CAN REFERENCE MANUAL

DOCUMENT NUMBER: 198-0000088

Record of Revisions:

Revision	Date	Description	Comments
001	9-26-2022	Initial release	Document created.

Read This First

While ElectroCraft believes that the information and guidance given in this manual is correct, all parties must rely upon their own skill and judgment when making use of it. ElectroCraft does not assume any liability to anyone for any loss or damage caused by any error or omission in the work, whether such error or omission is the result of negligence or any other cause. Any and all such liability is disclaimed.

About This Manual

This document describes the ElectroCraftCAN protocol supported by ElectroCraft drives with a CAN interface. Note that not all drives may support the full ElectroCraftCAN protocol, see specific drive user manual for details.

ElectroCraft
2 Marin Way, Suite 3
Stratham, NH 03885-2578

If you need Assistance …

Visit ElectroCraft online	World Wide Web: www.electrocraft.com

If you would like to …

- Receive general information or assistance
- Ask questions about product operation or report suspected problems
- Make suggestions or report errors in documentation

Contact ElectroCraft …

For regional technical, application and sales support for ElectroCraft and Hansen Products:

North America / USA / Mexico / South America / Central America
Contact: (844) 338-8114, sales@electrocraft.com

Europe (except Germany), Middle East, Africa, Australia
Contact EMEA Sales Team, +44 (0) 1270 508800, EMEAsales@electrocraft.com

Germany
Contact customer service, +49 (0) 711 727205 0, info@de.electrocraft.com

Asia
Contact customer service, sales@electrocraft.com

Contents

1 Introduction to CAN

ElectroCraft CAN employs a high-speed CAN protocol based on ISO 11898-2 and is capable of speeds up to 1Mbit/s. ElectroCraft CAN is used by a host controller to command and monitor ElectroCraft drives (or axis points).

ElectroCraft CAN defines the following:

- The process for how an address is assigned to a drive.
- Format and contents of message fields.
- Commands to be used for the host to interface with a drive.
- How a drive will respond to requests from the host.

ElectroCraft CAN will support a typical CAN network topology. This topology consists of two wires and at least two axis points. Axis points are any device capable of CAN communication. An example of this is shown in Figure 1 with multiple axis points (and their groups), and a PC with CAN tool acting as the host.

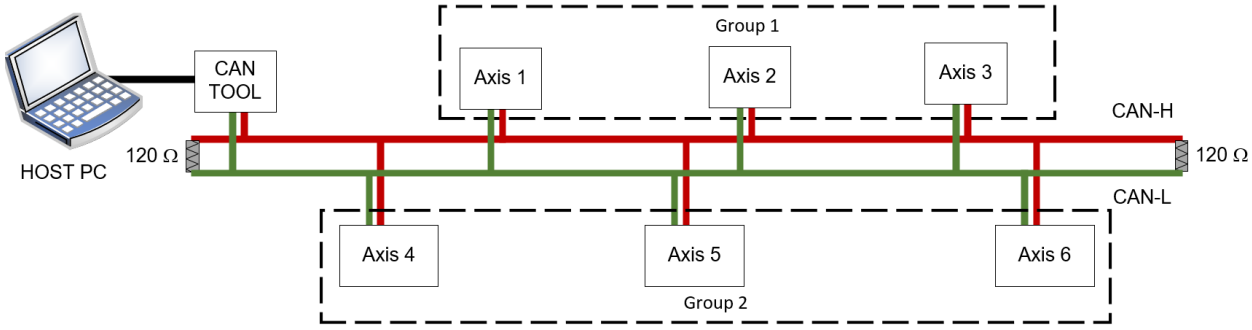

Figure 1: CAN Bus Topology

Also shown in Figure 1 is a 120 Ohm termination -- this is necessary at each end of the bus for ElectroCraft CAN (it is not required at each axis point). ElectroCraft standard drives do NOT include this termination.

The two wires (red and green) in Figure 1 represent the CAN high and CAN low differential signals. Figure 2 below shows an example of what these CAN signals may look like.

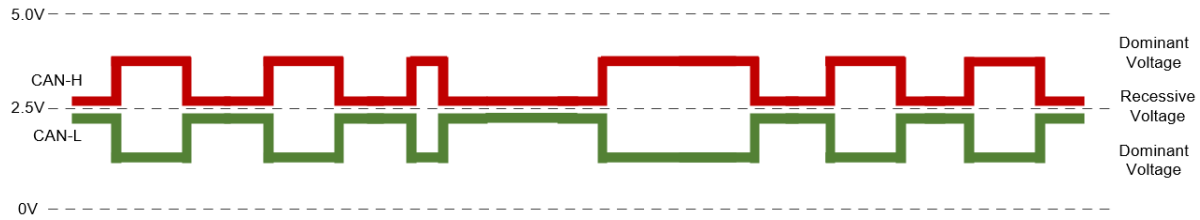

Figure 2: CAN Differential Signal

2 Communication Overview

The ElectroCraft CAN communications are organized as a series of host-node transactions, where the drive is always the client. The general transaction sequence is as follows:

1) Host transmits a CAN message to one or more drives.
2) If the Axis-ID matches a drive on the network, that drive will reply to the sender with a message. The response will depend on type of service message.
   a) Case 1: Set data or drive control. The drive will respond with a status indicating if the command is accepted or no by the drive.
   b) Case 2: Get data – the drive will respond with the requested data.

No response indicates the addressed drive is not active on the network because it is not connected or some other condition that prevents communications.

2.1 Addressing

Axis-ID Addressing

To communicate on the CAN bus network, each drive must have a unique non-zero AXIS-ID. The AXIS-ID is non-volatile and needs to be set just once. AXIS-IDs in the range 1-127 are available.

The AXIS-ID can be set either:

a) Assigned by CompleteArchitect™ (see software user manual, document number 198-0000021).
b) In hardware using the Axis-ID 4 position switch on the drive (SW1). Axis-IDs in the range 1-15 are available by this method.
c) CAN Axis-ID configured in CompleteArchitect™ as a base address plus hardware switches.
d) Dynamically assigned at power up over the CAN bus (see section 2.2 for more detail).

