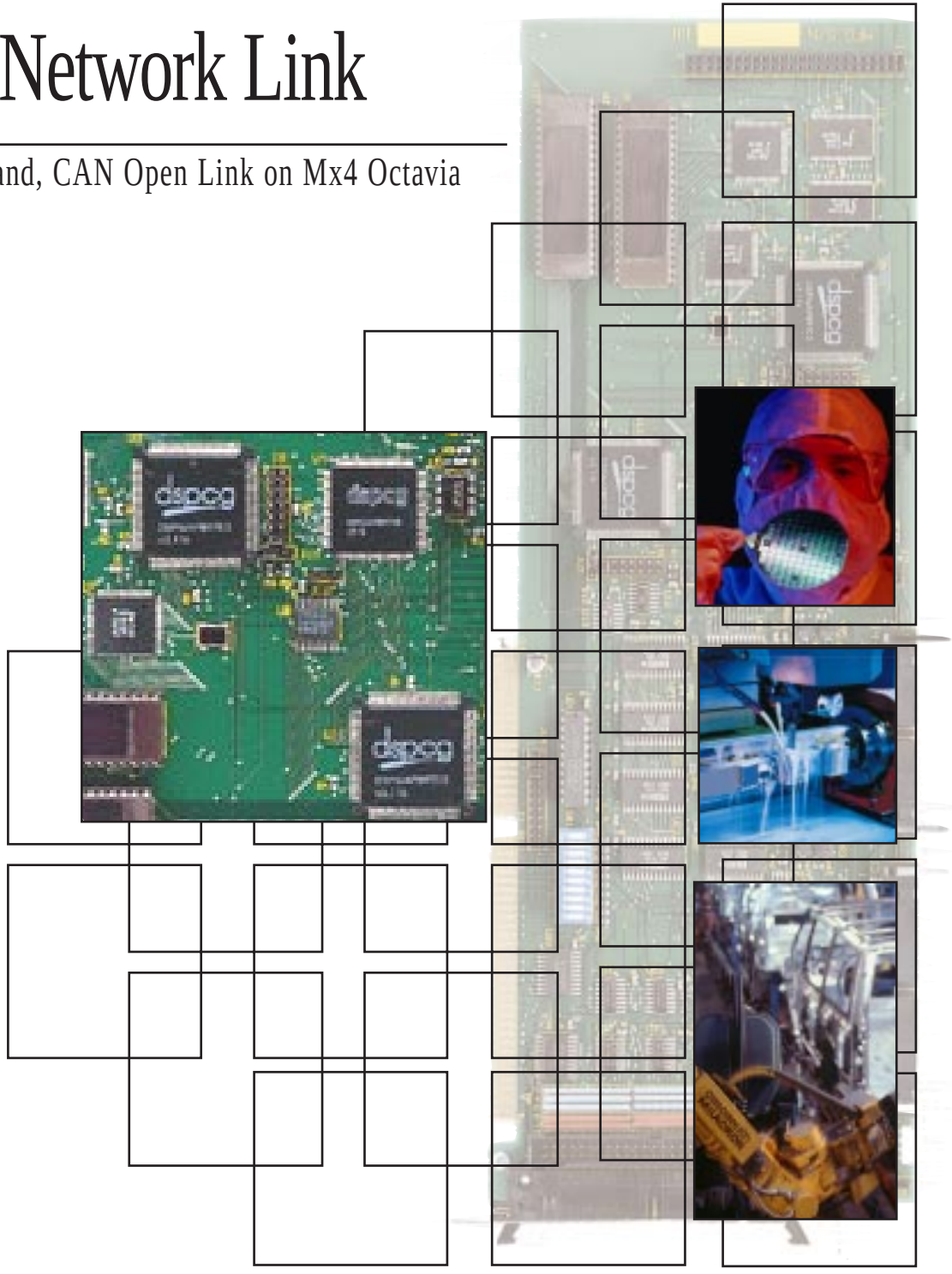
Mx4 Network Link

DeviceNet and, CAN Open Link on Mx4 Octavia



DSP Control Group, Inc.

Acc8 DeviceNet CAN Option Manual

User's Guide

v1.0

 DSP Control Group, Inc.
PO Box 39331
Minneapolis, MN 55439
Phone: (612) 831-9556
FAX: (612) 831-4697

Contents

This page intentionally blank.

1 Introduction

The ACC8 DeviceNet & CAN Option [Mx4 Octavia CAN/DeviceNet Adapter] enables you to use the Mx4 Octavia controller as a stand-alone unit or as a CAN/DeviceNet programmable unit inside the computer. The ACC8 DeviceNet & CAN Option board works with both PC/AT Mx4 Octavia and VME Mx4 Octavia.

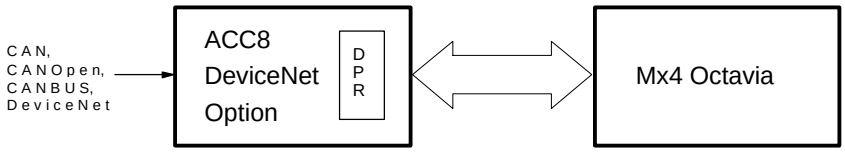


Fig. 1-1: ACC8 DeviceNet & CAN Option – Mx4 Octavia
Communication Block Diagram

Both the PC/AT Mx4 Octavia and VME Mx4 Octavia (in their bus versions) communicate with the host computer through Dual Port RAM (DPR) which is contained on the Mx4 Octavia board. If the user desires to communicate with the Mx4 Octavia with CAN or DeviceNet serial link, the ACC8 DeviceNet & CAN Option must be used.

CAN Protocols

CAN (Controller Area Networking) is the definition of a high performance communication protocol for serial data communications. It features a 15 bit CRC word, high EMI tolerance, non-destructive message arbitration, low latency, a high degree of determinism, minimal software requirements, and low

cost. Several industrial communications protocols are based on CAN including CanOPEN, CANBUS, and DeviceNet.

The ACC8 DeviceNet & CAN Option supports the CAN 2.0A standard which uses an 11-bit identifier field and a variable data field that can be up to 8 bytes long. This protocol allows 2032 unique identifiers to be used on the network. Baud rates up to 1 Mbit/sec can be used with the ACC8 DeviceNet & CAN Option board.

It is important to note that the ACC8 DeviceNet & CAN Option board optically isolates its CAN driver, so it is necessary to supply power to the CAN connector to support CAN communications. See Appendix C: Specifications for the power requirements.

DeviceNet

DeviceNet is a CAN based factory automation protocol developed by Allen-Bradley. It also uses the CAN 2.0A definition. The DeviceNet specification is maintained by the ODVA (Open DeviceNet Vendors Association). The ACC8 & CAN Option board complies with Release 2.0 of this specification. A Group 2 Only Predefined Master-Slave connection is implemented in software on the ACC8 DeviceNet & CAN Option board to provide DeviceNet access to the features of the Mx4 Octavia board. The following sections give more details about DeviceNet and usage of the Mx4 Octavia via DeviceNet.

2 What is DeviceNet

This section is intended to introduce DeviceNet to engineers and technicians who may not have used it before. It is not intended to provide specific help but describe the philosophy behind DeviceNet. It is hoped that this will provide insight into the operation and debugging of DeviceNet installations using the Mx4 Octavia board. DeviceNet is a simple, quick, and efficient means of connecting factory automation devices in an effective network. Originally developed by Allen-Bradley, the ease of use and simplicity of DeviceNet has inspired several manufacturers to provide and support DeviceNet capable products. In order to understand the basics of DeviceNet it is important to have a basic knowledge of networks in general. Networks can be thought of being composed of several layers. Each layer performs a different function. Every message sent on a network must traverse each of these seven layers at both the sending and receiving nodes. Each of these seven layers performs a different function. Some of these functions are not required by DeviceNet and are not implemented. The layers that DeviceNet does implement are the Physical Layer (hardware), the Protocol Layer (interpretation of individual bits in a message), and the DeviceNet Application Layer (interface with the product specific code). These layers are now described.

Physical Layer

The DeviceNet physical layer consists of all the hardware required to send a message from one node to another. This includes the cable, connectors, connector pinouts, wiring, power supplies, termination resistors, chips used to convert the signals on the cable into digital format (transceivers), and the chips used to interpret these digital signals (controllers). The ODVA (Open DeviceNet Vendors Association) DeviceNet Specification has sections devoted to each of these hardware elements. Fig. 2-1 shows how these pieces of hardware are related.

