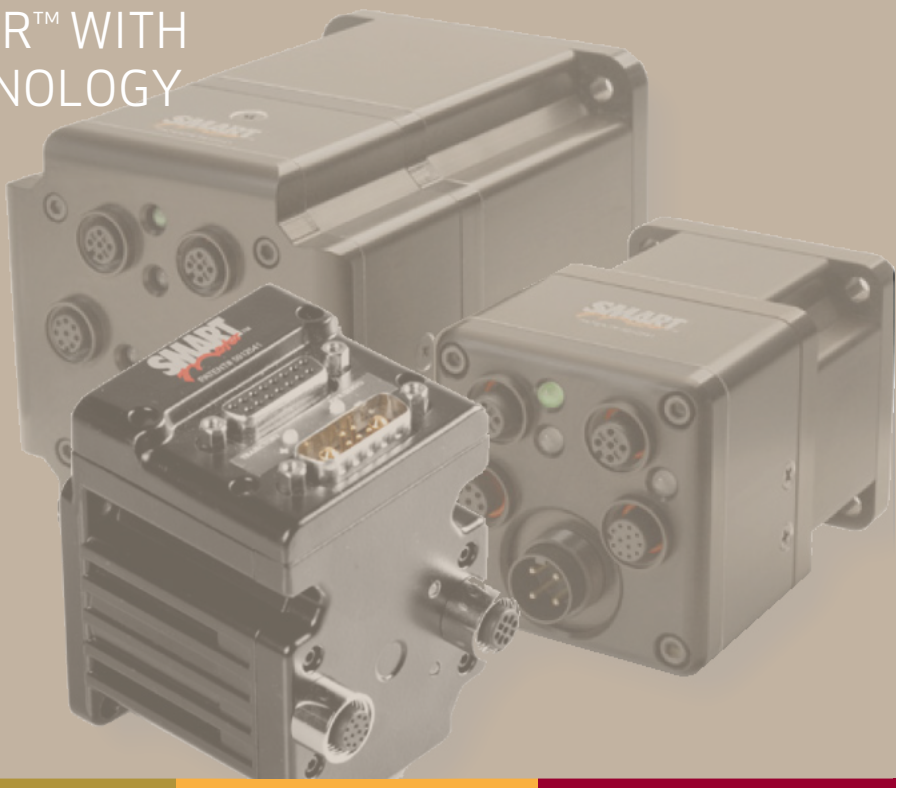
DEVICENET™ IMPLEMENTATION FOR

FULLY INTEGRATED SERVO MOTOR

CLASS 5 SMARTMOTOR™ WITH COMBITRONIC™ TECHNOLOGY



Rev. C, February 2021

DESCRIBES THE CLASS 5 SMARTMOTOR™ SUPPORT FOR THE DEVICENET PROTOCOL

www.animatics.com

MOOG ANIMATICS

Copyright Notice

Moog Animatics Class 5 SmartMotor™ DeviceNet™ Guide, Rev. C, SC80100011-001.

Please let us know if you find any errors or omissions in this manual so that we can improve it for future readers. Such notifications should contain the words "DeviceNet Guide" in the subject line and be sent by e-mail to: animatics_marcom@moog.com. Thank you in advance for your contribution.

Contact Us:

Americas - West
Moog Animatics
2581 Leghorn Street
Mountain View, CA 94043
USA

Tel: 1 650-960-4215

Americas - East
Moog Animatics
1995 NC Hwy 141
Murphy, NC 28906
USA

Tel: 1 828-837-5115
Fax: 1 540-557-6509

Support: 1 (888) 356-0357

Website: www.animatics.com

Email: animatics_sales@moog.com

Table of Contents

Introduction

This chapter provides information on the purpose and scope of this manual. It also provides information on safety notation, related documents and additional resources.

Purpose

This manual explains the Moog Animatics Class 5 SmartMotor™ support for the DeviceNet™ protocol. It describes the major concepts that must be understood to integrate a SmartMotor slave with a PLC or other DeviceNet master. However, it does not cover all the low-level details of the DeviceNet protocol.

NOTE: The feature set described in this version of the manual refers to motor firmware 5.16.x.y and 5.97.x.y.

This manual is intended for programmers or system developers who have read and understand *THE CIP NETWORKS LIBRARY Volume 1 - Common Industrial Protocol (CIP™)* and *THE CIP NETWORKS LIBRARY Volume 3 - DeviceNet Adaptation of CIP*, which are published and maintained by ODVA.org (http://www.odva.org). Therefore, this manual is not a tutorial on those specifications or the DeviceNet protocol. Instead, it should be used to understand the specific implementation details for the Moog Animatics SmartMotor. For a general overview of DeviceNet, see DeviceNet Overview on page 16.

The reference chapters of this manual include details about the specific commands available in the SmartMotor through the DeviceNet protocol. The commands include those required by the specifications and those added by Moog Animatics.

Combitronic Technology

The most unique feature of the SmartMotor is its ability to communicate with other SmartMotors and share resources using Moog Animatics' Combitronic™ technology. Combitronic is a protocol that operates over a standard CAN interface. It may coexist with either CANopen or DeviceNet protocols. It requires no single dedicated master to operate. Each SmartMotor connected to the same network communicates on an equal footing, sharing all information, and therefore, sharing all processing resources.

For more information on Combitronic communications, see the *SmartMotor™ Developer's Guide*.

Abbreviations and Definitions

The following table provides a list of abbreviations and definitions of terms that may be used in this manual or related documents.

Abbreviation/ Term	Description
API	Actual Packet Interval
ASCII	American Standard Code for Information Interchange
AT	Acceleration Target
BOI	Buss-Off Interrupt
Client	A device that sends a request to, and expects a response from, a server.
Consumer	Network device that reads messages from a producer device.
CoS	Change of State I/O trigger
DC	Direct Current
DT	Deceleration Target
EDS	Electronic Data Sheet. A text file that contains configuration information for the device.
EPR	Expected Packet Rate
EtherNet/IP	Ethernet Industrial Protocol
Explicit messaging	(Non-cyclic) Not time-sensitive, typically used for network and device configuration, and setup of cyclic connections.
FOC	Field Oriented Current
FTP	File Transfer Protocol
IE	Industrial Ethernet
Implicit messaging	(Cyclic) Timely, repetitive transfer of data, typically used for I/O control (e.g., PID loop closure).
IN	Input
LAN	Local Area Network
MACID	Media Access Control Identifier
NASM	Network Access State Machine
ODVA	Open DeviceNet Vendors Association, Inc, which is the standards organization that maintains the specifications for the CIP industrial network protocols.
OUT	Output
PDU	Protocol Data Unit
PLC	Programmable Logic Controller
Producer	A device that puts messages on the network for "consumer" devices (other network devices that will read the messages).
PU	Position Units

Abbreviation/ Term	Description
PV	Profile Velocity (mode)
PT	Position Target
Rx	Receive
Server	A device that receives requests from clients and sends responses to them.
Slave device	Device consuming data transfers from a Network Master. A PLC (Programmable Logic Controller) is a good example of a Master.
SMI	SmartMotor Interface (software)
STD	State Transition Diagram
TCP	Transmission Control Protocol
TQ	Torque (mode)
Tx	Transmit
UDP	User Datagram Protocol
UCMM	Unconnected Message Manager
VU	Velocity Units
VT	Velocity Target

Safety Information

This section describes the safety symbols and other safety information.

Safety Symbols

The manual may use one or more of the following safety symbols:

WARNING: This symbol indicates a potentially nonlethal mechanical hazard, where failure to follow the instructions could result in serious injury to the operator or major damage to the equipment.

CAUTION: This symbol indicates a potentially minor hazard, where failure to follow the instructions could result in slight injury to the operator or minor damage to the equipment.

NOTE: Notes are used to emphasize non-safety concepts or related information.

Other Safety Considerations

The Moog Animatics SmartMotors are supplied as components that are intended for use in an automated machine or system. As such, it is beyond the scope of this manual to attempt to cover all the safety standards and considerations that are part of the overall machine/system design and manufacturing safety. Therefore, the following information is intended to be used only as a general guideline for the machine/system designer.

It is the responsibility of the machine/system designer to perform a thorough "Risk Assessment" and to ensure that the machine/system and its safeguards comply with the safety standards specified by the governing authority (for example, ISO, OSHA, UL, etc.) for the locale where the machine is being installed and operated. For more details, see Machine Safety on page 12.

Motor Sizing

It is the responsibility of the machine/system designer to select SmartMotors that are properly sized for the specific application. Undersized motors may: perform poorly, cause excessive downtime or cause unsafe operating conditions by not being able to handle the loads placed on them. The *System Best Practices* document, which is available on the Moog Animatics website, contains information and equations that can be used for selecting the appropriate motor for the application.

Replacement motors must have the same specifications and firmware version used in the approved and validated system. Specification changes or firmware upgrades require the approval of the system designer and may require another Risk Assessment.

Environmental Considerations

It is the responsibility of the machine/system designer to evaluate the intended operating environment for dust, high-humidity or presence of water (for example, a food-processing environment that requires water or steam wash down of equipment), corrosives or chemicals that may come in contact with the machine, etc. Moog Animatics manufactures specialized IP-rated motors for operating in extreme conditions. For details, see the *Moog Animatics Product Catalog*, which is available on the Moog Animatics website.

Machine Safety

In order to protect personnel from any safety hazards in the machine or system, the machine/system builder must perform a "Risk Assessment", which is often based on the ISO 13849 standard. The design/implementation of barriers, emergency stop (E-stop) mechanisms and other safeguards will be driven by the Risk Assessment and the safety standards specified by the governing authority (for example, ISO, OSHA, UL, etc.) for the locale where the machine is being installed and operated. The methodology and details of such an assessment are beyond the scope of this manual. However, there are various sources of Risk Assessment information available in print and on the internet.

NOTE: The following list is an example of items that would be evaluated when performing the Risk Assessment. Additional items may be required. The safeguards must ensure the safety of all personnel who may come in contact with or be in the vicinity of the machine.

In general, the machine/system safeguards must:

- Provide a barrier to prevent unauthorized entry or access to the machine or system. The barrier must be designed so that personnel cannot reach into any identified danger zones.
- Position the control panel so that it is outside the barrier area but located for an unrestricted view of the moving mechanism. The control panel must include an E-stop mechanism. Buttons that start the machine must be protected from accidental activation.
- Provide E-stop mechanisms located at the control panel and at other points around the perimeter of the barrier that will stop all machine movement when tripped.
- Provide appropriate sensors and interlocks on gates or other points of entry into the protected zone that will stop all machine movement when tripped.
- Ensure that if a portable control/programming device is supplied (for example, a hand-held operator/programmer pendant), the device is equipped with an E-stop mechanism.

  NOTE: A portable operation/programming device requires *many* additional system design considerations and safeguards beyond those listed in this section. For details, see the safety standards specified by the governing authority (for example, ISO, OSHA, UL, etc.) for the locale where the machine is being installed and operated.
- Prevent contact with moving mechanisms (for example, arms, gears, belts, pulleys, tooling, etc.).
- Prevent contact with a part that is thrown from the machine tooling or other part-handling equipment.
- Prevent contact with any electrical, hydraulic, pneumatic, thermal, chemical or other hazards that may be present at the machine.
- Prevent unauthorized access to wiring and power-supply cabinets, electrical boxes, etc.
- Provide a proper control system, program logic and error checking to ensure the safety of all personnel and equipment (for example, to prevent a run-away condition). The control system must be designed so that it does not automatically restart the machine/system after a power failure.
- Prevent unauthorized access or changes to the control system or software.

Documentation and Training

It is the responsibility of the machine/system designer to provide documentation on safety, operation, maintenance and programming, along with training for all machine operators, maintenance technicians, programmers, and other personnel who may have access to the machine. This documentation must include proper lockout/tagout procedures for maintenance and programming operations.

It is the responsibility of the operating company to ensure that:

- All operators, maintenance technicians, programmers and other personnel are tested and qualified before acquiring access to the machine or system.
- The above personnel perform their assigned functions in a responsible and safe manner to comply with the procedures in the supplied documentation and the company safety practices.
- The equipment is maintained as described in the documentation and training supplied by the machine/system designer.

Additional Equipment and Considerations

The Risk Assessment and the operating company's standard safety policies will dictate the need for additional equipment. In general, it is the responsibility of the operating company to ensure that:

- Unauthorized access to the machine is prevented at all times.
- The personnel are supplied with the proper equipment for the environment and their job functions, which may include: safety glasses, hearing protection, safety footwear, smocks or aprons, gloves, hard hats and other protective gear.
- The work area is equipped with proper safety equipment such as first aid equipment, fire suppression equipment, emergency eye wash and full-body wash stations, etc.
- There are no modifications made to the machine or system without proper engineering evaluation for design, safety, reliability, etc., and a Risk Assessment.

Safety Information Resources

Additional SmartMotor safety information can be found on the Moog Animatics website; open the topic "Controls - Notes and Cautions" located at:

https://www.animatics.com/support/downloads/knowledgebase/controls---notes-and-cautions.html

OSHA standards information can be found at:

https://www.osha.gov/law-regs.html

ANSI-RIA robotic safety information can be found at:

http://www.robotics.org/robotic-content.cfm/Robotics/Safety-Compliance/id/23

UL standards information can be found at:

http://ulstandards.ul.com/standards-catalog/

ISO standards information can be found at:

http://www.iso.org/iso/home/standards.htm

EU standards information can be found at:

http://ec.europa.eu/growth/single-market/european-standards/harmonised-standards/index_en.htm

Additional Documents

The Moog Animatics website contains additional documents that are related to the information in this manual. Please refer to the following list.

Related Guides

- *Class 5 SmartMotor™ Installation & Startup Guide*

  http://www.animatics.com/cl-5-install-startup-guide

- *SmartMotor™ Developer's Guide*

  http://www.animatics.com/smartmotor-developers-guide

- *SmartMotor™ System Best Practices*

  http://www.animatics.com/system-best-practices-application-note

Other Documents

- SmartMotor™ Certifications

  https://www.animatics.com/certifications.html

- *SmartMotor Developer's Worksheet*
  (interactive tools to assist developer: Scale Factor Calculator, Status Words, CAN Port Status, Serial Port Status, RMODE Decoder and Syntax Error Codes)

  https://www.animatics.com/support/downloads.knowledgebase.html

- *Moog Animatics Product Catalog*, which is available on the Moog Animatics website

  http://www.animatics.com/support/moog-animatics-catalog.html

Additional Resources

The Moog Animatics website contains useful resources such as product information, documentation, product support and more. Please refer to the following addresses:

- General company information:
  http://www.animatics.com
- Product information:
  http://www.animatics.com/products.html
- Product support (Downloads, How To videos, Forums, Knowledge Base, and FAQs):
  http://www.animatics.com/support.html
- Contact information, distributor locator tool, inquiries:
  https://www.animatics.com/contact-us.html
- Application ideas (including videos and sample programs):
  http://www.animatics.com/applications.html

ODVA Resources

DeviceNet is a common standard maintained by ODVA.org:

- ODVA.org website:
  http://www.odva.org/

DeviceNet Overview

This chapter provides an overview of DeviceNet features. These sections briefly summarize the technical information provided on the ODVA.org website. To view the fully detailed information or to obtain the specifications, see the ODVA.org website at: http://www.odva.org.

DeviceNet Introduction

DeviceNet was introduced in 1994 by Allen Bradley. It is now a CIP-based technology that is managed by the Open DeviceNet Vendors Association (ODVA), which is a standards organization that manages all CIP network technologies.

