

# **DeviceNet™** communication profile for **SERVOSTAR® 600**



**Previous editions**

<b>Edition</b>	<b>Comments</b>
12 / 02	First edition

SERVOSTAR is a registered trademark of Kollmorgen Corporation

**Technical changes to improve the performance of the equipment may be made without prior notice !**

Printed in the Federal Republic of Germany

All rights reserved. No part of this work may be reproduced in any form (by printing, photocopying microfilm or any other method) or processed, copied or distributed by electronic means, without the written permission of Kollmorgen Seidel.

	Page
<b>Contents</b> .....	3
<b>Abbreviations / Symbols</b> .....	7
<b>I General</b>	
I.1 About this manual .....	7
I.2 Application of this manual .....	7
I.3 Permitted use ("Use as directed") of the DeviceNet interface .....	7
I.4 Basic features implemented through DeviceNet .....	8
I.5 System requirements .....	8
I.6 Transmission rate and procedure .....	8
I.7 Bus cable .....	9
I.8 Response to BUSOFF communication faults .....	10
I.9 Combined Module/Network Status LED .....	11
<b>II Installation / Setup</b>	
II.1 Assembly, Installation .....	13
II.1.1 Connection methods .....	13
II.1.2 Setting the station address .....	13
II.1.3 Setting the transmission rate .....	14
II.2 Setup .....	14
<b>III Position controller object model</b>	
III.1 Functionality Chart .....	15
III.1.1 Object Model .....	15
III.1.1.1 Object: Identity .....	15
III.1.1.2 Object: Message Router .....	15
III.1.1.3 Object: DeviceNet .....	15
III.1.1.4 Object: Assembly .....	15
III.1.1.5 Object: Assembly .....	16
III.1.1.6 Object: Explicit Connection .....	16
III.1.1.7 Object: I/O Connection .....	16
III.1.1.8 Object: Discrete Input Point .....	16
III.1.1.9 Object: Discrete Output Point .....	16
III.1.1.10 Object: Analog Input Point .....	16
III.1.1.11 Object: Analog Output Point .....	16
III.1.1.12 Object: Parameter .....	16
III.1.1.13 Object: Position Controller Supervisor .....	17
III.1.1.14 Object: Position Controller .....	17
III.1.1.15 Object: Block Sequencer .....	17
III.1.1.16 Object: Command Block .....	17
III.1.2 Firmware Version .....	17
III.1.3 Supported Services .....	17
III.1.4 Data Types .....	17
III.2 Position Controller Supervisor Object Class (ID=36) .....	18
III.2.1 Error Codes .....	18
III.2.1.1 Object State Conflicts – 0x0C .....	18
III.2.2 Supervisor Attributes .....	18
III.2.2.1 Attribute ID 3: Axis Instance Number .....	18
III.2.2.2 Attribute ID 5: General Fault .....	19
III.2.2.3 Attribute ID 6: Input Command Assembly Type .....	19
III.2.2.4 Attribute ID 7: Response Assembly Type .....	19
III.2.2.5 Attribute ID 14: Index Active Level .....	20
III.2.2.6 Attribute ID 21: Registration Arm .....	20
III.2.2.7 Attribute ID 22: Registration Input Level .....	20

III.3	Position Controller Object Class (ID=37)	20
III.3.1	Error Codes	20
III.3.2	Position controller attributes	21
III.3.2.1	Object State Conflicts – 0x0C	21
III.3.2.2	Attribute 1: Number of Attributes	21
III.3.2.3	Attribute 2: Attribute List	21
III.3.2.4	Attribute 3: Mode	21
III.3.2.5	Attribute 6: Target Position	22
III.3.2.6	Attribute 7: Target Velocity	22
III.3.2.7	Attribute 8: Acceleration	22
III.3.2.8	Attribute 9: Deceleration	22
III.3.2.9	Attribute 10: Incremental Position Flag	23
III.3.2.10	Attribute 11: Trajectory Start/Complete	23
III.3.2.11	Attribute 12: On Target Position	23
III.3.2.12	Attribute 13: Actual Position	24
III.3.2.13	Attribute 14: Actual Velocity	24
III.3.2.14	Attribute 17: Enable	24
III.3.2.15	Attribute 20: Smooth Stop	25
III.3.2.16	Attribute 21: Hard Stop	25
III.3.2.17	Attribute 22: Jog Velocity	25
III.3.2.18	Attribute 23: Direction	26
III.3.2.19	Attribute 24: Reference direction	26
III.3.2.20	Attribute 25: Torque	26
III.3.2.21	Attribute 100: Clear Faults	26
III.3.2.22	Attribute 101: Save Parameters	27
III.3.2.23	Attribute 102: Drive Status	27
III.3.2.24	Attribute 103: Trajectory Status	27
III.4	Block Sequencer Object Class (ID=38)	28
III.4.1	Attribute 1: Block	28
III.4.2	Attribute 2: Block Execute	28
III.4.3	Attribute 3: Current Block	29
III.4.4	Attribute 4: Block Fault	29
III.4.5	Attribute 5: Block Fault Code	29
III.4.6	Attribute 6: Counter	29
III.5	Command Block Object Class (ID=39)	30
III.5.1	Command 01 – Modify Attribute	30
III.5.1.1	Attribute 1: Block Command	30
III.5.1.2	Attribute 2: Block Link #	30
III.5.1.3	Attribute 3: Target Class	30
III.5.1.4	Attribute 4: Target Instance	31
III.5.1.5	Attribute 5: Attribute #	31
III.5.1.6	Attribute 6: Attribute Data	31
III.5.2	Command 02– Wait Until Equals	32
III.5.2.1	Attribute 1: Block Command	32
III.5.2.2	Attribute 2: Block Link #	32
III.5.2.3	Attribute 3: Target Class	32
III.5.2.4	Attribute 4: Target Instance	33
III.5.2.5	Attribute 5: Attribute #	33
III.5.2.6	Attribute 6: Timeout	33
III.5.2.7	Attribute 7: Compare Data	33
III.5.3	Command 03– Conditional Link Greater Than Command	34
III.5.3.1	Attribute 1: Block Command	34
III.5.3.2	Attribute 2: Block Link #	34
III.5.3.3	Attribute 3: Target Class	34
III.5.3.4	Attribute 4: Target Instance	35
III.5.3.5	Attribute 5: Attribute #	35
III.5.3.6	Attribute 6: Compare Link #	35
III.5.3.7	Attribute 7: Compare Data	35
III.5.4	Command 04– Conditional Link Less Than Command	36
III.5.4.1	Attribute 1: Block Command	36
III.5.4.2	Attribute 2: Block Link #	36
III.5.4.3	Attribute 3: Target Class	36
III.5.4.4	Attribute 4: Target Instance	37
III.5.4.5	Attribute 5: Attribute #	37
III.5.4.6	Attribute 6: Compare Link #	37
III.5.4.7	Attribute 7: Compare Data	37

	Page	
III.5.5	Command 05– Decrement Counter . . . . .	38
III.5.5.1	Attribute 1: Block Command . . . . .	38
III.5.5.2	Attribute 2: Block Link # . . . . .	38
III.5.6	Command 06– Delay Command . . . . .	38
III.5.6.1	Attribute 1: Block Command . . . . .	38
III.5.6.2	Attribute 2: Block Link # . . . . .	39
III.5.6.3	Attribute 3: Delay . . . . .	39
III.5.7	Command 08– Trajectory Command and Wait . . . . .	39
III.5.7.1	Attribute 1: Block Command . . . . .	39
III.5.7.2	Attribute 2: Block Link # . . . . .	40
III.5.7.3	Attribute 3: Target Position . . . . .	40
III.5.7.4	Attribute 4: Target Velocity . . . . .	40
III.5.7.5	Attribute 5: Incremental . . . . .	40
III.6	Polled I/O Command Assemblies . . . . .	41
III.6.1	Running a Stored Sequence Through DeviceNet . . . . .	41
III.6.2	Stopping a Program Through DeviceNet . . . . .	41
III.6.3	Command Assembly 1 – Target Position . . . . .	42
III.6.4	Command Assembly 2 – Target Velocity . . . . .	44
III.6.5	Command Assembly 3 – Acceleration . . . . .	44
III.6.6	Command Assembly 4 – Deceleration . . . . .	45
III.6.7	Command Assembly 5 – Torque . . . . .	46
III.7	I/O Response Assemblies . . . . .	46
III.7.1	Response Assembly 1 – Actual Position . . . . .	46
III.7.2	Response Assembly 2 – Commanded Position . . . . .	48
III.7.3	Response Assembly 3 – Actual Velocity . . . . .	49
III.7.4	Response Assembly 5 – Torque . . . . .	49
III.7.5	Response Assembly 20 – Command/Response Error . . . . .	50
III.8	Identity Object Class . . . . .	51
III.9	Message Router Object Class . . . . .	52
III.10	DeviceNet Object Class . . . . .	52
III.11	Connection Object Class (Explicit) . . . . .	53
III.12	Connection Object Class (Polled I/O) . . . . .	54
III.13	Discrete Input Point Object . . . . .	55
III.14	Discrete Output Point Object . . . . .	56
III.15	Analog Input Point Object . . . . .	57
III.16	Analog Output Point Object . . . . .	58
III.17	Parameter Object . . . . .	59
<b>IV</b>	<b>Appendix</b>	
IV.1	Examples . . . . .	61
IV.1.1	Simple Motion Sequence . . . . .	61
IV.2	Allen Bradley SLC5/0x . . . . .	61
IV.2.1	Polled I/O . . . . .	61
IV.2.2	Polled Amplifier Input (PLC Output) . . . . .	62
IV.2.3	Command Assembly Example . . . . .	63
IV.2.4	Polled Amplifier Output (PLC Input) . . . . .	64
IV.2.5	Response Assembly Example . . . . .	65
IV.2.6	Explicit Messaging . . . . .	66
IV.2.6.1	Explicit Message Sequence of Events . . . . .	67
IV.2.6.1.1	Example . . . . .	67
IV.2.7	Example Program . . . . .	68
IV.2.7.1	Running the Test Program . . . . .	68
IV.2.7.2	Subroutines . . . . .	69
IV.2.7.3	Data Mapping . . . . .	69
IV.2.7.4	Output File Mapping . . . . .	69
IV.2.7.5	Input File Mapping . . . . .	70
IV.3	Baud Rate Switch Settings . . . . .	70
IV.4	MAC ID Switch Configuration . . . . .	70
IV.5	Default Input/Output Configuration . . . . .	70
IV.6	Index . . . . .	71


### Abbreviations used in this manual

The abbreviations used in this manual are explained in the table below.

Abbrev.	Meaning
ACC	Acceleration
BOI	Bus Off interrupt
CAN	Controller area network
CCW	Counter clockwise
COS	Change of state
CW	Clockwise
EMC	Electromagnetic compatibility
ISO	International Standardization Organization

Abbrev.	Meaning
LED	Light-emitting diode
LSD	Least significant digit
MAC ID	Media access control identifier
M/S	Master/slave
MSD	Most significant digit
N/A	Not applicable
ODVA	Open DeviceNet Vendor Association
PC	Personal Computer with 80x86 processor

### Symbols used in this manual

	danger to personnel from electricity and its effects		general warning general instructions mechanical hazard
	see ... (cross-reference)		special emphasis

## I General

### I.1 About this manual

This manual describes the setup, range of functions and software protocol of the SERVOSTAR® 600 servo amplifiers with the *DeviceNet*™ communication profile. It forms part of the complete documentation for the SERVOSTAR 600 family of servo amplifiers.

The installation and setup of the servo amplifier, as well as all standard functions, are described in the corresponding installation manuals.

#### Other parts of the complete documentation for the digital servo amplifier series:

Title	Publisher
Online-Help for setup software DRIVE.EXE	Seidel
Assembly/Installation/Setup Instructions SERVOSTAR 600	Seidel

#### Additional documentation:

Title	Publisher
DeviceNet Specification, Volumes I, II, Release 2.0	ODVA
CAN Specification Version 2.0	CiA e.V.
ISO 11898 ... Controller Area Network (CAN) for high-speed communication	ISO

This manual is intended for the following qualified personnel:



**Wiring:** *Professionally qualified electrical technicians*  
**Programming:** *Software developers, project-planners*

Training and familiarization courses are available on request.

### I.2 Application of this manual

Specific examples for individual chapters can be found in the appendix of this manual.

### I.3 Permitted use ("Use as directed") of the DeviceNet interface

Please observe the chapter "Permitted use" in the setup manual for the servo amplifier.

The interface is a component part of the SERVOSTAR 600 series of digital servo amplifiers. The DeviceNet interface serves only for the connection of the servo amplifier to a master via the DeviceNet bus.

The servo amplifiers are components that are built into electrical apparatus or machinery, and can only be setup and operated as integral components of such apparatus or machinery.



