Binary serial interface

Introduction

The serial port on Copley digital servo drives supports two different protocols; a binary interface and an ASCII interface. The binary serial interface is used by the CME interface software, and allows all drive functions which are available through the serial port to be accessed. The ASCII interface allows a limited number of drive functions to be performed and is intended as a simplified method of controlling the drive for customer use.

The ASCII interface uses a text based command set which allows commands to be manually sent to the drive using a terminal emulating program. A user can easily enter ASCII commands and read the response that the drive sends back. For example, a typical ASCII command to read the value of a drive parameter would look like this:

```
g r0x17
v 1234567
```

In this example, the ASCII command to get parameter 0x17 from RAM was entered, and the drive responded with the value of that parameter which was 1234567.

The comparable binary command (as described below) would consist of a 4 byte header and a two byte parameter number. The response from the drive would start with a 4 byte header followed by a 4 byte parameter value.

The binary interface has several advantages over the ASCII interface:

- All serial commands are supported via the binary interface. The ASCII interface only supports a limited subset of drive commands.
- The binary interface includes a checksum byte on both commands to the drive and responses from the drive. This can be used to detect communications errors and prevent incorrect command execution.
- In general, the binary interface is more efficient then the ASCII interface.

For more information on the ASCII interface, please refer to the “ASCII Programmer's guide” available on the Copley web site.

Serial port settings

On power-up or reset, the serial port on Copley drives will always be configured for 9600 baud, no parity, 8 data bits and 1 stop bit.

The baud rate can be changed by setting parameter 0x90 to the desired baud rate. This 32-bit parameter gives the baud rate in units of bits/second, so for example the value 115200 would be passed to request that baud rate. The drive will set it's actual baud rate to the closest possible value to the requested rate. The exact baud rate used by the drive will depend on the frequency of the drive's main clock and may be slightly different from the requested rate. For example, an XEL drive which runs with a 100MHz internal clock will use an actual baud rate of 115207 when requested to communicate at 115200 bits/sec. This slight discrepancy should not cause any communication problems.

When setting the drive baud rate parameter the drive will change it's baud rate before sending the response to the set command, therefore the response will be sent at the new speed. In general, the simplest way to update the baud rate is to send the set command with the new baud rate value, then ignore any response from the drive which will be at a different rate. After a brief delay (10ms should be plenty), change the baud rate on the local serial port to the new value and resume communications with the drive.

If at any time a loss of synchronization occurs between the drive and the local computer, the drive's serial interface can be reset to it's default state by sending a break signal on the serial port. A break signal is a condition where the serial port line is lowered and held low for longer then a single character period. When the drive detects a break signal on the serial interface it immediately resets it's serial port settings to the default 9600 baud, flushes all input and output characters, and waits for a new command.

Command structure

All binary commands sent to the drive start with a 4 byte header. The command header consists of the following bytes sent in order:

Node number	This byte is only used when working in a multi-drop communications mode. When multi-drop serial mode is not is use, this byte should always be passed as zero. In multi-drop mode, multiple drives may be accessed using a single serial port. In this mode, each drive is assigned a node ID number. The drive that is physically connected to the serial line is always ID number zero. This drive will pass commands to other drives through a separate communications interface (such as the CANopen network) and will relay the remote drive's response back to the host through the serial port. The value passed in this byte should always be either zero (when not using multi-drop communication), or equal to the value 0x80 plus the node ID of the destination drive. Values in the range 0x01 to 0x7F should not be passed in the first byte since these will be interpreted as the start of an ASCII command sequence.

Checksum	This byte gives a checksum for the message which can be used to detect communication errors. The checksum is calculated by performing an exclusive OR operation on every byte of the command packet (including the header and checksum value). The result should equal the hex constant 5A.
Data size	This byte gives the number of 16-bit words of data to follow the header.
Opcode	This byte identifies the command being sent.

Following the header structure is a block of zero or more 16-bit words of command data. Only even numbers of bytes are legal for command data values, and the number of bytes sent must equal twice the value in the data size location in the header. Data values are always sent most significant byte first.

Response structure

Similar to the command, the response from the drive starts with a 4 byte header.