Group-ID Addressing

Every drive is a member of GROUP-ID 0 and so will always accept and act upon valid Group Messages to Group 0 (if AXIS-ID is non-zero). Group Messages to GROUP-ID 0 are “all call messages” that are active regardless of any other drive address settings. They can be used for network management for example network discovery or high priority functions such as immediate stop of all axes since Group Messages have a higher intrinsic CAN priority than Axis Messages and can address all drives simultaneously. A second GROUP-ID in the range 1-127 can be assigned to a drive using CompleteArchitect™.

2.2 Dynamic Addressing

In Dynamic Addressing, the drive will not join the network until signaled by the customer system controller (aka the host) through the CAN Enable on the CAN input connector. Each drive has a CAN enable input and a CAN enable output. These signals are intended to be connected in a daisy chain configuration where the output of one drive connects to the input of the next drive. Alternatively, the host can connect to each drive individually if desired.

In the daisy chain configuration, the CAN enable input of the first drive in the network is connected to the host. The host initiates the dynamic addressing process by activating the CAN enable input of the first drive. This drive joins the bus using a preset address stored in “CAN Axis-ID” parameter. The host will then assign a new unique AXIS-ID to the “Host Assigned CAN Axis-ID” variable of that drive (see section 4.2 for more detail). This address is stored in volatile RAM and is valid until the next power cycle or RESET.

The host can then command the newly configured drive to signal the next drive in the daisy chain to join the network with its preset address and can be similarly assigned a new unique address in RAM. The process is repeated for all drives on the CAN network to dynamically assign unique addresses to each. After the power is cycled, the drives join the bus using their original preset address.

2.3 CAN Frame

Data transmitted across the CAN network is organized into packets called frames. Figure 4 below shows a typical CAN frame.

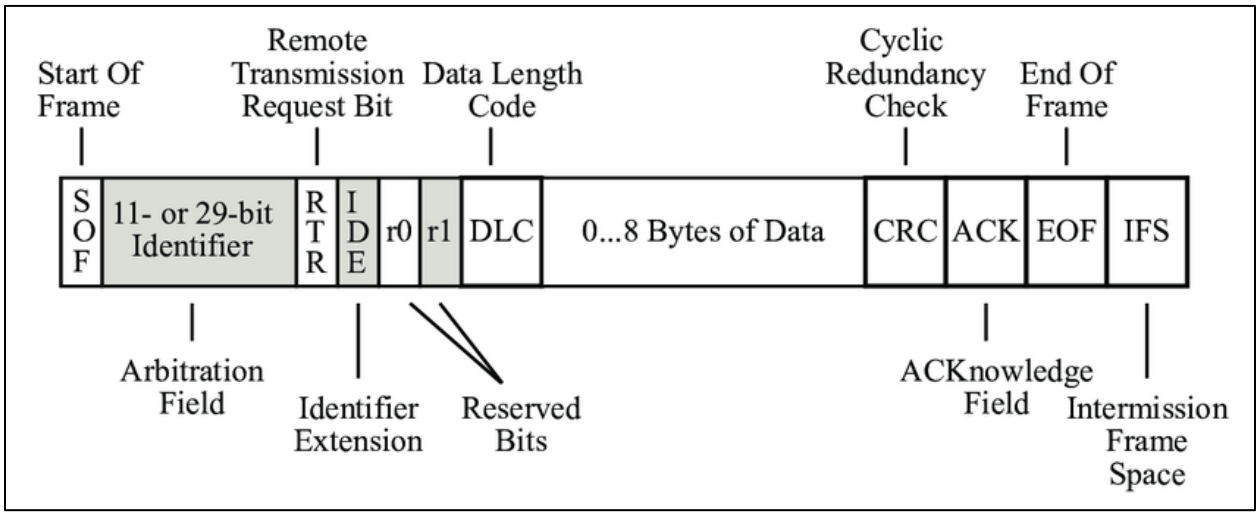

Figure 4: CAN Frame

2.3.1 CAN-ID Structure

ElectroCraft CAN uses a 29-bit extended Identifier (or CAN-ID) for arbitration. See details below in Figure 5.

	Priority {0,7}			Service Bit	Req/!Resp	Service ID {0,255}								Axis/!Group	Destination ID {0,127}								Host	Source ID {0,127}							
CAN Bit	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BYTE	3					2								1								0								

Figure 5: 29-bit Extended CAN-ID

1) **Priority:** The priority bits are used to provide 8 levels of prioritization to the protocol.
2) **Service Bit:** Service message bit will always be set to 1.
3) **Request/!Response bit**: This denotes whether the CAN message is a transmitted request bit (bit=1) or a received response bit (bit = 0).
4) **Service ID**: This identifies the action of the CAN message.
5) **Axis/!Group Bit**: This denotes whether the CAN message is intended for a single drive (bit = 1) or a group of drives (bit = 0).
6) **Destination ID**: Identifies the destination of the CAN message – this would correspond to the Axis (or Group-ID) for the destination drive (or drives)
7) **Host Bit**: Identifies that the CAN message is to be sent to the host (bit =1). This is used when the Basic CAN network is used between drives and the Host PC is connected by USB to one of the drives.
8) **Source ID Bits:** Identifies the sender of the CAN message – this would correspond to the Axis (or Group-ID) for the transmitting drive (or drives)

2.3.2 CAN Data Structure

The CAN data structure can vary depending on the command being used and the length of the data requested. More details are provided for each command in section 3.

3 Commands

3.1 Command Status Codes

Code	Name	Description
0x00	ACK_ACCEPTED	Command was accepted
0x01	ACK_ERROR	Command was rejected or failed
0x02	ADDRESS_OUT_OF_RANGE	Parameter address does not exist
0x03	VALUE_OUT_OF_RANGE	Value received is too large or small for the desired parameter
0x04	PROTECTED_PARAMETER	Parameter address is locked
0x05	NOT_WRITABLE_WHEN_ENABLED	Parameter is locked when drive is enabled
0x06	NOT_WRITABLE_WHEN_DISABLED	Parameter is locked when drive is disabled
0x07	DATA_LENGTH_MISMATCH	Data length in message does not match target parameter length
0x08	MOVE_IN_PROCESS	Command not valid while a move is in process. Command was ignored.

