

ABOUT THIS GUIDE

Chapter 1. Implementing 6K DeviceNet..... 1

DeviceNet Overview	2
6Kn-DN.....	2
Technical Assistance	2
Implementation Process.....	2
EDS File	2
Hardware Interface	3
LED Status Indicators	3
DeviceNet Connector Pin Out	4
Termination.....	4
Node Address.....	4
Baud Rate.....	5
Programming Notes.....	6
Data Format.....	6
Implementing Data Exchange	6
Handling a DeviceNet Fault	7
Affected Commands and Features.....	7
DeviceNet Objects	8
Identity Object, Class 0x01.....	8
Message Router Object, Class 0x02	8
DeviceNet Object, Class 0x03.....	9
Assembly Object, Class 0x04.....	9
Connection Object, Class 0x05	10
Acknowledge Handler Object, Class 0x2B	12
Command Descriptions.....	13
ERROR Error Checking Enable	13
FBADDR Fieldbus Address	14
FBBAUD Fieldbus Baud Rate	14
[FBS] Fieldbus Status.....	15
FBSIZE Fieldbus Data Packet Size	16
OPTEN Option Card Enable/Disable	16
OUTFNC Output Function	17
TFBS Transfer Fieldbus Status.....	17
TFBSF Fieldbus Status Full Text	18
TOPSTS Option Card Status Full Text	18
Programming Scenario	19

Chapter 2. Implementing Gemini DeviceNet..... 21

DeviceNet Overview	22
Gemini GT6-DN and GV6-DN.....	22
Technical Assistance	22
Implementation Process.....	22
EDS File	22
Data Types	23
Hardware Interface	24
LED Status Indicators	24
DeviceNet Connector Pin Out	25
Termination.....	25
Node Address.....	26
Baud Rate.....	26
Configuration and Programming	27
Basic Requirements.....	27
Hardware Requirements	27
Object Model.....	27
Explicit Messaging	28
Position Controller Supervisor Object Class (#164)(A4hex) and Position Controller Object Class (#165)(A5hex).....	28
Explicit Message Command.....	28

Explicit Message Response	28
Services.....	29
Identity Object, Class 0x01	30
DeviceNet Object, Class 0x03.....	31
Assembly Object, Class 0x04.....	31
Connection Object, Class 0x05	32
Acknowledge Handler Object, Class 0x2B	34
Position Controller Object Class (0xA5)	37
Polled I/O.....	40
Command Assemblies	40
Response Assemblies	41
Running a Stored Program through DeviceNet	42
Stopping the Program through DeviceNet.....	42
Command Assembly Codes.....	43
No Operation 0x00.....	43
Target Position 0x01	43
Target Velocity 0x02	43
Acceleration 0x03	44
Deceleration 0x04	44
Variable Access 0x0C.....	44
Continuous Velocity 0x11.....	44
Homing 0x12.....	45
Position Controller Supervisor Attribute 0x1A.....	45
Position Controller Attribute 0x1B	45
Alarm Clear 0x1E.....	46
Response Assembly Codes.....	47
Actual Position 0x01 - Servo Only.....	47
Commanded Position 0x02	47
Actual Velocity 0x03 - Servo Only	48
Commanded Velocity 0x04	48
Variable access 0x0C.....	48
Command/Response Error 0x14.....	49
Position Controller Supervisor Attribute 0x1A.....	50
Position Controller Attribute 0x1B	50
Status Code 0x1E.....	51
Alarm Code 0x1F.....	51
Operating Modes	52
Positioning.....	52
Continuous Velocity.....	54
Homing.....	56
Command Descriptions.....	60
ERROR Error Checking Enable	60
FBADDR Fieldbus Address	60
FBBAUD Fieldbus Baudrate Setting.....	61
FBMASK Fieldbus I/O Mask.....	61
[FBS] Fieldbus Status.....	62
OUTFNC Output Function	63
TASX Transfer Extended Axis Status	63
TFBS Transfer Fieldbus Status.....	63

Appendix A. Gemini DeviceNet CE Compliance..... 65

CE Compliance.....	66
Installation Instructions	66

Appendix B. Gemini Drive Dimensions..... 67

Gemini Drive Dimensions.....	68
Gemini Panel Layout Dimensions	71

Change Summary – Revision B

August 1, 2002

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

Topic	Description
Document clarifications and corrections	<ul style="list-style-type: none">• 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 Acknowledge 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).