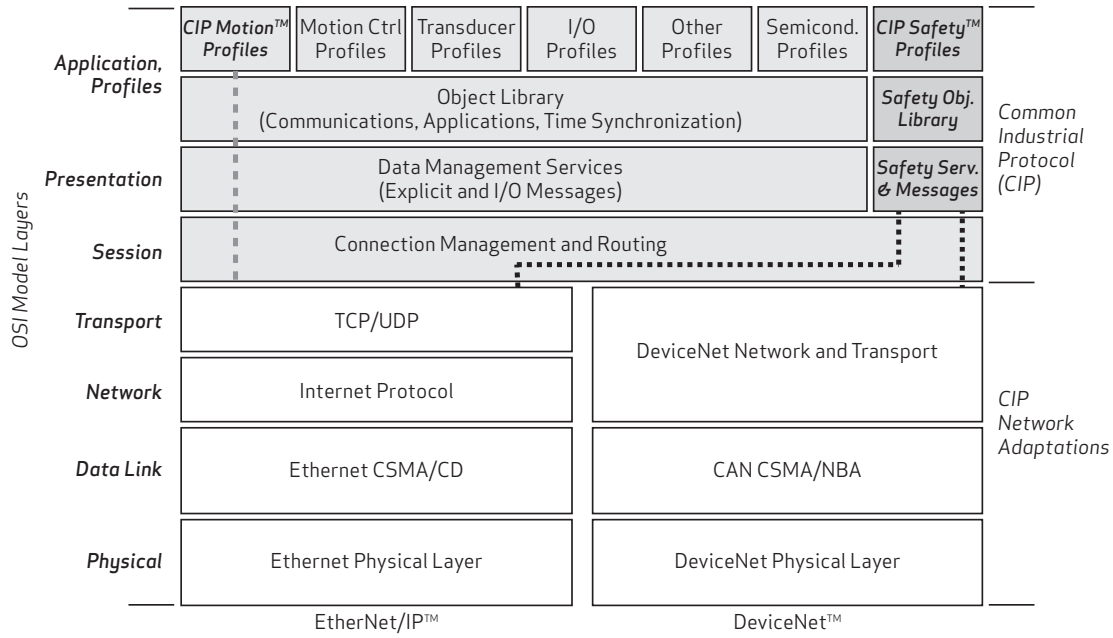
EtherNet/IP and DeviceNet are two CIP network technologies that are supported by Moog Animatics (see The OSI Model on page 17). These networks share the same CIP layers and use objects to describe the network devices (this collection of objects specific to a device is the device profile). Because of this, they are able to communicate with each other. For example, a device on an EtherNet/IP network can communicate with one on a DeviceNet network. For more information on CIP objects, see Objects on page 19.

The Class 5 DeviceNet SmartMotor is designed to operate as a device on a DeviceNet network. This allows the system designer to take advantage of SmartMotor technology through its device profile (for example, start a user program stored in the SmartMotor). For details on the SmartMotor device profile using the Position Controller device, see SmartMotor Device Profile Overview on page 35.

The full specification for DeviceNet is available from the ODVA.org website. For details, see *THE CIP NETWORKS LIBRARY Volume 1 - Common Industrial Protocol (CIP™)* and *THE CIP NETWORKS LIBRARY Volume 3 - DeviceNet Adaptation of CIP*.

The OSI Model

The OSI model describes the architecture for the CIP-based industrial network protocols. Moog Animatics supports EtherNet/IP and DeviceNet using the Position Controller Supervisor and Position Controller profiles. The other profiles shown are not currently supported.



OSI Model for EtherNet/IP and DeviceNet

The following table provides a brief description of each of the seven OSI model layer.

Layer	Description
Physical	The physical properties—electrical and mechanical—of the network (e.g., cables, connectors, pin-outs, voltages, flow control, etc.).
Data Link	How packets of data will be transmitted between devices (MAC, CRC, etc.).
Network	The switching and routing layer, i.e, anything related to the device IP address, DNS, datagrams, cyclic and non-cyclic.
Transport	Controls how much data (size of block) that will be sent and received, manages the delay time between messages, maintains the quality of service (QoS).
Session	Opens/closes and manages the connection between devices and applications, explicit and implicit messages are used. This layer is part of CIP.
Presentation	Delivers and formats information to/from the application layer (translates data from the network to the application or from the application to the network). This layer is part of CIP.
Application	Handles the application that provides the user interaction. This layer is part of CIP.

For more details, see the ODVA.org website at: http://www.odva.org.

Objects

This section briefly describes the features of CIP device objects. For more details, see *THE CIP NETWORKS LIBRARY, Volume 1: Common Industrial Protocol (CIP™)*, which is available on the ODVA.org website at: http://www.odva.org.

Objects

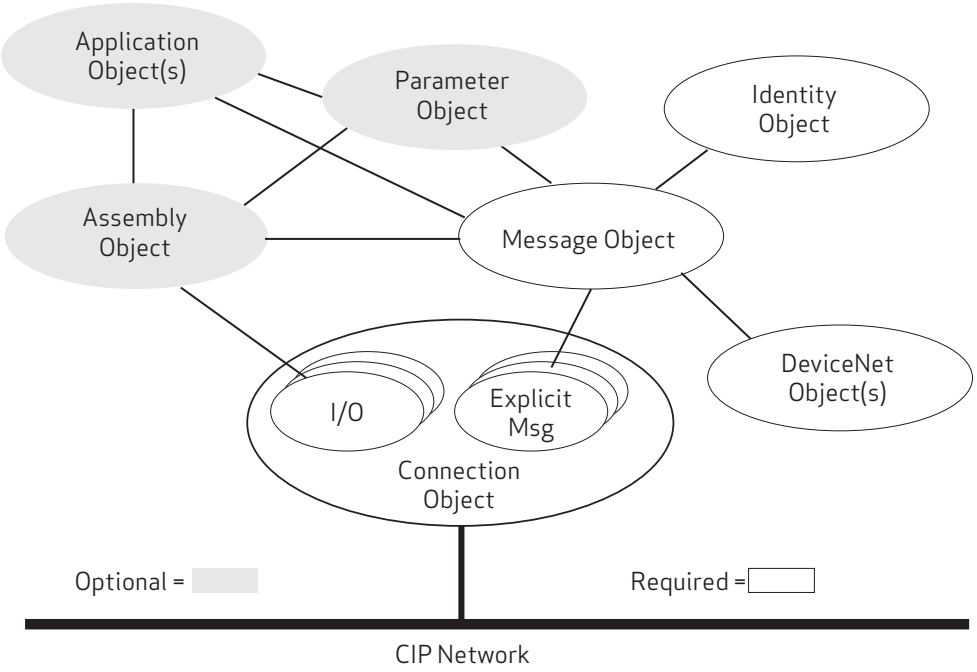
Because DeviceNet is a CIP-based network, the network devices are described through sets of objects. Each set of objects is organized in a specific manner with specific attributes so that each network device operates in a certain way—that organization is the object model (or device model). Every device with the same object model will operate in the same manner.

NOTE: All device features must be described through objects in order to be accessible through CIP.

The following types of objects are used in a device profile:

- Required objects: these must be present in every CIP device.
- Application objects: these are specific to the type of device and its function.
- Manufacturer-Specific objects: these are optional objects that are specific to each device manufacturer.

The following figure shows a basic version of the DeviceNet object model.



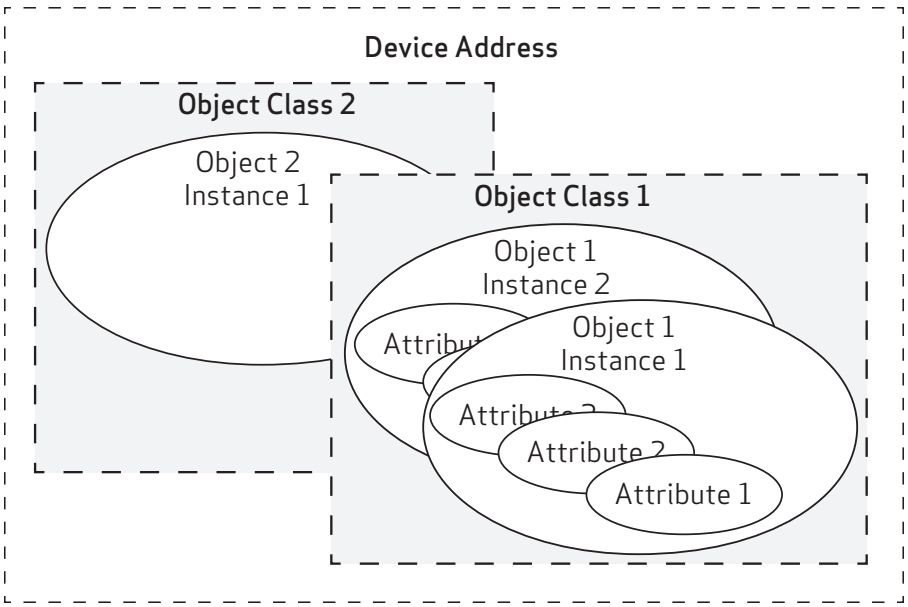
DeviceNet Object Model

For the SmartMotor-specific model, see the object model for your SmartMotor application, which is described later in this guide.

Classes, Instances and Attributes

The CIP object model uses classes, instances and attributes to describe each device. Refer to the following figure.

- Class: a fixed collection of objects with each object having a fixed set of attributes. The CIP object library contains three primary object classes: general use, application specific, and network specific.
- Instance: an occurrence of a particular object (in other words, there can be more than one occurrence of the same object but with different attribute values).
- Attributes: a set of data values that describe an object instance (instance attributes). They can also describe an object class (class attributes).



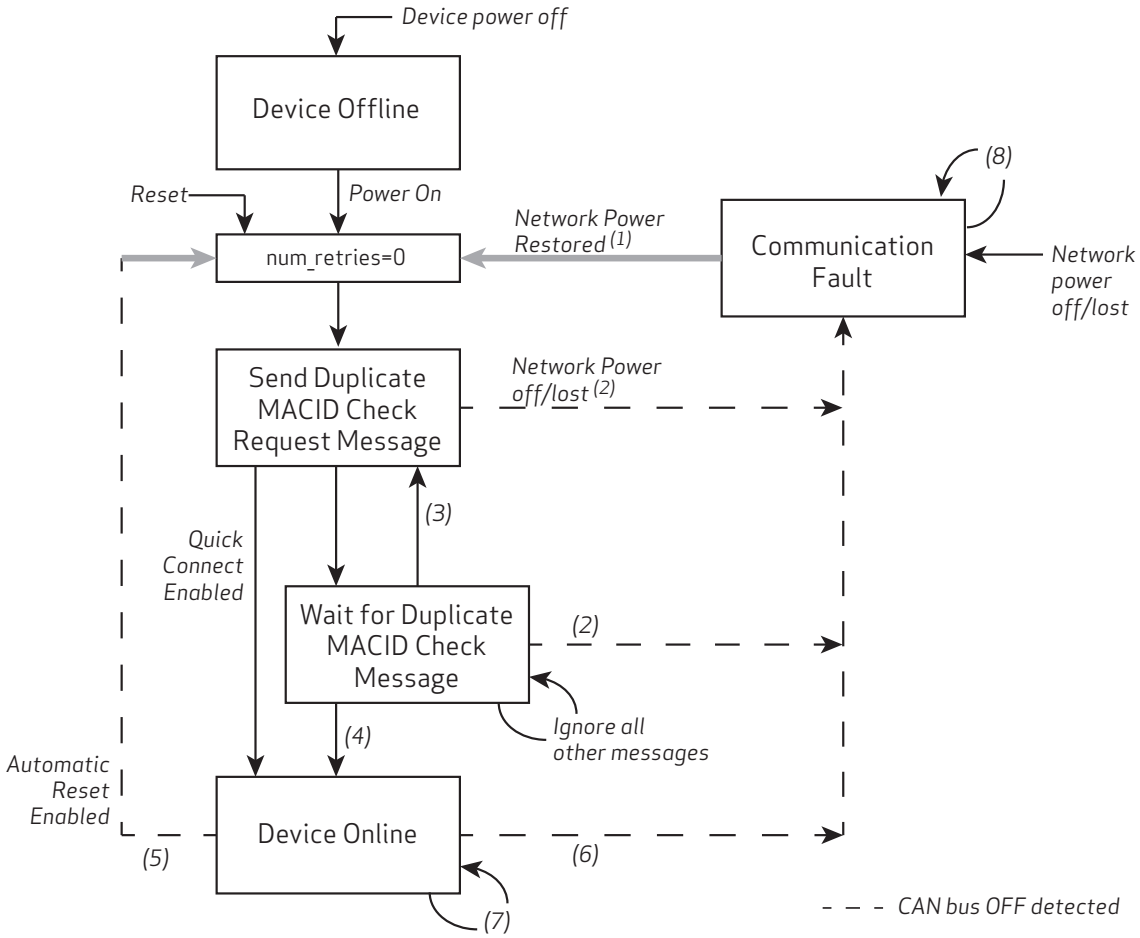
Classes, Instances, Attributes

The device description (class, instance, attributes) information is also contained in the Electronic Data Sheet (EDS) file, which is supplied by the equipment vendor (see EDS File on page 37).

For more details, see the ODVA.org website at: http://www.odva.org.

Network Access State Machine

The following figure shows the DeviceNet Network Access State Machine (NASM). A major function of the NASM is to keep two devices with the same MACID, or network address, from both going online together and causing communication failures. For more details, see *THE CIP NETWORKS LIBRARY, Volume 3: DeviceNet Adaptation of CIP*, which is available on the ODVA.org website at: http://www.odva.org.



NOTES:

1. Manual intervention, Communication Faulted Request message (Change MACID message received, changed to New MACID)
2. Duplicate MACID check request/response received
3. Duplicate MACID check request/response not received (timeout), num_retries=0, increment num_retries
4. Duplicate MACID check request/response not received (timeout), num_retries=1
5. DeviceNet Object Attribute BOI = Automatic Reset (01)
6. Duplicate MACID check response received, DeviceNet Object Attribute BOI = Hold in Reset (00)
7. Duplicate MACID check request received, transmit Duplicate MACID Check Response message; Quick Connect Enabled and Duplicate MACID check request/response not received (timeout), num_retries=0, increment num_retries and send MACID Check Request message
8. (Optional) Respond to Communication Faulted Request message

DeviceNet Network Access State Machine

Unconnected Message Manager

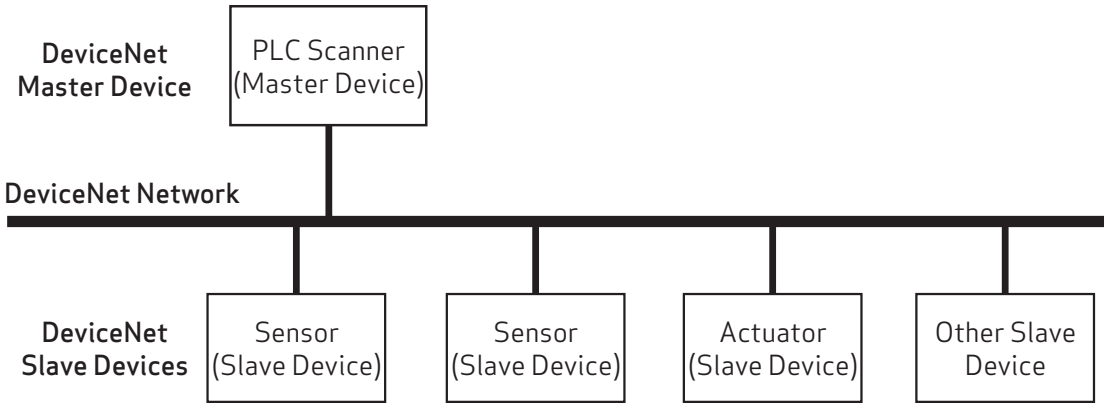
After a Slave device is online, a DeviceNet Master will typically make and manage connections to that device. The SmartMotor supports Unconnected Message Manager (UCMM) requests to allow quick connections while minimizing explicit messaging overhead.