3.2 Reset, Enable and Brake Service Commands

Reset Command
This command will reset and disable the drive. This requires the drive to be re-enabled before operating the motor. No data is requested or returned with this command.

Name	Description	Type	Service Bit	Req/!Resp Bit	Service ID (Hex)
Reset	Sends Reset Command	Tx	1	1	0xFF

Enable Command
This command will enable the drive for motor operation. No data is requested in this case, but a status message will be returned.

Description	Type	Serv. Bit	Req/!Res Bit	Serv. ID (Hex)	DLC	Data1
Send Command	Tx	1	1	0x00	0x01	CMD
Status Reply	Rx	1	0	0x00	0x01	Status

CMD = 0x00 to disable drive
CMD = 0x01 to enable drive

Brake Command
This command will dynamically brake the motor output (like setting the brake digital input). No data is requested in this case, but a status message will be returned.

Description	Type	Serv. Bit	Req/!Res Bit	Serv. ID (Hex)	DLC	Data1
Send Command	Tx	1	1	0x01	0x01	CMD
Status Reply	Rx	1	0	0x01	0x01	Status

CMD = 0x00 to disable brake
CMD = 0x01 to enable brake

3.3 SET (Write) Service Command

For variables that can be modified, the “SET” command can be used.

Description	Type	Service Bit	Req/!Resp Bit	Service ID (Hex)
Set Command	Tx	1	1	0x20
Set CMD Status Reply	Rx	1	0	0x20

Description	DLC	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Data8
Set CMD for 8Bit (Tx)	0x04	Length=1	Addr0	Addr1	Byte0	-	-	-	-
Set CMD for 16Bit (Tx)	0x05	Length=2	Addr0	Addr1	Byte0	Byte1	-	-	-
Set CMD for 32Bit (Tx)	0x07	Length=4	Addr0	Addr1	Byte0	Byte1	Byte2	Byte3	-
Set CMD Status Reply	0x03	Status	Addr0	Addr1	-	-	-	-	-

3.4 GET (Read) Service Command

Parameters and read only variables can be retrieved using the “GET” command.

Description	Type	Service Bit	Req/!Resp Bit	Service ID (Hex)
Get Command	Tx	1	1	0x30
Get CMD Status Reply	Rx	1	0	0x30

Description	DLC	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Data8
Get CMD for 8Bit (Tx)	0x03	Length=1	Addr0	Addr1	-	-	-	-	-
Get CMD Status + Data	0x04	Status	Addr0	Addr1	Byte0	-	-	-	-
Get CMD for 16Bit (Tx)	0x03	Length=2	Addr0	Addr1	-	-	-	-	-
Get CMD Status + Data	0x05	Status	Addr0	Addr1	Byte0	Byte1	-	-	-
Get CMD for 32Bit (Tx)	0x03	Length=4	Addr0	Addr1	-	-	-	-	-
Get CMD Status + Data	0x07	Status	Addr0	Addr1	Byte0	Byte1	Byte2	Byte3	-

3.5 EEPROM Commands

Table 1.6: EEPROM Commands

EEPROM Commands	CAN ID			Data		Description
	Serv Bit	Serv Req	Serv ID	DLC	Data1	
EEPROM Restore Parameters	1	x	0xFA	0x01	0x04	Read parameters from EEPROM into active RAM.
EEPROM Save Parameters	1	x	0xFA	0x01	0x05	Save active RAM parameters to EEPROM.

3.6 SET Position Command

The SET position command can be used to instruct the drive to set the current position with a new position value (entered in Byte0 – Byte3).

Description	Type	Service Bit	Req/!Resp Bit	Service ID (Hex)
Set Position Command	Tx	1	1	0x40
Set Position Command Status Reply	Rx	1	0	0x40

Description	DLC	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Data8
Set Position Command	0x04	Byte0	Byte1	Byte2	Byte3	-	-	-	-
Set Position Command Status Reply	0x01	Status	-	-	-	-	-	-	-

3.7 Move Command

The MOVE command can be used to instruct the drive to move (or stop) the motor. The nature of the move (or stop) depends on the command code provided. See Table 1.7 for more details.

Description	Type	Service Bit	Req/!Resp Bit	Service ID (Hex)
MOVE Command	Tx	1	1	0x41
MOVE Command Status Reply	Rx	1	0	0x41

Description	DLC	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Data8
MOVE Command	0x05	CMD	Byte0	Byte1	Byte2	Byte3	-	-	-
MOVE Command Status Reply	0x02	Status	CMD	-	-	-	-	-	-

Table 1.7: MOVE Command Options

Name	CMD	Size	Description
Move Absolute Immediate	0x00	int32	Immediate step move to new position without trajectory control.
Move Absolute	0x01	int32	Move to specified position using trapezoidal trajectory.
Move Relative Immediate	0x02	int32	Immediate move of a specified number of encoder counts from current position.
Move Relative	0x03	int32	Move a specified number of encoder counts from current position using trapezoidal trajectory.
Current Move	0x10	int16	Set drive current output to specified value.
Speed Move	0x20	int32	Set drive speed setpoint to specified value.
Abort Torque Off	0x80	-	Abort current move by turning off torque to motor.
Abort Decel. to Stop	0x81	-	Abort current move. Decelerate to zero using current trajectory parameters. Hold position when stopped.

4 Commonly Used Parameters and Variables

The master table in section 7 includes address for all read only and read/write parameters available to the user. This section will highlight a handful of those parameters that are regarded as most useful to the user.

4.1 CAN Axis-ID

Address (Hex)	0x4A7	Access	RW	Type	uint32	Units	value
Description	Preset address of the drive on the CAN network. Value is in non-volatile EEPROM and can be saved (persists through power cycle or reset).						