*We can only guarantee the conformity of the servo amplifier with the following standards for industrial areas when the components that we specify are used, and the installation regulations are followed:*

<i>EC EMC Directive</i>	<i>89/336/EEC</i>
<i>EC Low-Voltage Directive</i>	<i>73/23/EEC</i>

## I.4 Basic features implemented through DeviceNet

When working with the position controller that is integrated in SERVOSTAR 600 digital servo amplifiers, the following functions are available:

### **Setup and general functions:**

- homing, set reference point
- jogging, with a variable speed
- provision of a digital setpoint for speed and torque control

### **Positioning functions:**

- execution of a motion task from the motion block memory of the servo amplifier
- execution of a direct motion task
- absolute trajectory

### **Data transfer functions:**

- transmit a motion task to the motion block memory of the servo amplifier  
A motion task consists of the following elements:
  - » position setpoint (absolute task) or path setpoint (relative task)
  - » speed setpoint
  - » acceleration time, braking time, rate-of-change/jolt limiting (in preparation)
  - » type of motion task (absolute/relative)
  - » number of a following task (with or without pause)
- Transmit a non-motion task to the motion block memory of the servo amplifier

In addition to motion tasks, the following task types can be modified through DeviceNet:

- modify attribute
- wait until parameter = value
- branch if greater than/less than
- decrement counter
- delay
- read a motion task from the motion block memory of the servo amplifier
- read actual values
- read the error register
- read the status register
- read/write configuration and control parameters
- read actual values from analog and digital inputs
- write control values to analog and digital outputs

## I.5 System requirements

- servo amplifier SERVOSTAR 600
- DeviceNet option card for the SERVOSTAR 600
- master station with a DeviceNet interface (e.g. PC with DeviceNet card)

## I.6 Transmission rate and procedure

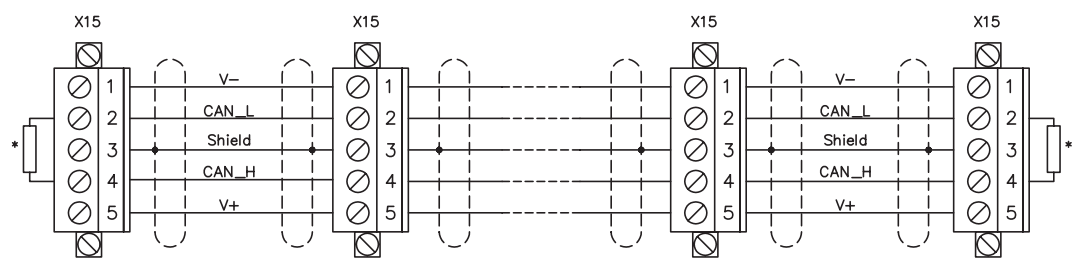
- bus connection and bus medium: CAN-standard ISO 11898 (CAN high-speed)
- transmission rate: max. 500 kbit/s  
possible settings for the servo amplifier: 125, 250, 500 kbit/s

## I.7 Bus cable

In accordance with ISO 11898 you should use a bus cable with a characteristic impedance of  $120\Omega$ . The usable cable length for reliable communication is reduced as the transmission rate is increased. The following values that we have measured can be used as a guide. They should not, however, be interpreted as limiting values:

General characteristic	Specification
Bit rates	125K, 250K, 500K
Distance with thick trunk	500m at 125Kbaud 250m at 250Kbaud 100m at 500Kbaud
Number of nodes	64
Signaling	CAN
Modulation	baseband
Media coupling	DC coupled differential Tx/Rx
Isolation	500V (optional optocouplers on node side of transceiver)
Differential input impedance typical (recessive state)	Shunt C = 5pF Shunt R = $25K\Omega$ (power on)
Differential input impedance min. (recessive state)	Shunt C = 24pF plus 12pF/ft of permanently attached dropline Shunt R = $20K\Omega$
Absolute max. voltage range	-25V to +18V (CAN_H, CAN_L)*

\* Voltages at CAN\_H and CAN\_L are referenced to the transceiver IC ground pin. This Voltage will be higher than the V- terminal by an amount equal to the voltage drop on the Schottky diode. This voltage should be 0.6V maximum.



\* according to line impedance about  $120\Omega$

### Grounding:

To prevent ground loops, the DeviceNet network should be earth grounded in only one location. The physical layer circuitry in all devices is referenced to the V- bus signal. Connection to earth ground is provided at the bus power supply; no current flow between V- and earth ground may occur via any device other than a power supply.

### Bus Topology:

The DeviceNet media has a linear bus topology. Terminating resistors are required on each end of the trunk line. Drop lines as long as 6 m (20 feet) each are permitted, allowing one or more nodes to be attached.

### Terminating Resistors:

DeviceNet requires a terminating resistor to be installed at each end of the trunk. The resistor requirements are:

- $121\Omega$
- 1% Metal Film
- 1/4 W

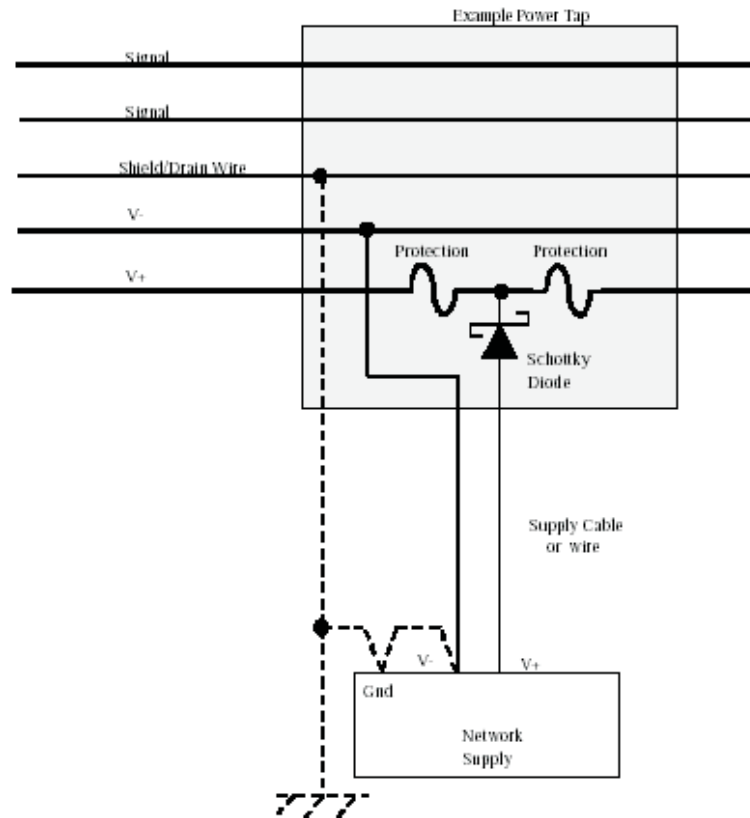


**Important:** Terminating resistors should not be installed at the end of a drop line, only at the two ends of the trunk line.

**Network Power:**

Power taps for DeviceNet should have the following characteristics:

- Specified ratings for power supply and network currents (24V)
- Fuses or circuit breakers to limit current on the bus if current limiting in the power supply is insufficient.
- 10 ft. maximum cable length from power supply to power tap



## I.8 Response to BUSOFF communication faults

The communication fault BUSOFF is directly monitored and signaled by Level 2 (CAN controller). This message may have various causes.

A few examples:

- telegrams are transmitted, although there is no other CAN node connected
- CAN nodes have different transmission rates
- the bus cable is faulty
- faulty cable termination causes reflections on the cable.

The DeviceNet Object (class 0x03, attributes 3 and 4) determines the response to a BUSOFF condition.

## I.9 Combined Module/Network Status LED

For this state:	LED is	To indicate:
Not powered / not online	off	Device is not online. The device has not completed the Dup_MAC_ID test yet. The device may not be powered.
Device operational AND online, connected	green	The device is operating in a normal condition and the device is online with connections in the established state. - The device is allocated to a Master
Device operational AND online, not connected or Device online AND device needs commissioning	flashing green	The device is operating in a normal condition and the device is online with no connections in the established state. The device has passed the Dup_MAC_ID test, is online, but has no established connections to other nodes. This device is not allocated to a master. Configuration missing, incomplete or incorrect.
Minor fault and/or connection time-out	flashing red	Recoverable fault and/or one or more I/O Connections are in the Timed-Out state.
Critical fault or critical link failure	red	- The device has an unrecoverable fault; may need replacing. - Failed communication device. The device has detected an Error that has rendered it incapable of communicating on the network (eg. Duplicate MAC ID, or Bus-off).

This page has been deliberately left blank.

## II Installation / Setup

### II.1 Assembly, Installation



*Only install and wire up the equipment in a de-energized condition, i.e. neither the mains/line supply voltage nor the 24V auxiliary voltage nor the operating voltage of any other connected equipment may be switched on. Take care that the switchgear cabinet is safely disconnected (lockout, warning signs etc.). The individual voltages are switched on for the first time during setup. Never disconnect the electrical connections to the servo amplifier while it is live. This could destroy the electronics.*

*Residual charges in the capacitors can still have dangerous levels several minutes after switching off the supply power. Measure the voltage in the DC-link circuit and wait until the voltage has fallen below 40V.*

*Even when the motor is not running, power and control cables can still be live.*

- Assemble the servo amplifier as described in the installation instructions. Observe all safety instructions in the installation instructions that belong to the servo amplifier. Follow all the notes on mounting position, ambient conditions, wiring, and fusing.
- The connections for the motor, controls and power, as well as advice on system layout for EMC conformance, can be found in the installation instructions for the servo amplifier.

#### II.1.1 Connection methods

**Supply power, motor, analog setpoints, digital control signals, DeviceNet™ connection**  
see installation instructions for SERVOSTAR 600.

#### II.1.2 Setting the station address

The station address (instrument address on the DeviceNet-Bus) for the servo amplifier can be set in three different ways:

- Set the rotary switches on the front panel of the option card to a value between 0 and 63. Each switch represents one decimal digit. To set the drive to address 10, set the MSD (most significant digit) switch to 1 and the LSD (least significant digit) switch to 0.
- Set the rotary switches on the front panel of the option card to a value greater than 63. The station address can now be set using the ASCII commands DNMACID x, SAVE, COLDSTART where x is the station address.
- Set the rotary switches on the front panel of the option card to a value greater than 63. The station address can now be set through the DeviceNet object (class 0x03, attribute 1). This is typically done through a DeviceNet commissioning tool. The parameter must be saved to nonvolatile memory (class 0x25, attribute 0x65) and the drive must be restarted after modifying the address.

### II.1.3 Setting the transmission rate

The DeviceNet transmission rate can be set in three different ways:

- Set the rotary baudrate switch on the front panel of the option card to a value between 0 and 2. 0=125kbit/s, 1=250kbit/s, 2=500kbit/s.
- Set the baudrate switch on the front panel of the option card to a value greater than 2. The baud rate can now be set using the terminal commands DNBAUD x, SAVE, COLDSTART where x is 125, 250 or 500.
- Set the baudrate switch on the front panel of the option card to a value greater than 2. The baud rate can now be set through the DeviceNet object (class 0x03, attribute 2) to a value between 0 and 2. This is typically done through a DeviceNet commissioning tool. The parameter must be saved to nonvolatile memory (class 0x25, attribute 0x65) and the drive must be restarted after modifying the baud rate.

Possible transmission rates are: 125, 250, 500 kbit/s.

## II.2 Setup



*Only professional personnel with extensive knowledge of control and drive technology are allowed to setup the servo amplifier.*

Check assembly / installation	Check that all the safety instructions in the installation instructions for the servo amplifier and this manual have been observed and implemented. Check the setting for the station address.
Connect PC, start setup software	Use the setup software DRIVE.EXE to set the parameters for the servo amplifier.
Setup the basic functions	Start up the basic functions of the servo amplifier and optimize the current and speed controllers. This section of the setup is described in detail in the setup software manual.
Save parameters	When the parameters have been optimized, save them in the servo amplifier.
Start up bus communi- cation	Requirement: the software protocol described in Chapter IV must be implemented in the master. Adjust the station address and the transmission rate of the SERVOSTAR 600 to match the master.
Test the communication	Connect to the SERVOSTAR 600 with a master device. Try viewing/modifying a parameter with explicit messaging (eg. Position Controller Object class 0x25, instance 0x01, Acceleration attribute 0x08 → terminal parameter ACC).
<p><b>Caution !</b> <i>Make sure that any unintended movement of the drive cannot endanger machinery or personnel.</i></p>	
Setup the position controller	Setup the position controller, as described in the setup software manual.

### III Position controller object model

#### III.1 Functionality Chart

DeviceNet™	ODVA Requirements
Device Type	Position Controller
Explicit Peer-to-Peer Messaging	N
I/O Peer-to-Peer Messaging	N
Baud Rates:	125, 250 and 500 kB
Polled Response Time	<10ms
Explicit Response Time	<50ms (except parameter object, <500ms)
Master/Scanner	N
Configuration Consistency Value	N
Faulted Node Recovery	Y
I/O Slave Messaging	
Bit Strobe	N
Polling	Y
Cyclic	N
Change-of-State (COS)	N

#### III.1.1 Object Model

The following DeviceNet objects are supported in the drive:

##### III.1.1.1 Object: Identity

<b>Class Code</b>	0x01
<b>Instance #</b>	1
<b>Description</b>	This object provides identification of any general information about the device. The Identity Object is present in all DeviceNet products.

##### III.1.1.2 Object: Message Router

<b>Class Code</b>	0x02
<b>Instance #</b>	1
<b>Description</b>	This object provides a messaging connection point through which a client may address a service to any object class or instance residing in the physical device.

##### III.1.1.3 Object: DeviceNet

<b>Class Code</b>	0x03
<b>Instance #</b>	1
<b>Description</b>	This object provides the configuration and status of a DeviceNet port. Each DeviceNet product supports only one DeviceNet object per physical connection to the DeviceNet communication link.

##### III.1.1.4 Object: Assembly

<b>Class Code</b>	0x04
<b>Instance #</b>	1
<b>Description</b>	This object binds attributes of multiple objects, which allows data to or from each object to be sent or received over a single connection. Assembly objects can be used to bind input or output data. An input produces data on the network and an output consumes data from the network.

**III.1.1.5 Object: Assembly**

<b>Class Code</b>	0x04
<b>Instance #</b>	2
<b>Description</b>	This object stores I/O output message data.

**III.1.1.6 Object: Explicit Connection**

<b>Class Code</b>	0x05
<b>Instance #</b>	1
<b>Description</b>	This object manages the explicit messages.

**III.1.1.7 Object: I/O Connection**

<b>Class Code</b>	0x05
<b>Instance #</b>	2
<b>Description</b>	This object manages the I/O messages.

**III.1.1.8 Object: Discrete Input Point**

<b>Class Code</b>	0x08
<b>Instance #</b>	1-4
<b>Description</b>	The discrete input point objects give access to the drive's four digital inputs.

**III.1.1.9 Object: Discrete Output Point**

<b>Class Code</b>	0x09
<b>Instance #</b>	1-2
<b>Description</b>	The discrete output point objects give access to the drive's two digital outputs.

**III.1.1.10 Object: Analog Input Point**

<b>Class Code</b>	0x0A
<b>Instance #</b>	1-2
<b>Description</b>	The analog input point objects give access to the drive's two analog inputs.

**III.1.1.11 Object: Analog Output Point**

<b>Class Code</b>	0x0B
<b>Instance #</b>	1-2
<b>Description</b>	The analog output point objects give access to the drive's two analog outputs. Object: Position Controller Supervisor

**III.1.1.12 Object: Parameter**

<b>Class Code</b>	0x0F
<b>Instance #</b>	1-255
<b>Description</b>	The parameter object gives direct access to drive configuration parameters.

### III.1.1.13 Object: Position Controller Supervisor

Class Code	0x24
Instance #	1
Description	The position controller supervisor handles errors for the position controller as well as home inputs.

