VI Name	**Service Performed**	**Multiple Point Support**	**Service Description**
Get Channel Attribute	Get Member (0x18)	Yes	Gets one member (member Id = *Attribute Number)* from the Attribute Descriptor
Get Channel Command	Get Member (0x18)	Yes	Sets one member (member Id = *Command Number)* from the Command Descriptor
Get COS Deadband	Get (0x0E)	Yes	Gets the current setting of the COS Trigger Delta. Only recommended to be used for

			Analog Input points.
Get Current Range	Get (0x0E)	No	Gets the current range setting for an I/O point
Get Num Supported	Get (0x0E)	No	Gets the number of ranges, attributes, and commands supported for a give I/O point
Get Point Status	Get (0x0E)	Yes	Gets the current Status and Fault Code (if set)
Reset Point	Reset (0x05)	Yes	Resets I/O point (class, instance, or attribute) to the desired setting
Restore Config	Restore (0x15)	Yes	Restores I/O point (class, instance, or attribute) configuration saved previously
Send Channel Command	Set Member (0x19)	Yes	Sends a command to I/O Point(s)
Set Channel Attribute	Set Member (0x19)	Yes	Sets Channel attribute (member Id = *Attribute Number)* to the desired setting
Set COS Deadband	Set (0x10)	Yes	Sets the COS Trigger Delta for I/O Point(s)
Set New Range	Set (0x10)	Yes	Set I/O Point(s) range to the desired setting Id
Save Config	Save (0x16)	Yes	Saved I/O point (class, instance, or attribute) current configuration in non-volatile memory.

The Conversion VIs convert the raw data bytes sent to/received from the DeviceNet network. These VIs typically precede or follow one of the Service VIs. A brief list of the Conversion VIs and their functionality is given in the following table. All of the VIs listed in the table use one of LabVIEW clusters (.ctl) defined for the I/O Point Object.

Conversion VIs

VI Name	**Description**
Convert Log To Phys	This VI scales logical data received from ncReadDnetIO into its physical units. This VI expects the current range of the I/O point to be passed in the form of **I/O Range.ctl** and the byte offset for the data to be converted as input parameters.
Convert Phys to Log	This VI converts scaled physical value into a logical number that can be sent to the device via ncWriteDnetIO. This VI takes in the current range of the I/O point in the **I/O Range.ctl** form, the physical data to be converted, the output array, and the byte offset where the logical data should be placed in the output array, as input parameters.
ParseAsmStatus	This VI parses out the Module Status and Channel Status information from the data received from the device via an ncReadDnetIO call, and puts it in a variable of **I/O Status.ctl** type (see Appendix D, *Assembly*

	Decoding, for details).
Parse Channel Attribute	This VI converts the data received from **Get_Member** service and puts it in the **Channel Attribute.ctl** form before sending it out of the GetChannelAttribute VI.
Parse Channel Command	This VI converts the data received from **Get_Member** service and puts it in the **Channel Command.ctl** form before sending it out of the GetChannelCommand VI.
Parse Channel Range	This VI converts the data received from the **Get_Attribure_Single** service and puts it in **I/O Range.ctl** form before sending it out of the GetCurrentRange VI.

In addition to the VIs listed in the preceding tables, all VIs shipped with NI-DNET can be used to communicate with the FP-1100 module.

Appendix B
I/O Module Range and Attribute Settings

Range Table

Module Name	Range	Settings [Entry]
FP-AI-100	0-20 mA 4-20 mA +/- 20 mA +/- 5 V 0-5 V +/- 1 V 0-1 V 0-15 V +/- 30 V 0-30 V +/- 15 V	00 01 02 05 06 07 08 0E 0F 11 12
FP-AI-102	0-20 V +/- 20 V 0-60 V +/- 60 V 0-120 V +/- 120 V	13 14 15 16 17 18
FP-AI-110	0-20 mA 4-20 mA +/- 20 mA +/- 10 V 0-10 V +/- 5 V 0-5 V +/- 1 V 0-1 V +/- 250 mV +/- 50 mV	00 01 02 03 04 05 06 07 08 09 0A
FP-AI-111	0-20 mA 4-20 mA +/- 20 mA	00 01 02
FP-AO-200	0-20 mA 4-20 mA	00 01
FP-CTR-500	0-65535	40
	Boolean	10
	Boolean	10
FP-DI-300	Boolean	10
FP-DI-301	Boolean	10
FP-DI-330	Boolean	10
FP-DO-400	Boolean	10
FP-DO-401	Boolean	10
FP-DO-403	Boolean	10
FP-DO-410	Boolean	10
FP-PG-522	Boolean	10