4.2 CAN Axis-ID Host Assigned

Address (Hex)	0x2049	Access	RW	Type	uint32	Units	value
Description	CAN address assigned by the host. This value is in volatile RAM. CAN Axis-ID (section 4.1) is written into it after each power cycle or reset to serve as a preset value. This memory location is meant to be used for the dynamic addressing scheme, see section 2.2 for more detail.						

4.3 CAN Bit Rate

Address (Hex)	0x4A9	Access	RW	Type	uint32	Units	bit defined
Description	All nodes of the network must use the same bit rate to communicate properly. Select the bit rate used on the CAN network by adjusting Bit0 & Bit1 to the following: 125 Kbps: Bit1 & Bit0 = 0 (0b00) 250 Kbps: only Bit0 = 1 (0b01) 500 Kbps: only Bit1 = 1 (0b10) 1 Mbps: Bit1 & Bit0 = 1 (0b11) Bit2 to Bit31: Reserved						

4.4 CAN Group-ID

Address (Hex)	0x4A8	Access	RW	Type	uint32	Units	value
Description	Address of a group of drives on the CAN network. The host can send commands to multiple drives at the same time by using the Group-ID address.						

4.5 CAN Options

Address (Hex)	0x4AA	Access	RW	Type	uint32	Units	bit defined
Description	CAN Options can be adjusted by using SET command for following bits on Byte0 & Byte1: Bit0 & Bit1: Both = 0 to use CAN Axis-ID as address Only Bit0 = 1 to use HW switches as address Only Bit1 = 1 to use CAN Axis-ID as base address and use HW Switches as 4 LSB of address. Bit8 & Bit9: Both = 0 for CAN enable not used Only Bit8 = 1 to use CAN enable input as CAN enable Only Bit9 = 1 to use CAN enable Input as drive enable Both = 1 to use CAN enable input as CAN and drive enable Bit12 to Bit15: Bit 12: Current limit over CAN, 1 = enabled, 0 = disabled Bit 13: Speed limit over CAN, 1 = enabled, 0 = disabled Bit 14: Speed feedback over CAN, 1 = enabled, 0 = disabled Bit 15: Position feedback over CAN, 1 = enabled, 0 = disabled Bit16 to Bit31: Reserved						

4.6 Actual Current

Address (Hex)	0x2004	Access	R0	Type	int32	Units	amps
Description	Current drawn by the motor in Amperes. Applying the GET command will provide data from Byte0-Byte3. See Section 5 for scaling details.						

4.7 Reference Current

Address (Hex)	0x2005	Access	RO	Type	Int16	Units	amps
Description	Commanded current of the motor in Amperes. Applying the GET command will provide data from Byte0-Byte3. See Section 5 for scaling details.						

4.8 Current Error

Address (Hex)	0x2016	Access	RO	Type	int32	Units	amps
Description	Allowable window of error (+/-) around the current command. Current error = commanded current - actual current						

4.9 Digital Input Status

Address (Hex)	0x2011	Access	RO	Type	uint32	Units	bit defined
Description	Bit0: Enable Bit1: Brake Bit2: Limit + Bit3: Limit – Bit4: Step Bit5: Direction Bit6: Capture Bit7: Hall 1 Bit8: Hall 2 Bit9: Hall 3 Bit10: Encoder A Bit11: Encoder B Bit12: Encoder Z Bit13: CAN Enable Bit14: Reserved Bit15: Reserved Bit16: Addr0 Bit17: Addr1 Bit18: Addr2 Bit19: Addr3 Bit20 to Bit31: Reserved See product HW manual for functional details of digital inputs.							

4.10 Digital Output Reset

Address (Hex)	0x2041	Access	RW	Type	uint32	Units	bit defined
Description	Setting BitN = 1 will disable the output, resetting it to the OFF state. Bit0 to Bit27: Reserved Bit28: CAN enable – CAN enable output. Bit29: eBrake Bit30: Ready Bit31: Fault *Note: eBrake, Ready & Fault will be overwritten by drive firmware when their states change.						

4.11 Digital Output Set

Address (Hex)	0x2040	Access	RW	Type	uint32	Units	bit defined
Description	Setting BitN = 1 will enable the output, setting it to the ON state. Bit0 to Bit27: Reserved Bit28: CAN enable – CAN enable output. Bit29: eBrake Bit30: Ready Bit31: Fault *Note: eBrake, Ready & Fault will be overwritten by drive firmware when their states change.						

4.12 Digital Output Status

Address (Hex)	0x2012	Access	RO	Type	uint32	Units	bit defined
Description	Bit0 to Bit27: Reserved Bit28: CAN enable – CAN enable output. Bit29: eBrake Bit30: Ready Bit31: Fault See product HW manual for functional details of digital outputs.						

4.13 Encoder Index Position

Address (Hex)	0x2033	Access	RO	Type	Int32	Units	encoder counts
Description	Position captured on rising edge of Encoder Index signal. In units of Actual Position when using encoder feedback (affected by Set Position command) or raw encoder counts when using Analog or CAN feedback. See Encoder Position below for raw encoder position.						

4.14 Encoder Position

Address (Hex)	0x2048	Access	RO	Type	Int32	Units	encoder counts
Description	Encoder Position. Position of the actual encoder from power-up or Reset. Unaffected by Set Position command.						

4.15 Encoder Capture Position

Address (Hex)	0x2036	Access	RO	Type	Int32	Units	encoder counts
Description	When a capture digital input is triggered on a falling edge, it will take a snapshot of the encoder position at that time.						

4.16 Actual Position

Address (Hex)	0x2000	Access	RO	Type	int32	Units	encoder counts
Description	Actual position of motor in encoder counts. Applying the GET command will provide data from Byte0-Byte3.						

4.17 Reference Position

Address (Hex)	0x2001	Access	RO	Type	int32	Units	encoder counts
Description	Commanded position of motor in encoder counts. Applying the GET command will provide data from Byte0-Byte3.						

