# CMO Programmer's Guide


P/N 16-01041
Revision 01
June 2015

# TABLE OF CONTENTS

# ABOUT THIS MANUAL

## Related Documentation

- CANopen Programmer's Manual
- CAN Bus Cabling Guide
- CME 2 User Guide

Information about CANopen can be found on the CAN in Automation website at:

http://www.can-cia.de/index.php?id=canopen

Copley Controls software and related information can be found at:
http://www.copleycontrols.com/Motion/Products/Software/index.html

For more information on Microsoft® .NET please refer to: http://www.microsoft.com.

## Copyrights



## Document Validity

We reserve the right to modify our products. The information in this document is subject to change without notice and does not represent a commitment by Copley Controls. Copley Controls assumes no responsibility for any errors that may appear in this document.

## Product Warnings

Observe all relevant state, regional and local safety regulations when installing and using Copley Controls amplifiers. For safety and to assure compliance with documented system data, only Copley Controls should perform repairs to amplifiers.

**WARNING**

**Use caution in designing and programming machines that affect the safety of operators.**

The examples in this book are for demonstration purposes only, providing guidelines for programming. The programmer is responsible for creating program code that operates safely for the amplifiers and motors in any given machine.

**Failure to adhere to this warning can cause equipment damage, injury, or death.**

**WARNING**

**Do not use Copley Motion Objects to implement an Emergency Stop**

An Emergency Stop must be hardwired directly to the amplifier. Do not depend on the Copley Motion Objects software to provide for a timely emergency stop. Due to the non-deterministic nature of Microsoft Windows, the software cannot guarantee a timely emergency stop operation.

**Failure to adhere to this warning can cause equipment damage, injury, or death.**

## Revision History

| Revision | Date | Applies to | Comments |
|---|---|---|---|
| 00 | August 2014 | CMO Version 4.0 and 5.0 | Re-formatted text, added descriptions for new methods and properties. |
| 01 | June 2015 | CMO V5.1 Release | Added info for multi-axis CAN drives. Added table to the debug levels. Added description of new Linkage settings object. |

# CHAPTER 1: FUNDAMENTAL CONCEPTS AND PROCEDURES

# 1.1: Introduction

The Copley Motion Objects (CMO) simplifies the creation of Windows-based software for the control of Copley Controls amplifiers over a CANopen or EtherCAT network. CMO is an API that gives programmers access to an amplifier's CANopen/EtherCAT functions without having to learn the complexities of the underlying network protocol. CMO is a managed .NET assembly which means that it can be used with client code that supports .NET assemblies.

# 1.2: System Requirements

## Operating System and Hardware

- Operating Systems Supported: Windows 7 and XP SP2 or above.
- CMO currently supports the following CAN Interface cards:
  - Copley
  - Kvaser
  - IXXAT:
- Dedicated Ethernet adapter for the EtherCAT network.

# 1.3: Firmware Feature Sets

Copley amplifiers are grouped into different feature sets based on the processor architecture. In cases where a feature is implemented differently depending on the model, it will be mentioned in this document.

| Feature Set | Models |
|---|---|
| A | ASP, ASC, JSP |
| B | ACJ, ACK, ACM, ACP, R21, R22, R23, STL, STM, STP, XSL |
| C | ACJ-R, ACK-R, ACM-R, ADP, R10, R11, R20, R21-R, R22-R, R23-R, STX, XSJ, XTL |
| D | AEP, AMP |
| E | AEM, APM, AE2, AP2, BEL, BPL, BE2, BP2, SEM, SPM, SE2, SP2, SP4, TE2, TP2, TEL, XEL, XE2, XPL, XP2, XML |

Latest version of firmware is recommended and can be downloaded from Copley's website:
http://www.copleycontrols.com/Motion/Downloads/firmware.html

# 1.4: .NET Framework Compatibility

CMO is designed as a .NET Assembly, which means that it can be used in applications that are designed to run under the Microsoft .NET Framework. This includes applications built with Visual Studio. Occasionally, new versions of the .NET Framework released that are not backward compatible with earlier versions. When this occurs, Copley has to branch CMO and maintain multiple versions. This recently occurred when V4.0 of the .NET Framework was released. This version is not backward-compatible with any application that targeted version 2.0 through 3.5, and the result was to branch CMO to V4.x and V5.x.

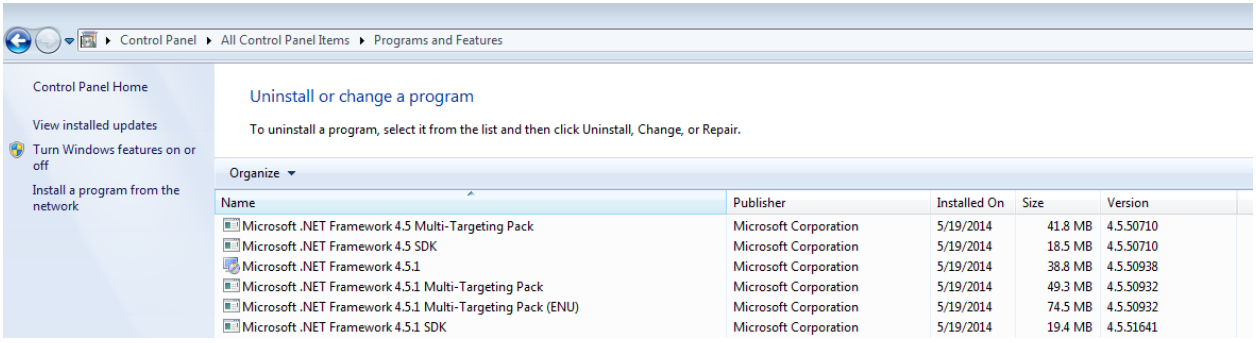
## CMO V4.x

The V4.x branch of CMO is compatible with .NET versions 2.0 through 3.5. The examples installed with V4.x were made with Visual Studio 2008 and target the .NET Framework 3.5. This CMO branch is not compatible with applications that target the .NET Framework 4.0 and 4.5.

## CMO V5.x

The V5.x branch of CMO is compatible with .NET versions 4.0 through 4.5. The examples installed with V5.x were made with Visual Studio 2010 and target the .NET Framework 4.0. This CMO branch is not compatible with applications that target the .NET framework 2.0 through 3.5.

To determine which versions of the .NET Framework are installed on your PC, go to the Control Panel and select Programs and Features (on Win XP, choose Add or Remove Programs). Scroll through the list of installed programs to the entries for Microsoft .NET Framework as shown below:




# 1.5: 32-bit vs. 64-bit Compatibility

Starting with V5.0, the installer allows the user to choose either the 32-bit or 64-bit version of CMO to be installed. This is done so that the user can target a different architecture when compiling their application with CMO. For instance, an application can be set up to target a 32-bit architecture, even though it is being compiled on a 64-bit machine. In this case, the user must install the 32-bit version of CMO so that it will work with their application on the 32-bit architecture. Please consult the owner's manual for your compiler for information on settings the target architecture.

**Important Note**

The application that uses CMO must target the same architecture as the version of CMO that is installed. The "any CPU" setting in Microsoft Visual Studio should never be used with CMO. Using this setting with either the 32-bit or 64-bit version of CMO will cause unpredictable behavior in the application (e.g. exceptions and breakpoints may not work).

# 1.6: CANopen Network

## Addressing and Bit Rate

Use CME 2 software to set up the amplifier's CAN node id and bit rate. CMO supports the following bit rates: 1Mb/s, 800kb/s, 500kb/s, 250kb/s, 125kb/s, 50kb/s, and 20kb/s.

CAN addresses (node id's) have a range of 1 to 127. Setting the node id to 0 disables the CAN operation for that amplifier.

## Multi-axis

For multi-axis amplifiers, each axis is treated as a separate node on the CAN network and requires its own AmpObj. Only one node id is configured for a multi-axis drive. That node id is assigned to axis A. The amplifier automatically configures the subsequent axes by increments of one. Therefore, if the amplifier was configured with a node id of 1 on a four axis drive, then the node ids for that amplifier will be:

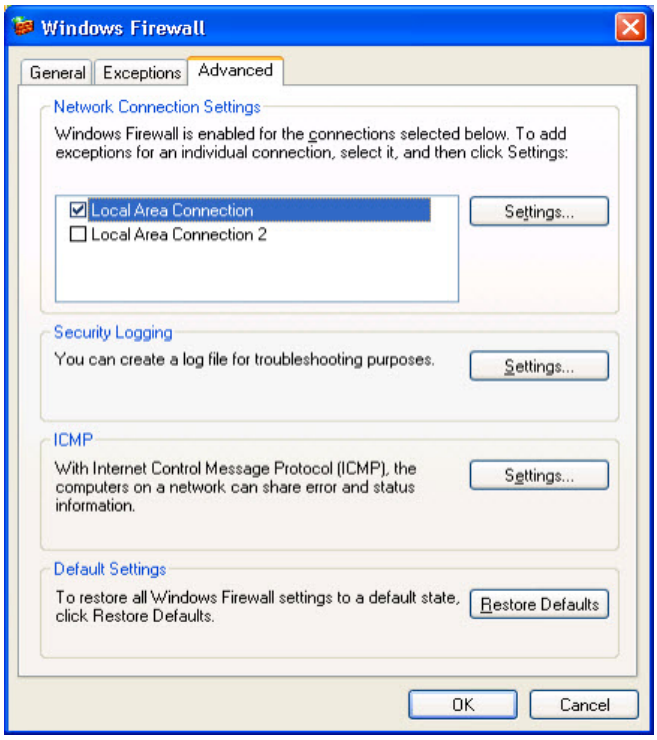
Axis A: 1
Axis B: 2
Axis C: 3
Axis D: 4

# 1.7: EtherCAT Network

With a standard Ethernet adapter on the PC, CMO can be used to communicate with Copley amplifiers that use UDP EtherCAT communications. This is achieved through the use of CAN application layer over EtherCAT (CoE), providing the same communication mechanisms as CANopen. The EtherCAT network should be dedicated only to EtherCAT communications and not be shared with the internet or a company's internal network. A second Ethernet adapter is needed for a single PC to maintain both an EtherCAT network and a standard network. If more than one Ethernet adapter is present, the firewall from the Ethernet adapter being used for the EtherCAT network must be disabled.

## Disable the firewall:

1. Click Start→Control Panel
2. When the Windows Control Panel is displayed, double click the Windows Firewall icon 
3. When the Windows Firewall dialog is displayed, click the Advanced tab and un-check the box to disable the firewall on the appropriate Ethernet adapter as shown below:




### Addressing

In an EtherCAT network, slaves are automatically assigned fixed addresses based on their position on the bus. In cases where the slaves must have an address that is independent of cabling, a Device ID is used. This can be set through switches (S1 & S2) on Plus Panels and through the switch and LED interface (SLI) on Plus Modules. The range of Device IDs is 0x01~0xFF.

### Multi-axis

Multi-axis EtherCAT amplifiers are treated as a single node on the network, so only one address is used per amplifier. This address is associated with axis A. The remaining axes are referred to as sub-axes. Each sub-axis requires a separate ampObj and is initialized using the InitializeEcatSubAxis() method (see EcatObj).

