CAN-Controlled Application User Manual For Azure Dynamics DMOC Motor Controller

MAN-080003-001 DECEMBER 2009

AZURE DYNAMICS INC.
An ISO 9001:2000 Certified Company
9 Forbes Road
Woburn, MA
USA 01801

T 781.932.9009
F 781.932.9219

productsupport@azuredynamics.com
www.azuredynamics.com

Table of Contents

Forward 3
- Caution 3
- Contact Information 3
- How to Report Errors 3

Safety 4
- Warning Labels 4
- Safety Symbols 4

Overview 5

Application Software 6
- CAN Protocol / Message Identifiers 6
- Communication Management 6

CAN Messages 8
- Number Formats 8
- Messages 8

Control Modes 10
- Speed Regulator 10
- Pedal Control for Debug Mode 11
- Interlocks & Safety 12
- Principal Application Variables 12
- Application Parameters 13

Electrical Interface 15

Troubleshooting 19

Forward

Caution

The information provided in this manual is intended for use by persons with appropriate technical skills. Any effort to perform repairs to or service your unit without the proper tools or knowledge required for the work can result in personal injury and product damage and will void your warranty!

Contact Information

Please feel free to call with any suggestions that you may have regarding the content of your manual. If additional service information is needed or to order replacement parts, please call Monday–Friday 8:30AM–5:30PM USA Eastern Time:

T 781.932.9009

F 781.932.9219

E productsupport@azuredynamics.com

How to Report Errors

If, while reading through this manual, you discover an error in the technical information provided, Azure Dynamics asks that you notify its Product Support Department. Please be prepared to provide the following information:

- Your name
- Name and edition of your manual
- Page number(s) where the error(s) appear
- Part number and serial number of your unit

Information contained in this manual is based on the latest product information available at the time of publication. The right is reserved to make changes at any time without notice.

Safety

For your safety and the safety of others, please read and understand this entire manual before installing the components you have received from Azure Dynamics. If you have questions regarding the contents of this manual, please call the Azure Dynamics Product Support Department before proceeding.

Warning Labels

Labels indicate areas in a procedure where you should take appropriate precautions. Labels include:

WARNING AND DANGER

RISK OF ELECTRIC SHOCK

Safety Symbols

Always use caution when working on or around any electrical equipment. Wear eye protection at all times. The following symbols will be located in your manual to indicate sections in a procedure where extra caution and/or safety equipment is required.

HEARING PROTECTION REQUIRED

EYE PROTECTION REQUIRED

Always follow any safety instructions that are given at the beginning of a procedure. If you are uncertain as to the safe and proper handling of your equipment, contact Azure Dynamics Product Support.

Overview

The Azure Dynamics Digital Motor Controller (DMOC) is a rugged traction inverter for controlling three-phase AC motors and generators. Flexible software architecture allows for application-specific customization by loading software application modules. These application modules communicate with the motor control core and implement the interface to the higher level controls or directly to the driver inputs and outputs.

This manual discusses the "CAN Controlled" application layer which configures the DMOC for a system in which a central vehicle control unit (VCU) commands the DMOC over CAN (Controller Area Network). For general information regarding the DMOC, including important safety instructions and warnings, the DMOC445 and DMOC645 User Manual should be consulted, which is distributed and revised separately. See Table 1 for a list of relevant manuals.

At the heart of the "CAN control" application layer is a proportional-integral (PI) speed regulator with variable torque limits. The speed set-point as well as the torque limits are transmitted over CAN and may be modified by the VCU at a rate of up to 20 messages per second. If the speed set value can be reached within the torque limits, then speed regulation is achieved. If, however, the limits are too restrictive, then the regulator will run against its limits and the drive becomes, essentially, torque controlled. This gives the user the option to run the DMOC as a classical torque source or to have it maintain a speed, as necessary.

Regenerative braking occurs automatically whenever the sign of the torque and the sign of speed are opposite. If a lower speed than the motor is currently at is commanded, negative torque will be applied (and the system will regen) to the set torque limit. If a higher speed than the motor is currently at is commanded, positive torque will be applied (and the system will accelerate) to the set torque limit.

For testing and troubleshooting convenience, the application layer can also be configured to read a speed set-point from an analog input. In this mode, two digital inputs act as forward/reverse/neutral switches and determine the direction of rotation for the drive.

Azure's PC-based diagnostics/calibration tool "ccShell" allows the user to access and modify DMOC calibration parameters and to visualize and capture signals in real time. While the meanings of the most important calibration parameters and signals of the DMOC core are described in this document, the reader is referred to the ccShell User Manual for information on how to install and use this tool.

Table 1: List of Relevant Manuals

Document Name	Document Number
DMOC445 and DMOC645 User Manual	MAN-080001
Pedal Controller Application User Manual	MAN-080002
CAN Controlled Application User Manual	MAN-080003
ccShell User Manual (Please note, this manual is distributed as part of the ccShell software. It is available under ccShell's "Help" menu.)	MAN-080008

Application Software

CAN Protocol / Message Identifiers

The “CAN Controlled” application layer supports CAN 2.0 with standard 11-bit identifiers. Please see the DMOC445 and DMOC645 User Manual for information on configuring the baud rate and the sampling point.