Predefined Master/Slave Connection Set

The SmartMotor uses the DeviceNet Predefined Master/Slave connection set which is designed to provide simplified I/O communications. It removes many steps that would normally be required to establish those communications. As a result, it decreases the use of network bandwidth and device resources.

The master device takes control of the slave devices through a scan list (i.e., the MACIDs of the slave devices are on the scan list of the maser device). Note that slave devices can only initiate Duplicate MACID Check messages. Any other slave communications are not permitted (i.e., they are ignored) until the master device authorizes it.

The following figure provides a basic example of a DeviceNet Master/Slave network.



Example Master/Slave DeviceNet Network

For more details, see *THE CIP NETWORKS LIBRARY, Volume 3: DeviceNet Adaptation of CIP*, which is available on the ODVA.org website at: http://www.odva.org.

For more information on DeviceNet connections to the SmartMotor, see Connections, Wiring and Status LEDs on page 27.

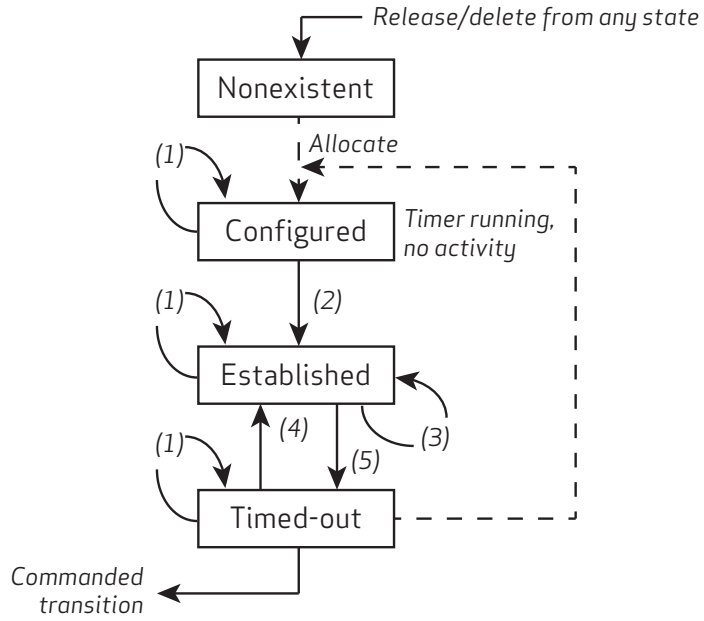
SmartMotor Connection IDs

The following two Connection IDs are used for the DeviceNet implementation on the SmartMotor:

1. Explicit messaging content to the server (for details, see Attributes for Instance 1: Explicit Connection on page 66)
2. Polled I/O connection peer-to peer only (for details, see Attributes for Instance 2: Polled I/O Connection on page 67)

Master/Slave I/O Connection State Transition

The following figure shows the I/O state transition diagram for the predefined Master/Slave I/O connection.



Notes:

1. Get_attribute, Set_attribute - values changed during timeout are reset to initial state if Allocate/Release is used to exit.
2. Set_attribute_single - expected_packet_rate attribute.
3. Send - data received - pre-consumption inactivity timer runs until first I/O consumed, then timeout=value from expected_packet_rate attribute (see note 2).
4. Reset (optional).
5. Inactivity/Watchdog Timeout - watchdog_time-out_action= transition to timed-out.

For more details, see *THE CIP NETWORKS LIBRARY, Volume 3: DeviceNet Adaptation of CIP*, which is available on the ODVA.org website at: http://www.odva.org.

Predefined Master/Slave Connection Set Identifier Fields

The following table provides the Identifier Fields for the Predefined Master/Slave Connection Set. This can be useful when developing a master to communicate with the SmartMotor as a slave device over DeviceNet.

Identifier Bits											Identity Usage	Hex Range
10	9	8	7	6	5	4	3	2	1	0		
0	Group 1 Message ID				Source MACID						Group 1 Messages	000 – 3ff
0	1	1	0	0	Source MACID						Slave I/O Multicast Poll Response Message	
0	1	1	0	1	Source MACID						Slave I/O Change of State or Cyclic Message	
0	1	1	1	0	Source MACID						Slave I/O Bit-Strobe Response Message	
0	1	1	1	1	Source MACID						Slave I/O Poll Response or Change of State/Cyclic Acknowledge Message	
1	0	MACID						Group 2 Message ID			Group 2 Messages	400 – 5ff
1	0	Source MACID						0	0	1	Master I/O Bit-Strobe Command Message	
1	0	Multicast MACID						0	0	1	Master I/O Multicast Poll Command Message	
1	0	Destination MACID						0	1	0	Master Change of State or Cyclic Acknowledge Message	
1	0	Source MACID						0	1	1	Slave Explicit/ Unconnected Response Messages	
1	0	Destination MACID						1	0	0	Master Explicit Request Messages	
1	0	Destination MACID						1	0	1	Master I/O Poll Command/Change of State/Cyclic Message	
1	0	Destination MACID						1	1	0	Group 2 Only Unconnected Explicit Request Messages (reserved)	
1	0	Destination MACID						1	1	1	Duplicate MACID Check Messages	

For more details, see *THE CIP NETWORKS LIBRARY, Volume 3: DeviceNet Adaptation of CIP*, which is available on the ODVA.org website at: http://www.odva.org.

UCMM and Predefined Connection Set Example

The following provides an example of making a Master/Slave connection over DeviceNet. This example is intended for programmers interested in communicating to the SmartMotor from an embedded or PC platform using their own DeviceNet Master source code.

This example assumes the Master MACID =1 and the SmartMotor slave MACID=63. Transmissions from the master are shown in highlighted rows. The EPR (Expected Packet Rate) is set to zero (0) to ensure the connections stay open and keep the SmartMotor from timing out on either the Explicit or Polled connection during development.

CAN ID Bits (hex)	Data Frame Bytes (hex)	Description
5FF	00 2A 03 FF FF FF 00	Duplicate MACID Check from 63
5FF	00 2A 03 FF FF FF 00	Duplicate MACID Check from 63
5FE	01 4B 03 01 03 01	Allocate 3 from Master, Explicit&Polled
5FB	01 CB 00	Allocate Response from Slave MACID 63
5FC	41 10 05 01 09 00 00	Set EPR of Explicit to 0 (Class 0x5->Inst 1->Attr 9)
5FB	41 90 00 00	Response from 63 to set EPR
5FC	01 10 05 02 09 00 00	Set EPR to Polled to 0 (Class 0x5->Inst 2->Attr 9)
5FB	01 90 00 00	Response from 63 to set EPR
The Explicit and Polled Connections have been instantiated		
5FC	41 10 25 01 31 E0	Explicit Set Hardware Limits Action to OFF(E0h = 224d) (Class 0x25->Inst 1->Attr 49)
5FB	41 90	Explicit set success response from 63
5FD	01 00 22 01 A0 0F 00 00	Master Poll to Position Controller Implicit Connection Set velocity target = A0 0F 00 00 (RVT = 32768)
6FF	00 00 8? 21 00 00 00 00	SmartMotor Poll response from Position Controller SmartMotor Position = 00 00 00 00
NOTE: SmartMotor Feedback = 4000 cnts/rev, and update rate = 62.6 µSec.		

Messaging

There are two types of messages used by DeviceNet: explicit messages and implicit messages. Each is described in the following sections.

Explicit (Non-cyclic) Messages

Explicit messages are non-cyclic, i.e., they are typically sent once instead of at regular intervals. Further, explicit messages are not time sensitive. They are used for communicating information such as configuration, diagnostic, data logs, and other information that is not time critical. They can also be used to set up implicit (cyclic) connection content (see the next section).

Explicit messages are point-to-point messages. In other words, a device sends out a message directed to a specific recipient device. The recipient device will return a response to that message. As a result, the explicit messages are much larger than implicit messages (refer to the next section) and can generate a lot of network traffic; therefore, they are not used for transmitting cyclic data.

Implicit (Cyclic) Messages

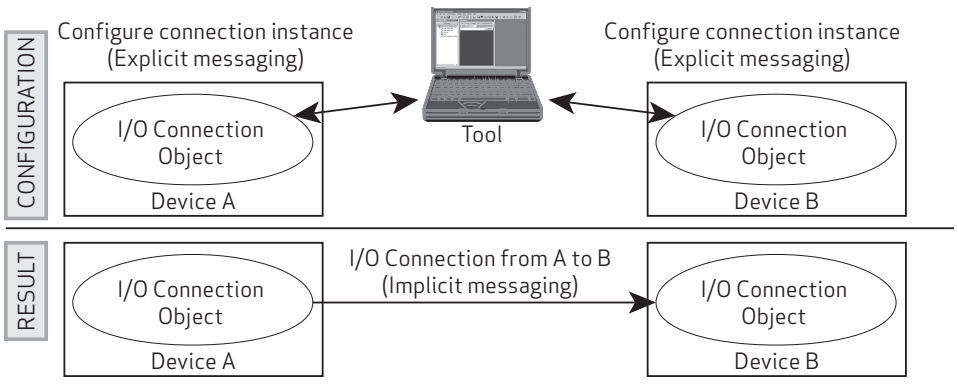
Implicit messages (also referred to as I/O messages) are cyclic, i.e., they are sent at regular intervals. Implicit messages are used to communicate critical, time-sensitive information. They are typically used for I/O control, PID loop closure, and Motion or Application control.

The implicit message connection between the two devices is established up front and connection ID assignment is made. Therefore, the actual implicit messages contain just the connection ID and the data. As a result, implicit messages are very small, they can travel quickly, and they do not use much network bandwidth.

For DeviceNet, a Polled Connection is considered an implicit connection.

Explicit/Implicit Messaging Example

In the following figure, a tool uses explicit messaging to configure the connections between two network devices. Once that I/O connection is established, the devices can communicate using implicit messaging. For more details, see *THE CIP NETWORKS LIBRARY, Volume 1: Common Industrial Protocol (CIP™)*, which is available on the ODVA.org website at: http://www.odva.org.



Explicit/Implicit Messaging Example

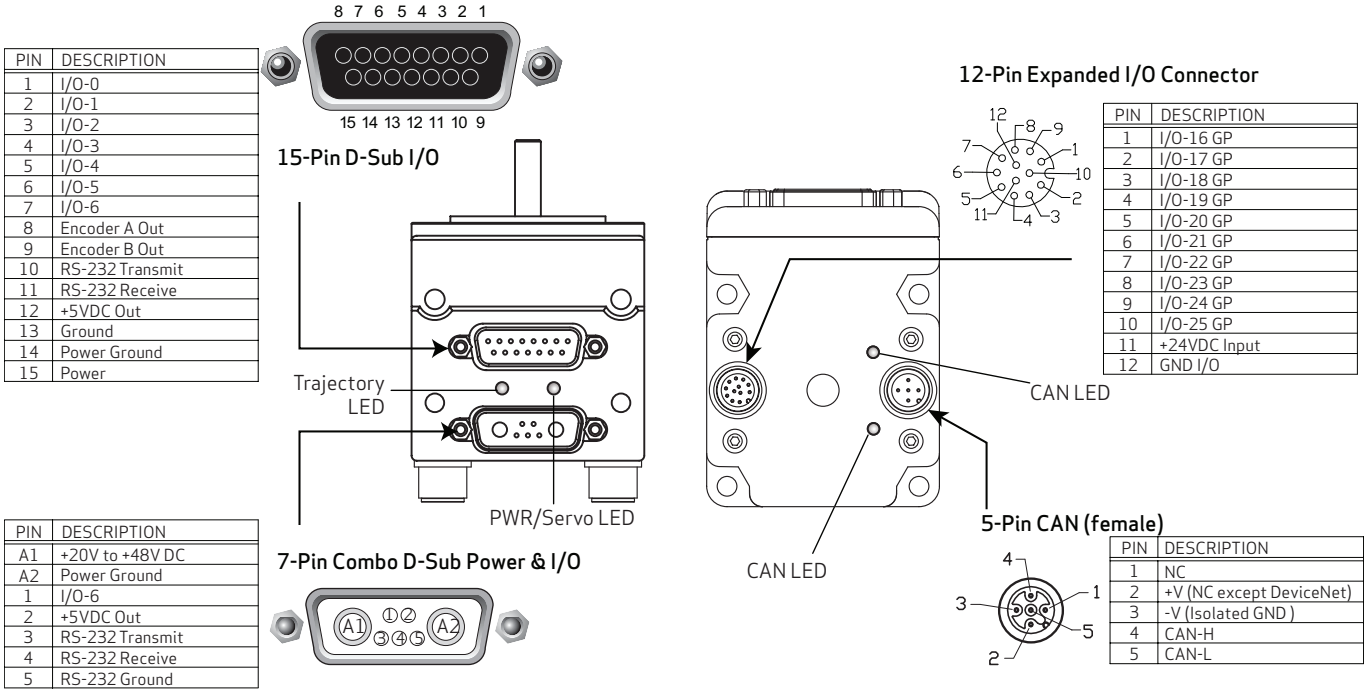
Connections, Wiring and Status LEDs

This chapter provides information on the SmartMotor connectors, a multidrop cable diagram, and a description of the SmartMotor status LEDs.

Connectors and Pinouts

D-Style Motor Connectors and Pinouts

The following figure provides a brief overview of the connectors and pinouts available on the D-style SmartMotors. For details, see the *Class 5 SmartMotor™ Installation & Startup Guide*. For LED functions, see Status LEDs on page 32.