## 1.8: Communication Errors

### Access Denied

This error indicates that CMO could not find the network hardware (CAN card, or device drivers).

### SDO Timeout:

This error indicates that an SDO was sent, but no response was received. Possible causes are:

- The address is incorrect
- The bit rate is incorrect
- The wrong CAN channel is connected on a multiple-channel CAN card.
- The CAN bus is improperly terminated.
- CAN bus is wired improperly or disconnected.
- EtherCAT IP address is set incorrectly.
- The wrong EtherCAT name was selected.

- Firewall is enabled.

# 1.9: Node Guarding

## Overview

Node guarding is a CANopen device-monitoring feature. The network manager configures the amplifier to expect node-guarding messages at some interval. The network manager then sends a message to the amplifier at that frequency, and the amplifier responds with a node-guarding message. This allows both the network manager and the amplifier to identify a network failure if the guarding messages stop. CMO can turn node guarding on or off, and change the interval. If the amplifier detects that the guarding messages stop, it will abort a move in progress and set the AMPEVENT_NODEGUARD bit active in the AmpEvent status register. If node guarding is turned on, we recommend monitoring amplifier events for the node guard event. This can be done through the EventObj or through a timer, which periodically reads the event mask.

## Possibility of False Node Guarding Conditions

In a Windows environment, various factors can delay node-guarding messages, resulting in "false" node guarding conditions. These factors include the non-deterministic nature of Windows operating systems and the performance effects of other processes running on the PC. Thus, by default, node guarding is disabled in CMO. If node guarding is required, do not enable node guarding without first testing the performance characteristics and usage load of the PC being used, and adjusting the node guarding parameters accordingly using the ampSettingObj properties.

# 1.10: Exception Handling

If an error occurs, CMO reports the error by throwing an exception. Try/catch blocks should encapsulate all calls to CMO. For better error handling, each program should include error-handling procedures to prevent unexpected motion from occurring.

# 1.11: Units

## Default Amplifier Units

- **Position or Distance:** encoder counts
- **Velocity:** 0.1 encoder counts per second
- **Acceleration:** 10 encoder counts per second$^2$
- **Deceleration:** 10 encoder counts per second$^2$
- **Jerk:** 100 encoder counts per second$^3$

## User-Defined Units

The AmpObj property CountsPerUnit is a scaling factor for converting between an drive's default units and user-defined units.

### Example

To set user units to millimeters with a 5-micron encoder on a linear motor:

Set CountsPerUnit = 200, since there are 200 encoder counts in one millimeter.

# 1.12: Stepnet Amplifiers

## Stepper and Servo Modes

On power up/reset, Stepnet amplifiers start in stepper mode. If it is necessary to switch from stepper mode to servo mode, change the AmpModeWrite property of the AmpObj to one of the servo modes listed in CML_AMP_MODE. This should be done immediately after amplifier initialization.

In the following example, the amplifier is initialized and then the amplifier's mode of operation is switched to the servo Can profile mode:

```
ampObj.Initialize(canOpen, 1)

ampObj.AmpModeWrite = CML_AMP_MODE.AMPMODE_SERVO_CAN_PROFILE
```

## Open Loop Stepper Mode Actual Position and Velocity

When running open loop stepper mode, actual position and actual velocity readings remain at zero. The motor's commanded position can be monitored with AmpObj.PositionCommand (Units: microsteps).
The motor's commanded velocity can be monitored with AmpObj.TrajectoryVel (Units microsteps/second).

When the amplifier is disabled, PositionCommand goes to zero because the amplifier cannot tell if the motor moves while disabled. As long as the amplifier is enabled, relative and absolute moves can be made based on PositionCommand.

## Stepper Mode with Encoder Actual Position and Velocity

When running in stepper mode with an encoder, actual position can be monitored with AmpObj.PositionActual (Units: microsteps). Actual velocity can be monitored with AmpObj.VelocityLoad (Units microsteps/second).

NOTE: Actual velocity can also be monitored with AmpObj.VelocityActual, but the units will be in encoder counts/second. This is not recommended, because user units will also be applied to this value.

# CHAPTER 2: USING CMO IN AN APPLICATION

# 2.1: Building an Application

Regardless of the programming language or development environment, there are common steps to follow when building an application that uses CMO.

1. Determine the target CPU for the application to run on. This must be either x86 (32-bit) or x64 (64-bit). "Any CPU" cannot be chosen with CMO. See 32-bit vs. 64-bit Compatibility.
2. Determine the target .NET Framework for the application to run on. See .NET Framework Compatibility.
3. Install the version of CMO to match the target CPU in step 1. See Download and Install CMO.
4. Create the project for the application and set the target CPU and .NET Framework.
5. Add a reference to CMO in the project. See Adding a Reference to CMO in Visual Studio.
6. Declare a variable for the network object.
7. Declare one or more variables for the node objects (AmpObj or IOObj) and create instances of those variables.
8. Declare and instantiate settings objects for each node object declared in step 7 (AmpSettingsObj or ioSettingsObj).
9. Set the enableOnInit property of each settings object to False.
10. In the method or procedure that is called when the application, initialize the network and node objects. See Object Initialization Sequence.
11. Enclose all code that accesses CMO methods or properties with exception handling code.

## 2.2: Before Running a CMO Program

The following general steps must be completed before running any CMO program, including the demonstration programs described in this manual:

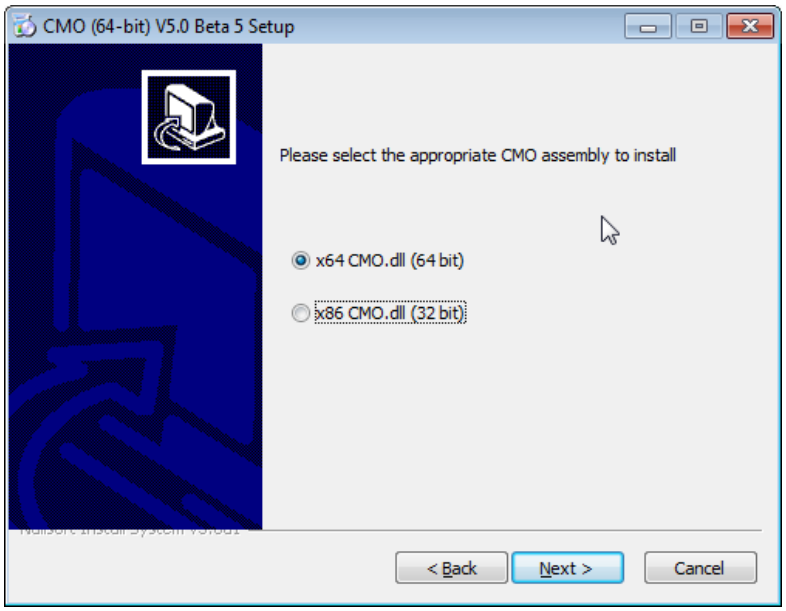
1. Review Product Warnings at the beginning of this manual.
2. Set up and tune the motor and amplifier using Copley Controls CME 2 software. If using a CANopen network, be sure to set the CAN node ID and bit rate.
3. Install CMO.
4. If using a CANopen network, install the CAN interface card and drivers. If using and EtherCAT network, make sure that a dedicated Ethernet adapter and network is being used.
5. Connect the amplifier, motor, and network.
6. Read through the steps in Building an Application to make sure that the application is set up properly.

## 2.3: Download and Install CMO

1. Navigate to:

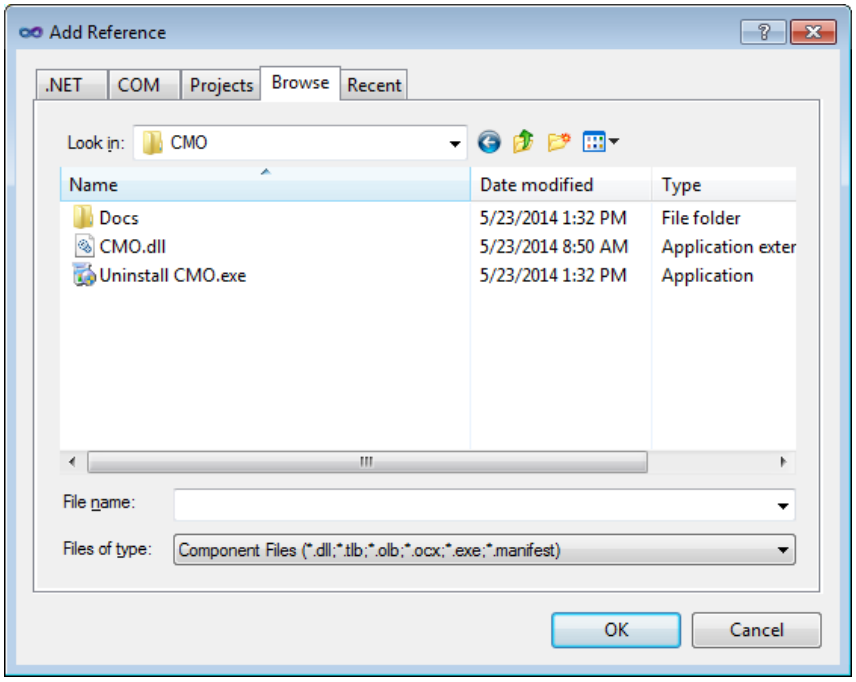
   http://www.copleycontrols.com/Motion/Downloads/index.html

   Select CMO to start the download.
2. Navigate to the folder where CMO was downloaded to and extract the contents of CMO.zip.
3. Run Setup.exe and follow the instructions on the installer screens. When prompted, choose the version of CMO that your application is targeting (32-bit or 64-bit). It is recommended to install CMO in the default location.


CMO (64-bit) V5.0 Beta 5 Setup
Please select the appropriate CMO assembly to install
x64 CMO.dll (64 bit)
x86 CMO.dll (32 bit)
< Back
Next >
Cancel

# 2.4: Adding a Reference to CMO in Visual Studio

1. From the Project menu, choose the Add Reference menu item, then select the Browse tab.




2. Browse to the folder where the CMO folder is installed.

3. Select CMO.dll and click ok.

# 2.5: Object Initialization Sequence

Every CMO application requires the creation and initialization of a network object, and node objects for each node on the network. These objects should always be initialized in the following order:

1. Network objects (CANOpenObj or EcatObj).
2. Node objects (AmpObj or IOObj).

Initializing the network establishes a connection to the network hardware (but not out on the network). If the call to the network object's initialize() method is successful, then CMO was able to find the network drivers and hardware. Before initializing, the network properties should be set if they are different than the defaults. See the properties of Network Objects for details. Initializing the nodes establishes communication to that particular node on the network. If the call to the node's initialize method is successful, then CMO was able to communicate with the node.