4.18 Position Error

Address (Hex)	0x2018	Access	RO	Type	int32	Units	encoder counts
Description	Allowable window of error (+/-) around the position command. Position error = commanded position - actual position						

4.19 Actual Speed

Address (Hex)	0x2002	Access	RO	Type	int32	Units	RPM
Description	Actual speed of motor in RPM. Applying the GET command will provide data from Byte0-Byte3. See Section 5 for scaling details.						

4.20 Reference Speed

Address (Hex)	0x2003	Access	RO	Type	int32	Units	RPM
Description	Commanded speed of the motor in RPM. Applying the GET command will provide data from Byte0-Byte3. See Section 5 for scaling details.						

4.21 Speed Error

Address (Hex)	0x2017	Access	RO	Type	int32	Units	RPM
Description	Allowable window of error (+/-) around the speed command. Speed error = commanded speed - actual speed						

4.22 Move Status

Address (Hex)	200D	Access	RO	Type	uint32	Units	bit defined
Description	Status of the trajectory generator. Bit0 to Bit2 describes trajectory generator position move phase Phase 0 (0b000) = pre-move phase Phase 2 (0b011) = constant acceleration Phase 4 (0b100) = constant speed Phase 6 (0b110) = constant deceleration Bit3 describes trajectory generator completion status, and Bit3 = 1 when trajectory generator is complete. Bit4 describes trajectory generator process status and Bit4 = 1 when trajectory generator is running. Bit5 = 0 by default and Bit5 = 1 when abort was initiated. Bit6 = 0 when abort occurs without deceleration and Bit6 = 1 when abort occurs with deceleration. Bit7 to Bit31: Reserved.						

4.23 Drive Status

Address (Hex)	0x200E	Access	RO	Type	uint32	Units	bit defined
Description	Bit0: Drive ready state (Bit0 = 1 when drive is ready) Bit1: Drive enable state (Bit1 = 1 when drive is enabled) Bit2: Current loop saturation (Bit2 = 1 when current loop is saturated) Bit3: Speed loop saturation (Bit3 = 1 when speed loop is saturated) Bit4: Position loop saturation (Bit4 = 1 when position loop is saturated) Bit5: Diagnostics script status (Bit5 = 1 when running script) Bit6: Dynamic brake status (Bit6 = 1 when dynamic brake is enabled) Bit7: Reserved Bit8: Fault Bit9 to Bit31: Reserved						

4.24 Fault Status

Address (Hex)	0x203E	Access	RO	Type	uint32	Units	bit defined
Description	BitN = 1 indicates fault was detected. Bit0: Hardware Bit1: Software Bit2: Configuration Bit3: Phase over-current Bit4: Shunt over-power Bit5: Over-voltage Bit6: Under-voltage Bit7: Drive over-temperature Bit8: Hall Sensor Bit9: Encoder Bit10: Encoder Index Bit11: Motor over-temperature Bit12: Over-speed Bit13: Control error Bit14 to Bit31: Reserved						

4.25 Drive Temperature

Address (Hex)	0x203C	Access	RO	Type	int16	Units	degrees C
Description	Internally measured drive PCB temperature.						

4.26 Actual Voltage

Address (Hex)	0x2006	Access	RO	Type	int32	Units	volts
Description	Actual voltage applied to drive motor, used in control loop(s).						

4.27 Bus Voltage

Address (Hex)	0x2013	Access	RO	Type	int32	Units	volts
Description	Externally applied B+ bus as monitored by the drive.						

5 Scaling

5.1 Voltage Scaling

Voltage (Volts) = IU * (100V / 100V_ADC_Counts)

IU: Internal units
100V_ADC_Counts: Set at factory
*Found at memory location 0x0025 – see section 6.1 for additional detail.
Example:
100V_ADC_Counts = 3895
Hex value for Voltage = 0x3A8
Decimal value (IU) for Voltage = 936
Actual Voltage = 24V

5.2 Current Scaling

Current (Amps) = IU * (Rated Current Max / (2048))

IU: Internal units
Example:
Rated Current Max = 40A
Hex value for Actual Current = 0xF
Decimal value (IU) for Actual Current = 15
Actual Current = 0.29A

5.3 Speed Scaling

Speed (RPM) = IU * (1RPM / 2048)

IU: Internal units
Example:
Hex value for Actual Speed = 0xBB8000
Decimal value (IU) for Actual Speed = 12288000
Actual Speed = 6000 RPM

5.4 Temperature Scaling for Motor Feedback

Temperature values for motor feedback will vary depending on what is being used (NTC, PTC, threshold, etc.). More detail can be found in the product HW Manual(s).

6 Parameter Tables

6.1 Factory Parameters

Factory parameters are stored in non-volatile memory and are read only. These values can be read over CAN by using the GET command for the addresses listed below. They also can be viewed in CompleteArchitect™ on the drive info tab.

Address	Name	Type	Units
0x0013	Firmware Revision	uint32	decimal
0x0014	Max Bus Voltage	uint16	volts
0x0015	Min Bus Voltage	uint16	volts
0x0016	Model Number 1	char[4]	ASCII
0x0017	Model Number 2	char[4]	ASCII
0x0018	Model Number 3	char[4]	ASCII
0x0019	Model Number 4	char[4]	ASCII
0x001A	Model Number 5	char[4]	ASCII
0x001B	Model Number 6	char[4]	ASCII
0x001C	Rated Current Continuous	uint16	amps
0x001D	Rated Current Max (mA)	uint32	milliamps
0x001E	Rated Current Peak	uint16	amps
0x0020	Rated Temperature	uint16	degrees C
0x0021	Serial Number 1	char[4]	ASCII
0x0022	Serial Number 2	char[4]	ASCII
0x0023	Serial Number 3	char[4]	ASCII
0x0024	Serial Number 4	char[4]	ASCII
0x0025	Vbus 100V_ADC_Counts	uint32	value

6.2 User Parameters

User parameters are stored in non-volatile memory. All user parameters are read/write, however some are limited to being read only while the drive is enabled. See the table below for more information.

