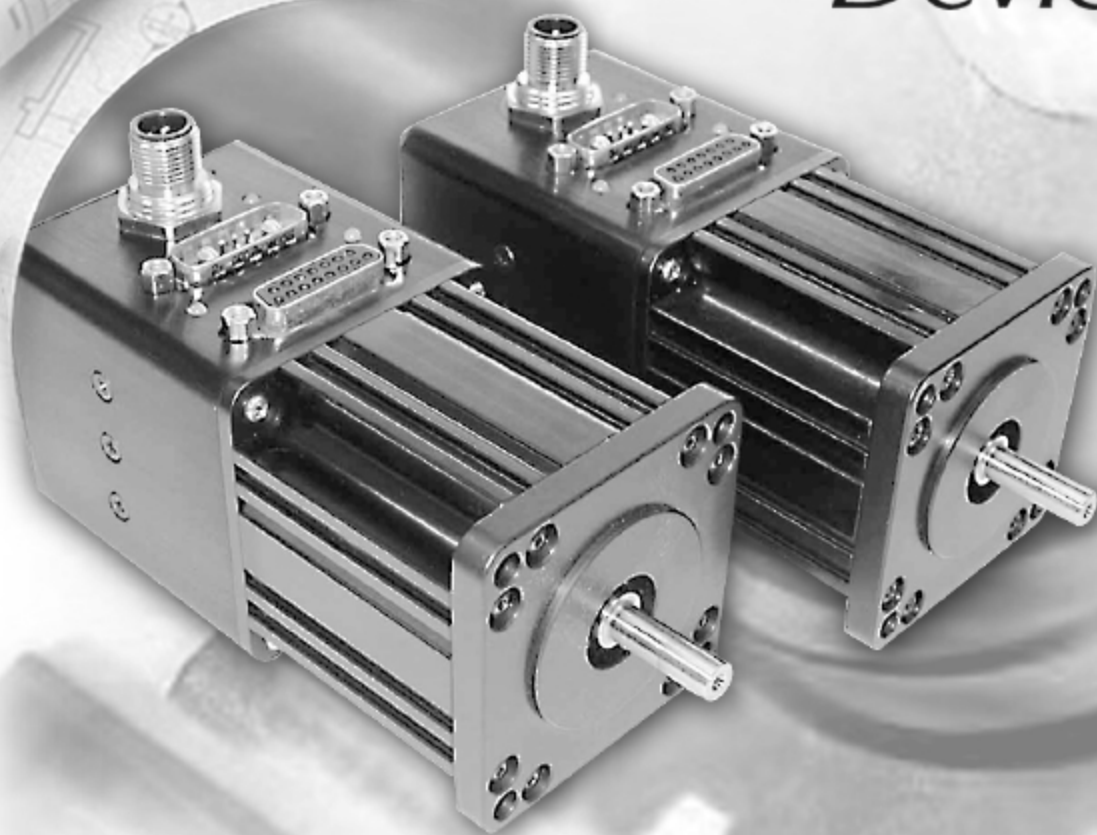


SMART
motor

with **DeviceNet**



SPECIFICATIONS



ANIMATICS®

Defining the Future in Motion Control

Unit Specifications

<i>Description</i>	<i>Min</i>	<i>Typical</i>	<i>Max</i>	<i>Units</i>
Absolute Maximum Ratings				
Current draw from on-board +5VDC output			210	mA DC
Ambient Temperature:				
Operating	0	0	+70	°C
Storage	0		+70	°C
RS-232 Operating Characteristics				
RS-232 Transmitters			± 12	± 12 VDC
RS-232 Receivers			± 12	VDC
Power supply requirements				
Motor Power Supply (20 to 48V)	0.1	Depends on Model	40	A
DeviceNet Supply (16 to 32VDC)			100	mA
I/O specifications				
Input TTL "ON" threshold	2.0			V
Input TTL "OFF" threshold			0.7	V
Output TTL "ON"	2.2			V
Output TTL "OFF"			0.7	V
Output Current, sinking			10	mA
Output Current, sourcing			4	mA
+5V DC Supply, Output Current		210		mA

DeviceNet Overview

DeviceNet is a low-cost, open network standard that provides for reduced system complexity and significant reductions in wiring costs. DeviceNet allows different industrial devices such as a Smartmotor™ and devices from other manufacturers (sensors, actuators, et al.) to work together on a single network, controlled by any number of commercially available DeviceNet master (or scanner) systems.. DeviceNet may also provide communications links between subsystems or system-level components. The results are improved control communications between devices and important device-level diagnostics. The DeviceNet Smartmotor™ can only be configured as a DeviceNet slave device, not a master.