## CANopen Initialization

```
'Set the bit rate to 1 Mbit per second
canOpen.BitRate = CML_BIT_RATES.BITRATE_1_Mbit_per_sec
'Indicate that channel 0 of a Copley CAN card should be used
canOpen.PortName = "copley0"
canOpen.Initialize()
ampSettings.enableOnInit = False
'Initialize the AmpObj with the settings object
ampObj.InitializeExt(canOpen, 1, ampSettings)
```

## EtherCAT Initialization

```
'Indicate that the first Ethernet adapter is to be used
ecatObj.PortName = "eth0"
ecatObj.Initialize()
ampSettings.enableOnInit = False
'Initialize the AmpObj with the settings object
ampX.InitializeEcatExt(ecatObj, -1, ampSettings)
'Initialize the second axis by passing in the previously inititalized ampX object
ampY.InitializeEcatSubAxis(ampX)
```

## Initialization Errors

If the call to the network's initialize method fails, then CMO could not find and initialize the hardware. This is typically caused by one of the following:

- Network hardware not present
- CAN card drivers not installed
- Incorrect portName specified
- Incorrect channel specified (CANopen only)

If the call to the node's initialize method fails, then CMO could not communicate with the node. Typical causes are:

- Incorrect bit rate (CANopen only)
- No termination on the bus (CANopen only)
- Network settings of the program do not match the node (bit rate, node id, etc.).
- Node is not connected to the network
- Node is not powered up
- Node has a fault or is not enabled and the ampSettingObj was not used to turn off enableOnInit

# CHAPTER 3: NETWORK OBJECTS

# 3.1: canOpenObj

## Methods

### Initialize ()

Description:

Initializes the CANopen network.

Parameters:

None

### ClearErrorFrameCounter ()

Description:

Clears the CAN error frame counter.

Parameters:

None

## Properties

### ErrorFrameCounter

| | |
|---|---|
| Type: | Integer |
| Description: | Read-only. The number of error frames received over then CAN network since the last time the counter was cleared |
| Units: | None |
| Default: | None |

### BitRate

| | |
|---|---|
| Type: | CML_BIT_RATES |
| Description: | CANopen Bit Rate. |
| Units: | None |
| Default: | 1 Mb/s |

### CML_BIT_RATES

BITRATE_1_Mbit_per_sec = 1000000

BITRATE_800_Kbit_per_sec = 800000

BITRATE_500_Kbit_per_sec = 500000

BITRATE_250_Kbit_per_sec = 250000

BITRATE_125_Kbit_per_sec = 125000

BITRATE_50_Kbit_per_sec = 50000

BITRATE_20_Kbit_per_sec = 20000

### PortName

Type: String

Description: Port name for the network hardware. For CANopen, the port name is a combination of the CAN card name and the channel number as shown in the table.

| CAN Card | Port Name |
|---|---|
| Copley | copley0, copley1 |
| Kvaser | kvaser0, kvaser1 |
| IXXAT (V2.0 drivers) | ixxat0, ixxat1 |
| IXXAT (V3.0 drivers) | ixxatV30, ixxatv31 |

Units: None

Default: The port name defaults to channel 0 of the first supported CAN card found. CMO will search for the CAN cards in the order that they are listed in the table.

# 3.2: EcatObj

## Methods

### Initialize ()

Description:

Initializes the EtherCAT network.

Parameters:

None

## Properties

### PortName

Type: String

Description: The name "eth" is used in conjunction with the adapter number to specify the port name. For example, "eth0" is used for the first Ethernet adapter, "eth1" for the second adapter. Alternatively, the port name can be set to the IP address of the Ethernet adapter.

Units: None

Default: "eth0"

# CHAPTER 4: AMPLIFIER AND RELATED OBJECTS

# 4.1: ampSettingsObj

## Overview

The Amp Settings Object contains information about the amplifier's network settings. All of the properties have both read and write access. This object is passed in as a parameter in the InitializeExt method of the Amplifier Object to customize the network settings.

## Example:

1. Declare and create an instance of ampSettingsObj.

```
Dim ampSettings As ampSettingsObj
ampSettings = New ampSettingsObj()
```

2. Change one or more properties of the ampSettingsObj.

```
ampSettings.enableOnInit = False
```

3. Call one of the Extended Initialization methods of the ampObj.

```
ampObj.InitializeExt(canOpen, CAN_ADDRESS, ampSettings)
```

## Properties

### guardTime

Type: Short

Description: Node guarding guard time. This property gives the node-guarding period for use with this node. This is the period between node guarding request messages sent by the master controller.

Units: mS

Default: 0

### heartbeatPeriod

Type: Short

Description: Configures the heartbeat period used by this amplifier to transmit its heartbeat message. If this property is set to zero, then the heartbeat protocol is disabled on this node.

Units: mS

Default: 0

### heartbeatTimeout

Type: Short

Description: Additional time to wait before generating a heartbeat error.

Units: mS

Default: 0

### lifeFactor

Type: Short

Description: Node guarding lifetime factor. The lifetime factor is treated as a multiple of the guard time. If this property and the node guard time are both non-zero, and the heartbeatTimeout is zero, then node guarding will be setup for the amplifier.

Units: mS

Default: 3

### resetOnInit

Type: Boolean

Description: If *True*, the amplifier will be reset when it is initialized. This has the advantage of clearing out any fault conditions and putting the amplifier in a known state.

Units: None

Default: False

### enableOnInit

Type: Boolean

Description: Enable amplifier at initialization. If true, then the amplifier will be enabled at the end of a successful initialization. If false, the amplifier will be disabled at the end of a successful initialization

Units: None

Default: True

### synchID

Type: Integer

Description: Synch object CAN message ID. This is the message ID used for the synch message.

Units: None

Default: 128 (0x00000080)

### synchPeriod

Type: Integer

Description: Synch object period. The synch object is a message that is transmitted by one node on a CANopen network at a fixed interval. This message is used to synchronize the devices on the network.

Units: microseconds

Default: 10000

### synchProducer

Type: Boolean

Description: If true, this node will produce synch messages. If 'synchUseFirstAmp' property is set to true, this property will not be used and will be overwritten during initialization.

Units: None

Default: False

### synchUseFirstAmp

Type: Boolean

Description: Use first initialized amplifier as synch producer. If this setting is true (default), then the first amplifier to be initialized will be set as the synch producer, and all other amplifiers will be setup as synch consumers.

Units: None

Default: True

### timeStampID

Type: Integer

Description: High-resolution time stamp CAN ID. The time stamp is a PDO that is generated by the synch producer. It is used to synchronize the clocks of the amplifiers. Setting this to zero will disable the time stamp message.

Units: None

Default: 384 (0x00000180)

# 4.2: Amplifier Initialization

## Methods

### Initialize (canOpenObj As CANopenObj, nodeId As Short)

Description:

Initializes the amplifier with the CANopen network using default Amplifier Settings.

Parameters:

| | | |
|---|---|---|
| canOpenObj | An instance of a CanOpenObj that has already been initialized | Units: None |
| nodeId | The CAN node ID of the amplifier | Units: None |

### InitializeExt (canOpenObj As CANopenObj, nodeId As Short, ampSettings As AmpSettingsObj)

Description:

Initializes amplifier with the CANOpenObj, the specified node ID, and the AmpSettingsObj.

Parameters:

| | | |
|---|---|---|
| canOpenObj | An instance of a CanOpenObj that has already been initialized | Units: None |
| nodeId | The node ID of the amplifier | Units: None |
| ampSettingsObj | An instance of an AmpSettingsObj with customized settings | Units: None |

### InitializeEcat (ecatObj As EcatObj, nodeId As Short)

Description:

Initializes the amplifier with the EcatObj, the specified node ID, and default Amplifier Settings

Parameters:

| | | |
|---|---|---|
| EcatObj | An instance of a EcatObj that has already been initialized | Units: None |
| nodeId | The node ID of the amplifier | Units: None |

### InitializeEcatExt (ecatObj As EcatObj, nodeId As Short, ampSettings As AmpSettingsObj)

Description:

Initializes amplifier with the EcatObj, the specified node ID, and the AmpSettingsObj

Parameters:

| | | |
|---|---|---|
| EcatObj | An instance of a EcatObj that has already been initialized | Units: None |
| nodeId | The node ID of the amplifier | Units: None |
| ampSettingsObj | An instance of an AmpSettingsObj with customized settings | Units: None |

### InitializeEcatSubAxis (primaryAxis as AmpObj)

Description:

Multi-axis amplifiers are treated as a single node on the EtherCAT network. This method allows the sub-axes to be initialized.

Parameters:

| | | |
|---|---|---|
| primaryAxis | An instance of the AmpObj that has been initialized for the primaryAxis | Units: None |

### ReInit ()

Description:

Re-initializes an amplifier using the same properties that were previously used.

Parameters:

None

# 4.3: Amplifier Enable/Disable

## Methods

### Enable ()

Description:

Software enables the amplifier.

Parameters:

None

### Disable ()

Description:

Software disables the amplifier.

Parameters:

None

## Properties

### IsHardwareEnabled

| | |
|---|---|
| Type: | Boolean |
| Description: | Read-only. Returns True if amplifier's Enable input is currently active. Amplifier outputs may still be disabled due to error condition. |
| Units: | None |
| Default: | None |

### IsSoftwareEnabled

| | |
|---|---|
| Type: | Boolean |
| Description: | Read-only. Returns True if amplifier is software enabled. Amplifier outputs may still be disabled due to error condition. |
| Units: | None |
| Default: | None |

### IsPWMEnabled

| | |
|---|---|
| Type: | Boolean |
| Description: | Read-only. Returns true if the amplifier's PWM outputs are currently enabled. |
| Units: | None |
| Default: | None |

# 4.4: Objects Contained by AmpObj

## Overview

In an effort to reduce the number of methods and properties of the AmpObj, several objects were created and added to the AmpObj as a property. Each sub object contains a set of related method and properties.

| Object | Description |
|---|---|
| AmpInfo | Read-only amplifier characteristics. |
| MotorInfo | Motor and feedback parameters. |
| CurrentLoopSettings | Parameters used for tuning the current loop. |
| VelocityLoopSettings | Parameters used to tune the velocity loop. |
| PositionLoopSettings | Parameters used to tune the position loop. |
| HomeSettings | Used to configure homing. |
| ProfileSettings | Used to configure a point-to-point move. |
| TrackingWindows | Used to configure the position and velocity error windows. |

## Example

The following example demonstrates the use of the objects contained by the AmpObj. Please note that the AmpObj must be initialized prior to accessing the sub-objects.

**1** Create an instance. There are two ways to do this:

**Obtain the instance from the AmpObj.** This is the preferred method, because it sets all of the properties of the ProfileSettings object equal to the values set in the AmpObj.

```
Dim profileSettings As ProfileSettingsObj
profileSettings = ampObj.ProfileSettings
```

OR

**Create a new instance.** This sets default values for all of the properties.

```
Dim profileSettings As ProfileSettingsObj
profileSettings = New ProfileSettingsObj
```

**2** Modify one or more properties.

```
profileSettings.ProfileType = CML_PROFILE_TYPE.PROFILE_SCURVE
```

**3** Write the new settings to the AmpObj

```
ampObj.ProfileSettings = profileSettings
```

# 4.5: AmpInfoObj

The properties of the AmpInfoObj provide information about the amplifier. All of the properties are Read-Only.