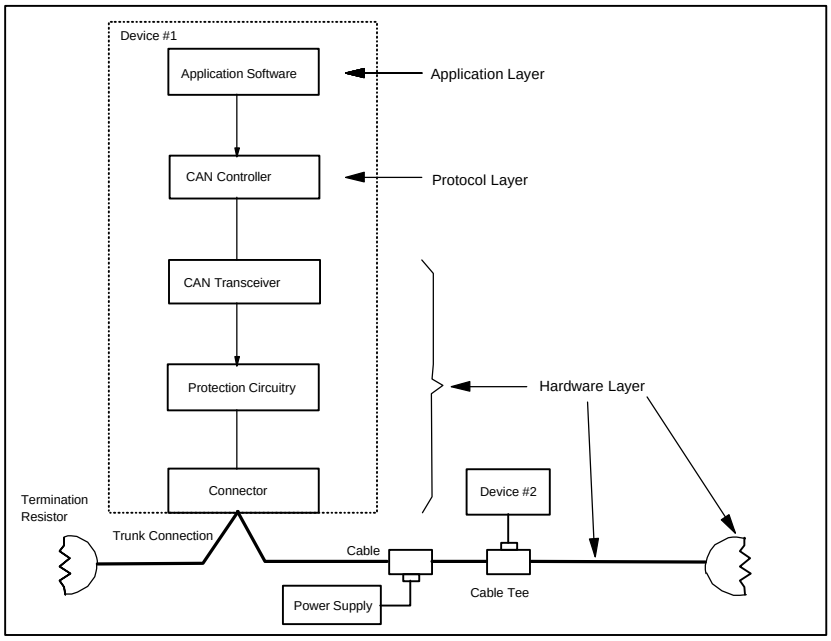


Fig. 2-1: DeviceNet Layers

The DeviceNet physical layer has the following features:

- Nodes can be removed without breaking the network
- Nodes can be attached at drops off of a network backbone
- One cable can carries both signal and power for the network
- The cable is shielded and communications lines are twisted to reduce EMI problems
- The network can be up to 500 m long (using 125 KBaud data rates)
- Data rates up to 500 KBaud can be used (network length limited to 100 m)
- Signal are carried differentially which reduces problems associated with high common-mode voltages which are typically present in long networks

Protocol Layer

DeviceNet uses the Controller Area Network or CAN protocol developed in Europe by Robert Bosch for automotive controls applications. CAN has several features that make it attractive for industrial applications:

- CAN employs a 15 bit Cyclical Redundancy Check (CRC) to detect errors in a CAN message. This yields an error rate (possibility of a "damaged" message getting through) of less than $3*10^{-5}$ or 0.003%.
- CAN was designed to work in the high EMI environment of a car engine compartment.
- CAN uses nondestructive message arbitration, so messages are not lost when two or more nodes try to communicate at the same time.
- Response times are quick (low latency) because the protocol and message arbitration is performed in hardware.
- Only a minimal amount of software is required to process a CAN message. This leaves more processor resources (time and speed) available for application code.
- CAN must be low cost in order to be attractive to automotive market.

Also, low cost CAN controllers and transceivers are readily available since CAN is widely used in automobiles.

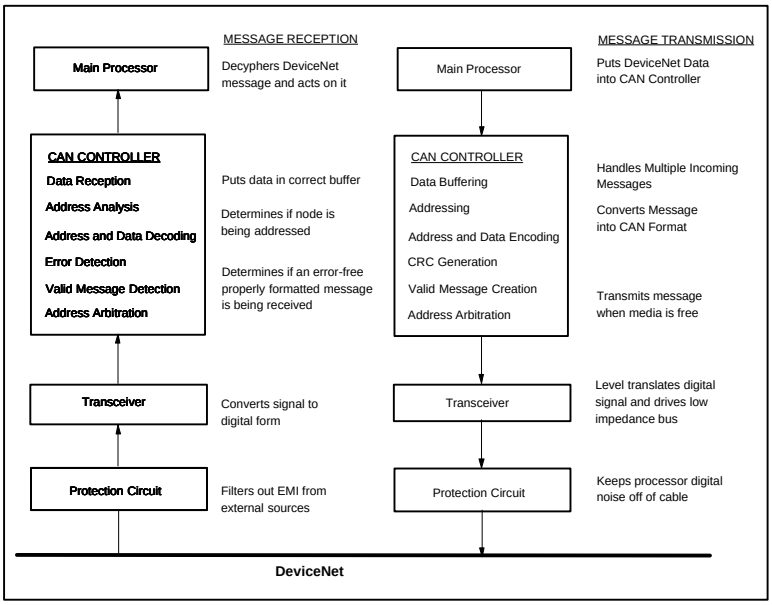


Fig. 2-2: CAN Protocol Layer

As shown in Fig. 2-2, the great advantage of CAN is that the entire protocol is performed in hardware. The application code is responsible for putting data and address information into the CAN controller and telling it to transmit. The CAN controller formats the message and drives the CAN transceiver. The receiving CAN controller performs the message arbitration, error detection, and message decoding. When a valid CAN message addressed to the node is received, the CAN controller signals the node's processor which the reads the address and data from the controller. The DeviceNet Application Layer is responsible for using the address and data in a meaningful way.

DeviceNet Application Layer

DeviceNet messages may be composed of one or more CAN messages. The DeviceNet Application layer interprets CAN messages and determines the instructions contained therein. Nodes on DeviceNet communicate with each other by establishing connections with each other. Connections are made by identifying each node using the connection and the type of connection that is being made. Nodes using a connection are identified with a decimal number between 0 and 63 called a Media Access Control Identifier or MacId. There are two types of DeviceNet connections: explicit and IO. Explicit connections are used for network maintenance (such as setting MacId and baud rate), configuring nodes, and transferring large blocks of data. IO connections have no defined use in the DeviceNet specification. Typically, they are used for transferring small amounts of data (less than 8 bytes) and instructions. DeviceNet messages are formatted in such a way that IO messages are interpreted by the CAN protocol layer to have a higher priority than explicit messages. This means that explicit messages can be delayed by the transmission of IO messages and, therefore, cannot be expected to arrive at predetermined times.

Since explicit messages can be used to “Get” and “Set” several parameters within a DeviceNet node, they must identify the data upon which they are operating. This identification is called a “path” and includes the “Class ID”, “Instance Number”, and “Attribute ID” for data. This path is shown in Fig. 2-3.

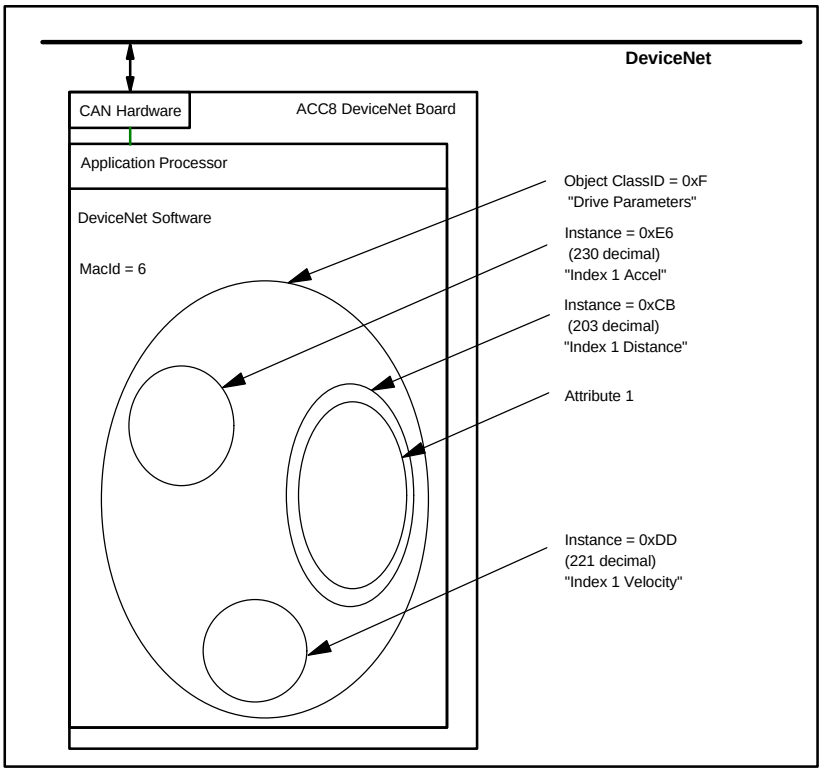


Fig. 2-3: Attribute Path for Axis on a Motion Controller

If the path is incorrectly specified, then the node will return an (explicit) error message.

There are four types of IO connections:

- Polled IO Messaging is used by a DeviceNet master to query specific devices.
- A Strobed IO message is a broadcast to the entire network which requests specific devices (identified in the broadcast message) to respond with predefined data.
- Cyclic messaging is used to repetitively transfer data with devices on the network at a predetermined rate.
- Change of State messaging is used to transfer data whenever a change occurs that warrants transmission.

IO messaging and Strobed IO are the easiest to implement; Change of State is the most efficient in terms of bandwidth and network loading.

The DeviceNet Network

DeviceNet is used to connect disparate sensors, actuators, and controllers to form a single coherent unit that performs a useful task. One node on the system must be a master and the other nodes slaves to that master. A typical system would consist of a PLC or personal computer connected to a scanner that controls/monitors a drive and various sensors.

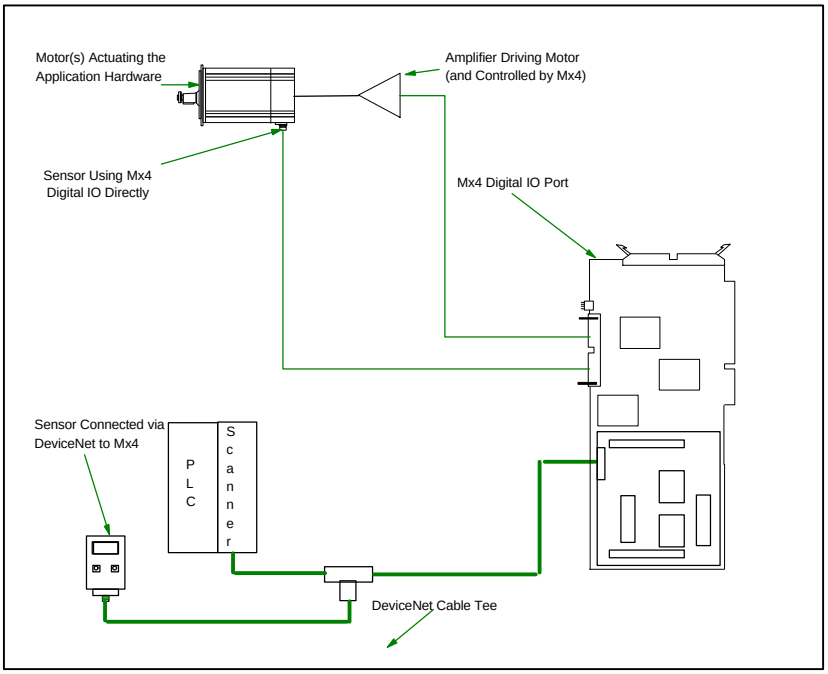


Fig. 2-4: Mx4 Octavia in a Typical DeviceNet Installation

Fig. 2-4 shows how an Mx4 Octavia board can be connected to DeviceNet using the ACC8 CAN/DeviceNet Option Card. The drive is directly controlled by an amplifier connected to the Mx4 Octavia board; its encoder output is also directly connected to the Mx4 Octavia for precise high-speed control. DeviceNet can also be used to connect low speed sensors and actuators to the Mx4 Octavia. The Mx4 Octavia is controlled by a PLC/Scanner via DeviceNet. The Scanner can be a board in the PC or a module attached to the PLC. In either case, it acts as the DeviceNet interface and system controller. It is responsible for downloading Real Time Commands and DSPL programs to the Mx4 Octavia board, transferring sensor data to the Mx4 Octavia, and monitoring network health. Data tables inside the scanner list the status of

discrete DeviceNet inputs and outputs and large data blocks for RTC's and DSPL programs. Fig. 2-5 illustrates the how the PLC interacts via the scanner with the Mx4 Octavia.