### III.1.1.14 Object: Position Controller

Class Code	0x25
Instance #	1
Description	The position controller object performs the control output velocity profiling and handles input and output to and from the motor drive unit, limit switches, etc.

### III.1.1.15 Object: Block Sequencer

Class Code	0x26
Instance #	1
Description	This object handles the execution of command blocks or command block chains.

### III.1.1.16 Object: Command Block

Class Code	0x27
Instance #	1 to 255
Description	Each instance of the command block object defines a specific command. These blocks can be linked to other blocks to form a command block chain.

## III.1.2 Firmware Version

The codes and services described in this manual applies to the firmware version 5.42 or greater:

## III.1.3 Supported Services

The only services supported by the Kollmorgen DeviceNet Block Command Object, Block Sequence Object, Position Controllers, Position Controller Supervisor Object and Position Controller Object are:

Get\_Single\_Attribute (service code 14) (0x0E) and Set\_Single\_Attribute (service code 16) (0x10).

**Also note that the Axis Instance is always 1.**

For additional information, we recommend that you review this entire document.

## III.1.4 Data Types

The table below describes the data type, number of bits, minimum and maximum Range.

Data Type	Number of Bits	Minimum Value	Maximum Value
Boolean	1	0 (False)	1 (True)
Short Integer	8	-128	127
Unsigned Short Integer	8	0	255
Integer	16	-32768	32767
Unsigned Integer	16	0	65535
Double Integer	32	-2 <sup>31</sup>	2 <sup>31</sup> - 1
Unsigned Double Integer	32	0	2 <sup>32</sup> - 1

## III.2 Position Controller Supervisor Object Class (ID=36)

### III.2.1 Error Codes

The drive returns one of the following codes when an error is generated while communicating via Explicit Messaging.

Action	Error	Error Code
Set	Attribute Not Settable	0x0E
Set or Get	Attribute Not Supported	0x14
Set or Get	Service Not Supported	0x08
Set or Get	Class Not Supported	0x16
Set	Value is Out of Range	0x09

#### III.2.1.1 Object State Conflicts – 0x0C

Three conditions could cause the drives to return this error code. To proceed, check and clear the condition.

Condition	Solution
On hard or soft limit and then issuing a command to move in the direction of the limit	Move in opposite direction of the limit
Issuing a command not support in the current mode (i.e., trying to do registration in velocity mode)	Change the mode to fit the application or issue the proper command
Trying to enable a faulted drive	Correct the fault before enabling the drive.

### III.2.2 Supervisor Attributes

These are attributes supported by the unit in the Position Controller Supervisor Class.

#### III.2.2.1 Attribute ID 3: Axis Instance Number

**Access Rule** Get

**Data Type** Unsigned Short Integer

**Description** Returns the axis number which is the same as the instance for this object.

**Range** This is always 1.

**Default** 1

**Non-Volatile** N/A

**See Also** N/A

### III.2.2.2 Attribute ID 5: General Fault

**Access Rule** Get

**Data Type** Boolean

**Description** When active, this indicates that a drive-related failure has occurred, (Short Circuit, Over-Voltage, etc.). It is not related to the FAULT input. It is reset when the fault condition is removed.

**Range** 1 = Fault condition exists  
0 = No fault exists

**Default** None

**Non-Volatile** N/A

**See Also** Fault Status Bits

### III.2.2.3 Attribute ID 6: Input Command Assembly Type

**Access Rule** Get  
Set

**Data Type** Unsigned Short Integer

**Description** This attribute specifies which Input Command Assembly Type is used during polled I/O commands. See the Polled I/O Command Assemblies section for additional details.

**Range** Drive Dependent – Please see the Polled I/O Command Assemblies section for more information

**Default** 0x00

**Non-Volatile** N/A

**See Also** N/A

### III.2.2.4 Attribute ID 7: Response Assembly Type

**Access Rule** Get  
Set

**Data Type** Unsigned Short Integer

**Description** Sets the response message that is returned to the controlling device

**Range** Valid Message Type Codes are: 0x00, 0x01, 0x02, 0x03, and 0x1E

**Default** 0x00

**Non-Volatile** N/A

**See Also** N/A

### III.2.2.5 Attribute ID 14: Index Active Level

**Access Rule** Get/Set

**Data Type** Boolean

**Description** This attribute is used to set the active level of the indexing input.

**Range** 0 = Active Low  
1 = Active High

**Default** None

**Non-Volatile** N/A

**See Also** N/A

### III.2.2.6 Attribute ID 21: Registration Arm

**Access Rule** Get/Set

**Data Type** Boolean

**Description** Set the value to 1 to arm the registration input. The values reads 0 when triggered

**Range** 0 = registration triggered (Get)  
1 = registration armed (Get/Set)

**Default** None

**Non-Volatile** N/A

**See Also** N/A

### III.2.2.7 Attribute ID 22: Registration Input Level

**Access Rule** Get

**Data Type** Boolean

**Description** This attribute returns the actual value of the registration input.

**Range** 0 = Low  
1 = High

**Default** None

**Non-Volatile** N/A

**See Also** N/A

## III.3 Position Controller Object Class (ID=37)

### III.3.1 Error Codes

The drive returns one of the following error codes when an error is generated while communicating via Explicit Messaging.

Action	Error	Error Code
Set	Attribute Not Settable	0x0E
Set or Get	Attribute Not Supported	0x14
Set or Get	Service Not Supported	0x08
Set or Get	Class Not Supported	0x16
Set	Value is Out of Range	0x09

## III.3.2 Position controller attributes

### III.3.2.1 Object State Conflicts – 0x0C

Three conditions could cause the drives to return this error code. To proceed, check and clear the condition.

Condition	Solution
On hard or soft limit and then issuing a command to move in the direction of the limit	Move in opposite direction of the limit
Issuing a command not support in the current mode (i.e. trying to do registration in velocity mode)	Change the mode to fit the application or issue the proper command
Trying to enable a faulted drive	Correct the fault before enabling the drive.

### III.3.2.2 Attribute 1: Number of Attributes

**Access Rule** Get

**Data Type** Unsigned Short Integer

**Description** The total number of attributes supported by the unit in the Position Controller Class.

**Range** N/A

**Default** 41

**Non-Volatile** N/A

**See Also** Attribute List

### III.3.2.3 Attribute 2: Attribute List

**Access Rule** Get

**Data Type** Array of Unsigned Short Integer

**Description** Returns an array with a list of the attributes supported by this unit in the Position Controller Class. The length of this list is specified in Number of Attributes.

**Range** Array size is defined by Attribute 1.

**Default** 1, 2, 6-16, 17, 20, 21, 23, 24, 36, 47, 49-58, 101-106, 250-255

**Non-Volatile** N/A

**See Also** Number of Attributes

### III.3.2.4 Attribute 3: Mode

**Access Rule** Get  
Set

**Data Type** Unsigned Short Integer

**Description** This attribute is used to get or set the operating mode.

**Range** 0 = Position Mode  
1 = Profiled Velocity Mode  
2 = Torque Mode

**Default** 0

**Non-Volatile** No

**See Also** Mode, Trajectory Start/Complete

### III.3.2.5 Attribute 6: Target Position

<b>Access Rule</b>	Get Set
<b>Data Type</b>	Double Integer
<b>Description</b>	This attribute specifies the target position in counts.
<b>Range</b>	$-2^{31}$ to $2^{31}$
<b>Default</b>	0
<b>Non-Volatile</b>	No
<b>See Also</b>	Actual Position, Incremental Mode Flag, Mode, Position Units

### III.3.2.6 Attribute 7: Target Velocity

<b>Access Rule</b>	Get Set
<b>Data Type</b>	Double Integer
<b>Description</b>	This attribute specifies the target velocity in counts per second.
<b>Range</b>	Set to a positive number
<b>Default</b>	According to setup
<b>Non-Volatile</b>	Yes
<b>See Also</b>	Actual Position, Incremental Mode Flag, Mode, Position Units

### III.3.2.7 Attribute 8: Acceleration

<b>Access Rule</b>	Get Set
<b>Data Type</b>	Double Integer
<b>Description</b>	This attribute specifies the acceleration for positioning, continuous velocity and homing in counts per second <sup>2</sup> .
<b>Range</b>	Set to a positive number
<b>Default</b>	According to setup
<b>Non-Volatile</b>	Yes
<b>See Also</b>	Deceleration, Profile Units

### III.3.2.8 Attribute 9: Deceleration

<b>Access Rule</b>	Get Set
<b>Data Type</b>	Double Integer
<b>Description</b>	This attribute specifies the deceleration for positioning, continuous velocity and homing in counts per second <sup>2</sup> .
<b>Range</b>	Set to a positive number
<b>Default</b>	According to setup
<b>Non-Volatile</b>	Yes
<b>See Also</b>	Acceleration, Profile Units

### III.3.2.9 Attribute 10: Incremental Position Flag

<b>Access Rule</b>	Get Set
<b>Data Type</b>	Boolean
<b>Description</b>	This bit is used to define the position value as either absolute or incremental.
<b>Range</b>	0 = Absolute Position 1 = Incremental Position
<b>Default</b>	1
<b>Non-Volatile</b>	No
<b>See Also</b>	Target Position, Trajectory Start/Complete

### III.3.2.10 Attribute 11: Trajectory Start/Complete

<b>Access Rule</b>	Get Set
<b>Data Type</b>	Boolean
<b>Description</b>	Sets a start a trajectory move. Reads cleared when a profile move is complete.
<b>Range</b>	0 = Move Complete 1 = Start Trajectory (In Motion)
<b>Default</b>	0
<b>Non-Volatile</b>	No
<b>See Also</b>	Hard Stop, Smooth Stop

### III.3.2.11 Attribute 12: On Target Position

<b>Access Rule</b>	Get
<b>Data Type</b>	Boolean
<b>Description</b>	This flag, when set, indicates that the motor is within the deadband distance to the target.
<b>Range</b>	0 = Not On Target Position 1 = In Position
<b>Default</b>	1
<b>Non-Volatile</b>	N/A
<b>See Also</b>	Feedback Resolution, Position Deadband, Trajectory Start/Complete

### III.3.2.12 Attribute 13: Actual Position

<b>Access Rule</b>	Get Set
<b>Data Type</b>	Double Integer
<b>Description</b>	The absolute position value equals the real position in counts. This is set to re-define the actual position.
<b>Range</b>	$-2^{31}$ to $2^{31}$
<b>Default</b>	0
<b>Non-Volatile</b>	No
<b>See Also</b>	Feedback Enable, Incremental Mode Flag, Position Units, Target Position

### III.3.2.13 Attribute 14: Actual Velocity

<b>Access Rule</b>	Get
<b>Data Type</b>	Double Integer
<b>Description</b>	This attribute specifies the actual velocity in counts per second.
<b>Range</b>	Positive read value
<b>Default</b>	0
<b>Non-Volatile</b>	No
<b>See Also</b>	Feedback Enable, Profile Units, Target Velocity

### III.3.2.14 Attribute 17: Enable

<b>Access Rule</b>	Get Set
<b>Data Type</b>	Boolean
<b>Description</b>	This flag is used to control the enable output. Clearing this bit sets the enable output inactive and the currently executing motion profile is aborted.
<b>Range</b>	0 = Disable 1 = Enable
<b>Default</b>	0
<b>Non-Volatile</b>	N/A
<b>See Also</b>	Actual Position, Brake Status, Peak Motor Current, RMS Motor Current, Running Current, Standing Current

### III.3.2.15 Attribute 20: Smooth Stop

**Access Rule** Get  
Set

**Data Type** Boolean

**Description** This bit is used to bring the motor to a controlled stop at the currently implemented deceleration rate.

**Range** 0 = No Action  
1 = Perform Smooth Stop

**Default** 0

**Non-Volatile** No

**See Also** Acceleration, Deceleration, Hard Stop, Trajectory Start/Complete

### III.3.2.16 Attribute 21: Hard Stop

**Access Rule** Get  
Set

**Data Type** Boolean

**Description** This bit is used to bring the motor to an immediate stop.

**Range** 0 = No Action  
1 = Perform Hard Stop

**Default** 0

**Non-Volatile** No

**See Also** Smooth Stop, Peak Motor Current, Trajectory Start/Complete

### III.3.2.17 Attribute 22: Jog Velocity

**Access Rule** Get  
Set

**Data Type** Double Integer

**Description** This attribute is used to set the target velocity in profiled velocity mode. The Direction attribute is used to select the direction of the velocity move. The Trajectory Start attribute is used to begin motion.

**Range** Positive

**Default** 0

**Non-Volatile** Yes

**See Also** Mode (velocity), Direction, Trajectory Start/Complete

### III.3.2.18 Attribute 23: Direction

<b>Access Rule</b>	Get Set
<b>Data Type</b>	Boolean
<b>Description</b>	This bit is used to control the direction of the motor in profiled velocity mode
<b>Range</b>	0 = Negative Direction 1 = Positive Direction
<b>Default</b>	1
<b>Non-Volatile</b>	No
<b>See Also</b>	Mode (velocity), Reference Direction

### III.3.2.19 Attribute 24: Reference direction

<b>Access Rule</b>	Get Set
<b>Data Type</b>	Boolean
<b>Description</b>	Defines positive direction (when viewed from the motor shaft side).
<b>Range</b>	0 = Positive Clockwise Motion 1 = Positive Counter-Clockwise Motion
<b>Default</b>	0
<b>Non-Volatile</b>	Yes
<b>See Also</b>	Direction

### III.3.2.20 Attribute 25: Torque

<b>Access Rule</b>	Get Set
<b>Data Type</b>	Double Integer
<b>Description</b>	Set a new torque command in torque mode or read the current torque command.
<b>Range</b>	0 to 3280 (3280 = peak torque)
<b>Default</b>	0
<b>Non-Volatile</b>	No
<b>See Also</b>	Mode (torque), Trajectory Start

### III.3.2.21 Attribute 100: Clear Faults

<b>Access Rule</b>	Set
<b>Data Type</b>	Boolean
<b>Description</b>	Set to 1 to clear drive faults.
<b>Range</b>	0 = Do nothing. 1 = Clear Faults
<b>Default</b>	0
<b>Non-Volatile</b>	No
<b>See Also</b>	

### III.3.2.22      **Attribute 101: Save Parameters**

**Access Rule** Set

**Data Type** Boolean

**Description** Set to 1 to save drive parameters to non-volatile storage.

**Range**        0 = Do nothing.  
                 1 = Save parameters.

**Default**       0

**Non-Volatile** No

**See Also**

### III.3.2.23      **Attribute 102: Drive Status**

**Access Rule** Get

**Data Type** Double Integer

**Description** Read the drive status words. See the ASCII reference for a description of the status bits (DRVSTAT).

**Range**

**Default**

**Non-Volatile** No

**See Also**

### III.3.2.24      **Attribute 103: Trajectory Status**

**Access Rule** Get

**Data Type** Double Integer

**Description** Read the drive trajectory status words. See the ASCII reference for a description of the status bits (TRJSTAT).