## Properties

### crntCont

Type: Double
Description: Amplifier continuous current rating.
Units: 0.01 A

### crntPeak

Type: Double
Description: Amplifier peak current rating
Units: 0.01 A

### crntScale

Type: Short
Description: Current scaling factor
Units: None

### crntTime

Type: Double
Description: The maximum time for which the amplifier is rated to output peak current
Units: mS

### mfgInfo

Type: String
Description: Amplifier's manufacturing information string
Units: None

### mfgName

Type: String
Description: Name of the amplifier manufacturer
Units: None

### mfgWeb

Type: String
Description: Web address of the manufacturer
Units: None

### model

Type: String
Description: Model number string
Units: None

### modes

Type: Integer

Description: Supported modes of operation as described in *CANopen Profile for Drives and Motion Control (DSP 402)*.

| Bit | Mode Description |
|---|---|
| 0 | Position profile mode (pp). |
| 2 | Profile velocity mode (pv). |
| 3 | Profile torque mode (tq). |
| 5 | Homing mode (hm). |
| 6 | Interpolated position mode (ip). |
| 7 | Cyclic sync position mode (csp). |
| 8 | Cyclic sync velocity mode (csv). |
| 9 | Cyclic sync torque mode(cst). |

Units: None

### pwm_dbcont

Type: Short

Description: PWM dead time used at or above the continuous current limit

Units: servo cycles

### pwm_dbzero

Type: Short

Description: PWM deadband at zero current

Units: servo cycles

### pwm_off

Type: Short

Description: PWM off time

Units: tens of nanoseconds

### pwmPeriod

Type: Double

Description: PWM period

Units: tens of nanoseconds

### refScale

Type: Short

Description: Reference scaling factor

Units: None

**serial**

Type: Integer
Description: Serial number of the amplifier's printed circuit board
Units: None

**servoPeriod**

Type: Double
Description: Servo loop update period as a multiple of the pwm period
Units: None

**swVer**

Type: String
Description: The firmware version number
Units: None

**tempHyst**

Type: Double
Description: Temperature hysteresis for over temperature fault
Units: degrees C

**tempMax**

Type: Double
Description: Set point for over temperature fault
Units: degrees C

**type**

Type: Short
Description: Amplifier type
Units: None

**voltMax**

Type: Double
Description: Set point for an over voltage fault
Units: 0.1V

**voltMin**

Type: Double
Description: Set point for under voltage fault
Units: 0.1 V

**voltScale**

Type: Short

Description: Voltage scaling factor
Units: 0.1 V

### aencScale

Type: Short
Description: The analog encoder-scaling factor.
Units: None

### regenPeak

Type: Short
Description: The internal regen circuit peak current limit
Units: 0.01 A

### regenCont

Type: Short
Description: The internal regen circuit continuous current limit
Units: 0.01 A

### regenTime

Type: Short
Description: The internal regen circuit time at peak current
Units: mS

### voltHyst

Type: Double
Description: Bus voltage hysteresis for over voltage shutdown
Units: 0.1 Volts

# 4.6: Motor/Feedback Information

## Methods

### ReadAnalogFeedback (Sin As Short, Cos As Short)

Description:

Reads the raw voltage on the two analog feedback inputs.

Parameters:

| | | |
|---|---|---|
| Sin | This parameter will contain the value read on the analog feedback Sin input upon function return | Units: 0.1 mV |
| Cos | This parameter will contain the value read on the analog feedback Sin input upon function return | Units: 0.1 mV |

## Properties

### HallState

| | |
|---|---|
| Type: | Short |
| Description: | Read-only. Contains the current digital hall sensor state. The Hall state is the value of the Hall lines AFTER the ordering and inversions specified in the Hall wiring configuration have been applied. |
| Units: | None |
| Default: | None |

### PhaseAngle

| | |
|---|---|
| Type: | Short |
| Description: | Read-only. Contains the motor phase angle. The phase angle describes the motor's electrical position with respect to its windings |
| Units: | degrees |
| Default: | None |

### MotorInfoObj

| | |
|---|---|
| Type: | MotorInfoObj |
| Description: | This property contains the MotorInfoObj. |
| Units: | None |
| Default: | None |

## MotorInfoObj

### Properties

#### backEMF

| | |
|---|---|
| Type: | Double |
| Description: | Back EMF constant |
| Units: | Rotary: V/KRPM, Linear: V/m/S |

Default: 0.01

**brakeDelay**

Type: Short
Description: Delay between applying brake & disabling PWM.
Units: mS
Default: 0

**brakeVel**

Type: Double
Description: Velocity below which the brake will be applied.
Units: User-defined units/second.
Default: 0.0

**ctsPerRev**

Type: Integer
Description: Encoder counts/revolution. Rotary motors only
Units:
Default: 4000

**eleDist**

Type: Integer
Description: Motor electrical distance. Linear motors only.
Units: Units: encoder units/electrical phase
Default: 100000

**encRes**

Type: Short
Description: Encoder resolution. Linear motors only
Units: encoder units/count
Default: 100

**encReverse**

Type: Boolean
Description: Reverse encoder direction if True.
Units:
Default: False

**encType**

Type: Short
Description: Motor Encoder type

| Value | Description |
|---|---|
| 0 | Incremental quadrature encoder. |
| 1 | No encoder. |
| 2 | Analog encoder. |
| 3 | Secondary quad encoder from input lines. |

| | |
|---|---|
| 4 | Low frequency analog encoder. For use with Copley ServoTube motor. |
| 5 | Resolver. |
| 6 | Use digital hall signals for position & velocity estimates. |
| 7 | Analog encoder updated at current loop rate. |
| 8 | Reserved for custom encoder. |
| 9 | Panasonic |
| 10 | SPI command (reserved for custom firmware use). |
| 11 | EnDat |
| 12 | SSI |
| 13 | BiSS |
| 14 | Serial encoders from Sanyo Denki, Tamagawa, Panasonic and HD systems. |
| 15 | Custom encoders from HD systems. |
| 16 | Simple analog potentiometer feedback. |
| 17-19 | Reserved for custom encoder. |

Units:

Default: 0

**encUnits**

Type: Short

Description: Encoder units. Linear motor only

Units:

Default: 0

**hallOffset**

Type: Short

Description: Hall offset

Units: degrees

Default: 0

**hallType**

Type: Short

Description: Type of hall sensors on the motor.

| Value | Description |
|---|---|
| 0 | No hall sensors available. |
| 1 | Digital hall sensors. |
| 2 | Analog hall sensors. |

Units: None

Default: 1

### hallWiring

Type: Short

Description: Hall wiring code. This bit-mapped value defines the wiring of the hall sensors.

| Bit | Description |
|---|---|
| 0-2 | The hall wiring code which defines the order of the hall connections<br>**Hall Wiring Code** / **Description**<br>0 / U V W<br>1 / U W V<br>2 / V U W<br>3 / V W U<br>4 / W V U<br>5 / W U V<br>6,7 / Reserved |
| 3 | Reserved. |
| 4 | Invert W hall input if set. |
| 5 | Invert V hall input if set. |
| 6 | Invert U hall input if set. |
| 7 | Reserved. |
| 8 | Swap analog halls if set. |
| 9-15 | Reserved. |

Units: None

Default: 0

### hallVelocityShift

Type: Short

Description: This value is used to scale up the calculated velocity in Hall velocity mode (Halls used for feedback in velocity mode). It specifies a left shift value for the position and velocity information calculated in that mode

Units: None

Default: 1

### hasBrake

Type: Boolean

Description: Motor has a brake if True

Units:

Default: False

### inductance

Type: Double

Description: Motor inductance

Units: Henrys

Default: 0.001

**inertia**

Type: Double
Description: Inertia
Units: $Kg\text{-}cm^2$
Default: 0.00001

**mfgName**

Type: String
Description: Name of the motor manufacturer
Units: None
Default: None

**model**

Type: String
Description: Motor model number
Units: None
Default: None

**mtrReverse**

Type: Boolean
Description: Reverse motor wiring if true
Units: None
Default: False

**poles**

Type: Short
Description: Number of pole pairs (number of electrical phases) per rotation. Rotary motors only
Units:
Default: 2

**resistance**

Type: Double
Description: Motor resistance
Units: Ω
Default: 1.0

**stopTime**

Type: Short
Description: Delay between disabling amplifier and applying brake. During this time, amplifier attempts to stop motor
Units: mS
Default: 0

**tempSensor**

Type: Boolean
Description: Motor has a temperature sensor

Units: None
Default: False

**trqConst**

Type: Double
Description: Torque constant (rotary), Force constant (linear). For stepper motors, the value returned is Rated Torque/Rated Current
Units: Rotary: Newton Meters/A; Linear: Newtons/A
Default: 0.001

**trqCont**

Type: Double
Description: Continuous torque (rotary), Continuous force (linear). This parameter is not used for stepper motors
Units: Units: Rotary: Newton Meters; Linear: Newtons
Default: 0.0001

**trqPeak**

Type: Double
Description: Peak torque (rotary), Peak force (linear), Rated Torque (stepper motors)
Units: Rotary, Stepper: Newton Meters; Linear: Newtons
Default: 0.0001

**type**

Type: Short
Description: Bit-mapped value that contains the motor type and family.

| Bits | Description |
|---|---|
| 0 - 1 | Motor Type: 0 = Rotary, 1 = Linear |
| 5 - 6 | Motor Family: 1 = Brush, 2 = Stepper, 3 = Brushless |

Units: None
Default: 0

**velMax**

Type: Double
Description: Maximum motor velocity
Units: User-defined units/second.
Default: 1.0

**encShift**

Type: Short
Description: Analog feedback interpolation value (used only with Analog feedback)
Units: None
Default: 0

**ndxDist**

Type: Integer
Description: Index mark distance (reserved for future use)

Units: None

Default: 0

### stepsPerRev

Type: Integer

Description: Microsteps/revolution (used for Stepnet amplifiers)

Units: None

Default: 4000

### loadEncType

Type: Short

Description: Load Encoder Type. There are two different encodings of this property. The model/firmware version determines which encoding should be used.

For Feature Set E (all versions) and V2.10 or greater for Feature Set C and D, the encoding is as follows:

| Bit | Description |
|---|---|
| 0-11 | Encoder type<br>**Value** / **Description**<br>0 – No load encoder present.<br>1 – Primary (differential) quadrature encoder.<br>2 – Analog encoder.<br>3 – Secondary quadrature encoder from input lines.<br>4 – Low-frequency analog encoder. For use with Copley ServoTube motor.<br>5 – Resolver.<br>6 – Use digital hall signals for position & velocity estimates.<br>7 – Analog encoder updated at current loop rate.<br>8 – Reserved for custom encoder.<br>9 – Panasonic<br>10 – SPI command (reserved for custom firmware use).<br>11 – EnDat<br>12 – SSI<br>13 – BiSS<br>14 – Serial encoders from Sanyo Denki, Tamagawa, Panasonic and HD systems.<br>15 – Custom encoders from HD systems. |
| 12 | Always set to use this new encoding. |
| 13 | Linear if set, rotary if clear. |
| 14 | If set, do not use this encoder for position feedback (passive mode). |
| 15 | Reserved and must be set to zero. |

For Feature Set A and B, the encoding is as follows:

| Bit | Description |
|---|---|
| 0-3 | Encoder type<br>**Value** / **Description**<br>0 – No load encoder present.<br>1 – Primary (differential) quadrature encoder. |

| | | |
|---|---|---|
| | 2 | Analog encoder. |
| | 3 | Secondary quadrature encoder from input lines. |
| | 4 | Low-frequency analog encoder. For use with Copley ServoTube motor. |
| | 5 | Resolver. |
| | 6 | Use digital hall signals for position & velocity estimates. |
| | 7 | Analog encoder updated at current loop rate. |
| | 8 | Reserved for custom encoder. |
| | 9 | Panasonic |
| | 10 | SPI command (reserved for custom firmware use). |
| | 11 | EnDat |
| | 12 | SSI |
| | 13 | BiSS |
| | 14 | Serial encoders from Sanyo Denki, Tamagawa, Panasonic and HD systems. |
| | 15 | Custom encoders from HD systems. |
| 4 | Linear if set, rotary if clear. | |
| 5 | If set don't use this encoder for position feedback (passive mode). | |
| 6-15 | Reserved and must be set to zero. | |

Units: None

Default: 0

### loadEncRes

Type: Integer

Description: Load Encoder Resolution: This is encoder counts/rev for rotary encoders and nanometers/count for linear encoders

Units:

Default: 0

### loadEncReverse

Type: Boolean

Description: Load Encoder Reverse: Reverse load encoder direction if true

Units:

Default: False

### resolverCycles

Type: Short

Description: Number of resolver cycles per motor revolution.

