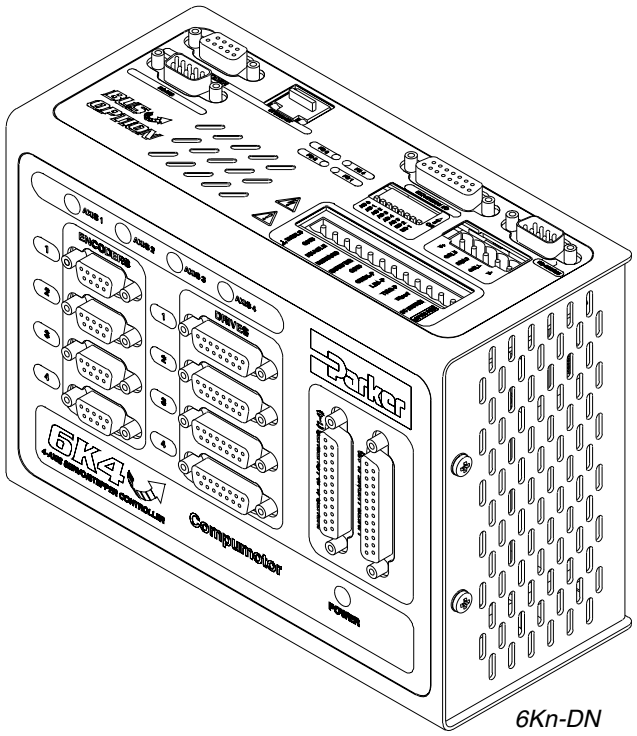
Parker Automation

p/n 88-019049-01 B

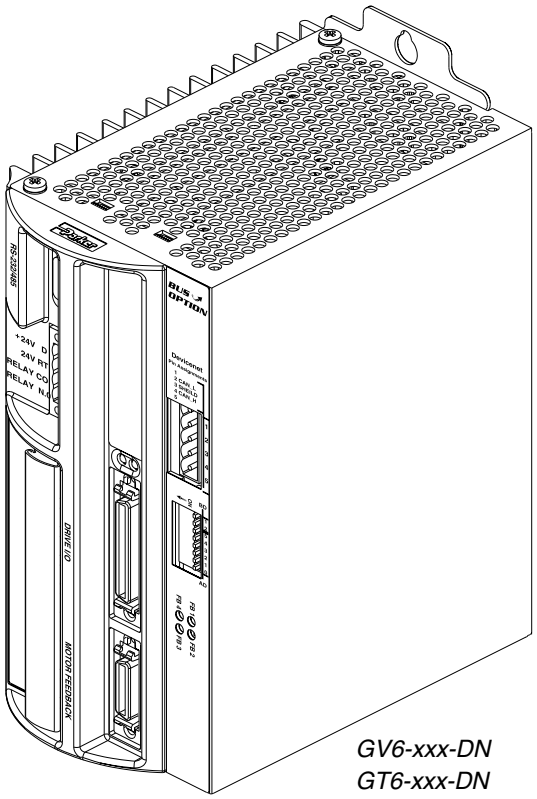
DeviceNet Guide

for 6K and Gemini products

Effective: August 1, 2002



6Kn-DN



GV6-xxx-DN
GT6-xxx-DN

IMPORTANT

User Information

WARNING

Parker Automation products are used to control electrical and mechanical components of motion control systems. You should test your motion system for safety under all potential conditions. Failure to do so can result in damage to equipment and/or serious injury to personnel.

Technical Assistance ⇨ *Contact your local automation technology center (ATC) or distributor,* ***or ...***

North America and Asia:
Compumotor Division of Parker Hannifin
5500 Business Park Drive
Rohnert Park, CA 94928
Telephone: (800) 358-9070 or
(707) 584-7558
Fax: (707) 584-3793
e-mail: tech_help@cmotor.com
Website: www.compumotor.com

Europe:
Parker Hannifin plc
Electromechanical Division – Digiplan
21 Balena Close
Poole, Dorset
England BH17 7DX
Telephone: +44 (0)1202-699000
Fax: +44 (0)1202-695750
e-mail: sales.digiplan@parker.com
e-mail: support.digiplan@parker.com
Website: www.parker-emd.com

Germany:
Parker Hannifin GmbH
Electromechanical
Division – Hauser
P. O. Box: 77607-1720
Robert-Bosch-Str. 22
D-77656 Offenburg, Germany
Telephone: +49 (0)781 509-0
Fax: +49 (0)781 509-176
e-mail: sales.hauser@parker.com
Website: www.parker-emd.com

Italy:
Parker Hannifin SpA
Divisione SBC
20092 Cinisello Balsamo
Milan,
Italy Via Gounod, 1
Telephone: +39 02 6601 2478
Fax: +39 02 6601 2808
e-mail: sales.sbc@parker.com
Website: www.parker-emd.com

Parker
Automation

Technical Support Email
tech_help@cmotor.com

ABOUT THIS GUIDE

Change Summary – Revision B

August 1, 2002

This document, 88-019049-01**B**, supersedes 88-019049-01**A**. Changes and corrections are noted below.

Topic	Description
Document clarifications and corrections	• DeviceNet Overview: Added ODVA software self-certification notice – see page 22. • LED Status indicators: Enhanced description of LEDs – see page 24. • Node Address: Rearranged table for clarity – see page 26. • Object Model: added Acknowledge handler and renumbered position controller and position controller supervisor objects – see page 27. • Added Identity object, DeviceNet Object, Assembly Object, Connection Object and Acknowldege Handler Object descriptions. See pages 30 – 34. • Position Controller Supervisor object: Added class attribute description, added Object Instance Attributes 100-107 (see page 35). • Position Controller Object: Removed attribute 1, added attributes 106-112. Renamed attribute 26 to 113. Modified attributes 13, 50, 51; – see page 37. • Polled I/O: Replaced Valid Data with Reg Arm. – see page 40. • Polled I/O: Added more explanation how to run stored program – see page 42. • Polled I/O: Added command assembly 0x0C – see page 44. • Polled I/O: Removed Response assembly 0x00 – see page 47. • Added Command/Response Error 0x14 – see page 49. • Operating Modes: Enhanced examples - see pages 52 – 58. • Added more description to [FBS] command - see page 62. • Added CE Compliance information for Gemini products – see page 66. • Added dimensions for Gemini drives (page 68) and panel layout dimensions for Gemini drives (page 71).

Purpose of This Guide

This document is designed to help you implement the DeviceNet features provided in your 6K and Gemini series products, as ordered with the DeviceNet option. This publication addresses only the installation and programming tasks for the DeviceNet features. For all other installation and programming instructions, refer to:

- *6K Series Hardware Installation Guide*, part number 88-017547-01
- *6K Series Command Reference*, part number 88-017136-01
- *6K Series Programmer's Guide*, part number 88-017137-01
- *Gemini GV6 Hardware Installation Guide*, part number 88-018364-01
- *Gemini GT6 Hardware Installation Guide*, part number 88-018374-01
- *Gemini Series Programmer's Guide*, part number 88-017778-01
- Refer also to the online help system in Motion Planner

What You Should Know

To install and troubleshoot your 6K and Gemini series products with the DeviceNet option, you should have a fundamental understanding of:

- Electronics concepts such as voltage, current, switches.
- Implementing and maintaining a given fieldbus network.
- Mechanical motion control concepts, such as inertia, torque, velocity, distance, force.
- Ethernet or serial (RS-232 or RS-485) communication, depending on which communication protocol you are using.

⚠ WARNINGS ⚠

The 6K and Gemini products are used to control your system's electrical and mechanical components. Therefore, you should test your system for safety under all potential conditions. Failure to do so can result in damage to equipment and/or serious injury to personnel.

ALWAYS REMOVE POWER from the product before connecting any electrical devices (e.g., fieldbus connections, drives, encoders, I/O bricks, inputs, outputs, etc.).

CHAPTER ONE

1 Implementing 6K DeviceNet

IN THIS CHAPTER

DeviceNet Overview

6Kn-DN

The DeviceNet option allows a 6K controller to be controlled via a DeviceNet master, utilizing the DeviceNet protocol for robust data exchange. The 6K functions as a “Communications Adapter” on the DeviceNet network, allowing the user's application to fully define the data exchanged with a DeviceNet master.

Cabling is not provided by Compumotor for implementing a given fieldbus topology.

Technical Assistance

Technical questions regarding DeviceNet should be addressed to your local DeviceNet User Group. An address list is available on the DeviceNet Internet site at www.odva.org.

For support with 6K specific questions, contact Applications at 800-358-9070 or e-mail us at tech_help@cmotor.com.

Implementation Process

DeviceNet Master (user defined):

Use the provided CMTR1.EDS file. Do not modify.

Configure communication baud rate.

Configure data packet size

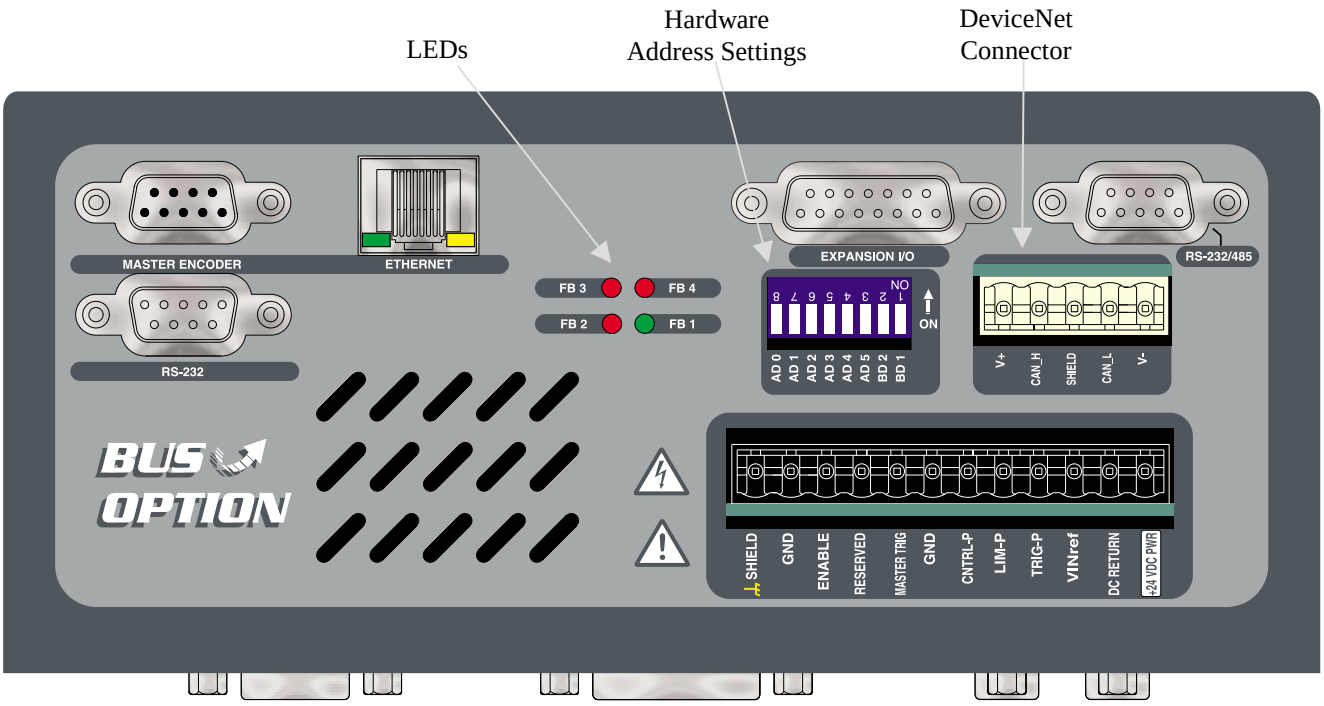
6K Controller:

1. Install terminating resistors as needed (see page 4).
2. Launch Motion Planner (CD-ROM is provided in your ship kit).
3. Establish a direct communication link (serial) with the 6K. Refer to the *6K Series Hardware Installation Guide* for connection instructions.
4. Configure node address (see `FBADDR` on page 14 or use the hardware method on page 4). Setting the address via hardware overrides the software setting.
5. Configure baud rate (see `FBBAUD` on page 14 or use the hardware method on page 5). Setting the baud rate via hardware overrides the software setting.
6. Configure data packet size (`FBSIZE` must match master configuration). Refer to step 3 in the DeviceNet Master implementation process above.
7. Reset the 6K controller to initialize DeviceNet card.
8. Write user code using `VARB1-VARB8` (depending on data packet size) for sending data from the 6K controller to the DeviceNet master.
9. Write user code to read data from `VARB9-VARB16` (depending on data packet size) for receiving data from DeviceNet master to 6K controller.

For more information, refer to the Programming Scenario on page 19.

EDS File

Each device in a DeviceNet network is associated with an EDS file, containing all necessary information about the device. The latest version of the 6K EDS file (CMTR1.EDS) can be downloaded from www.compumotor.com.



LED Status Indicators

Bicolor LED indicators are provided on the DeviceNet option card. Refer to the following table for troubleshooting information provided by these LEDs.

LED	Steady	Flash	Function	Status *
FB1	--	--	Not used	--
FB2	Off		Not powered/not online	--
	Green		Network link ok	FBS bit #4 = 1
	Red		Network critical link failure	FBS bit #5 = 1
		Green	1 flash/second – Network link not connected	FBS bit #6 = 1
		Red	1 flash/second – Network connection timeout	FBS bit #7 = 1
FB3	Off		No power	--
	Green		Module device operational	FBS bit #8 = 1
	Red		Module unrecoverable fault	FBS bit #9 = 1
		Green	1 flash/second – Module in standby	FBS bit #10 = 1
		Red	1 flash/second – Module minor fault	FBS bit #11 = 1
FB4	--	--	Not used	--

* To check status, execute the TFBS command (bit status report) or the TFBSF command (full text status report) in the terminal emulator. You can also use the FBS operator to assign or compare one or more status bits (e.g., use in an IF expression, assign to VARB variable, etc.). Refer to the TFBS command description on page 17.

DeviceNet Connector Pin Out

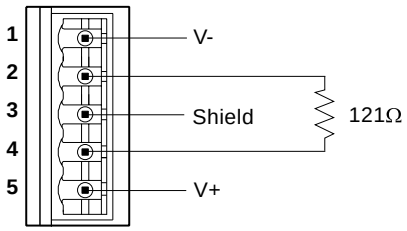
The following table gives the pin out for the DeviceNet connector. Industry standard DeviceNet connections are used.

Pin	Name	Function
1	V-	DC Return
2	CAN_L	CANBUS LOW
3	SHIELD	Protective Earth
4	CAN_H	CANBUS HIGH
5	V+	+24 VDC Power *

* 30mA in standby and 100mA in rush

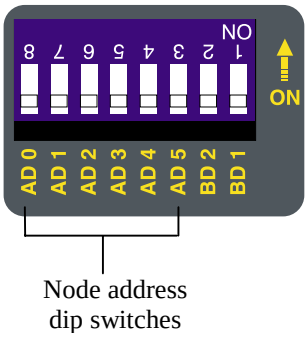
Termination

If the 6K controller is used as the first or last node in a network, a termination resistor must be used. Compumotor recommends a resistor value of 121 ohms. Connect the resistor as shown below.



Node Address

To configure node address via hardware, dip switches are provided to set a node address of 0-63. Setting the dip switches to 0xFF (all ON), enables software configuration of node address (see FBADDR on page 14).



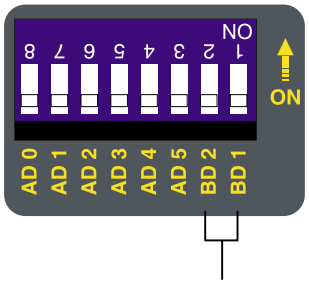
Address	AD0	AD1	AD2	AD3	AD4	AD5
0	OFF	OFF	OFF	OFF	OFF	OFF
1	ON	OFF	OFF	OFF	OFF	OFF
2	OFF	ON	OFF	OFF	OFF	OFF
3	ON	ON	OFF	OFF	OFF	OFF
…						
62	ON	ON	ON	ON	ON	OFF
63	ON	ON	ON	ON	ON	ON

Example:

Switch AD0-AD5 = ON-ON-ON-OFF-ON-OFF, node address is 23
Switch AD0-AD5 and BD1-BD2 = all ON, node address determined by FBADDR

Baud Rate

To configure baudrate via hardware, dip switches are provided to set a baudrate of 125, 250, or 500kb. Setting the dip switches to 0xFF (all ON), enables software configuration of baudrate (see FBBAUD on page 14).



Baudrate
dip switches

Baudrate (Bit/sec)	BD1	BD2
125k	OFF	OFF
250k	OFF	ON
500k	ON	OFF
Reserved	ON	ON

Example**:**

Switch BD1-BD2 = ON-OFF, baudrate 500kbit/s
Switch AD0-AD5 and BD1-BD2 = all ON, baudrate determined by FBBAUD

Data Format

The 6K maps its data to the DeviceNet master in the following manner (low to high address):

Data Out	VARB1	VARB2	VARB3	VARB4	VARB5	VARB6	VARB7	VARB8
Data In	VARB9	VARB10	VARB11	VARB12	VARB13	VARB14	VARB15	VARB16

Binary variables within the 6K programming language follow an unconventional format for bit assignment: bit 1 is the left-most bit and bit 32 is the right-most bit. When binary variables are exchanged with a DeviceNet fieldbus master, bit 1 corresponds to the right-most bit, and bit 32 corresponds to the left-most bit.

Example:

```
VARB1=h12345678                 ;assign hex value to binary variable 1
                                ;DeviceNet Master receives 0x87654321

VARI1=4PE                       ;assume encoder position is +230
VARB1=VCVT(VARI1)               ;convert integer variable to binary
                                ;DeviceNet Master receives 0x0000 00E6
```

Implementing Data Exchange

It is up to the user's 6000 program to facilitate handshaking between the DeviceNet master and the motion controller. There is no built-in handshaking or data synchronization performed by the motion controller.

To implement mailbox messaging (handshaking) between the 6K controller and a DeviceNet master, you must set aside 2-bits/message within `VARB1-16`. One bit is used to acknowledge reading a message, a second bit is used notify the recipient a new message is available.

A message is user defined, but could be used to control motion on a particular axis, update a task, update I/O, control a set of axes from a single message, or report motion status.