DeviceNet is based on a broadcast-oriented, communications protocol called CAN (Controller Area Network). This protocol was originally developed for the European automotive market to replace expensive wire harnesses with low-cost network cable. As a result, CAN has a fast response time and is highly reliable. Consumer and commercial demand for CAN was a key factor in lowering the price and increasing the performance of CAN. With CAN, any node may transmit if the bus is not busy. If two or more nodes begin transmitting at the same time, the message with the lowest CAN ID will complete the transmission. DeviceNet adds a layer above CAN that allows logical connections to exist among nodes and defines message formats. A single DeviceNet node may have up to 64 nodes, each with a unique address (MAC ID). DeviceNet supports baud rates of 125, 250, and 500 KB. As the baud rate increases, the maximum allowable distance of cable between any two devices decreases. There is only one baud rate allowed per network, and all devices must operate at the same baud rate. The table below lists some of the major features of DeviceNet.

Table 1: Device Net Features

Feature	Description								
Network Size	Up to 64 nodes								
Network Length	Selectable end-to-end network distance varies with speed <table> <tr> <th>Baud Rate</th><th>Distance</th></tr> <tr> <td>125 Kbps</td><td>500 m (1640 ft)</td></tr> <tr> <td>250 Kbps</td><td>250 m (820 ft)</td></tr> <tr> <td>500 Kbps</td><td>100 m (328 ft)</td></tr> </table>	Baud Rate	Distance	125 Kbps	500 m (1640 ft)	250 Kbps	250 m (820 ft)	500 Kbps	100 m (328 ft)
Baud Rate	Distance								
125 Kbps	500 m (1640 ft)								
250 Kbps	250 m (820 ft)								
500 Kbps	100 m (328 ft)								
Data Packets	0-8 bytes								
Bus Topology	Linear (trunkline/dropline): power and signal on the same network cable								
Bus Addressing	Peer-to-Peer with Multicast (one-to-many); Multi-Master and Master/Slave special case; Polled or change-of-state (exception-based)								
System Features	Removal and replacement of devices from the network under power								

More DeviceNet information can be obtained from the ODVA at: <http://www.odva.org/>

Connector Information

Animatics SmartMotor™ uses a standard Device Net connector . The connecetor on the SmartMotor is a :
Micro-change, 5P, male, straight , M12 external thread.

PINOUT

- Pin 1: Drain *Not internally connected*
- Pin 2: V+ , DeviceNet Power *
- Pin 3: V- , DeviceNet Power *
- Pin 4: CAN H
- Pin 5: CAN L

* Must supply Device Net power 16 VDC to 32 VDC to Device Net connector.
SmartMotor™ internally isolates Device Net Power from controller and amplifier power.

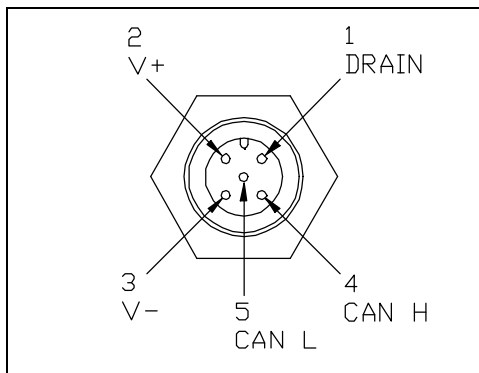


Figure 1: DeviceNet Connector Pinout
(Face View of Motor)

LED Status Indicators

DeviceNet motors are equipped with Net status and Mod status LED indicators, refer to Figure 2 and 3 for location, in addition to the trajectory and servo-amplifier status LEDs. The status of the LED indicators are:

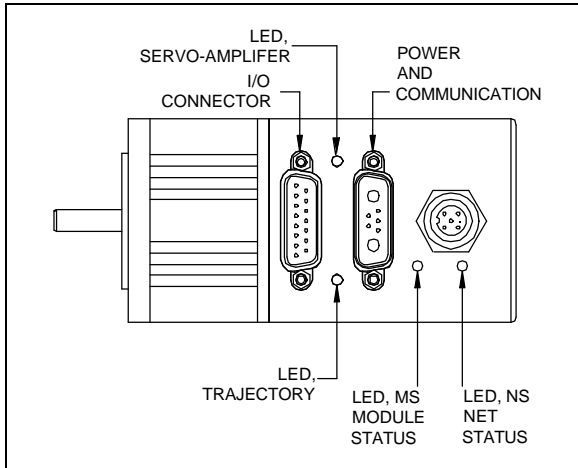


Figure 2: SM2300D-DN SERIES LED INDICATORS

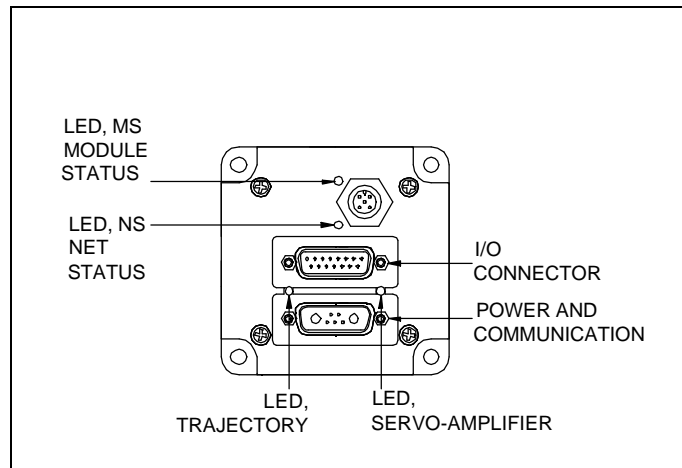


Figure 3: SM3400D-DN SERIES LED INDICATORS

MS, MODULE STATUS

GREEN=

FLASHING GREEN=

RED=

FLASHING RED=

FLASHING RED/GREEN=

OFF (CLEAR)=

Operating in normal condition.

Device is on stand-by and needs commissioning due to missing, incomplete or incorrect configuration. Have not yet established communications between the SmartMotor DeviceNet board and the SmartMotor controller.

Device has non-recoverable fault.

Device has recoverable fault.

Device is in self-test

No power to DeviceNet connector.

NS, NET STATUS

GREEN=

FLASHING GREEN=

RED=

FLASHING RED=

FLASHING RED/GREEN=

OFF (CLEAR)=

Online and has established connection to other devices.

On-line, but has no connection to other devices.

Failed communication. SmartMotor™ can not communicate on the network due to duplicate MAC ID or CAN BUSOFF error. Check MAC ID and Baudrate.

Check MAC ID and Baudrate.

One or more Device Net I/O are in timed out state.

Flashing Red/Green NS, in conjunction with the MS LED, indicates the device is in self-test.

No power to device connector.

The standard SmartMotor status indicate:

TRAJECTORY STATUS

GREEN=

OFF (CLEAR)=

In a calculated motion profile.

Not in a calculated motion profile or motion profile mode <default state>

SERVO-AMPLIFIER STATUS

GREEN=

Servo-amplifier on.

RED =

Stand by, servo- amplifier off <default state>

How DeviceNet works with our Smartmotor™

The DeviceNet Smartmotor™ is designed in a modular fashion, with the standard Smartmotor™ module adapted to work cooperatively with a DeviceNet gateway. The DeviceNet gateway uses a separate dedicated controller for DeviceNet operation, which means that varying network traffic demands will not affect the ability of the Smartmotor™ to handle motion and I/O tasks. The standard SM is equipped with two serial ports. These ports are configured such that one is RS-232 format and one is RS-485 format. On the DeviceNet version of the Smartmotor™, the RS-485 port has been retained for use with the DeviceNet gateway, **so the RS-485 port is no longer available for external use.**

Firmware Versions

Animatics' motion control firmware defines the motion modes, functions, I/O and communication functions that the SmartMotor™ .

The motion control firmware version that are compatible with the Devicenet are:

Standard firmware version: 4.15, 4.15B, 4.15C and newer

Optional Firmware versions: 4.75, 4.76 and newer

To find out which version of the firmware you have, send the command "RSP" to the serial communication line using an ASCII terminal or Animatic's SmartMotor™ Interface (SMI) software. The terminal screen show:

24576/415

Indicates firmware Version 4.15

24576/415B

Indicates firmware Version 4.15B

24576/415C

Indicates firmware Version 4.15C

24576/475

Indicates firmware Version 4.75

Setting the MAC ID and Baud Rate

The DeviceNet MAC ID and baud rate are set from non-volatile memory at power-up. The motor is initially set for 500 kbaud and MAC ID 3. To change these values, from your PC, over the RS232 serial port connected to the motor, using the Animatics SmartMotor™ Interface (SMI) software, in the SmartMotor™ Terminal window (Communicate>Talk to SmartMotors™), issue the following commands:

```
RSP      `returns version number—to verify communications with the
          `motor.
y=125    `desired baud rate in kbaud:  125, 250, or 500 .
z=3      `desired MAC ID: 0 to 63 .
EPTR=32004 `set EEPROM memory pointer.
VST(y,2) `store into EEPROM.
```

To verify the values have been set correctly, issue the following commands:

```
Rw      `report value of w, verify returns 0 .
Rx      `report value of x, verify returns 0 .
EPTR=32004 `set EEPROM memory pointer .
VLD(w,2) `retrieve data from EPROM .
Rw      `verify baudrate in kbaud .
Rx      `verify MAC ID .
```

Cycle power on the motor to cause the new values to take effect.