Units:

Default: 1

# 4.7: Current Loop

## Methods

### ReadMotorCurrent (Ucurrent As Short, Vcurrent As Short)

Description:

The actual current values read directly from the amplifier's current sensors. Note that if the motor wiring is being swapped in software, the U and V reading will be swapped.

Parameters:

| | | |
|---|---|---|
| Ucurrent | This parameter will contain the value read on the U winding upon function return | Units: 0.01 A |
| Vcurrent | This parameter will contain the value read on the V winding upon function return | Units: 0.01 A |

## Properties

### CurrentLimited

| | |
|---|---|
| Type: | Short |
| Description: | Read-only. The limited motor current. The commanded current is passed to the current limiter. The output of the current limiter is the limited current, which is passed as an input to the current loop |
| Units: | 0.01 A |
| Default: | None |

### CurrentCommand

| | |
|---|---|
| Type: | Short |
| Description: | Read-only. This current is the input to the current limiter. |
| Units: | 0.01 A |
| Default: | None |

### CurrentActual

| | |
|---|---|
| Type: | Short |
| Description: | Read-only. Gets the actual motor current. This current is based on the amplifier's current sensors, and indicates the portion of current that is being used to generate torque in the motor. |
| Units: | 0.01 A |
| Default: | None |

### TorqueTarget

| | |
|---|---|
| Type: | Short |
| Description: | In profile torque mode, this property is an input to the amplifier's internal trajectory generator. Any change to the target torque triggers an immediate update to the trajectory generator |
| Units: | Thousandths of the rated motor torque |

Default: 0

### TorqueDemand

Type: Short
Description: Read-only. In Profile Torque mode, this is the output value of the torque limiting function
Units: Thousandths of the rated motor torque
Default: None

### TorqueActual

Type: Short
Description: Read-only. Instantaneous torque in the motor
Units: Thousandths of the rated motor torque
Default: None

### TorqueSlope

Type: Integer
Description: Torque acceleration or deceleration
Units: Thousandths of the rated motor torque per second
Default: 0

### CurrentLoopSettings

Type: CurrentLoopSettingsObj
Description: An instance of the CurrentLoopSettingsObj which contains the values set in the amplifier.
Units: None
Default: None

## CurrentLoopSettingsObj

### Properties

#### CrntLoopKp

Type: Short
Description: Current loop proportional gain value
Units: None
Default: 0

#### CrntLoopKi

Type: Short
Description: Current loop integral gain value
Units: None
Default: 0

#### CrntLoopCrntOffset

Type: Short

Description: Current loop offset value
Units: 0.01 A
Default: 0

### CrntLoopPeakCrntLim

Type: Short
Description: Peak current limit. The maximum current that can be applied to the load at any time. In stepper mode, this is the boost current
Units: 0.01 A
Default: 0

### CrntLoopContCrntLim

Type: Short
Description: Continuous current limit. Max current that can continuously be applied to load. In stepper mode, this is the run current
Units: 0.01 A
Default: 0

### CrntLoopPeakCrntTime

Type: Short
Description: Time at peak current limit. In stepper mode, this is time at boost current
Units: mS
Default: 0

### CrntLoopStepHoldCrnt

Type: Short
Description: The Stepper Hold Current. Current used to hold the motor at rest
Units: 0.01A
Default: 0

### CrntLoopStepRunToHoldTime

Type: Short
Description: The Stepper Run To Hold Time. The period of time, beginning when a move is complete, to when the output current is switched to the hold current
Units: mS
Default: 0

### CrntLoopVolControlDelayTime

Type: Short
Description: The Voltage Control Delay Time. If set to zero, feature is disabled.
Units: mS
Default: 0

# 4.8: Velocity Loop

## Properties

### VelocityLimited

Type: Double

Description: Read-only. Gets the limited velocity, which is the result of applying the velocity limiter to the commanded velocity.

Units: User-defined units/second

Default: None

### VelocityCommand

Type: Double

Description: Read-only. The commanded velocity is the velocity value passed to the velocity limiter, and, from there, to the velocity control loop

Units: User-defined units/second

Default: None

### VelocityActual

Type: Double

Description: Read-only. The motor velocity is calculated by the amplifier based on the change in position. For dual encoder systems, the load velocity can be queried by reading the VelocityLoad property

Units: User-defined units/second

Default: None

### VelocityLoad

Type: Double

Description: Read-only. The load velocity is estimated by the amplifier based on the change in position seen at the load encoder. For dual encoder systems, the motor velocity can be queried reading the VelocityActual property

Units: User-defined units/second

Default: None

### VelocityLoopSettings

Type: VelocityLoopSettingsObj

Description: This property contains the VelocityLoopSettings

Units: None

Default: None

# VelocityLoopSettingsObj

## Properties

### VelLoopKp

Type: Short

Description: Velocity loop proportional gain value.

Units: None

Default: 0

### VelLoopKi

Type: Short

Description: Velocity loop integral gain value.

Units: 0

Default:

### VelLoopKaff

Type: Short

Description: Velocity loop acceleration feed forward value.

Units: None

Default: 0

### VelLoopShift

Type: Short

Description: Velocity shift value. After velocity loop is calculated, the result is right-shifted this many times to arrive at the commanded current value. This allows the velocity loop gains to have reasonable values for high-resolution encoders.

Units: None

Default: 0

### VelLoopMaxVel

Type: Double

Description: Velocity loop maximum allowed velocity. Limits the velocity command before the velocity loop uses it to calculate output current.

Units: User-defined units/second

Default: 0.0

### VelLoopMaxAcc

Type: Double

Description: Velocity loop maximum acceleration limit. Limits the rate of change of the velocity command input to the velocity loop. It is used when the magnitude of the command is increasing.

Units: User-defined units/second$^2$

Default: 0.0

### VelLoopMaxDec

Type: Double

Description: Velocity loop maximum deceleration limit. Limits the rate of change of the velocity command input to the velocity loop. It is used when the magnitude of the command is decreasing.

Units: User-defined units/second$^2$

Default: 0.0

### VelLoopEstopDec

Type: Double

Description: Deceleration used for emergency stop. Setting this value to zero indicates that the deceleration is unlimited.

Units: User-defined units/second$^2$

Default: 0.0

# 4.9: Position Loop

## Properties

### PositionError

Type: Double

Description: The position error (difference between position command and actual position).

Units: User-defined units

Default: None

### PositionCommand

Type: Double

Description: The instantaneous position command. This position is the command input to the servo loop. The position command is calculated by the trajectory generator and updated every servo cycle.

Units: User-defined units

Default: None

### PositionActual

Type: Double

Description: The actual position used by the servo loop. For dual encoder systems, this property contains the load encoder position and the PositionMotor property should be used to read the motor encoder position.

Units: User-defined units

Default: None

### PositionMotor

Type: Double

Description: The actual motor position. For single encoder systems, this value is identical to the PositionActual property. For dual encoder systems, this property contains the actual motor position and the PositionActual property may be used to get the load encoder position.

Units: User-defined units

Default: None

### PositionLoadEncoder

Type: Double

Description: Dual encoder systems only. This value is the load encoder position and is the identical to the PositionActual property. When the load encoder is configured for passive mode, this value is the passive load encoder value. This property is not used in single encoder systems.

Units: User-defined units

Default: None

### PositionLoopSettings

Type: PositionLoopSettingsObj
Description: This property contains the PositionLoopSettings.
Units: None
Default: None

## PositionLoopSettingsObj

## Properties

### PosLoopKp

Type: Short
Description: Position loop proportional gain value.
Units: None
Default: 0

### PosLoopKvff

Type: Short
Description: Position loop velocity feed forward value.
Units: None
Default: 0

### PosLoopKaff

Type: Short
Description: Position loop acceleration feed forward value.
Units: None
Default: 0

### PosLoopScale

Type: Short
Description: The output of the position loop is multiplied by this value before being passed to the velocity loop. This scaling factor is calculated such that a value of 100 is a 1.0 scaling factor. This parameter is most useful in dual loop systems.
Units: None
Default: 100

# 4.10: Tracking Windows

## Properties

### TrackingWindows

Type: TrackingWindowsObj

Description: This property contains the TrackingWindows object.

Units: None

Default: None

## TrackingWindowsObj

## Properties

### PositionWarnWindow

Type: Double

Description: Position warning window. If the absolute value of the position error exceeds this value, then a tracking warning will result. A tracking warning causes a bit in the amplifier's status to be set.

Units: User-defined units

Default: 0.0

### SettlingWindow

Type: Double

Description: Position settling window. An amplifier is settled in position after a move when its absolute position error value has been within the settling window for a time greater than the settling time.

Units: User-defined units

Default: 0.0

### SettlingTime

Type: Short

Description: Position settling time value. An amplifier is settled in position after a move when its absolute position error value has been within the settling window for a time greater than the settling time value.

Units: mS

Default: 0

### VelocityWarnWindow

Type: Double

Description: Velocity warning window. If the absolute value of the velocity error exceeds this value, then a velocity warning results. A velocity warning causes a bit in the amplifier's status to be set.

Units: User-defined units

Default: 0.0

### VelocityWarnTime

| | |
|---|---|
| Type: | Short |
| Description: | Velocity warning window time value. If velocity error exceeds velocity warning window, a bit is set in the amplifier status word. Bit is not cleared until velocity error stays within warning window for at least this long. |
| Units: | mS |
| Default: | 0 |

# 4.11: Homing

## Methods

### GoHome ()

Description:

Executes a homing move using the values set in the HomeSettings object.