**Range**

**Default**

**Non-Volatile** No

**See Also**

### III.4 Block Sequencer Object Class (ID=38)

This object handles the execution of Command Blocks or Command Block chains.

#### III.4.1 Attribute 1: Block

**Access Rule** Get  
Set

**Data Type** Unsigned Short Integer

**Description** This value defines the starting Command Block instance number to execute.

**Range** 1 to 255

**Default** N/A

**Non-Volatile** N/A

**See Also** Block Execute

#### III.4.2 Attribute 2: Block Execute

**Access Rule** Get  
Set

**Data Type** Boolean

**Description** Executes the starting command block defined by Attribute 1.

**Range** 0 = Clear or Complete  
1 = Block Executing

**Default** 0

**Non-Volatile** N/A

**See Also** Block, Block Fault

### III.4.3 Attribute 3: Current Block

<b>Access Rule</b>	Get
<b>Data Type</b>	Unsigned Short Integer
<b>Description</b>	This attribute returns the command block instance number of the currently-executing block.
<b>Range</b>	1 to 255
<b>Default</b>	N/A
<b>Non-Volatile</b>	N/A
<b>See Also</b>	Block, Block Execute

### III.4.4 Attribute 4: Block Fault

<b>Access Rule</b>	Get
<b>Data Type</b>	Boolean
<b>Description</b>	Set when a block error occurs. When a block fault error occurs, block execution stops. This bit is reset when the block fault code (5) is read.
<b>Range</b>	0 = No Block Faults 1 = Block Fault Occurred
<b>Default</b>	0
<b>Non-Volatile</b>	
<b>See Also</b>	Block Execute, Block Fault Code

### III.4.5 Attribute 5: Block Fault Code

<b>Access Rule</b>	Get
<b>Data Type</b>	Boolean
<b>Description</b>	This attribute defines the specific block fault.
<b>Range</b>	0 = No Fault 1 = Invalid or Empty Block 2 = Command Time-out (Wait Equals) 3 = Execution Fault
<b>Default</b>	N/A
<b>Non-Volatile</b>	N/A
<b>See Also</b>	Block Fault

### III.4.6 Attribute 6: Counter

<b>Access Rule</b>	Get Set
<b>Data Type</b>	Double Integer
<b>Description</b>	This value is used as a global counter for motion tasks.
<b>Range</b>	Positive
<b>Default</b>	0
<b>Non-Volatile</b>	No
<b>See Also</b>	Decrement Counter Command (Block Object)

### III.5 Command Block Object Class (ID=39)

Each instance of the Command Block object defines a specific command. These blocks can be linked to other blocks to form a command block chain. Looping and branching commands are supported. See the individual commands for additional attribute information.

#### III.5.1 Command 01 – Modify Attribute

This command is used to modify an Attribute's value.

##### III.5.1.1 Attribute 1: Block Command

**Access Rule** Get  
Set

**Data Type** Unsigned Short Integer

**Description** This attribute specifies the command to be performed.

**Range** N/A

**Default** 0x01 = Command 01

**Non-Volatile**

**See Also**

##### III.5.1.2 Attribute 2: Block Link #

**Access Rule** Get  
Set

**Data Type** Unsigned Short Integer

**Description** This attribute provides a link to the next block instance to execute. When this block is complete, the link block is executed.

**Range** 1 to 255

**Default** 1

**Non-Volatile**

**See Also**

##### III.5.1.3 Attribute 3: Target Class

**Access Rule** Get  
Set

**Data Type** Double Integer

**Description** This attribute defines the target class number to be sequenced.

**Range**

**Default**

**Non-Volatile**

**See Also**

### III.5.1.4 Attribute 4: Target Instance

**Access Rule** Get  
Set

**Data Type** Double Integer

**Description** This attribute defines the instance number of the target class to be sequenced.

**Range**

**Default**

**Non-Volatile**

**See Also**

### III.5.1.5 Attribute 5: Attribute #

**Access Rule** Get  
Set

**Data Type** Double Integer

**Description** This attribute defines the Position Controller Class Attribute Number. The Position Controller Class Attribute Number must be settable.

**Range**

**Default**

**Non-Volatile**

**See Also**

### III.5.1.6 Attribute 6: Attribute Data

**Access Rule** Get  
Set

**Data Type** Double Integer

**Description** This is the new attribute data.

**Range**

**Default**

**Non-Volatile**

**See Also**

## III.5.2 Command 02– Wait Until Equals

This command is used to wait until a parameter equals a desired value.

### III.5.2.1 Attribute 1: Block Command

**Access Rule** Get  
Set

**Data Type** Unsigned Short Integer

**Description** This attribute specifies the command to be performed.

**Range** 0x02 = Command 02

**Default** N/A

**Non-Volatile**

**See Also**

### III.5.2.2 Attribute 2: Block Link #

**Access Rule** Get  
Set

**Data Type** Unsigned Short Integer

**Description** This attribute provides a link to the next block instance to execute. When this block is complete, the link block is executed.

**Range**

**Default**

**Non-Volatile**

**See Also**

### III.5.2.3 Attribute 3: Target Class

**Access Rule** Get  
Set

**Data Type** Double Integer

**Description** This attribute defines the target class number to be sequenced.

**Range**

**Default**

**Non-Volatile**

**See Also**

### III.5.2.4 Attribute 4: Target Instance

**Access Rule** Get  
Set

**Data Type** Double Integer

**Description** This attribute defines the instance number of the target class to be sequenced.

**Range**

**Default**

**Non-Volatile**

**See Also**

### III.5.2.5 Attribute 5: Attribute #

**Access Rule** Get  
Set

**Data Type** Double Integer

**Description** This attribute defines the Position Controller Class Attribute Number. The Position Controller Class Attribute Number must be settable.

**Range**

**Default**

**Non-Volatile**

**See Also**

### III.5.2.6 Attribute 6: Timeout

**Access Rule** Get  
Set

**Data Type** Double Integer

**Description** Maximum time in ms to wait for the parameter to equal the desired value. A fault is issued if the timer expires. If set to 0, motion tasking will wait without faulting.

**Range**

**Default**

**Non-Volatile**

**See Also**

### III.5.2.7 Attribute 7: Compare Data

**Access Rule** Get  
Set

**Data Type** Dependent on Attribute #

**Description** The attribute is compared to this value. If they are equal, motion will continue; otherwise, the drive will wait.

**Range**

**Default**

**Non-Volatile**

**See Also**

### III.5.3 Command 03– Conditional Link Greater Than Command

This command is used for conditional linking or branching in a linked chain of commands.

#### III.5.3.1 Attribute 1: Block Command

**Access Rule** Get  
Set

**Data Type** Unsigned Short Integer

**Description** This attribute specifies the command to be performed.

**Range** 0x03 = Command 03

**Default** N/A

**Non-Volatile**

**See Also**

#### III.5.3.2 Attribute 2: Block Link #

**Access Rule** Get  
Set

**Data Type** Unsigned Short Integer

**Description** This attribute provides a link to the next block instance to execute. When this block is complete, the link block is executed.

**Range**

**Default**

**Non-Volatile**

**See Also**

#### III.5.3.3 Attribute 3: Target Class

**Access Rule** Get  
Set

**Data Type** Double Integer

**Description** This attribute defines the target class number to be sequenced.

**Range**

**Default**

**Non-Volatile**

**See Also**

### III.5.3.4 Attribute 4: Target Instance

**Access Rule** Get  
Set

**Data Type** Double Integer

**Description** This attribute defines the instance number of the target class to be sequenced.

**Range**

**Default**

**Non-Volatile**

**See Also**

### III.5.3.5 Attribute 5: Attribute #

**Access Rule** Get  
Set

**Data Type** Double Integer

**Description** This attribute defines the Position Controller Class Attribute Number. The Position Controller Class Attribute Number must be settable.

**Range**

**Default**

**Non-Volatile**

**See Also**

### III.5.3.6 Attribute 6: Compare Link #

**Access Rule** Get  
Set

**Data Type** Double Integer

**Description** Conditional link number/alternate link block if this attribute exceeds compare data.

**Range**

**Default**

**Non-Volatile**

**See Also**

### III.5.3.7 Attribute 7: Compare Data

**Access Rule** Get  
Set

**Data Type** Dependent on Attribute #

**Description** This attribute compares the data for the conditional link. If Attribute 6 is greater than the compare data, the normal link (Attribute 2) is ignored and the next block executed is the compare link block.

**Range**

**Default**

**Non-Volatile**

**See Also**

### III.5.4 Command 04– Conditional Link Less Than Command

This command is used for conditional linking or branching in a linked chain of commands.

#### III.5.4.1 Attribute 1: Block Command

**Access Rule** Get  
Set

**Data Type** Unsigned Short Integer

**Description** This attribute specifies the command to be performed.

**Range** 0x04 = Command 04

**Default** N/A

**Non-Volatile**

**See Also**

#### III.5.4.2 Attribute 2: Block Link #

**Access Rule** Get  
Set

**Data Type** Unsigned Short Integer

**Description** This attribute provides a link to the next block instance to execute. When this block is complete, the link block is executed.

**Range** Depends on command

**Default**

**Non-Volatile**

**See Also**

#### III.5.4.3 Attribute 3: Target Class

**Access Rule** Get  
Set

**Data Type** Double Integer

**Description** This attribute defines the target class number to be sequenced.

**Range**

**Default**

**Non-Volatile**

**See Also**

### III.5.4.4 Attribute 4: Target Instance

**Access Rule** Get  
Set

**Data Type** Double Integer

**Description** This attribute defines the instance number of the target class to be sequenced.

**Range**

**Default**

**Non-Volatile**

**See Also**

### III.5.4.5 Attribute 5: Attribute #

**Access Rule** Get  
Set

**Data Type** Double Integer

**Description** This attribute defines the Position Controller Class Attribute Number. The Position Controller Class Attribute Number must be settable.

**Range**

**Default**

**Non-Volatile**

**See Also**

### III.5.4.6 Attribute 6: Compare Link #

**Access Rule** Get  
Set

**Data Type** Double Integer

**Description** Conditional link number/alternate link block if this attribute is less than compare data.

**Range**

**Default**

**Non-Volatile**

**See Also**

### III.5.4.7 Attribute 7: Compare Data

**Access Rule** Get  
Set

**Data Type** Dependent on Attribute #

**Description** Compares the data for the conditional link. If Attribute 6 is less than the compare data, the normal link (Attribute 2) is ignored and the next block executed is the compare link block.

**Range**

**Default**

**Non-Volatile**

**See Also**

### III.5.5 Command 05– Decrement Counter

This command is used to decrement the global counter.

#### III.5.5.1 Attribute 1: Block Command

**Access Rule** Get  
Set

**Data Type** Unsigned Short Integer

**Description** This attribute specifies the command to be performed.

**Range** 0x05 = Command 05

**Default** N/A

**Non-Volatile**

**See Also**

#### III.5.5.2 Attribute 2: Block Link #

**Access Rule** Get  
Set

**Data Type** Unsigned Short Integer

**Description** This attribute provides a link to the next block instance to execute. When this block is complete, the link block is executed.

**Range** Depends on command

**Default**

**Non-Volatile**

**See Also**

### III.5.6 Command 06– Delay Command

This command is used to delay a linked chain of commands.

#### III.5.6.1 Attribute 1: Block Command

**Access Rule** Get  
Set

**Data Type** Unsigned Short Integer

**Description** This attribute specifies the command to be performed.

**Range** 0x06 = Command 06

**Default** N/A

**Non-Volatile**

**See Also**

### III.5.6.2 Attribute 2: Block Link #

**Access Rule** Get  
Set

**Data Type** Unsigned Short Integer

**Description** This attribute provides a link to the next block instance to execute. When this block is complete, the link block is executed.

**Range** Depends on command

**Default**

**Non-Volatile**

**See Also**

### III.5.6.3 Attribute 3: Delay

**Access Rule** Get  
Set

**Data Type** Double Integer

**Description** This attribute sets the delay in milliseconds.

**Range**

**Default**

**Non-Volatile**

**See Also**

## III.5.7 Command 08– Trajectory Command and Wait

This command is used to initiate a move and wait for completion.

### III.5.7.1 Attribute 1: Block Command

**Access Rule** Get  
Set

**Data Type** Unsigned Short Integer

**Description** This attribute specifies the command to be performed.

**Range** 0x08 = Command 08

**Default** N/A

**Non-Volatile**

**See Also**

### III.5.7.2 Attribute 2: Block Link #

**Access Rule** Get  
Set

**Data Type** Unsigned Short Integer

**Description** This attribute provides a link to the next block instance to execute. When this block is complete, the link block is executed.

**Range**

**Default**

**Non-Volatile**

**See Also**

### III.5.7.3 Attribute 3: Target Position

**Access Rule** Get  
Set

**Data Type** Double Integer

**Description** This attribute defines the target profile position in position units.

**Range**

**Default**

**Non-Volatile**

**See Also**

### III.5.7.4 Attribute 4: Target Velocity

**Access Rule** Get  
Set

**Data Type** Double Integer

**Description** This attribute defines the target profile velocity in profile units per second.

**Range**

**Default**

**Non-Volatile**

**See Also**

### III.5.7.5 Attribute 5: Incremental

**Access Rule** Get  
Set

**Data Type** Boolean

**Description** This flag defines if the motion is incremental or absolute.

**Range** 0 = Absolute Position  
1 = Incremental Position

**Default** 0

**Non-Volatile**

**See Also**

### III.6 Polled I/O Command Assemblies

Polled I/O Command Assemblies are a method of communicating to devices a group of specific commands. This method of communication is preferred, as it is faster than explicit messaging. In this section, the format for each Command Assembly is defined and examples of each are provided.