Reserved	This byte is reserved for future use and should be ignored.
Checksum	The checksum of the response. This checksum is calculated exactly like the checksum in the command message.
Data size	This byte gives the number of 16-bit words of data to follow the header.
Error code	This byte will be zero to indicate success, or will contain an error code if some error occurred.

Data returned by the amplifier will follow the response header. There will be exactly two times the ‘data size’ number of bytes of response data, and all values are sent most significant byte first.

Example command

The example ASCII command above which read parameter 0x17 from the drive would be sent via the binary protocol using the following byte values:

Command to drive:

```
0x00 0x40 0x01 0x0C 0x00 0x17
```

The first 4 characters are the header:

`0x00` – Node ID

`0x40` – Checksum

`0x01` – Number of words of data passed with command

`0x0C` – Command opcode (0x0C is the opcode used to read a parameter).

This is followed by a single word (2 bytes) of command data. For opcode 0x0C, the passed data word gives the drive parameter number being requested, sent MSB first. 0x0017 is the parameter number.

Response from drive:

```
0x00 0x1B 0x02 0x00 0x00 0x12 0xD6 0x87
```

The first 4 characters are the response header:

`0x00` – reserved character

`0x1B` – Checksum

`0x02` – Number of words of data returned

`0x00` – Error code (no error)

This is followed by 2 words (4 bytes) of data which give the value 0x0012D687 (1234567 decimal) sent most significant byte first.

Command codes

The following command codes are available for customer use.

No Operation

Op-code 0

Modes Available in normal and boot modes

Description No operation is performed. Useful for establishing communication.

Input data None

Output data None

Retrieve operating mode

Op-code 7

Modes Available in normal and boot modes

Description Returns the current amplifier operating mode. The operating mode determines if the amplifier is running normally or in boot mode.

Input data None.

Output data Operating mode. The modes are:

Mode	Description
0	Normal operating mode
1	Boot mode.

Get Flash CRC Value

Op-code	10
Modes	Available in boot mode and normal mode.
Description	The amplifier will calculate a CRC value for either the program Flash space (firmware) or the boot flash space (boot code). This 32-bit value will be returned. The CRC polynomial value used for all amplifier CRC calculations is 0xEDB88320.
Input data	Single word identifying which memory bank to calculate the CRC value for. 0 = Main firmware area 1 = Boot flash area 2 = Parameter flash area.
Output data	A 32-bit CRC value as calculated by the amplifier.

Swap operating modes

Op-code	11
Modes	Available in boot mode and normal mode.
Description	This command may be used to switch between normal operating mode and boot mode
Input data	None
Output data	None

Get variable value

Op-code 12

Modes Available in normal mode only.

Description Returns the current value of one of the amplifiers variables.

Input data One word of data is passed with this command. This word identifies the variable to be read. The variable identifier includes both the variable number, and the bank of memory to read the variable from. This word is encoded as follows:

Bits	Description
0-8	Variable identifier
9-11	Reserved for future use
12	Bank to read variable from
13-15	Axis number (0 for axis 1, 1 for axis 2, etc)

The variable ID identifies which variable to read. The “Parameter Dictionary” is a document available on the Copley web site which lists all drive parameters.

If the bank selection bit is zero, then the currently active value of the variable is read. If this bit is set then the variable’s value is read from flash storage. The value stored in flash is the value that will be assigned to the variable on reset. Not all variables have both RAM and flash versions.

Reserved bits should be written as zero.

Output data The requested variable’s value is returned. The size of this value is dependent on the variable being read. See the parameter dictionary for more information.

Set variable value

Op-code 13

Modes Available in normal mode only.

Description Allows the value of an amplifier variable to be updated.

Input data This command takes a variable identifier word followed by the new value for the variable. The identifier word has the following format:

Bits	Description
0-8	Variable identifier
9-11	Reserved for future use
12	Bank to write variable to
13-15	Axis number (0 for axis 1, 1 for axis 2, etc)

The variable ID identifies which variable to write. The parameter dictionary, available on Copley's web site, gives a list of all drive parameters.

If the bank selection bit is zero, then the currently active value of the variable is written. If this bit is set then the default value for the variable is written. This default value is stored in flash and is copied to the active variable's value after a reset. Not all variables have both an active and default value, refer to the parameter dictionary for more information.