These values can be read using the GET command. Using the SET command to modify these will change the value in volatile RAM and can be utilized in that power cycle. To update and save the values for the next power cycle or reset, the EEPROM save command is required.

These can also be viewed and adjusted in CompleteArchitect™.

Address	Name	Type	Access	Description
0x0400	Drive Commutation Waveform Selection	uint16	RW (RO if enabled)	bit defined
0x0401	Motor Pole Pairs	uint16	RW (RO if enabled)	value
0x0402	Drive Runtime Options	uint16	RW (RO if enabled)	bit defined
0x0403	Drive PWM Frequency KHz	uint16	RW (RO if enabled)	*CompleteArchitect™
0x0404	Drive Mode	uint16	RW (RO if enabled)	bit defined
0x0405	Motor Type	uint16	RW (RO if enabled)	*CompleteArchitect™
0x0406	Motor Stepper Microstep Shifter	uint16	RW (RO if enabled)	value
0x0407	Motor Hall Options	uint16	RW (RO if enabled)	bit defined
0x0408	Drive Control Interface	uint16	RW (RO if enabled)	value
0x0409	Drive IO Options	uint16	RW	bit defined
0x040A	Encoder Enabled	uint16	RW (RO if enabled)	bit defined
0x040C	Encoder Options	uint16	RW (RO if enabled)	bit defined
0x040E	Encoder Lines	uint32	RW (RO if enabled)	value
0x0410	Motor Stepper Steps Per Rev	uint16	RW (RO if enabled)	value
0x0411	Motor Stepper Moving Current	uint16	RW	amps
0x0412	Motor Stepper Standby Current	uint16	RW	amps
0x0413	Motor Stepper Standby Timeout	uint16	RW	ms
0x0414	Motor Resistance	uint16	RW	ohms
0x0415	Motor Resistance Calibration	uint16	RW	ohms
0x0416	Motor Ke	uint16	RW	V/KRPM
0x0417	Motor Ke Calibration	uint16	RW	percent (%)
0x0418	Closed Loop Steps Per Rev	uint16	RW (RO if enabled)	value
0x0419	Closed Loop Stepper Start Time AB	uint16	RW	ms
0x041A	Closed Loop Stepper Start Current AB	uint16	RW	amps
0x041B	Current Loop Proportional Gain	uint32	RW	*CompleteArchitect™
0x041C	Current Loop Integral Gain	uint32	RW	*CompleteArchitect™
0x041D	Current Loop Integrator Saturation Limit High	int32	RW	*CompleteArchitect™

Address	Name	Type	Access	Description
0x041E	Current Loop Integrator Saturation Limit Low	int32	RW	*CompleteArchitect™
0x041F	Current Loop Output Saturation Limit High	int32	RW	*CompleteArchitect™
0x0420	Current Loop Output Saturation Limit Low	int32	RW	*CompleteArchitect™
0x0421	Current Loop Ramp Step	uint16	RW	*CompleteArchitect™
0x0422	Current Loop Ramp Loops Per Step	uint16	RW	*CompleteArchitect™
0x0423	Speed Loop Speed Feed Forward Gain Shift	uint8	RW	*CompleteArchitect™
0x0424	Speed Loop Proportional Gain Shift	uint8	RW	*CompleteArchitect™
0x0425	Speed Loop Integral Gain Shift	uint8	RW	*CompleteArchitect™
0x0426	Speed Loop Acceleration Feed Forward Gain Shift	uint8	RW	*CompleteArchitect™
0x0427	Speed Loop Speed Feed Forward Gain	uint32	RW	*CompleteArchitect™
0x0428	Speed Loop Proportional Gain	uint32	RW	*CompleteArchitect™
0x0429	Speed Loop Integral Gain	uint32	RW	*CompleteArchitect™
0x042A	Speed Loop Acceleration Feed Forward Gain	uint32	RW	*CompleteArchitect™
0x042B	Speed Loop Integrator Saturation Limit High (L)	int64	RW	*CompleteArchitect™
0x042C	Speed Loop Integrator Saturation Limit High (H)		RW	*CompleteArchitect™
0x042D	Speed Loop Integrator Saturation Limit Low (L)	int64	RW	*CompleteArchitect™
0x042E	Speed Loop Integrator Saturation Limit Low (H)		RW	*CompleteArchitect™
0x042F	Speed Loop Output Saturation Limit High	int32	RW	*CompleteArchitect™
0x0430	Speed Loop Output Saturation Limit Low	int32	RW	*CompleteArchitect™
0x0431	Speed Loop Rate	uint16	RW	*CompleteArchitect™
0x0432	Speed Loop Speed Ramp Step	uint16	RW	*CompleteArchitect™
0x0433	Speed Filter	uint32	RW	percent (%)
0x0437	Position Loop Proportional Gain Shift	uint8	RW	*CompleteArchitect™
0x0438	Position Loop Integral Gain Shift	uint8	RW	*CompleteArchitect™
0x0439	Position Loop Derivative Gain Shift	uint8	RW	*CompleteArchitect™
0x043A	Position Loop Proportional Gain	uint32	RW	*CompleteArchitect™
0x043B	Position Loop Integral Gain	uint32	RW	*CompleteArchitect™
0x043C	Position Loop Derivative Gain	uint32	RW	*CompleteArchitect™
0x043D	Position Loop Derivative Filter	uint32	RW	*CompleteArchitect™
0x043E	Position Loop Output Saturation Limit	int32	RW	*CompleteArchitect™
0x043F	Position Loop Integrator Saturation Limit (L)	int64	RW	*CompleteArchitect™
0x0440	Position Loop Integrator Saturation Limit (H)		RW	*CompleteArchitect™
0x0441	Position Loop Derivative Saturation Limit (L)	int64	RW	*CompleteArchitect™
0x0442	Position Loop Derivative Saturation Limit (H)		RW	*CompleteArchitect™
0x0443	Position with Speed Loop Proportional Gain Shift	uint8	RW	*CompleteArchitect™
0x0444	Position with Speed Loop Integral Gain Shift	uint8	RW	*CompleteArchitect™
0x0445	Position with Speed Loop Derivative Gain Shift	uint8	RW	*CompleteArchitect™
0x0446	Position with Speed Loop Proportional Gain	uint32	RW	*CompleteArchitect™
0x0447	Position with Speed Loop Integral Gain	uint32	RW	*CompleteArchitect™
0x0448	Position with Speed Loop Derivative Gain	uint32	RW	*CompleteArchitect™
0x0449	Position with Speed Loop Derivative Filter	uint32	RW	*CompleteArchitect™
0x044A	Position with Speed Loop Output Saturation Limit	int32	RW	*CompleteArchitect™