FP-PWM-520	0-100 %	38
FP-RLY-420	Boolean	10
FP-RLY-422	Boolean	10
FP-RTD-122	73-1123 K -200, +850 °C -328, +1562 °F	26 27 28
	0-400 Ω 0-4000 Ω	30 31
FP-TC-120	0-2048 K -270, +1770 °C -454, +3218 °F +/- 50 mV +/- 100 mV	20 21 22 0A 0D
	223-358 K -50, +85 °C -58, +185 °F	23 24 25

Attribute Table

Module Name	Attribute Name (type)
FP-AI-110	Noise Rejection (enum)
FP-AI-111	Noise Rejection (enum)
FP-CTR-500	Noise Rejection (enum) Terminal Count (int) Count Source (enum) Gate Source (enum) Read Reset mode (enum) Output Source (enum) Output Mode (enum)
FP-PG-522	Pulse Mode (enum) On Time (int) Off Time (int) Resolution (enum)
FP-PWM-520	Period (ms) (int)
FP-RTD-122	RTD Type (enum)
FP-TC-120	Thermocouple Type (enum) CJC Source (enum)

Appendix D
Assembly Decoding

The Assembly Object supported by the FP-1100 is pseudo-dynamic, i.e. it can't be created or deleted via EM service, but its member list can be changed via Insert/Remove Member services. That is why all three instances of the assembly object get their Ids in the vendor specific Id range (required by DeviceNet Specifications). Each assembly instance is associated with a producing or consuming side of an I/O Connection Object (the consuming assembly instance is shared by both COS and poll consuming side, since for both connection types the same consuming connection is used. The producing side of COS and Poll get their own assembly instances). If there is no configuration saved in the flash, a default member list is created for each assembly instance on power up. In that case, both COS and Poll assembly instances will look exactly alike. However, a user can change one instance (e.g. by inserting a new member or changing the status reporting option) without affecting the other assembly instance.

The assembly configuration is saved to the flash anytime a save service is requested for the I/O Point Class instance. If the user changes the member list for the assembly instance that has a copy saved in the flash, then a restore service for the I/O Point class will restore the old assembly configuration. A Save/Restore on individual I/O Points or attributes does not effect the assembly configuration.

If the system setup doesn't match the saved assembly member list on power-up, the device reports a major recoverable fault by flashing the **MOD/NET STATUS** LED red. This fault is cleared when the old system is restored or an Identity Object's *Out of Box* reset is requested (I/O Point Object's reset on class instance can also reset the assembly configuration). If a request to allocate an I/O connection, that uses the faulty assembly instance, is made, the request will be accepted (given all other conditions are met, e.g. no other master, etc), and the I/O assembly of the expected size is produced/consumed on the bus. The data of such an I/O assembly may or may not be valid, depending on the system configuration and the cause behind the major recoverable fault.

Input (Consuming) Assembly Member Structure & Requirements

Any I/O point instance that represents an output channel (e.g. discrete output) can become a member of the input assembly instance. On power up, the FP-1100 generates a default assembly (if powered up in factory default or no assembly configuration is saved previously) that includes all the output points (digital and analog) in its member list. There is only one consuming assembly created for the whole system. This instance is shared between the Poll and COS/Cyclic connections, if they are running simultaneously. The insert service for this instance checks the direction (input or output) of the I/O instance before inserting it in the list. If the instance represents an input channel, then an error is returned from the device and the requested I/O point is not inserted in the list.

Output (Producing) Assembly Member Structure & Requirements

Unlike the input assembly, the output assembly allows both input and output points to be its members. The default assembly, however, only has the I/O instances that represent physical input channels (not included in the beta software). The user has the option to add any output point in the assembly via Insert Member service.

In addition to the I/O points, the output assembly includes the status information of the I/O modules. Each output assembly has at least two bytes reserved for the status info. These two bytes represent status of all the I/O modules, and precede all the data bytes in the output assembly. Each of the first 9 bits of the mod-status field represents the overall status of the corresponding I/O module (bit 0 => I/O module 1 status). A value of 0 implies that the corresponding module is functioning properly, and a value of 1 implies that at least one channel in that module is in fault state. The MSB (channel status bit) of the mod-status field, if set, implies that individual channel status is added to the end of the assembly. A value of 0 for this field means that the assembly is reporting only channel data. This bit is programmable via the **Set Attribute Single** service on the assembly instance (attribute Id 100). The default value for the channel-status bit is TRUE (include status).