The number of words of data following the identifier is variable dependent.

Output data None

Copy variable value

Op-code 14

Modes Available in normal mode only.

Description Allows a variables value to be copied from one bank to another.

Input data A variable identifier is supplied with this command. The identifier has the following format:

Bits	Description
0-8	Variable identifier
9-11	Reserved for future use
12	Bank to read variable FROM
13-15	Axis number (0 for axis 1, 1 for axis 2, etc)

The variable ID identifies which variable to copy. The parameter dictionary, available on Copley's web site, gives a list of all drive parameters.

If the bank selection bit is zero, then the currently active value of the variable is copied to the flash data area. If this bit is set then the value from flash is copied to the active variable area. Not all variables have both an active and default value, refer to the parameter dictionary for more information.

Output data None

Trace command

Op-code 15

Modes Available in normal mode only.

Description This command gives access to the amplifier’s variable trace feature. The trace command is split into a number of sub-functions which are all accessed through this op-code.

Details of the trace sub-commands are given later in this document.

Input data The first word gives the sub-function code. Additional words of data may be required depending on the sub-function selected.

Output data Depends on sub-command.

Reset

Op-code	16
Modes	Available in normal and boot modes
Description	Reset the amplifier immediately. Note: Because the amplifier is **immediately** reset, no response message is returned for this command.
Input data	None
Output data	None

Trajectory command

Op-code 17

Modes Available in normal mode

Description Send a command the the trajectory generator.

This command is used to start moves, abort moves, and start the homing state machine. It takes a single word of data formatted as follows:

Bits	Description
0-3	Trajectory sub-command
4-11	Reserved for future use
12	If set, apply the command to axis 1
13	If set, apply the command to axis 2
14	If set, apply the command to axis 3
15	If set, apply the command to axis 4

If none of the bits 12-15 is set, then the command is applied to axis 1 by default. Multiple axes can be selected by setting more then one bit in the 12-15 range.

The trajectory sub-command must take one of the following values:

Value	Command
0	Abort the move in progress. If a move is currently in progress, then it will be aborted (slowed down using the abort acceleration until the velocity is zero). If there is no move in progress, then the command will be ignored.
1	Start a new move, or update the parameters of the current move. If no move is in progress, then this command causes a new move to be started using the various move parameters (position, velocity, acceleration, etc). If there is currently a move in progress, then this command causes the trajectory parameters to be updated (if supported by the profile mode).
2	Start the homing state machine. This command starts the homing sequence defined by the various homing parameters (mode, velocity, acceleration, etc). It is an error to send this command when a move is currently in progress.

The default sub-command is 0 (abort move). If no data is passed to the command then the current move on axis 0 will be aborted.

Input data One word of data which specifies the trajectory sub-command.

Output data None

Error Log command

Op-code	18
Modes	Available in normal mode only.
Description	This command gives access to the amplifier's error log. The error log command is split into a number of sub-functions which are all accessed through this op-code. Additional details on error log sub-commands are provided later in this document.
Input data	The first word gives the sub-function code. Additional words of data may be required depending on the sub-function selected.
Output data	Depends on sub-command.

Copley Virtual Machine command

Op-code	20
Modes	Available in normal mode
Description	This command handles all aspects of communicating with the CVM (Copley Virtual Machine). It can also be used to manipulate the file system on the drive which holds CVM programs along with CAM tables and other data. A list of CVM commands is provided later in this document.
Input data	The first word gives the sub-function code. Additional words of data may be required depending on the sub-function selected.
Output data	Depends on sub-command.

Encoder command

Op-code 27

Modes Available in normal mode

Description This command may be used to perform special functions with some absolute encoders.

Input data One or more words. The first word gives a sub-command code and any additional words are treated as data used by the sub-command.

The first word of data passed with this command identifies which encoder the command is addressed to (motor encoder or load encoder), and gives a sub-command code. This first word is formatted as follows.

Bits	Contents
0	Clear when sending commands to motor encoder. Set to send commands to load encoder.
4-7	Sub-command code
12-13	Axis number

The following encoder sub-commands are currently supported:

Code	Description
0	Read a register from within the encoder. The register number is passed as a second word of data to the encoder command. One or more words of data will be returned.
1	Reset any errors currently latched on the encoder
2	Zero the encoders internal position.
3	reserved
4	Set a register in the encoder. The register number is passed as the second word, and the new value is passed as the third word.