The DMOC is controlled by one command message and responds with two status messages. Message identifiers can be configured by means of the parameters in Table 2.

Table 2: CAN Message Identifiers

Parameter	Description
EEXCANCommandID	CAN identifier for control message
EEXCANStatusID1	CAN identifier for status message 1
EEXCANStatusID2	CAN identifier for status message 2

Note: all CAN IDs are displayed as decimal numbers in ccShell, while many CAN tools work more naturally in hex. For example, the default value for the command message ID “EEXCANCommandID” displays in ccShell as 528, but in hex is 0x210.

Communication Management

The CAN communication is monitored by means of two finite state machines (FSM). The states of the individual FSM are represented by variables which can be viewed using ccShell.

A first FSM is used to indicate that CAN messages are being received and to handle and clear CAN faults. The variable ISR2CANComState can take one of the values indicated in Table 3.

Table 3: ISR2CANComState Enumerations

Variable	ISR2CANComState	
State	Name	Description
0	CAN_STATE_POWERUP	CAN not enabled
1	CAN_STATE_OFFLINE	CAN offline – no CAN link established, sending “Ping” message
2	CAN_STATE_ONLINE	Receiving messages via CAN, sends all status messages
3	CAN_STATE_FAULT	Fault on CAN line

ISR2CANComState is used to provide diagnostics on the CAN communication state with the DMOC. Typically the CAN state will be OFFLINE on startup, when the CANComState is OFFLINE, the DMOC will continue to send one CAN message, termed the “Ping” message, to show that it is alive on the CAN bus. If a command message is received the DMOC transitions to CANComState ONLINE, and all status messages are sent at the configured frequency. If the CAN command messages are not received, the CAN state will time out, transition through FAULT and return to OFFLINE (when it will again send out a “Ping” message).

While the first CAN FSM monitors and manages the CAN bus in general, a second FSM, ISR2DeviceConnectState, is used to monitor the communication with the VCU. Contact is considered to be established if the Command Message is received within a certain (configurable) interval. Only if the FSM is in the “ON” or “CONNECTED” state, will the drive enable. In all other states the inverter will be passive and the electric motor will not produce torque. Please see Table 4.

Table 4: CAN FSM Variables

Variable	ISR2DeviceConnectState	
State	Name	Description
0	CONNECT_STATE_POWERUP	Communication not enabled
1	CONNECT_STATE_DISCONNECTED / OFF	Waiting for contact with system controller; “Ping” message being sent
2	CONNECT_STATE_CONNECTED / ON	Contact established, all messages are being sent
3	CONNECT_STATE_FAULT	Contact lost with system controller; remains in this state for two seconds (programmable) before transitioning to state 1

The parameters in Table 5 are used for configuring the CAN FSMs:

Table 5: CAN FSM Configuration Parameters

Name	Description
EEXCANConnectFaultTime	Controller disable time after CAN watchdog trip (in increments of 10ms)
EEXCANConnectTimeout	CAN watchdog 2 timeout for Connect State (in increments of 10ms)
EEXCANRxTimeout	CAN watchdog 1 timeout for Com State FSM (in increments of 10ms)
EEXCANTxPeriod	Transmission interval for status messages (in increments of 10ms)

Other general CAN parameters are shown in Table 6.

Table 6: Other general CAN parameters

Name	Description
EEXAutoFaultClearTime	Azure internal use only
EEXCANDMOCDiagReqtID	Reserved for future Azure Diagnostics; you may change this ID if it conflicts with another CAN device on your vehicle.
EEXCANDMOCDiagRespCycleSec	Azure internal use only
EEXCANDMOCDiagRespID	Reserved for future Azure Diagnostics; you may change this ID if it conflicts with another CAN device on your vehicle

CAN Messages

Number Formats

All 16 bit signals in CAN messages are in the signed Q10 format. This means that 2^10 = 1024 corresponds to 1 p.u. (per-unit), and -1024 equals -1 p.u. Negative numbers are represented in the two's complement, as shown in Figure 1. All messages are little endian (Motorola format), least significant byte (LSB) first.

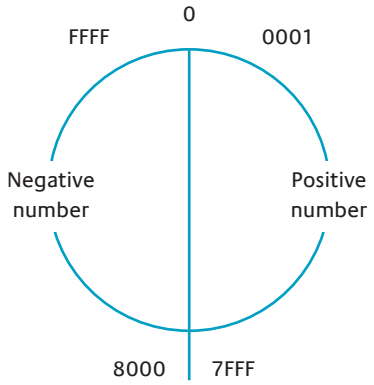


Figure 1: CAN Message Negative Numbers

In order to translate the per unit numbers into physical values, they need to be multiplied by their reference. The reference values are as follows:

Speed:	1 p.u. = 2500 rpm
Torque:	1 p.u. = 100 Nm
Temperature:	1 p.u. = 100 C
Voltage:	1 p.u. = 200 V
Current:	1 p.u. = 400 A

Please note that the motor temperature reported is raw PTC voltage and therefore not reported in ºC or ºF.