Consult the User's Guide for more information on developing user programs for the SmartMotor™.

Connecting to a DeviceNet network

The SmartMotor can connect to DeviceNet using standard micro-change cables and tees. Refer to figure for a sample set up.

The SmartMotor requires 16 to 32VDC provided to the DeviceNet connector from the DeviceNet in addition to the 20 to 48VDC that powers the SmartMotor controller/amplifier/motor .

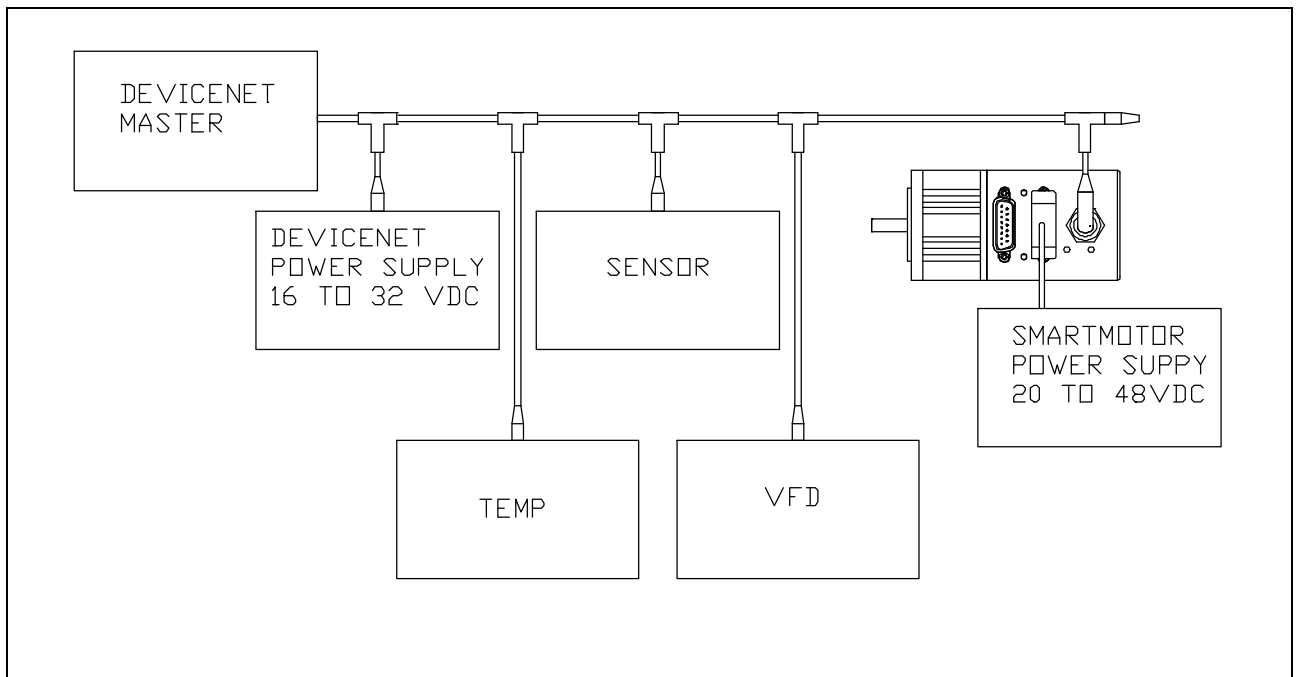


Figure 4: DeviceNet Network

How to use the EDS file

To configure your network to communicate with the DeviceNet Smartmotor™, you will need an .EDS file, which can be downloaded from our website at <http://www.animatics.com>. Be certain that the .EDS file's version number or revision matches your actual device before using the file. If the revision number is more recent than your physical device, it may contain parameters that aren't supported by the device. Use the software for the master scanner device to incorporate this configuration file into the network setup.

Scaling Motion Units

When reading/writing motion parameters to and from the SmartMotor™, there are some conversion factors that the user should be aware of:

Distance Units – There is a 1:1 correlation between distance units when using DeviceNet parameters to read/write values. Distances are measured in encoder counts.

Velocity Units – Due to the nature of our trajectory generator, there is a scale factor required when comparing DeviceNet velocity units and motor velocity units. There is a multiplier value of **16.10604266** to get from DeviceNet counts/sec to internal velocity values. In other words, setting an actual speed of **1000** counts/sec_r would result in a motor **V** value of **16,106** units. On a motor with an encoder resolution of **4000** counts/rev, this would result in an approximate speed of **0.25** revs/sec.