Note that not all encoder types support all commands.

Output data Zero or more words of data depending on the sub-command passed

Get CAN object command

Op-code	28
Modes	Available in normal mode
Description	This command may be used on a CANopen or EtherCAT drive to read the value of an object in the CANopen object dictionary
Input data	One or two words. The first word gives the CANopen object ID of the object to read. The second word gives the sub-index. If the second word isn't passed, the sub-index defaults to zero.
Output data	The current value of the object is returned.

Set CAN object command

Op-code	29
Modes	Available in normal mode
Description	This command may be used on a CANopen or EtherCAT drive to set the value of an object in the CANopen object dictionary
Input data	The first word passed gives the object ID of the object to access. The second word gives the object sub-index. The remainder of the data input to the command is used to set the value of the identified object.
Output data	none

Dynamic file command interface.

Op-code 33

Modes Available in normal mode. FPGA based drives only

Description This command is used to up/download dynamically created files to/from the drive. The files that are downloaded over this interface are created on the fly by the drive firmware. Files uploaded to this interface are not stored in the flash file system, but rather processed on the fly by the firmware.

As of version 1.68 firmware, amp firmware files (*.cff) can be uploaded to the drive, and 'driveconfig' files can be up/downloaded. Driveconfig files are xml formatted files which contain all drive configuration data.

Input data The first word passed with this command gives a sub-command. The following sub-commands are supported:

code	Description
1	Start reading a file. The file name must be passed as additional data. Currently, only the name driveconfig is supported. Any other name passed will cause an error to be returned.
2	Start writing a file. No additional data is required, the drive determines the type of file being received from it's contents.
3	Write data to the file. The second word gives the number of bytes passed. The remaining data is the file contents to write. If the drive is not ready to take this many bytes of additional data it will return a timeout error (code 50). In that case, none of the passed data was accepted and the command can be tried again.
4	Read data from a file. No additional data is passed with this command. The first word of the response is the number of valid bytes of data returned, followed by the file data.
5	If an error occurs during a transfer, this command may be used to retrieve a string from the drive describing the nature of the error.
6	Finish the file transfer. This command should be sent after any file transfer whether successful or not.

Output data Depends on command. See above.

Trace sub-commands

The variable trace system allows one or more internal amplifier variable to be sampled and stored at a specified interval. The stored data may later be downloaded and used to facilitate loop tuning, etc.

The trace feature is controlled through the use of the trace command (op-code 15). A sub-command value is passed as the first data word with this command. The following sub-commands are defined.

Code	Description
0x00	Get channel count. The number of trace channels available in the amplifier is returned. This number will remain constant for a particular version of firmware, but could change with new firmware versions. It will always be at least 2.
0x01	Get trace status. A 16-bit trace system status value is returned. The format of this word is: **Bit** **Description** 0 Set if trace data is currently being collected. 1 Set if the trigger has occurred. 2-15 Reserved for future use
0x02	The trace is first stopped (if it was running) and then started from the beginning, data is collected to the trace buffer and the trigger condition is constantly checked.
0x03	Stop collecting data. Data collection is immediately stopped. Normally, data collection is stopped when the trigger condition and delay have been met. This function allows the trace collection to be stopped at any time.
0x04	Set sample period. This command is followed by one additional word of data that gives the period between trace samples. The sample period is set in units of fundamental trace periods (see below).
0x05	Get sample period. Returns the 16-bit sample period value.
0x06	Get available samples. Returns the number of samples collected so far (one 16-bit word).
0x07	Download samples. Two additional words of data must be passed, the (zero based) index of the first sample, and the number of samples requested. The requested samples will be returned limited to the number that will fit in the communications buffer. Each sample consists of N variables that are each 32-bits long where N is the number of active channels.
0x08	Get fundamental period. Returns a 32-bit value containing the fundamental trace period in units of nanoseconds. The fundamental period is the maximum frequency that the trace system can sample data. The actual trace period is set in integer multiples of this value using the Set Sample Period command (command 4).