Messages

The Command Message is sent by the VCU to the DMOC and determines the motor speed set-point while imposing certain torque limits, see Table 7. The control message needs to be sent periodically, at an interval which is shorter than both EEXCANConnectTimeout and EEXCANRxTimeout. If no message is received for the given timeout period, the DMOC automatically enters a safe state for a minimal duration of EEXCANConnectFaultTime. Afterward, the DMOC will resume normal operation, provided that the command message is received at its specified interval.

Note, for the message structure defined in this section to be valid, the DMOC parameter EEXCANControlScheme must be set to 1.0.

Table 7: Control Message (from VCU)

ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
	LSB	MSB	LSB	MSB	LSB	MSB	LSB	MSB
EEXCANCommandID	CANSpeedSet (1024 = 2500 rpm)		LowerTorqueLimit (1024 = 100 Nm)		UpperTorqueLimit (1024 = 100 Nm)		reserved for factory use VCU MUST SEND 0x00, 0x00 (ALL ZEROS)	

The DMOC responds with two status messages, which are being transmitted at the rate configured by EEXCANTxPeriod. While Status Message 1 is transmitted at all times (as the "Ping" message) and can serve as a DMOC "heartbeat" message to the VCU, Status Message 2 is only sent when command messages are being received. See Table 8 and Table 10.

Table 8: Status Message 1 (to VCU)

ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
	LSB	MSB	LSB	MSB				
EEXCANStatusID1	Actual Speed (1024 = 2500 rpm)		Actual Torque (1024 = 100 Nm)		Status	reserved		
ccShell variable	ISR2Hertz		ISR2RealTorque		ISR2 Status Code			

The ISR2StatusCode encodes status information in a bit-wise fashion, as shown in Table 9.

Table 9: ISR2StatusCode Descriptions

Bit #	0	1
0	contactor open	contactor closed
1	power stage not ready	power stage ready
2	power stage OK	power stage faulted
3	more torque available	max torque limit reached
4	more power available	max power limit reached
5	no thermal limit active	thermal limit active
6-7	reserved	

Note that in the ready state, the thermal limit active signal can be ignored, as the IGBTs are "limiting" current before the power stage is enabled.

Table 10: Status Message 2 (to VCU)

ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
	LSB	MSB	LSB	MSB	LSB	MSB	LSB	MSB
EEXCANStatusID2	Heatsink Temp. (1024 = 100ºC)		Motor Temp. (PTC) (1024 = 1, a normalized voltage)		DC Voltage (1024 = 200 VDC)		Estimated DC Battery Current* (1024 = 400A)	
ccShell variable	ISR2HeatSinkTemp		ISR2MotorPTCVoltage		ISR2BatVoltage		ISR2EstBatCurrent Note, positive is flowing out of battery (into DMOC)	

* Note that the ccShell parameter EEXCANTxBatteryCurrent must be set to 1 (which is the default) for the battery current status information to be transmitted. This setting is only present in CAN code starting in December 2008.

Control Modes

Speed Regulator

A block diagram of the speed regulator is shown in Figure 2. It consists of a proportional-integral (PI) controller with output saturation and anti-windup. This structure lends itself well for operating the drive in either a speed-controlled or torque-controlled mode. Furthermore, the anti-windup feature assures smooth transition from one mode to the other.

The limits of the saturation block (located after the PI block) can be modified over CAN by means of the “UpperTorqueLimit” and “LowerTorqueLimit” signals. Depending on the values of those limits, the following scenarios exist:

1. LowerTorqueLimit < UpperTorqueLimit: The speed regulator is active within the window given by the two limits.
2. LowerTorqueLimit = UpperTorqueLimit: The output of the speed regulator block is forced to the exact value given by the limits; the drive is torque controlled.
3. LowerTorqueLimit > UpperTorqueLimit: The motor torque is set to zero.

It should be noted that the torque of the motor may be restrained further based on slew rates, thermal limits (inverter, motor) or for other protection purposes.

Furthermore, the speed set-point can be slew rate limited by means of the EEXHertzSetSlewPSec parameter.