PIN	DESCRIPTION
1	I/O-0
2	I/O-1
3	I/O-2
4	I/O-3
5	I/O-4
6	I/O-5
7	I/O-6
8	Encoder A Out
9	Encoder B Out
10	RS-232 Transmit
11	RS-232 Receive
12	+5VDC Out
13	Ground
14	Power Ground
15	Power

PIN	DESCRIPTION
1	I/O-16 GP
2	I/O-17 GP
3	I/O-18 GP
4	I/O-19 GP
5	I/O-20 GP
6	I/O-21 GP
7	I/O-22 GP
8	I/O-23 GP
9	I/O-24 GP
10	I/O-25 GP
11	+24VDC Input
12	GND I/O

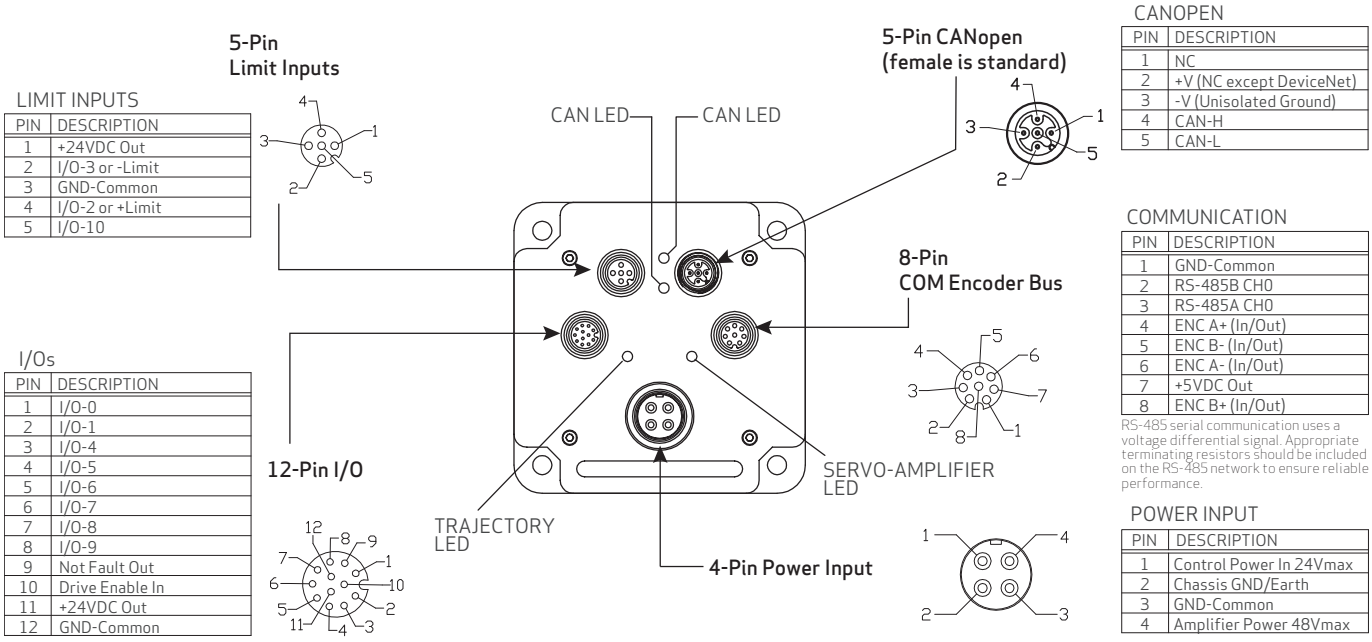
PIN	DESCRIPTION
A1	+20V to +48V DC
A2	Power Ground
1	I/O-6
2	+5VDC Out
3	RS-232 Transmit
4	RS-232 Receive
5	RS-232 Ground

PIN	DESCRIPTION
1	NC
2	+V (NC except DeviceNet)
3	-V (Isolated GND)
4	CAN-H
5	CAN-L

NOTE: The DE power option is recommended. For details, see the *Class 5 SmartMotor™ Installation & Startup Guide*.

M-Style Motor Connectors and Pinouts

The following figure provides a brief overview of the connectors and pinouts available on the M-style SmartMotors. For details, see the *Class 5 SmartMotor™ Installation & Startup Guide*. For LED functions, see Status LEDs on page 32.



LIMIT INPUTS

PIN	DESCRIPTION
1	+24VDC Out
2	I/O-3 or -Limit
3	GND-Common
4	I/O-2 or +Limit
5	I/O-10

I/Os

PIN	DESCRIPTION
1	I/O-0
2	I/O-1
3	I/O-4
4	I/O-5
5	I/O-6
6	I/O-7
7	I/O-8
8	I/O-9
9	Not Fault Out
10	Drive Enable In
11	+24VDC Out
12	GND-Common

CANOPEN

PIN	DESCRIPTION
1	NC
2	+V (NC except DeviceNet)
3	-V (Unisolated Ground)
4	CAN-H
5	CAN-L

COMMUNICATION

PIN	DESCRIPTION
1	GND-Common
2	RS-485B CH0
3	RS-485A CH0
4	ENC A+ (In/Out)
5	ENC B- (In/Out)
6	ENC A- (In/Out)
7	+5VDC Out
8	ENC B+ (In/Out)

RS-485 serial communication uses a voltage differential signal. Appropriate terminating resistors should be included on the RS-485 network to ensure reliable performance.

POWER INPUT

PIN	DESCRIPTION
1	Control Power In 24Vmax
2	Chassis GND/Earth
3	GND-Common
4	Amplifier Power 48Vmax

Cable Diagram

The DeviceNet network specification allows for branch or star configurations. However, the network communications will be most reliable when a straight bus (also called the "trunk" line) with single drops is used (see the following figure).

The DeviceNet network can support up to 64 devices (each node on the network receives an address number from 0 to 63). If additional devices are needed, repeaters, bridges, routers and gateways can be added. For more details, see The Specification for DeviceNet, which is available at the ODVA.org website.

NOTE: Address number 63 is typically reserved for system commissioning.

Common problems with network wiring are often traced to branches or star configurations. If not designed properly, these configurations can create multipath signal reflections that cause communication errors.

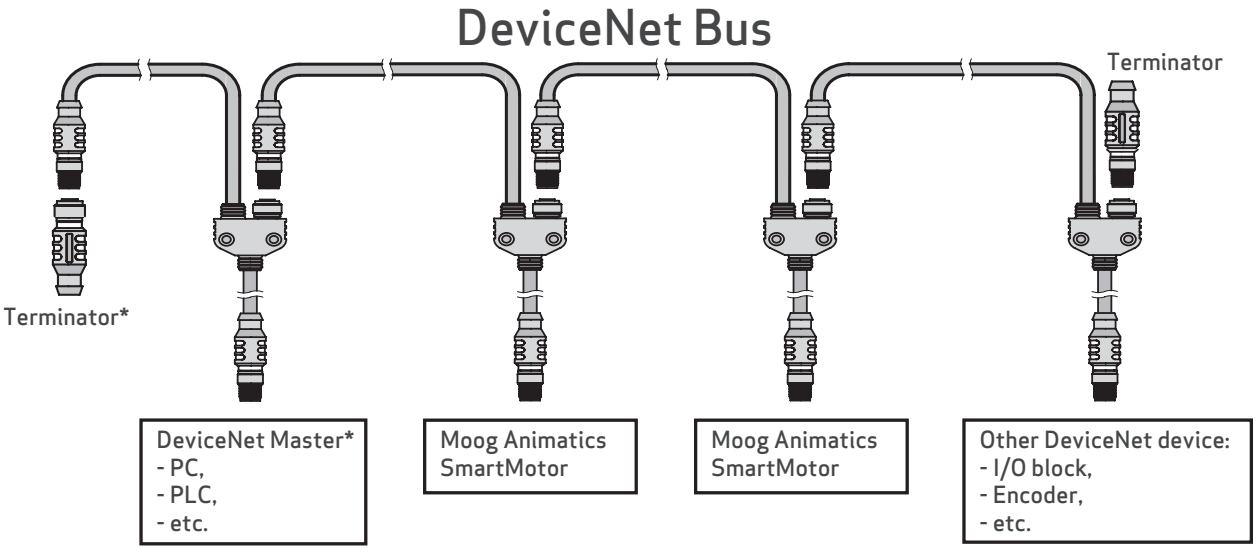
CAUTION: If a branch is absolutely necessary due to wiring constraints, it is the responsibility of the system designer to test and prove the layout is not causing communication errors. Moog Animatics cannot ensure the success of branched layouts.

The following figure shows a straight network with no branches. The short drop to each motor is acceptable. These drops from the Y connector to the motor should be 0.3 meters or less.

CAUTION: If drops from the Y connecter to the motor need to exceed 0.3 meters, it is up to the system designer to test and prove the additional drop length is not causing communication errors. Moog Animatics cannot ensure the success of longer drops.

The wire length between any two motors should be at least 0.1 meter, including the drop length. For details, see Maximum Bus Length on page 31.

Multidrop Cable Diagram



*Master may have termination option; see master's documentation for details.

Proper termination is critical for successful network communications. There must be two terminators (120 Ohms each), and they must be located at the two ends of the network (typically called the "trunk"). Because the network trunk is a straight line, there are exactly two ends of the network to place the terminators.

CAUTION: Using less than two terminators is not acceptable; using more than two terminators is not acceptable.

NOTE: Not shown in the previous figure are drops from the trunk that can have more than one device on them. When implementing multiple devices on a drop, it is important to follow ODVA guidelines for robust DeviceNet network operation.

If the master device specifically provides a terminating resistor, then that may be used instead of the terminator plug. However, in that case, the master must be at one end of the bus; it cannot be in the middle.

Maximum Bus Length

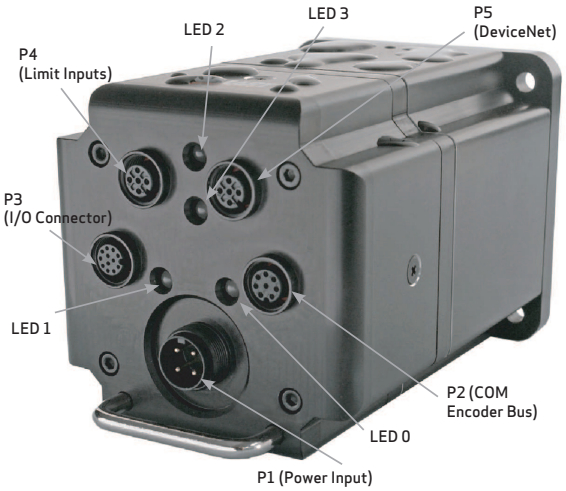
The following table shows the DeviceNet transmission bit rates and corresponding maximum bus lengths when thick cable is used for the main trunk line. The bus length is the calculated maximum distance of the straight bus from one terminated end to the other terminated end.

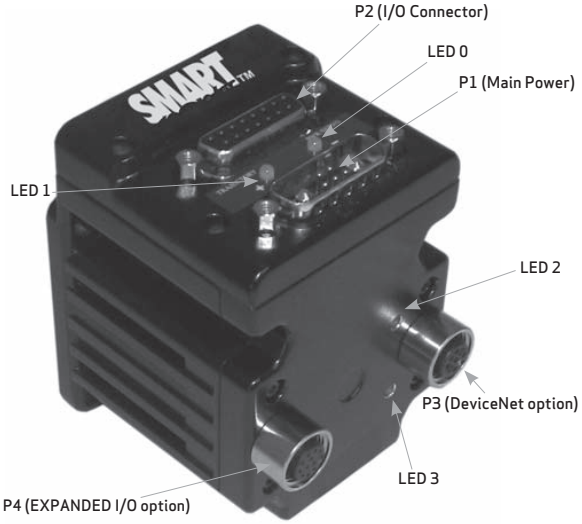
Bit rate (bits/second)	Thick Cable Bus Length (meters)
500000	100
250000	250
125000	500

NOTE: Maximum bus length of 500 meters requires round, large-diameter (thick) cable. If small diameter or flat cable is used, the maximum bus length will be reduced.

Status LEDs

The Status LEDs provide the same functionality for the D-style and M-style (including IP-sealed) SmartMotors.





LED 0: Drive Status Indicator

Off	No power
Green solid	Drive on
Green flashing	Drive off
Red flashing	Watchdog fault
Red solid	Major fault
Alt. red/green	In boot load, needs firmware

LED 2: Network Status (Red/Green LED)

Off	No error
Green flashing	Online, not connected or allocated to a master
Green solid	Online, at least one connection allocated to a master
Red flashing	One or more I/O connections in timed-out state
Red solid	Communications failed, offline; possible MacID conflict or errors across network

LED 1: Trajectory Status Indicator

Off	Not busy
Solid green	Drive on, trajectory in progress

LED 3: Module Status (Red/Green LED)

Off	Power off or not applied correctly
Green solid	Normal operation
Red solid	Unrecoverable fault, needs factory service
Red flashing	Recoverable fault
Green flashing	Configuration missing
Alt. red/green	Self test

LED Status on Power-up:

- With no program and the travel limit inputs are low:
  LED 0 will be solid red indicating the motor is in a fault state due to travel limit fault.
  LED 1 will be off.
- With no program and the travel limit inputs are high:
  LED 0 will be solid red for 500 milliseconds and then begin flashing green.
  LED 1 will be off.
- With a program that disables only travel limits and nothing else:
  LED 0 will be solid red for 500 milliseconds and then begin flashing green.
  LED 1 will be off.

DeviceNet on Class 5 SmartMotors

This section provides an overview of the DeviceNet communications protocol implementation on the Moog Animatics Class 5 SmartMotor.

DeviceNet Implementation

This section describes DeviceNet implementation information for the Class 5 SmartMotor.

For Class 4 SmartMotor Users

The DeviceNet implementation on the Class 5 SmartMotor has been designed to be very similar to that of the Moog Animatics Class 4 DN2 SmartMotors. Therefore, users having previous experience with those DeviceNet motors should transition easily to the Class 5 SmartMotors. For those users, it is recommended to review the information in the following chapters:

- For DeviceNet Command details, see Object Reference on page 57
- For DeviceNet I/O details, see Position Controller Implicit I/O Messages on page 45

DeviceNet Identity

The following identity information is available when the SmartMotor is queried by the DeviceNet master.

- Product Codes:
    - 2 – Class 5 SMxx195M (OEM Specific Motor)
    - 3 – Class 5 SMxx165D-Series Motors
    - 4 – Class 5 SMxx165M-Series Motors
- Device Name: Factory data in nonvolatile EEPROM memory.
- Serial Number: Factory data in nonvolatile EEPROM memory.

NOTE: These identity items match those shown on the SmartMotor name plate.

DeviceNet Software Version Numbers

The initial DeviceNet software release is version 1.001

In the version number, the Major and Minor versions are separated by the decimal point. In this case, the Major version = 1, and the Minor version = 001

Important EEPROM Addresses

The following nonvolatile EEPROM addresses contain important DeviceNet information for the Class 5 SmartMotors:

EEPROM Address	Description
32465	Network Timeout Action: 0 to 9, default= 0
32466	DeviceNet MACID: 0 to 63, default=63
32467	DeviceNet Baud Rate Index: 4=125, 3=250, 2=500, default=125 Kb
32512	DeviceNet Serial Number

NOTE: The default values are considered the DeviceNet "out of the box" configuration or DeviceNet Identity Object Reset "Service 1" condition.

Device Profile

The Class 5 DeviceNet SmartMotor profile uses the Position Controller (0x10) device. For more information, see Position Controller Device (0x10) on page 43.

SmartMotor Device Profile Overview

This section provides an overview of the objects used in the SmartMotor device profile. It includes: CIP required objects and network objects (the CIP objects), ODVA "Device" set of objects (the Application objects), and Moog Animatics vendor-specific objects (the Additional objects).

For a full description of each object, see the corresponding "For details..." section. For more details on the Position Controller Device, see Position Controller Device (0x10) on page 43.

CIP Objects for DeviceNet Devices

The following table shows the minimum objects required for any DeviceNet device.

Object Class	Class Code	Description	Required/ Optional	Instances	For details, see...
Identity Object	0x01	Provides identification and general device information. Object is required in every network device.	Required	1	Identity Object (0x01) on page 60
Message Router Object	0x02	Provides message handling for communicating with objects in the physical device.	Required	1	Message Router Object (0x02) on page 63
DeviceNet Object	0x03	Provides the configuration and status of a physical attachment to DeviceNet. A product must support one (and only one) DeviceNet Object per physical network attachment.	Conditional (required for DeviceNet)	1	DeviceNet Object (0x03) on page 64
Connection Object	0x05	Establishes and manages the communications connections (exchanges of messages).	Conditional (required for DeviceNet)	2 (Explicit, I/O)	Connection Object (0x05) on page 66

Application Objects for Position Controller Devices

The Position Controller Device type is 0x10; there is one instance. The following table shows the required application objects for a Position Controller device.