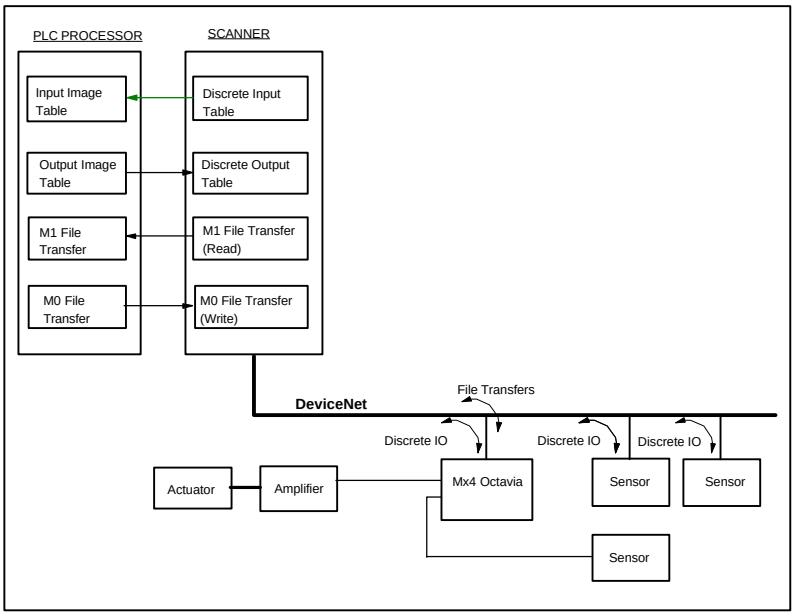


Fig. 2-5: PLC Connected to Mx4 Octavia

This page intentionally blank.

3 Installing The Hardware

The ACC8 DeviceNet & CAN Option board plugs into the Mx4 Octavia controller and is secured via two headers. Fig. 3-1 shows the board layout and dimensions.

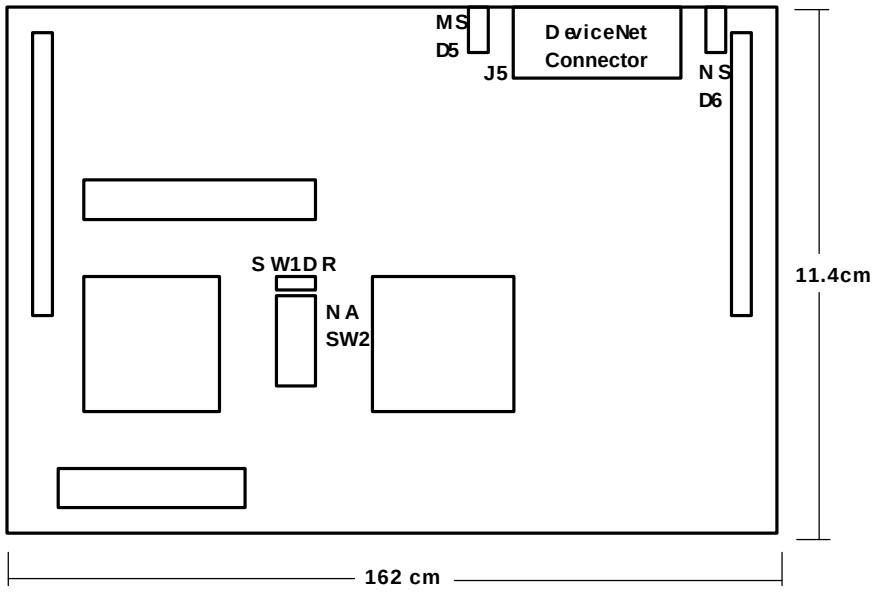


Fig. 3-1: ACC8 DeviceNet & CAN Option Dimensions, Connectors and Indicators.

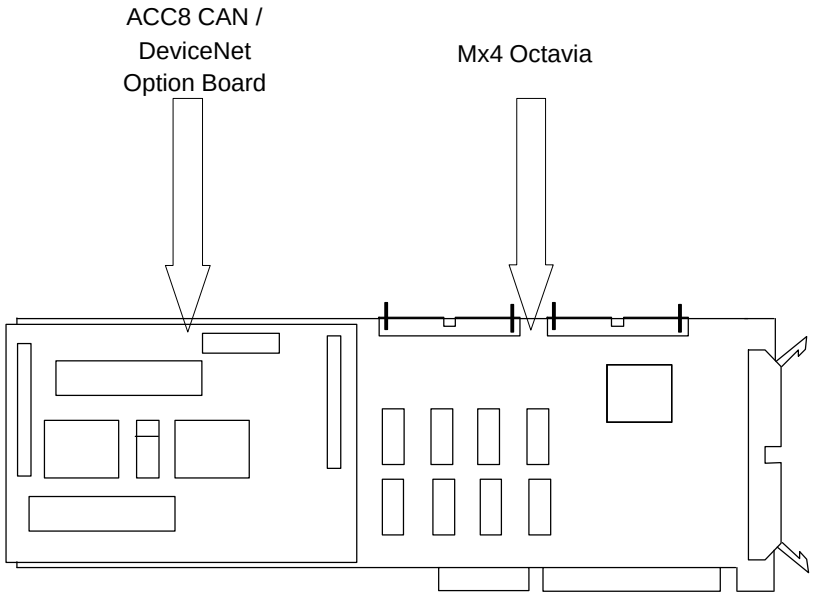


Fig. 3-2: ACC8 DeviceNet & CAN Option Mounted on PC/AT Mx4 Octavia

ACC8 DeviceNet & CAN Option Cabling

The ACC8 board uses a five position COMBICON MSTBA 2,5/5-G-5,08-AU header for the DeviceNet connector. This header mates with a pluggable open style, screw-connector such as the COMBICON MSTB 2,5/5-ST-5,08-AU or COMBICON MSTBP 2,5/5-ST-5,08-AU. The pinouts of the header are:

Pinout

Pin	Cable Color	Designation
1	Black	V- (0 Volts)
2	Blue	CAN L
3	Bare	Shield
4	White	CAN H
5	Red	V+ (+24 Volts)

Table 3-1: ACC8 DeviceNet & CAN Option Cabling Pinout

The ACC8 card is shipped with a default MacId of 63 and Baud Rate of 125. When the board is first powered up, the Module LED will flash green indicating that the board needs to be commissioned.

This page intentionally blank

4 Connecting to DeviceNet

WARNING! This chapter describes how to commission your ACC8 DeviceNet & CAN Option. ONLY commission your ACC8 DeviceNet & CAN Option when it is disconnected from the DeviceNet network. Failure to do so will cause a BUSOFF condition on a live network and result in network downtime.

Commissioning the ACC8 DeviceNet & CAN Option involves three steps:

- Set the baud rate switches
- Set the Mac Id switches
- Add the ACC8 DeviceNet Interface Option Card's Electronic Data Sheet (EDS) to the system library

The ACC8 DeviceNet Interface Option Card comes with factory default settings of 125 KBaud and MacId of 63. If the Module Status LED is flashing green on power up, then the device needs to be commissioned. Switches SW1 (**DR**) and SW2 (**NA**) used to set the Bau**D R**ate and **N**etwork **A**ddress are located on the board as shown in Fig. 4-1 below.

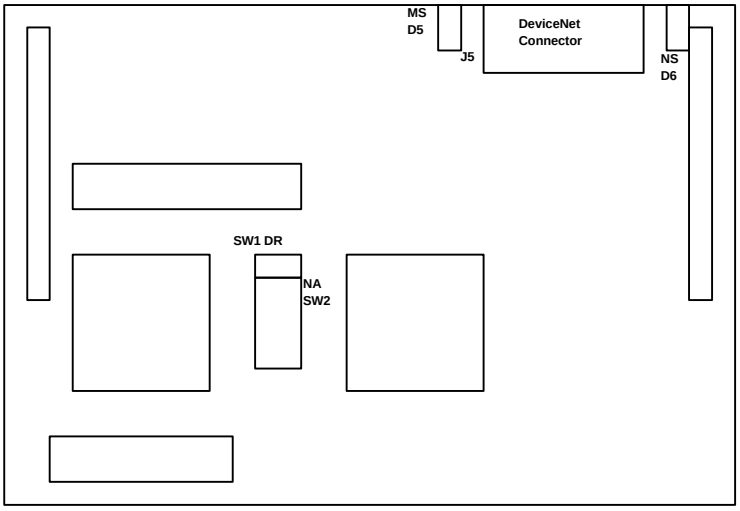


Fig. 4-1: Location of Baud rate (SW1) and Node Address (SW2) Switches

MacId (Media Access Control ID)

Refer to your system documentation or use DeviceNet management software to determine available MacIds for the ACC8 DeviceNet Interface Option Card. Once a MacId is chosen for the card, place the Mx4 Octavia board /ACC8 DeviceNet Interface Option Card on an electrostatic dissipative surface with the angle bracket on the left and the ISA/VME edge connector facing you. The DeviceNet connector on the ACC8 DeviceNet Interface Option Card should be facing away from you. The MacId is set with DIP switch SW2 located to the left center of the board. (Switches SW1 and SW2 are adjacent to each other; SW1 is the two position DIP switch closest to the DeviceNet connector; SW2 is the eight position DIP switch closest to the side of the board without connectors.) Once the Mx4 Octavia board is installed in your application, the baud rate switch is not user accessible. The switch settings can be overridden with a DeviceNet command as described in Chapter 5.