Acceleration Units - Again, due to the nature of our trajectory generator, there is a scale factor required when comparing acceleration units. There is a divisor value of **252.6362193** to get from counts/sec² to internal acceleration values. In other words, setting an actual acceleration of **4,000,000** counts/sec², would result in a motor **A** value of **15,833** units (remember that Version 4.xx SmartMotors only support integer math, so we must use an integer). On a motor with an encoder resolution of **4000** counts/rev, this would result in an approximate acceleration of **1000** revs/sec².

DeviceNet control of the Smartmotor™

To send and receive messages to and from the DeviceNet SmartMotor™, the user must thoroughly understand the command structures used by DeviceNet. DeviceNet communications are via explicit messaging (GET/SET attribute) and/or I/O Polling. The I/O polling messages conform to a subset of the Position Controller Device Profile, specified in ODVA DeviceNet Specification Volume II Release 2.0 Errata 3, sections 3-12.4, 3-12.5, 6-24, and 6-25. Please refer to these specifications as a reference for detailed descriptions.

Using DeviceNet in conjunction with internal user programs

The motor may execute a user program simultaneously with DeviceNet operation subject to the following rules:

1. The program must contain the open Devicenet communication channel command near the beginning of the program. The open devicenet communication command is :
 - i. **OCHN(RS4,1,N,19200,1,8,C) 'Reserve channel 1 for Device Net activity**
2. The program must not access communications channel 1, which are I/O pin E and I/O pin F located on pin 5 and pin 6 on the DB15 connector.
3. The program must not modify variables a through e. These are used internally for Device Net activity.
4. The program must not conflict with the effects of commands being sent through DeviceNet.

Execution of the program will tend to increase DeviceNet latency. PRINTs, especially long PRINTs to channel 0 should be avoided. Communications polling (on channel 0) should also be avoided if possible. A program intended to run only on power up or in specific circumstances should END, so the full processing power of the motor can be devoted exclusively to DeviceNet .

WAIT=nnn commands interspersed in the user program would decrease the proportion of communication time devoted to the user program. During a WAIT or a TWAIT command, the user program would not affect DeviceNet latency.

The length of commands should be kept to a minimum, not to exceed 16 characters including the command terminator (space or enter). Pending DeviceNet channel 1 commands, serial port channel 0 commands, and user program commands are executed one at a time, in round robin order. High bandwidth applications may need to give consideration to the total command data flow to the motor from all three sources.

Explicit Messages - Object Class, Instance, and Attribute List

This is a listing of attribute id, access rule, name, DeviceNet data type, Description, and a “Rosetta stone” correspondence between DeviceNet Get and Set actions and SmartMotor™ ASCII language commands (where there is a translation).

Objects described are the Position Controller Supervisor class and object instance, the Position Controller class and object instance, and a (vendor-specific) SmartMotor™ I/O class and object instances for the motor's digital i/o pins.

Table 2: Position Controller Supervisor, Class 36 decimal, Instance 0 (Class 0)

Attribute ID	Access Rule	Name	DeviceNet Data Type	Description of Attribute	Get Action	Set Action
1	Get	Revision	USINT	Revision of this object	02	N/A
32	Get	Consumed Command Message	Array of byte	Content of the Command Message		N/A
33	Get	Produced Response Message	Array of byte	Content of the Response Message		N/A

Table 3: Position Controller Supervisor, Class 36 decimal, Instance 1

Attribute ID	Access Rule	Name	DeviceNet Data Type	Description of Attribute	Get Action	Set Action
3	Get	Axis Number	USINT	Returns the axis number (which is the same as the instance for this object).	fixed at 1	N/A
5	Get	General Fault	BOOL	General Fault flag. 1=fault	RW, logical OR of status bits 1, 2, 4, 5, 6, 11, 12, 13, 14 & 15	N/A
6	Get ¹	Command Message Type	USINT	The command message type being sent by the controlling device		N/A ¹
7	Get ¹	Response Message Type	USINT	The response message type to be returned to the controlling device		N/A ¹
15	Set	Index Arm	BOOL	Index trigger arm flag used to arm the Index input		0: 1: RI
18	Get	Index Position	DINT	Index trigger position reflects the position at the time the index is triggered	RI	N/A
25	Set	Follow Enable	BOOL	0=disabled 1=enabled	RMODE == F or S or X-> 1	1, <encoder>: MF4 MFR G 1, <step+dir>: MS MSR G 0:
27	Set	Follow Divisor	DINT	Used to calculate the Command Position by dividing the Follow Axis position with this value	a=MFDIV Ra	MFDIV=nnn
28	Set	Follow Multiplier	DINT	Used to calculate the Command Position by dividing the Follow Axis position with this value	a=MFMUL Ra	MFMUL=nnn
100	Set	Follow Type	USINT	0=Step+Dir 1=Encoder		