Object Class	Class Code	Description	Required/ Optional	Instances	For details, see...
CIP Required Objects	(see previous table)	(see previous table)	Required	(see previous table)	(see previous table)
Position Controller Supervisor	0x24	Handles errors for Position Controller, also home and registration inputs.	Required	1	Position Controller Supervisor (0x24) on page 70
Position Controller	0x25	Performs control output velocity profiling; handles input/output to/from the motor, limit switches registration, etc.	Required	1	Position Controller (0x25) on page 72

Additional Objects

In addition to the object classes in the previous tables, the following manufacturer-specific and other objects have been added for the SmartMotor.

Object Class	Class Code	Description	Required/ Optional	Instances	For details, see...
SmartMotor I/O Object	0x71	Manufacturer-specific object associated with the Position Controller Device		1	I/O Object (0x71) on page 1

EDS File

The Electronic Data Sheet (EDS) file is supplied by the equipment manufacturer. It is an ASCII text file that is structured as specified by the CIP specification. The EDS file contains all of the necessary information for the configurable parameters of the corresponding device (i.e., it contains information required by the CIP specification and may include vendor-specific information provided by the manufacturer). For example, there is an EDS file supplied by Moog Animatics for the DeviceNet SmartMotor.

All CIP network configuration tools have the ability to read EDS files. The information in the EDS file is used by the configuration tool to guide the user through the configuration process.

For more details, see *THE CIP NETWORKS LIBRARY, Volume 1: Common Industrial Protocol (CIP™)*, which is available on the ODVA.org website at: http://www.odva.org.

To obtain the EDS file for the SmartMotor:

1. Access the Download Center on the Moog Animatics website at:

   https://www.animatics.com/products/smartmotor.resources.html
2. From the Fieldbus Configurator Files list, select DeviceNet.
3. Locate the EDS file (.eds extension) and click it.
4. Save the file to a location on your computer.

DeviceNet User Program Commands

This chapter provides details on the DeviceNet commands used with the SmartMotor and its user program. SmartMotor programming is described in the *SmartMotor™ Developer's Guide*. The SmartMotor user program allows the motor to take on autonomous or distributed control functions needed in an application.

NOTE: The CAN network must have all devices set to the same baud rate for proper operation.

Address and Baud Rate Commands

This section describes the address and baud rate commands for DeviceNet.

CADDR=frm

Set CAN address

Where frm is a number from 1 to 127. The value is stored in the SmartMotor's EEPROM. However, the SmartMotor must be powered off and on for it to take effect.

CBAUD=frm

Set CAN baud rate

Where frm may be one of the following bit rates (bits/second): 125000, 250000 or 500000. The value is stored in the SmartMotor's EEPROM. However, the SmartMotor must be powered off and on for it to take effect.

Settings other than those shown are not supported.

=CBAUD, RCBAUD

Get CAN baud rate

These commands are used to assign/report the CAN baud rate as one of the values shown above.

- Assigned to a program variable: x=CBAUD
- As a report: RCBAUD

x=CADDR, RCADDR

Get node ID

These commands are used to assign/report the node ID.

- Assigned to a program variable: x=CADDR
- As a report: RCADDR

CAN Error Reporting Commands

This section describes the CAN error reporting commands for DeviceNet.

=CAN, RCAN

Get CAN error

The =CAN and RCAN commands are used to assign/report errors and certain status information for the CAN bus.

- Assigned to a program variable: x=CAN(y)
- As a report: RCAN(y)

Where y is the following:

Assignment	Report	Description
=CAN(0)	RCAN(0)	Gets the CAN bus status bits: (*Indicates an error bit) 0 CAN Power Okay 1* DeviceNet COM fault occurred (Duplicate MACID failure or CAN MAC Bus is OFF) 2 DeviceNet Power Ignore option enabled 3 Reserved 4* User attempted a Combitronic read from broadcast address 5* Combitronic debug, internal issue. 6* Timeout (Combitronic client) 7* Combitronic server ran out of buffer slots 8* Errors reached warning level 9* Receive Errors reached warning level 10* Transmit Errors reached warning level 11* Receive Passive Error 12* Transmit Passive Error 13* Bus Off Error 14* Rx buffer 1 overflowed 15* Rx buffer 0 overflowed
=CAN(1)	RCAN(1)	Network Access State Machine state: 1 Sending duplicate MACID 2 Waiting duplicate MACID 3 SmartMotor is online 4 SmartMotor device is faulted 5 SmartMotor is waiting for DeviceNet power at CAN network connector
=CAN(2)	RCAN(2)	Connections Allocated: Bit 0 - Explicit connection Bit 1 - Polled connection
=CAN(3)	RCAN(3)	Master MACID from open connections
=CAN(4)	RCAN(4)	Established Connections: Bit 0 - Explicit connection Bit 1 - Polled connection (expected packet rate was set)
=CAN(5)	RCAN(5)	Timed Out connections: Bit 0 - Explicit connection Bit 1 - Polled connection
=CAN(6)	RCAN(6)	Expected packet rate of Explicit connection
=CAN(7)	RCAN(7)	Expected packet rate of Polled connection

The =CAN(x) and RCAN(x) commands are used to report conditions that could occur over the CAN bus or conditions specific to the DeviceNet network. Not all bits are error bits. Therefore, it cannot be assumed that a non-zero value for RCAN is an error.

RCAN, which is the same as RCAN(0), reports a decimal number that is a combination of the bits shown in the =CAN(0) or RCAN(0) row of the previous table. Use the CAN command, which is the same as =CAN(0), in a program to assign the decimal number to a variable, for example:

```
x=CAN
```

A calculator with a binary display function can convert this decimal number to indicate the set of bits shown. Also, the *SmartMotor Developer's Worksheet* can be used for this conversion. It is available from the Moog Animatics website at:

https://www.animatics.com/support/downloads.knowledgebase.html

Network Control Commands

This section describes the network control commands for DeviceNet.

CANCTL(action, value)

Control network features

Commands execute based on the action argument, which controls CAN functions.

NOTE: DeviceNet is only one protocol a SmartMotor can support over the CAN network. Some setting are nonvolatile and saved in the EEPROM to maintain behavior from one power-up to the next.

Action =	Description
0	Network Timeout Action. Setting is nonvolatile. This action uses the following value arguments: • Value = 0: Servo off • Value = 1: Smooth stop • Value = 2: Hard stop • Value = 3: Motor reset • Value = 4: No action • Value = 5-9: GOSUBn
1	Reset the CAN communications controller in the motor and all errors. Resets the CANopen or DeviceNet protocol in the motor. The value argument is ignored.
2	Override the DeviceNet power status. This action uses the following value arguments: • Value = 1: The Network Access State Machine attempts to go online even if the DeviceNet network power is off. Once online, it will not drop connections if network power turns off. • If value is anything but one (1): Behavior is reset to the default, and the DeviceNet Access State Machine behaves per the ODVA specification. For more details, see *THE CIP NETWORKS LIBRARY, Volume 3: DeviceNet Adaptation of CIP*, which is available on the ODVA.org website at: http://www.odva.org. The setting is nonvolatile and retained from one power-up to the next. Therefore, it can be executed through the SMI Terminal and remembered thereafter.
3	Reset the CANopen interpolation buffer through a user command. The value argument is ignored.

Action =	Description
4	Force entry into CANopen interpolation mode through a user command. This action uses the following value arguments: • Value = 7: Force interpolation mode
5	Set Combitronic timeout. The value argument specifies the time in milliseconds; it defaults to 30 (for 30 milliseconds).
99	Change behavior. This action uses the following value arguments: • Value = 0: Resume default behavior. • Value = 1: Override the drive enable input if equipped (see following Note). If there are no faults, the drive can be enabled. • Value= 2: Change the fault input bit in I/O assembly response to be the Servo Bus Voltage status. • Value= 99: Override the drive enable input and enable reporting of the bus voltage status through DeviceNet. **NOTE:** The drive enable hardware input is only available on Class 5 IP SmartMotors with firmware versions 5.97.x.x and 5.96.x.x.

Position Controller Device (0x10)

This chapter provides information on the Position Controller Device (0x10). For more details, see *THE CIP NETWORKS LIBRARY, Volume 1: Common Industrial Protocol (CIP™)*, which is available on the ODVA.org website at: http://www.odva.org.

Position Controller Device Application Objects

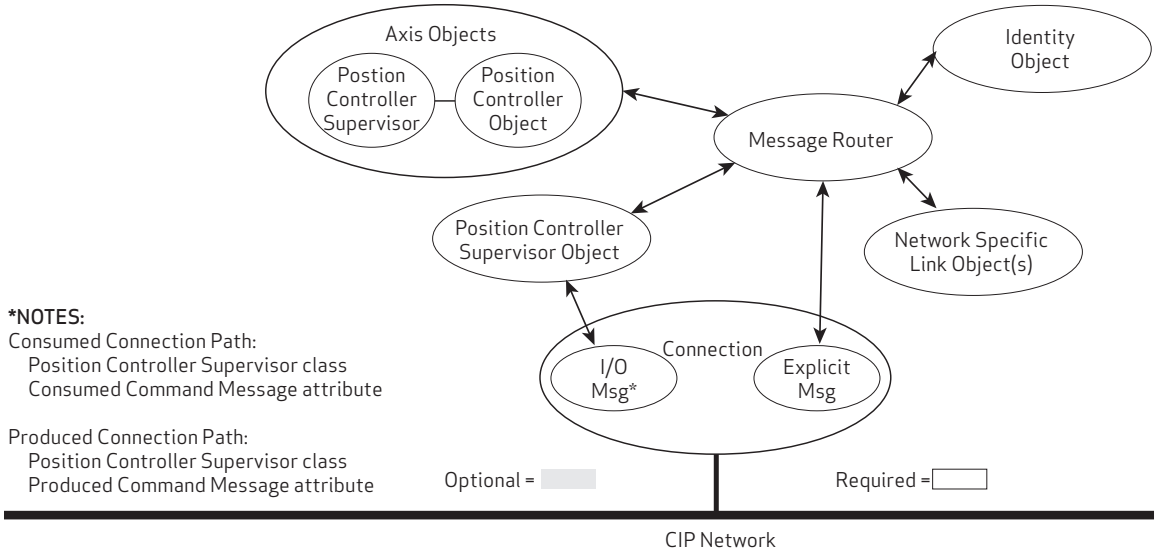
The following table shows the application objects for the Position Controller device and describes their functions. For a full description of each object, see the corresponding "For details..." section.

Object Class	Class Code	Description	Required/ Optional	Instances	For details, see...
Position Controller Supervisor	0x24	Handles errors for the Position Controller as well as Home and Registration inputs.	Required	1	Position Controller Supervisor (0x24) on page 70
Position Controller	0x25	Performs the control output velocity profiling and handles input and output to and from the motor drive unit, limit switches registration etc.	Required	1	Position Controller (0x25) on page 72

Note that these objects are the Application Objects in the overall SmartMotor device profile. For a listing of all objects in the device profile used for the SmartMotor, see SmartMotor Device Profile Overview on page 35.

Position Controller Device Object Model

The following figure provides a diagram of the Position Controller Device object model.



Position Controller Device Object Model

Position Controller Implicit I/O Messages

This section describes the details about implicit I/O messages for the device.

Over DeviceNet, this is the Polled IO Connection; the data frames are intended to fit within the 8 bytes of a CAN data packet.

General Command and Response Message Types

This section describes the formats for consumer and producer general messages.

NOTE: These message types for the Position Controller Polled I/O Connection are typically the most used and most valuable to command the integrated motor shaft position or applied torque.

Polled I/O: Consumed General Message Format

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Enable	Reg. Arm[1]	Hard Stop	Smooth Stop	Direction (Vel. mode)	Incremental	Start Block[1]	Load Data/ Start Profile
1	Block #							
2	Command Axis Number			Command Message Type (1 through 5)				
3	Response Axis Number			Response Message Type (1 through 5)				
4	Command Data Low Byte							
5	Data Middle Low							
6	Data Middle High							
7	Command Data High Byte							

Notes:
1. Byte 0 bits 6 and 1 are not supported in Class 5 motors.
2. For Semantics, refer to Command Message Semantics on page 48.

Polled I/O: Produced General Message Format

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Enable	Reg. Level[1]	Home Level[1]	Current Direction	General Fault	On Target Position	Block in Execution[1]	Profile in Progress
1	Block #							
2	Load Complete	Block Fault[1]	Following Error Fault	Negative Limit	Positive Limit	Reverse Limit	Forward Limit	Fault Input[1,2]
3	Response Axis Number			Response Message Type (1 through 5)				
4	Response Data Low Byte							
5	Data Middle Low							
6	Data Middle High							
7	Response Data High Byte							

Notes:
1. Not supported.
2. Byte 2 bit 0 can be configured as the Servo Bus Voltage Okay status. See CANCTL(action, value) on page 41.
3. For Semantics, refer to Response Message Semantics on page 49.

General Message Types

Refer to the consumed bytes 2 and 3, and the response byte 3 in the previous tables. When setting the "Start Profile" (Bit 0) to true, the Mode setting has to match the desired command type in order to start the profile move.

Command/Response Message Type	Command Data	Response Data	Class 0x25 Attr#3 Mode Setting
1	Target Position	Actual Position	0
2	Target Velocity	Command Position	1
3	Acceleration	Actual Velocity	N/A
4	Deceleration	Command Velocity	N/A
5	Torque	Torque	2

Attribute GET/SET Command Types 0x1A and 0x1B

This section describes the formats for consumer and producer GET/SET messages. These message types allow the set and get services of the Position Controller Supervisor(type 0x1A) and Position Controller(type 0x1B) classes and their objects attributes. This is a form of indirect addressing over the Polled I/O Connection instead of using an Explicit Connection transfer.

Polled I/O: Consumed Message Format

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Enable	Reg. Arm[1]	Hard Stop	Smooth Stop	Direction (Vel. mode)	Incremental	Start Block[1]	Load Data/ Start Profile
1	Attribute Number to GET							
2	Command Axis Number			Command Message Type (**0x1A or 0x1B**)				
3	Attribute Number to SET							
4	Command (SET) Data Low Byte							
5	Data Middle Low							
6	Data Middle High							
7	Command Data High Byte							

Notes:
1. Byte 0 bits 6 and 1 are not supported Class 5 motors.
2. For Semantics, refer to Command Message Semantics on page 48.

Polled I/O: Produced Message Format

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Enable	Reg. Level[2]	Home Level[2]	Current Direction	General Fault	On Target Position	Block in Execution[2]	Profile in Progress
1	GET Attribute Number for the Response Data							
2	Load Complete	Block Fault[2]	Following Error Fault	Negative Limit	Positive Limit	Reverse Limit	Forward Limit	Fault Input Fault[1]
3	Response Axis Number			Response Message Type (**0x1A or 0x1B**)				
4	Response (GET) Data Low Byte							

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
5	Data Middle Low							
6	Data Middle High							
7	Response Data High Byte							

Notes:
1. Byte 2 bit 0 can be configured as the Servo Bus Voltage Okay status. See CANCTL(action, value) on page 1.
2. Not supported.
3. For Semantics, refer to Response Message Semantics on page 1.

Attribute Message Types

NOTE: See byte 3 in the previous tables.