*All eight bytes of data are ignored unless a valid command assembly type is specified in byte 2 (valid command assembly types are 1 through 5).*

*A valid Response Command Assembly, (decimal; 1 to 8) is required before setting any other attribute. The controllers do not respond if the Response Command Assembly is not valid.*

*Data outside the range of the attribute is ignored and the Invalid Poll Data bit of the Response Assembly is set. This applies to all Command Assemblies, except Assembly 1.*

*The drive must be homed before motion is begun. Failure to home the drive will result in a drive alarm that must be cleared before drive operation can continue.*

#### III.6.1 Running a Stored Sequence Through DeviceNet

A motion tasking sequence may be setup in the Drive program or through DeviceNet (see the Command Block object) and then executed later through DeviceNet. See the setup software manual (onlinehelp) for instructions on creating a motion tasking sequence.

To execute a motion block sequence, set Block Number equal to the index of the block to begin executing and transition the Start Block bit high. Enable must be high and the stop bits must be low.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Enable	Reg Arm	Hard Stop	Smooth Stop	Dir (Vel Mode)	Incremental	Start Block	Trajectory Start
1	Block Number							
2	0	0	1	0	Input Command Assembly Type (0000)			
3	0	0	1	0	Output Response Assembly Type			
4	0							
5	0							
6	0							
7	0							

#### III.6.2 Stopping a Program Through DeviceNet

To stop an executing sequence, set the Smooth Stop or Hard Stop bit high.

### III.6.3 Command Assembly 1 – Target Position

This command assembly is used to start a trajectory (position mode only) of the specified distance. The trajectory can be absolute or relative.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Enable	Reg Arm	Hard Stop	Smooth Stop	Dir (Vel Mode)	Incremental	Start Block	Trajectory Start
1	Block Number							
2	0	0	1	0	Input Command Assembly Type (0001)			
3	0	0	1	0	Output Response Assembly Type			
4	Target Position Low Byte							
5	Target Position Low Middle Byte							
6	Target Position High Middle Byte							
7	Target Position High Byte							

**Enable** Setting this bit enables the drive. Also see Enable, Position Controller Object Class (ID=37).

**Registration Arm** Not supported at this time.

**Hard Stop** Setting this bit causes the drive to stop immediately (without decelerating). Also see Hard Stop, Position Controller Object Class (ID=37).

**Smooth Stop** Setting this bit causes the drive to decelerate to a stop. Also see Smooth Stop, Position Controller Object Class (ID=37).



**To stop motion, issue either HARD STOP or SMOOTH STOP only. Changing either one of these bits at the same time as the Start Trajectory bit causes indeterminate action from the controller.**

**Direction** This bit is used only in velocity mode. It is used to instantaneously change the direction of travel. Also see Direction, Position Controller Class Object Class (ID=37). Used only with Command Assemblies 2 and 10.

**Incremental** This bit is used in only in position mode. This bit indicates whether the position specified in bytes 4 through 7 of the Command Assembly 1 – Target Position, is absolute (0) or incremental (1). See the description for Incremental Mode Flag.

**Start Block** By setting this bit high (1) and the Block Number to zero (0), the command executes programs previously generated and stored in the drive. The program executed is defined by the last four bytes of the Command Assembly. To stop program execution, set Start Block high (1) and Block Number to high (1). Programs can be executed with any Command Assembly. See the example provided.



**Setting Start Block High (1) and issuing a transition from 0 to 1 for Trajectory Start simultaneously causes indeterminate action.**

**Trajectory Start** The transition of this bit from 0 to 1 starts a move in Command Assembly 1: Target Position. For all other command assemblies, the transition of this bit sets the data value (i.e., velocity, acceleration, etc.). Also see Trajectory Start, Position Controller Object Class (ID=37).

**Block Number** Used with Start Block to run a program previously defined in the drive. With Start Block set to high (1) and Block Number set to zero (0), the drive executes the program indicated by the last four bytes of the Assembly. To stop the program, set Start Block high (1) and Block Number to high (1). Programs can be executed with any Command Assembly, except Assembly 1.



***Setting an out of range attribute value causes the Invalid Poll Data bit to be set in the Polled I/O Response Assembly. This applies to all Assemblies.***

Shown below is an example for the SERVOSTAR 600 drives. In this example, motor resolution was set to 1000 counts per rev using PGEARI=1000 and PGEARO=1048576. The target position is set to 1 rev (1000 position units, or in hexadecimal, 0x000003E8 position units). The enable bit is set, along with the incremental bit and the start bit. Lastly, this example indicates that the desired Polled I/O Response Assemblies is the actual velocity (refer to Response Assembly 3 – Actual Velocity for more information).

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	0	0	0	0	1	0	1
1	0							
2	0	0	1	0	0	0	0	1
3	0	0	1	0	0	0	1	1
4	0xE8							
5	0x03							
6	0x00							
7	0x00							



***You must make sure the transition of Trajectory Start is seen by the drive. The scan time of the system determines if the drive sees the change in bit 0. You may have to wait one scan cycle or more to make sure the change is seen by the drive.***

***Below is an example you may wish to use to determine if the drive has accepted the new command. In this example, we wish to change the Trajectory Start bit. By changing the Response Assembly at the same time, the drive is forced to respond with a message that may not correspond to the actual values in the system***

Example	Command	Response	Notes
Current Value	0x80 0x00 0x21 0x21	0x94 0x00 0x00 0x21	The current value is shown
Desired Value	<b>0x81</b> 0x00 0x21 <b>0x22</b>	0x94 0x00 0x00 0x21	Desired value shown in bold, along with bit to indicate change
Next Cycle	<b>0x81</b> 0x00 0x21 <b>0x22</b>	0x94 0x00 0x00 0x21	Trajectory Start bit not seen yet because the last byte stay the same
Next Cycle	0x81 0x00 0x21 0x22	0x94 0x00 0x00 0x22	Trajectory Start bit seen because the last byte changed.

### III.6.4 Command Assembly 2 – Target Velocity

This command assembly is used to change the target velocity. This can only be used in position or velocity mode. The Dir bit sets the desired direction in Velocity Mode. Set Trajectory Start with this command assembly to begin motion in Velocity Mode or load the target velocity in Position Mode.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Enable	Reg Arm	Hard Stop	Smooth Stop	Dir (Vel Mode)	Incremental	Start Block	Trajectory Start
1	Block Number							
2	0	0	1	0	Input Command Assembly Type (0010)			
3	0	0	1	0	Output Response Assembly Type			
4	Target Velocity Low Byte							
5	Target Velocity Low Middle Byte							
6	Target Velocity High Middle Byte							
7	Target Velocity High Byte							

See Command Assembly 1: Target Position for bit descriptions.

Shown below is an example for the SERVOSTAR 600 drives. In this example, motor resolution was set to 1000 counts per rev using PGEARI=1000 and PGEARO=1048576. The target velocity is set to 1200 rpm (or 20000 position units/sec, or in hexadecimal, 0x00004e20 position units/sec). The enable bit is set, along with the start trajectory (which instructs the drive to change the target velocity). In velocity mode, the drive immediately accelerates or decelerates to 1200 rpm. In position mode, the next trajectory has a target velocity of 20 RPS. Lastly, this example indicates that the desired Polled I/O Response Assemblies is the actual position (refer to Response Assembly 1 – Actual Position for more information).

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	0	0	0	0	0	0	1
1	0							
2	0	0	1	0	0	0	1	0
3	0	0	1	0	0	0	0	1
4	0x20							
5	0x4e							
6	0x00							
7	0x00							

### III.6.5 Command Assembly 3 – Acceleration

This command assembly is used to change the acceleration. This can only be used in position or velocity mode.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Enable	Reg Arm	Hard Stop	Smooth Stop	Dir (Vel Mode)	Incremental	Start Block	Trajectory Start
1	Block Number							
2	0	0	1	0	Input Command Assembly Type (0011)			
3	0	0	1	0	Output Response Assembly Type			
4	Acceleration Low Byte							
5	Acceleration Low Middle Byte							
6	Acceleration High Middle Byte							
7	Acceleration High Byte							

See Command Assembly 1: Target Position for bit descriptions.

Shown below is an example for the SERVOSTAR 600 drives. In this example, motor resolution was set to 1000 counts per rev using PGEARI=1000 and PGEARO=1048576. The acceleration is set to 1200 rpm<sup>2</sup> (or 20000 position units/sec<sup>2</sup>, or in hexadecimal, 0x00004e20 position units/ sec<sup>2</sup>). The enable bit is set, along with the Start Trajectory/Load Data bit (which instructs the drive to change the acceleration). Lastly, this example indicates that the desired Polled I/O Response Assemblies is the actual position (refer to Response Assembly 1 – Actual Position for more information).

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	0	0	0	0	0	0	1
1	0							
2	0	0	1	0	0	0	1	1
3	0	0	1	0	0	0	0	1
4	0x20							
5	0x4e							
6	0x00							
7	0x00							

### III.6.6

#### Command Assembly 4 – Deceleration

This command assembly is used to change the deceleration. This can only be used in position or velocity mode.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Enable	Reg Arm	Hard Stop	Smooth Stop	Dir (Vel Mode)	Incremental	Start Block	Trajectory Start
1	Block Number							
2	0	0	1	0	Input Command Assembly Type (0100)			
3	0	0	1	0	Output Response Assembly Type			
4	Deceleration Low Byte							
5	Deceleration Low Middle Byte							
6	Deceleration High Middle Byte							
7	Deceleration High Byte							

See Command Assembly 1: Target Position for bit descriptions.

Shown below is an example for the SERVOSTAR 600 drives. In this example, motor resolution was set to 1000 counts per rev using PGEARI=1000 and PGEARO=1048576. The deceleration is set to 1200 rpm<sup>2</sup> (or 20000 position units/sec<sup>2</sup>, or in hexadecimal, 0x00004e20 position units/ sec<sup>2</sup>). The enable bit is set, along with the Start Trajectory/Load Data bit (which instructs the drive to change the deceleration). Lastly, this example indicates that the desired Polled I/O Response Assemblies is the actual position (refer to Response Assembly 1 – Actual Position for more information).

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	0	0	0	0	0	0	1
1	0							
2	0	0	1	0	0	1	0	0
3	0	0	1	0	0	0	0	1
4	0x20							
5	0x4e							
6	0x00							
7	0x00							

### III.6.7 Command Assembly 5 – Torque

This command assembly is used to change the torque. This can only be used in torque mode.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Enable	Reg Arm	Hard Stop	Smooth Stop	Dir (Vel Mode)	Incremental	Start Block	Trajectory Start
1	Block Number							
2	0	0	1	0	Input Command Assembly Type (0101)			
3	0	0	1	0	Output Response Assembly Type			
4	Torque Low Byte							
5	Torque Low Middle Byte							
6	Torque High Middle Byte							
7	Torque High Byte							

See Command Assembly 1: Target Position for bit descriptions.

Shown below is an example for the SERVOSTAR 600 drives. In this example, the torque (current) is set to 3.0A in a 6.0A peak drive. Torque units are scaled to 3280=peak current, so the command value is  $3280 \times 3.0 / 6.0 = 1640$  torque units (in hexadecimal, 0x00000668 torque units). The enable bit is set, along with the start trajectory (which instructs the drive to change the torque). Lastly, this example indicates that the desired Polled I/O Response Assemblies is the actual position (refer to Response Assembly 1 – Actual Position for more information).

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	0	0	0	0	0	0	1
1	0							
2	0	0	1	0	0	1	0	1
3	0	0	1	0	0	0	0	1
4	0x68							
5	0x06							
6	0x00							
7	0x00							

## III.7 I/O Response Assemblies

### III.7.1 Response Assembly 1 – Actual Position

This response assembly is used to return the Actual Position of the motor (in position units).

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Enable state	Reg level	Home level	Current dir	General fault	On target position	Block in execution	Trajectory Start
1	Executing Block Number							
2	Load complete	Block fault	Following error	Negative limit	Positive limit	CCW limit	CW limit	Fault input active
3	0	0	1	Output Response Assembly Type (00001)				
4	Actual position Low Byte							
5	Actual position Low Middle Byte							
6	Actual position High Middle Byte							
7	Actual position High Byte							

<b>Enable State</b>	This bit reflects the enable state of the drive. See the description for Enable (Class 37 – Position Controller, Attribute 17).
<b>Registration Level</b>	Not supported at this time.
<b>Home Level</b>	This bit reflects the level of the Home Input of the drive. See the description for Home Level (Class 36 – Position Controller Supervisor, Attribute 16).
<b>Current Direction</b>	This bit reflects the direction of motion. See the description for Direction (Class 37 – Position Controller, Attribute 23).
<b>General Fault</b>	This bit indicates whether or not a fault has occurred. See the description for General Fault (Class 36 – Position Controller Supervisor, Attribute 5).
<b>On Target Position</b>	This bit indicates whether or not the motor is on the last targeted position (1-On Target). See the description for Incremental Position Flag (Class 37 – Position Controller, Attribute 12).
<b>Block in Execution</b>	When set, indicates drive is running a program.
<b>Trajectory in Progress</b>	This bit indicates whether a trajectory is in progress (1) or has completed (0). This bit is set immediately for Command Assemblies 1, 10, & 11 and remains set for the entire motion. See the description for Start Trajectory (Class 37 – Position Controller, Attribute 11).
<b>Load Complete</b>	This bit indicates that the command data contained in the command message has been successfully loaded into the device.
<b>Executing Block Number</b>	Indicates whether the drive is running a task sequence or not. When value = 0, the drive is running a sequence. When set to 1, the drive is not running a sequence.
<b>Following Error</b>	This bit indicates when a following (static or dynamic) error occurs. See the description for Start Trajectory (Class 37 – Position Controller, Attribute 47).
<b>Negative Limit</b>	This bit indicates when the position is less than or equal to the Negative Soft Limit Position. See the description for Negative Soft Limit (Class 37 – Position Controller, Attribute 57).
<b>Positive Limit</b>	This bit indicates when the position is less than or equal to the Positive Soft Limit Position. See the description for Positive Soft Limit (Class 37 – Position Controller, Attribute 56).
<b>CCW Limit</b>	This bit indicates the state of the CCW Limit Input. See the description for CCW Limit (Class 37 – Position Controller, Attribute 51).
<b>CW Limit</b>	This bit indicates the state of the CW Limit Input. See the description for CW Limit (Class 37 – Position Controller, Attribute 50).
<b>Fault Input Active</b>	This bit indicates the state of the Fault input. See the description for CW Limit (Class 36 – Position Controller Supervisor, Attribute 8).