Changing the Assembly Object

The Member List of an Assembly Object is a configurable attribute (changing it may result in changing some of the I/O connections attributes, e.g. producing or consuming size). Both insert and remove services are supported for the Member List (see *Chapter 3* for details). The Member List can only be changed when the assembly instance is not active (that is, its associated I/O connection is not established).

When the Member List of the Assembly Object is altered, some Filler (path size = 0) members may be added or removed from the list to pad unused bits of bytes that represent data for sequential discrete channels. A request to remove these members will result in an error.

> *NOTE You can use the SimpleWho utility shipped with the FP-1100 software to change the Assembly Object's Member List attribute.*

Getting Assembly Object Attributes

All assembly object attributes are readable at all times. Attribute 3 of the Assembly Object instance is the same as the actual set of bytes returned (received) by the I/O connection over the network. Since the member list of the Assembly Object can be very long (depending on the I/O points it contains as members), a get on this attribute (#2) may result in *Too Much Data* error. In that case, a copy of the member list can be created inside a host application by getting each member separately via **Get_Member** service.

Example

In the setup below, a combination of analog and discrete I/O modules is used to illustrate how the assembly size and its data offset can be calculated for a given FieldPoint bank.

> *NOTE You can use the SimpleWho utility shipped with the FP-1100 software to calculate the data offset for I/O points by saving the configuration to a File.*

System Setup

FP - 1100	FP-AI - 100	FP-AO - 200	FP-DO-401	FP-D1-300

Example System Setup

Module Name	Number of I/O channels	Number of Data bytes in the Assembly (2 bytes per channel for analog 1 bit per channel for Discrete)	Number of Status bytes in the assembly (1 bit per channel)
FP-AI-100	8	16	1
FP-AO-200	8	16	1
FP-DO-401	16	2	2
FP-DI-300	8	1	1

The Consuming Assembly will only contain the data for output channels so the default length for the assembly will be **18** (16 bytes for FP-AO-200 and 2 bytes for FP-DO-401). The first 2 bytes will refer to the first channel on FP-AO-200. The last two bytes will refer to FP-DO-401 channels, with the least significant bit (LSB) of the first byte referring to the first DO channel.

The Producing Assembly will have all the I/O data and the status information. The producing assembly data can be broken into three sections: Mod Status Field, I/O Data Field, and Channel Status Field.

Mod Status Field

The first WORD in the output assembly contains status bits for each I/O module (Figure 2).

Detailed Status (b15)	Reserved (b14-b9)	I/O Module Status (b8-b0) where b_n refers to status bit for I/O module at address n+1

Figure 2: Module Status Fields for Producing Assembly

The Detailed Status bit, if set, implies that the last [Total Points/8] bytes represent status bits for individual I/O channels, with LSB of the first byte representing Channel 0 or Module 1 and MSB of the last byte representing the last channel of the last module.

I/O Data Field

The I/O data for an individual channel follows the Mod Status word. The first item in this sequence represents the current I/O value of the first I/O point instance (usually Channel 0 or Module 1). If the I/O Point is represents an analog point, then this item will take 2 bytes. If it's discrete, it will take one bit. All sequential discrete channels are OR'ed in one data byte, with up to 8 channels per byte. If any bit in this byte is unused it is added as a Filler member to the member list and is always set to 0. The remaining items in the I/O Data field follow similar rules. The length of this field depends on the Number of I/O points. For default assembly the length will

equal [(Number of Analog I/O points * 2) + (Number of Discrete I/O Points/8)]. For this example, the length of the I/O Data field will be 35.

Channel Status Field

If the *Detailed Status* bit of the Module Status Field is set, then remaining bytes of the assembly represent status bits for individual I/O channels, with LSB of the first byte representing channel statutes of Channel 0 of Module 1, and MSB of the last byte representing status attribute of the last channel of the last module. For default assembly, the length of this field equals [Total I/O Points/8]. In this example, the length for the Channel Status field is 5.

The total length of the producing assembly is equal to the sum of the Mod Status Field, I/O Data Field, and Channel Status Fields. For this example, the total length is **42**.

NOTE The actual length of the input and output assembly can be verified by running the SimpleWho utility.

Bibliography

[1] DeviceNet specification 2.0
[2] Smart IO Proposal