Message Type	Class Number	Class Description	Command Data	Response Data
26(0x1A)	36(0x24)	Position Controller Supervisor	Attribute Value to Set	Attribute Value to Get
27(0x1B)	37(0x25)	Position Controller	Attribute Value to Set	Attribute Value to Get

Error Response Message Type (0x14)

This section describes the formats for producer Error Response messages. It also provides the error codes for the Position Controller Device.

Polled I/O: Produced Error Response Message Format

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Enable	Reg. Level[1]	Home Level[1]	Current Direction	General Fault	On Target Position	Block in Execution[1]	Profile in Progress
1	Reserved = 0							
2	Load Complete	Block Fault[1]	Following Error Fault	Negative Limit	Positive Limit	Reverse Limit	Forward Limit	Fault Input[1,2]
3	Response Axis Number			Response Message Type (0x1A or 0x1B)				
4	General Error Code							
5	Additional Error Code							
6	Copy of Command message Byte 2							
7	Copy of Command Message Byte 3							

Notes:
1. Not supported.
2. Byte 2 bit 0 can be configured as the Servo Bus Voltage Okay status. See CANCTL(action, value) on page 41.
3. For Semantics, refer to Response Message Semantics on page 49.

Error Codes for Position Controller Device

General Error Code	Additional Error Code	Error Description	Semantics
0x08	0x01	Service Not Supported	Command message type is not supported; this will take precedence over additional code 2
	0x02	Service Not Supported	Response message type not supported
0x05	0x01	Path Destination Unknown	Consumed axis number was requested that does not exist
	0x02	Path Destination Unknown	Asked to produce data for a axis number that does not exist
0x09	0xFF	Invalid Attribute Value	Value typically out of range
0x0E	0xFF	Attribute Not Settable	Requested to modify something not changeable
0x13	0xFF	Not Enough Data	I/O Command message was lees then 8 bytes
0x14	0xFF	Attribute Not Supported	Requested or specified attribute is not supported

Semantics for Command and Response Messages

This section provides semantic information for the command and response messages in the previous tables.

Command Message Semantics

This following table provides semantic information for the command messages.

Item	Cmd Msg Type	Description
Load Data/ Start Profile		Set from zero to one to load command data. The transition of this bit from zero to one will also start a Profile Move when the command message type contained in the command message field is the message type that starts a Profile Move for the mode selected.
Start Block		Not supported.
Incremental		This bit is used to define the position value as either absolute or incremental. 0 = absolute position value and 1 = incremental position value.
Direction (V. Mode)		This bit is used to control the direction of the motor in Velocity mode. 1 = forward, positive; 0 = reverse, negative.
Smooth Stop		This bit is used to bring the motor to a controlled stop at the currently implemented deceleration rate.
Hard Stop		This bit is used to bring the motor to an immediate stop.
Registration Arm		Not supported.

Item	Cmd Msg Type	Description
Enable		This bit is used to control the enable output. Clearing this bit will set the enable output inactive and the currently executing motion profile will be aborted.
Block #		This byte defines the block number to be executed when the Start Block bit transitions from zero to one.
Command Message Type		This field defines the Command Message Type
Response Message Type		This field defines the Response Message Type
Command Axis Number		These three bits define the Consumed Axis Connection attribute of the Position Controller Supervisor class. This attribute value specifies the instance number of all of the axis objects whose data are contained in the I/O command message. The SmartMotor is a single-axis device. Therefore, only axis 1 is used, which can be specified by either 0 or 1.
Target Position	0x01	This double word defines the Profile Move's Target Position in position units, when the Load Data /Start Profile bit transitions from zero to one.
Target Velocity	0x02	This double word defines the Profile Move's Target Velocity in profile units, when the Load Data /Start Profile bit transitions from zero to one
Acceleration	0x03	This double word defines the Profile Move's Acceleration in profile units, when the Load Data /Start Profile bit transitions from zero to one.
Deceleration	0x04	This double word defines the Profile Move's Target Position in profile units, when the Load Data /Start Profile bit transitions from zero to one.
Torque	0x05	This double word is used to set the output torque, when the Load Data /Start Profile bit transitions from zero to one. The torque value will only take effect when in torque mode. (Position Controller Object Attribute 3 = 2)
Attribute Value	0x1A-0x1B	This double word defines the value of the attribute to set, when the Load Data/Start Profile bit transitions from zero to one.
Object Attribute to Get	0x1A-0x1B	This byte defines the object attribute to get the value of and return in the response message.
Object Attribute to Set	0x1A-0x1B	This byte defines the object attribute to set to the new value defined by the Attribute Value when the Load Data/Start Profile bit transitions from zero to one.

Response Message Semantics

This following table provides semantic information for the response messages.

Item	Description
Profile in Progress	This bit indicates that a profile move is in progress.
Block in Execution	Not supported.

Item	Description
On Target Position	This bit indicates whether or not the motor is on the last targeted position. (1 = Current position equals the last target position.)
General Fault	This bit indicates the logical "or" of all fault conditions.
Current Direction	This bit shows the current direction of the motor. If the motor is not moving the bit indicates the direction of the last commanded move. 0 = reverse, negative direction; 1 = forward, positive direction.
Home Level	Not supported.
Registration Level	Not supported.
Enable	This bit indicates the state of the enable output. A 1 indicates the enable output is active.
Executing Block #	This byte defines the currently executing block if the Block In Execution bit is active.
Fault Input	Not supported.
Forward Limit	This bit indicates that the forward input is active.
Reverse Limit	This bit indicates that the reverse input is active.
Positive Limit	This bit indicates that the motor has attempted to travel past the programmed positive limit position. This bit remains valid until the motor is moved within the limits or the programmed limit value is set greater than the current position.
Negative Limit	This bit indicates that the motor has attempted to travel past the programmed negative limit position. This bit remains valid until the motor is moved within the limits or the programmed limit value is set less than the current position.
Following Error Fault	This bit indicates that a following error fault has occurred. This fault occurs when the following error, or difference between the commanded and actual position, exceeds the programmed allowable following error.
Block Fault	Not supported.
Load Complete	This bit indicates that the command data contained in the command message has been successfully loaded into the device.
Response Message Type	This byte defines the Response Message Type
Response Axis Number	These three bits report the Produced Axis Connection attribute of the Position Controller Supervisor class. This attribute value specifies the instance number of all of the axis objects whose data is contained in the I/O response message. The SmartMotor is a single-axis device. Therefore, only axis 1 is used, which can be reported as either 0 or 1.
Actual Position	This double word reflects the actual position in position units. If position feedback is not used, this word will report the commanded position.
Commanded Position	This double word reflects the commanded or calculated position in position units.
Actual Velocity	This double word reflects the actual velocity in profile units.
Command Velocity	This double word reflects the commanded or calculated velocity in profile units.
Torque	This double word reflects the torque.
Home Position	This double word reflects the captured home position in position units.
Index Position	This double word reflects the captured index position in position units.

Item	Description
Registration Position	This double word reflects the captured registration position in position units.
General Error Code	This byte identifies an error has been encountered. The specific behavior for the Position Controller Profile is summarized in Error Response Message Type (0x14) on page 47. For a complete list of General Error codes, see Appendix B in *THE CIP NETWORKS LIBRARY Volume 1 - Common Industrial Protocol (CIP™)*, which is available on the ODVA.org website.
Additional Code	This byte contains an object/service-specific value that further describes the error condition. If the responding object has no additional information to specify, then the value 0xFF is placed within this field.
Attribute Value	This double word reflects the value of the attribute to get.
Object Attribute to Get	This byte defines the object attribute from which to get the value.

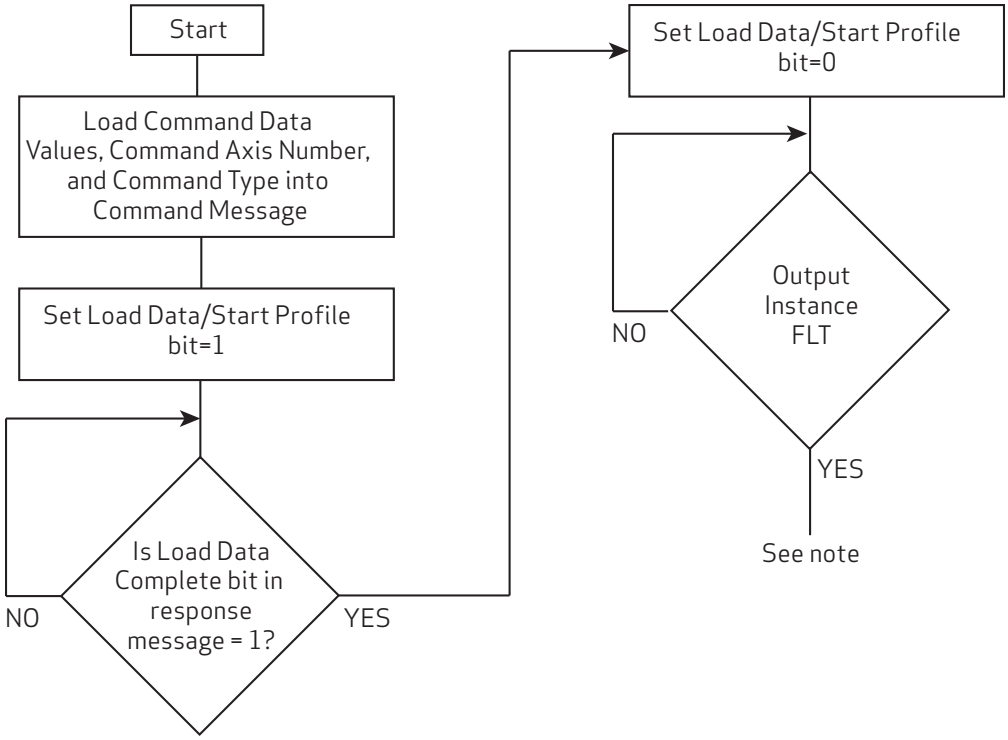
Position Controller I/O Handshaking

This section describes I/O handshaking for the Position Controller device.

NOTE: References to "client" are from the viewpoint of the Master device.

Client Data Loading Procedure

The following figure describes the client data loading procedure.

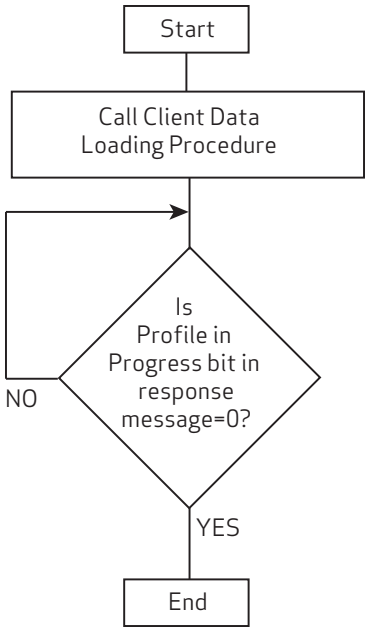


Client Data Loading Procedure

NOTE: When the Event Data Block is present, additional elements are included based on the Event Checking Status element. If the Extended Format bit is set in the Event Checking Status word, then the following are included: Reg Data Ack, Home Data Ack, and Watch Data Ack. The Event Block Count field determines the repetition (from zero to seven times) of the following elements: Event ID #, Event Status #, Event Type #, Event Position #, and Event Time Stamp #.

Client Profile Move Procedure

The following figure describes the client profile move procedure.

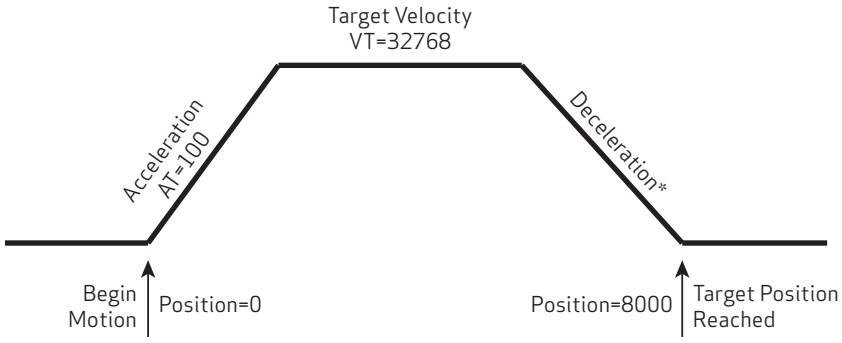


Client Profile Move Procedure

For more details, see *THE CIP NETWORKS LIBRARY, Volume 1: Common Industrial Protocol (CIP™)*, which is available on the ODVA.org website at: http://www.odva.org.

Profile Moves

Attribute 3 of the Position Controller Object determines how the device operates and responds for a profile move. A profile move uses Acceleration (and Deceleration) and a Target Velocity to operate the device at the Target Velocity or move it to a Target Position. It can also output a Torque if the Position Controller (0x25) Attribute 3 value is set for that. Refer to the following figure—the values shown are the same ones used in the I/O Assembly Examples later in this section.



*NOTE: For the SmartMotor, there is an override that automatically sets DT equal to AT if the motor power is turned on and only AT is set.

Motion Profile For I/O Assembly Examples

For more details, see *THE CIP NETWORKS LIBRARY, Volume 1: Common Industrial Protocol (CIP™)*, which is available on the ODVA.org website at: http://www.odva.org.

Torque Command

Attribute 3 of the Position Controller Object determines how the device operates. The SmartMotor can output a Torque command to the integrated motor if the Position Controller (0x25) Attribute 3 value is set for Torque mode using a setting of 2 for the attribute.

Refer to the *SmartMotor™ Developer's Guide* for further information on Torque mode.

Control Mode Change - Change Dynamic

With the SmartMotor, it is possible to change from one control mode to another during operation or while in a move profile in any other mode. In other words, while operating in Position mode with Position Controller mode Attribute 3 set to the value 0 and using Implicit Message Type 1, the programmer can change to Torque mode by setting Attribute 3 to the value 2 and then executing a new "Start Profile" with Implicit Message Type 5.

This "Change Dynamic" feature can be fully executed using the Implicit Messages of the Position Controller Device. This is facilitated using Message Type 0x1B during a profile once it has started, and then starting another profile after changing the mode attribute.

Position Controller Implicit I/O Message Examples

These examples are meant for reference only. They depict the Frame data on the bus over the Implicit Connection for the Position Controller Device (0x10) I/O assembly to create a profile move (refer to the following description). When operating on DeviceNet, these frames are for the Polled I/O Implicit Connection.

NOTE:
To simplify the presentation, these examples do not include extended data transfer across the network.

SmartMotor Notes

For proper operation, external hardware input conditions must be satisfied to get the desired results. For example, to enable the drive stage on the model SM34165MT-IP SmartMotor, the drive-enable input (pin 10 of the 12-pin I/O connector) must be at 24 Volts (enabled).

The SmartMotor is a single axis. Therefore, the value 0 or 1 may be used for the axis number in the I/O Data (the examples typically use 0 for easier reading). Also, the symbol "0x??" is used to indicate there are bits within the byte that are determined by the present state of the SmartMotor.

The following are further assumptions for the SmartMotor:

- Motor sample rate set to: 8kHz
- Motor counts per revolution: 4000
- Motor mode: Position
- Hardware Travel Limits are Satisfied or Disabled
- SmartMotor Hardware Enable is Satisfied

NOTE: In the next two examples, some items in the first row of each are colored to aid comprehension.

Set Acceleration

For the following example, Command Frame = CF, Response Frame = RF.