CHAPTER ONE

Implementing 6K DeviceNet

IN THIS CHAPTER

- DeviceNet Overview 2
- Hardware Interface 3
- Programming Notes 6
- DeviceNet Objects 8
- Command Descriptions 13
- Programming Scenario 19

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:

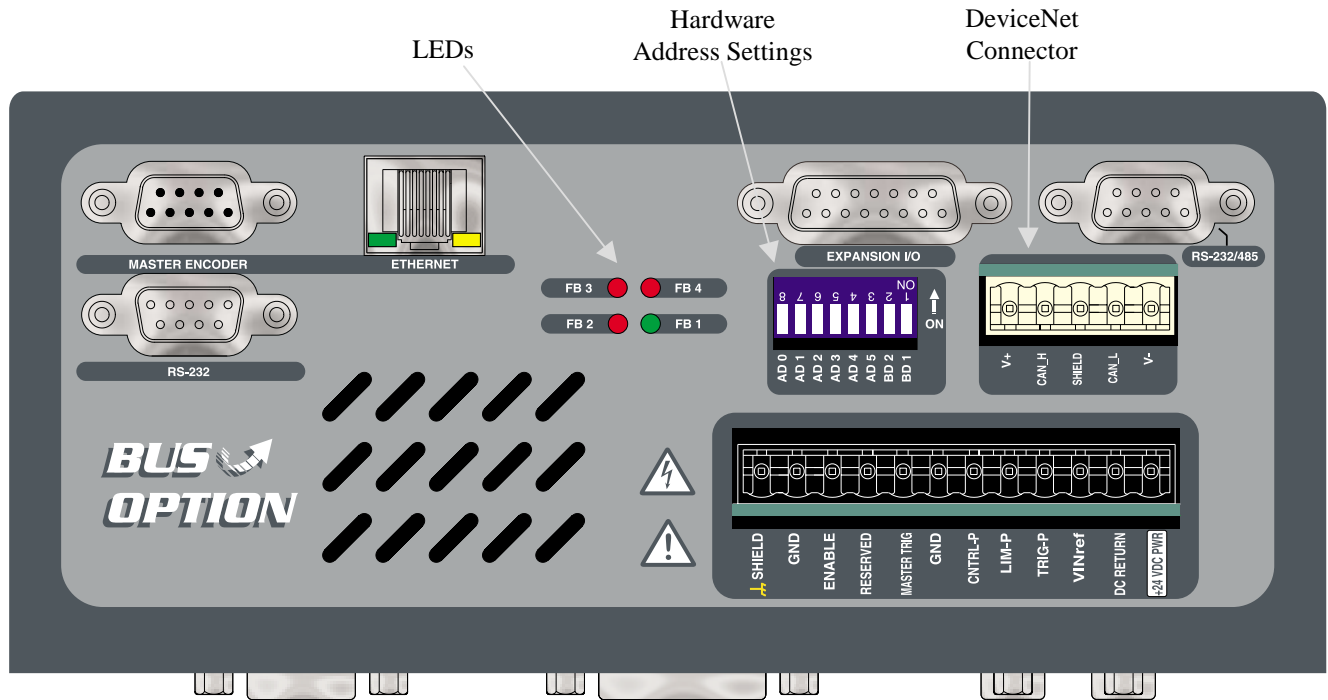
1. Install terminating resistors as needed (see page 4).
2. Launch Motion Planner (CD-ROM is provided in your ship kit).
3. Establish a direct communication link (serial) with the 6K. Refer to the *6K Series Hardware Installation Guide* for connection instructions.
4. Configure node address (see FBADDR 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.