Table 4: Position Controller, Class 37 decimal, Instance 0 (Class 0)

Attribute ID	Access Rule	Name	DeviceNet Data Type	Description of Attribute	Get Action	Set Action
1	Get	Revision	USINT	Revision of this object	02	N/A

Table 5: Position Controller, Class 37 decimal Instance 1

Attribute ID	Access Rule	Name	DeviceNet Data Type	Description of Attribute	Get Action	Set Action
1	Get	Number of Attributes	USINT	Returns the total number of attributes supported by this object in this device	fixed	N/A
2	Get	Attribute List	Array of USINT	Returns an array with a list of the attributes supported by this object in this device	fixed	N/A
3	Set	Mode	USINT	Operating Mode	RMODE, P/R=0, V=1, T=2	0: MP 1: MV 2: MT
6	Set	Target Position	DINT	Position value to set		absolute: P=nnn incremental: D=nnn
7	Set	Target Velocity	DINT	Velocity value to set	RV	V=nnn
8	Set	Acceleration	DINT	Acceleration rate	RA	A=nnn
10	Set	Incremental Position Flag	BOOL	0=absolute, 1=incremental		
11	Set	Load Data/ Start Profile/ Profile in Progress	BOOL	On set, loads data and starts the current profile. On get, reports Profile in Progress	RW status bit 0	1: G 0: N/A
13	Set	Actual Position	DINT	Actual absolute position. Set to redefine actual position.	RP (or RPW)	O=nnn
14	Get	Actual Velocity	DINT	Reports actual velocity	RV (only valid in Torque Mode)	N/A
15	Get	Commanded Position	DINT	The instantaneous calculated position	RP (or RPW) + RPE	N/A
16	Get	Commanded Velocity	DINT	The instantaneous calculated velocity	RV	N/A
17	Set	Enable	BOOL	0=disable 1=enable		1: G or MP D=0 G, allow G or MT 0: OFF
20	Set	Smooth Stop	BOOL	Smooth Stop motor	0	1: X
21	Set	Hard Stop	BOOL	Hard Stop motor	0	1: S
23	Set	Direction	BOOL	Instantaneous Direction 0=reverse, 1=forward	Position Mode (direction of move) Velocity Mode (sign of Velocity). Torque Mode, sign of Torque	V=+/-nnn G T=+/-nnn
25	Set	Torque	DINT	Output Torque	RT	T=nnn or T=-nnn
29	Get	Wrap Around	BOOL	Position Wrap Around Indicator Flag	RW status bit 4	N/A
30	Set	Kp	INT	Proportional Gain	RKP	KP=nnn F
31	Set	Ki	INT	Integral Gain	RKI	KI=nnn F
32	Set	Kd	INT	Derivative Gain	RKD	KD=nnn F
33	Set	MaxKi	INT	Integration Limit	RKL	KL=nnn F
35	Set	Velocity Feed Forward	INT	Velocity feed forward gain value	RKV	KV=nnn F
37	Get	Sample Rate	INT	Update sample rate in micro-seconds	RSP	N/A
40	Get	Feedback Resolution	DINT	Number of actual position feedback counts per revolution	EPTR=32000 VLD(a,1) Ra	N/A

Attribute ID	Access Rule	Name	DeviceNet Data Type	Description of Attribute	Get Action	Set Action
41	Get	Motor Resolution	DINT	Number of motor steps per revolution	EPTR=32000 VLD(a,1) Ra	N/A
45	Set	Max Dynamic Following Error	DINT	Maximum allowable following error when the motor is in motion	RE	E=nnn
47	Set	Following Error Fault	BOOL	Following error occurrence flag	RW status bit 5	

Table 5 (Continued): Position Controller, Class 37 decimal Instance 1

Attribute ID	Access Rule	Name	DeviceNet Data Type	Description of Attribute	Get Action	Set Action
48	Get	Actual Following Error	DINT	Actual Following error	RPE	N/A
49	Set	Hard Limit Action	USINT	Hard Limit Action code 0=Servo off 2=Smooth Stop		0: F=0 2: F=1
50	Get	Forward Limit	BOOL	Forward Limit stop active	RW status bit 1	N/A
51	Get	Reverse Limit	BOOL	Reverse Limit stop active	RW status bit 2	N/A
58	Get	Load Data Complete	BOOL	valid data for a valid I/O command message type has been loaded into the position controller		N/A
100	Set	Current Limit	DINT	Current limit of motor 0 to 1023	RAMPS	AMPS=nnn