Shown below is an example for the SERVOSTAR 600 drives. In this example, motor resolution was set to 1000 counts per rev using PGEARI=1000 and PGEARO=1048576. The actual position is 10 revolutions (or 10,000 position units, or in hexadecimal, 0x00002710 position units). The enable bit is set, the Registration Level is 0, the Home Level is 0, the direction is positive (1), there are no faults (0), we are On Target Position (1), no Block in Execution (0) and no Trajectory in Progress, no Block Fault (0), no Following Error (0), not on Negative Limit (0), on Positive Limit (1), not on CCW Limit (0), on CW Limit (1), the command data was loaded successfully, and the Fault Input is not active (0).

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	0	0	1	0	1	0	1
1	0							
2	1	0	1	0	1	0	1	0
3	0	0	1	0	0	0	0	1
4	0x10							
5	0x27							
6	0x01							
7	0x00							

### III.7.2

#### Response Assembly 2 – Commanded Position

This response assembly is used to return the commanded position of the motor (in position units).

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Enable state	Reg level	Home level	Current dir	General fault	On target position	Block in execution	Trajectory Start
1	Executing Block Number							
2	Load complete	Block fault	Following error	Negative limit	Positive limit	CCW limit	CW limit	Fault input active
3	0	0	1	Output Response Assembly Type (00010)				
4	Commanded position Low Byte							
5	Commanded position Low Middle Byte							
6	Commanded position High Middle Byte							
7	Commanded position High Byte							

Shown below is an example for the SERVOSTAR 600 drives. In this example, motor resolution was set to 1000 counts per rev using PGEARI=1000 and PGEARO=1048576. The commanded position is 10 revolutions (or 10,000 position units, or in hexadecimal, 0x00002710 position units). The enable bit is set, the Registration Level is 0, the Home Level is 0, the direction is positive (1), there are no faults (0), we are On Target Position (1), no Block in Execution (0) and no Trajectory in Progress, no Block Fault (0), no Following Error (0), not on Negative Limit (0), on Positive Limit (1), not on CCW Limit (0), on CW Limit (1), the command data was loaded successfully, and the Fault Input is not active (0).

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	0	0	1	0	1	0	1
1	0							
2	1	0	0	0	1	0	0	0
3	0	0	1	0	0	0	1	0
4	0x10							
5	0x27							
6	0x00							
7	0x00							

### III.7.3 Response Assembly 3 – Actual Velocity

This response assembly returns the actual velocity of the motor (in position units/sec).

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Enable state	Reg level	Home level	Current dir	General fault	On target position	Block in execution	Trajectory Start
1	Executing Block Number							
2	Load complete	Block fault	Following error	Negative limit	Positive limit	CCW limit	CW limit	Fault input active
3	0	0	1	Output Response Assembly Type (00011)				
4	Actual velocity Low Byte							
5	Actual velocity Low Middle Byte							
6	Actual velocity High Middle Byte							
7	Actual velocity High Byte							

Shown below is an example for the SERVOSTAR 600 drives. In this example, motor resolution was set to 1000 counts per rev using PGEARI=1000 and PGEARO=1048576. The actual velocity is 10 revs/sec (or 10,000 position units/sec, or in hexadecimal, 0x00002710 position units/sec). The enable bit is set, the Registration Level is 0, the Home Level is 0, the direction is positive (1), there are no faults (0), we are On Target Position (1), no Block in Execution (0) and no Trajectory in Progress, no Block Fault (0), no Following Error (0), not on Negative Limit (0), on Positive Limit (1), not on CCW Limit (0), on CW Limit (1), the command data was loaded successfully, and the Fault Input is not active (0).

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	0	0	1	0	1	0	1
1	0							
2	1	0	0	0	1	0	0	0
3	0	0	1	0	0	0	1	1
4	0x10							
5	0x27							
6	0x00							
7	0x00							

### III.7.4 Response Assembly 5 – Torque

This response assembly returns the actual torque (current) of the motor.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Enable state	Reg level	Home level	Current dir	General fault	On target position	Block in execution	Trajectory Start
1	Executing Block Number							
2	Load complete	Block fault	Following error	Negative limit	Positive limit	CCW limit	CW limit	Fault input active
3	0	0	1	Output Response Assembly Type (00101)				
4	Torque Low Byte							
5	Torque Low Middle Byte							
6	Torque High Middle Byte							
7	Torque High Byte							

Shown below is an example for the SERVOSTAR 600 drives. The actual torque (current) is 3.0A in a 6.0A peak drive. Torque units are scaled to 3280=peak current, so the actual torque value is  $3280 \times 3.0/6.0 = 1640$  torque units (in hexadecimal, 0x00000668 torque units). The enable bit is set, the Registration Level is 0, the Home Level is 0, the direction is positive (1), there are no faults (0), we are On Target Position (1), no Block in Execution (0), no Trajectory in Progress, no Block Fault (0), no Following Error (0), not on Negative Limit (0), on Positive Limit (1), not on CCW Limit (0), not on CW Limit (1) and the Fault Input is not active (0).

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	0	0	1	0	1	0	1
1	0							
2	0	0	0	0	1	0	0	0
3	0	0	1	0	0	1	0	1
4	0x68							
5	0x06							
6	0x00							
7	0x00							

### III.7.5

#### Response Assembly 20 – Command/Response Error

This response identifies an error that has occurred. This response will always be returned in response to an invalid Command Assembly.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Enable state	Reg level	Home level	Current dir	General fault	On target position	Block in execution	Trajectory Start
1	Executing Block Number							
2	Load complete	Block fault	Following error	Negative limit	Positive limit	CCW limit	CW limit	Fault input active
3	0	0	1	Output Response Error (10100)				
4	General error code							
5	Additional code							
6	Copy of command message byte2							
7	Copy of command message byte3							

Error Code (hex)	Additional Code (hex)	DeviceNet Error
2	FF	RESOURCE_UNAVAILABLE
5	FF	PATH_UNKNOWN
5	1	COMMAND_AXIS_INVALID
5	2	RESPONSE_AXIS_INVALID
8	FF	SERVICE_NOT_SUPP
8	1	COMMAND_NOT_SUPPORTED
8	2	RESPONSE_NOT_SUPPORTED
9	FF	INVALID_ATTRIBUTE_VALUE
B	FF	ALREADY_IN_STATE
C	FF	OBJ_STATE_CONFLICT
D	FF	OBJECT_ALREADY_EXISTS
E	FF	ATTRIBUTE_NOT_SETTABLE
F	FF	ACCESS_DENIED
10	FF	DEVICE_STATE_CONFLICT
11	FF	REPLY_DATA_TOO_LARGE
13	FF	NOT_ENOUGH_DATA
14	FF	ATTRIBUTE_NOT_SUPP
15	FF	TOO_MUCH_DATA
16	FF	OBJECT_DOES_NOT_EXIST
17	FF	FRAGMENTATION_SEQ_ERR
20	FF	INVALID_PARAMETER

Shown below is an example for the SERVOSTAR 600 drives. The previous Command Assembly requested command 0x06, which is not supported and response 0x01, which is supported. The drive returns Response Assembly 20 (Command/Response Error) with General Error = 8 (SERVICE\_NOT\_SUPPORTED) and Additional Code = 1 (COMMAND\_NOT\_SUPPORTED). The command and response bytes from the Command Assembly are echoed in the Error Response Assembly.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	0	0	1	0	1	0	1
1	0							
2	0	0	0	0	0	0	0	0
3	0	0	1	1	0	1	0	0
4	0x08							
5	0x01							
6	0x26							
7	0x21							

### III.8

### Identity Object Class

Identity Object 0x01						
Object Class	ID	Description	Get	Set	Value Limits	
Attributes	Open	1	Revision			
		2	Max instance			
X None Supported		3	Number of Instances			
		4	Optional attributes list			
		5	Optional services list			
		6	Max Id of class attributes			
		7	Max Id of instance attributes			
<b>DeviceNet Services</b>			<b>Parameter Options</b>			
Services		Get_Attributes_All				
		Reset				
X None Supported		Get_Attribute_Single				
		Find_Next_Object_instance				
Object Instance	ID	Description	Get	Set	Value Limits	
Attributes	Open	1	Vendor	X		=(452)
		2	Device type	X		=(16)
		3	Product code	X		=(3)
		4	Revision	X		=(1.1)
		5	Status (bits supported)	X		
		6	Serial number	X		
		7	Product name	X		ServoStar 603
		8	State			
		9	Config. Consistency Value			
		10	Heartbeat Interval			
<b>DeviceNet Services</b>			<b>Parameter Options</b>			
Services		Get_Attributes_All				
		X Reset		0,1		
		X Get_Attribute_Single				
		Set_Attribute_Single				

### III.9 Message Router Object Class

Message Router Object 0x02						
Object Class	ID	Description	Get	Set	Value Limits	
Attributes      Open	1	Revision				
	2	Max instance				
	X None Supported	3	Number of Instances			
		4	Optional attributes list			
		5	Optional services list			
		6	Max Id of class attributes			
		7	Max Id of instance attributes			
		DeviceNet Services	Parameter Options			
Services		Get_Attributes_All				
X None Supported		Get_Attribute_Single				
Object Instance	ID	Description	Get	Set	Value Limits	
Attributes      Open	1	Object list				
	2	Maximum connections supported				
X None Supported	3	Number of active connections				
	4	Active connections list				
		DeviceNet Services	Parameter Options			
Services		Get_Attributes_All				
		Get_Attribute_Single				
X None Supported		Set_Attribute_Single				

### III.10 DeviceNet Object Class

DeviceNet Object 0x03						
Object Class	ID	Description	Get	Set	Value Limits	
Attributes      Open	1	Revision	X			
	2	Max instance				
	None Supported	3	Number of Instances			
		4	Optional attributes list			
		5	Optional services list			
		6	Max Id of class attributes			
		7	Max Id of instance attributes			
		DeviceNet Services	Parameter Options			
Services	X	Get_Attribute_Single				
None Supported						
Object Instance	ID	Description	Get	Set	Value Limits	
Attributes      Open	1	MAC ID	X			
	2	Baud rate	X			
None Supported	3	BOI	X	X		
	4	Bus-off counter	X	X		
	5	Allocation information	X			
	6	MAC ID switch changed				
	7	Baud rate switch changed				
	8	MAC ID switch value	X			
	9	Baud rate switch value	X			
		DeviceNet Services	Parameter Options			
Services	X	Get_Attributes_All				
	X	Set_Attribute_Single				
None Supported	X	Allocate M/S connection set				
	X	Release M/S connection set				

### III.11 Connection Object Class (Explicit)

Connection Object 0x05							
Object Class	ID	Description	Get	Set	Value Limits		
Attributes      Open	1	Revision					
	2	Max instance					
	X None Supported	3	Number of Instances				
		4	Optional attributes list				
		5	Optional services list				
		6	Max Id of class attributes				
		7	Max Id of instance attributes				
<b>DeviceNet Services</b>			<b>Parameter Options</b>				
Services		Reset					
		Create					
		Delete					
X None Supported		Get Attribute Single					
		Find Next Object instance					
<b>Object Instance</b>	<b>Connection type</b>		<b>Max. connection instances</b>				
	M/S Explicit message		1 Server	Client	1 Total		
	Production trigger(s)		Cyclic	COS	App. trig.		
	Transport type(s)		Server	X	Client		
	Transport class(es)			2	3	X	
Object Class	ID	Description	Get	Set	Value Limits		
Attributes      Open	1	State	X				
	2	Instance type	X				
	3	Transport class trigger	X				
	4	Produced connection ID	X				
	5	Consumed connection ID	X				
	6	Initial comm. characteristics	X				
	7	Produced connection size	X				
	8	Consumed connection size	X				
	9	Expected packet rate	X	X			
	12	Watchdog time-out action	X	X			
	13	Produced connection path len	X				
	14	Produced connection path	X				
	15	Consumed connection path len	X				
	16	Consumed connection path	X				
	17	Production inhibit time					
	<b>DeviceNet Services</b>			<b>Parameter Options</b>			
	Services	X	Reset				
		Delete					
		Apply attributes					
	X	Get Attribute Single					
	X	Set Attribute Single					

## III.12 Connection Object Class (Polled I/O)

Connection Object 0x05							
Object Class	ID	Description	Get	Set	Value Limits		
Attributes      Open	1	Revision					
	2	Max instance					
	X None Supported	3	Number of Instances				
		4	Optional attributes list				
		5	Optional services list				
		6	Max Id of class attributes				
		7	Max Id of instance attributes				
	<b>DeviceNet Services</b>		<b>Parameter Options</b>				
Services	X	Reset					
		Create					
		Delete					
	X None Supported		Get Attribute Single				
		Find Next Object instance					
<b>Object Instance</b>	<b>Connection type</b>		<b>Max. connection instances</b>				
	M/S poll		1 Server	Client	1 Total		
	Production trigger(s)		Cyclic	X	COS	App. trig.	
	Transport type(s)		Server	X		Client	
	Transport class(es)				2	X	
						3	
	<b>ID</b>	<b>Description</b>	<b>Get</b>	<b>Set</b>	<b>Value Limits</b>		
Attributes      Open	1	State	X				
	2	Instance type	X				
	3	Transport class trigger	X				
	4	Produced connection ID	X				
	5	Consumed connection ID	X				
	6	Initial comm. characteristics	X				
	7	Produced connection size	X				
	8	Consumed connection size	X				
	9	Expected packet rate	X	X			
	12	Watchdog time-out action	X	X			
	13	Produced connection path len	X				
	14	Produced connection path	X				
	15	Consumed connection path len	X				
	16	Consumed connection path	X				
	17	Production inhibit time					
		<b>DeviceNet Services</b>		<b>Parameter Options</b>			
	Services	X	Reset				
		Delete					
		Apply attributes					
X		Get Attribute Single					
X		Set Attribute Single					