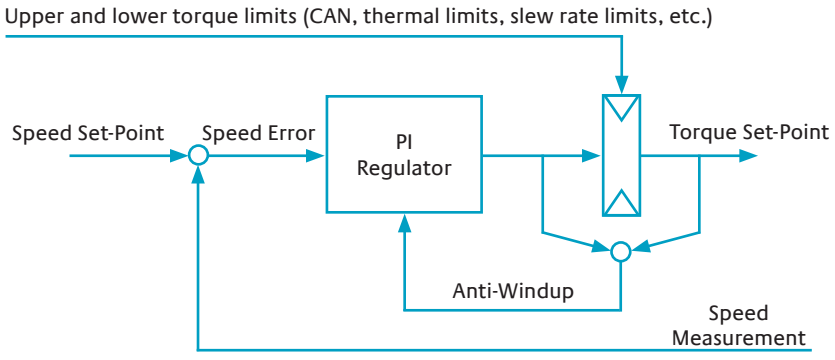


Figure 2: CAN Speed Regulator

The speed regulator updates its torque command at a rate of 100 Hz. The parameters in Table 11 can be accessed and modified using ccShell:

Table 11: Speed Regulator Parameters

Parameter	Description
EEXHertzKi	Integral gain of speed regulator [unitless] Replaced by EEXHertzKiHiRes in newer code
EEXHertzKiHiRes	Integral gain of speed regulator [unitless] Replaces EEXHertzKi and EEXHertzKiPrescale in newer code
EEXHertzKp	Proportional gain of speed regulator [unitless]
EEXHertzSetSlewPSec	Slew-rate limit of speed set-point [rpm/s]
EEXHertzKiPrescale	Scaling factor (<= 1) for Ki (in order to increase resolution) Replaced by EEXHertzKiHiRes in newer code

In newer code, EEXHertzKi and EEXHertzKiPrescale are replaced by EEXHertzKiHiRes. The relationship is EEXHertzKiHiRes = EEXHertzKi * EEXHertzKiPrescale.

Pedal Control for Debug Mode

For testing and troubleshooting, it is possible to bypass the CAN control and use analog and digital controls instead for commanding speed. This is achieved by setting EEXControlMode = 1. While pedal-controlled, the forward/reverse digital inputs are used to determine the direction of rotation. When in "neutral" mode, no torque is applied to the motor.

More information regarding wiring for pedal controlled operation can be found in the "Electrical Interface Section". Please note that this is a very simple pedal input, designed for testing and troubleshooting only, and is not equivalent to the Azure Dynamics Pedal Controlled Software. See Table 12 and Table 13. For the pedal map when in pedal test mode, see Figure 3.

Please also note that if you plan to use EEXControlMode = 1, you should short pins 7 and 20 on the DMOC 35-pin connector together. This removes the standby torque request and allows DMOC operation.

Table 12: DMOC Direction in Pedal Mode (for testing only in CAN slave software)

Fwd	Rev	Mode
Floating	Floating	DMOC in "neutral"
GND	GND	DMOC in "neutral"
GND	Floating	"forward" rotation
Floating	GND	"reverse" rotation

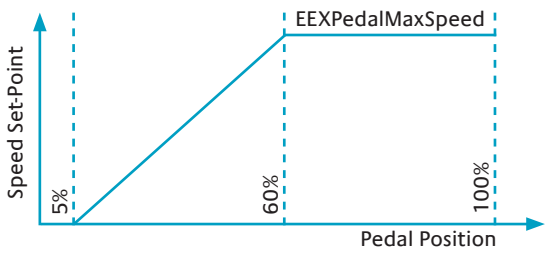


Figure 3: Pedal Map for Pedal Mode (for testing only in CAN slave software)

Table 13: Parameters for Pedal Mode (for testing only in CAN slave software)

Parameter	Description
EEXControlMode (formerly called EEXPedalControlled)	0 = CAN control 1 = Pedal test mode; analog inputs used 2 = Not recommended for in-vehicle use. Sets desired speed to EEXPedalMaxSpeed if the pedal direction switch input is at FORWARD.
EEXPedalMaxSpeed	Maximum speed set-point for analog control (aka pedal control)
EEXPosMaxTorque	Maximum positive torque when pedal controlled
EEXNegMaxTorque	Maximum negative torque when pedal controlled

In pedal control debug mode, DMOC faults are cleared by releasing the pedal completely.

Interlocks & Safety

At powerup, the DMOC must receive control messages with the torque limits set to zero in order to activate the powerstage. An activated powerstage is indicated in CAN Status Message 1 when Status-Bit 0 and Status-Bit 1 are both 1. Faults can be acknowledged by setting the torque limits to zero in the VCU control message. This will clear any non-critical faults and the DMOC will reset Status-Bit 2 back to zero.

The DMOC also has a “drive enable” and a “drive disable” signal (digital inputs) which are both active low (i.e. need to be pulled to GND to be active). For the DMOC to enable “drive enable” has to be active and “drive disable” needs to be passive. If the “drive enable” feature is not desired, it can be switched off by setting the calibration EEXNoIgnSwitch to 1. See Table 14. Grounding “drive disable” will always result in disabling the DMOC.