For example, you would like to send a message from the DeviceNet master to the 6K controller, and have the 6K controller send a response message to the master. The DeviceNet master will use `VARB9` bits 1 and 2, and the 6K controller will use `VARB1` bits 1 and 2.

To send a mailbox message to the DeviceNet master:

1. Make sure `VARB1.1` is equal to `VARB9.1`, no unprocessed messages.
2. Place the message in `VARB2` through `VARB8`.
3. Toggle `VARB1.1` to indicate new message is available. `VARB1.1` is now not equal to `VARB9.1`.

To receive a message from the DeviceNet master:

1. Make sure `VARB1.2` is not equal to `VARB9.2`, new message available.
2. Read the message from `VARB10-VARB16`.
3. Toggle `VARB1.2` to acknowledge reading the message. `VARB1.2` is now equal to `VARB9.2`.

The same operation would be repeated on the master side, except bits 1 and 2 would be reversed. An application scenario using mailbox messaging is provided on page 19.

Handling a DeviceNet Fault

If a DeviceNet fault (Option card fault) occurs, the event causing the fault can be determined by checking the Fieldbus Status bit values (see FBS on page 15).

- If error bit #19 is disabled (ERROR.19-0), the controller performs a kill all when a DeviceNet fault occurs.
- If error-checking bit #19 is set (ERROR.19-1), the controller performs a kill all and error status bit #19 is set (reported with ER, TER, and TERF). If an error program is assigned with the ERRORP command, the 6K controller branches (GOTO) to the program.

On power-up or out of reset, the events that can generate a DeviceNet fault are ignored. This allows the DeviceNet master time to commission individual nodes without causing the 6K controller to fault. Error bit #19 is edge sensitive to fault conditions.

To recover from an ER.19 fault, resolve the cause (see FBS command on page 15) or reset the controller. To acknowledge the fault condition, issue the ERROR.19-0 command and then the ERROR.19-1 command. For an example, see the application scenario on page 19.

Affected Commands and Features

When the DeviceNet option is installed, the following 6K commands and features are affected:

- Binary variables are affected by updates performed over the DeviceNet network. See FBSIZE on page 16 for the exact binary variables affected by the DeviceNet network.
- VARCLR will have no affect on binary variables assigned to the DeviceNet network.
- A user's application will not be permitted to write to VARB9-16. If you attempt to change the state of VARB9-16, the controller will respond with an error message "VARB USED BY OPTION CARD" and the VARB command will not be executed; however command processing will continue.
- A new bit definition for ERROR (bit #19) has been added for supporting the option card.
- A new function (option "I") has been added to the OUTFNC command, to support detection of ERROR bit #19 being set.
- All commands preceded by FB or TFB (e.g., FBADDR, TFBS, etc.) will be enabled when a DeviceNet card is detected, and disabled when no DeviceNet card is present or enabled.
- Ethernet will be disabled on the 6K when a DeviceNet card is enabled (OPTEN1).

DeviceNet Objects

Connections Supported

Objects which are Implemented

Identity Object, Class 0x01

Class Attributes (0)

#	Attribute Name	Service Name	Description	Default,Min,Max	Data Type
1	Revision	Get_Attribute_Single	Revision of identity object	1,1,65535	UINT

Instance Attributes (1)

#	Attribute Name	Service Name	Description	Default,Min,Max	Data Type
1	Vendor Id	Get_Attribute_Single	Parker Hannifin/Compumotor Division	620	UINT
2	Device Type	Get_Attribute_Single	Communications Adapter	12	UINT
3	Product Code	Get_Attribute_Single	6K Controller	1	UINT
4	Revision	Get_Attribute_Single	Consists of major.minor	{1,1}, {1,1}, {1,1}	Array of USINT
5	Status	Get_Attribute_Single	Represents the current status of the entire device.	0,0,255	WORD
6	Serial Number	Get_Attribute_Single	Used in conjunction with the Vendor ID to form a unique identifier for each device on DeviceNet.	n/a	UDINT
7	Product Name	Get_Attribute_Single	Short description of the product represented by the Product Code.	“6K Controller”	SHORT_ STRING
9	Config. Consist. Value	Get_Attribute_Single	Contents identify configuration of device.	n/a	UINT

Message Router Object, Class 0x02

Class Attributes (0)

#	Attribute Name	Service Name	Description	Default,Min,Max	Data Type
1	Revision	Get_Attribute_Single	Revision of message router object.	1,1,65535	UINT

DeviceNet Object, Class 0x03

Class Attributes (0)

#	Attribute Name	Service Name	Description	Default,Min,Max	Data Type
1	Revision	Get_Attribute_Single	Revision of identity object.	2,1,65535	UINT

Instance Attributes (1)

#	Attribute Name	Service Name	Description	Default,Min,Max	Data Type
1	MAC_ID	Get_Attribute_Single	Node Address	1*,0,63	USINT
2	Baud Rate	Get_Attribute_Single	Indicates the selected baud rate. Values are: 00 - 125k 01 - 250k 10 - 500k	2*,0,2	USINT
3	BOI	Get_Attribute_Single Set_Attribute_Single	Bus off interrupt	n/a	BOOL
4	Bus Off Counter	Get_Attribute_Single Set_Attribute_Single	Number of times device went to bus off state.	0,0,255	USINT
5	Allocation Information	Get_Attribute_Single	Struct of: BYTE: Allocation choice USINT: Master's MAC ID	n/a	Struct of: BYTE USINT

* Assumes software configuration. If using dip switches, then the dip switches determines the default value.

Assembly Object, Class 0x04

Class Attributes (0)

#	Attribute Name	Service Name	Description	Default,Min,Max	Data Type
1	Revision	Get_Attribute_Single	Revision of assembly object.	2,1,65535	UINT
2	Max Instance	Get_Attribute_Single	Maximum instance number of an object currently created in this class level of the device.		UINT

Static Input Instance Attributes (100)

#	Attribute Name	Service Name	Description	Default,Min,Max	Data Type
3	DATA	Get_Attribute_Single	VARB1-8, data put into the network from the controller	n/a	Array of BYTE

Static Output Instance Attributes (150)

#	Attribute Name	Service Name	Description	Default,Min,Max	Data Type
3	DATA	Get_Attribute_Single Set_Attribute_Single	VARB9-16, data from the network and into the controller	n/a	Array of BYTE

Connection Object, Class 0x05

Class Attributes (0)

#	Attribute Name	Service Name	Description	Default,Min,Max	Data Type
1	Revision	Get_Attribute_Single	Revision of connection object.	1,1,65535	UINT

Explicit Connecting Instance (1)

#	Attribute Name	Service Name	Description	Default,Min,Max	Data Type
3	Transport Class Trigger	Get_Attribute_Single	Defines behavior of connection	0x83	BYTE
4	Produced Conn Id	Get_Attribute_Single	Placed in CAN identifier field when the connection transmits	n/a	UINT
5	Consumed Conn Id	Get_Attribute_Single	CAN identifier field value that denotes message to be received	n/a	UINT
6	Initial Comm Characteristics	Get_Attribute_Single	Defines the Message Group(s) across which productions and consumptions associated with this connection	n/a	BYTE
7	Produced Conn Size	Get_Attribute_Single	Maximum number of bytes transmitted across this connection	32,4,32	UINT
8	Consumed Conn Size	Get_Attribute_Single	Maximum number of bytes received across this connection	32,4,32	UINT
9	Expected Packet Rate	Get_Attribute_Single Set_Attribute_Single	Defines timing associated with this connection. Resolution 10 ms	n/a	UINT
12	Watchdog Timeout Action	Get_Attribute_Single	Defines how to handle inactivity/watchdog timeouts	n/a	USINT
13	Produced Connection Path Length	Get_Attribute_Single	Number of bytes in Produced_Connection_Path length attribute	0	UINT
14	Produced Connection Path	Get_Attribute_Single Set_Attribute_Single	Application object producing data on this connection	0	Array of: USINT
15	Consumed Connection Path Length	Get_Attribute_Single Set_Attribute_Single	Number of bytes in the Consumed_Connection_Path length attribute	0	UINT
16	Consumed Connection Path	Get_Attribute_Single	Specifies the application object(s) that are to receive data consumed by this connection object	n/a	Array of: 01 UINT
17	Production Inhibit Time	Get_Attribute_Single	Minimum delay time between new data production	n/a	UINT

* Assumes software configuration. If using dip switch, then the dip switch determines the default value.

Polled IO Connection Instance (2)

#	Attribute Name	Services	Description	Default,Min,Max	Data Type
1	State	Get_Attribute_Single	State of the object: 0=nonexistent, 1=configuring 3=established, 4=timed out	1,0,4	USINT
2	Instance Type	Get_Attribute_Single	Indicates either IO or messaging connection	0,0,1	USINT
3	Transport Class Trigger	Get_Attribute_Single	Defines behavior of connection	n/a	BYTE
4	Produced Conn Id	Get_Attribute_Single	Placed in CAN identifier field when the connection transmits	n/a	UINT

#	Attribute Name	Services	Description	Default,Min,Max	Data Type
5	Consumed Conn Id	Get_Attribute_Single	CAN identifier field value that denotes message to be received	n/a	UINT
6	Initial Comm Characteristics	Get_Attribute_Single	Defines the Message Group(s) across which productions and consumptions associated with this connection	n/a	BYTE
7	Produced Conn Size	Get_Attribute_Single	Maximum number of bytes transmitted across this connection	32,4,32	UINT
8	Consumed Conn Size	Get_Attribute_Single	Maximum number of bytes received across this connection	32,4,32	UINT
9	Expected Packet Rate	Get_Attribute_Single Set_Attribute_Single	Defines timing associated with this connection	n/a	UINT
12	Watchdog Timeout Action	Get_Attribute_Single	Defines how to handle inactivity/watchdog timeouts	n/a	USINT
13	Produced Connection Path Length	Get_Attribute_Single	Number of bytes in produced_connection_path length attribute	6	UINT
14	Produced Connection Path	Get_Attribute_Single Set_Attribute_Single	Application object producing data on this connection	20 04 24 64 30 03, n/a, n/a	ARRAY OF: USINT
15	Consumed Connection Path Length	Get_Attribute_Single	Number of bytes in the Consumed_Connection_Path length attribute	6	UINT
16	Consumed Connection Path	Get_Attribute_Single Set_Attribute_Single	Specifies the application object(s) that are to receive data consumed by this connection object	20 04 24 96 30 03, n/a, n/a	ARRAY OF: 01 UINT
17	Production Inhibit Time	Get_Attribute_Single	Minimum delay time between new data production.	n/a	UINT

Change of state/cyclic (4) (Acknowledged)

#	Attribute Name	Services	Description	Default,Min,Max	Data Type
1	State	Get_Attribute_Single	State of the object: 0=nonexistent,1=configuring 3=established, 4=timed out	1, n/a, n/a	USINT
2	Instance Type	Get_Attribute_Single	Indicates either IO or messaging connection	1,0,1	USINT
3	Transport Class Trigger	Get_Attribute_Single	Defines behavior of connection	n/a	BYTE
4	Produced Conn Id	Get_Attribute_Single	Placed in CAN identifier field when the connection transmits	n/a	UINT
5	Consumed Conn Id	Get_Attribute_Single	CAN identifier field value that denotes message to be received	n/a	UINT
6	Initial Comm Characteristics	Get_Attribute_Single	Defines the Message Group(s) across which productions and consumptions associated with this connection	n/a	BYTE
7	Produced Conn Size	Get_Attribute_Single	Maximum number of bytes transmitted across this connection	0,0, n/a	UINT
8	Consumed Conn Size	Get_Attribute_Single	Maximum number of bytes received across this connection	0,0, n/a	UINT
9	Expected Packet Rate	Get_Attribute_Single Set_Attribute_Single	Defines timing associated with this connection	0,0,0xffff	UINT
12	Watchdog Timeout Action	Get_Attribute_Single	Defines how to handle inactivity/watchdog timeouts	n/a	USINT
13	Produced Connection Path Length	Get_Attribute_Single	Number of bytes in Produced_Connection_Path length attribute	0,0,6	UINT

#	Attribute Name	Services	Description	Default,Min,Max	Data Type
14	Produced Connection Path	Get_Attribute_Single Set_Attribute_Single	Application object producing data on this connection	20 66 24 01 30 03, 0, n/a	ARRAY OF: USINT
15	Consumed Connection Path Length	Get_Attribute_Single	Number of bytes in the Consumed_Connection_Path length attribute	4	UINT
16	Consumed Connection Path	Get_Attribute_Single Set_Attribute_Single	Specifies the application object(s) that are to receive data consumed by this connection object	20 2B 24 01	ARRAY OF: UINT
17	Production Inhibit Time	Get_Attribute_Single, Set_Attribute_Single	Minimum delay time between new data production	n/a	UINT

Acknowledge Handler Object, Class 0x2B

Class Attributes (0)

#	Attribute Name	Service Name	Description	Default,Min,Max	Data Type
1	Revision	Get_Attribute_Single	Revision of assembly object	1,1,65535	UINT
2	Max Instance	Get_Attribute_Single	Maximum instance number of an object currently created in this class level of the device		UINT

Instance Attributes (1)

#	Attribute Name	Services	Description	Default,Min,Max	Data Type
1	Acknowledge Timer	Get_Attribute_Single Set_Attribute_Single	Time to wait for acknowledge before resending	16,1, 65535	UINT
2	Retry Limit	Get_Attribute_Single Set_Attribute_Single	Number of Ack timeouts to wait before informing the producing application of a RetryLimit_Reached event	1,0,255	USINT
3	COS Producing Connection Instance	Get_Attribute_Single Set_Attribute_Single	Connection instance which contains the path of the producing IO application object which will be notified of Ack handle error event	n/a	UINT
4	Ack List Size	Get_Attribute_Single	Maximum number of member in Ack list	n/a	BYTE
5	Ack List	Get_Attribute_Single	List of active connection instances, which are receiving Acks	n/a	BYTE Array of USINT
6	Data with Ack Path List Size	Get_Attribute_Single	Maximum number of members in data with Ack path list	n/a	BYTE
7	Data with Ack Path List	Get_Attribute_Single	List of connection instance/consuming application object pairs. This attribute is used to forward data received with acknowledgment.	n/a	BYTE Array of UINT USINT Array of USINT

Command Descriptions

The following is a list of all the DeviceNet specific commands. For a complete listing of 6K commands see the *6K Series Command Reference*.

ERROR Error Checking Enable

		Product	Rev
Type:	Communication Setup		
Syntax:	<!><%>ERROR<b><b>…<b><b> (32bits)	6Kn-DN	5.2
Units:	n/a		
Range:	b=0 (disable), 1 (enable), or X (don't change)		
Default:	0		
Response:	ERROR: *ERROR0000_0000_0000_0000_0000_0000_0000_0000		
See Also:	OUTFNC, FBS, TER, TERF		

A new bit assignment is added for the ERROR command, bit #19.

See FBS on page 15 for the event causing the error condition. To clear the error event, first resolve the cause, and then issue the ERROR.19-0 command followed by the ERROR.19-1 command. Error bit 19 is edge sensitive to error events.

In the event an option card fault occurs, VARB1-16 are cleared on the controller side.

Bit #	Function	Branch Type
19	Option card fault	GOTO

FBADDR Fieldbus Address

		Product	Rev
Type:	Communication Setup		
Syntax:	<!>FBADDR<i>	6Kn-DN	5.2
Units:	i = fieldbus address		
Range:	i = 0-63		
Default:	1		
Response:	FBADDR: *FBADDR3		
See Also:	FBBAUD, FBSIZE, TOPSTS		

Use the FBADDR command to report the controller's current node address assignment and set the node address via software. The new value is saved into nonvolatile memory, and becomes effective after the controller is reset. This command cannot report the hardware configuration setting. In order to set the node address via software, the hardware configuration method must be disabled (default from factory). See Node Addresses on page 4.

When setting the node address via software, the baudrate must also be set via software, and vice versa. If the hardware configuration method is used to set the node address, an attempt to set the node address via software will be ignored. No message is reported indicating success or failure.

Network configuration of node address is not supported.

Example:
Assume controller was assigned node address 1 out of reset:

```
>FBADDR
*FBADDR1
>FBADDR3        ;Set node address to 3
>FBADDR
*FBADDR3        ;New network setting will take effect after unit is reset!
```

FBBAUD Fieldbus Baud Rate

		Product	Rev
Type:	Communication Setup		
Syntax:	<!>FBBAUD<r>	6Kn-DN	5.2
Units:	r = kbits/second (kbps)		
Range:	r = 125, 250, or 500		
Default:	500		
Response:	FBBAUD: * FBBAUD500		
See Also:	FBADDR, FBSIZE, TOPSTS		

Use the FBBAUD command to report the controller's current baudrate assignment (used during boot-up) and set the baudrate via software. This command cannot report the hardware configuration setting. In order to set the baudrate via software, the hardware configuration method must be disabled (default from factory). See Baud Rate on page 5.

When setting the baudrate via software, the node address must also be set via software, and vice versa. If the hardware configuration method is used to set the baudrate, an attempt to set the baudrate via software will be ignored. No message is reported indicating success or failure.

The new value is saved into nonvolatile memory, and becomes effective after the controller is reset.

Example:

```
>FBBAUD
*FBBAUD500
>FBBAUD250      ;Set baudrate to 250
>FBBAUD
*FBBAUD250      ;New network setting will take effect after unit is reset!
```

[FBS] Fieldbus Status

		Product	Rev
Type:	Communication Setup		
Syntax:	See below	6Kn-DN	5.2
Units:	n/a		
Range:	n/a		
Default:	n/a		
Response:	n/a		
See Also:	ER.19, TFBS, TFBSF		

Use the FBS command to assign the fieldbus status to a binary variable or in a comparison command.