Table 6: SmartMotor I/O, Class 112 decimal, Instance 0 (Class 0)

Attribute ID	Access Rule	Name	DeviceNet Data Type	Description of Attribute	Get Action	Set Action
1	Get	Discrete Data	Array of byte	Array of Discrete Data bits for all defined instances. Bit 0 = Instance 1 Discrete Data		N/A

Table 7: SmartMotor I/O, Class 112 decimal, Instance 1 (Pin A)

Attribute ID	Access Rule	Name	DeviceNet Data Type	Description of Attribute	Get Action	Set Action
1	Set	Function	USINT	0=output, 1=input		0: UAO 1: UAI
2	Set	Output Latch	BOOL	0=low, 1=high		0: UA=0 1: UA=1
3	Get	Discrete Data	BOOL	0=low, 1=high	RUA	N/A
4	Get	Analog Data	INT	0 to 1023	RUAA	N/A

Table 8: SmartMotor I/O, Class 112 decimal, Instance 2 (Pin B)

Attribute ID	Access Rule	Name	DeviceNet Data Type	Description of Attribute	Get Action	Set Action
1	Set	Function	USINT	0=output, 1=input		0: UBO 1: UBI
2	Set	Output Latch	BOOL	0=low, 1=high		0: UB=0 1: UB=1
3	Get	Discrete Data	BOOL	0=low, 1=high	RUB	N/A
4	Get	Analog Data	INT	0 to 1023	RUBA	N/A

Table 9: SmartMotor I/O, Class 112 decimal, Instance 3 (Pin C)

Attribute ID	Access Rule	Name	DeviceNet Data Type	Description of Attribute	Get Action	Set Action
1	Set	Function	USINT	0=output, 1=input, 3=special (positive limit action)		0: UCO 1: UCI 2: UCP
2	Set	Output Latch	BOOL	0=low, 1=high		0: UC=0 1: UC=1
3	Get	Discrete Data	BOOL	0=low, 1=high	RUC	N/A
4	Get	Analog Data	INT	0 to 1023	RUCA	N/A

Table 10: SmartMotor I/O, Class 112 decimal, Instance 4 (Pin D)

Attribute ID	Access Rule	Name	DeviceNet Data Type	Description of Attribute	Get Action	Set Action
1	Set	Function	USINT	0=output, 1=input, 3=special (negative limit action)		0: UDO 1: UDI 2: UDM
2	Set	Output Latch	BOOL	0=low, 1=high		0: UD=0 1: UD=1
3	Get	Discrete Data	BOOL	0=low, 1=high	RUD	N/A
4	Get	Analog Data	INT	0 to 1023	RU DA	N/A

Table 11: SmartMotor I/O, Class 112 decimal, Instance 5 (Pin E)

Attribute ID	Access Rule	Name	DeviceNet Data Type	Description of Attribute	Get Action	Set Action
1	Set	Function	USINT	0=output, 1=input		0: UEO 1: UEI
2	Set	Output Latch	BOOL	0=low, 1=high		0: UE=0 1: UE=1
3	Get	Discrete Data	BOOL	0=low, 1=high	RUE	N/A
4	Get	Analog Data	INT	0 to 1023	RUEA	N/A

Table 12: SmartMotor I/O, Class 112 decimal, Instance 6 (Pin F)

Attribute ID	Access Rule	Name	DeviceNet Data Type	Description of Attribute	Get Action	Set Action
1	Set	Function	USINT	0=output, 1=input		0: UFO 1: UFI
2	Set	Output Latch	BOOL	0=low, 1=high		0: UF=0 1: UF=1
3	Get	Discrete Data	BOOL	0=low, 1=high	RUF	N/A
4	Get	Analog Data	INT	0 to 1023	RUFA	N/A

Table 13: SmartMotor I/O, Class 112 decimal, Instance 7 (Pin G)

Attribute ID	Access Rule	Name	DeviceNet Data Type	Description of Attribute	Get Action	Set Action
1	Set	Function	USINT	0=output, 1=input, 3=special (GO)		0: UGO 1: UGI 2: UG
2	Set	Output Latch	BOOL	0=low, 1=high		0: UG=0 1: UG=1
3	Get	Discrete Data	BOOL	0=low, 1=high	RUG	N/A
4	Get	Analog Data	INT	0 to 1023	RUGA	N/A

Sample Producing Motion via Explicit Messaging**Table 14: Velocity Mode – The following sequence of commands will result is velocity mode motion**