Table 14: Drive Enable Parameter

Parameter	Description
EEXNoIgnSwitch	1 = drive enable input ignored

For a shutoff, it is recommended that the 12V be disconnected. Removing the 12V power will immediately disable the DMOC power supply and shut the unit down. For additional safety, it is recommended that an emergency high voltage disconnect be provided as well; however, this should be a normally closed switch that is only actuated in emergencies or for maintenance. If a contactor is used as a HV disconnect, it could interfere with the operation of the DMOC’s internal contactor, resulting in excessively long precharge times, or damage the DMOC.

Principal Application Variables

Table 15 shows the most frequently viewed application variables. The reader is referred to the DMOC445 and DMOC645 User Manual for information about other important DMOC variables. Note, some variables are not directly relevant to customers, but are used by Azure for diagnostics and troubleshooting. These are marked with an * in Table 15. Also, not all variables are viewable in ccShell, depending on each customer’s .ccs file and software revision level.

Table 15: DMOC Application Variables

Variable	Description
ISR2AbsStandbyTorqueDesired*	CAN slave software only; for Azure internal use
ISR2CANComState	Reflects state of CAN bus Finite State Machine
ISR2CANFFTorque*	CAN slave software only; for Azure internal use
ISR2CANLowerTorque	Lower torque limit received over CAN
ISR2CANSpeedSet	Speed set-point received over CAN
ISR2CANStandbyTorque*	CAN slave software only; for Azure internal use
ISR2CANUpperTorque	Upper torque limit received over CAN
ISR2DeviceConnectCtr1	Countdown timer for CAN message interval
ISR2DeviceConnectCtr2	Countdown timer for CAN message interval
ISR2DeviceConnectState	Reflects state of CAN connection FSM, monitors communication with VCU: CONNECT_STATE_POWERUP = 0 — Communication not enabled. CONNECT_STATE_DISCONNECTED = 1 — Waiting for system controller contact, "Ping" message CAN Status Message 1, CANID = EEXCANStatusID1 being sent. CONNECT_STATE_CONNECTED = 2 — Contact established, all messages being sent. CONNECT_STATE_FAULT = 3 — Contact lost with system controller; remains in this state for two seconds (programmable) before transitioning to CONNECT_STATE_ DISCONNECTED
ISR2FwdRv	When in CAN slave pedal test mode, shows position of direction switch: DIRECTION_SW_REVERSE = -1 DIRECTION_SW_NEUTRAL = 0 DIRECTION_SW_FORWARD = 1
ISR2HertzDesired	Desired speed set-value to regulator
ISR2HertzError	Difference between desired and actual speed, used as input to speed regulator
ISR2HertzSet	Actual speed set-point
ISR2PedalS	Only active if CAN pedal test mode is activated; shows pedal position
ISR2PedMode	Pedal zone indicator (acceleration, neutral, or braking)
ISR2SpeedRegLowerTorque*	Torque limit for speed regulator output
ISR2SpeedRegTorqueDesired	Output from speed regulator (before limits)
ISR2SpeedRegUpperTorque*	Torque limit for speed regulator output

Application Parameters

Table 16 summarizes the most frequently used application parameters. Please refer to the DMOC445 and DMOC645 User Manual for information about other important DMOC parameters.

Table 16: DMOC Application Parameters

Parameter	Description
EEXBrakeRamp	Ramp over which the torque is derated to zero below EEXNoBrakeSpeed (only if EEXEnableBrakeRamp = 1)
EEXCANCommandID	CAN identifier for control message
EEXCANConnectFaultTime	Controller disable time after CAN watchdog trip (in increments of 10ms)
EEXCANConnectTimeout	CAN watchdog timeout for Connect State (in increments of 10ms)
EEXCANControlScheme	1 = message structure defined in this document (including estimated DC battery current transmitted) is valid
EEXCANKBitS	CAN baud rate (in kBits/s)
EEXCANRxTimeout	CAN watchdog timeout for Com State FSM (in increments of 10ms)
EEXCANStatusID1	CAN identifier for status message 1
EEXCANStatusID2	CAN identifier for status message 2
EEXCANTSeg1	CAN bit timing TSeg1
EEXCANTSeg2	CAN bit timing TSeg2
EEXCANTxBatteryCurrent	1 = estimated DC battery current transmitted NOTE, THIS PARAMETER IS VALID IN CAN SLAVE CODE RELEASED IN DECEMBER 2008 OR LATER.
EEXCANTxPeriod	Transmission interval for status messages (in increments of 10ms)
EEXControlMode	(formerly EEXPedalControlled) 0 = CAN controlled 1 = controlled by pedal box 2 = Not recommended for in-vehicle use. Sets desired speed to EEXPedalMaxSpeed if pedal direction switch input is at FORWARD.
EEXEnableBrakeRamp	0 = No brake ramp 1 = No negative speed allowed; brake ramp implemented (for engine protection)
EEXHertzKi	Integral gain of speed regulator (Replaced by EEXHertzKiHiRes)
EEXHertzKiHiRes	Integral gain of speed regulator (Replaces EEXHertzKi and EEXHertzKiPrescale)
EEXHertzKiPrescale	Scaling factor (<= 1) for Ki (in order to increase resolution) (Replaced by EEXHertzKiHiRes)
EEXHertzKp	Proportional gain of speed regulator
EEXHertzSetSlewPSec	Slew-rate limit of speed setpoint
EEXMaxAccelPower	Max power out of batteries
EEXMaxRegenPower	Max power in to batteries
EEXNegMaxTorque	Max negative torque when pedal controlled (positive direction according to EE2ShaftDirection)
EEXNoBrakeSpeed	Lowest allowable speed at full torque (only if EEXEnableBrakeRamp = 1)
EEXNoIgnSwitch	0 = enable controlled by external signal 1 = always enabled
EEXPedalControlled	Renamed EEXControlMode
EEXPedalMaxSpeed	Maximum speed when using pedal box for controlling DMOC
EEXPosMaxTorque	Max positive torque when pedal controlled (positive direction according to EE2ShaftDirection)
EEXTorquePUScale	Scaling between internal torque reference and CAN torque reference NOTE, THIS PARAMETER MUST BE SET TO 0.32715 FOR THE DMOC445 AND DMOC645. THIS PARAMETER IS NOT INCLUDED IN CAN SLAVE SOFTWARE RELEASED IN DECEMBER 2008 OR LATER VERSIONS
EEXTorqueSlew	Slew rate limit on torque set-point
EEXZeroTorqueTimeout	Azure internal use only