Example:
```
IF(FBS.4=b1)     ;Branch based on the status of FBS bit 4
```

The FBS register bits are defined as follows:

Bit #	Function (1=Yes, 0=No)	Description
1	TIMEOUT [1,2]	Watchdog timed out. Controller has lost communication with fieldbus card.
2	CHECKSUM FAULT [1,2]	Fieldbus card failed hardware check on boot-up.
3	Reserved	
4	NETWORK LINK OK [1,3]	Communication established and active.
5	NETWORK CRITICAL LINK FAILURE [4]	Cannot communicate with the fieldbus. Duplicate node address or bus-off state exists.
6	NETWORK LINK NOT CONNNECTED [4]	Passed Dup_MAC_ID test and online, but no connection established with another node
7	NETWORK CONNECTION TIME OUT [4]	Connection has timed out
8	MODULE DEVICE OPERATIONAL [4]	Operating in a normal condition
9	MODULE UNRECOVERABLE FAULT [4]	Fieldbus card may need to be replaced.
10	MODULE DEVICE IN STANDBY[4]	Device needs commissioning due to configuration missing, incomplete, or incorrect.
11	MODULE MINOR FAULT [4]	Fieldbus card is indicating a recoverable fault.
12-32	Reserved	

[1] If any of these error conditions occur (bit #1 = 1, bit #2 = 1, or bit #4 = 0), the motion controller will perform a Kill (K command) on all axes. If error-checking bit #19 is enabled with the ERROR command (ERROR.19-1) the controller will also set error status bit #19 (ER, TER, and TERF) and branch to the ERRORP program.

[2] Error event is latched. Reset the controller to clear the error.

[3] Error event is recoverable, if error checking (ERROR) bit #19 is enabled and ERRORP program exists. If error bit #19 is disabled or no ERRORP program exists, the event becomes latched, and you will need to reset the controller to clear the error.

[4] After error checking (ERROR) bit #19 is set, it can take up to 600mS to correctly set these status bits.

NOTE: If FBS bits 4-11 are all 0, no external power has been applied to the bus.

FBSIZE Fieldbus Data Packet Size

		Product	Rev
Type:	Communication Setup		
Syntax:	<!>FBSIZE<i>	6Kn-DN	5.2
Units:	n/a		
Range:	i = 1-8		
Default:	8		
Response:	FBSIZE: *FBSIZE8		
See Also:	FBADDR, FBBAUD		

Use the FBSIZE command to set the number of binary variables exchanged with a DeviceNet master. Data received or sent to the master is of the same size (cyclic), and each binary variable is 4 bytes. The new value is saved into nonvolatile memory, and becomes effective after the controller is reset.

Example:

```
FBSIZE8      ;Set fieldbus data packet size to 8 binary variables.
FBSIZE2      ;Set fieldbus data packet size to 2 binary variables.
```

Here are the variable assignments (from the controller's perspective) for each possibility of FBSIZE:

Command	Data out	Data in
FBSIZE1	VARB1 – VARB1	VARB9 – VARB9
FBSIZE2	VARB1 – VARB2	VARB9 – VARB10
FBSIZE3	VARB1 – VARB3	VARB9 – VARB11
FBSIZE4	VARB1 – VARB4	VARB9 – VARB12
FBSIZE5	VARB1 – VARB5	VARB9 – VARB13
FBSIZE6	VARB1 – VARB6	VARB9 – VARB14
FBSIZE7	VARB1 – VARB7	VARB9 – VARB15
FBSIZE8	VARB1 – VARB8	VARB9 – VARB16

Regardless of FBSIZE setting, VARB1-16 are reserved for DeviceNet activity and not available for general use.

OPTEN Option Card Enable/Disable

		Product	Rev
Type:	Communication Setup		
Syntax:	<!>OPTEN<i>	6Kn-DN	5.2
Units:	n/a		
Range:	i = 0(disable), 1(enable)		
Default:	1		
Response:	OPTEN: *OPTEN1		
See Also:	TOPSTS		

Use the OPTEN command to enable (OPTEN1) or disable (OPTEN0) the option card on power-up. This feature allows Ethernet to be enabled when a DeviceNet card is installed but disabled (if applicable). It also restores VARB1-16 for use by user's application. Caution: If you later re-enable OPTEN1, then VARB1-16 are reserved for fieldbus activity.

The new value is saved into nonvolatile memory, and becomes effective **after power is cycled**.

OUTFNC Output Function

		Product	Rev
Type:	Output		
Syntax:	<!><B>OUTFNC<i><-<a>c>	6Kn	5.2
Units:	i = output #, a = axis, c = function identifier (letter)		
Range:	i = 1-32 (I/O brick dependent), a = 1-8 (depends on product), c = A-H		
Default:	c = A (programmable output function - default)		
Response:	OUTFNC: (function and status of onboard outputs) 1OUTFNC: (function and status of outputs on I/O brick 1) 1OUTFNC1: *1OUTFNC1-A PROGRAMMABLE OUTPUT - STATUS OFF		
See Also:			

Identifier	Function Description
I	**Option Card Fault**: Output activates when error bit #19 is set for the option card fault. See the ERROR command for description of events. This requires ERROR.19-1 to be set, or the output will not activate. The OUTFNC-I command can only be assigned to task 0. If it is assigned to other than task 0, the error message "ALTERNATE TASK NOT ALLOWED" will be generated. OUTFNC-I cannot be assigned to a specific axis.

Example:

```
0%1OUTFNC8-i   ;assign brick 1, output 8 to option card fault
0%OUTFNC1-i    ;assign on-board output 1 to option card fault
2%OUTFNC1-i    ;only task 0 allowed
2%ALTERNATE TASK NOT ALLOWED
```

TFBS Transfer Fieldbus Status

		Product	Rev
Type:	Communication Status		
Syntax:	<!>TFBS<.i>	6Kn-DN	5.2
Units:	i = system status bit number		
Range:	1-32		
Default:	n/a		
Response:	TFBS: TFBS.4: *1 (unit online or link ok, yes) *TFBS001_0000_0000_0000_0000_0000_0000_0000		
See Also:	ER.19, [FBS], TFBSF		

The TFBS command provides information on the 32 fieldbus status bits. The TFBS command reports a binary bit report. If you would like to see a more descriptive text based report, use the TFBSF command.

Response for TFBS: *TFBS0001_0000_0000_0000_0000_0000_0000_0000

Bit#1…bit#32

For bit description, see FBS on page 15.

TFBSF Fieldbus Status Full Text

		Product	Rev
Type:	Communication Status		
Syntax:	<!>TFBSF	6Kn-DN	5.2
Units:	n/a		
Range:	n/a		
Default:	n/a		
Response:	see example		
See Also:	ER.19, [FBS], TFBS, TOPSTS		

Use the TFBSF command to check the status of the fieldbus and display the status in full ASCII text to a terminal.

For status description, see FBS on page 15.

Example TFBSF **response**:

```
* TIMEOUT                              NO      RESERVED       NO
* CHECKSUM FAULT                       NO      RESREVED       NO
* RESERVED                             NO      RESERVED       NO
* NETWORK LINK OK                      YES     RESERVED       NO

* NETWORK CRITICAL LINK FAILURE        NO      RESERVED       NO
* NETWORK LINK NOT CONNECTED           NO      RESERVED       NO
* NETWORK CONNECTION TIMEOUT           NO      RESERVED       NO
* MODULE DEVICE OPERATIONAL            YES     RESERVED       NO
*
* MODULE UNRECOVERABLE FAULT           NO      RESERVED       NO
* MODULE DEVICE IN STANDBY             NO      RESERVED       NO
* MODULE MINOR FAULT                   NO      RESERVED       NO
* RESERVED                             NO      RESERVED       NO
*
* RESERVED                             NO      RESERVED       NO
* RESERVED                             NO      RESERVED       NO
* RESERVED                             NO      RESERVED       NO
* RESERVED                             NO      RESERVED       NO
```

TOPSTS Option Card Status Full Text

		Product	Rev
Type:	Option Status		
Syntax:	<!>TOPSTS	6Kn-DN	5.2
Units:	n/a		
Range:	n/a		
Default:	n/a		
Response:	see example		
See Also:	OPTEN, TFBSF		

Use the TOPSTS command to check the status of the option card and display the status in full ASCII text to a terminal.

Example TOPSTS **response:**

```
*6K OPTION CARD STATUS
*
*Option Card Enabled: Yes
*Option Card Type: DeviceNet
*Option Card Firmware Rev: 92-018750-01-1.1
*Option Card Serial Number: 8-65535-65535
*
*6K DeviceNet Vendor ID: 620 (decimal)
*6K DeviceNet Product Code: 1 (decimal)
*6K DeviceNet Packet Size: FBSIZE8
*6K DeviceNet Address: FBADDR1
*
*6K DeviceNet Baudrate: FBBAUD500
```

Programming Scenario

NOTE: To understand the overall implementation process, refer to page 2.

```
;*******************************************************************************

DEL ERHND
DEF ERHND

;-------------------------------------------------------------------------------
;Fieldbus error event
;If the error event can be resolved, an unconditional jump is made to
;re-initialize the controller.
;-------------------------------------------------------------------------------
IF (ER.19 = b1)
       ;Insert application specific events to execute when a fieldbus error occurs.

       ;Wait for controller to go back online
       WAIT(FBS = b00X1)

       ; Controller back online
       ERROR.19-0          ;Acknowledge error event has been resolved
       ERROR.19-1          ;

       JUMP MAIN           ;Call to MAIN or other suitable initializer.
NIF

;-------------------------------------------------------------------------------
;Post power-up error event
;If the error event can be resolved, an unconditional jump is made to
;re-initialize the controller.
;-------------------------------------------------------------------------------
IF (FBS <> b00X1)
       ;Wait for controller to go back online
       WAIT(FBS = b00X1)

       JUMP MAIN
NIF

END ;ERHND program


;*******************************************************************************


;-------------------------------------------------------------------------------
;MAIN program
;
;In this program, the fieldbus error handler is assigned, enabled, and an output
;is activated when a fieldbus error occurs.
;
;Next a power-up check is made to determine if the 6k is active on the fieldbus.
;If not, the controller makes an unconditional jump to the error handler.
;
;After completing configuration and power-up checks, the controller begins
;exchanging data with the master. This section demonstrates mailbox messaging.
;
;-------------------------------------------------------------------------------
DEL MAIN
DEF MAIN

       ;Initialize controller
       ERRORP ERHND        ;Assign error handler program
       ERROR.19-1          ;Run ERRORP program (ERHND) when fieldbus error occurs
       OUTFNC8-I           ;Activate onboard output 8 if fieldbus fault occurs
```

```
        ;Post power-up check to verify no fieldbus errors exist.
        IF(FBS <> b00X1)     ;Check to see if it's online
                JUMP ERHND    ;Fieldbus error, jump to error program
        NIF


        ;Application's main loop
        L
                IF(VARB9.1 <> VARB1.1)
                        ;SEND NEW MESSAGE TO MASTER
                        WRITE"SENT NEW MESSAGE"

                        VARI1=4PE            ;Assign axis 4 encoder position to VARI1
                        VARB2=VCVT(VARI1)    ;Send encoder position out

                        VARB1=VARB1^H1       ;Notify master new message exists
                        T2
                NIF

                IF(VARB9.2 <> VARB1.2)
                        ;READ MESSAGE FROM MASTER
                        WRITE"GOT NEW MESSAGE"

                        VAR10=VCVT(VARB10)
                        A,(VAR10)            ;Assign new accel value

                        VAR11=VCVT(VARB11)
                        D,(VAR11)            ;Assign new distance value

                        VARB1=VARB1^H2       ;Acknowledge message received
                        T2
                NIF
        LN


END ; MAIN program

;*****************************************************************************

STARTP MAIN   ;Assign MAIN as the program to be run automatically on power-up and reset.
```

CHAPTER TWO

2 Implementing Gemini DeviceNet

IN THIS CHAPTER

DeviceNet Overview

Gemini GT6-DN and GV6-DN

The Gemini GT6-DN and GV6-DN are high performance stepper and servo drives designed to support the Open DeviceNet Vendor's Association (ODVA) specification for DeviceNet. These products have been self-tested by the vendor and found to comply with ODVA Protocol Conformance Test Software Version A-16.

These compact digital drives will offer enhancements over today's drive offering with features such as digital tuning and notch filters for our servos, as well as sensorless stall detect and ABS damping for our steppers. These drive/controllers are configured over RS232/485 with Motion Planner on a full size PC or laptop.

Cabling is not provided by Compumotor.

Technical Assistance

Technical questions regarding DeviceNet should be addressed to your local DeviceNet User Group. An address list is available on the DeviceNet Internet site at www.odva.org.

For support with Gemini specific questions, contact Applications at 800-358-9070 or e-mail us at tech_help@cmotor.com.

Implementation Process

DeviceNet Master (user defined):

1. Use the provided file, CMTR2.EDS. Do not modify.
2. Configure communication baudrate.

Gemini Drive:

1. Enable/disable terminating resistors as needed (see page 25).
2. Launch Motion Planner (CD-ROM is provided in your ship kit).
3. Establish a direct communication link (serial) with the Gemini. Refer to your hardware installation documentation for connection instructions.
4. Configure node address (see `FBADDR`, or use hardware method on page 26).
5. Configure baud rate (see `FBBAUD` on page 61 or use the hardware method on page 26). Setting the baud rate via hardware overrides the software setting.
6. Reset the Gemini drive to initialize the DeviceNet card.

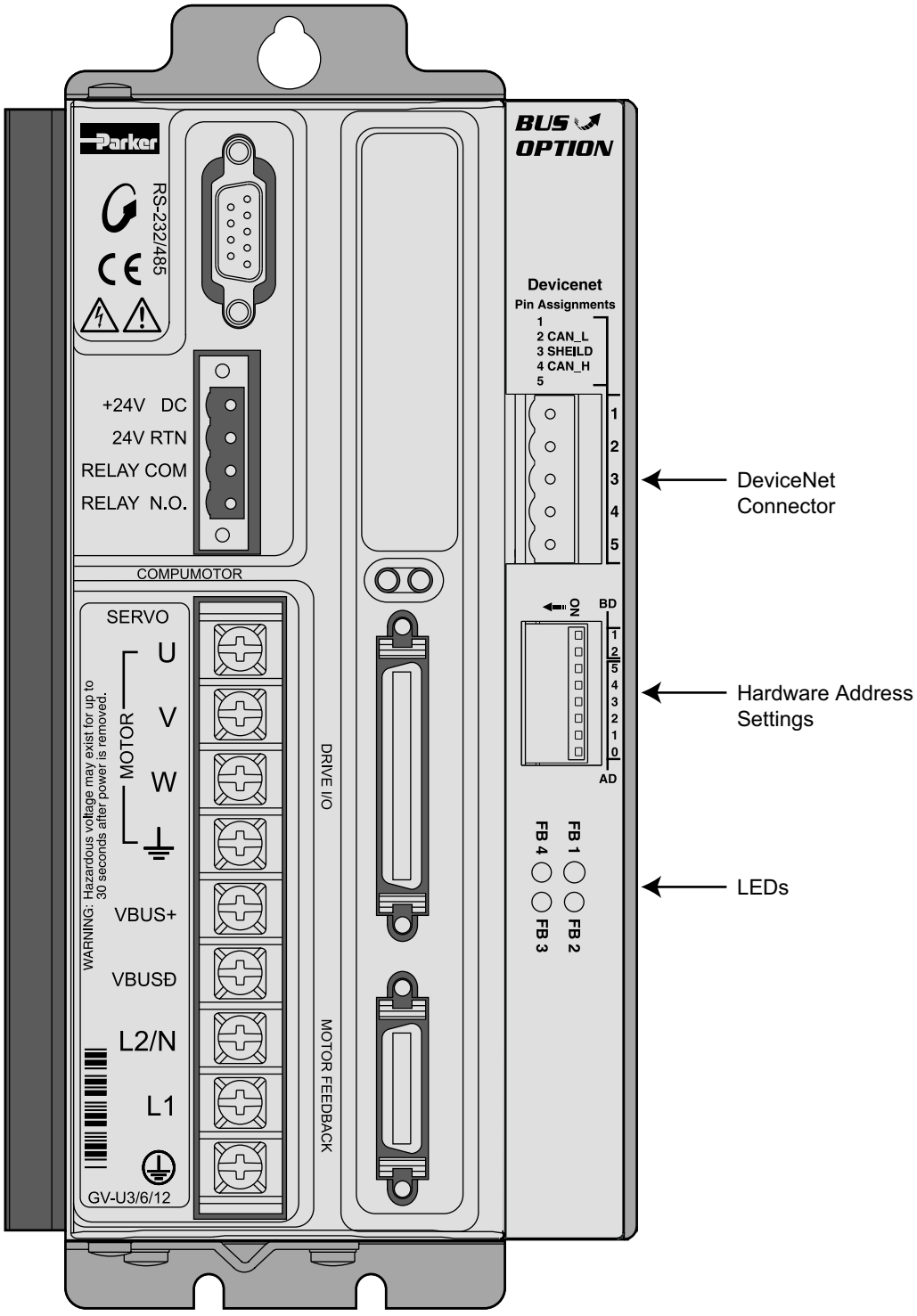
EDS File

Each device in a DeviceNet network is associated with an EDS file, containing all necessary information about the device. The latest version of the Gemini EDS file (CMTR2.EDS) can be downloaded from www.compumotor.com.

Data Types

The table below describes the various data types associated with the DeviceNet object attributes, number of bits, and min/max values.

Data Type	# of Bits	Min Value	Max Value
BOOL (Boolean)	1	0 (False)	1 (True)
SINT (Short Integer)	8	-128	127
USINT (Unsigned Short Integer)	8	0	255
INT (Integer)	16	-32768	32767
UINT (Unsigned Integer)	16	0	65535
DINT (Double Integer)	32	-2^{31}	$2^{31}-1$
UDINT (Unsigned Double Integer)	32	0	$2^{32}-1$



LED Status Indicators

Bicolor flashing LED indicators are provided on the DeviceNet option card. Refer to the following table for troubleshooting information provided by these LEDs.

LED	Steady	Flash	Function	Status *
FB1	--	--	Not used	--
FB2	Off		Not powered/not online	--
	Green		Network link ok	FBS bit #4 = 1
	Red		Network critical link failure	FBS bit #5 = 1
		Green	1 flash/second – Network link not connected	FBS bit #6 = 1
		Red	1 flash/second – Network connection timeout	FBS bit #7 = 1
FB3	Off		No power	--
	Green		Module device operational	FBS bit #8 = 1
	Red		Module unrecoverable fault	FBS bit #9 = 1
		Green	1 flash/second – Module in standby	FBS bit #10 = 1
		Red	1 flash/second – Module minor fault	FBS bit #11 = 1
FB4	--	--	Not used	--

* To check status, execute the `TFBS` command (bit status report) or the `TFBSF` command (full text status report) in the terminal emulator. You can also use the `FBS` operator to assign or compare one or more status bits (e.g., use in an `IF` expression, etc.). Refer to the `TFBS` command description on page 63.