Parameters:

None

## Properties

### IsReferenced

| | |
|---|---|
| Type: | Boolean |
| Description: | Read-only. Returns True if successfully referenced (homed). |
| Units: | None |
| Default: | False |

### SoftPositionPosLimit

| | |
|---|---|
| Type: | Double |
| Description: | Positive limit position. Any time the motors actual position is greater then this value, a positive software limit condition will be in effect on the amplifier. Software limits are enabled after the amplifier is referenced, and disabled by setting the positive limit equal to the negative limit. |
| Units: | None |
| Default: | 0 |

### SoftPositionNegLimit

| | |
|---|---|
| Type: | Double |
| Description: | Negative limit position. Any time the motors actual position is less then this value, a negative software limit condition will be in effect on the amplifier. Software limits are enabled after the amplifier is referenced, and disabled by setting the positive limit equal to the negative limit. |
| Units: | None |
| Default: | 0 |

### HomeSettingsObj

| | |
|---|---|
| Type: | HomeSettingsObj |
| Description: | Contains the HomeSettingsObj. |
| Units: | None |
| Default: | None |

# HomeSettingsObj

## Properties

### HomeOffset

Type: Double

Description: The home offset value. After the home position is found as defined by the home method, this offset will be added to it and the resulting position will be considered the zero position.

Units: User-defined units

Default: 0.0

### HomeVelFast

Type: Double

Description: Velocity to use for fast moves during the home procedure.

Units: User-defined units/second

Default: 0.0

### HomeVelSlow

Type: Double

Description: Velocity to use when seeking a sensor edge.

Units: User-defined units/second

Default: 0.0

### HomeAccel

Type: Double

Description: Acceleration/deceleration value used for all homing procedure moves.

Units: User-defined units/second$^2$

Default: 0.0

### HomeCurrentLimit

Type: Short

Description: Home current limit in hard stop mode, in which the amplifier drives the motor to the mechanical end of travel (hard stop). End of travel is recognized when the amplifier outputs the HomeCurrent for the HomeDelay time.

Units: 0.01A

Default: 0

### HomeDelay

Type: Short

Description: Delay used for homing to a hard stop in hard stop mode.

Units: mS

Default: 0

### HomeMethod

Type: CML_HOME_METHOD

Description: The method used for homing the amplifier.

Units: None

Default: CHOME _NONE

**CML_HOME_METHOD**

CHOME_NEGATIVE_LIMIT_OUTTO_INDEX = 1

Move into the negative limit switch, then back to the first encoder index pulse beyond it. Index position is home.

CHOME_POSITIVE_LIMIT_OUTTO_INDEX = 2

Move into the positive limit switch, then back to the first encoder index pulse beyond it. Index position is home.

CHOME_POSITIVE_HOME_OUTTO_INDEX = 3

Move to a positive home switch, then back to the first encoder index outside the home region. Index position is home.

CHOME_POSITIVE_HOME_INTO_INDEX = 4

Move to a positive home switch, and continue to the first encoder index inside the home region. Index position is home.

CHOME_NEGATIVE _HOME_OUTTO_INDEX = 5

Move to a negative home switch, then back to the first encoder index outside the home region. Index position is home.

CHOME_NEGATIVE _HOME_INTO_INDEX = 6

Move to a negative home switch, and continue to the first encoder index inside the home region. Index position is home.

CHOME_LOWER_HOME_OUTSIDE_INDEX_POSITIVE = 7

Move to the lower side of a momentary home switch. Then find the first encoder index pulse outside the home region. If the home switch is not active when the home sequence starts, then the initial move will be positive.

CHOME_LOWER_HOME_INSIDE_INDEX_POSITIVE = 8

Move to the lower side of a momentary home switch. Then find the first encoder index pulse inside the home region. If the home switch is not active when the home sequence starts, then the initial move will be positive.

CHOME_UPPER_HOME_INSIDE_INDEX_POSITIVE = 9

Move to the upper side of a momentary home switch. Then find the first encoder index pulse inside the home region. If the home switch is not active when the home sequence starts, then the initial move will be positive.

CHOME_UPPER_HOME_OUTSIDE_INDEX_POSITIVE = 10

Move to the upper side of a momentary home switch. Then find the first encoder index pulse outside the home region. If the home switch is not active when the home sequence starts, then the initial move will be positive

CHOME_UPPER_HOME_OUTSIDE_INDEX_NEGATIVE = 11

Move to the upper side of a momentary home switch. Then find the first encoder index pulse outside the home region. If the home switch is not active when the home sequence starts, then the initial move will be negative.

CHOME_UPPER_HOME_INSIDE_INDEX_NEGATIVE = 12

Move to the upper side of a momentary home switch. Then find the first encoder index pulse inside the home region. If the home switch is not active when the home sequence starts, then the initial move will be negative.

CHOME_LOWER_HOME_INSIDE_INDEX_NEGATIVE = 13

Move to the lower side of a momentary home switch. Then find the first encoder index pulse inside the home region. If the home switch is not active when the home sequence starts, then the initial move will be negative.

CHOME_LOWER_HOME_OUTSIDE_INDEX_NEGATIVE = 14

Move to the lower side of a momentary home switch. Then find the first encoder index pulse outside the home region. If the home switch is not active when the home sequence starts, then the initial move will be negative.

CHOME_POSITIVE_LIMIT = 18

Move into the positive limit switch. The edge of the limit is home.

CHOME _ POSITIVE_HOME = 19

Move to a positive home switch. The edge of the home region is home.

CHOME_ NEGATIVE_HOME = 21

Move to a negative home switch. The edge of the home region is home.

CHOME _LOWER_HOME_POSITIVE = 23

Move to the lower side of a momentary home switch. The edge of the home region is home. If the home switch is not active when the home sequence starts, then the initial move will be positive.

CHOME _UPPER_HOME_POSITIVE = 25

Move to the upper side of a momentary home switch. The edge of the home region is home. If the home switch is not active when the home sequence starts, then the initial move will be positive.

CHOME _UPPER_HOME_ NEGATIVE = 27

Move to the upper side of a momentary home switch. The edge of the home region is home. If the home switch is not active when the home sequence starts, then the initial move will be negative.

CHOME _LOWER_HOME_ NEGATIVE = 29

Move to the lower side of a momentary home switch. The edge of the home region is home. If the home switch is not active when the home sequence starts, then the initial move will be negative.

CHOME _INDEX_ NEGATIVE = 33

Move in the negative direction until the first encoder index pulse is found. The index position is home.

CHOME _INDEX_POSITIVE = 34

Move in the positive direction until the first encoder index pulse is found. The index position is home.

CHOME _NONE = 35

Set the current position to home.

CHOME_HARDSTOP_OUTSIDE_INDEX_NEG = 252

Home to a hard stop. Move in the negative direction until the homing current has been reached. This current will be held until the homing delay has expired. Then move away from the hard stop until an index mark is located. The index position is home.

CHOME_HARDSTOP_OUTSIDE_INDEX_POS = 253

Home to a hard stop. Move in the positive direction until the homing current has been reached. This current will be held until the homing delay has expired. Then move away from the hard stop until an index mark is located. The index position is home.

CHOME_HARDSTOP_NEG = 254

Home to a hard stop. The motor will start running in the negative direction until the homing current has been reached. It will hold this current until the homing delay has expired. The actual position after that delay is home.

CHOME_HARDSTOP_POS = 255

Home to a hard stop. The motor will start running in the positive direction until the homing current has been reached. It will hold this current until the homing delay has expired. The actual position after that delay is home.

# 4.12: Quick Stop

## Methods

### QuickStop ()

Description:

Performs a quick stop on axis using the programmed Quick Stop Mode.

Parameters:

None

## Properties

### QuickStopMode

| | |
|---|---|
| Type: | CML_QUICK_STOP_MODE |
| Description: | Defines how the motor motion is stopped when the QuickStop() command is issued. |
| Units: | None |
| Default: | None |

### CML_QUICK_STOP_MODE

QSTOP_DISABLE = 0

Disable the amplifier immediately

QSTOP_DECEL = 1

Slow down using the ProfileDecel property of the ProfileSettingsObj, then disable.

QSTOP_QUICKSTOP = 2

Slow down using the QuickStopDec property then disable.

QSTOP_ABRUPT = 3

Slow down with unlimited deceleration then disable

QSTOP_DECEL_HOLD = 5

Slow down using the ProfileDecel property of the ProfileSettingsObj, and then hold. Amplifier must be disabled and re-enabled before motion is allowed again.

QSTOP_QUICKSTOP_HOLD = 6

Slow down using the QuickStopDec property then hold. Amplifier must be disabled and re-enabled before motion is allowed.

QSTOP_ABRUPT_HOLD = 7

Slow down with unlimited deceleration then hold. Amplifier must be disabled and re-enabled before motion is allowed.

# 4.13: Halt

## Methods

### HaltMove ()

Description:

Halts current move using the halt mode programmed in the amplifier.

Parameters:

None

## Properties

### HaltMode

| | |
|---|---|
| Type: | CML_HALT_MODE |
| Description: | Defines how the motor motion is stopped when the HaltMove() command is issued. |
| Units: | None |
| Default: | None |

### CML_HALT_MODE

HALT_DISABLE = 0
: Disable the amplifier immediately

HALT_DECEL = 1
: Slow down using the ProfileDecel property (see ProfileSettingsObj).

HALT_QUICKSTOP = 2
: Slow down using the QuickStopDec property.

HALT_ABRUPT = 3
: Slow down with unlimited deceleration

# 4.14: Point-to-Point Moves

## Methods

### MoveRel (distance As Double)

Description:

Performs a relative point-to-point move of the specified distance.

Parameters:

| | | |
|---|---|---|
| distance | Trajectory distance | Units: User-defined units |

### MoveAbs (position As Double)

Description:

Performs an absolute point-to-point move to the specified position.

Parameters:

| | | |
|---|---|---|
| position | Trajectory target position | Units: User-defined units |

### WaitMoveDone (timeout As Long)

Description:

Waits for current move to finish. This method is blocking. When called, it will not return until either the event occurs, the timeout expires, a fault occurs, or a move is aborted. If a timeout occurs, CMO will report the timeout by throwing an exception.

Parameters:

| | | |
|---|---|---|
| timeout | The timeout for the wait. If < 0, then wait indefinitely | Units: mS |

## Properties

### TargetPos

| | |
|---|---|
| Type: | Double |
| Description: | Read-only. Reads the profile target position. |
| Units: | User-defined units |
| Default: | |

### TrajectoryAcc

| | |
|---|---|
| Type: | Double |
| Description: | Read-only. Gets the instantaneous commanded acceleration passed out of the trajectory generator. This acceleration is used by the position loop to calculate its acceleration feed forward term. |
| Units: | User-defined units/second$^2$ |
| Default: | |

### TrajectoryVel

| | |
|---|---|
| Type: | Double |

Description: Read-only. Gets the instantaneous commanded velocity passed out of the trajectory generator. This velocity is used by the position loop to calculate its velocity feed forward term.

Units: User-defined units/second

Default:

#### ProfileSettingsObj

Type: ProfileSettingsObj

Description: Contains the ProfileSettings object.

Units: None

Default: None

## ProfileSettingsObj

### Properties

#### ProfileType

Type: CML_PROFILE_TYPE

Description: Motion profile type.

Units: None

Default: PROFILE_TRAP

#### CML_PROFILE_TYPE

PROFILE_VELOCITY = -1

Velocity profile mode. In this profile mode the velocity, acceleration and deceleration values are used. The position value is also used, but it only defines the direction of motion (positive if position is >= 0, negative if position is < 0).

PROFILE_TRAP = 0

Trapezoidal profile mode.