Electrical Interface

Besides the high voltage connections, which are documented in the DMOC445 and DMOC645 User Manual, a number of low voltage signals are used for the “CAN Controlled” application module. The 12V auxiliary supply needs to be able to source 10A of current and must be protected by a 15A fuse. The auxiliary supply also acts as an enable signal for the internal power supply of the DMOC. In other words, a DMOC requires 12V to be present in order to operate. If pedal controlled operation is desired for testing or troubleshooting, a 5 kΩ potentiometer and a forward-reverse switch need to be wired to the DMOC as well.

Please see Figure 4 and Figure 5 for the Azure Dynamics DMOC Foundation Harness (part of the Azure Dynamics DMOC Interface Kit; mates to 35 pin connector) diagram. As mentioned above, the pedal and forward-reverse switch is only functional in the “CAN controlled” application module if the pedal test mode is enabled. i.e. if the DMOC parameter EEXControlMode = 1. For more information, please see the section entitled “Pedal Control for Debug Mode” in this manual.

Please note, some existing versions of the DMOC Foundation Harness may not have pins 30 and pins 23 and 12 populated on the mating DMOC 35-pin connector. Also, existing versions of the Foundation Harness do not include the back-up light relay. If you need assistance, please ask your Azure Dynamics or distributor contact.

Note, the CAN Controlled DMOC has no brake light function, no regen disable function and no power saver. Please be aware of this when you review Figure 4 and Figure 5.