DeviceNet Connector Pin Out

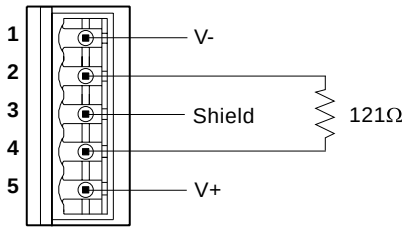
The following table gives the pin out for the DeviceNet connector. Industry standard DeviceNet connections are used.

Pin	Name	Function
1	V-	DC Return
2	CAN_L	CANBUS LOW
3	SHIELD	Protective Earth
4	CAN_H	CANBUS HIGH
5	V+	+24 VDC Power *

* 30mA in standby and 100mA in rush

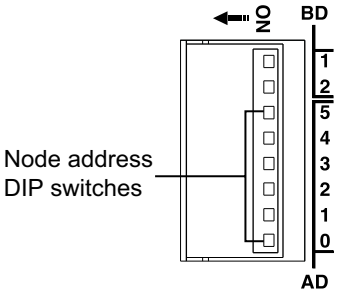
Termination

If the Gemini drive/controller is used as the first or last node in a network, a termination resistor must be used. Compumotor recommends a resistor value of 121 ohms. Connect the resistor as shown below.



Node Address

To configure node address via hardware, dip switches are provided to set a node address of 0-63. The switches AD0-AD5 give a binary representation of the node address. Setting the dip switches to 0xFF (all ON), enables software configuration of node address (see FBADDR on page 60).



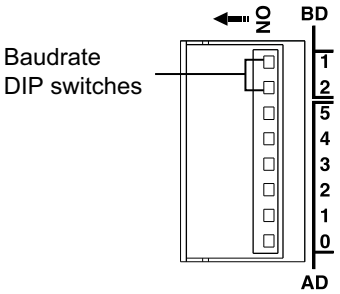
Address	AD5 (32)	AD4 (16)	AD3 (8)	AD2 (4)	AD1 (2)	AD0 (1)
0	OFF	OFF	OFF	OFF	OFF	OFF
1	OFF	OFF	OFF	OFF	OFF	ON
2	OFF	OFF	OFF	OFF	ON	OFF
3	OFF	OFF	OFF	OFF	ON	ON
…						
62	OFF	ON	ON	ON	ON	ON
63	ON	ON	ON	ON	ON	ON

Example**:**

Switch AD0-AD5 = OFF-ON-OFF-ON-ON-ON, node address is 23 (16+4+2+1).
Switch AD0-AD5 and BD1-BD2 = all ON, node address determined by FBADDR

Baud Rate

To configure baud rate via hardware, dip switches are provided to set a baud rate of 125, 250, or 500kb. Setting the dip switches to 0xFF (all ON), enables software configuration of baud rate (see FBBAUD on page 61).



Baudrate (Bit/sec)	BD1	BD2
125k	OFF	OFF
250k	OFF	ON
500k	ON	OFF
Reserved	ON	ON

Example**:**

Switch BD1-BD2 = ON-OFF, baud rate 500kbit/s
Switch AD0-AD5 and BD1-BD2 = all ON, baud rate determined by FBBAUD

All parameters on the drives are set over RS232/485 except with regard to the DeviceNet card. This card provides dip switches for device number and baud rate.

Basic Requirements

The Gemini DeviceNet options are based on the Gemini GV6 and GT6 platforms. All commands supported by these platforms, with the exception of RS232/485 commands, will also be supported by the DeviceNet options. See the *Gemini Series Programmer's Reference* for a complete listing of these commands. Additional DeviceNet specific commands are listed on page 60 of this manual.

Hardware Requirements

Devices are expected to operate in the following environment:

Explicit Peer-to-Peer Messaging	No
I/O Peer to Peer Messaging	No
Baud Rate	500 kB
Polled Response Time	< 10msec
Explicit Response Time	< 50msec
Master/Scanner	No
I/O Slave Messaging	
Bit Strobe	No
Polling	Yes
Cyclic	No
Change-of-State	No

Object Model

Object Class	Class code	Instance #	Description
Identity	0x01	1	Provides identification of and general information about the device. The Identity Object MUST be present in all DeviceNet products.
Message Router	0x02	1	Provides a messaging connection point through which a Client may address a service to any object class or instance residing in the physical device
DeviceNet	0x03	1	Provides the configuration and status of a DeviceNet port. Each DeviceNet product must support one (and only one) DeviceNet object per physical connection to the DeviceNet communication link.
Assembly	0x04	1	Binds attributes of multiple objects, which allows data to or from each object to be sent or received over a single connection. Stores I/O output Message Data.
Assembly	0x04	2	Stores I/O Input Message Data
Explicit Connection	0x05	1	Manages the explicit messages
I/O Connection	0x05	2	Manages the I/O messages
Acknowledge Hander	0x2B	1	Handles/configures acknowledgement of data.
Position Controller Supervisor	0xA4	1	Handles errors for the position controller as well as Home and Registration inputs
Position Controller	0xA5	Axis	Performs the control output velocity profiling and handles input and output to and from the motor drive unit, limit switches registration etc.

Position Controller Supervisor Object Class (#164)(A4hex) and Position Controller Object Class (#165)(A5hex)

Explicit Messaging Connections provide generic, multi-purpose communication paths between two devices. Explicit Messages are exchanged across Explicit Messaging Connections. Explicit Messages are used to command the performance of a particular task and to report the results of performing the task. The meaning/intended use of an Explicit Message is stated within the CAN Data Field. Explicit Messaging provides the means by which typical request/response oriented functions are performed (e.g. module configuration). DeviceNet defines an Explicit Messaging protocol that states the meaning of the message.

An Explicit Message consists of a Connection ID and associated messaging protocol information.

Explicit Message Command

TXID	COMMAND	Word 0
PORT	SIZE	Word 1
SERCE	MAC ID	Word 2
CLASS		Word 3
AXIS INSTANCE		Word 4
ATTRIBUTE		Word 5
LOWER DATA WORD		Word 6
UPPPER DATA WORD		Word 7

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 1 - Channel B
SIZE	Size in bytes of all data after MAC ID.
SERVICE	0E(hex) - Get_Attribute_Single 10(hex) - Set_Attribute_Single
MAC ID	The DeviceNet ID of the amplifier.
CLASS	Position Controller Supervisor – 164(A4 hex) Position Controller Object - 165(A5 hex)
AXIS INSTANCE	Always 1 (01 hex)
ATTRIBUTE	The attribute number of the attribute being accessed (set or get).

Explicit Message Response

TXID	STATUS	Word 0
PORT	SIZE	Word 1
SERCE	MAC ID	Word 2
DATA		Word 3 - 31

TXID	Transmit ID This value must be unique for each explicit message sent.
STATUS	The Status specified for this block of data. 01 - Transaction Successful XX - For all other response definitions, refer to the DeviceNet Master documentation.
PORT	0 - Channel A 1- Channel B
SIZE	Size in bytes of all data after MAC ID.
SERVICE	0E(hex) - Get_Attribute_Single 10(hex) - Set_Attribute_Single
MACID	The DeviceNet ID of the amplifier.

Services

Service Code	Name	Description
0x0E	Get_Attribute_Single	Returns the specified attribute.
0x10	Set_Attribute_Single	Sets the specified attribute.

Example:

In this example, we will use the Set_Attribute_Single service (10 hex) to set the Target Position attribute (06 hex) to 65536 (00010000 hex) of the Position Controller Class 165 (A5 hex) of MACID 25 (19 hex). In all cases with Gemini motion controllers, the axis instance is "1" (01 hex).

TXID	COMMAND	0101
PORT	SIZE(IN BYTES)	000A
SERVICE	MAC ID	1019
CLASS		00A5
AXIS INSTANCE		0001
ATTRIBUTE		0006
LOWER DATA WORD		0000
UPPER DATA WORD		0001

Identity Object, Class 0x01

Class Attributes (0)

#	Attribute Name	Service Name	Description	Default,Min,Max	Data Type
1	Revision	Get_Attribute_Single	Revision of identity object	1,1,65535	UINT

Instance Attributes (1)

#	Attribute Name	Service Name	Description	Default,Min,Max	Data Type
1	Vendor Id	Get_Attribute_Single	Parker Hannifin/Compumotor Division	620	UINT
2	Device Type	Get_Attribute_Single	Communications Adapter	12	UINT
3	Product Code	Get_Attribute_Single	Gemini Drive	2	UINT
4	Revision	Get_Attribute_Single	Consists of major.minor	{1,1}, {1,1}, {1,1}	Array of USINT
5	Status	Get_Attribute_Single	Represents the current status of the entire device.	0,0,255	WORD
6	Serial Number	Get_Attribute_Single	Used in conjunction with the Vendor ID to form a unique identifier for each device on DeviceNet.	n/a	UDINT
7	Product Name	Get_Attribute_Single	Short description of the product represented by the Product Code.	“Gemini Drive”	SHORT_ STRING
9	Config. Consist. Value	Get_Attribute_Single	Contents identify configuration of device.	n/a	UINT

DeviceNet Object, Class 0x03

Class Attributes (0)

#	Attribute Name	Service Name	Description	Default,Min,Max	Data Type
1	Revision	Get_Attribute_Single	Revision of identity object.	2,1,65535	UINT
32	Consumed Command Message	Get_Attribute_Single	Contains the content of the Command Message	Contains the content of the Command Message	
33	Produced Response Message	Get_Attribute_Single	Contains the content of the Response Message	Contains the content of the Response Message	

Instance Attributes (1)

#	Attribute Name	Service Name	Description	Default,Min,Max	Data Type
1	MAC_ID	Get_Attribute_Single	Node Address	1*,0,63	USINT
2	Baud Rate	Get_Attribute_Single	Indicates the selected baud rate. Values are: 00 - 125k 01 - 250k 10 - 500k	2*,0,2	USINT
3	BOI	Get_Attribute_Single Set_Attribute_Single	Bus off interrupt	n/a	BOOL
4	Bus Off Counter	Get_Attribute_Single Set_Attribute_Single	Number of times device went to bus off state.	0,0,255	USINT
5	Allocation Information	Get_Attribute_Single	Struct of: BYTE: Allocation choice USINT: Master's MAC ID	n/a	Struct of: BYTE USINT

* Assumes software configuration. If using dip switches, then the dip switches determines the default value.

Assembly Object, Class 0x04

Class Attributes (0)

#	Attribute Name	Service Name	Description	Default,Min,Max	Data Type
1	Revision	Get_Attribute_Single	Revision of assembly object.	2,1,65535	UINT
2	Max Instance	Get_Attribute_Single	Maximum instance number of an object currently created in this class level of the device.		UINT

Static Input Instance Attributes (100)

#	Attribute Name	Service Name	Description	Default,Min,Max	Data Type
3	DATA	Get_Attribute_Single	Data put into the network from the controller	n/a	Array of BYTE

Static Output Instance Attributes (150)

#	Attribute Name	Service Name	Description	Default,Min,Max	Data Type
3	DATA	Get_Attribute_Single Set_Attribute_Single	Data from the network and into the controller	n/a	Array of BYTE

Connection Object, Class 0x05

Class Attributes (0)

#	Attribute Name	Service Name	Description	Default,Min,Max	Data Type
1	Revision	Get_Attribute_Single	Revision of connection object.	1,1,65535	UINT

Explicit Connecting Instance (1)

#	Attribute Name	Service Name	Description	Default,Min,Max	Data Type
1	State	Get_Attribute_Single	State of the object: 0=nonexistent, 1=configuring 3=established, 4=timed out	1,0,4	USINT
2	Instance Type	Get_Attribute_Single	Indicates either IO or messaging connection 0=Explicit, 1=I/O	0,0,1	USINT
3	Transport Class Trigger	Get_Attribute_Single	Defines behavior of connection	0x83	BYTE
4	Produced Conn Id	Get_Attribute_Single	Placed in CAN identifier field when the connection transmits	n/a	UINT
5	Consumed Conn Id	Get_Attribute_Single	CAN identifier field value that denotes message to be received	n/a	UINT
6	Initial Comm Characteristics	Get_Attribute_Single	Defines the Message Group(s) across which productions and consumptions associated with this connection	n/a	BYTE
7	Produced Conn Size	Get_Attribute_Single	Maximum number of bytes transmitted across this connection	516,4,516	UINT
8	Consumed Conn Size	Get_Attribute_Single	Maximum number of bytes received across this connection	516,4,516	UINT
9	Expected Packet Rate	Get_Attribute_Single Set_Attribute_Single	Defines timing associated with this connection. Resolution 10 ms	n/a	UINT
12	Watchdog Timeout Action	Get_Attribute_Single Set_Attribute_Single	Defines how to handle inactivity/watchdog timeouts	1 – Auto Delete 3 – Deferred Delete	USINT
13	Produced Connection Path Length	Get_Attribute_Single	Number of bytes in Produced_Connection_Path length attribute	0	UINT
14	Produced Connection Path	Get_Attribute_Single Set_Attribute_Single	Application object producing data on this connection	0	Array of: USINT
15	Consumed Connection Path Length	Get_Attribute_Single Set_Attribute_Single	Number of bytes in the Consumed_Connection_Path length attribute	0	UINT
16	Consumed Connection Path	Get_Attribute_Single	Specifies the application object(s) that are to receive data consumed by this connection object	n/a	Array of: 01 UINT
17	Production Inhibit Time	Get_Attribute_Single	Minimum delay time between new data production	n/a	UINT

Polled IO Connection Instance (2)

#	Attribute Name	Services	Description	Default,Min,Max	Data Type
1	State	Get_Attribute_Single	State of the object: 0=nonexistent, 1=configuring 3=established, 4=timed out	1,0,4	USINT
2	Instance Type	Get_Attribute_Single	Indicates either IO or messaging connection 0=Explicit, 1=I/O	0,0,1	USINT
3	Transport Class Trigger	Get_Attribute_Single	Defines behavior of connection	n/a	BYTE
4	Produced Conn Id	Get_Attribute_Single	Placed in CAN identifier field when the connection transmits	n/a	UINT
5	Consumed Conn Id	Get_Attribute_Single	CAN identifier field value that denotes message to be received	n/a	UINT
6	Initial Comm Characteristics	Get_Attribute_Single	Defines the Message Group(s) across which productions and consumptions associated with this connection	n/a	BYTE
7	Produced Conn Size	Get_Attribute_Single	Maximum number of bytes transmitted across this connection	32,4,32	UINT
8	Consumed Conn Size	Get_Attribute_Single	Maximum number of bytes received across this connection	32,4,32	UINT
9	Expected Packet Rate	Get_Attribute_Single Set_Attribute_Single	Defines timing associated with this connection	n/a	UINT
12	Watchdog Timeout Action	Get_Attribute_Single	Defines how to handle inactivity/watchdog timeouts	n/a	USINT
13	Produced Connection Path Length	Get_Attribute_Single	Number of bytes in produced_connection_path length attribute	6	UINT
14	Produced Connection Path	Get_Attribute_Single Set_Attribute_Single	Application object producing data on this connection	20 04 24 64 30 03, n/a, n/a	ARRAY OF: USINT
15	Consumed Connection Path Length	Get_Attribute_Single	Number of bytes in the Consumed_Connection_Path length attribute	6	UINT
16	Consumed Connection Path	Get_Attribute_Single Set_Attribute_Single	Specifies the application object(s) that are to receive data consumed by this connection object	20 04 24 96 30 03, n/a, n/a	ARRAY OF: 01 UINT
17	Production Inhibit Time	Get_Attribute_Single	Minimum delay time between new data production.	n/a	UINT

NB: Attributes 14,15 and 16 can only be set when the connection is configuring.

Acknowledge Handler Object, Class 0x2B

Class Attributes (0)

#	Attribute Name	Service Name	Description	Default,Min,Max	Data Type
1	Revision	Get_Attribute_Single	Revision of assembly object	1,1,65535	UINT
2	Max Instance	Get_Attribute_Single	Maximum instance number of an object currently created in this class level of the device		UINT

Instance Attributes (1)

#	Attribute Name	Services	Description	Default,Min,Max	Data Type
1	Acknowledge Timer	Get_Attribute_Single Set_Attribute_Single	Time to wait for acknowledge before resending	20,1, 65535	UINT
2	Retry Limit	Get_Attribute_Single Set_Attribute_Single	Number of Ack timeouts to wait before informing the producing application of a RetryLimit_Reached event	1,0,255	USINT
3	COS Producing Connection Instance	Get_Attribute_Single Set_Attribute_Single	Connection instance which contains the path of the producing IO application object which will be notified of Ack handle error event	n/a	UINT
4	Ack List Size	Get_Attribute_Single	Maximum number of member in Ack list	n/a	BYTE
5	Ack List	Get_Attribute_Single	List of active connection instances, which are receiving Acks	n/a	BYTE Array of USINT
6	Data with Ack Path List Size	Get_Attribute_Single	Maximum number of members in data with Ack path list	n/a	BYTE
7	Data with Ack Path List	Get_Attribute_Single	List of connection instance/consuming application object pairs. This attribute is used to forward data received with acknowledgment.	n/a	BYTE Array of UINT USINT Array of USINT

Position Controller Supervisor Object (0xA4)

This section describes the required behavior of the Position Controller Supervisor (0xA4).