### III.13 Discrete Input Point Object

Discrete input point Object 0x08						
Object Class	ID	Description	Get	Set	Value Limits	
Attributes      Open	1	Revision	X			
	2	Max instance				
	None Supported	3	Number of Instances			
		4	Optional attributes list			
		5	Optional services list			
		6	Max Id of class attributes			
		7	Max Id of instance attributes			
<b>DeviceNet Services</b>			<b>Parameter Options</b>			
Services		Get_Attributes_All				
None Supported	X	Get_Attribute_Single				
Object Instance	ID	Description	Get	Set	Value Limits	
Attributes      Open	1	Number of attributes				
	2	Attribute list				
None Supported	3	Value	X			
	4	Status				
	5	Off_On_Delay				
	6	On_Off_Delay				
<b>DeviceNet Services</b>			<b>Parameter Options</b>			
Services		Get_Attributes_All				
		Set_Attributes_All				
X None Supported	X	Get_Attribute_Single				
		Set_Attribute_Single				

### III.14 Discrete Output Point Object

Discrete output point Object 0x09						
Object Class	ID	Description	Get	Set	Value Limits	
Attributes      Open	1	Revision				
	2	Max instance				
	X None Supported	3	Number of Instances			
		4	Optional attributes list			
		5	Optional services list			
		6	Max Id of class attributes			
		7	Max Id of instance attributes			
<b>DeviceNet Services</b>			<b>Parameter Options</b>			
Services		Get_Attributes_All				
		Get_Attribute_Single				
X None Supported						
Object Instance	ID	Description	Get	Set	Value Limits	
Attributes      Open	1	Number of attributes				
	2	Attribute list				
None Supported	3	Value	X	X		
	4	Status				
	5	Fault state				
	6	Fault value				
	7	Idle state				
	8	Idle value				
	9	Command				
	10	Flash				
	11	Flash rate				
	12	Object state				
<b>DeviceNet Services</b>			<b>Parameter Options</b>			
Services		Get_Attributes_All				
		Set_Attributes_All				
X None Supported		Get_Attribute_Single				
	X	Set_Attribute_Single				

### III.15 Analog Input Point Object

Analog input point Object 0x0A						
Object Class	ID	Description	Get	Set	Value Limits	
Attributes      Open	1	Revision	X			
	2	Max instance				
	None Supported	3	Number of Instances			
		4	Optional attributes list			
		5	Optional services list			
		6	Max Id of class attributes			
		7	Max Id of instance attributes			
<b>DeviceNet Services</b>			<b>Parameter Options</b>			
Services		Get Attributes All				
None Supported	X	Get Attribute Single				
Object Instance	ID	Description	Get	Set	Value Limits	
Attributes      Open	1	Number of attributes				
	2	Attribute list				
None Supported	3	Value	X			
	4	Status				
	5	Owner vendor ID				
	6	Owner serial number				
	7	Input range				
	8	Value data type				
<b>DeviceNet Services</b>			<b>Parameter Options</b>			
Services		Get Attributes All				
		Set Attributes All				
X None Supported	X	Get Attribute Single				
		Set Attribute Single				

## III.16 Analog Output Point Object

Analog output point Object 0x0B						
Object Class	ID	Description	Get	Set	Value Limits	
Attributes	Open	1	Revision			
		2	Max instance			
X None Supported		3	Number of Instances			
		4	Optional attributes list			
		5	Optional services list			
		6	Max Id of class attributes			
		7	Max Id of instance attributes			
		<b>DeviceNet Services</b>	<b>Parameter Options</b>			
Services		Get_Attributes_All				
		Get_Attribute_Single				
X None Supported						
Object Instance	ID	Description	Get	Set	Value Limits	
Attributes	Open	1	Number of attributes			
		2	Attribute list			
None Supported		3	Value	X	X	$\text{=}(-10000..10000)$
		4	Status			
		5	Owner vendor ID			
		6	Owner serial number			
		7	Output range			
		8	Value data type			
		9	Fault state			
		10	Idle state			
		11	Fault value			
		12	Idle value			
		13	Command			
		14	Object state			
		<b>DeviceNet Services</b>	<b>Parameter Options</b>			
Services		Get_Attributes_All				
		Set_Attributes_All				
X None Supported		Get_Attribute_Single				
	X	Set_Attribute_Single				

## III.17 Parameter Object

Parameter Object 0x0F						
Object Class	ID	Description	Get	Set	Value Limits	
Attributes      Open	1	Revision				
	2	Max instance	X			
	None Supported	3	Number of Instances			
		4	Optional attributes list			
	5	Optional services list				
	6	Max Id of class attributes				
	7	Max Id of instance attributes				
	8	Parameter class descriptor	X			
	9	Configuration assembly instance	X			
	10	Native language				
<b>DeviceNet Services</b>			<b>Parameter Options</b>			
Services		Get_Attributes_All				
		Reset				
X None Supported	X	Get_Attribute_Single				
		Set_Attribute_Single				
		Restore		Save		
Object Instance	ID	Description	Get	Set	Value Limits	
Attributes      Open	1	Parameter value	X	X		
	2	Link path size				
X None Supported	3	Link path	X			
	4	Descriptor	X			
	5	Data type	X			
	6	Data size	X			
	7	Parameter name setting				
	8	Units string				
	9	Help string	X			
	10	Minimum value	X			
	11	Maximum value	X			
	12	Default value				
	13	Scaling multiplier				
	14	Scaling divisor				
	15	Scaling base				
	16	Scaling offset				
	17	Multiplier link				
	18	Divisor link				
	19	Base link				
	20	Offset link				
	21	Decimal precision				
<b>DeviceNet Services</b>			<b>Parameter Options</b>			
Services		Get_Attributes_All				
	X	Get_Attribute_Single				
X None Supported	X	Set_Attribute_Single				

This page was deliberately left blank.

## IV Appendix

### IV.1 Examples

The examples shown in this section are intended to assist you with segments of code to illustrate specific techniques.

#### IV.1.1 Simple Motion Sequence

Before beginning the motion sequence, configure the drive properly so that the drive can be operated properly from the MMI. For this example, we will set the position units so that one revolution equals 1000 counts (PGEARI=1000, PGEARO=1048576) and will move one revolution. Homing is accessed through the parameter object.

DeviceNet Command	Serial Terminal Verification
Set_Single_Attribute (Mode, Position_Mode)	OPMODE 8
Set_Single_Attribute (Enable, True)	READY 1
Set_Single_Attribute (Parameter Object Instance #141 – Homing – Attribute #1- Value, 1)	TRJSTAT bit 0x40000 is set (homed)
Set_Single_Attribute (Increment Mode Flag, Increment)	O_C bit 0x01 is set (incremental move)
Set_Single_Attribute (Target_Position, 1000)	O_P 1000 O_C 12289 = 0x3001 (SI units, incremental move)
Set_Single_Attribute (Target_Velocity, 1000)	O_V 1000
Set_Single_Attribute (Trajectory_Start, True)	Motor will rotate one revolution
While (Get_Single_Attribute (On_Target_Position)=Not_True); * This will wait until move is complete. (repeat for next index)	
Set_Single_Attribute (Trajectory_Start, True)	
While (Get_Single_Attribute (On_Target_Position)=Not_True); * This will wait until move is complete.	

### IV.2 Allen Bradley SLC5/0x

This part describes the interface between Allen Bradley's SLC-5/0X Series PLC's DeviceNet Scanners (1747-SDN/B) and SERVOSTAR 600.

There are two methods of communication, Explicit Messaging and Polled I/O.

#### IV.2.1 Polled I/O

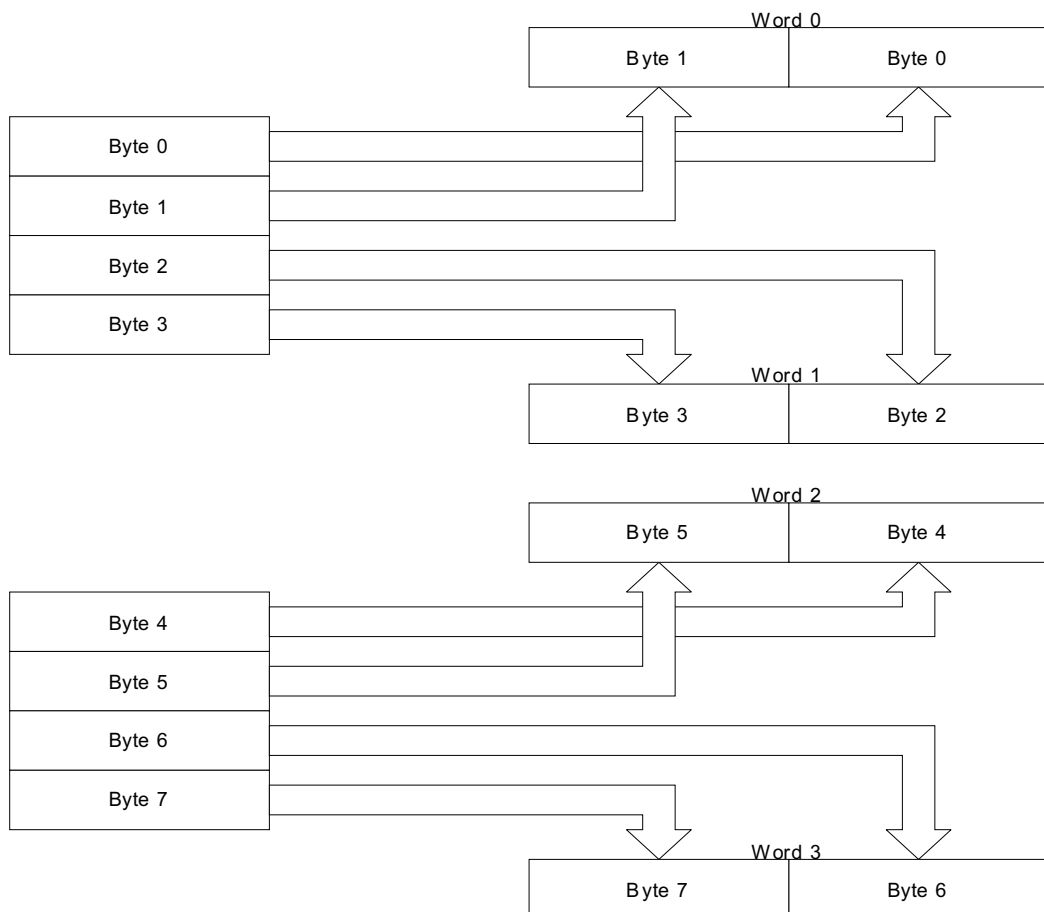
In order to communicate with SERVOSTAR 600 drives via Polled I/O, the drive must first be mapped into the PLC's scan list (mapping is not described in this document). This is the fastest way to send commands and receive the status from the drive.

## IV.2.2 Polled Amplifier Input (PLC Output)

Input command assemblies are used to issue polled I/O commands from the PLC to the amplifier. Each drive supports eight bytes of input command (output from the PLC). The format of this assembly is shown below (refer to Command Assembly 1: Target Position for further definition).

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Enable	Reg Arm	Hard Stop	Smooth Stop	Dir (Vel Mode)	Incremental	Start Block	Trajectory Start / Load Data
1	Block # To Execute							
2	0	0	1	0	Input Command Assembly Type			
3	0	0	1	0	Output Response Assembly Type			
4	Attribute Lower Word Lower Byte							
5	Attribute Lower Word Upper Byte							
6	Attribute Upper Word Lower Byte							
7	Attribute Upper Word Upper Byte							

Now the tricky part is mapping the above eight bytes into the output memory of the PLC. See your DeviceNet Scanner manual for instructions on setting up a scan list and mapping the memory with RSNetworkx. Ensure that you are using the correct slot number. Since the PLC memory is sixteen bits wide, and the DeviceNet data is eight bits wide, the following data mapping must be used:



### IV.2.3 Command Assembly Example

Set the scanner's scan rate to about 100ms. The actual application scan rate can be "fine tuned" at a later time.

To perform a simple move, configure the drive, then set target velocity and target position. The move will begin when the target position is set.

The following configuration is assumed for this example:

```
PGEARI=1000
PGEARO=1048576
OPMODE=8
```

First, set the target velocity to 1000 (000003E8 hex) feedback counts.



*Bytes are "built" backwards in the PLC*

*All eight bytes of data are ignored unless a valid command assembly type is specified in byte 2 (valid command assembly types are 0 through 5).*

Change the PLC output command assembly (Scanner output data) to the following:

Byte	Function	Data Value (hex)
0	Enable Start Trajectory / Load Data	81
1	Block Number	00
2	Axis Instance Command Assembly 2 – Target Velocity	22
3	Axis Instance Response Assembly 0 – no response	20
4	Target Position – Lower Word Lower Byte	E8
5	Target Position – Lower Word Upper Byte	03
6	Target Position – Upper Word Lower Byte	00
7	Target Position – Upper Word Upper Byte	00

Mapping this to the PLC memory should look similar to the table shown below:

Word Number	Data Value (hex)
0	0081
1	2022
2	03E8
3	0000

Now, set the target position to 1000 (000003E8 hex) feedback counts to begin the move.



**Data is only loaded when Start Trajectory/Load Data transitions to high. Set the bit low between commands.**

Toggle bit 0 in word 0 (Start Trajectory / Load Data) to prepare for the next command.

Change the PLC output command assembly (Scanner output data) to the following:

Byte	Function	Data Value (hex)
0	Enable Start Trajectory / Load Data	81
1	Block Number	00
2	Axis Instance Command Assembly 1 – Target Position	21
3	Axis Instance Response Assembly 1 – Actual Position	21
4	Target Position – Lower Word Lower Byte	E8
5	Target Position – Lower Word Upper Byte	03
6	Target Position – Upper Word Lower Byte	00
7	Target Position – Upper Word Upper Byte	00

Now, mapping this to the PLC memory should look similar to the table shown below;

Word Number	Data Value (hex)
0	0081
1	2121
2	03E8
3	0000

This should cause the motor to turn 1000 feedback counts (1 rev if resolution is 1000) in the clockwise direction.

#### IV.2.4 Polled Amplifier Output (PLC Input)

Output response assemblies are used to send polled I/O responses to the PLC from the drive. Each drive supports eight bytes of output response (input to the PLC). The format of this assembly is shown below (refer to Response Assembly 1 – Actual Position for further definition).