<table>
<tr><td>0x09</td><td>Get maximum samples. The maximum number of samples that the internal trace memory buffer can hold is calculated and returned as a 16-bit value. Note that the maximum number of samples is dependent on the number and type of active trace variables. For an accurate value, the trace variables should be set first, then the maximum number of samples available may be requested.</td></tr>
<tr><td>0x10</td><td>Set trigger. Three additional words of data are supplied that identify the type of trigger used to start the trace. The first word has the following format:

Bits Description
0-3 Channel number to trigger on (if applicable).
4-7 Reserved for future use.
8-11 Trigger type
12-13 Axis number
14 Reserved
15 Take one sample per trigger event if set.

Normally, the two following data words specify a 32-bit trigger level (sent high word first). These data values may be interpreted differently for some trigger types.

The trigger types are as follows:

Type Description
0 No trigger in use.
1 Trigger as soon as the selected channel's input is greater then or equal to the trigger level.
2 Trigger as soon as the selected channel's input is less then or equal to the trigger level.
3 Trigger when the selected channel's input changes from below to above the trigger level.
4 Trigger when the selected channel's input changes from above to below the trigger level.
5 Trigger when any selected bits in the channel value are set. The bits are selected using the trigger level value as a mask.
6 Trigger when any selected bits in the channel value are clear. The bits are selected using the trigger level value as a mask.
7 Trigger any time the selected channel value changes.
8 The trigger level mask selects one or more bits in the event status word. The trigger occurs when any of these bits change from 0 to 1. In this mode, the channel number selected by the trigger is not used.
9 Like type 8, but the trigger occurs when the bit(s) change from 1 to 0.
10 Trigger on the start of the next function generator cycle. This trigger type is only useful when running in function generator mode. The trigger channel number isn't used.</td></tr>
<tr><td>0x11</td><td>Get trigger. The three words of trigger data are returned as a response to this</td></tr>
</table>

	command.
0x12	Set trigger delay. This command is followed by one additional word of data that gives the delay between the trigger occurring and the start of captured data. The delay is given in units of trace periods. Note that the delay may be either positive or negative. A negative delay means that the data captured will precede the trigger event by the specified number of cycles. Although any input value is accepted, the number of samples preceding the trigger is limited to the length of the trace buffer and the number (and size) of channels being captured.
0x13	Get trigger delay. Returns the 16-bit trigger delay value.
0x14	Reserve space in the trace buffer for other uses. This command takes a single word of data which gives the number of words of data to reserve. Reserved data will no longer be used by the trace system and can be used for certain other uses such as holding CAM tables.
0x15	Return the number of words of trace buffer data current reserved.
0x16	Write to reserved locations in the trace buffer. This command is used when storing a CAM table to an area of the trace buffer which was previously reserved using sub-command 0x14. The first word of data passed with this sub-command gives the word offset into the reserved area where the data should be written. Additional words passed with the command contain the data values to be stored to the trace buffer starting at that offset.
0x17	Read values from the reserved area of the trace buffer. Two additional words of data should be passed with this sub-command. The first gives the offset into the reserved area of the trace buffer from which the data should be read, the second gives the number of words of data to be returned.
0x10x	Set trace channel. The lower 4 bits of the sub-command code give the channel number (0 to max-1). This command takes one additional data word that gives the trace variable ID to associate with this channel. See the list below of valid trace Ids.
0x20x	Get trace channel. The lower 4 bits give the channel number (0 to max-1). The 16-bit value returned in response to this command contains the variable ID associated with the channel.

The following trace variables have been defined. Note that not all trace variables are available in all products and firmware versions.

ID	Description
0	No data. Setting a channel to this value disables it. Disabling unused channels saves