The following figure shows the SW2 setting for MacIds of 01, 38, and 63.

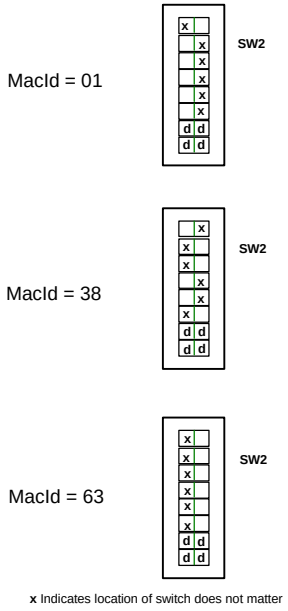


Fig. 4-2: Setting the MacId with Switch SW2

Baud Rate

The baud rate is set with DIP switch SW1. (Switches SW1 and SW2 are adjacent to each other; SW1 is the two position DIP switch closest to the DeviceNet connector; SW2 is the eight position DIP switch closest to the side of the board without connectors.) The following figure shows the SW1 setting for 125, 250, and 500 KBaud. Once the Mx4 Octavia board is installed in your application, the baud rate switch is not user accessible. The switch settings can be overridden with a DeviceNet command as described in Chapter 5.

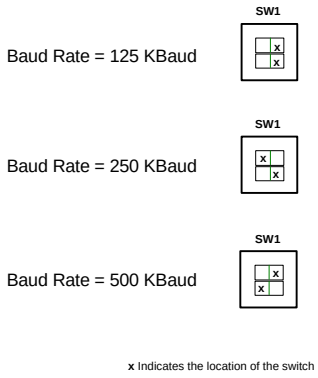


Fig. 4-3: Setting Baud Rate with Switch SW1

(A fourth configuration, with both switches moved to the left side, is invalid and will result in a baud rate of 500k Baud.)

5 Programming Reference

The ACC8 DeviceNet & CAN Option uses a DSPCG specific device profile with a Class ID of 56 hex (86 decimal). The ACC8 gives access to the digital inputs and outputs to and from the Mx4 Octavia, Real Time Commands, DSPL programs, and selected memory locations. The Mx4 Octavia can be thought of as an "object" or a set of "attributes" and "services" that can be "Set" using DeviceNet. Alternatively, DeviceNet can be used to "Get" the value of an attribute. The concepts of "objects", "attributes", and "services" are more fully described in Appendix C.

Data Transfer

The ACC8 DeviceNet & CAN Option uses a Group 2 Only Predefined Master-Slave connection as shown below in Fig. 5-1.

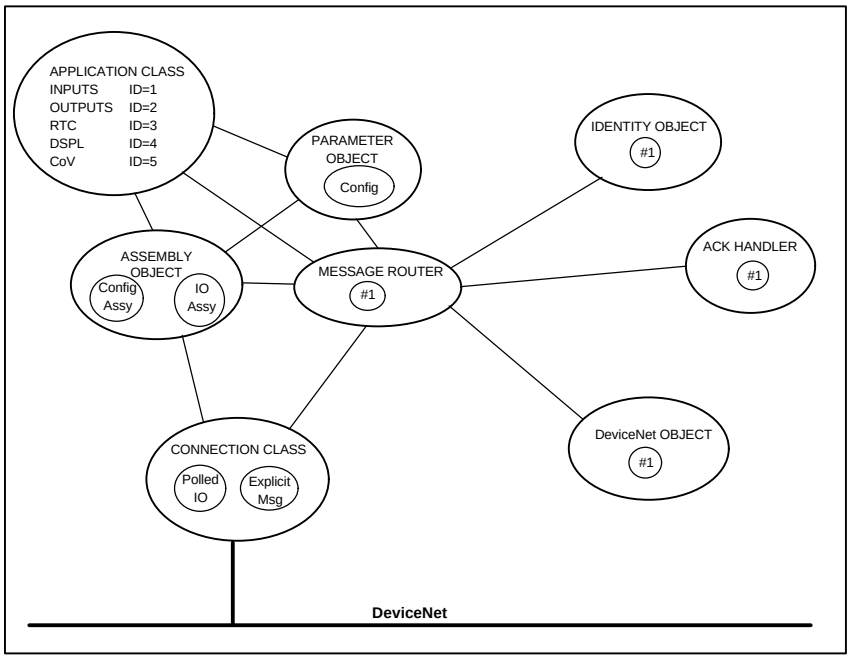


Fig. 5-1: ACC8 DeviceNet & CAN Option Object Map

The following definitions are used to define data sizes throughout this document.

Identifier	Mnemonic	Description
1	WORD	16-bit word
2	UINT	16-bit unsigned integer
3	INT	16-bit signed integer
4	BOOL	boolean
5	SINT	short integer (8 bit signed)
6	DINT	double integer (32 bit signed)
7	LINT	long integer (64 bit signed)
8	USINT	unsigned short integer
9	UDINT	unsigned double integer
10	REAL	single floating point format (IEEE 754)
20	STRING	8-bit per character string
23	SHORT_STRING	N-Byte long character string
24	BYTE	bit string, 8 bits

Table 5-1: DeviceNet Data Type Definitions

The Identity Object, Message Router Object, and DeviceNet Object are defined as follows:

Identity Object
Class ID 1, Instances (1)

The Identity Object identifies to node to DeviceNet and provides the general information listed in the table below

Service	ID	Attribute	Description	Type
GET_REQ Service Code = 14	1	Vendor ID	Vendor ID= DSP Control Group	UINT
	2	Device Type	DSP Motion Controller (Type 86)	UINT
	3	Product Code	Product Code=1	UINT
	4	Product Revision	Revision of the item Major Revision Minor Revision	STRUCT of: USINT USINT
	5	Status	Status of device (See Value Table next)	WORD
	6	Serial Number	Unique serial number	UDINT
	7	Product Name	"Mx4 Octavia"	SHORT_ STRING
Reset_REQ Service Code = 05		None	0=emulate reset as closely as possible a cycling of power. If data is omitted, this is default action 1=return as closely as possible to "out-of-box" configuration, then emulate power cycle reset.	USINT

Table 5-2: Identity Object Attributes

Status Attribute

The Status Attribute of the Identity Object is used to let the network master know the status of the node including if the device is faulted and to what degree the fault is recoverable.

Bit (s)	Called	Definition
0	Owned	TRUE indicates device has an owner
1		Reserved, set to 0
2	Configured	TRUE indicates settings change from "out of box" defaults
3		Reserved, set to 0
4, 5, 6, 7	Device Mode	0xFF=SELF TEST, other values TBD
8	Minor recoverable fault	TRUE if self diagnosis detects a minor fault
9	Minor unrecoverable fault	TRUE if self diagnosis detects an unrecoverable major fault
10	Major recoverable fault	TRUE if self diagnosis detects a major fault
11	Major unrecoverable fault	TRUE if self diagnosis detects an unrecoverable major fault
12, 13		Reserved, set to 0
14, 15		Reserved, set to 0

Table 5-3: Status Attribute Interpretation

Message Router Object
Class ID 2, Instances (1)

The Message Router Object allows access to any instance of any object class in the device. The ACC8 DeviceNet & CAN Option has a Class ID of 56 hex (86 decimal).

Services	ID	Attributes	Description	Type
None				

Table 5-4: Message Router Object

DeviceNet Object
Class 3, Instances (1)

The DeviceNet Object contains the MacId, baud rate, and other status information concerning the DeviceNet connection.

Class Attribute DeviceNet revision (ID=1) Revision=2

Service	ID	Attribute	Description	Type
GET_REQ Service Code = 14	**1**	MAC ID	0 – 63	USINT
	2	Baud Rate	0 = 125K, 1 = 250K, 2 = 500K	USINT
SET_REQ Service Code = 16	**1**	MAC ID	0 - 63	USINT
	2	Baud Rate	0=125k, 1=250k, 2=500k	
	3	BUS-OFF Interrupt	0=CAN controller held in reset	BOOL
	4	BUS-OFF Counter	Number of times device has gone BUS-OFF since power-up/reset.	USINT
ALLOC_GRP2_ IDSET Service Code = 75	**5**	Connection Choice	bit mapped byte 1 = Explicit message 2 = Polled 3 = Both	USINT
RELEASE_GRP2_ IDSET Service Code = 76	**5**	Connection Choice	bit mapped byte 1 = Explicit message 2 = Polled 3 = Both	USINT
Get_Req	**6**	Has MacID switch been changed?	0= No 1= Yes	BOOL

Table 5-5: DeviceNet Object Attributes

Service	ID	Attribute	Description	Type
	7	Has Baud Rate switch been changed?	0= No 1= Yes	BOOL
	8	MacId switch value	Actual value of MacId switch	USINT
	9	Baud Rate switch value	Actual value of Baud Rate switch	USINT

Table 5-5: DeviceNet Object Attributes (cont.)