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Enable State	Reg Level	Home Level	Current Dir	General Fault	On Target Position	Block in Execution	Trajectory in Progress
1	Block # Executing							
2	Load Complete	Block Fault	Following Error	Negative Limit	Positive Limit	CCW Limit	CW Limit	Fault Input Active
3	0	0	1	0	Output Response Assembly Type			
4	Attribute Lower Word Lower Byte							
5	Attribute Lower Word Upper Byte							
6	Attribute Upper Word Lower Byte							
7	Attribute Upper Word Upper Byte							

## IV.2.5 Response Assembly Example

At the completion of the move shown in the previous example, the response assembly data should be as shown below:

Byte	Function	Data Value (hex)
0	Enable Current Direction (depends on overshoot) On Target Position	84 or 94
1	Block Executing	00
2	Load Complete No errors/warnings	80
3	Axis Instance Response Assembly 1 – Actual Position	21
4	Actual Position – Lower Word Lower Byte	E8
5	Target Position – Lower Word Upper Byte	03
6	Target Position – Upper Word Lower Byte	00
7	Target Position – Upper Word Upper Byte	00

Now, mapping this to the PLC memory should look similar to the table shown below:

Word Number	Data Value (hex)
0	0094
1	2180
2	03E8
3	0000

## IV.2.6 Explicit Messaging

With the Allen Bradley PLC's, sometimes it takes up to 2 seconds for the PLC to process an explicit message command. This delay is not controlled by the SERVOSTAR 600 in any way, as the SERVOSTAR 600 responds to the explicit message command in less than 5 ms.

M0 file area words 224 through 255 contain a 32-word to execute an Explicit Message Request write.

M1 file area words 224 through 255 contain a 32-word to execute an Explicit Message Response read.

Shown below is the memory structure (in the PLC & the M0 file) for explicit messaging commands.

		PLC Memory	M0 Memory
TXID	COMMAND	WORD 0	WORD 224
PORT	SIZE (in bytes)	WORD 1	WORD 225
SERVICE	MAC ID	WORD 2	WORD 226
CLASS		WORD 3	WORD 227
INSTANCE		WORD 4	WORD 228
ATTRIBUTE		WORD 5	WORD 229
LOWER DATA WORD		WORD 6	WORD 230
UPPER DATA WORD		WORD 7	WORD 231

**TXID** Transmit ID. This value must be unique for each explicit message sent.

**COMMAND** The command specified for this block of data.

01 – Send Explicit Message.

04 – Clear response buffer.

**PORT** 0 – Channel A. (typical choice)

1 – Channel B.

**SIZE** Size in bytes of all data after MAC ID.

**SERVICE** 0E (hex) – Get.

10 (hex) – Set.

**MAC ID** The DeviceNet ID of the SERVOSTAR 600 as specified by the two MACID switches.



*If the switches are set to 25 (the switches read as decimal), then this value is set to 19 hex.*

**CLASS** DeviceNet class to access. Examples:

Parameter Object – 15 (0F hex).

Position Controller Supervisor – 36 (24 hex).

Position Controller Object – 37 (25 hex).

**INSTANCE** Always 1 (01 hex) for objects 36, 37.

Parameter number for Parameter Object.

Port number for Analog and Digital I/O.

**ATTRIBUTE** The attribute number of the attribute being accessed (set or get).

Shown below is the memory structure (in the PLC & the M1 file) for explicit messaging responses:

		PLC Memory	M0 Memory
TXID	STATUS	WORD 0	WORD 224
PORT	SIZE (in bytes)	WORD 1	WORD 225
SERVICE	MAC ID	WORD 2	WORD 226
DATA		WORD 3 - 31	WORD 227 - 255

**TXID** Transmit ID. This value must be unique for each explicit message sent.

Matches the TXID in the command message.

<b>STATUS</b>	The Status specified for this block of data. 0 – Ignore block (empty) 1 – Transaction completed successfully 2-15 – Scanner error (see Scanner documentation)
<b>PORT</b>	0 – Channel A. (typical choice) 1 – Channel B.
<b>SIZE</b>	Size in bytes of all data after MAC ID.
<b>SERVICE</b>	Echoes the service code from the command message, setting the upper bit for a response. 1E (hex) – Get. 90 (hex) – Set. 94 (hex) – DeviceNet Error. Error code follows.
<b>MAC ID</b>	The DeviceNet ID of the amplifier as specified by the two MACID switches.
<b>DATA</b>	Response data (length in bytes given by SIZE)

### IV.2.6.1 Explicit Message Sequence of Events

- Put Explicit Message Request data into a file in the SLC-500 and using the file copy instruction (COP) in the SLC-500, copy the data to the M0 file, words 224 through 255. Minimum data size is 6 words for an Explicit Message Request and maximum size is 32 words.
- Wait until bit 15 of 1747-SDN Module Status Register (typically mapped to word 0 of the input file) goes to a 1, which tells you that a response has been received.
- Using the file copy instruction (COP) in the SLC-500, copy the data from M1 file words 224 through 255 into a file in the SLC-500, size of 32 words. This file will contain the Explicit Message Response. Test TXID field of this file to make sure it matches Explicit Message Request TXID value. Test the response code for an Allen Bradley error code. Test the service code for a DeviceNet error code (94 hex).
- Using the Move instruction (MOV) in the SLC-500, copy a word from a file into M0 file word 224. The upper byte of this word should be the TXID of the Explicit Message just executed and the lower byte should contain a 4, which is the command to clear out the response buffer. After this move is executed, bit 15 of the 1747-SDN Module Status Register should go to a 0 and the next Explicit Message can be executed starting at step 1. This removed the transaction from the scanner's memory and frees the TXID to be used again.

#### IV.2.6.1.1 Example

In this example, use the Set service (10 hex) to set the Target Position attribute (06 hex) to 65536 (00010000 hex) of the Position Controller Class 37 (25 hex) of MACID 25 (19 hex). With SERVOSTAR 600, the axis instance is always 1 (01 hex).

TXID	COMMAND	0101
PORT	SIZE (in bytes)	000A
SERVICE	MAC ID	1019
CLASS		0025
INSTANCE		0001
ATTRIBUTE		0006
LOWER DATA WORD		0000
UPPER DATA WORD		0001

If successful, the response packet would look as follows:

TXID	COMMAND	0101
PORT	SIZE (in bytes)	0002
SERVICE	MAC ID	9019
STATUS		0000

## IV.2.7 Example Program

The example program S600\_Test\_Program.RSS (available from Danaher – contact the Customer Support Network) uses explicit messaging to configure and home the drive and polled messaging to initiate a move. The program also provides example subroutines that may be easily used in other programs for transmitting explicit and polled messages. Note that the subroutines do not currently do any error checking on response codes.

The program assumes the SERVOSTAR 600 has MAC ID set to 01 (switches on the front of the option card set to 0 and 1). The drive must be mapped into the DeviceNet Scanner input file at word 1 and the output file at word 1.

The assumed drive configuration for this example is:

```
PGEARI=1000
PGEARO=1048576
OPMODE=8
```

### IV.2.7.1 Running the Test Program

Numbers correspond to ladder rungs in the MAIN ladder U:2.

1. To load a command assembly from B3 (static command data) into the scanner's polled output, set B3:0 to the index of the assembly you wish to send (see U:3 for the available assemblies).
2. To perform data handshaking over polled messaging, load the desired command assembly (see 1), then set N47:0 = 1 to call U:40.
3. To do a hard stop set B9:0/0 = 1
4. To do a smooth stop set B9:0/1 = 1
5. To start a motion task (like MOVE x), put the task number in the high byte of N10:0 and set N15:0 = 1 to call U:5.
6. To send an explicit message and receive a response, load the explicit message into N11 (explicit message output buffer) and set N13:0 = 1 to call U:6. The response will be loaded into N12 (explicit message input buffer) and N13:0 will be reset to 0 when the routine is complete.
7. To send an explicit message transaction to the DN scanner (for testing), load the message into N11 and set B9:0/2 = 1.
8. To receive an explicit message transaction from the DN scanner into N12 (for testing), set B9:0/4 = 1.
9. To execute the complete test program in U:7 (home and move), set N14:0 = 1.

### IV.2.7.2 Subroutines

U:2	MAIN
U:3	Initialize memory (static command assemblies)
U:4	Write assembly to output
U:5	Start Block (Execute a motion task - task # stored in N:10)
U:6	Send Explicit Message (transmits the command message in N11 and stores the response in N12)
U:7	Move (NREF=0, MH, ACC=10, DEC=10, O_V=1000, O_P=1000 and MOVE 0)
U:40	Start Trajectory / Load Data - handshaking for sending a Polled msg

### IV.2.7.3 Data Mapping

B3:0	Index of user's command assembly selection
B3:1	Index of previous command assembly selection
B3:2	Temp var for calculation of command assembly address
B3:10-73	Static command assemblies
B9	Flags for user to control the drive
B9:0/0	Do a Hard Stop
B9:0/1	Do a Smooth Stop
B9:0/2	Set to send the Exp Msg in N11
B9:0/3	One-shot bit for previous bit
B9:0/4	Set to receive an Exp Msg in N12
B9:0/5	One-shot bit for previous bit
N10:0	High byte is the block number to execute with B9:0/2 and U:5
N11	Explicit Message output buffer. Load a message here, then set N13:0 = 1 to run U:6 and send the message. For the format of this buffer, see the Explicit Messaging section above.
N12	Explicit Message input buffer. Loaded with response message by U:6. For the format of this buffer, see the Explicit Messaging section above.
N13	State variable for U:6
N14	State variable for U:7
B43:0	Temporary variable used for calculations in Start Trajectory sequence U:40.
N47:0	State variable for Start Trajectory / Load Data sequence in U:40

### IV.2.7.4 Output File Mapping

O:1.0	DN Scanner Control Word
O:1.0/0	Enable DN Scanner Outputs
O:1.1	DN command word 0 (control flags, block number)
O:1.2	DN command word 1 (command, response)
O:1.3	DN command word 2 (data LSW)
O:1.4	DN command word 3 (data MSW)
O:1.5-31	Unmapped area of DN Scanner (available for other devices)

### IV.2.7.5 Input File Mapping

I:1	DN status word
I:1/15	Explicit Msg Response is available
I:1/0-7	Polled Msg Byte 0 (status byte 1)
I:1/8-15	Polled Msg Byte 1 (block in execution)
I:1/32-39	Polled Msg Byte 2 (status byte 2)
I:1/40-47	Polled Msg Byte 3 (axis/response type)
I:1/48-79	Polled Msg Bytes 4-8 (data, LSB first)

### IV.3 Baud Rate Switch Settings

For SERVOSTAR 600 drives, the baud rate switch may be set to 0 (125 KBaud), 1 (250 KBaud) or 2 (500 KBaud). If the switch is set to a value greater than 2, the baud rate is configurable through the terminal parameter DNBAUD and through DeviceNet. If the switch is set from 0 to 2, the baud rate cannot be controlled with DNBAUD or DeviceNet.

### IV.4 MAC ID Switch Configuration

Values 0 to 63 are valid. If the switches are set to a value greater than 63, the MAC ID is configurable through the terminal parameter DNMACID and through DeviceNet. If the switches are set from 0 to 63, the MAC ID cannot be controlled with DNMACID or DeviceNet.

### IV.5 Default Input/Output Configuration

For the SERVOSTAR 600, the following input configuration is applicable:

O1MODE=23 (DeviceNet control of digital output 1)  
O2MODE=23 (DeviceNet control of digital output 2)  
ANOUT1=6 (DeviceNet control of analog output 1)  
ANOUT1=6 (DeviceNet control of analog output 1)

## IV.6 Index

<b>A</b>	Abbreviations . . . . .	6	<b>P</b>	Parameter object . . . . .	59
	Additional documentation . . . . .	7		Permitted use . . . . .	7
	Analog input point . . . . .	57		Polled I/O . . . . .	41
	Analog output point . . . . .	58		Position controller . . . . .	15
<b>B</b>	Basic features . . . . .	8		Positioning functions . . . . .	8
	Bus cable . . . . .	9	<b>S</b>	Setup . . . . .	14
<b>C</b>	Cable length . . . . .	9		Setup functions . . . . .	8
	Communication faults . . . . .	10		Station address . . . . .	13
	Connection methods . . . . .	13		Status LED . . . . .	11
<b>D</b>	Data transfer functions . . . . .	8		Supervisor attributes . . . . .	18
	Data types . . . . .	17		Symbols . . . . .	6
	Discrete input point . . . . .	55		System requirements . . . . .	8
	Discrete output point . . . . .	56	<b>T</b>	Transmission procedure . . . . .	8
<b>E</b>	Error codes . . . . .	18		Transmission rate . . . . .	8
	Examples . . . . .	61		Transmission rate setting . . . . .	14
<b>I</b>	I/O response . . . . .	46	<b>U</b>	Use as directed . . . . .	7
	Installation . . . . .	13			
<b>L</b>	LED . . . . .	11			
<b>O</b>	Object class				
	Block sequencer . . . . .	28			
	Command block . . . . .	30			
	Connection (explicit) . . . . .	53			
	Connection (polled I/O) . . . . .	54			
	DeviceNet . . . . .	52			
	Identity . . . . .	51			
	Message router . . . . .	52			
	Position controller . . . . .	20			
	Position controller supervisor . . . . .	18			
	Object model . . . . .	15			

## Sales and Service

We are committed to quality customer service. In order to serve in the most effective way, please contact your local sales representative for assistance. If you are unaware of your local sales representative, please contact us.

### *Europe*

Visit the European Danaher Motion web site at [www.DanaherMotion.de](http://www.DanaherMotion.de) for Setup Software upgrades, application notes, technical publications and the most recent version of our product manuals.

#### **Danaher Motion Customer Support - Europe**

Internet	<a href="http://www.DanaherMotion.de">www.DanaherMotion.de</a>
E-Mail	<a href="mailto:info@danaher-motion.de">info@danaher-motion.de</a>
Phone.:	+49(0)203 - 99 79 - 0
Fax:	+49(0)203 - 99 79 - 155

### *North America*

Visit the North American Danaher Motion web site at [www.DanaherMotion.com](http://www.DanaherMotion.com) for Setup Software upgrades, application notes, technical publications and the most recent version of our product manuals.

#### **Danaher Motion Customer Support - Radford**

Internet	<a href="http://www.DanaherMotion.com">www.DanaherMotion.com</a>
E-Mail	<a href="mailto:servo@kollmorgen.com">servo@kollmorgen.com</a>
Phone:	(800) 777-3786 or (815) 226-3100
Fax:	(540) 731-5641