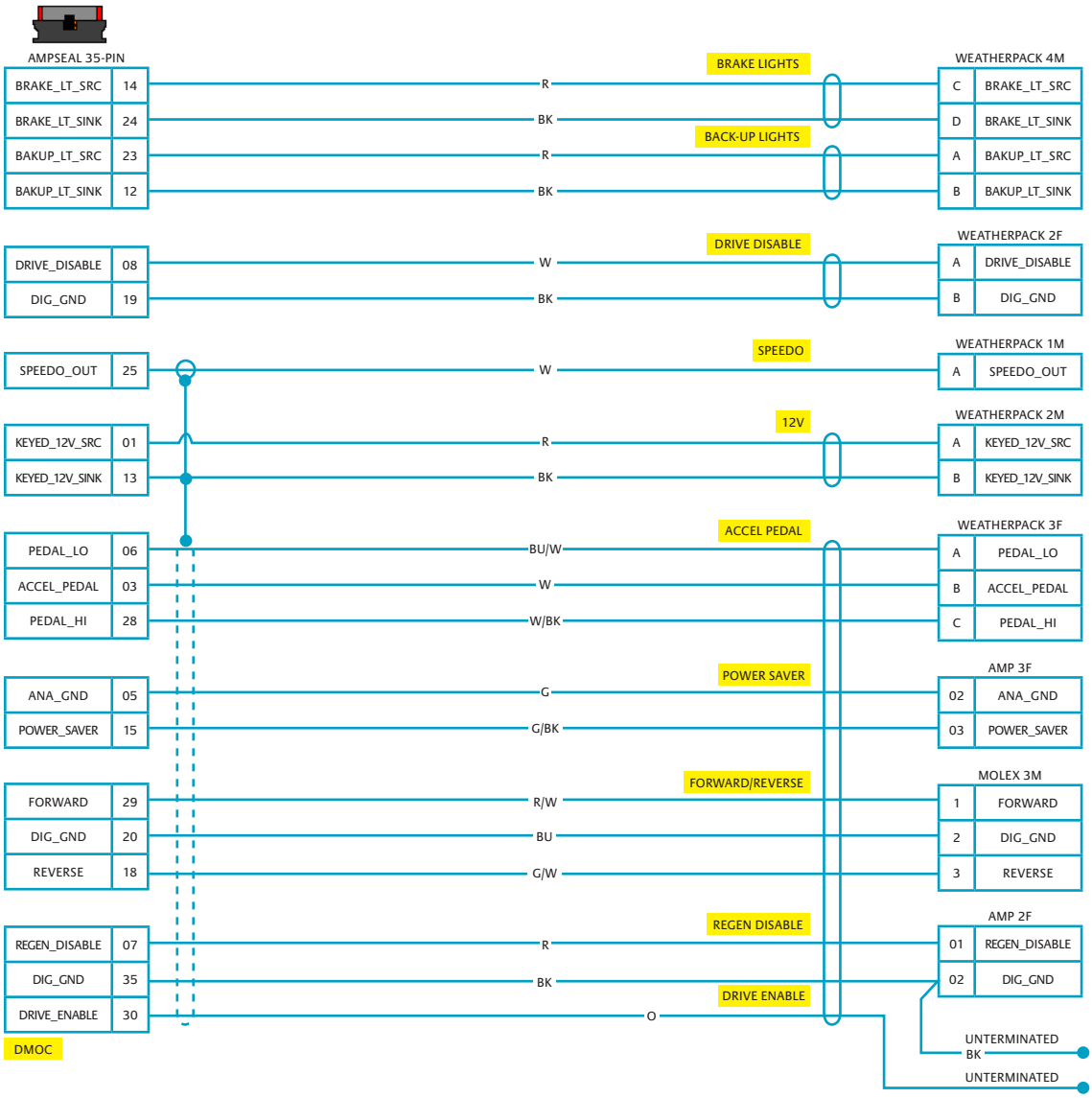
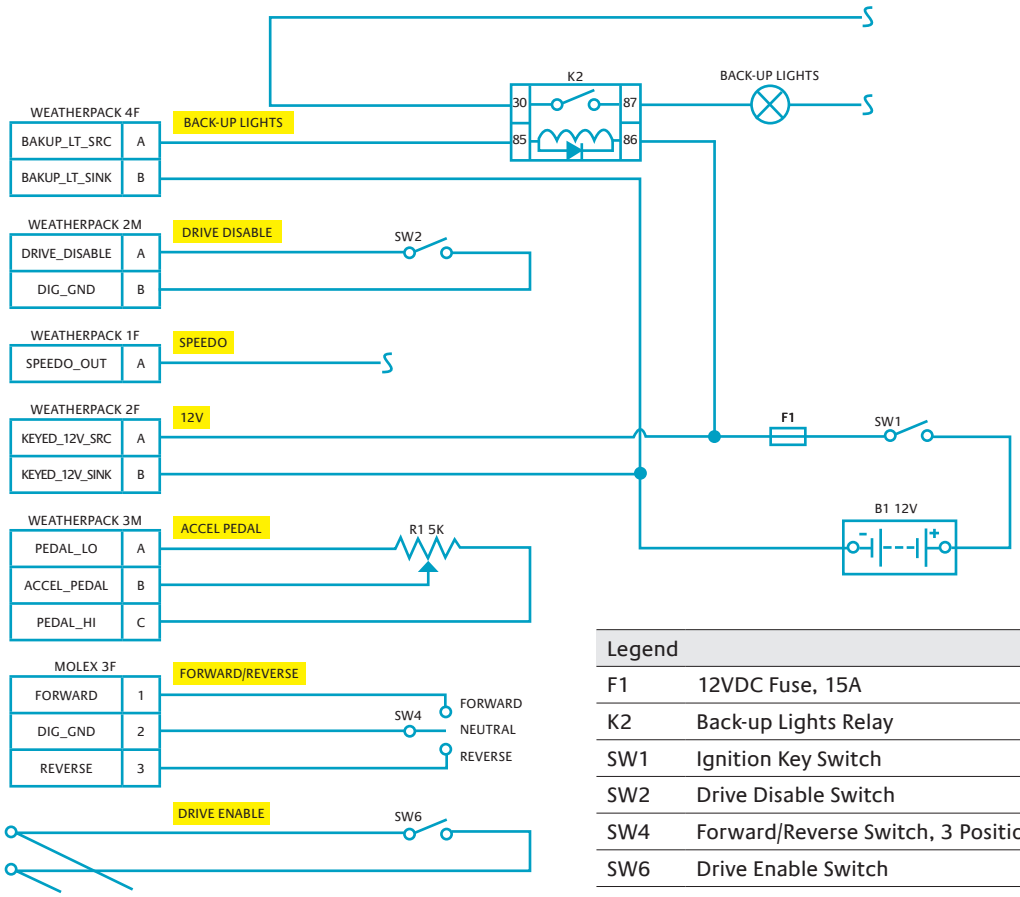


Figure 4: DMOC I/O Interface Harness (Foundation Harness)

Note, the CAN Controlled DMOC has no brake light function, no regen disable function and no power saver. Please be aware of this when you review Figure 4 and Figure 5.