Notes:

1) No data is required when setting the BUS-OFF Counter; accessing this attribute with a Set Request will automatically reset the counter to zero (0). The counter will count up to 255 BUS-OFF interrupts and will **not** rollover when it reaches 255.
2) Use the DeviceNet **Set Req**uest to override the switch setting for MacId and Baud Rate. A **Get Req**uest can be used for network debugging to determine if either of the MacId or Baud Rate switches have been overridden.

Assembly Object
Class 4, Instances (2)

The Assembly Object in a Group 2 Only Predefined Master-Slave connection only supports the data attribute of the configuration assembly and the IO assembly. The ACC8 DeviceNet & CAN Option supports Polled IO. The IO assembly is used to transfer the Mx4 Octavia's 32 bits of discrete IO.

The Assembly object is used to group together multiple attributes from one or more object(s). Once grouped or "assembled" together, the entire block of attributes can be communicated via a single connection instance.

The configuration Instance of the Assembly Object is present in the ACC8 DeviceNet & CAN Option software, but is not used at this time.

Connection Class
Class 5, Instances (2)

There are two Connection Objects as shown in Fig. 5-1. One maintains the Explicit Message connection, and the other maintains the Polled IO connection.

Service	ID	Attribute	Description	Type
RESET_REQ	5	None	Transition back to just powered up state	
GET_REQ	1	State	Bit mapped byte	USINT
	2	Type	0x00 (Explicit Connection)	BYTE
	3	Transport Trigger	0x83 (XPORT_SERVER3D)	
	4	Produced ID	Unsigned ID	UINT
	5	Consumed ID	Unsigned ID	UINT
	6	Initial Settings	0x21	BYTE
	7	Produced Size	261 (Maximum allowed)	UINT
	8	Consumed Size	261 (Maximum allowed)	UINT
	9	Packet Rate	Expected Packet Rate	UINT
	12	Watchdog Action	0x01 Auto delete	USIT
	13	Produce Path Length	0x00 (Not applicable to explicit connections)	UINT
	14	Produce Path	NULL (Not applicable to explicit connections)	ARRAY OF USINT
	15	Consume Path Length	0x00 (Not applicable to explicit connections)	UINT
	16	Consume Path	NULL (Not applicable to explicit connections)	ARRAY OF USINT
SET_REQ	9	Packet Rate	Expected Packet Rate	UINT
	12	Watchdog Action	0x01 Auto delete 0x03 Deferred delete	USINT

Table 5-6: Connection Object, Instance 1: Explicit Message Connection

Service	ID	Attribute	Description	Type
RESET_REQ	5	None	Transition back to just powered up state	
GET_REQ	1	State	Bit mapped byte	USINT
	2	Type	0x01 (IO Connection)	BYTE
	3	Transport Trigger	0x82 (XPORT_SERVER2D)	
	4	Produced ID	Unsigned ID	UINT
	5	Consumed ID	Unsigned ID	UINT
	6	Initial Settings	POLLED = 0x01	BYTE
	7	Produced Size	5 (Size of producing connection)	UINT
	8	Consumed Size	5 (Size of consuming connection)	UINT
	9	Packet Rate	Expected Packet Rate	UINT
	12	Watchdog Action	0x00 (Transition to Timed Out)	USINT
	13	Produce Path Length	0x06	UINT
	14	Produce Path	0x20, CLASS # 0x24, INSTANCE # 0x30, ATTRIBUTE #	ARRAY OF USINT
	15	Consume Path Length	0x06	UINT
	16	Consume Path	0x20, CLASS # 0x24, INSTANCE # 0x30, ATTRIBUTE #	ARRAY OF USINT
SET_REQ	9	Packet Rate	Expected Packet Rate	UINT
	12	Watchdog Action	0x00 (Transition to Timed Out) 0x02 Auto reset	USINT

Table 5-7: Connection Object, Instance 2: Polled IO Connection

Parameter Object
Class 15, Instances (1)

The Parameter Object is present in the ACC8 DeviceNet & CAN Option software, but is not used at this time.

Mx4 Octavia Object
Class 86, Instances (1)

Service	ID	Attribute	Description	Type
GET_REQ	1	N_Bytes	N bytes of Data: Number of bytes of Data Address LSB Address MSB	Struct of USINT USINT USINT
SET_REQ	1	N_Bytes	N Bytes of Data: Number of bytes of Data Address LSB Address MSB Data byte 1 . . . Data byte N	Struct of USINT USINT USINT Array of USINT(N)
	2	RTC	Real Time Command: Number of bytes in argument list RTC Code Argument 1 . . . Argument N	STRUCT of USINT USINT Array of USINT(N)

Table 5-8: Mx4 Octavia Attribute

The Mx4 Octavia Object allows the user to download data to the Mx4 Octavia, ran Real Time Commands, and as load results via DeviceNet as shown below.

Polled IO

The Polled IO connection to the ACC8 DeviceNet & CAN Option is used to set the digital outputs on the Mx4 Octavia board and returns the current state of its digital inputs.

Notes on Polled IO Messages:

1) The status of the Mx4 Octavia board and its digital inputs is returned in response to every polled IO message addressed to the ACC8 DeviceNet & CAN Option.

2) The format of the Polled message that sets the Mx4 Octavia outputs is shown in the following table.

Byte 0			Byte 1			Byte 2			Byte 3		
Mx4 Output Bits			Mx4 Output Bits			Mx4 Output Bits			Mx4 Output Bits		
7	…	0	15	…	8	23	…	16	31	…	24

Table 5-9: Format of Polled IO Messages Sent to ACC8

3) The format of the Polled message that the ACC8 DeviceNet & CAN Option sends in response to an incoming Polled IO message is:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Status Hi Byte	Status Lo Byte	Mx4 Input Bits 7-0	Mx4 Input Bits 15-8	Mx4 Input Bits 23-16	Mx4 Input Bits 31-24

Table 5-10: Format of Polled IO Messages Received from ACC8

4) Format of the Status Word is described in the Status Attribute section

This page intentionally blank

6 Operating the Mx4 Octavia Via DeviceNet

The ACC8 DeviceNet & CAN Option gives the Mx4 Octavia board the additional flexibility of being programmed and run over DeviceNet. With the ACC8 DeviceNet & CAN Option it is possible to transfer data and parameters such as individual gain settings for each axis, configuration data, and motion parameters as well as issue real time commands.

I/O

The ultimate goal of connecting an Mx4 Octavia board to DeviceNet is to be able to manipulate data in the Mx4 Octavia over DeviceNet so that devices connected to it can be controlled. Before this can be done, the Mx4 Octavia must be "mapped" into the DeviceNet scanner's list of devices. Mapping the Mx4 Octavia into the scan list does the following things:

- assigns PLC bits to specific functions on the board
- sets the IO messaging mode (Polled, Bit Strobe, Cyclic, or Change of State)
- avoids conflict with other nodes on the DeviceNet network

Once the Mx4 Octavia is entered in the scan list and the DeviceNet network is fully configured, the PLC can access individual Mx4 input and output bits from ladder logic. The following tables show a typical mapping.

Mx4 Octavia Output Bit	PLC Output Bit
Initiate Motion	O0
Axis 1 Enable	O1
Axis 1 Preset	O2
Axis 1 Fault Reset	O3
Axis 1 Mode	O4
Axis 1 Master/Slave	O5
Axis 1 Reserved	O6
Axis 2 Enable	O7
Axis 2 Preset	O8
Axis 2 Fault Reset	O9
Axis 2 Mode	O10
Axis 2 Master/Slave	O11
Axis 2 Reserved	O12
Axis 3 Enable	O13
Axis 3 Preset	O14
Axis 3 Fault Reset	O15
Axis 3 Mode	O16
Axis 3 Master/Slave	O17
Axis 3 Reserved	O18
Reserved	O19

Table 6-1: Example of Mapping Mx4 Octavia Inputs Bits to PLC Output Bits

Mx4 Octavia Input Bit	**PLC Input Bit**
Axis 1 Registration	I0
Axis 1 Limit Switch	I1
Axis 1 Force Limit	I2
Axis 1 Homed	I3
Axis 1 Reserved	I4
Axis 2 Registration	I5
Axis 2 Limit Switch	I6
Axis 2 Force Limit	I7
Axis 2 Homed	I8
Axis 2 Reserved	I9
Axis 3 Registration	I10
Axis 3 Limit Switch	I11
Axis 3 Force Limit	I12
Axis 3 Homed	I13
Axis 3 Reserved	I14
Move Complete	I15
User Start Button	I16
Reserved	I17

Table 6-2: Example of Mapping Mx4 Octavia Outputs Bits to PLC Input Bits

Fig. 6-1 shows how these IO bits can be programmed in ladder logic on a PLC.

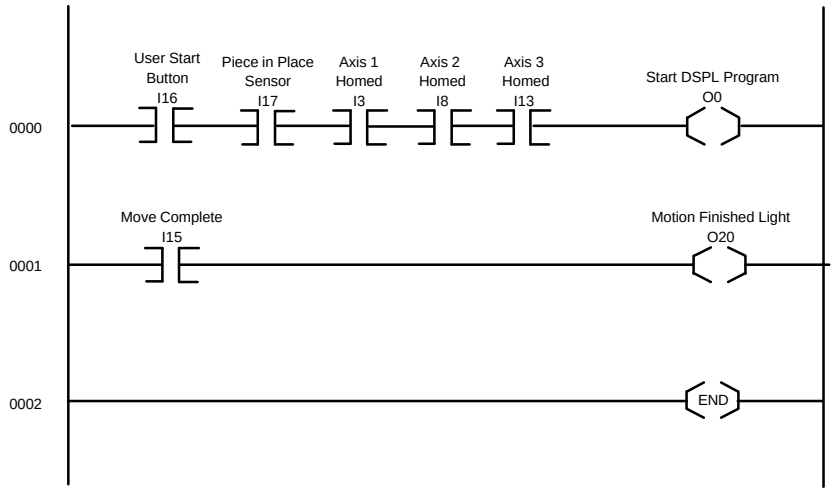


Fig.6-1: PLC Program to Control the Mx4 Octavia

The figure shows the logic for a 3 axis CNC machine which waits until the part being machined is in position, the operator presses the start button, and all three axes are in the home position. When the part is complete, the “Move Complete” input is asserted and the “Motion Finished” light is turned on. The motion of the three axes is controlled directly by an Mx4 Octavia DSPL program that was previously downloaded. The manufacturing cell can be controlled just as if all the components were directly connected to a PLC with the added benefits of reduced wiring and improved control with the Mx4 Octavia.