Action	Class	Instance	Attribute	Name	Decimal Value	Hex Value (Lo byte→Hi byte)	Description
SET	37	01	03	Mode	1	0x01	velocity mode
SET	37	01	08	Acceleration	25263	0xAF620000	A=100 (scaled 25263)
SET	37	01	07	Velocity	31044	0x44790000	V=500000 (scaled 31044)
SET	37	01	11	Load Data	1	0x01	load data and execute

Table 15: Torque Mode – The following sequence of commands will result is torque mode motion

Action	Class	Instance	Attribute	Name	Decimal Value	Hex Value (Lo byte→Hi byte)	Description
SET	37	01	03	Mode	2	0x02	torque mode
SET	37	01	25	Torque	-255	0x00FFFFFF	torque=> ¼ Full CCW
SET	37	01	17	Enable	1	0x01	motor power enabled
SET	37	01	11	Load Data	1	0x01	load data and execute
SET	37	01	11	Load Data	0	0x00	load data clear
GET	37	01	23	Instantaneous Direction	00	0x00	reverse direction
SET	37	01	25	Torque	512	0x00020000	torque=> ½ Full CW
GET	37	01	23	Instantaneous Direction	1	0x01	forward direction

Vendor Specific Descriptions

This is a list of Animatics-specific descriptions of attributes. Descriptions with more elaboration than the Object Class, Instance, and Attribute List Appendix are included for those attributes whose behavior has a vendor-specific character.

Table 16: POSITION CONTROLLER CLASS 37 DECIMAL

Attribute	Name	Description
14	Actual Velocity	Actual velocity, subject to the granularity one encoder count per Sample, is reported in Torque mode. In position and velocity modes, the calculated velocity, which is being enforced by the PID feedback, is reported. This value is more precise over an aggregate of servo-Samples.
16	Commanded Velocity	This attribute has no meaning in torque mode.
17	Enable	Value = 0 → 1: permits motion initiation, initiates motion, or initiates servo in place Value = 1 → 0: motor is turned OFF, initiation of motion or servo is not permitted
23	Instantaneous Direction	Value = 0: reverse or negative or CCW viewed facing shaft Value = 1: forward or positive or CW viewed facing shaft Value change: reverses sign on velocity in velocity mode
25	Output Torque	Range is -1023 to 1023.
49	Hard Limit Action	Values supported are: 0 motor turned OFF on limit crash 2 motor decelerates at value of acceleration on limit crash
100	Current Limit	Range is 0 to 1023

Definition of a Profile Move

A profile move is a move that uses Acceleration and Target Velocity to run at a Target Velocity or to a Target Position. In addition, the position controller device can output a Torque command. Whether or not the position controller device runs at a Target Velocity, to a Target Position or outputs a Torque command depends on the Operation Mode (Position Controller Object Attribute 3), to which the position controller device is set. The position controller device is set to Position, Velocity or Torque mode using Position Controller Attribute 3 (which can be set via Command Message type 0x1B).

Starting a Profile Move

The Position Controller Profile is mode-sensitive. The Position Controller Object Attribute 3 sets the mode of the controller to the following:

- 0 = Position (default)
- 1 = Velocity
- 2 = Torque

A profile move starts when the command message type for the specified mode is loaded and the Load Data/ Start Profile bit transitions from 0->1. The table below shows the command message type which starts a profile move for each mode.

Table 17: Start Motion Command Message Type

Mode (Attribute 3)	Command Message Type which starts motion
0 = Position	01 = Position
1 = Velocity	02 = Velocity
2 = Torque	05 = Torque

Table 18: Command Message Format

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Enable	n/a	Hard Stop	Smooth Stop	Direction (V. Mode)	Incremental	n/a	Load Data/Start Profile
1	Command Data 1							
2	Command Axis Number			Command Message Type				
3	Command Data 2							
4	Command Data 3							
5	Command Data 4							
6	Command Data 5							
7	Command Data 6							

Table 19 : Response Message Format

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Enable	Index Level	n/a	Curent Direction	General Fault	n/a	n/a	Profile in Progress
1	Response Data 1							
2	Load Complete	n/a	FE Fault	n/a	n/a	Rev Limit	Fwd Limit	n/a
3	Response Axis Number			Response Message Type				
4	Response Data 2							
5	Response Data 3							
6	Response Data 4							
7	Response Data 5							

TROUBLESHOOTING

What could be wrong?

If the DeviceNet network fails to communicate with the motor, please check the following:

- Motor is powered. The motor servo led should be green or red, not off.
- Motor has a user program beginning with the required OCHN command.
- Motor's network baud rate, held in non-volatile memory, matches the network baud rate.
- Motor's MACID, held in non-volatile memory, is set correctly.
- DeviceNet network trunk lines, drop lines, and terminators are within specifications.