Address	Name	Type	Access	Description
0x044B	Position with Speed Loop Integrator Saturation Limit (L)	int64	RW	*CompleteArchitect™
0x044C	Position with Speed Loop Integrator Saturation Limit (H)		RW	*CompleteArchitect™
0x044D	Position with Speed Loop Derivative Saturation Limit (L)	int64	RW	*CompleteArchitect™
0x044E	Position with Speed Loop Derivative Saturation Limit (H)		RW	*CompleteArchitect™
0x044F	Analog Input Current Input Low	int16	RW	*CompleteArchitect™
0x0450	Analog Input Current Input High	int16	RW	*CompleteArchitect™
0x0451	Analog Input Current Output Low	int32	RW	*CompleteArchitect™
0x0452	Analog Input Current Output High	int32	RW	*CompleteArchitect™
0x0453	Analog Input Current Deadband Enable	uint16	RW	*CompleteArchitect™
0x0454	Analog Input Current Deadband Width	int16	RW	*CompleteArchitect™
0x0455	Analog Input Current Deadband Position	int32	RW	*CompleteArchitect™
0x0456	Analog Input Current Deadband Hysteresis	int16	RW	*CompleteArchitect™
0x0457	Analog Input Speed Input Low	int16	RW	*CompleteArchitect™
0x0458	Analog Input Speed Input High	int16	RW	*CompleteArchitect™
0x0459	Analog Input Speed Output Low	int32	RW	*CompleteArchitect™
0x045A	Analog Input Speed Output High	int32	RW	*CompleteArchitect™
0x045B	Analog Input Speed Deadband Enable	uint16	RW	*CompleteArchitect™
0x045C	Analog Input Speed Deadband Width	int16	RW	*CompleteArchitect™
0x045D	Analog Input Speed Deadband Position	int32	RW	*CompleteArchitect™
0x045E	Analog Input Speed Deadband Hysteresis	int16	RW	*CompleteArchitect™
0x045F	Analog Input Position Input Low	int16	RW	*CompleteArchitect™
0x0460	Analog Input Position Input High	int16	RW	*CompleteArchitect™
0x0461	Analog Input Position Output Low	int32	RW	*CompleteArchitect™
0x0462	Analog Input Position Output High	int32	RW	*CompleteArchitect™
0x0463	Analog Input Position Deadband Enable	uint16	RW	*CompleteArchitect™
0x0464	Analog Input Position Deadband Width	int16	RW	*CompleteArchitect™
0x0465	Analog Input Position Deadband Position	int32	RW	*CompleteArchitect™
0x0466	Analog Input Position Deadband Hysteresis	int16	RW	*CompleteArchitect™
0x0467	Analog Input Current Limit Input Low	int16	RW	*CompleteArchitect™
0x0468	Analog Input Current Limit Input High	int16	RW	*CompleteArchitect™
0x0469	Analog Input Current Limit Output Low	int32	RW	*CompleteArchitect™
0x046A	Analog Input Current Limit Output High	int32	RW	*CompleteArchitect™
0x046B	Analog Input Current Limit Deadband Enable	uint16	RW	*CompleteArchitect™
0x046C	Analog Input Current Limit Deadband Width	int16	RW	*CompleteArchitect™
0x046D	Analog Input Current Limit Deadband Position	int32	RW	*CompleteArchitect™
0x046E	Analog Input Current Limit Deadband Hysteresis	int16	RW	*CompleteArchitect™
0x046F	Analog Input Speed Limit Input Low	int16	RW	*CompleteArchitect™
0x0470	Analog Input Speed Limit Input High	int16	RW	*CompleteArchitect™
0x0471	Analog Input Speed Limit Output Low	int32	RW	*CompleteArchitect™
0x0472	Analog Input Speed Limit Output High	int32	RW	*CompleteArchitect™