	space in the trace buffer.
3	Current reading winding A (0.01 amps)
4	Current reading winding B (0.01 amps)
5	Reference A/D reading (millivolts)
6	High voltage reference (0.1 volts)
7	Commanded torque
8	Limited torque
9	Commanded current (D rotor axis) (0.01 amps)
10	Commanded current (Q rotor axis) (0.01 amps)
11	Actual current (X stator axis) (0.01 amps)
12	Actual current (Y stator axis) (0.01 amps)
13	Actual current (D rotor axis) (0.01 amps)
14	Actual current (Q rotor axis) (0.01 amps)
15	Current Error (D rotor axis) (0.01 amps)
16	Current Error (Q rotor axis) (0.01 amps)
17	Current Integral (D rotor axis)
18	Current Integral (Q rotor axis)
19	Current loop output (D rotor axis)
20	Current loop output (Q rotor axis)
21	Current loop output (X stator axis)
22	Current loop output (Y stator axis)
23	Actual motor velocity (0.1 counts/sec or 0.01 RPM if using back EMF velocity estimate).
24	Commanded motor velocity.
25	Limited motor velocity command.
26	Velocity loop error.
27	Velocity loop integral.
28	Actual position for position loop (encoder counts).
29	Commanded position.
30	Position loop error

31	Motor encoder position (encoder counts)
32	Position loop output velocity
33	Raw input pin readings (no debounce)
34	Digital output state
35	Phase angle from motor encoder
36	Motor phase angle (1 degree units)
37	Amplifier temperature (degrees C)
38	Amplifier event status word
39	Amplifier event latch word
40	Hall sensor state
41	Position capture status register
42	Index capture register
43	Load encoder velocity (0.1 counts / second).
44	Velocity command from trajectory generator (0.1 cts/sec)
45	Acceleration command from trajectory generator (10 cts/sec^2)
46	The analog encoder sine input. Only valid for amplifiers with analog encoder support.
47	The analog encoder cosine input. Only valid for amplifiers with analog encoder support.
48	The value of the digital inputs (after debounce)
49	The destination position input to the trajectory generator.
50	Actual motor velocity as seen by velocity loop. This is an unfiltered version of trace variable 23.
51	Load encoder position
52	Gain scheduling key parameter value
53	Position loop P gain
54	Velocity loop P gain
55	Velocity loop I gain
56	Amplifier I2t running sum (0 to 10000)

57	User programmed I2t running sum (0 to 10000)
58	Second analog reference input for drives that have one (XEL for example)
59	Analog encoder index input for drives that support this (XEL,XPL,XML)
60	UV mode U input
61	UV mode V input
62	Current offset in EtherCAT CSP mode
63	Velocity offset in EtherCAT CSP mode
64	Sample a short (16-bit) value from CVM memory space.
65	Sample a long (32-bit) value from CVM memory space.

Error log commands

The amplifier error log provides access to a list of events that have occurred in the amplifier. The error log resides in non-volatile memory and therefore persists between amplifier power cycles.

Access to the error log is provided by the error log command. This command takes a sub-command code as it's first parameter. The sub-commands that are presently available are the following:

Code	Description
0x00	Get the total errors since manufacturing. There are up to 32 error codes that are presently logged by the amplifier. This command returns an array of 32 bytes where each byte gives the number (0 – 255) of errors of that type that have occurred since the amp was manufactured. The number of errors for error code 0 is given in the first byte, error code 1 in the second byte, etc. A list of valid error codes is provided below.
0x01	Get the total errors since error log clear. This function works the same as the one above, but the number of errors since the last log clear is returned.
0x02	Return the number of power/reset cycles since manufacturing.
0x03	Clear the error log.
0x04	Return the last N errors from the error log. The value N is passed as a data word with this command. If N is omitted, it defaults to 10. The amp returns the last N error events that have been logged (or all events since the last log clear if less then N). Each event consists of 2 32-bit values. The first 32-bit value for each event gives the event type in the upper 6 bits, and the time the event occurred in the lower 26 bits. The time is given as a number of seconds since reset. The 6-bit event type bit mapped as follows: **Bits** **Description** 0-3 Identify the type of event. The following types are currently defined 0 Amplifier was powered-up or reset 1 An amplifier system error occurred. 2 Error log was cleared 3 A latching amplifier fault occurred. 4-5 Define the axis number issuing the event.

	The second 32-bit value passed with each event gives data associated with the event. The interpretation of this data is dependent on the event type. For power-up events the data value isn't used (should be zero). For error codes it contains a bit-mask of all the new error conditions that occurred. For each error code (0 to 31) the corresponding bit will be set in the data value if that error occurred. For example, if errors 3 and 7 occurred the data value would contain the code 0x00000088. For amplifier faults the new fault conditions are reported. See amplifier variable 0xA4 in the parameter dictionary for details on the fault mapping.