NOTE: Vehicle reverse lights wiring will vary. Diagram is provided for reference only.



NOTE: Splice directly to the wires or add a connector of your choice.

Legend	
F1	12VDC Fuse, 15A
K2	Back-up Lights Relay
SW1	Ignition Key Switch
SW2	Drive Disable Switch
SW4	Forward/Reverse Switch, 3 Position
SW6	Drive Enable Switch
R1	Accelerator Pedal Potentiometer, 5K
B1	12VDC Battery

Figure 5: DMOC Customer Interface I/O

The CAN connection is made at the communications connector (8 pin Ampseal); see Table 17. Note that the DMOC does not provide any CAN termination. A 120 Ω termination resistor on each end of the CAN bus should be provided by the customer.

Table 17: DMOC 8-pin Connector Pin-out

Communications Connector, 8-pin Ampseal		
Pin #	Signal	Function
1	RS232_TXD	RS232 Tx
2	RS232_RXD	RS232 Rx
3	GND_B	Comm. Gnd: RS232 & CAN
4	CANH	CAN High
5	GND_B	Comm. Gnd: RS232 & CAN
6	CANL	CAN Low
7	GND_B	Comm. Gnd: RS232 & CAN
8	Chassis	Chassis GND

The DMOC interface cables (to the DMOC 8-pin connector) use a standard DB9 pinout for CAN, as shown in Table 18:

Table 18: DB9 CAN Pin-Out

Signal Name	Pin on DB9 Connector
CAN L	2
CAN H	7
Shield	3

Troubleshooting

To clear DMOC faults in CAN slave control…send a zero torque command.

If you can't communicate with your DMOC using ccShell…see the ccShell User Manual.

If you are communicating with your DMOC using ccShell but are not communicating with it via CAN…

1. Check that you are using decimal IDs instead of hex IDs. All CAN IDs are displayed as decimal numbers in ccShell. Many CAN tools work more naturally in hex, so be aware that for example in in Azure's CAN drive code, the default "Ping" message of 592 in decimal = 0x250 in hex.

2. Confirm there are no CAN errors detected by your hardware. Your CAN monitor should provide diagnostics about the CAN error rates (which are detected by the hardware). Check the following:
   - There are no passive errors on the CAN bus (check that the bus is properly terminated—120 Ohm terminating resistor on each end of CAN bus) and that the CAN Baud rate is set correctly, CAN H and CAN L not shorted together, etc.)
   - There are no active errors on the CAN bus (check the CAN bus is grounded correctly); in very noisy systems a few errors per minute may be observed when the DMOC is switching. Lots of CAN errors usually indicates inadequate shielding or grounding. You may also need to adjust the sample point of data in the CAN message to reduce errors, although typically no change is required (EEXCANTSeg1, EEXCANTSeg2).
   - Check that the baud rate for all CAN nodes is the same.
3. Verify "Ping" message is observed on the CAN bus.

4. Verify your CAN monitor hardware works by observing some CAN traffic on the bus. Disconnect all other devices but the DMOC and your CAN hardware and observe the "Ping" message is being sent from the DMOC. Even when ISR2CANComState = OFFLINE, the DMOC should send a "Ping" message to let other devices see that it is alive. This "Ping" message is the CAN status message 1 (CAN ID is EEXCANStatusID1).

5. If the "Ping" message is observed, then verify your command message is being sent on the bus, by using the CAN observation tools that came with your hardware. If you have verified your CAN command is transmitted and observed at the appropriate rate on the bus, then contact Azure as your DMOC may have been damaged. However, if "Transmit" is working, it is very unlikely that "Receive" is not working.

6. If no "Ping" message is observed, verify that your wiring to the DMOC is correct, that the DMOC is powered on (high and low voltage required). If ccShell is working, then the DMOC is on. If the CAN wiring is also correct (including termination), then the DMOC may have been damaged; contact Azure.

Also see the DMOC445 and DMOC645 User Manual for additional information about CAN baud rate settings and general CAN Configuration Parameters.

Please note, Hyperterminal only confirms one-way communication (DMOC transmitting to laptop). ccShell, on the other hand, requires two-way communication (DMOC transmitting to, and receiving from, laptop).

AZD
AZURE DYNAMICS
part of the solution™

DETROIT
14925 W 11 Mile Road
Oak Park, MI
USA 48237

T 248.298.2403
F 249.298.2410

VANCOUVER
3900 North Fraser Way
Burnaby, BC
Canada V5J 5H6

T 604.224.2421
F 604.419.6392

BOSTON
9 Forbes Road
Woburn, MA
USA 01801

T 781.932.9009
F 781.932.9219

TORONTO
4020A Sladeview Crescent, Unit 6
Mississauga, ON
Canada L5L 6B1

T 905.607.3486
F 905.607.6391

www.azuredynamics.com