Class Attributes (0)

#	Attribute Name	Service Name	Description	Default,Min,Max	Data Type
32	Consumed Command Message	Get_Attribute_Single	Contains the content of the Command Message	Contains the content of the Command Message	
33	Produced Response Message	Get_Attribute_Single	Contains the content of the Response Message	Contains the content of the Response Message	

Object Instance Attributes(1)

#	Attribute Name	Service Name	Description	Default	Range	Data Factor	Data Type
2	Attribute List	Get_Attribute_Single	Returns an array with a list of the attributes supported by this object in this device.	1,2,3,5,6, 7,8, 11,16, 100,101, 102,103, 104,105, 106,107	See Default column	1	ARRAY OF UNSIGNED SHORT INTEGERS
3	Axis Number	Get_Attribute_Single	Returns the axis number which is the same as the instance for this object. The value can be in the range from 0 to 7, but will always be 1 for the Gemini single axis controllers	1	0-7	1	UNSIGNED SHORT INTEGER
5	General Fault	Get_Attribute_Single	The logical OR of all fault condition attribute flags in the device. This bit is reset when the fault condition is removed. When active, this indicates a drive failure has occurred (bridge fault, overcurrent, etc.).	0	0 – No Fault 1 – Fault	1	BOOLEAN
6	Input Command Message Type	Get_Attribute_Single Set_Attribute_Single	Specifies which Command Message Type is used during polled I/O commands.	0	Dependent on Product 1 to 1F is Valid 0 is 'no I/O command'	1	UNSIGNED SHORT INTEGER
7	Output Command Message Type	Get_Attribute_Single Set_Attribute_Single	Specifies which Response Message Type is used during polled I/O commands.	1	Dependant on Product 1 to 1F is Valid 0 is 'no I/O command returned'	1	UNSIGNED SHORT INTEGER
8	Fault Input	Get_Attribute_Single	Displays the state of the Hardware Fault Input.	0	0 - Inactive 1 - Active	1	BOOLEAN
11	Home Active Level	Get_Attribute_Single Set_Attribute_Single	Sets the active level of the Hardware Home Input.	0	0 – Low 1 – High	1	BOOLEAN
16	Home Input Level	Get_Attribute_Single	Actual level of the Home Input.	0	0 – Off 1 – ON	1	BOOLEAN
100	Go Home (HOM)	Set_Attribute_Single	Used to initiate homing function in specified direction.	0	0 – Negative direction	1	BOOLEAN

					1 – Positive Direction		
101	Home Velocity (HOMV)	Get_Attribute_Single Set_Attribute_Single	Specifies initial homing velocity.	1.0000	0.0000 – 200.0000	0.0001	DOUBLE INTEGER
102	Home Final Velocity (HOMVF)	Get_Attribute_Single Set_Attribute_Single	Specifies final approach homing velocity	0.1000	0.0000 – 200.0000	0.0001	DOUBLE INTEGER
103	Home Acceleration (HOMA)	Get_Attribute_Single Set_Attribute_Single	Specifies accel/decel rate for next Go Home command	10.0000	0.0001-9999.9999	0.0001	DOUBLE INTEGER
104	Home Backup Enable (HOMBAC)	Get_Attribute_Single Set_Attribute_Single	Enables/disables backup to home switch function	0	0 – Disable 1 – Enable	1	BOOLEAN
105	Home Final Direction (HOMDF)	Get_Attribute_Single Set_Attribute_Single	Specifes final approach direction	0	0 – Positive Direction 1 – Negative Direction	1	BOOLEAN
106	Home Reference Edge (HOMEDG)	Get_Attribute_Single Set_Attribute_Single	Specifies which edge of the home switch the homing operation considers as its final destination.	0	0 – Positive Direction Edge 1 – Negative Direction Edge	1	BOOLEAN
107	Home to Encoder Z-Channel Enable (HOMZ)	Get_Attribute_Single Set_Attribute_Single	Enables homing to an encoder z-channel after initial home input has gone active.	0	0 – Disable 1 – Enable	1	BOOLEAN

Position Controller Object Class (0xA5)

The position controller object performs the control output velocity profiling and handles input and output to and from the motor drive unit, limit switches, registration, etc.

Object Instance Attributes

#	Attribute Name	Service Name	Description	Default	Range	Data Factor	Data Type
2	Attribute List	Get_Attribute_Single	Returns an array with a list of the attributes supported by this object in this device.	1,2,5,6,7,8, 9,10,11,13, 14,15,16, 17,20,21, 23,26,45, 48,52,54, 55, 58,100, 101,102, 103, 104, 105, 106, 107,108, 109,110, 111	Defined by attribute 1	1	ARRAY OF UNSIGNED SHORT INTEGERS
5	Profile Units	Get_Attribute_Single Set_Attribute_Single	Sets/returns the number of motor/feedback counts per second for velocity and acceleration/deceleration. This is the ERES command for servos and the DRES command for steppers.	1	-2^{31} to 2^{31}	1	DOUBLE INTEGER
6	Target Position (D)	Get_Attribute_Single Set_Attribute_Single	Sets/returns the target position in counts.	0	-2^{31} to 2^{31}	1	DOUBLE INTEGER
7	Target Velocity (V)	Get_Attribute_Single Set_Attribute_Single	Sets/returns the target velocity. The units are set by the PROFILE UNIT(5).	1	0.0000 - 200.0000	0.0001	DOUBLE INTEGER
8	Target Acceleration (A)	Get_Attribute_Single Set_Attribute_Single	Sets/returns the target acceleration. The units are set by the PROFILE UNIT(5).	10	0.0001 - 9999.9999	0.0001	DOUBLE INTEGER
9	Target Deceleration (AD)	Get_Attribute_Single Set_Attribute_Single	Sets/returns the target deceleration. The units are set by the PROFILE UNIT(5).	10	0.0001 - 9999.9999	0.0001	DOUBLE INTEGER
10	Incremental Position Flag (MA)	Get_Attribute_Single Set_Attribute_Single	Sets/returns incremental VS absolute mode.	0	0 - Absolute 1 - Incremental	1	BOOLEAN
11	Start a Profile Move (GO)	Get_Attribute_Single Set_Attribute_Single	Used to load command data, start a profile move, and indicate that a profile move is in progress. This will remain set until the motion is complete (or a hard/soft stop is issued).	0	0 - No Action 1 - Start Move	1	BOOLEAN
13	Actual Position (TPE/PSET)	Get_Attribute_Single Set_Attribute_Single	Reports actual position/Sets a new position.	0	-2^{31} to 2^{31}	1	DOUBLE INTEGER
14	Actual Velocity (TVELA)	Get_Attribute_Single	Gemini GV6-DN only. Reports the actual velocity.	n/a	-200.000000 to 200.000000	0.0001	DOUBLE INTEGER
15	Commanded Position (TPC)	Get_Attribute_Single	Reports position.	n/a	-2^{31} to 2^{31}	1	DOUBLE INTEGER
16	Commanded Velocity(TVEL)	Get_Attribute_Single	Reports velocity	n/a	-200.000000 to 200.000000	0.0001	DOUBLE INTEGER
17	Enable (DRIVE)	Get_Attribute_Single Set_Attribute_Single	Enables/disables motor current.	0	0 - Disable 1 - Enable	1	BOOLEAN
20	Smooth Stop (S1)	Get_Attribute_Single Set_Attribute_Single	Causes a controlled deceleration to a stop. This is the !S1 command.	0	0 - No action 1 – Start	1	BOOLEAN
21	Hard Stop (K)	Get_Attribute_Single Set_Attribute_Single	Causes a fast deceleration. This is the !K1 command.	0	0 - No action 1 - Kill	1	BOOLEAN

#	Attribute Name	Service Name	Description	Default	Range	Data Factor	Data Type
23	Direction (D-/D+)	Get_Attribute_Single Set_Attribute_Single	Specifies the direction of travel.	1	0 - Negative 1 - Positive	1	BOOLEAN
45	Maximum Dynamic Following Error (SMPER)	Get_Attribute_Single Set_Attribute_Single	Gemini GV6-DN only. This specifies the maximum following error allowed before a Following Error Fault.	4000	0 to 2^{31}	1	DOUBLE INTEGER
48	Actual Following Error (TPER)	Get_Attribute_Single	Gemini GV6-DN only. Reports the actual following error in counts.	n/a	0 to 2^{31}	1	DOUBLE INTEGER
50	Positive Hardware Limit (TAS.15)	Get_Attribute_Single	Motion is not allowed in the positive direction when active. Set when the positive limit is active.	0	0 - Positive limit not active 1 - Positive Limit Active	1	BOOLEAN
51	Negative Hardware Limit (TAS.16)	Get_Attribute_Single	Motion is not allowed in the negative direction when active. Set when the negative limit is active.	0	0 - Negative limit not active 1 - Negative limit active	1	BOOLEAN
52	Soft Limit Enable (LS)	Get_Attribute_Single Set_Attribute_Single	Enables Soft Limit checking.	0	0 - Disable 1 - Enable	1	BOOLEAN
54	Positive Soft Limit Position (LSPOS)	Get_Attribute_Single Set_Attribute_Single	Epecifies the positive soft limit position in Position Units.	0	-2^{31} to 2^{31}	1	DOUBLE INTEGER
55	Negative Soft Limit Position (LSNEG)	Get_Attribute_Single Set_Attribute_Single	Specifies the negative soft limit position in Position Units.	0	-2^{31} to 2^{31}	1	DOUBLE INTEGER
58	Load Data Complete	Get_Attribute_Single	Indicates that valid data for a valid I/O command message type has been loaded into the position controller device.	n/a	0 - Invalid Data 1 - Valid Data	1	BOOLEAN
100	Position Bandwidth (DPBW)	Get_Attribute_Single Set_Attribute_Single	Gemini GV6-DN only. Specifies Position loop bandwidth (Hz)	5.0	1.00 - 100.00	Norm-alized	INTEGER
101	Position Damping Ratio (SGPRAT)	Get_Attribute_Single Set_Attribute_Single	Gemini GV6-DN only SGPRAT command	1	0.500 - 2.000	Norm-alized	INTEGER
102	Load to Rotor Inertia Ratio (LJRAT)	Get_Attribute_Single Set_Attribute_Single	LJRAT command	0.0	0.0 - 100.00	0.1	INTEGER
103	Load Damping (LDAMP)	Get_Attribute_Single Set_Attribute_Single	Gemini GV6-DN only LDAMP command	0.0000	0.0000 - 1.0000	Norm-alized	INTEGER
104	Integrator Enable (SGINTE)	Get_Attribute_Single Set_Attribute_Single	Gemini GV6-DN only SGINTE command	1	0 - Disable 1 - Enable	1	BOOLEAN
105	Mode Continuous (MC)	Get_Attribute_Single Set_Attribute_Single	MC command	1	0 - Normal 1 - Mode Continuous	1	BOOLEAN
106	Input Status (TIN)	Get_Attribute_Single		n/a		1	INTEGER
107	Get/Set Outputs (TOUT/OUT)	Get_Attribute_Singl e Set_Attribute_Single		0		1	INTEGER
108	Variable Access - Variable number	Get_Attribute_Single Set_Attribute_Single	Allows access to VARI 1-99	1	1-99	1	INTEGER
109	Variable Access – Variable value	Get_Attribute_Single Set_Attribute_Single	Accesses contents of variable defined in #109	0	-2^{31} to 2^{31}-1	1	DOUBLE INTEGER
110	Hardware Limit (LH)	Get_Attribute_Single Set_Attribute_Single	LH command	0	0-3	1	INTEGER
111	Drive Status (TAS)	Get_Attribute_Single	Reports the drive status of the drive. See TAS command	n/a	0000_0000_ 0000_0000 to 1111_1111_	1	DOUBLE INTEGER

#	Attribute Name	Service Name	Description	Default	Range	Data Factor	Data Type
					1111_1111		
112	Error Status (TASX)	Get_Attribute_Single	Reports the error status of the drive. See TASX command	n/a	0000_0000_ 0000_0000 to 1111_1111_ 1111_1111	1	DOUBLE INTEGER
113	Positive Torque Limit (DMTIP)	Get_Attribute_Single Set_Attribute_Single	Gemini GV6-DN only. Specifies the maximum torque the motor can achieve.	depends on motor	0.00 - 128.00	Norm-alized	INTEGER

Command Assemblies

This section defines the command assembly types. This table shows the format for a Command Assembly I/O message.

Command Assembly

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	Direction (V Mode)	Absolute / Incremental	Start Program	Start Trajectory
1	RUN (0) / PRUN (1)	Program #						
2	Axis Instance (001)				Command Assembly Code			
3	Axis Instance (001)				Response Assembly Code			
4	Data Low Byte							
5	Data Low Middle Byte							
6	Data High Middle Byte							
7	Data High Byte							

Command	Description
Start Trajectory (Edge triggered)	Starts a move. Setting this bit to 1 in conjunction with command assembly 0x01 (positioning). For all other command assemblies, the transition of this bit sets the data value(i.e. velocity, acceleration, etc.).
Start Program (Edge triggered)	By setting this bit high (1) , the command will execute programs and profiles previously generated by Motion Planner and stored in the drive. The program executed is defined by the second byte of the Command Assembly. To stop program execution, set Start Program high (1) and Program Number to 0. Programs can be executed with any Command Assembly.
Absolute/ Incremental	Defines the position value as either absolute or incremental. This data is valid when the Start Trajectory bit is set. 0 = absolute position value; 1 = incremental or relative position value.
Direction (Velocity Mode)	Controls the direction of the motor in Profiled Velocity mode. This bit is only valid in 0x11 (velocity mode) command assembly. 1 = forward, positive or clockwise; 0 = reverse, negative or counter clockwise.
Smooth Stop (Level triggered)	Brings the motor to a controlled stop at the currently implemented deceleration rate.
Hard Stop (Level Triggered)	Brings the motor to an immediate stop.
Reg Arm (Level Triggered)	By setting this bit, the registration input will be armed. If a registration input is then triggered the registration action will be executed if a profile is currently being executed. Clearing this bit will turn off registration.
Enable (Level triggered)	Enables/disables the drive. Clearing this bit will disable the drive and the currently executing motion profile will be aborted. This attribute is level, not edge, sensitive.
Program #	Used in conjunction with Start Program to run a program or profile previously defined in the drive. With Start Program set to high (1) , the drive will execute the program indicated by the second byte of the Assembly. To stop a program, set Start Program high (1) and Program Number to zero (0). Programs can be executed with any Command Assembly.
Axis Instance	Always set to 1. Any other values should cause a command error.
Command Assembly Code	Defines the Command Assembly Code. The first four bytes of the command always have the same meaning. The last four bytes are defined by the command assembly code.
Response Assembly Code	Defines the Response Assembly Code. The first four bytes of the response always have the same meaning. The last four bytes are defined by the response assembly code.

Response Assemblies

This section defines the response assembly types. Here is the format for a Response Assembly I/O message. Each response assembly should echo the trajectory start bit from the previous 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 Direction	General Fault (Alarm)	On Target Position	Program In Execution	Trajectory In Progress
1	Executing Program # / Attribute number							
2	Load Complete	Program Fault	Trajectory Start Echo	Negative Software Limit	Positive Software Limit	CCW Hardware Limit	CW Hardware Limit	Fault Input Fault
3	Axis Instance (001)			Response Assembly Code				
4	Data Low Byte							
5	Data Low Middle Byte							
6	Data High Middle Byte							
7	Data High Byte							

Command	Description
Trajectory in Progress	Indicates whether or not a trajectory move is commanded or has completed. “0” indicates a move has completed. “1” indicates a move has been commanded. For Home routines, the trajectory in progress bit is used to determine home complete. The “Trajectory in Progress” bit must be set high during the data valid response to a successful command.
Program in Execution	Indicates that a Program is in execution. The Program that is currently being executed is returned in byte 1 of the response assembly.
On Target Position	Indicates whether or not the motor is within an end of move condition. The end of move condition can be position, velocity and/or time based. This bit is set to 1 indicates ‘In Position’. The attribute used to define the end of move position condition is specified in Object 0x25, attribute 38, position deadband.
General Fault (Alarm)	Set if a device alarm occurs; 1 = alarm, 0 = no alarms.
Current Direction	Shows the current direction of the motor. If the motor is not moving the bit will indicate the direction of the last commanded move: 0=reverse or negative direction and 1=forward or positive direction.
Home Level	The Home Flag; 1 = off home, 0 = on home.
Reg Level	Reflects the level of the registration input.
Enable State	Indicates whether the drive is enabled or disabled. A 1 indicates the device is enabled.
Fault Input Fault	Indicates the fault input is active.
CW/CCW Hardware Limit	Indicates that the hardware limits are active; 1 = active.
Negative/Positive Software Limit	Indicate that the motor has attempted a move past the programmed limit position; 1 = active.
Trajectory Start Echo	This is a copy of the Start Trajectory bit in the Command Assembly (bit 0 of byte 1).
Program Fault	Indicates that a program execution fault occurred. When this happens, the program/profile move will stop. 1 = Program Fault.
Load Complete	Indicates that the command data contained in the command message has been successfully loaded into the device.

Running a Stored Program through DeviceNet

1. Create and store a program in the drive with Motion Planner
2. Convert the program number from decimal (a.k.a. Prog32) to binary.

   For example, if you store a program as PROG5, this is converted into binary as 00000101 (1+4).
3. Save the binary value into byte 1 of the command assembly for standard programs. Add 128 to the binary value (ie set bit 7) for compiled programs, and save into byte 1 again.
4. Using any of the command assemblies, send out Start Program = 1.

Example:

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	X	X	X	X	X	1	X
1	0	0	1	0	0	0	0	0
2	X	X	X	X	X	X	X	X
3	X	X	X	X	X	X	X	X
4	X	X	X	X	X	X	X	X
5	X	X	X	X	X	X	X	X
6	X	X	X	X	X	X	X	X
7	X	X	X	X	X	X	X	X

This would run program 32. Bit 1 of byte 0 must transition from a “0” to a “1” for the drive to begin running the program.

Stopping the Program through DeviceNet

To stop the program, the user must send out a “0” in the Program Number with Start Program = 1.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	X	X	X	X	X	X	1	X
1	0	0	0	0	0	0	0	0
2	X	X	X	X	X	X	X	X
3	X	X	X	X	X	X	X	X
4	X	X	X	X	X	X	X	X
5	X	X	X	X	X	X	X	X
6	X	X	X	X	X	X	X	X
7	X	X	X	X	X	X	X	X

Command Assembly Codes

Available command assembly codes:

Code	Description	Data Type
0x00	No Operation	-
0x01	Target Position	DINT
0x02	Target Velocity	DINT
0x03	Acceleration	DINT
0x04	Deceleration	DINT
0x05 to 0x0B	Reserved	
0x0C	Variable Access	
0x0D to 0x10	Reserved	
0x11	Continuous Velocity	DINT
0x12	Home Type	USINT
0x13 to 19	Reserved	
0x1A	Position Controller Supervisor Attribute	DINT
0x1B	Position Controller Attribute	DINT
0x1C to 1D	Reserved	
0x1E	Alarm Clear	BOOL
0x1F	Reserved	

No Operation 0x00

This code does not have any meaning. The command data is ignored.

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

Target Position 0x01

The data bytes define the target position in units of steps (1 LSB = 1 step). This data is valid when the Valid Data bit is set.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
4	Target Position Low Byte							
5	Target Position Low Middle Byte							
6	Target Position High Middle Byte							
7	Target Position High Byte							

Target Velocity 0x02

The data bytes define the target velocity in units of steps/sec (1 LSB = 1 step/sec). This data is valid when the Valid Data bit is set. The data is always a positive number.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
4	Target Velocity Low Byte							
5	Target Velocity Low Middle Byte							
6	Target Velocity High Middle Byte							
7	Target Velocity High Byte							

Acceleration 0x03

The data bytes define the acceleration in units of steps/sec^2 (1 LSB = 1 step/sec^2). This data is valid when the Valid Data bit is set. The data is always a positive number. The data is used for positioning, velocity and home modes.