The following system error codes are presently defined. System errors occur if the amplifier is functioning normally but incorrectly connected or driven.

ID	**Description**
0	Output short circuit. The amplifier detected a short-circuit condition on the motor outputs.
1	Amplifier over temperature. The amplifier temperature sensor detected an over-temperature condition.
2	Amplifier over voltage. The high-voltage supply voltage is over spec.
3	Amplifier under voltage. The high-voltage supply voltage is under spec.
4	Motor temperature sensor.
5	Encoder power error.
6	Amplifier phasing error.
7	Current limited.
8	Voltage limited.
9	Positive limit switch entered.
10	Negative limit switch entered.
11	Tracking error.
12	Encoder position wrapped.
14	CANopen node guarding error
15	Control input error.

CVM commands

Control of the CVM (Copley Virtual Machine) is handled through the standard amplifier serial interface. Amplifier command code 20 is used for all communication with the CVM and access to the CVM file system. The first word of data passed to command 20 is a sub-command which defines the operation to perform. The following sub-commands are defined:

Code	Description
0	Allocate space in the CVM file system for a new file. In: 0 Sub-command code. 1 File data length (in 16-bit words) 2 File name length (words) 3 File ID number (optional) Out: 0 The file ID number of the allocated file. Before a new file may be uploaded to the CVM file system, space must be allocated for it. This command attempts to allocate the requested number of 16-bit words from the file system. If the file ID number is not specified, then the first available file will be used. If the specified file ID is already in use, an error will be returned. On success, the file ID number will be returned.
1	Set file name. In: 0 Sub-command code. 1 File ID number 2 Word offset into name. 3... File name The file name passed starting at word 3 is stored in the file header starting at the offset specified in word 2. Normally word 2 should be 0.
2	Upload file data. In: 0 Sub-command code 1 File ID number. 2 Offset into file data 3... File data to upload The data passed starting at word 3 is uploaded into the specified file, starting at the specified offset.
3	Delete a file. In: 0 Sub-command code

Code	Description
	1 File ID number. The specified file is deleted.
4	Download file name. In: 0 Sub-command code 1 File ID number. 2 Word offset into name. Out: 0... The file name The file name for the specified file is downloaded, starting with the word at the specified offset from the beginning of the name. If the specified offset is greater then or equal to the length of the name, then zero words of data will be returned. Otherwise, the maximum amount of data possible will be returned.
5	Download file data. This command is exactly like sub-command 4, except that the file contents are returned rather then the name.
6	Get File system information Out: 0 Total number of available programs 1 Size of a data block 2 Total number of data blocks 3 Number of free data blocks. This command just returns some useful information about the file system. The total number of available programs is a constant (currently 32). The size of a data block is the size (in words) of a data block which is the unit in which data is allocated by the file system.
7	Select the startup program. This command allows one file to be designated as a startup program. The startup program will begin running each time the amplifier is powered up or reset. In: 0 Sub-command code 1 File ID number Specifying ID number -1 will prevent any file from running on startup.
8	Get the startup program number. The file ID of the startup program is returned, or -1 if none is selected. Out: 0 File ID number

Code	Description
9	Run a CVM program. The file specified by this command is executed. In: 0 Sub-command code 1 File ID number
10	Stop the currently running CVM program.
11	Load the specified CVM program into the active memory map, but don't start running it. This is mostly useful for debugging. In: 0 Sub-command code 1 File ID number
12	Single step the current CVM program. The next instruction of the currently loaded CVM program is executed and then the program execution is stopped.
13	Get the specified CVM register. In: 0 Sub-command code 1 Register ID number Out: 0 First word of register data 1 Second word of register data (for 32-bit registers only) The register ID numbers are 0-15 for the 32-bit registers (R0-R15), and 32-63 for the 16-bit registers.
14	Update a register value In: 0 Sub-command code 1 Register ID number 2 First word of register data 3 Second word of register data (32-bit only)
15	Read a CVM memory location In: 0 Sub-command code 1 Address 2 Number of words of data to return. Out: 0 Value at specified memory location
16	Write a value to a CVM memory location In: 0 Sub-command code 1 Address 2... New value(s) to write to memory starting at the passed address.
17	Read CVM status information