Address	Name	Type	Access	Description
0x0473	Analog Input Speed Limit Deadband Enable	uint16	RW	*CompleteArchitect™
0x0474	Analog Input Speed Limit Deadband Width	int16	RW	*CompleteArchitect™
0x0475	Analog Input Speed Limit Deadband Position	int32	RW	*CompleteArchitect™
0x0476	Analog Input Speed Limit Deadband Hysteresis	int16	RW	*CompleteArchitect™
0x0477	Analog Input Speed Feedback Input Low	int16	RW	*CompleteArchitect™
0x0478	Analog Input Speed Feedback Input High	int16	RW	*CompleteArchitect™
0x0479	Analog Input Speed Feedback Output Low	int32	RW	*CompleteArchitect™
0x047A	Analog Input Speed Feedback Output High	int32	RW	*CompleteArchitect™
0x047B	Analog Input Speed Feedback Deadband Enable	uint16	RW	*CompleteArchitect™
0x047C	Analog Input Speed Feedback Deadband Width	int16	RW	*CompleteArchitect™
0x047D	Analog Input Speed Feedback Deadband Position	int32	RW	*CompleteArchitect™
0x047E	Analog Input Speed Feedback Deadband Hysteresis	int16	RW	*CompleteArchitect™
0x047F	Analog Input Position Feedback Input Low	int16	RW	*CompleteArchitect™
0x0480	Analog Input Position Feedback Input High	int16	RW	*CompleteArchitect™
0x0481	Analog Input Position Feedback Output Low	int32	RW	*CompleteArchitect™
0x0482	Analog Input Position Feedback Output High	int32	RW	*CompleteArchitect™
0x0483	Analog Input Position Feedback Deadband Enable	uint16	RW	*CompleteArchitect™
0x0484	Analog Input Position Feedback Deadband Width	int16	RW	*CompleteArchitect™
0x0485	Analog Input Position Feedback Deadband Position	int32	RW	*CompleteArchitect™
0x0486	Analog Input Position Feedback Deadband Hysteresis	int16	RW	*CompleteArchitect™
0x0487	Analog Output 1 Selection	uint8	RW	*CompleteArchitect™
0x0488	Analog Output 1 Shift	uint8	RW	*CompleteArchitect™
0x0489	Analog Output 1 Gain	int32	RW	*CompleteArchitect™
0x048A	Analog Output 1 Offset	int32	RW	*CompleteArchitect™
0x048B	Analog Output 2 Selection	uint8	RW	*CompleteArchitect™
0x048C	Analog Output 2 Shift	uint8	RW	*CompleteArchitect™
0x048D	Analog Output 2 Gain	int32	RW	*CompleteArchitect™
0x048E	Analog Output 2 Offset	int32	RW	*CompleteArchitect™
0x048F	Fault enable Mask	uint32	RW	bit defined
0x0490	Current Limit Peak	uint16	RW	amps
0x0491	Current Limit Continuous	uint16	RW	amps
0x0492	Current Limit Peak Time	uint16	RW	ms
0x0493	Shunt Enable	uint16	RW	bit defined
0x0494	Shunt On Voltage	uint16	RW	volts
0x0495	Shunt Off Voltage	uint16	RW	volts
0x0496	Protections Max Speed	int32	RW	RPM
0x0497	Protections Speed Control Error	int32	RW	RPM
0x0498	Protections Speed Control Error Time	uint16	RW	ms
0x0499	Protections Position Control Error	int32	RW	encoder counts
0x049A	Protections Position Control Error Time	uint16	RW	ms

Address	Name	Type	Access	Description
0x049B	Power On Enable Delay Time	uint16	RW	ms
0x049F	CAN Protocol Setting (J1939 or EC-CAN)	int16	RW	bit defined
0x04A1	Trajectory Generator Acceleration	uint32	RW	RPM/s
0x04A2	Trajectory Generator Speed	uint32	RW	RPM
0x04A7	CAN Axis ID	uint32	RW	value
0x04A8	CAN Group ID	uint32	RW	value
0x04A9	CAN Bit Rate	uint32	RW	bit defined
0x04AA	CAN Options	uint32	RW	bit defined
0x04C1	eBrake Enable	uint32	RW	bit defined
0x04C2	eBrake Enable On Delay	uint32	RW	ms
0x04C3	eBrake Enable Off Delay	uint32	RW	ms
0x04C8	Analog Input Setup	uint32	RW	bit defined
0x04C9	eBrake Pull-In Voltage	uint16	RW	%
0x04CA	eBrake Hold Voltage	uint16	RW	%
0x04CB	eBrake Pull-In Time	uint16	RW	ms
0x04CD	Motor Temperature Limit	int16	RW	value (see section 5)

6.3 System Variables

System variables are stored in volatile memory and most of them are read only. There are some values that can be written, please see the table below for details. Values can be read or written over CAN by using the GET & SET commands respectively.

These can also be viewed and adjusted in CompleteArchitect™.

Address	Name	Type	Access	Units
0x2000	Actual Position	int32	RO	encoder counts
0x2001	Reference Position (Commanded Value)	int32	RO	encoder counts
0x2002	Actual Speed	int32	RO	RPM
0x2003	Reference Speed (Commanded Value)	int32	RO	RPM
0x2004	Actual Current	int32	RO	amps
0x2005	Reference Current (Commanded Value)	int32	RO	amps
0x2006	Actual Voltage	int32	RO	volts
0x2008	Electrical Angle	int16	RO	value
0x2009	Hall State (Motor Phase State)	uint16	RO	bit defined
0x200A	Motor Phase (Hall Sector 1 through 6)	uint16	RO	value
0x200D	Move Status	uint32	RO	bit defined
0x200E	Drive Status	uint32	RO	bit defined
0x2011	Digital Input Status	uint32	RO	bit defined
0x2012	Digital Output Status	uint32	RO	bit defined
0x2013	Bus Voltage	int32	RO	volts
0x2015	PWM Percent	int16	RO	%
0x2016	Current Error	int32	RO	amps
0x2017	Speed Error	int32	RO	RPM
0x2018	Position Error	int32	RO	encoder counts
0x2033	Encoder Index Position	int32	RO	encoder counts
0x2035	Home Position - reserved	int32	RO	encoder counts
0x2036	Encoder Capture Position	int32	RO	encoder counts
0x203C	Drive Temperature	int16	RO	degrees C
0x203D	Motor Temperature	int16	RO	*See section 5
0x203E	Fault Status	uint32	RO	bit defined
0x2040	Digital Out Set	uint32	R/W	bit defined
0x2041	Digital Out Reset	uint32	R/W	bit defined
0x2042	User Speed Limit	int32	R/W	RPM
0x2043	User Current Limit	int32	R/W	amps
0x2044	User Speed Feedback	int32	R/W	RPM
0x2045	User Position Feedback	int32	R/W	encoder counts
0x2048	Encoder Position	int32	RO	encoder counts
0x2049	CAN Axis ID Host Assigned	uint32	R/W	value

7 Troubleshooting Tips

Below are a few basic trouble shooting tips the user can try if there are issues establishing a connection between the host and the drive(s).

- CAN to USB tools (such as Peak Systems P-CAN) can be used to check bus traffic.
- Confirm that CAN-H and CAN-L signals are connected to the appropriate pins/circuits.
- Confirm that CAN-H and CAN-L wires are in a twisted pair arrangement.
- Confirm other connections, such as GND or CAN enable (if configured) are connected properly.
- Confirm that the 120 Ω termination is present on each end of the network.
- Confirm baud rate setting in the drive matches the host baud rate.
- Confirm destination ID of the message matches the Axis-ID of the drive.