Explicit Messaging

The previous section alluded to the possibility of downloading real time commands (RTC) to the Mx4 Octavia board. This is done using DeviceNet Explicit Messages. These messages can also be used to set any user accessible variables such as gain settings and error limits. The PLC program must use file copy blocks to specify and set or get these attributes. Before an attribute can be set, a PLC Integer File must be created which holds the address and value of the attribute. Next, a ladder logic rung must be added be added to the PLC program to download (perform the file copy) the variable. Finally, the file must copied to the node.

The following example illustrates how a Real Time Command to change the corner frequency of the low-pass filter used on axis 5 to 1000 Hz can be run by a PLC/Scanner. It is assumed that the PLC/Scanner has a MacId of 1 and that the ACC8 CAN/DeviceNet Interface Option has a MacId of 32.

a) Identify the data required for the explicit message used to download the RTC. From the **Programming Reference** section of this manual, the following data is obtained:

```
Service Code  0x10   ; Set Attribute Request
Class Id      0x56   ; DSP Motion Controller
Instance      0x01   ;Instance of
                      motion controller present
Attribute     0x02   ; Real Time Command
```

b) The value of the attribute is the actual RTC code and arguments used. The format of the RTC value is as follows:

Byte	Content
0	N = Number of arguments in the RTC
1	RTC Code
2	1st argument
N + 2	Nth argument

Table 6-3: Format of Real Time Command in DeviceNet Message

These values are obtained from the Mx4 Octavia's *User's Guide*.

```
Value     0x03      ; N = 3 byte argument
          0x8E      ; "LOW_PASS" RTC code
          0x05      ; Specifies axis 5
          0xE8      ; 1000 decimal = 0x03E8
          0x03
```

c) The explicit message can now be assembled. DeviceNet has a specific format for explicit messages as shown below.

Byte	Content	Description
0	0x01	Master MacId
1	0x10	DeviceNet Set Request
2	0x56	DSP Motion Controller Class ID
3	0x01	Instance #1
4	0x02	RTC Attribute ID
5	0x03	Number of RTC Arguments = 3
6	0x8E	LOW_PASS RTC Code
7	0x05	Axis # 5
8	0xE8	1000 Hz = 0x03E8; Low byte
9	0x03	1000 Hz = 0x03E8; High byte

Table 6-4: Format of Explicit Message used to send a Real Time Command

Note: This ten byte message will be fragmented by DeviceNet to fit into its 8 byte data field. The fragmenting protocol will insert a Fragment Control byte after Byte 0 (Master MacId) and set bit 7 of Byte 0. The second DeviceNet message will contain the Master MacId, another Fragment control byte, and the remaining data.

The data in Table 6-4 must be organized into a PLC integer data file. This is where the PLC gets the data to download to the ACC8 CAN/DeviceNet Interface Option. The data in the table looks like this in the PLC file:

01 10 56 01 02 03 8E 05 E8 03

d) A ladder logic rung must now be added to the PLC program so that the PLC can execute the file copy.

Fig. 6-2: PLC Ladder Logic Used to Download on RTC

The ladder logic program shown in Fig. 6-2 causes the data stored in file <filename> to be downloaded using a M0 file transfer to address 181 when the input 180 is closed. Eight bytes of data is the specified length of the file. This causes the input file to be copied to the PLC M0 file where it is sent to the scanner which then uses DeviceNet to download the file to the ACC8 DeviceNet & CAN Option. When the card receives the RTC set request, it sends an explicit message response to the scanner indicating the success of failure of the transfer. This response uses a M1 file transfer from the ACC8 DeviceNet & CAN Option to the scanner.

The Mx4 Octavia will execute the command as soon as it receives it.

This page intentionally blank

Appendix A:

Solving Network Problems

The ACC8 DeviceNet & CAN Option supports both the Module Status and Network Status LED's which can be used in debugging DeviceNet problems. The user should first consult the documentation pertaining to individual network nodes in an attempt to isolate the problem. A walk-through of the installation to map the physical location of each node in the network (and also note the status of individual Module and Status LED's) is usually helpful. Comparing the physical map with the virtual map helps to localize the problems to specific parts of the network.

Bi-Color LED's

The ACC8 DeviceNet & CAN Option Module Status LED is located on the board to the right of the DeviceNet connector (looking at the board edge with the connector) and is designated "MS". The Network Status LED is located to the left of the DeviceNet connector and is designated "NS". The following tables describe the functionality of these indicators.

Power Up Sequence
Module and Network Status LED's both off
Module LED lights up green for about 250 mS Module LED lights up red for about 250 mS If the ACC8 DeviceNet & CAN Option has not been commissioned: Module LED will flash green If the ACC8 DeviceNet & CAN Option has been commissioned and has passed the self diagnostics: Module LED will light up steady green
Network LED lights up green for about 250 mS Network LED lights up red for about 250 mS Network LED lights up steady green if network is functioning and the node is owned

Table A-1: Power up Sequence as Reflected by Module and Network Status LEDs

Module Status	**LED**	**Diagnosis**
No power	Off	No power applied to the node.
Operational	Steady Green	Node is fully functional.
Standby	Flashing Green	Node needs to be commissioned
Recoverable fault detected	Flashing Red	Self-diagnostics discovered a minor fault.
Unrecoverable fault detected	Steady Red	Self-diagnostics discovered a major fault. Board may need to be replaced.
Self Test	Alternately flashing Red-Green	Node is running self-diagnostics.

Table A-2: Module Status LED States

Network Status	**LED**	**Diagnosis**
Not powered or off-line	Off	• Check Module Status LED for state of node power. • The Mx4 Octavia is not powered. • The node has not completed the duplicate MacId check.
On-line, but not connected	Flashing Green	Node is not connected
On-line and connected	Steady Green	Node is connected and allocated to a master
Connection timed out	Flashing Red	One or more connections have exceeded the time-out limit.
Connection failure	Steady Red	Node has detected a duplicate MacId or has taken itself BUSOFF
Self Test	Alternately flashing Red-Green	Node is running Self-diagnostics.

Table A-3: Network Status LED States

Note: The ACC8 DeviceNet & CAN Option uses an optically isolated DeviceNet interface circuit. The ACC8 microprocessor (which drives the Module and Network Status LED's) gets its power from the Mx4 Octavia board. The status of the network power is optically coupled to the ACC8 microprocessor. Therefore, it is possible to have a lit Module Status LED and an unlit Network Status LED. It is impossible to have an unlit Module Status LED and a lit Network Status LED.

Problems with the Node

The most common source of node problems is improper setting of MacId or baud rate. Before tearing down the system, make sure that all nodes are using the same baud rate and have appropriate MacIds.

If a node goes BUSOFF (Network Status LED is Steady Red) it means that it has detected an excessive number of errors in transmission or reception. It is quite possible for one node to have problems that cause other nodes to go BUSOFF. If resetting a node does not stop it going BUSOFF, then another node is probably causing the error. Other possible causes of the error are:

- improper MacId setting
- improper baud rate setting
- incorrect network grounding
- intermittent power connections
- intermittent data connections
- electromagnetic interference

When a scanner goes BUSOFF, nodes connected to it will not be able to reallocate (Network Status LED flashing Green) although they are functioning correctly.

As mentioned earlier in this appendix, a walk through of the system to map physical and virtual locations of all nodes connected to the network is a valuable exercise. This exercise can identify MacId conflicts, provide visual feedback on all Module and Network Status LED's, and identify sections of the network where excessive DeviceNet power line draw may be causing problems.

In diagnosing problems, remember that some nodes (including the ACC8 CAN/DeviceNet Interface Option board) can have their MacId and Baud Rate switch setting overridden with DeviceNet Set Requests. If some nodes display a steady red Network Status LED and others do not, you may want to query the functioning nodes to determine if their MacId and Baud Rate switches have been overridden. If they have, get their actual MacId and Baud Rate Settings. The point of this exercise is to ensure that: 1) there are no duplicate MacIds and 2) the network baud rate is what is thought to be correct.

Node Failures

A steady Red Module Status LED indicates an unrecoverable fault with the node; the node hardware may need to be serviced or replaced.

A steady Red Network Status LED indicates that the node has detected a duplicate MacId or has gone BUSOFF. Use the system Master to perform a

duplicate MacId test. If the Network Status LED is still steady Red after the system passes the Master duplicate MacId test, then the node has a BUSOFF error. Software solutions to the this problem are to:

- check baud rate settings—make sure all nodes are at the same speed
- change the node MacId

Hardware solutions to this problem are to:

- check and/or replace the connection between the node and the network
- check the network layout—excessive current draw on the network can cause voltage drops that interfere with data transmission
- check power and data lines for excessive noise caused by external sources such as motor commutation, HVAC lines, fluorescent lighting power, or arc welding
- check the termination resistors at both ends of the DeviceNet cable—the cable should be terminated with one 120 ohm resistor at each end, i.e.: two termination resistors should be present.