Code	Description
	Out: 0 CVM Status words The status word is bit-mapped as follows: 0 Set if a CVM program is currently running 1 Set if CVM is currently single stepping.
18	Get the active CVM program number. Out: 0 The program number of the currently loaded CVM program.
19	Continue execution. This command causes the currently loaded CVM program to continue running starting from the current PC location. It's most often used during debugging after hitting a breakpoint or after issuing a stop command.
20	Get Breakpoint address. In: 0 Breakpoint ID number (0 to 7) Out: 0 Address breakpoint is set to. Negative if disabled.
21	Set Breakpoint address In: 0 Breakpoint ID number (0 to 7) 1 Breakpoint address. Negative disables breakpoint.
22	Return a stack trace. The first word returned is the current PC value. The next is the return address for the calling function, etc...

Error codes

Value	Meaning
0	Success
1	Too much data was sent with command.
2	Checksum error on received command.
3	Illegal op-code.
4	Not enough data passed with command.
5	Unexpected data was passed with command.
6	Error erasing flash memory.
7	Error writing to flash memory.
8	Illegal memory page specified with parameter.
9	Unknown variable ID.
10	Parameter value out of range.
11	Illegal attempt to write to a read-only parameter.
12	Invalid trace channel specified in command.
13	Invalid trace variable number specified.
14	Invalid mode of operation specified.
15	Variable doesn't exist in specified memory page.
16	Unable to forward serial message, not CAN master
17	Data flash page CRC error.
18	Illegal attempt to start a move while currently moving.
19	Illegal velocity limit for move.
20	Illegal acceleration limit for move.

Value	Meaning
21	Illegal deceleration limit for move.
22	Illegal jerk limit for move.
23	Trajectory buffer underflowed during move.
24	Trajectory buffer overflowed when adding data.
25	Bad trajectory mode.
26	That CVM program location is not available.
27	The specified operation is not legal while CVM is running.
28	The CVM program is too big to upload.
29	File system internal error.
30	Specified program doesn't exist.
31	Invalid node ID for serial forwarding.
32	CAN network communications failure.
33	ASCII command parse error.
34	Internal error.
35	File system can't be changed while in camming mode.
36	Illegal axis number passed.
37	Invalid FPGA data stored on amp.
38	Unable to initialize the FPGA.
39	FPGA failed to configure.
40	File already exists.
41	No free file entries in directory.

Value	Meaning
42	File doesn't exist.
43	No free space in file system.
44	Invalid file format.
45	End of file hit while reading..
46	Error sending command to encoder.
47	Operation is illegal in current serial port mode.
48	Can't calculate filter.
49	Failed to perform protected CVM command because Indexer register 31 wasn't set.
50	Timeout.

Binary serial via CANopen / EtherCAT

It's sometimes useful to be able to send binary serial commands to the drive when communicating over the CANopen or EtherCAT network. This can be done through the use of a object `0x2000` in the drive's object dictionary.

Serial port commands can be sent using object `0x2000` by first writing the command to the object using an SDO. The data written to `0x2000` consists of the command code followed by any data words. When writing to this object, data words are sent in the same order that they would be sent via serial interface, but each word is sent least significant byte first which is the standard for CANopen.

For example; to read the current position of the motor (parameter `0x17`), the command code `0x0C` would be passed first, followed by the parameter number send LSB first. The following three bytes would be written to object `0x2000` via SDO access:

```
0x0C 0x17 0x00
```

To read back the result of the command, read object 0x2000. The data returned consists of a one byte error code followed by zero or more words of data. Like with the write, data is returned as an array of 16-bit values where each value is sent least significant byte first.

For example, if the motor position were 0x12345678, then reading the value of object 0x2000 via SDO would have returned the following five bytes:

```
0x00 0x34 0x12 0x78 0x56
```

Revision History

Date	Version	Revision
1/13/2014	1.0	Initial release
2/13/2014	1.1	Added set register sub-command to encoder command.
9/2/2014	1.2	Added table of error codes.
1/26/2015	1.3	Added a description of object 0x2000 which can be used to send binary serial commands via CANopen or EtherCAT networks.