PROFILE_SCURVE = 3

S-curve profile mode (Jerk limited).

#### ProfileAcc

Type: Double

Description: The profile acceleration value that the motor uses when starting the move.

Units: User-defined units/second$^2$

Default: 0

#### ProfileDecel

Type: Double

Description: The profile deceleration value that the motor uses when ending the move. This property is not used for S-curve profiles.

Units: User-defined units/second$^2$

Default:

### ProfileJerk

Type: Double

Description: The jerk limit used with S-curve profiles. Jerk is rate of change of acceleration. Only used with S-curve profiles.

Units: User-defined units/second$^3$

Default: 100

### ProfileVel

Type: Double

Description: The profile velocity value that the motor attempts to reach during the move.

Units: User-defined units/second

Default: 0

### Profile Abort

Type: Double

Description: Deceleration value to use when aborting a running trajectory.

Units: User-defined units/second$^2$

Default: 0

.

# 4.15: Save/Restore Amplifier Data

## Methods

### LoadFromFile (name As String, line As Integer)

Description:

Loads specified amplifier data file. Presently supports loading **.ccx* files created by CME 2 V3.1 and later. This method is not implemented for multi-axis amplifiers.

NOTE: This method loads the file into the amplifier's RAM (except the motor data, which exists in Flash only). To save the data to the amplifier's Flash, call the SaveRamToFlash.

Parameters:

| | | |
|---|---|---|
| name | Name (and optionally path) of the file to load | Units: None |
| line | If not NULL, the last line number read from the file is returned here | Units: None |

### SaveRamToFlash ()

Description:

Saves parameters stored in the amplifiers volatile RAM memory to non-volatile flash memory.

Parameters:

None

# 4.16: Node Guarding

## Methods

### StartGuarding (guardTime As Short, lifeFactor As Short)

Description:

Starts node guarding with the specified guard time and life factor.

Parameters:

| | | |
|---|---|---|
| guardTime | Node guarding time | Units: mS |
| lifeFactor | Life Factor | Units: None |

### StopGuarding ()

Description:

Disables node guarding & heartbeat monitoring.

Parameters:

None

### ClearNodeGuardEvent ()

Description:

Attempts to clear a node guarding event condition.

Parameters:

None

# 4.17: Status, Events, and Faults

## Methods

### ReadEventStatus (eventStatus As CML_EVENT_STATUS)

Description:

Read amplifier's event status register. This is the main internal register, used to describe the amplifier's current state.

Parameters:

| | | |
|---|---|---|
| eventStatus | The value of the event status is returned here | Units: None |

### ReadEventSticky (eventSticky As CML_EVENT_STATUS)

Description:

Reads the amplifier's 'sticky' event status register, which is a copy of the amplifier's event status register. The bits of this register are set normally, but only cleared when the register is read (i.e., the bits are 'sticky').

Parameters:

| | | |
|---|---|---|
| eventSticky | The value of the event status is returned here | Units: None |

### ReadEventLatch (eventLatch As CML_EVENT_STATUS)

Description:

Reads the latched version of the amplifier's event status register, which is a copy of the amplifier's event status register. The bits of this register are set normally, but only cleared in response to an amplifier reset or power cycle or by calling ClearFaults (i.e., the bits are latched).

Parameters:

| | | |
|---|---|---|
| eventLatch | The value of the event status is returned here | Units: None |

### CML_EVENT_STATUS

| Value | Bit | Description |
|---|---|---|
| EVENT_STATUS_SHORT_CIRCUIT | 0 | Amplifier short circuit. |
| EVENT_STATUS_AMPLIFIER_TEMPERATURE | 1 | Amplifier over temperature. |
| EVENT_STATUS_OVER_VOLTAGE | 2 | Amplifier over voltage. |
| EVENT_STATUS_UNDER_VOLTAGE | 3 | Amplifier under voltage. |
| EVENT_STATUS_MOTOR_TEMPERATURE | 4 | Motor over temperature. |
| EVENT_STATUS_ENCODER_ERROR | 5 | Encoder error. |
| EVENT_STATUS_PHASE_ERROR | 6 | Phasing error. |
| EVENT_STATUS_CURRENT_LIMIT | 7 | Current limited. |
| EVENT_STATUS_VOLTAGE_LIMIT | 8 | Voltage limited. |
| EVENT_STATUS_POSITIVE_LIMIT | 9 | Positive limit is active. |
| EVENT_STATUS_NEGATIVE_LIMIT | 10 | Negative limit is active. |
| EVENT_STATUS_DISABLE_INPUT | 11 | Hardware disabled (enable pin not set). |

| | | |
|---|---|---|
| EVENT_STATUS_SOFTWARE_DISABLE | 12 | Disabled due to software request. |
| EVENT_STATUS_STOP | 13 | Try to stop motor (after disable, before brake). |
| EVENT_STATUS_BRAKE | 14 | Brake actuated. |
| EVENT_STATUS_PWM_DISABLE | 15 | PWM outputs disabled. |
| EVENT_STATUS_SOFTWARE_LIMIT_POSITIVE | 16 | Positive software limit reached. |
| EVENT_STATUS_SOFTWARE_LIMIT_NEGATIVE | 17 | Negative software limit reached. |
| EVENT_STATUS_TRACKING_ERROR | 18 | Tracking error. |
| EVENT_STATUS_TRACKING_WARNING | 19 | Tracking warning. |
| EVENT_STATUS_RESET | 20 | Amplifier has been reset. |
| EVENT_STATUS_POSITON_WRAP | 21 | Encoder position wrapped (rotary) or hit limit (linear). |
| EVENT_STATUS_FAULT | 22 | Latching fault in effect. |
| EVENT_STATUS_VELOCITY_LIMIT | 23 | Velocity is at limit. |
| EVENT_STATUS_ACCELERATION_LIMIT | 24 | Acceleration is at limit. |
| EVENT_STATUS_TRACKING_WINDOW | 25 | Not in tracking window if set. |
| EVENT_STATUS_HOME | 26 | Home switch is active. |
| EVENT_STATUS_MOVING | 27 | Trajectory generator active OR not yet settled. |
| EVENT_STATUS_VELOCITY_WIN | 28 | Velocity error outside of velocity window when set. |
| EVENT_STATUS_PHASE_INIT | 29 | Set when using algorithmic phase initialize mode and the phase is not initialized. |
| EVENT_STATUS_CMD_INPUT_LOST | 30 | Command input lost |
| | 31 | Undefined |

### ReadEventMask (eventMask As CML_AMP_EVENT)

Description:

Reads the current state of the amplifier's event register. The event mask is a bit-mapped variable that describes the state of the amplifier. The contents of this variable are built up from several different amplifier status words.

Parameters:

eventMask The value of the amp event mask is returned here Units: None

### CML_AMP_EVENT

| Value | Bit | Description |
|---|---|---|
| AMPEVENT_MOVE_DONE | 0 | Set when a move is finished and the amplifier has settled in to position at the end of the move. Cleared when a new move is started. |
| AMPEVENT_TRAJECTORY_DONE | 1 | Set when the trajectory generator finishes a move. The motor may not have settled into position at this point. Cleared when a new move is started. |
| AMPEVENT_NODEGUARD | 2 | A node guarding (or heartbeat) error has occurred. |
| AMPEVENT_START_ACKNOWLEDGE | 3 | The Amplifier Object uses this event bit internally. It is set when the amplifier acknowledges a new move |

| | | |
|---|---|---|
| | | start. |
| AMPEVENT_FAULT | 4 | A latching amplifier fault has occurred. The specifics of what caused the fault can be obtained by calling ReadFaults and the fault conditions cleared by calling ClearFaults |
| AMPEVENT_ERROR | 5 | A non-latching amplifier error has occurred. |
| AMPEVENT_POSITION_WARNING | 6 | The amplifier's absolute position error is greater than the window set with PositionWarnWindow. |
| AMPEVENT_POSITION_WINDOW | 7 | The amplifier's absolute position error is greater than the window set with SettlingWindow |
| AMPEVENT_VELOCITY_WINDOW | 8 | The amplifier's absolute velocity error is greater than the window set with VelocityWarnWindow |
| AMPEVENT_DISABLED | 9 | The amplifier's outputs are disabled. The reason for the disable can be determined by calling ReadEventStatus, |
| AMPEVENT_POSITIVE_LIMIT | 10 | The positive limit switch is active. |
| AMPEVENT_NEGATIVE_LIMIT | 11 | The negative limit switch is active. |
| AMPEVENT_SOFTWARE_LIMIT_POSITIVE | 12 | The positive software limit is active. |
| AMPEVENT_SOFTWARE_LIMIT_NEGATIVE | 13 | The negative software limit is active. |
| AMPEVENT_QUICKSTOP | 14 | The amplifier is presently performing a quick stop sequence. |
| AMPEVENT_ABORT | 15 | The last profile was aborted without finishing |
| AMPEVENT_SOFTDISABLE | 16 | The amplifier is software disabled. |
| AMPEVENT_HOME_CAPTURE | 17 | A new home position has been captured. |
| AMPEVENT_PVT_EMPTY | 18 | The PVT buffer is empty. |
| AMPEVENT_PHASE_INIT | 19 | Amplifier is currently performing a phase initialization. |
| | 20-30 | Undefined |
| AMPEVENT_NOT_INITIALIZED | 31 | This amplifier's event mask has not yet been initialized (internal use only). |

### ReadFaults (faults As CML_AMP_FAULT)

Description:

Reads the current state of the amplifier fault latch register.

Parameters:

faults The value of the amp fault latch is returned here Units: None

### ClearFaults ()

Description:

Clears amplifier faults. This function can be used to clear any latching faults on the amplifier

Parameters:

None

## Properties

### FaultMask

Type: CML_AMP_FAULT

Description: Amplifier's fault mask. Fault mask identifies which conditions will be treated as latching faults by the amplifier

Units: None

Default: None

### CML_AMP_FAULT

| Value | Bit | Description |
|---|---|---|
| FAULT_DATAFLASH = 1 | 0 | Fatal hardware error: the flash data is corrupt. |
| FAULT_ADCOFFSET = 2 | 1 | Fatal hardware error: an A/D offset error has occurred. |
| FAULT_SHORT_CIRCUIT = 4 | 2 | The amplifier detected a short circuit condition. |
| FAULT_AMP_TEMPERATURE = 8 | 3 | The amplifier is over temperature. |
| FAULT_MOTOR_TEMPERATURE = 16 | 4 | A motor temperature error was detected. |
| FAULT_OVER_VOLTAGE = 32 | 5 | The amplifier bus voltage is over the acceptable limit. |
| FAULT_UNDER_VOLTAGE = 64 | 6 | The amplifier bus voltage is below the acceptable limit. |
| FAULT_ENCODER_ERROR = 128 | 7 | Encoder error. |
| FAULT_PHASE_ERROR = 256 | 8 | Amplifier phasing error. |
| FAULT_TRACKING_ERROR = 512 | 9 | Tracking error, the position error is too large. |
| FAULT_$I^2T$_LIMIT_ERROR = 1024 | 10 | Current is limited by the $I^2T$ algorithm. |

# 4.18: Digital Inputs/Outputs

## Input Methods

### ReadInputDebouce (input As Integer, time As Long)

Description:

Reads the debounce time for the specified input. This time specifies how long an input must remain stable at a new state before the amplifier recognizes the state.