Hardware Interface



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
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

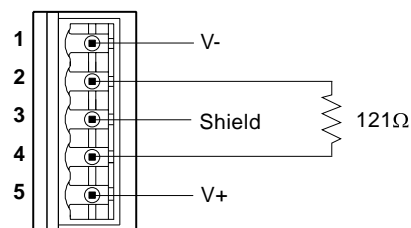
The following table gives the pin out for the DeviceNet connector. Industry standard DeviceNet connections are used.

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

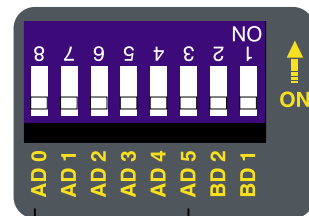
Termination

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).



Node address dip switches

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

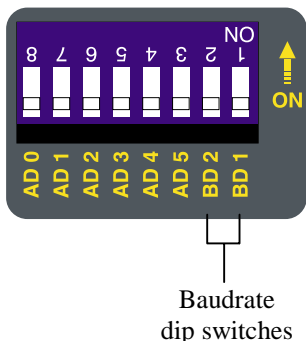
Example:

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 (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

Programming Notes

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 to 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	1 Explicit.....	See page 10
	1 Polled I/O.....	See page 10
	1 Change of state/Cyclic I/O.....	See page 11

Objects which are Implemented	Identity object, class 0x01.....	See page 8
	Message Router, class 0x02.....	See page 8
	DeviceNet object, class 0x03.....	See page 9
	Assembly object, class 0x04.....	See page 9
	Connection object, class 0x05.....	See page 10
	Acknowledge Handler object, class 0x2B.....	See page 12

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)..... See page 13
 FBADDR (Fieldbus Address) See page 14
 FBBAUD (Fieldbus Baud Rate Setting)..... See page 14
 [FBS] (Fieldbus Status)..... See page 15
 FBFSIZE (Fieldbus Data Size Packet)..... See page 16
 OPTEN (Option Card Enable/Disable) See page 16
 OUTFNC (Output Function)..... See page 17
 TFBS (Transfer Fieldbus Status)..... See page 17
 TFBSF (Fieldbus Status Full Text) See page 18
 TOPSTS (Option Card Status Full Text)..... See page 18

ERROR	Error Checking Enable		
Type:	Communication Setup	Product	Rev
Syntax:	<!><%>ERROR... (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_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

Type:	Communication Setup	Product	Rev
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

Type:	Communication Setup	Product	Rev
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

Type: Communication Setup
Syntax: See below
Units: n/a
Range: n/a
Default: n/a
Response: n/a
See Also: ER.19, TFBS, TFBSF

Product **Rev**
6Kn-DN 5.2

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 CONNECTED 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

Type:	Communication Setup	Product	Rev
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

Type:	Communication Setup	Product	Rev
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

Type: Output **Product** 6Kn **Rev** 5.2
Syntax: <!>OUTFNC<i><-<a>c>
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)
LOUTFNC: (function and status of outputs on I/O brick 1)
LOUTFNC1: *LOUTFNC1-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%LOUTFNC8-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

Type: Communication Status **Product** 6Kn-DN **Rev** 5.2
Syntax: <!>TFBS<.i>
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

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

Bit#1...bit#32

For bit description, see FBS on page 15.

TFBSF Fieldbus Status Full Text

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

Product **Rev**
6Kn-DN 5.2

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	RESERVED	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

Type: Option Status
Syntax: <!>TOPSTS
Units: n/a
Range: n/a
Default: n/a
Response: see example
See Also: OPTEN, TFBSF

Product **Rev**
6Kn-DN 5.2

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"

    VAR11=4PE          ;Assign axis 4 encoder position to VAR11
    VARB2=VCVT(VAR11) ;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.

```