Frame Type		Frame Data (Hexadecimal Bytes)	Result
CF	0-7	01 00 23 21 78 7D 01 00	(0x23 = 0010 0011) Axis 1, ADT=100, set acceleration using units for this object of: counts / sec^2: (100/65536)*8000^2 = 97656 (0x00017D78)
RF	0-7	?? 00 8? 21 00 00 00 00	Axis 1 at Position 0
Clear the Load Data, Power Stage ON			
CF	0-7	80 00 01 01 00 00 00 00	Ask for position
RF	0-7	8? 00 0? 01 00 00 00 00	Axis 1 ON, Position 0

Set Velocity, Leave Drive ON

For the following example, Command Frame = CF, Response Frame = RF.

Frame Type		Frame Data (Hexadecimal Bytes)	Result
CF	0-7	8D 00 02 01 A0 0F 00 00	(0x02 = 0000 0010) Axis 1 (note that 0 is interpreted as axis 1) VT=32768, set velocity using units for this object of: counts / sec: (32768/65536)*8000 = 4000 (0x00000FA0)
RF	0-7	8? 00 8? 01 00 00 00 00	Axis 1 at Position 0
Clear the Load Data			
CF	0-7	80 00 01 01 00 00 00 00	Ask for position
RF	0-7	8? 00 0? 01 00 00 00 00	Axis 1 ON, Position 0

Set Target Position, Perform Move

For the following example, Command Frame = CF, Response Frame = RF.

Frame Type		Frame Data (Hexadecimal Bytes)	Result
CF	0-7	81 00 01 01 40 1F 00 00	PT=8000 G
RF	0-7	81 00 8? 01 00 00 00 00	Axis 1 at Position 0
Clear the Load Data (assumes motor completed move)			
CF	0-7	80 00 01 01 00 00 00 00	Ask for position
RF	0-7	80 00 0? 01 40 1F 00 00	Position 8000

Disable Hardware Limits (Object 0x25, Attribute 49)

For the following example, Command Frame = CF, Response Frame = RF.

Frame Type		Frame Data (Hexadecimal Bytes)	Result
CF	0-7	01 31 1B 31 E0 00 00 00	
RF	0-7	0? 31 8? 1B E0 00 00 00	
Clear the Load Data			
CF	0-7	00 00 01 01 00 00 00 00	Ask for position
RF	0-7	0? 00 0? 01 00 00 00 00	At Position 0
NOTE: If the SmartMotor is on, this will turn it off.			

Object Reference

This chapter provides details on the DeviceNet protocol objects used with the Moog Animatics SmartMotor. The following TOC provides a listing of those objects.

The Moog Animatics Class 5 SmartMotor device profile supports the required object classes of the Position Controller Device 0x10. For more details, see *THE CIP NETWORKS LIBRARY, Volume 1: Common Industrial Protocol (CIP™)*, which is available on the ODVA.org website at: http://www.odva.org.

Required Objects

The following sections/tables list the attributes for each of the CIP required and network objects.

Identity Object (0x01)

The following tables provide the Class and Identity Attributes, and the Supported Services for the Identity Object (0x01) for the Class 5 DeviceNet SmartMotors.

Class Attributes

None required, no services needed.

Identity Instance 1 Attributes

Attrib ID	Access Rule	Name	Data Type	Description	Semantics
1	Get	Vendor ID	UINT	Identification of each vendor by number. Value = 810	Moog Animatics unique number
2	Get	Device Type	UINT	Indication of general type of product. Value = 16	Position Controller
3	Get	Product Code	UINT	Identification of a particular product of Moog Animatics.	Product codes is set at factory see previous product code section.
4	Get	Revision	STRUCT of:	Revision of the item the Identity Object represents.	Zero is not valid for either the Major or Minor Revision fields.
		Major Revision	USINT	Incremented by the vendor when there is a significant change to the product fit, form or function.	Set by Moog Animatics factory during fabrication of the SmartMotor.
		Minor Revision	USINT	Identifies product changes that do not affect user configuration choices.	Set by Moog Animatics factory during fabrication of the SmartMotor.
5	Get	Status	WORD	Summary status of device.	See Attribute 5: Status
6	Get	Serial Number	UDINT	Serial number of SmartMotor.	Set by factory during fabrication.
7	Get	Product Name	SHORT_ STRING	SmartMotor model	Set by factory during fabrication

Instance Attributes Semantics

Attribute 5: Status

NOTE: SmartMotor Identity Revision 1.1 and earlier do not update this attribute and always return ZERO (0); this is a known deviation from the ODVA specifications and fixed in later revisions.

Bits	Name	Description
0	Owned	TRUE indicates the device (or an object within the device) has an owner. In a Master/Slave configuration, the setting of this bit means that the Predefined Master/Slave Connection Set has been allocated to a master. Otherwise, the meaning of this bit is TBD.
1		Reserved; 0.
2	Configured	TRUE indicates the application of the device has been configured to do something other than the default behavior. This does not include communications configuration.
3		Reserved; 0.
4-7	Extended Device Status	Vendor specific or as defined by the following table. The EDS shall indicate if the device follows a vendor-specific definition for these bits by using the DeviceStatusAssembly keyword. See Default Values for Bits 4-7.
8	Minor Recoverable Fault	TRUE indicates the device detected a problem with itself, which is thought to be recoverable. The problem does not cause the device to go into one of the faulted states.
9	Minor Unrecoverable Fault	TRUE indicates the device detected a problem with itself, which is thought to be unrecoverable. The problem does not cause the device to go into one of the faulted states.
10	Major Recoverable Fault	TRUE indicates the device detected a problem with itself, which caused the device to go into the Major Recoverable Fault state.
11	Major Unrecoverable Fault	TRUE indicates the device detected a problem with itself, which caused the device to go into the Major Unrecoverable Fault state.
12-15	Extended Device Status 2	Reserved; either 0 or vendor specific.

Default Values for Bits 4-7

Value	Description
0	Self-testing or unknown
1	Firmware update in progress
2	At least one faulted I/O connection
3	No I/O connections established
4	Nonvolatile configuration bad
5	Major Fault – either bit 10 or bit 11 is true (1)
6	At least one I/O connection in run mode
7	At least one I/O connection established, all in idle mode
8-9	Reserved
10-15	Vendor specific[1]
1. No mechanism is provided to enumerate the vendor-specific use of these values in the EDS file.	

Services

Service Code	Supported in:		Name	Description
	Class	Instance		
0x05			Reset	Invokes the Reset service for the device.
0x0E	•	•	Get_Attribute_Single	Returns the contents of the specified attribute

For more details, see *THE CIP NETWORKS LIBRARY, Volume 1: Common Industrial Protocol (CIP™)*, which is available on the ODVA.org website at: http://www.odva.org.

Message Router Object (0x02)

The following sections provide the Class and Identity Attributes, and the Supported Services for the Message Router Object (0x02).

Class Attributes

Access Services are not required, and not available over DeviceNet.

Instance Attributes

Access Services are not required, and not available over DeviceNet.

Services

Services are not required for proper DeviceNet operation.

For more details, see *THE CIP NETWORKS LIBRARY, Volume 1: Common Industrial Protocol (CIP™)*, which is available on the ODVA.org website at: http://www.odva.org.

DeviceNet Object (0x03)

The following tables provide the Class and Identity Attributes, and the Supported Services for the DeviceNet Object (0x03).

Class Attributes

Attrib ID	Access Rule	Name	Data Type	Description	Semantics
1	Get	Revision	UINT	Revision of this object. Value = 2	Revision of Class definition in which object implementation is based.

Instance Attributes

Attrib ID	Access Rule	Name	Data Type	Description	Semantics
1	Set/Get	MACID	USINT	Node Address Range:0-63 Setting is nonvolatile	Modification(Set) will delete all connections and the SmartMotor will re-execute Network Access State Machine.
2	Set/Get	Baud Rate	USINT	Baud Rate setting: 0 = 125 Kbps 1 = 250 Kbps 3 = 500 Kbps Setting is nonvolatile	Modification(Set) will take effect on power cycle or a RESET service to the SmartMotor's Identity Object.
3	Get	Bus-Off Interrupt (BOI)	BOOL	Bus-Off Interrupt setting: 0 = Hold CAN MAC in reset 1 = Not supported Setting is nonvolatile	Held in RESET as seen by the Network Status LED.
4	Get	Bus-Off Counter	USINT	Number of times CAN MAC went to the bus-off state	Range: 0-255
5	Get	Allocation Information	STRUCT of:		
		Allocation Choice Byte	BYTE	Active Predefined Master/Slave Connections	Initialized to 0 at power-up/reset
		Master's MACID	USINT	MACID of Master (from Allocate)	Range: 0-63, 255 (Modified by Allocate only)

Services

Service Code	Supported in:		Name	Description
	Class	Instance		
0x0E	•	•	Get_Attribute_Single	Returns the contents of the specified attribute

Service Code	Supported in:		Name	Description
	Class	Instance		
0x10		•	Set_Attribute_Single	Modifies a single attribute
0x4B		•	Allocate_Master/Slave_ Connection_Set	Requests the use of the Predefined Master/Slave Connection Set
0x4C		•	Release_Master/Slave_ Connection_Set	The specified Connections within the Predefined Master/Slave Connection Set are no longer desired and will be released (deleted).

For more details, see *THE CIP NETWORKS LIBRARY, Volume 3: DeviceNet Adaptation of CIP*, which is available on the ODVA.org website at: http://www.odva.org.

Connection Object (0x05)

The following tables provide the Class and Identity Attributes, and the Supported Services for the Connection Object (0x05).

Class Attributes

Attrib ID	Access Rule	Name	Data Type	Description	Semantics
1	Get	Revision	UINT	Revision of this object. Value = 1	The current assigned value is 1. If updates require an increased value, the value increases by 1.

Attributes for Instance 1: Explicit Connection

Attrib ID	Access Rule	Name	Data Type	Description
1	Get	State	USINT	Indicates the connection status: 0 = Non-Existent, 1 = Configuring, 2 = Waiting ID, 3 = Established, 4 = Timed OUT, 5 = Deferred Delete, 6 = Closing.
2	Get	Instance_type	USINT	Indicates this is an Explicit Messaging Connection with a value = 0.
3	Get	TransportClass_ trigger	BYTE	Connection is a Server/Cyclic/Class 2 or 3. Value = 0x83
4	Get	Produced_ connection_id	UINT	CAN Identifier for connection transmit, set during connection creation.
5	Get	Consumed_ connection_id	UINT	CAN Identifier for connection received, set during connection creation.
6	Get	Initial_comm_ characteristics	BYTE	Initial Production and Consumption; Value = 0x21
7	Get	Produced_ connection_size	UINT	Maximum number of bytes transmitted across this Connection; no specified default
8	Get	Consumed_ connection_size	UINT	Maximum number of bytes received across this Connection; no specified default
9	Set/Get	Expected_packet_ rate	UINT	Defines timing (in milliseconds) associated with this Connection; Default Value: 2500
12	Set/Get	Watchdog_ timeout_action	USINT	Defines how to handle Inactivity/Watchdog timeouts action: 1 = Auto_Delete (default), 3 = Deferred Delete. Default Value: 1
13	Get	Produced_ connection_path_ length	UINT	Number of bytes in the produced_connection_path attribute Value = 0
14	Get	Produced_ connection_path	Packed EPATH	Empty return for explicit connection

Attrib ID	Access Rule	Name	Data Type	Description
15	Get	Consumed_ connection_path_ length	UINT	Number of bytes in the consumed_connection_path attribute, Value = 0
16	Get	Consumed_ connection_path	Packed EPATH	Empty return for explicit connection

Attributes for Instance 2: Polled I/O Connection

Attrib ID	Access Rule	Name	Data Type	Description
1	Get	State	USINT	Indicates the connection status: 0 = Non-Existent, 1 = Configuring, 2 = Waiting ID, 3 = Established, 4 = Timed OUT, 5 = Deferred Delete, 6 = Closing.
2	Get	Instance_type	USINT	Indicates this is an I/O Connection the value = 1.
3	Get	TransportClass_ trigger	BYTE	Connection is a Server/Cyclic/Class 2 or 3. Value = 0x83
4	Get	Produced_ connection_id	UINT	CAN Identifer for connection transmit, set during connection creation.
5	Get	Consumed_ connection_id	UINT	CAN Identifier for connection received, set during connection creation.
6	Get	Initial_comm_ characteristics	BYTE	Initial Production and Consumption; Value = 0x01
7	Get	Produced_ connection_size	UINT	Produced data packet size, Value = 8
8	Get	Consumed_ connection_size	UINT	Consumed data packet size, Value = 8
9	Set/Get	Expected_packet_ rate	UINT	Expected packet rate must be configured during instantiation of the connection; Default Value: 0
12	Get	Watchdog_ timeout_action	USINT	Transition to the Timed Out state Value = 0
13	Get	Produced_ connection_path_ length	UINT	Value = 6
14	Get	Produced_ connection_path	Packed EPATH	fixed = 0x20,0x24,0x24,0x00,0x30,0x21 hex
15	Get	Consumed_ connection_path_ length	UINT	Value = 6
16	Get	Consumed_ connection_path	Packed EPATH	fixed = 0x20,0x24,0x24,0x00,0x30,0x20 hex

Services

Service Code	Supported in:		Name	Description
	Class	Instance		
0x0E		•	Get_Attribute_Single	Used to read a Connection Object attribute.
0x10		•	Set_Attribute_Single	Used to modify a Connection Object attribute.

For more details, see *THE CIP NETWORKS LIBRARY, Volume 1: Common Industrial Protocol (CIP™)*, which is available on the ODVA.org website at: http://www.odva.org.

Application Objects

The following sections/tables list the attributes for each of the Application objects (ODVA "device" set of objects).

Position Controller Supervisor (0x24)

The following tables provide the Class and Identity Attributes, and the Supported Services for the Position Controller Supervisor Object (0x24).

Class Attributes

Attrib ID	Access Rule	Name	Data Type	Description	Semantics
1	Get	Revision	UINT	Revision of this object. Value = 2	The current assigned value is 2. If updates require an increased value, the value increases by 1.
32	Get	Consumed Axis Selection Number	USINT	Specifies the axis number to which the data contained in the I/O Command Message is routed.	SmartMotor value is typically 1 for this attribute.
33	Get	Produced Axis Selection Number	USINT	Specifies the axis number to which the data contained in the I/O Response Message is routed.	SmartMotor value is typically 1 for this attribute.

Object Instance 1 Attributes

Attr ID	Access Rule	Name	Data Type	Description
1	Get	Number of Attributes	USINT	Returns the total number of attributes supported by this object in this device.
2	Get	Attribute List	Array of USINT	Returns an array with a list of the attributes supported by this object in this device.
3	Get	Axis Number	USINT	Returns the axis number which is the same as the instance of the object.
4				Reserved
5	Get	General Fault	BOOL	Bit is a logical OR of all fault condition attribute flags. 1 = Fault condition
6	Get	Command Message Type	USINT	Returns the command message type that is being sent by the controlling device
7	Get	Response Message Type	USINT	Returns the response message type that is returned to the controlling device
15	Set/Get	Index Arm	BOOL	Used to arm the index input Values: 1 = arm index input, 0 = trigger occurred
18	Get	Index Position	DINT	The position at the time the home input is triggered.
25	Set/Get	Follow Enable	BOOL	Enables following of the Follow Axis. Values: 0 = disabled, 1 = enabled
27	Set/Get	Follow Divisor	DINT	Used to calculate the Command Position by dividing the Follow Axis position with this value.
28	Set/Get	Follow Multiplier	DINT	Used to calculate the Command Position by multiplying the Follow Axis position with this value.
100	Set/Get	Follow Type	USINT	Values: 0 = Step Dir, 1 = AqB

Services

Service Code	Supported in:		Name	Description
	Class	Instance		
0x0E	•	•	Get_Attribute_Single	Returns contents of specified attribute
0x10		•	Set_Attribute_Single	Modifies the attribute value

For more details, see *THE CIP NETWORKS LIBRARY, Volume 1: Common Industrial Protocol (CIP™)*, which is available on the ODVA.org website at: http://www.odva.org.