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

Deceleration 0x04

The data bytes define the acceleration in units of steps/sec^2 (1 LSB = 1 step/sec^2). This data is valid when the Valid Data bit is set. The data is always a positive number. The data is used for positioning, velocity and home modes.

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

Variable Access 0x0C

Gemini variables can be stored via this command. The value in data byte 1 (range 1-99) determines which variable is modified. Data bytes 4-7 contain the double integer value to be stored in the variable. This data is valid when the Valid Data bit is set.

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	Direction (V Mode)	Absolute / Incremental	Start Program	Start Trajectory
1	DeviceNet variable 1 through 99							
2	0	0	1	0	0x0C			
3	0	0	1	0	Response Assembly Code			
4	DeviceNet variable Value Low Byte							
5	DeviceNet variable Value Low Middle Byte							
6	DeviceNet Variable Value High Middle Byte							
7	DeviceNet variable Value High Byte							

Continuous Velocity 0x11

The data bytes define the continuous velocity in units of steps/sec (1 LSB = 1 step/sec). The direction of the velocity is defined by the Direction (Velocity Mode) bit. This data is valid when the Valid Data bit is set. The data is always a positive number.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
4	Continuous Velocity Low Byte							
5	Continuous Velocity Low Middle Byte							
6	Continuous Velocity High Middle Byte							
7	Continuous Velocity High Byte							

Homing 0x12

The data byte defines the Homing direction; 0 is positive, 1 is negative. The data is valid when the Valid Data bit is set. Home starts after setting the Start Trajectory bit. Bytes 5, 6 and 7 should be set to 0x00.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
4	Homing Direction							
5	0x00 (fixed)							
6	0x00 (fixed)							
7	0x00 (fixed)							

Position Controller Supervisor Attribute 0x1A

This assembly allows any attribute in the Position Controller Supervisor group to be set or accessed. The command assembly allows a set and a get function to be performed simultaneously.

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	Direction (V Mode)	Absolute / Incremental	Start Program	Start Trajectory
1	Attribute to Get							
2	0	0	1	1	0xA			
3	Attribute to Set							
4	Postion Controller Supervisor attribute value Low Byte							
5	Postion Controller Supervisor attribute value Low Middle Byte							
6	Postion Controller Supervisor attribute value High Middle Byte							
7	Postion Controller Supervisor attribute value High Byte							

Position Controller Attribute 0x1B

This assembly allows any attribute in the Position Controller group to be set or accessed. The command assembly allows a set and a get function to be performed simultaneously.

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	Direction (V Mode)	Absolute / Incremental	Start Program	Start Trajectory
1	Attribute to Get							
2	0	0	1	1	0xB			
3	Attribute to Set							
4	Postion Controller attribute value Low Byte							
5	Postion Controller attribute value Low Middle Byte							
6	Postion Controller attribute value High Middle Byte							
7	Postion Controller attribute value High Byte							

Alarm Clear 0x1E

The data bit clears the alarm occurring on the drive. Setting the Alarm Clear to 1 clears the alarm condition, on the rising edge of the "Start Trajectory" bit, with Data valid set high. Note, some alarm states can not be cleared using Alarm Clear 0x1E. In this case, the drive must be power cycled. This bit does not clear alarms due to communications errors. Sending Alarm Clear = 0 is equivalent to Command Assembly 0x00 (No Operation).

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
4	0 (fixed)	0 (fixed)	0 (fixed)	0 (fixed)	0 (fixed)	0 (fixed)	0 (fixed)	Alarm Clear
5	0x00 (fixed)							
6	0x00 (fixed)							
7	0x00 (fixed)							

Code	Description	Data Type
0x01	Actual Position - servo only	DINT
0x02	Commanded Position	DINT
0x03	Actual Velocity - servo only	DINT
0x04	Command Velocity	DINT
0x04 to 0x0B	Reserved	
0x0C	Variable access	DINT
0x0Dto 0x13	Reserved	
0x14	Command/Response Error	DINT
0x15to 0x19	Reserved	
0x1A	Position Controller Supervisor Attribute	DINT
0x1B	Position Controller Attribute	DINT
0x1E	Alarm Code	DINT
0x1F	Reserved	

Actual Position 0x01 - Servo Only

This double word reflects the actual encoder position in units of steps (1 LSB = 1 step).

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
4	Actual Position Low Byte							
5	Actual Position Low Middle Byte							
6	Actual Position High Middle Byte							
7	Actual Position High Byte							

Commanded Position 0x02

This double word reflects the commanded or calculated position in units of steps (1 LSB = 1 step).

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
4	Commanded Position Low Byte							
5	Commanded Position Low Middle Byte							
6	Commanded Position High Middle Byte							
7	Commanded Position High Byte							

Actual Velocity 0x03 - Servo Only

This double word reflects the actual velocity in units of steps/sec (1 LSB = 1 step/sec). The data is always a positive number.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
4	Actual Velocity Low Byte							
5	Actual Velocity Low Middle Byte							
6	Actual Velocity High Middle Byte							
7	Actual Velocity High Byte							

Commanded Velocity 0x04

This double word reflects the commanded velocity in units of steps/sec (1 LSB = 1 step/sec). The data is always a positive number.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
4	Commanded Velocity Low Byte							
5	Commanded Velocity Low Middle Byte							
6	Commanded Velocity High Middle Byte							
7	Commanded Velocity High Byte							

Variable access 0x0C

Gemini variables can be read via this command. The value in command assembly data byte 1 (range 1-99) determines which variable is read. Response assembly bytes 4-7 contain the double integer variable value.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
4	DeviceNet Variable Value Low Byte							
5	DeviceNet variable Value Low Middle Byte							
6	DeviceNet variable Value High Middle Byte							
7	DeviceNet Variable Value High Byte							

Command/Response Error 0x14

This assembly indicates whether there was an error in the command or response assembly, along with an error code describing the problem.

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 Direction	General Fault (Alarm)	On Target Position	Program In Execution	Trajectory In Progress
1	Executing Program #							
2	Load Complete	Program Fault	Trajectory Start Echo	Negative Software Limit	Positive Software Limit	CCW Hardware Limit	CW Hardware Limit	Fault Input Fault
3	Axis Instance (001)			Response Message Code (0x14)				
4	General Error code							
5	Additional Code							
6	Copy of Command Message Byte 2							
7	Copy of Command Message Byte 3							

The following is an explanation of the various error codes reported by this command:

General Error Code	Additonal Code	Response	Semantics
0x08	0x01	Service Not Supported	Command Message type not supported. Additional code 0x01 takes precedence over code 0x02
	0x02	Service Not Supported	Response type not supported
0x05	0x01	Path Destination Unknown	Command message axis number invalid
	0x02	Path Destination Unknown	Response message axis number invalid
0x09	0xFF	Invalid Attribute Value	Load value out of range
0x0E	0xFF	Attribute not Settable	A request to modify a non-modifiable attribute was received
0x13	0xFF	Not Enough Data	I/O command message contained fewer than 8 bytes.
0x14	0xFF	Attribute Not Supported	Attribute specifed in request was not supported

Position Controller Supervisor Attribute 0x1A

This assembly reflects the value of the attribute in command assembly data byte 1 when accessing the Position Controller Supervisor.

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 Direction	General Fault (Alarm)	On Target Position	Program In Execution	Trajectory In Progress
1	Attribute number							
2	Load Complete	Program Fault	Trajectory Start Echo	Negative Software Limit	Positive Software Limit	CCW Hardware Limit	CW Hardware Limit	Fault Input Fault
3	Axis Instance (001)			Response Message Code (0x1A)				
4	Position Controller Supervisor Low Byte							
5	Position Controller Supervisor Low Middle Byte							
6	Position Controller Supervisor High Middle Byte							
7	Position Controller Supervisor High Byte							

Position Controller Attribute 0x1B

This assembly reflects the value of the attribute in command assembly data byte 1 when accessing the Position Controller.

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 Direction	General Fault (Alarm)	On Target Position	Program In Execution	Trajectory In Progress
1	Attribute number							
2	Load Complete	Program Fault	Trajectory Start Echo	Negative Software Limit	Positive Software Limit	CCW Hardware Limit	CW Hardware Limit	Fault Input Fault
3	Axis Instance (001)			Response Message Code (0x1B)				
4	Position Controller Low Byte							
5	Position Controller Low Middle Byte							
6	Position Controller High Middle Byte							
7	Position Controller High Byte							

Status Code 0x1E

Indicates the status code. The functional mapping should follow the TAS command in the Gemini controller.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
4	Status Code							
5	Status Code							
6	Status Code							
7	Status Code							

Alarm Code 0x1F

Indicates the alarm code. A 0x00000000 alarm code is when no alarms have occurred. The specification doesn't require a specific mapping between alarms to alarm codes; however, a list of the alarm conditions are in the table below.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
4	Alarm Code							
5	Alarm Code							
6	Alarm Code							
7	Alarm Code7							

Alarm Codes

Alarm Code (low byte)	Alarm Description	Cleared by Alarm Clear
0x01	Motor temperature fault	
0x02	Low voltage fault	
0x03	Drive over-temperature fault	
0x04	Duplicate MAC ID Error	
0x05	Encoder failed	
0x06	Dual Port Ram Error	
0x07	Motor configuration error	
0x08	Incoming steps during startup	
0x09	Velocity Error limit	
0x10	Bridge Fault	
0x11	Bridge temperature fault	
0x12	Over-voltage	
0x13	Bus Off Error	
0x14	EEPROM Error	
0x15	Memory Error	
0x16	Over Current	
0x17	Stall Detected	
0x18	Override mode/Commanded velocity error	
0x19	Bridge is in foldback mode	
0x20	Power Disapation circuit has been active	
0x21	Bad hall state detected	

Alarm Code (low byte)	Alarm Description	Cleared by Alarm Clear
0x22	Unrecognized hardware	
0x23	User fault	
0x24	Keep alive active	
0x25	Power Disapation circuit fault	
0x26	Reserved	
0x27	Reserved	
0x28	Motor configuration warning	
0x29	ORES failure	
0x30	Motor thermal model fault	
0x31	Commanded torque/force is at limit	
0x32	Reserved	

This specification defines the following operating modes**:**

Positioning

Figure 1 describes the positioning mode. Note, all messages and I/O states are from the device prospective. The general positioning procedure is as follows:

1. Specify the relevant move parameters to the driver (target velocity 0x02, acceleration 0x03, and deceleration 0x04).
2. Specify the Target Position (0x01), either Absolute/Incremental positioning, and set the Start Trajectory bit.

The following is an example Command Assembly for positioning mode. For this example, the position data is 128,000 steps or 0x0001F400 and the move is absolute. The device is asked to respond with Actual Velocity 0x03.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1 Enable	0 Reg Arm	0 Hard Stop	0 Smooth Stop	0 Direction (V Mode)	0 Absolute / Incremental	0 Start Block	1 Start Trajectory
1	0x00							
2	001 Axis Instance			0x01 Command Assembly Code				
3	001 Axis Instance			0x03 Response Assembly Code				
4	0x00 Target Position Low Byte							
5	0xF4 Target Position Low Middle Byte							
6	0x01 Target Position High Middle Byte							
7	0x00 Target Position High Byte							

The Trajectory In Progress bit is set when the move begins. The bit is cleared when the pre-programmed in-position conditions have been met. The positioning mode is disabled when an alarm is reported.

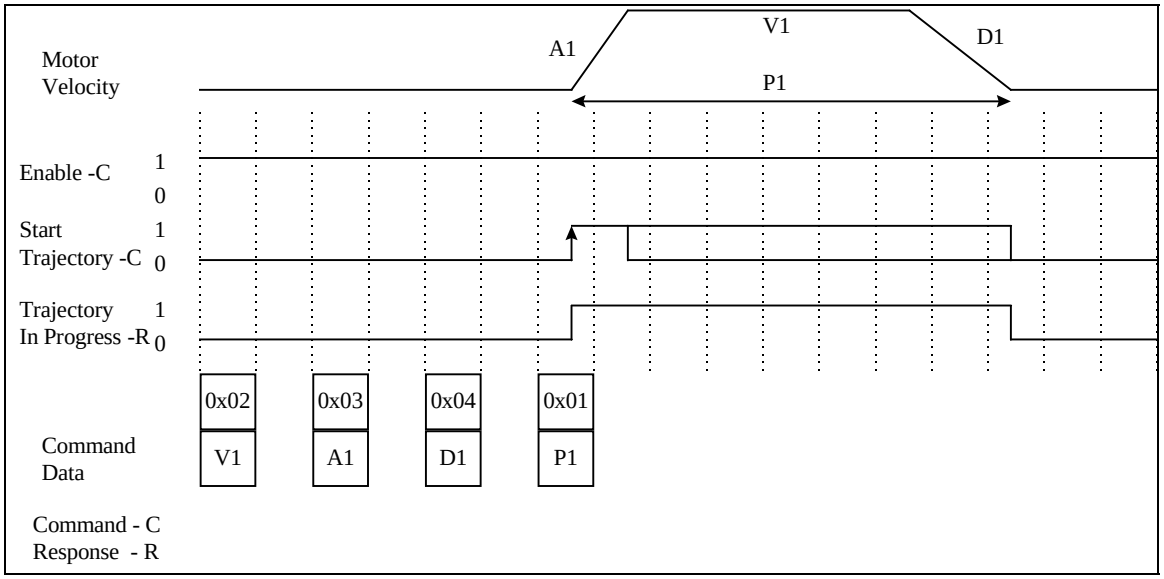


Figure 1. Positioning Mode

Continuous Velocity

Figure 2 describes the continuous velocity mode. The general method for defining a continuous velocity move is as follows:

1. Specify the relevant move parameters to the driver (acceleration 0x03, and deceleration 0x04)

Specify the Continuous Velocity (0x11), the direction bit, and set the Start Trajectory bit.

To stop the continuous velocity move, send a zero velocity move, set the smooth stop or hard stop bits, or toggle the direction bit (this will perform a hard stop).

The Trajectory In Progress bit is set when the move begins. The bit is cleared when the pre-programmed in-position conditions have been met. In Velocity Mode, the velocity and direction data can be modified while the move is occurring.

The following is an example Command Assembly for the Continuous Velocity mode. For this example, the direction is positive and the target velocity is 0x06666666 or 10rps (0x7FFFFFF is 200rps). The device is asked to respond with Actual Velocity 0x03.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1 Enable	0 Reg Arm	0 Hard Stop	0 Smooth Stop	1 Direction (V Mode)	0 Absolute / Incremental	0 Start Block	1 Start Trajectory
1	0x00							
2	001 Axis Instance			0x11 Command Assembly Code				
3	001 Axis Instance			0x03 Response Assembly Code				
4	0x66 Target Velocity Low Byte							
5	0x66 Target Velocity Low Middle Byte							
6	0x66 Target Velocity High Middle Byte							
7	0x06 Velocity Position High Byte							

The Trajectory In Progress bit is set when the move begins. The bit is cleared when the motor is stopped by setting the Smooth Stop bit. The continuous velocity mode is disabled when an alarm is reported.

By sending another continuous velocity command with another target velocity the motor will accelerate or decelerate to the new velocity. In order to change direction a soft or hard stop must be issued and the motor must come to rest before the new direction (“0” in this case) is issued and the ‘Start Trajectory’ bit is again set to “1” to initiate the move.

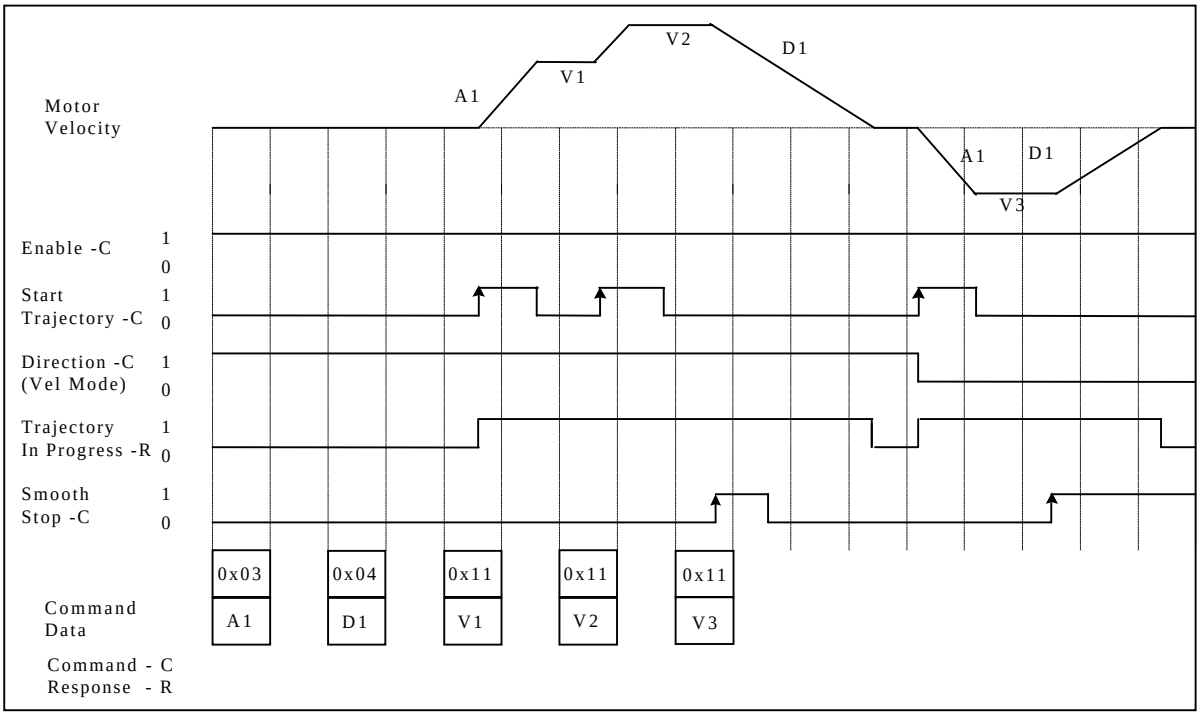


Figure 2. Continuous Velocity

Homing

Homing is done by sending the Home Type Command Assembly (0x12), setting the home direction and start trajectory bit.