DeviceNet Master Problems

Check for appropriate MacIds, consistent baud rate settings, and proper scan list. Make sure that the scanner/PLC series and revision is up-to-date and correct for the network hardware. Cycling the DeviceNet 24V power supply and resetting the master should take it out of the BUSOFF state. If the master goes BUSOFF again, the problem is some combination of:

- damaged or defective node device
- incorrect baud rate setting
- bad network topology
- faulty wiring
- faulty connectors and/or junction boxes
- faulty scanner
- faulty power supply
- bad grounding
- excessive DeviceNet current draw
- electrical noise

Cabling Problems

Improper cable installation can cause several problems for a DeviceNet network. The KISS principle—Keep It Short—should be the guiding rule, especially at higher baud rates. Avoid at all costs:

- excessive twisting or tension on the cable—in particular, check connectors for any sign of strain
- proximity to extreme temperature
- proximity to motors, relays, contactors, solenoids, power wiring, RF communications gear, arc welders, or any other source of electromagnetic interference
- improper installation of, or missing, termination resistors
- improper lead dress (excessive twisting, spaghetti, or improper stripping) in junction boxes
- dirty, worn, or wet connectors

DeviceNet Power Supply Problems

Make sure that the **minimum** current rating of the DeviceNet power supply meets or exceeds the sum of the **maximum** current requirements of all nodes connected to the network. The ACC8 DeviceNet & CAN Option draws a maximum of 75 mA from the DeviceNet power supply at 11 V d.c. Trunk cable has a maximum current rating of 8 amps and Drop cable has a maximum current rating of 3 amps. Excessive current draw can cause a voltage drop in the cable that brings the voltage available to a node below its requirements. Also, the cable voltage drop can interfere with reception of the data. Things to check for are:

- trunk and drop cable current draw
- size, length, and voltage drop on cable bring power to the trunk (if the cable is too long to easily measure the voltage drop, the voltage drop is probably too high)
- DeviceNet power supply voltages at both ends and the middle of the network
- the presence of noise on the DeviceNet network power

Use the following tables to verify your network.

Parameter	**Network Measurement**
Number of Nodes	
Individual Drop Lengths	
Branched Drop Length	
Cumulative Drop Length	
Total Trunk Length	
Power Supply Cable Length and Gauge	

Table A-4: Cable Checks to Perform

Parameter	**Network Measurement**
Termination resistor locations and values	
Disconnect network (power **and** shield) from power supply and verify that the network impedance between V- and frame ground is in excess of 1 Megohms	
Short circuit between: CAN HI and CAN LO CAN HI and shield CAN LO and shield CAN HI and V+ CAN LO and V+ CAN HI and V- CAN LO and V- V+ and shield V- and shield V+ and V-	
Total current draw on Trunk	
Distance of maximum Trunk current draw from DeviceNet power supply	
Maximum current draw on Drops	
Amplitude of noise on DeviceNet power supply	

Table A-5: Network Power Checks to Perform

Guiding Principles in DeviceNet Layout

- Keep overall length of cable to a minimum
- Keep power supply close to the maximum current draw
- Keep current draw even over entire network
- Keep the power supply near the center of the network, or off-center in the direction of the maximum current draw
- Keep nodes away from those nodes that draw high current

Appendix B:

Compliance

DeviceNet Statement of Conformance

DeviceNet Statement of Conformance

	Fill in the blank or 'x' the appropriate box				
General Device Data	Conforms to DeviceNet Specification	Volume I – Release Volume II – Release	_2.0_____ _2.0_____		
	Vendor Name		_DSP Control Group, Inc._		
	Device Profile Name		_DSP Motion Controller__		
	Product Catalog Number		_ACC8DN_____		
	Product Revision		_1.0_____ _____		
DSPCG Physical Conformance Data	Network Power Consumption (Max)		_0.072__A @11V dc (worst case)		
	Connector Style	Open-Hardwired Open-Pluggable	**X**	Sealed-Mini Sealed-Micro	
	Isolated Physical Layer	Yes No	**X**		
	LEDs Supported	Module Network	**X** **X**	Combo Mod/Net I/O	
	MAC ID Setting	DIP Switch Other	**X**	Software Selectable	
	Default MAC ID		**63**		
	Communication Rate Setting	DIP Switch Other	**X**	Software Selectable	
	Communication Rates Supported	125k bit/s 250k bit/s	**X** **X**	500k bit/s	**X**

DeviceNet Statement of Conformance

DeviceNet Statement of Conformance

Fill in the blank or 'x' the appropriate box (form cont.)

DSPCG Communication Data	**X**	Predefined Master/Slave Connection Set	Group 2 Client Group 2 Server		Group Only 2 Client Group Only 2 Server	**X**
		Dynamic Connections Supported (UCMM)	Group 1 Group 2		Group 3	
	Fragmented Explicit Messaging Implemented		Yes No If yes, Transmission Time Out Typical Target Address Class Instance Attribute	**X** **2000 mS** **0x01** **1** **7**		

DeviceNet Statement of Conformance

DeviceNet Required Object Implement-ation

Identity Object 0x01

Object Class		**ID**	**Description**	**Get**	**Set**	**Value Limit**
Attributes	Open	1	Revision			
		2	Max Instance			
X None		6	Max ID of class attributes			
Supported		7	Max ID of instance attributes			
			DeviceNet Services	**Parameter Options**		
Services			Get_Attribute_All			
X None Supported			Reset			
			Get_Attribute_Single			
			Find_Next_Object_Instance			
Object Instance		**ID**	**Description**	**Get**	**Set**	**Value Limit**
Attributes	Open	1	Vendor	**X**		999
		2	Product Type	**X**		6
X None		3	Product Code	**X**		12
Supported		4	Revision	**X**		1.0
		5	Status (bits supported)	**X**		8
		6	Serial Number	**X**		
		7	Product Name	**X**		Mx4 Octavia
		8	State	**X**		
			DeviceNet Service	**Parameter Options**		
Services			Reset	Type: 0.1		
None Supported			Get_Attribute_Single			
Vendor Specific Additions		If yes, fill out the Vendor Specific Additions form (Form 8)		Yes No **X**		

X **Get** to indicate that attribute value is returned by the use of Get_Attribute_Single service
X **Set** to indicate that attribute value is returned by the use of Get_Attribute_Single service

DeviceNet Statement of Conformance

*Device**Net*** **Statement of Conformance**

DeviceNet Required Object Implement-ation	**Message Router Object 0x02**					
	Object Class	**ID**	**Description**	**Get**	**Set**	**Value Limit**
	Attributes Open	1	Revision			_____
		4	Optional attribute list			_____
	X None	5	Optional service list			_____
	Supported	6	Max ID of class attributes			_____
		7	Max ID of instance attributes			
			DeviceNet Services	**Parameter Options**		
	Services		Get_Attribute_All	_____		
	X None Supported		Get_Attribute_Single	_____		
	Object Instance	**ID**	**Description**	**Get**	**Set**	**Value Limit**
	Attributes Open	1	Object list			_____
		2	Maximum connections			_____
	X None		Supported			_____
	Supported	3	Number of active			_____
			Connections			
		4	Active connections list			
			DeviceNet Service	**Parameter Options**		
	Services		Get_Attribute_All	_____		
	X None Supported		Get_Attribute_Single	_____		
	Vendor Specific Additions	If yes, fill out the Vendor Specific Additions form (Form 8)		Yes		
				No	**X**	

X **Get** to indicate that attribute value is returned by the use of Get_Attribute_Single service
X **Set** to indicate that attribute value is returned by the use of Get_Attribute_Single service

DeviceNet Statement of Conformance

DeviceNet Required Object Implement-ation	**DeviceNet Object 0x03**						
	Object Class		**ID**	**Description**	**Get**	**Set**	**Value Limit**
	Attributes None Supported	Open	1	Revision			_____
	Services None Supported			**DeviceNet Services** Get_Attribute_Single	**Parameter Options** _____		
	Object Instance		**ID**	**Description**	**Get**	**Set**	**Value Limit**
	Attributes None Supported	Open	1	MAC ID	**X**	**X**	_____
			2	Baud rate	**X**	**X**	_____
			3	BOI	**X**		_____
			4	Bus-off counter	**X**		_____
			5	Allocation information	**X**		_____
			6	MAC ID switch changed			_____
			7	Baud rate switch changed			_____
			8	MAC ID switch value			_____
			9	Baud rate switch value			
				DeviceNet Service	**Parameter Options**		
	Services		**X**	Get_Attribute_Single	_____		
	None Supported		**X**	Get_Attribute_Single	_____		
			X	Allocate M/S connection set	_____		
			X	Release M/S connection set	_____		
	Vendor Specific Additions			If yes, fill out the Vendor Specific Additions form (Form 8)	Yes No	 **X**	

X **Get** to indicate that attribute value is returned by the use of Get_Attribute_Single service
X **Set** to indicate that attribute value is returned by the use of Get_Attribute_Single service

DeviceNet **Statement of Conformance**

DeviceNet Required Object Implement-ation

Assembly Object 0x04

Object Class		**ID**	**Description**	**Get**	**Set**	**Value Limit**
Attributes	Open	1	Revision			_____
		2	Max Instance			_____
None Supported						

	DeviceNet Services	**Parameter Options**
Services	Create	_____
X None Supported	Delete	_____
	Get_Attribute_Single	_____

Object Instance	**Instance Type**	**Instance ID(s)**
	Static Input	_____
	Static Output	_____
	Static I/O	_____
	Static Configuration	_____
	Dynamic	_____

		ID	**Description**	**Get**	**Set**	**Value Limit**
Attributes	Open	1	Number of Members in List			_____
None		2	Member List			_____
Supported		3	Data	**X**	**X**	_____

Compliance

		DeviceNet Service	Parameter Options
Services		Delete	_____
None Supported	**X**	Get_Attribute_Single	_____
	X	Set_Attribute_Single	_____
		Get_ Member	_____
		Insert_ Member	
		Remove_Member	
Vendor Specific Additions		If yes, fill out the Vendor Specific Additions form (Form 8)	Yes No **X**