Position Controller (0x25)

The following tables provide the Class and Identity Attributes, and the Supported Services for the Position Controller Object (0x25).

Class Attributes

Attrib ID	Access Rule	Name	Data Type	Description	Semantics
1	Get	Revision	UINT	Revision of this object. Default Value: 2	The current assigned value is 2. If updates require an increased value, the value increases by 1.

Instance Attributes

In the following table, a "unit" is one count of the encoder resolution returned by attribute 40 (the Feedback resolution).

NOTE: The user program commands are not in the same units shown below. In other words, the report RVT value shown in the SMI Terminal window will not equal a get of Attr ID 7.

Attr ID	Access Rule	Name	Data Type	Description
1	Get	Number of Attributes	USINT	Returns the total number of attributes supported by this object in this device
2	Get	Attribute List	Array of USINT	Returns an array with a list of the attributes supported by this object in this device
3	Set/Get	Mode	USINT	Operating Mode: 0 = Position mode (default) 1 = Velocity mode 2 = Torque mode
6	Set/Get	Target Position	DINT	Profile Position value to set in Profile units (see Attr ID 40)
7	Set/Get	Target Velocity	DINT	Profile Velocity in units/sec, positive only
8	Set/Get	Acceleration	DINT	Profile Acceleration in units/sec-sec, positive only
9	Set/Get	Deceleration	DINT	Profile Deceleration in units/sec-sec, positive only.
10	Set/Get	Incremental Position Flag	BOOL	Defines Attr ID 6 as being either absolute or incremental: 0 = absolute move 1 = incremental move
11	Set/Get	Load Data/ Start Profile/ Profile in Progress	BOOL	On set, loads data and starts the current profile. On get, reports Profile in Progress
13	Set/Get	Actual Position	DINT	Actual absolute position. Can be Set to redefine actual position.
14	Get	Actual Velocity	DINT	Reports actual velocity in units/sec.
15	Get	Commanded Position	DINT	The instantaneous calculated position

Attr ID	Access Rule	Name	Data Type	Description
16	Get	Commanded Velocity	DINT	The instantaneous velocity in units/sec.
17	Set/Get	Enable	BOOL	On the enable edge, commanded position is set to equal actual position. Values: 0=disable 1=enable
20	Set/Get	Smooth Stop	BOOL	Smooth Stop motor
21	Set/Get	Hard Stop	BOOL	Hard Stop motor
23	Set/Get	Direction	BOOL	Instantaneous Direction Values: 0=reverse 1=forward Is set to change or select direction on Velocity Mode
24	Set/Get	Reference Direction	BOOL	Shaft Rotation Direction (forward facing shaft): 0 = CW 1 = CCW
25	Set/Get	Torque	DINT	Output Torque Range: -32767 to 32767
29	Get	Wrap Around	BOOL	Position Wrap Around Indicator Flag If 1, the motor has gone past its maximum position.
30	Set/Get	Kp	INT	Proportional Gain Range: 0 to 32767
31	Set/Get	Ki	INT	Integral Gain Range: 0 to 32767
32	Set/Get	Kd	INT	Derivative Gain Range: 0 to 32767
33	Set/Get	MaxKi	INT	Integration Limit Range: 0 to 32767
35	Set/Get	Velocity Feed Forward	INT	Velocity feed forward gain value Range: 0 to 32767
37	Get	Sample Rate	INT	Update sample rate in micro-seconds
40	Get	Feedback Resolution	DINT	Number of actual position feedback counts per revolution
41	Get	Motor Resolution	DINT	Motor resolution in motor steps. Number of motor steps in one revolution of the motor.
45	Set/Get	Max Dynamic Following Error	DINT	Maximum allowable following error when the motor is in motion
47	Set/Get	Following Error Fault	BOOL	Following error occurrence flag

Attr ID	Access Rule	Name	Data Type	Description
48	Get	Actual Following Error	DINT	Actual Following error
49	Set/Get	Hard Limit Action	USINT	Hard Limit Action code (applies to Soft Limit Action also): 0 = Servo off 1 = Hard Stop 2 = Smooth Stop 224 = Both hardware limits disabled
50	Get	Forward Limit	BOOL	Forward Limit stop input status active
51	Get	Reverse Limit	BOOL	Reverse Limit stop input status active
52	Set/Get	Soft Limits Enable	BOOL	Enables Soft Limits
53	Set/Get	Soft Limit Action	USINT	Soft Limit Action Code (applies to Hard Limit action, also): 0 = Servo off 1 = Hard Stop 2 = Smooth Stop
54	Set/Get	Positive Soft Limit Position	DINT	Soft limit positive boundary in position counts, enable BOTH soft limits
55	Set/Get	Negative Soft Limit Position	DINT	Soft limit negative boundary in position counts, enable BOTH soft limits
56	Get	Positive Limit Triggered	BOOL	Hard or Soft limit forward limit occurrence flag
57	Get	Negative Limit Triggered	BOOL	Hard or Soft limit forward limit occurrence flag
58	Get	Load Data Complete	BOOL	Valid data for a valid I/O command message type has been loaded into the position controller
100	Set/Get	Current Limit	DINT	Current limit of motor 0 to 1023
101	Set/Get	Reset Motor Faults	BOOL	Set to 1 for RESET action, the Get returns the Drive Ready Status
102	Set/Get	Reset Motor	BOOL	Set to 1 for RESET action, the Get returns the Drive Ready Status
103	Set/Get	Overheat Setpoint	USINT	Overheat setpoint 0-70 or 0-85 degrees C
104	Get	Temperature	INT	Real time temperature
105	Set/Get	Variable u	DINT	Motor user variable u
106	Set/Get	Variable v	DINT	Motor user variable v
107	Set/Get	Variable w	DINT	Motor user variable w
108	Set/Get	Variable x	DINT	Motor user variable x
109	Set/Get	GOSUBnnn	UINT	Setting executes a user program command GOSUBnnn. The Get returns the program running status.

Attr ID	Access Rule	Name	Data Type	Description
110	Get	Loss of Network Action	USINT	Action if DeviceNet network heartbeat lost: 0 = IGNORE (No Command) 1 = OFF (Motor Off) 2 = X (Smooth Stop) 3 = S (Hard Stop) 4 = GOSUB 5 = GOTO
111	Get	Status Word	WORD	Motor status word

Services

Service Code	Supported in:		Name	Description
	Class	Instance		
0x0E	•	•	Get_Attribute_Single	Returns the contents of the specified attribute
0x10		•	Set_Attribute_Single	Modifies the attribute value

Error Responses

Error Code	Additional Code	Name	Description
0x02	None	CIP_RESOURCE_UNAVAILABLE	Attribute 113 - Network GOSUB already active
0x09	None	CIP_INVALID_ATTRIB_VALUE	Attribute 113 - Invalid GOSUB number
0x10	None	CIP_DEVICE_STATE_CONFLICT	Attribute 113 - GOSUB stack overflow

For more details, see *THE CIP NETWORKS LIBRARY, Volume 1: Common Industrial Protocol (CIP™)*, which is available on the ODVA.org website at: http://www.odva.org.

Additional Objects

The following sections/tables list the attributes for the manufacturer-specific and other objects that don't fall into the previous categories.

SmartMotor I/O Object (0x70)

The following tables describe the attributes and instances for the SmartMotor I/O Object (0x70).

Attribute Table (Instance 0)

Attr ID	Access Rule	Name	Data Type	Description	Semantics
1	Get	I/O Word State	WORD	Returns the value of first 16 I/O points; bit 0 is I/O 0 (zero)	

Attribute Table (One instance per I/O pin)

Attr ID	Access Rule	Name	Data Type	Description	Semantics
1	Set/Get	Function	USINT	Sets (or gets) the function as Output, Input or Special.	Values: 0 = Output 1 = Input 2 = Special
2	Set/Get	Output State	BOOL	Sets (or gets) the output states as ON or OFF.	Values: 0 = OFF 1 = ON
3	Get	I/O State	BOOL	Returns present State of the I/O either ON or OFF.	Values: 0 = OFF 1 = ON
4	Get	Analog Raw Value	INT	Hardware dependent	

Instance	Standard Series		IP-M Series	
	I/O Number	Function	I/O Number	Function
1	0 A enc		I/O – 0	
2	1 B enc		I/O – 1	
3	2 C	Positive Limit	I/O – 2	Positive Limit
4	3 D	Negative Limit	I/O – 3	Negative Limit
5	4 E RS-485 IIC (SDA)		I/O – 4	
6	5 F RS-485 IIC (SCL)		I/O – 5	
7	6 G Index capture	Go	I/O – 6	Go
8	7 virtual bit		I/O – 7	
9	No Support		I/O – 8	External Brake
10	No Support		I/O – 9	
11	No Support		I/O – 10	
12	No Support		No Support	Output-11 Not Faulted

Instance	Standard Series		IP-M Series	
	I/O Number	Function	I/O Number	Function
13	No Support		No Support	Input-12 Drive Enable
…				
17	Ext 0		No Support	
18	Ext 1		"	
19	Ext 2		"	
20	Ext 3		"	
21	Ext 4		"	
22	Ext 5		"	
23	Ext 6		"	
24	Ext 7		"	
25	Ext 8		"	
26	Ext 9		"	

Troubleshooting

The following table provides troubleshooting information for solving SmartMotor problems that may be encountered when using DeviceNet. For additional support resources, see the Moog Animatics Support page at:

http://www.animatics.com/support.html

Issue	Cause	Solution
DeviceNet Communication Issues		
Master device does not recognize motor.	Motor not powered.	Check Drive Status LED. If LED is not lit, check wiring.
	Disconnected or miswired connector, or broken wiring between motors.	Check that connector is correctly wired and connected to motor. For details, see Connections, Wiring and Status LEDs on page 27.
	Wrong or duplicate MACID (address)	Check DeviceNet network MACID addresses.
	Wrong firmware	For Class 5 motors, the firmware should be 5.16.x.y or 5.97.x.y.
	Network flooded with traffic.	Stop user programs in all motors; make sure Combitronic communications is not interfering with DeviceNet. For details, see Class 5 User Guide.
Solid red error LED.	A warning or bus off condition has occurred.	A blinking red LED may indicate occasional issues from any of the causes listed above; a solid red LED indicates that these issues have occurred frequently, which causes the motor to stop communicating (bus off condition). In this case, the SmartMotor must be reset after fixing the cause of the problem.
Communication and Control Issues		
Motor control power light does not illuminate.	Motor is equipped with the DE option.	To energize control power, apply 24-48 VDC to pin 15 and ground to pin 14.
	Motor has routed drive power through drive-enable pins.	Ensure cabling is correct and drive power is not being delivered through the 15-pin connector.
Motor does not communicate with SMI.	Transmit, receive, or ground pins are not connected correctly.	Ensure that transmit, receive and ground are all connected properly to the host PC.
	Motor program is stuck in a continuous loop or is disabling communications.	To prevent the program from running on power up, use the Communications Lockup Wizard located on the SMI software Communications menu.
Motor disconnects from SMI sporadically.	COM port buffer settings are too high.	Adjust the COM port buffer settings to their lowest values.
	Poor connection on serial cable.	Check the serial cable connections and/or replace it.
	Power supply unit (PSU) brownout.	PSU may be too high-precision and/or

Issue	Cause	Solution
		undersized for the application, which causes it to brown-out during motion. Make moves less aggressive, increase PSU size, or change to a linear unregulated power supply.
Motor stops communicating after power reset, requires re-detection.	Motor does not have its address set in the user program. NOTE: Serial addresses are lost when motor power is off or reset.	Use the SADDR or ADDR= command within the program to set the motor address.
Red PWR SERVO light illuminated.	Critical fault.	To discover the source of the fault, use the Motor View tool located on the SMI software Tools menu.
Common Faults		
Bus voltage fault.	Bus voltage is either too high or too low for operation.	Check servo bus voltage. If motor uses the DE power option, ensure that both drive and control power are connected.
Overcurrent occurred.	Motor intermittently drew more than its rated level of current. Does not cease motion	Consider making motion less abrupt with softer tuning parameters or acceleration profiles.
Excessive temperature fault.	Motor has exceeded temperature limit of 85°C. Motor will remain unresponsive until it cools down below 80°C.	Motor may be undersized or ambient temperature is too high. Consider adding heat sinks or forced air cooling to the system.
Excessive position error.	The motor's commanded position and actual position differ by more than the user-supplied error limit.	Increase error limit, decrease load, or make movement less aggressive.
Historical positive/negative hardware limit faults.	A limit switch was tripped in the past.	Clear errors with the ZS command.
	Motor does not have limit switches attached.	Configure the motor to be used without limit switches by setting their inputs as general use.
Programming and SMI Issues		
Several commands not recognized during compiling.	Compiler default firmware version set incorrectly.	Use the "Compiler default firmware version option" in the SMI software Compile menu to select the default firmware version closest to the motor firmware version. In the SMI software, view the motor firmware version by right-clicking the motor and selecting Properties.

Reference Documents

This section lists the documents that were referenced for this guide.

ODVA Specifications

The following ODVA specifications were referenced for this guide:

- THE CIP NETWORKS LIBRARY, Volume 1: Common Industrial Protocol (CIP™), Edition 3.16, April 2014.
- THE CIP NETWORKS LIBRARY, Volume 3: DeviceNet Adaptation of CIP, Edition 1.14, November 2013.

These volumes comprise The DeviceNet™ Specification, which must be purchased using the order form on the ODVA.org website at:

https://secure.odva.org/forms/spec-vendor-id-order-form.htm

ODVA Libraries

The following ODVA libraries were referenced for introductory topics:

- The CIP Technology Library
- The DeviceNet Library

These ODVA libraries can be accessed directly on the ODVA.org website at:

http://www.odva.org/Publication-Download

TAKE A CLOSER LOOK

Moog Animatics, a sub-brand of Moog Inc. since 2011, is a global leader in integrated automation solutions. With over 30 years of experience in the motion control industry, the company has U.S. operations and international offices in Germany and Japan as well as a network of Automation Solution Providers worldwide.

Americas - West
Moog Animatics
2581 Leghorn Street
Mountain View, CA 94043
United States

Tel: +1 650-960-4215
Email: animatics_sales@moog.com

Americas - East
Moog Animatics
1995 NC Hwy 141
Murphy, NC 28906
United States

Europe
Moog GmbH
Memmingen Branch
Allgaeustr. 8a
87766 Memmingerberg
Germany

Tel: +49 8331 98 480-0
Email: info.mm@moog.com

Asia
Moog Animatics
Kichijoji Nagatani City Plaza 405
1-20-1, Kichijojihoncho
Musashino-city, Tokyo 180-0004
Japan

Tel: +81 (0)422 201251
Email: mcg.japan@moog.com

For Animatics product information, visit **www.animatics.com**

For more information or to find the office nearest you, email **animatics_sales@moog.com**

Moog Animatics Class 5 SmartMotor™ DeviceNet Guide, Rev. C
SC80100011-001