Example 1

Figure 3 presents example 1. This Home method uses the home flag and the index or Z phase of the encoder to define home.

Object 0x24, Object instance attributes 100-107 will define the velocity and direction attributes to perform the home. Figure 3 shows the attributes set-up in the following fashion: HOMZ1 (use encoder z-pulse) HOMBAC1 (enable back-up to home flag), HOMDF1 (final direction negative) and HOMDG0 (stop on positive travel side of active region). and.

The general procedure is as follows:

1. Move in a positive direction towards the home flag direction.
2. When the home flag changes state, reverse direction until the home flag returns to the original state.
3. Once the motor has moved past the home flag, continue moving until the first index pulse of the encoder.
4. Bring motor to a stop, perform a residual move back to the home indicator. This is HOME; Position = 0. Other index pulse may occur during deceleration, these should not be used as home.

Note: The Trajectory In Progress bit reports when Home is complete.

The following is an example Command Assembly for the Homing command. This example assumes the homing parameters have been set up as above. The drive is asked to Home in a positive direction. The device is asked to respond with Actual Velocity 0x03. Homing is complete when the Trajectory in Progress bit returns to "0"

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1 Enable	0 Reg Arm	0 Hard Stop	0 Smooth Stop	0 Direction (V Mode)	0 Absolute / Incremental	0 Start Block	1 Start Trajectory
1	0x00							
2	001 Axis Instance			0x12 Command Assembly Code				
3	001 Axis Instance			0x03 Response Assembly Code				
4	0x00 Home Direction							
5	0x00							
6	0x00							
7	0x00							

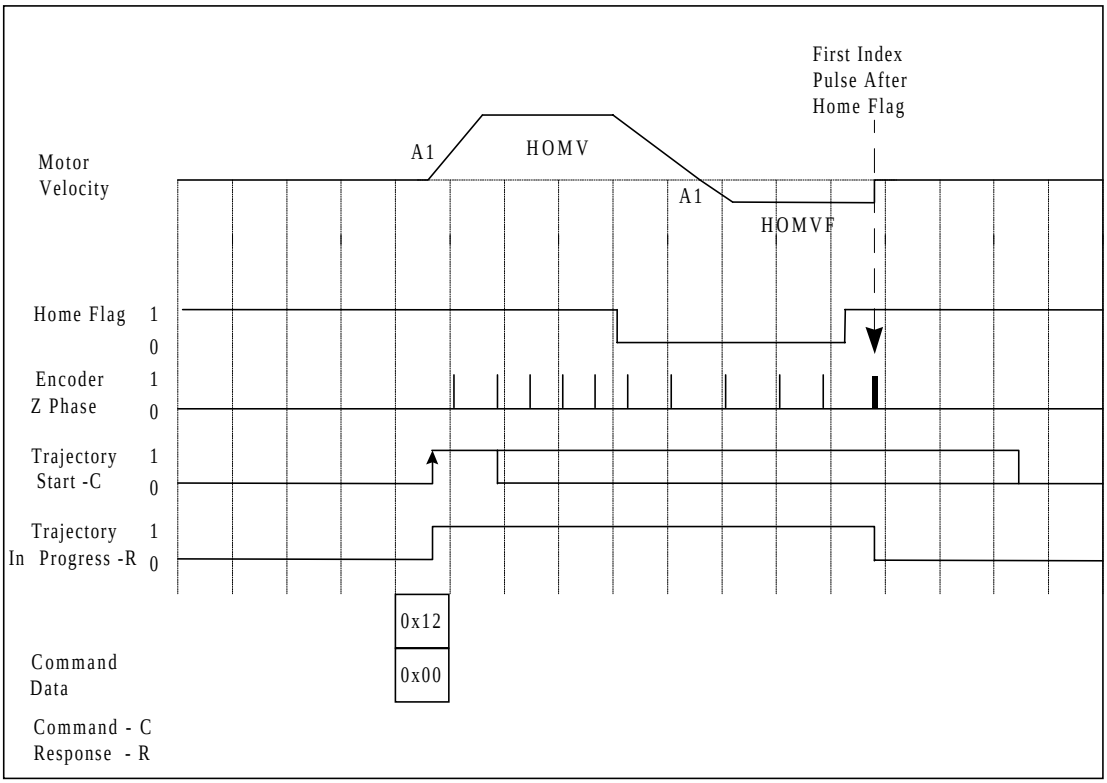


Figure 3. Home Example 1

Example 2

Figure 4 presents example 2. This Home method uses the home flag to define home.

Object 0x24, Object instance attributes 100-107 will define the velocity and direction attributes to perform the home. Figure 3 shows the attributes set-up in the following fashion: HOMZ0 (do not use encoder z-pulse) HOMBAC1 (enable back-up to home flag), HOMDF0 (final direction positive) and HOMDG1 (stop on negative travel side of active region).

The general procedure is as follows:

1. Move towards the home flag transition.
2. When the home flag changes state, reverse direction until the home flag returns to the original state.
3. Move forward again until the home flag transition.
4. This is HOME; Position = 0.

Note: The Trajectory In Progress bit reports when home is complete.

The following is an example Command Assembly for the Homing command. This example assumes the homing parameters have been set up as above. The drive is asked to Home in a negative direction. The device is asked to respond with Actual Velocity 0x03. Homing is complete when the Trajectory in Progress bit returns to "0"

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1 Enable	0 Reg Arm	0 Hard Stop	0 Smooth Stop	0 Direction (V Mode)	0 Absolute / Incremental	0 Start Block	1 Start Trajectory
1	0x00							
2	001 Axis Instance			0x12 Command Assembly Code				
3	001 Axis Instance			0x03 Response Assembly Code				
4	0x01 Home Direction							
5	0x00							
6	0x00							
7	0x00							

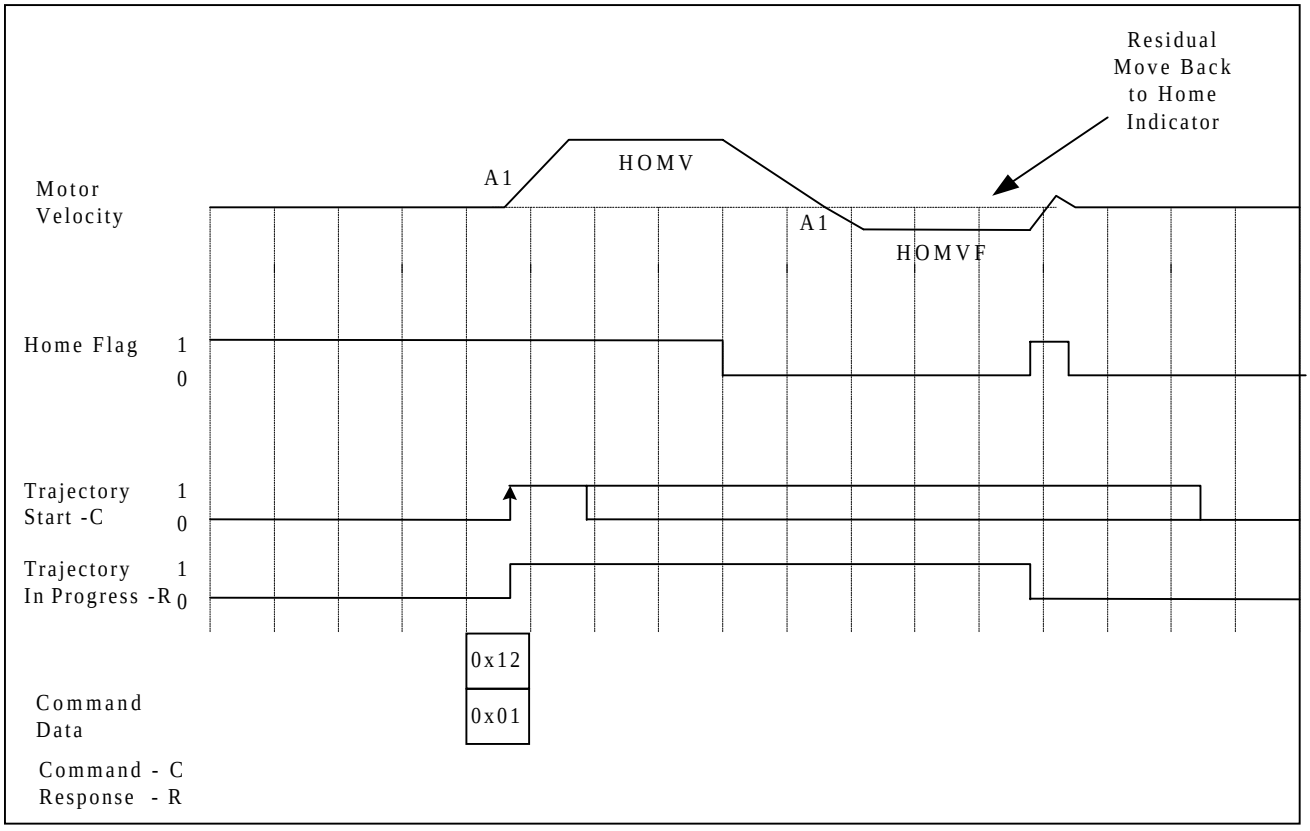


Figure 4. Home Example 2

The following is a list of all the DeviceNet specific commands. For a complete listing of Gemini commands see the *Gemini Series Programmer's Guide*.

ERROR — Error Checking Enable

		Product	Rev
Type:	Communication Setup		
Syntax:	<a_><!>ERROR<b><b>…<b><b> (32bits)	GV6-DN	2.00
Units:	n/a	GT6-DN	
Range:	b=0 (disable), 1 (enable), or X (don't change)		
Default:	0		
Response:	ERROR: *ERROR0000_0000_0000_0000_0000_0000_0000_0000		
See Also:	[ASX], ERRORP, INFNC, LH, LS, S, TASX, TER, TRGFN		

A new bit assignment is added for the ERROR command, bit #19.

See FBS on page 62 for the event causing the error condition. To clear the error event, first resolve the cause, and then issue the ERROR.19-0 command followed by the ERROR.19-1 command. Error bit 19 is edge sensitive to error events.

Bit#	Function	Branch Type
19	Option card fault	GOTO

FBADDR — Fieldbus Address

		Product	Rev
Type:	Communication Setup		
Syntax:	<!>FBADDR<i>	GV6-DN	2.00
Units:	i = fieldbus address	GT6-DN	
Range:	i = 1-125		
Default:	1		
Response:	FBADDR: *FBADDR3		
See Also:	FBSIZE		

Use the FBADDR command to report the controller's current node address assignment and set the node address via software. The new value is saved into nonvolatile memory, and becomes effective after the controller is reset. Network configuration of node address is not supported. This command cannot report the hardware configuration setting. In order to set the node address via software, the hardware configuration method must be disabled (default from factory). See Node Address on page 26.

Example:
Assume controller was assigned node address 1 out of reset:

```
>FBADDR
*FBADDR1
>FBADDR3        Set node address to 3
>FBADDR
*FBADDR3        New network setting will take effect after unit is reset!
```

FBBAUD Fieldbus Baudrate Setting

		Product	Rev
Type:	Communication Setup	GV6-DN	2.00
Syntax:	<!>FBBAUD<r>	GT6-DN	
Units:	n/a		
Range:	r = 0 (125kbaud) , 1 (250kbaud), or 2 (500kbaud)		
Default:	2 (500kbaud)		
Response:	FBBAUD: *FBBAUD2		
See Also:	FBADDR, FBSIZE		

Use the FBBAUD command to report the controller's current baudrate assignment (used during boot-up) and set the baudrate via software. This command cannot report the hardware configuration setting. In order to set the baudrate via software, the hardware configuration method must be disabled (default from factory). See Baud Rate on page 26.

When setting the baudrate via software, the node address must also be set via software, and vice versa. If the hardware configuration method is used to set the baudrate, an attempt to set the baudrate via software will be ignored. No message is reported indicating success or failure.

The new value is saved into nonvolatile memory, and becomes effective after the controller is reset.

Example:

```
>FBBAUD
*FBBAUD1
>FBBAUD1        ;Set baudrate to 250 kbaud
>FBBAUD
*FBBAUD1        ;New network setting will take effect after unit is reset!
```

FBMASK Fieldbus I/O Mask

		Product	Rev
Type:	I/O Setup	GV6-DN	2.00
Syntax:	<!>FBMASK <b>,<b>,<b>,<b>,<b>,<b>,<b>,<b>	GT6-DN	
Units:	n/a		
Range:	b=0 (Gemini control), 1 (Fieldbus control)		
Default:	0		
Response:	FBMASK: *FBMASK0000_0000		
See Also:	FBPIC, FBPOC		

Use FBMASK to prevent an output from being changed by fieldbus.

Example:

FBMASK00100000 Only output #3 can be changed by fieldbus

Response for FBMASK: *FBMASK0010_0000

[FBS] Fieldbus Status

		Product	Rev
Type:	Communication Setup	GV6-DN	2.00
Syntax:	See below	GT6-DN	
Units:	n/a		
Range:	n/a		
Default:	n/a		
Response:	n/a		
See Also:	ER.19, TFBS		

Use the FBS command to assign the fieldbus status to a binary variable or for use in a comparison command.

Note: The left-most bit corresponds to OUT.1

Example:

```
IF(FBS.4-1)     ;Branch based on the status of FBS bit 4
```

The Fieldbus Status register bits are defined as follows:

Bit #	Function (1=Yes, 0=No)	Description
1	TIMEOUT [1,2]	Watchdog timed out. Controller has lost communication with fieldbus card
2	CHECKSUM FAULT [1,2]	Fieldbus card failed hardware check on boot-up
3	HWD CFG MODE	0 - Configuration set via software, 1 - Configuration set via hardware
4	NETWORK LINK OK [1,3]	Controller is connected and data exchange is possible
5	NETWORK CRITICAL LINK FAILURE[4]	Cannot communicate with the fieldbus. Duplicate node address or bus-off state exists.
6	NETWORK LINK NOT CONNECTED[4]	Passed Dup_MAC_ID test and online, but no connection established with another node
7	NETWORK CONNECTION TIMEOUT[4]	Connection has timed out
8	MODULE DEVICE OPERATIONAL[4]	Operating in a normal condition
9	MODULE UNRECOVERABLE FAULT[4]	Fieldbus card may need to be replaced
10	MODULE DEVICE IN STANDBY[4]	Device needs commissioning due to configuration missing, incomplete, or incorrect
11	MODULE MINOR FAULT[4]	Fieldbus card is indicating a recoverable fault
12-31	RESERVED	
32	Incorrect Firmware	The DeviceNet card contains incorrect firmware.

[1] If any of these error conditions occur (bit #1 = 1, bit #2 = 1, or bit #4 = 0), the motion controller will perform a Kill (K command) on all axes. If error-checking bit #19 is enabled with the ERROR command (ERROR.19-1) the controller will also set error status bit #19 (ER, TER, and TERF) and branch to the ERRORP program.

[2] Error event is latched. Reset the controller to clear the error.

[3] Error event is recoverable, if error checking (ERROR) bit #19 is enabled and ERRORP program exists. If error bit #19 is disabled or no ERRORP program exists, the event becomes latched, and you will need to reset the controller to clear the error.

[4] After error checking (ERROR) bit #19 is set, it can take up to 600ms to correctly set these status bits.

OUTFNC Output Function

		Product	Rev
Type:	Output	GV6-DN	2.00
Syntax:	<!>OUTFNC<i>-<c>	GT6-DN	
Units:	i = output #,c = function identifier (letter)		
Range:	i = 1-7, c = A-G		
Default:	(see programmer's reference)		
Response:	OUTFNC1: *OUTFNC1-A PROGRAMMABLE OUTPUT - STATUS OFF		
See Also:	INFNC, OUT, OUTLVL, POUTA, SMPER, TAS, TASX, TOUT		

A new identifier is added for the OUTFNC command. The other function identifiers can be found in the Gemini Programmer's Reference p135-136.

Identifier	Function Description
I	**Option Card Fault**: Output activates when error bit #19 is set for the option card fault. See the ERROR command for description of events. This requires ERROR.19-1 to be set, or the output will not activate.

Example:

```
OUTFNC1-i      ;assign output 1 to option card fault
```

TASX Transfer Extended Axis Status

		Product	Rev
Type:	Communication Setup	GV6-DN	2.00
Syntax:	<a_><!>TASX	GT6-DN	
Units:	n/a		
Range:	n/a		
Default:	n/a		
Response:	TASX: *TASX 0001_0000_0000_0000_0000_0000_0000_0000		
See Also:	DCLRLR, DRIVE, RESET, TAS, TCS, TER		

A new bit assignment is added for the TASX command, bit #27.

See FBS on page 62 for the event causing the condition. To clear the status bit the drive must be reset or power must be cycled.

Bit#	Function	To Clear the status Bit
27	Fieldbus Fault	RESET or cycle power

TFBS Transfer Fieldbus Status

		Product	Rev
Type:	Communication Setup	GV6-DN	2.00
Syntax:	<!>TFBS<.i>	GT6-DN	
Units:	i = status bit number		
Range:	1-32		
Default:	n/a		
Response:	TFBS: TFBS.4: *1 (unit online or link ok, yes) *TFBS0001_0000_0000_0000_0000_0000_0000_0000		
See Also:	ER.19, [FBS]		

The TFBS command provides information on the 32 fieldbus status bits. The TFBS command reports a binary bit report.

Response for TFBS: *TFBS0001_0000_0000_0000_0000_0000_0000_0000

Bit#1…bit#32

For bit description, see FBS on page 62.

APPENDIX A

Gemini DeviceNet CE Compliance

IN THIS CHAPTER

CE Compliance

The Gemini family of products is designed to meet the requirements of global regulatory agencies.

Gemini products, when installed in accordance with instructions in the appropriate *Hardware Installation Guide*, are compliant with CE directives. This appendix gives instructions for additional steps necessary for installing the Gemini DeviceNet Option into a CE compliant system.

Installation Instructions

Gemini Drive – Installation Instructions

1. Follow the standard installation and configuration instructions in your Gemini drive's *Hardware Installation Guide*.
2. Follow additional installation instructions for CE compliance in *Appendix C – Regulatory Compliance* of your drive's *Hardware Installation Guide*.