X **Get** to indicate that attribute value is returned by the use of Get_Attribute_Single service
X **Set** to indicate that attribute value is returned by the use of Get_Attribute_Single service

DeviceNet Statement of Conformance

DeviceNet Required Object Implementation

Connection Object 0x05

Object Class	ID	Description	Get	Set	Value Limit
Attributes Open **X** None Supported	1	Revision			_____

	DeviceNet Services	Parameter Options
Services	Reset	_____
X None Supported	Create	_____
	Delete	_____
	Get_Attribute_Single	_____
	Find_Next_Object_Instance	_____

Total Active Connections Possible 2_____

Object Instance	Section	Information	Max Instance

Compliance

Instance Type	Explicit Message	**X**	**1**
	Polled I/O		
	Bit Strobed I/O		
	Dynamic I/O		
Production Trigger	Cyclic		
	Change of State		
	Application		
Transport Type	Triggered		
	Server		
Transport Class	Client		
	0		
	2		
	3		

*Device**Net*** **Statement of Conformance**

Connection Object 0x05 (cont)

		ID	Description	Get	Set	Value Limit
Attributes	Open	1	State	**X**		
None		2	Instance type	**X**		0
Supported		3	Transport class trigger	**X**		0x23
		4	Produced connection ID	**X**		
		5	Consumed connection ID	**X**		
		6	Initial comm. characteristics	**X**		
		7	Produced connection size	**X**		5
		8	Consumed connection size	**X**		255
		9	Expected packet rate	**X**		
		12	Watchdog time-out action	**X**	**X**	
		13	Produced connection path length	**X**		
		14	Produced connection path	**X**		
		15	Consumed connection path length	**X**		
		16	Consumed connection path	**X**		

	DeviceNet Service	**Parameter Options**
Services	Reset	
None Supported	Delete	
	Apply Attributes	
	Get_Attribute_Single	
	Set_Attribute_Single	

Vendor Specific Additions	If yes, fill out the Vendor Specific Additions form (Form 8)	Yes	
		No	**X**

X **Get** to indicate that attribute value is returned by the use of Get_Attribute_Single service
X **Set** to indicate that attribute value is returned by the use of Get_Attribute_Single service

DeviceNet Statement of Conformance

DeviceNet **Statement of Conformance**

DeviceNet Required Object Implementation

Connection Object 0x05

Object Class		**ID**	**Description**	**Get**	**Set**	**Value Limit**
Attributes	Open	1	Revision			_____
None Supported						

	DeviceNet Services	**Parameter Options**
Services	Reset	_____
None Supported	Create	_____
	Delete	_____
	Get_Attribute_Single	_____
	Find_Next_Object_Instance	_____

Total Active Connections Possible 2_____

Object Instance	**Section**	**Information**		**Max Instance**
	Instance Type	Explicit Message		_____
		Polled I/O	**X**	**1**_____
		Bit Strobed I/O		_____
		Dynamic I/O		_____
	Production Trigger	Cyclic		_____
		Change of State		_____
		Application		_____
	Transport Type	Triggered		_____
		Server		_____
	Transport Class	Client		_____
		0		_____
		2		_____
		3		_____

DeviceNet Statement of Conformance

Connection Object 0x05 (cont)

		ID	Description	Get	Set	Value Limit
Attributes	Open	1	State	**X**		
None		2	Instance type	**X**		1
Supported		3	Transport class trigger	**X**		
		4	Produced connection ID	**X**		
		5	Consumed connection ID	**X**		
		6	Initial comm. characteristics	**X**		
		7	Produced connection size	**X**		5
		8	Consumed connection size	**X**		5
		9	Expected packet rate	**X**		
		12	Watchdog time-out action	**X**	**X**	
		13	Produced connection path length	**X**		6
		14	Produced connection path	**X**		
		15	Consumed connection path length	**X**		6
		16	Consumed connection path	**X**		
			DeviceNet Service	**Parameter Options**		
Services		**X**	Reset			
None Supported			Delete			
			Apply Attributes			
		X	Get_Attribute_Single			
		X	Set_Attribute_Single			
Vendor Specific Additions		If yes, fill out the Vendor Specific Additions form (Form 8)		Yes		
				No	**X**	

X **Get** to indicate that attribute value is returned by the use of Get_Attribute_Single service
X **Set** to indicate that attribute value is returned by the use of Get_Attribute_Single service

DeviceNet Statement of Conformance

	*Device**Net***	**Statement of Conformance**			
DeviceNet Open Object Implementation	**OBJECT NAME**	**DSP Motion Controller**		**OBJECT ID**	**0x56**
	Object Class	**ID**	**Description**	**Get** **Set**	**Value Limit**
	Attributes Open	___	_____		_____
		___	_____		_____
		___	_____		_____
		___	_____		_____
		___	_____		_____
			DSPCG Services	**Parameter Options**	
	Services		_____	_____	
			_____	_____	
			_____	_____	
	Object Instance	**ID**	**Description**	**Get** **Set**	**Value Limit**

				Compliance		
Attributes	Open	1				
		2				
		3				
		4				
		5				
		6				
		7				
		8			**X**	
		9	PROGRAM SIZE		**X**	255
		10	DSPL PROGRAM	**X**	**X**	
		11	Download Status		**X**	USINT 0-0x0F
		12	RTC Size		**X**	255
		13	RTC	**X**	**X**	
		14	RTC Download Status	**X**	**X**	USINT 0-0x0F
		15	Change of Variable			
		16	Status Byte			USINT 0-0x0F
		17				
		18				
		19				
		20				

*Device**Net*** **Statement of Conformance**

OBJECT NAME	**DSP Motion Controller**	**OBJECT ID**	**__0x56___ (cont.)**
	DSPCG Services	**Parameter Options**	
Services	Get_Attribute_Single_____	_____	
	Set_Attribute_Single_____	_____	
	_____	_____	
		_____	
		_____	
Meaning of Zero Length I/O Data Production	__No new I/O data_____		
Vendor Specific Additions	If yes, fill out the Vendor Specific Additions form (Form 8)	Yes **X** No	

X **Get** to indicate that attribute value is returned by the use of Get_Attribute_Single service
X **Set** to indicate that attribute value is returned by the use of Get_Attribute_Single service

DeviceNet Statement of Conformance

DeviceNet Statement of Conformance

DeviceNet Open Object Implementation	**OBJECT NAME**		**Parmameter Object**		**OBJECT ID**	**0x56**
	Object Class		**ID**	**Description**	**Get** **Set**	**Value Limit**
	Attributes	Open	2	Max Instance	**X**	
			8	Class Descriptor	**X**	1
			9	Assembly Instance	**X**	0x01
				DSPCG Services	**Parameter Options**	
	Services					
	Object Instance		**ID**	**Description**	**Get** **Set**	**Value Limit**

Compliance

Attributes	Open	1	Value	**X**	**X**	
		2	Path Size	**X**		0
		3	Path	**X**		0
		4	Descriptor	**X**		0x10
		5	Type	**X**		2
		6	Size	**X**		2
		7				
		8				
		9				
		10				
		11				
		12				
		13				
		14				
		15				
		16				
		17				
		18				
		19				
		20				

Device*Net* **Statement of Conformance**

OBJECT NAME	**Parameter Object**	**OBJECT ID** **__0x56___ (cont.)**
	DSPCG Services	**Parameter Options**
Services	Get_Attribute_Single_____	_____
	Set_Attribute_Single_____	_____
	_____	_____
		_____
		_____
Meaning of Zero Length I/O Data Production	__No new I/O data_____	
Vendor Specific Additions	If yes, fill out the Vendor Specific Additions form (Form 8)	Yes No **X**

X **Get** to indicate that attribute value is returned by the use of Get_Attribute_Single service
X **Set** to indicate that attribute value is returned by the use of Get_Attribute_Single service

Appendix C:

Specifications

The ACC8 DeviceNet & CAN Option mounts directly on the Mx4 Octavia board with two 64-pin headers. It connects to DeviceNet with A 5-pin COMBICON MSTB 2,5/5-ST-5,08 connector.

The ACC8 DeviceNet & CAN Option is a self-contained microcontroller with flash memory, dual-port RAM, and a Watchdog timer. The Watchdog is kicked when the Mx4 Octavia board is first powered up. The Watchdog must be reset within a specified time delay, or else the software will shut down and display a steady RED on the bi-colored Module Status LED. At power up the ACC8 DeviceNet & CAN Option software checks the status of the Mx4 Octavia board. If any problems are found with the Mx4 Octavia board, the Module Status LED will be set to steady RED. The Module Status and Network Status LED's will display a steady green only when both the Mx4 and watchdog requirements are met.

ACC8 DeviceNet & CAN Option Specifications

DeviceNet Data	I/O Data	User Selectable Values
Power Consumption	DeviceNet current draw	33 mA @ 24 V d.c. 72 mA @ 11 V d.c. (worst case)
Isolation	2500 V r.m.s. for 1 minute per UL 1577	Not Applicable
Communication Rates (KBaud)		125 250 500
Messaging Capabilities		Group 2 Only Slave Explicit Messaging Polled IO
Status Indication	Module Status Network Status	Bi-color (red/green) LED Bi-color (red/green) LED
Network Address		00-63 (Factory default = 63)
Inputs		Maximum 110 mA @ 7.5 V d.c. minimum CAN isolation power; maximum of 50V d.c.
Dimensions	Depth Width Height	114 mm (4.50 in.) 162 mm (6.40 in.) 15 mm (0.59 in.)
Weight		257 g (0.53 lb.)
Environmental	Operating Temperature Storage Temperature	0^{o} to 100^{o} C -40^{o} to 125^{o} C

Table C-1: ACC8 Specifications