Gemini DeviceNet Option – Installation Instructions

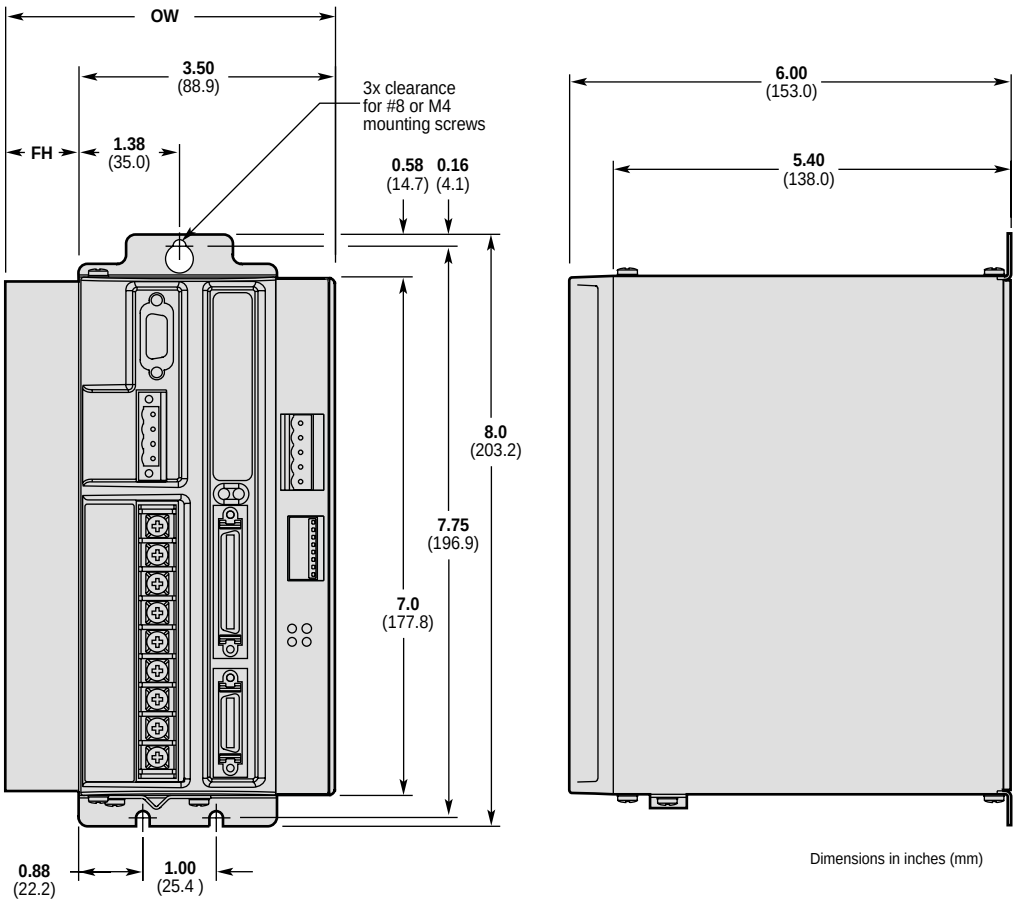
1. Install DeviceNet cabling and associated hardware in accordance with the DeviceNet specifications (available from www.odva.com). Technical papers discussing how to implement a DeviceNet network are also avilable at this web site.
2. Consult your DeviceNet Master's guide for additional requirements for the Master installation.

APPENDIX B

Gemini Dimensions

IN THIS CHAPTER

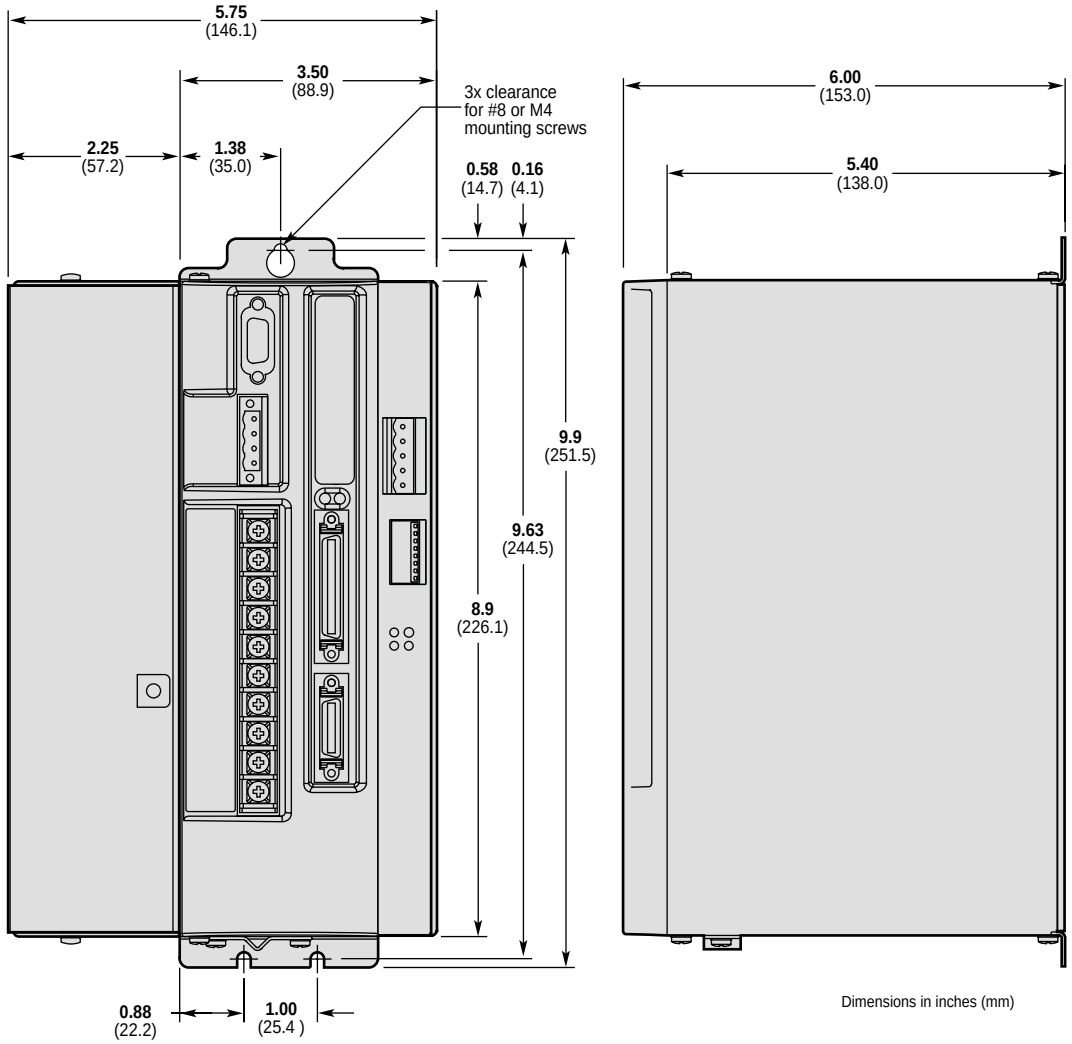
Gemini Drive Dimensions



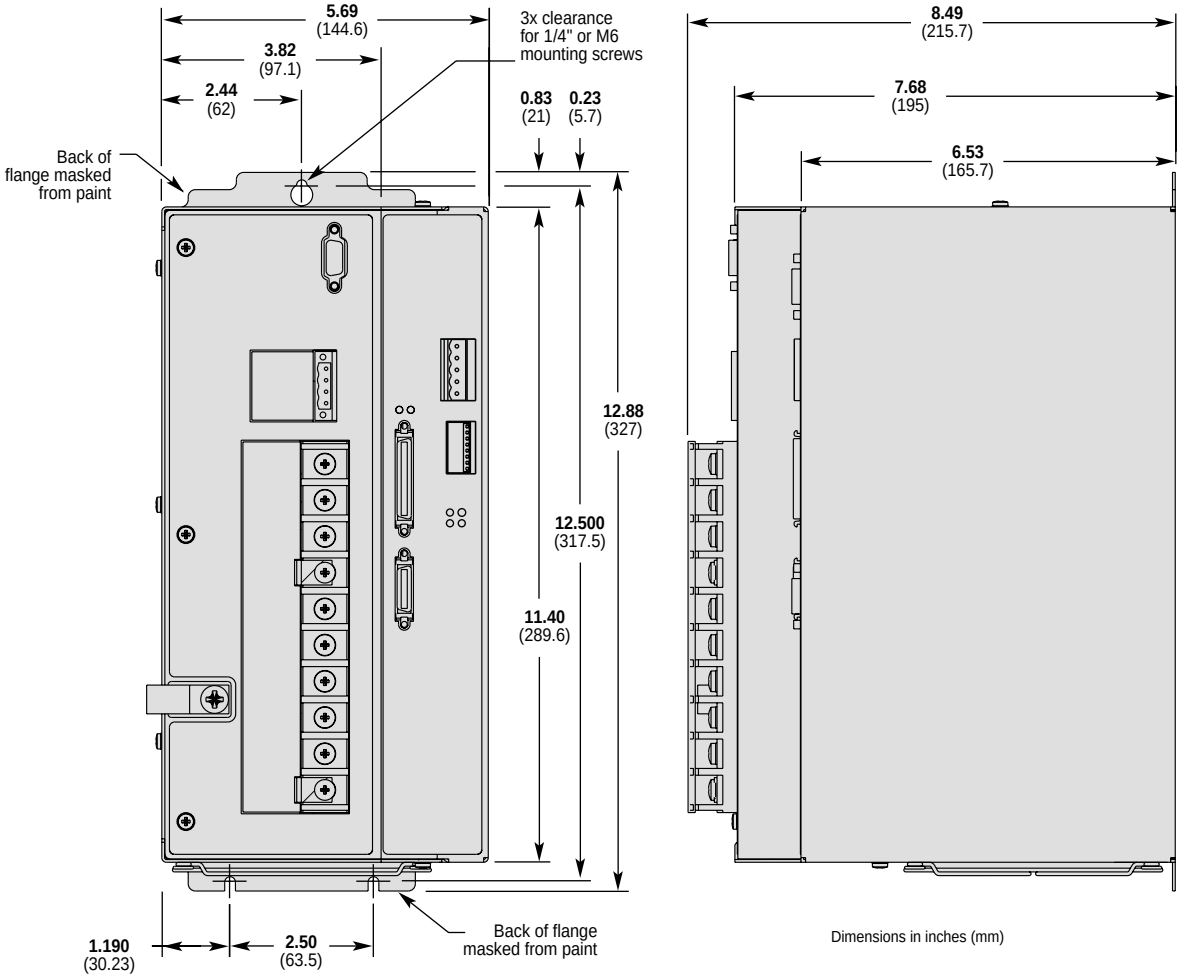
Product	OW Overall Width inches (mm)	FH Fin Height inches (mm)
GV6-L3n-DN	**3.88** (98.6)	**0.38** (9.5)
GV6-U3n-DN	**3.88** (98.6)	**0.38** (9.5)
GV6-U6n-DN	**4.50** (114.3)	**1.00** (25.4)
GV6-U12n-DN	**4.50** (114.3)	**1.00** (25.4)

GT6-L5-DN	**3.88** (98.6)	**0.38** (9.5)
GT6-L8-DN	**4.50** (114.3)	**1.00** (25.4)

Dimensions (Shorter Enclosure)

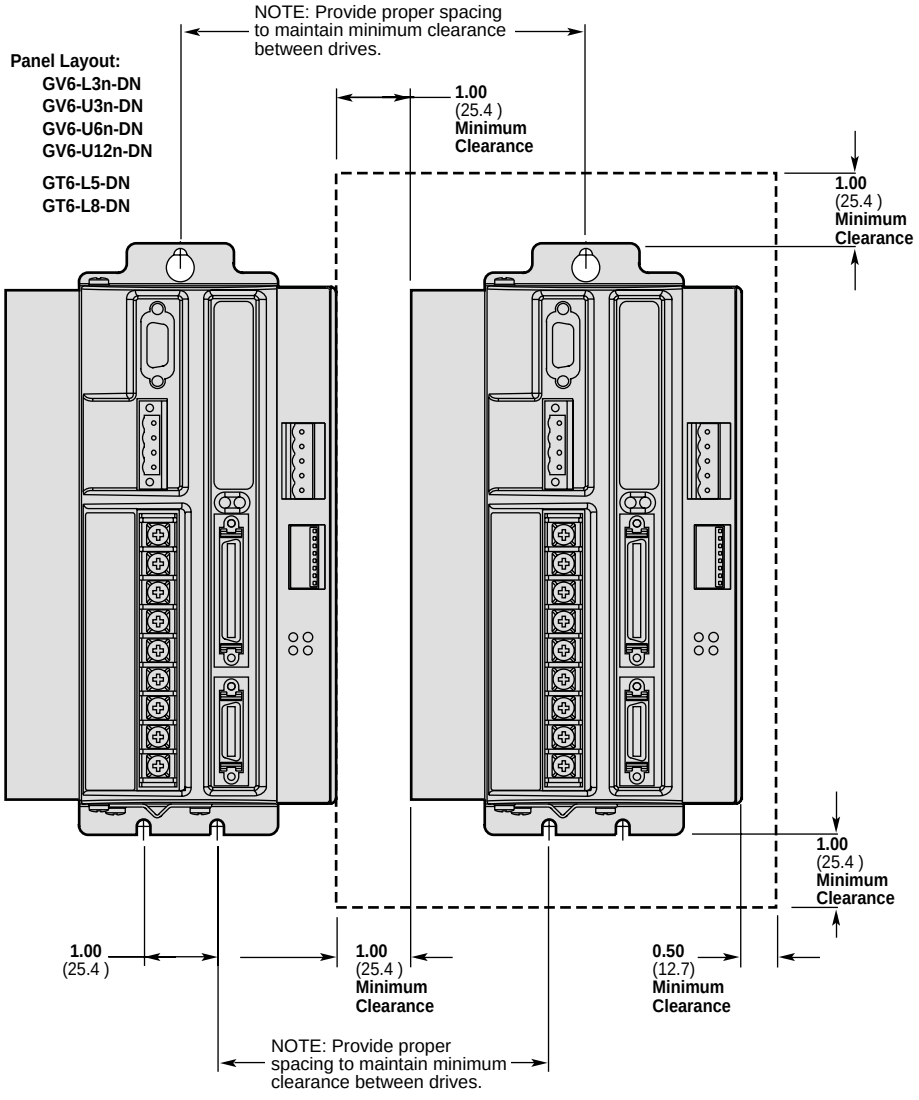


Dimensions – GV6-H20n-DN

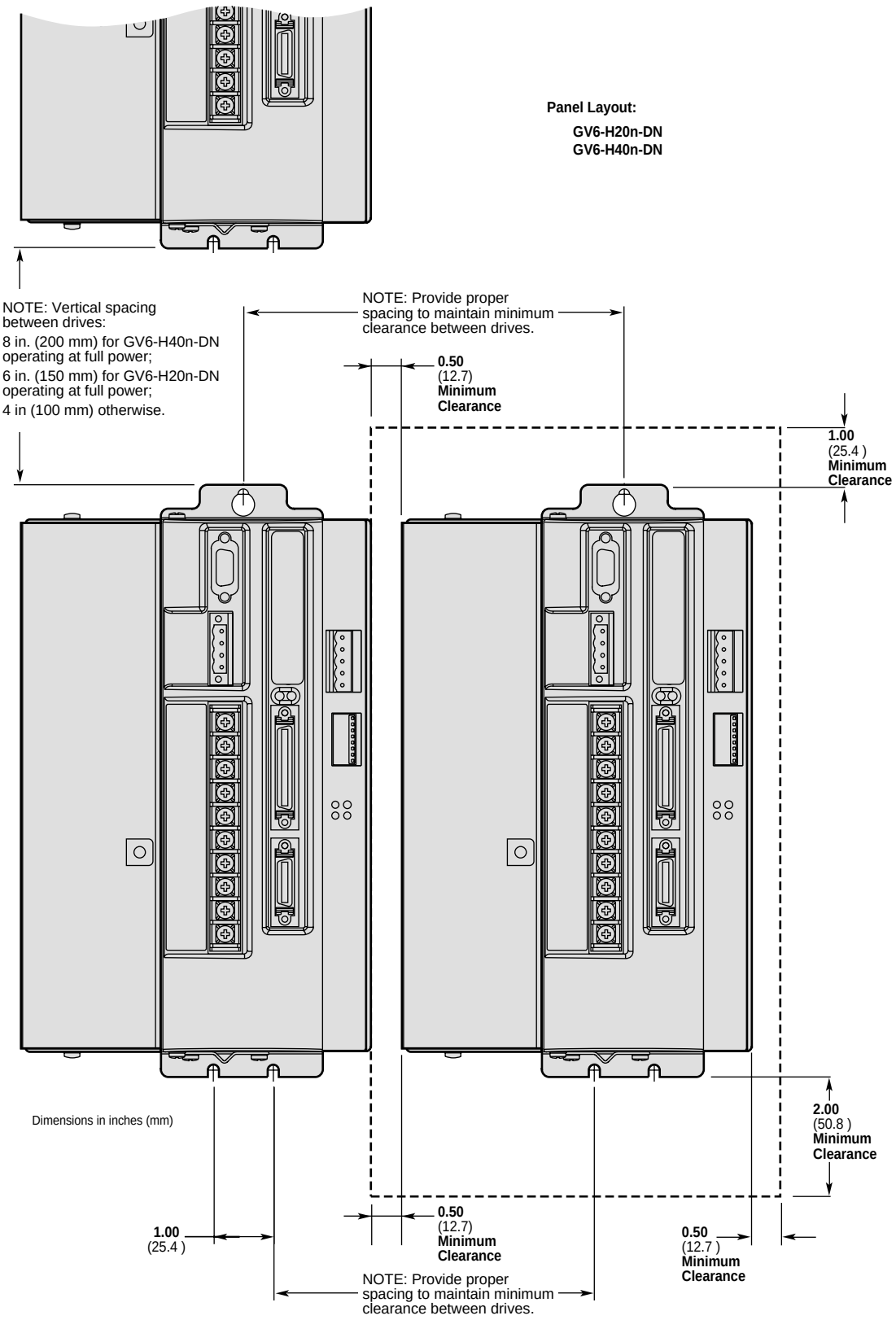


Dimensions – GV6-H40n-DN

Gemini Panel Layout Dimensions



Panel Layout Dimensions – Shorter Enclosure



Panel Layout Dimensions – Taller Enclosure