Parameters:

| | | |
|---|---|---|
| input | The input to configure. Inputs are numbered starting from 0. Check amplifier data sheet for the number of inputs available | Units: None |
| time | The debounce time assigned to this input | Units: mS |

### WriteInputDebounce (input As Integer, time As Long)

Description:

Writes the debounce time for the specified input. This time specifies how long an input must remain stable at a new state before the amplifier recognizes the state.

Parameters:

| | | |
|---|---|---|
| input | The input to configure. Inputs are numbered starting from 0. Check amplifier datasheet for the number of inputs available | Units: None |
| time | The debounce time assigned to this input. | Units: mS |

### ReadInputConfig (input As Integer, config As CML_INPUT_PIN_CONFIG)

Description:

Gets the input configuration for the specified input. Each of the amplifier's inputs can be configured to perform some function.

Parameters:

| | | |
|---|---|---|
| input | Input to read. Inputs are numbered starting from 0. Check amplifier datasheet for number of inputs available | Units: None |
| config | Function assigned to the input | Units: None |

### ReadInputConfigMultiAxis (input As Integer, config As CML_INPUT_PIN_CONFIG, axis as Short)

Description:

Gets the configuration and associated axis number for the specified input.

Parameters:

| | | |
|---|---|---|
| input | Input to read. Inputs are numbered starting from 0. Check amplifier datasheet for number of inputs available | Units: None |
| config | Function assigned to the input | Units: None |
| axis | The axis number this input is associated with (A=0, B=1, etc.) | Units: None |

### WriteInputConfig (input As Integer, config As CML_INPUT_PIN_CONFIG)

Description:

Sets the input configuration for the specified input. Each of the amplifier's inputs can be configured to perform some function. *WriteInputConfig* configures the specified input to perform the specified function.

Parameters:

| | | |
|---|---|---|
| input | Input to read. Inputs are numbered starting from 0. Check amplifier datasheet for number of inputs available | Units: None |
| config | Function assigned to the input | Units: None |

### WriteInputConfigMultiAxis (input As Integer, config As CML_INPUT_PIN_CONFIG, axis as Short)

Description:

Sets the input configuration for the specified input.

Parameters:

| | | |
|---|---|---|
| input | Input to read. Inputs are numbered starting from 0. Check amplifier datasheet for number of inputs available | Units: None |
| config | Function assigned to the input | Units: None |
| axis | The axis number this input is associated with (A=0, B=1, etc.) | Units: None |

### CML_INPUT_PIN_CONFIG

INPUT_CONFIGURATION_NONE = 0
No function assigned to the input.

INPUT_CONFIGURATION_RESET_RISING = 2
Reset the amplifier on the rising edge of the input.

INPUT_CONFIGURATION_RESET_FALLING = 3
Reset the amplifier on the falling edge of the input.

INPUT_CONFIGURATION_POSITIVE_LIMIT_HIGH = 4
Positive limit switch; active high

INPUT_CONFIGURATION_POSITIVE_LIMIT_LOW = 5
Positive limit switch; active low

INPUT_CONFIGURATION_NEGATIVE_LIMIT_HIGH = 6
Negative limit switch, active high

INPUT_CONFIGURATION_NEGATIVE_LIMIT_LOW = 7
Negative limit switch, active low.

INPUT_CONFIGURATION_MOTOR_TEMPERATURE_HIGH = 8
Motor temperature sensor; active high

INPUT_CONFIGURATION_MOTOR_TEMPERATURE_LOW = 9
Motor temperature sensor, active low

INPUT_CONFIGURATION_CLEAR_FAULTS_HIGH = 10
Clear faults on the rising edge; disable while high

INPUT_CONFIGURATION_CLEAR_FAULTS_LOW = 11
Clear faults on the falling edge, disable while low

INPUT_CONFIGURATION_RESET_DISABLE_RISING = 12
Reset on rising edge; disable while high.

INPUT_CONFIGURATION_RESET_DISABLE_FALLING = 13
Reset on falling edge; disable while low.

INPUT_CONFIGURATION_HOME_HIGH = 14
Home switch; active high.

INPUT_CONFIGURATION_HOME_LOW = 15
Home switch; active low

INPUT_CONFIGURATION_DISABLE_HIGH = 16
Amplifier disable; active high

INPUT_CONFIGURATION_DISABLE_LOW = 17
Amplifier disable; active low.

INPUT_CONFIGURATION_PWM_SYNCH = 19
PWM synchronization. Only for high speed inputs (see data sheet).

INPUT_CONFIGURATION_MOTION_ABORT_HIGH = 20
Abort move in progress; keep the amplifier enabled and servoing; active high

INPUT_CONFIGURATION_MOTION_ABORT_LOW = 21
Abort move in progress; keep the amplifier enabled and servoing; active low

INPUT_CONFIGURATION_HIGH_RES_ANALOG_DIVIDE_HIGH = 22
A high input causes the firmware to divide the level of the analog input signal by 8

INPUT_CONFIGURATION_HIGH_RES_ANALOG_DIVIDE_LOW = 23
A low input causes the firmware to divide the level of the analog input signal by 8

INPUT_CONFIGURATION_HIGHSPEED_CAPTURE_RISING = 24
High speed position capture on rising edge

INPUT_CONFIGURATION_HIGHSPEED_CAPTURE_FALLING = 25
High speed position capture on falling edge

INPUT_CONFIGURATION_COUNT_EDGES_RISING = 26
Count rising edges of input, store the results to an indexer register

INPUT_CONFIGURATION_COUNT_EDGES_FALLING = 27

Count falling edges of input, store the results to an indexer register

INPUT_CONFIGURATION_ABORT_WINDOW_RISING = 36
Abort move on rising edge if not within N counts of destination position

INPUT_CONFIGURATION_ABORT_WINDOW_FALLING = 37
Abort move on falling edge if not within N counts of destination position

INPUT_CONFIGURATION_HV_LOSS_DISABLE_HIGH = 38
Mark HV loss on rising edge, disable while high.

INPUT_CONFIGURATION_HV_LOSS_DISABLE_LOW = 39
Mark HV loss on falling edge, disable while low.

INPUT_CONFIGURATION_TRJ_UPDATE_RISING = 40
Trajectory update on rising edge.

INPUT_CONFIGURATION_TRJ_UPDATE_FALLING = 41
Trajectory update on falling edge.

INPUT_CONFIGURATION_CLR_FAULTS_EVENTS_RISING = 42
Clear faults and event latch on rising edge.

INPUT_CONFIGURATION_CLR_FAULTS_EVENTS_FALLING = 43
Clear faults and event latch on falling edge.

INPUT_CONFIGURATION_DIS_SIM_ENC_L_BURST_RISING = 44
Disable simulated encoder output when low. Burst current position on encoder output on rising edge.

INPUT_CONFIGURATION_DIS_SIM_ENC_H_BURST_FALLING = 45
Disable simulated encoder output when high. Burst current position on encoder output on falling edge.

## Input Properties

### Inputs

| | |
|---|---|
| Type: | Integer |
| Description: | Read-only. Gets the present hi/low states of the programmable inputs after debounce. The inputs are returned one per bit. The value of IN1 is returned in bit 0 (1 if high, 0 if low), IN2 in bit 1, etc. |
| Units: | None |
| Default: | None |

### Inputs32

| | |
|---|---|
| Type: | Integer |
| Description: | Read-only. This is the 32-bit version of the Inputs property above. |
| Units: | None |

Default: None

### IoPullup

Type: Integer

Description: State of the pull up/down resistors. Some Copley Controls amplifiers (see amplifier data sheet) have pull up/down resistors connected to a group of inputs. Each bit in the IoPullup property represents one pull up/down resistor; pull up/down resistor 1 is returned in bit 0, pull up/down resistor 2 is return in bit 2, etc. When the bit is set, the inputs connected to the resistor are pulled up to the high state when they are not connected. When the bit is cleared, the inputs are pulled down to a low state when they are not connected

Units: None

Default: None

### IoPullup32

Type: Integer

Description: This is the 32-bit version of the IoPullup property above.

Units: None

Default: None

## Output Methods

### ReadOutputConfig (output As Short, config As CML_OUTPUT_PIN_CONFIG, mask As Integer)

Description:

Reads the configuration for the specified output.

Parameters:

| | | |
|---|---|---|
| output | Input to read. Inputs are numbered starting from 0. Check amplifier datasheet for number of inputs available | Units: None |
| config | Function assigned to the input | Units: None |
| mask | A 32-bit mask used to select which status bits the output should track. If the output is configured for manual mode, then the mask is not used. | Units: None |

### ReadOutputConfigMultiAxis (output As Short, config As CML_OUTPUT_PIN_CONFIG, mask As Integer, axis As short)

Description:

Reads the configuration for the specified output.

Parameters:

| | | |
|---|---|---|
| output | Input to read. Inputs are numbered starting from 0. Check amplifier datasheet for number of inputs available | Units: None |
| config | Function assigned to the input | Units: None |
| mask | A 32-bit mask used to select which status bits the output should track. If the output is configured for manual mode, | Units: None |

| | | |
|---|---|---|
| | then the mask is not used. | |
| axis | The axis number this output is associated with (A=0, B=1, etc.) | Units: None |

### ReadOutputConfigExt (output As Short, config As CML_OUTPUT_PIN_CONFIG, param1 As Integer, param2 As Integer)

Description:

Reads the configuration for the specified output.

Parameters:

| | | |
|---|---|---|
| output | Input to read. Inputs are numbered starting from 0. Check amplifier datasheet for number of inputs available | Units: None |
| config | Function assigned to the input | Units: None |
| param1 | The first 32-bit parameter that defines an output function (used for functions requiring 5 words of data). | Units: None |
| param2 | The second 32-bit parameter that defines an output function (used for functions requiring 5 words of data). | Units: None |

### ReadOutputConfigExtMultiAxis (output As Short, config As CML_OUTPUT_PIN_CONFIG, param1 As Integer, param2 As Integer, axis As Short)

Description:

Reads the configuration for the specified output.

Parameters:

| | | |
|---|---|---|
| output | Input to read. Inputs are numbered starting from 0. Check amplifier datasheet for number of inputs available | Units: None |
| config | Function assigned to the input | Units: None |
| param1 | The first 32-bit parameter that defines an output function (used for functions requiring 5 words of data). | Units: None |
| param2 | The second 32-bit parameter that defines an output function (used for functions requiring 5 words of data). | Units: None |
| axis | The axis number this output is associated with (A=0, B=1, etc.) | Units: None |

### WriteOutputConfig (output As Short, config As CML_OUTPUT_PIN_CONFIG, mask As Integer)

Description:

Sets the configuration for the specified output. Each of the amplifier's outputs can be configured to event status tracking mode or manual mode, as specified by the *config* parameter.

Parameters:

| | | |
|---|---|---|
| output | The output to configure. Outputs are numbered starting from 0. Check amplifier datasheet for the number of outputs available | Units: None |
| config | The function to be assigned to this output. | Units: None |
| mask | A 32-bit mask used to select which status bits the output | Units: None |

should track. If the output is configured for manual mode, then the mask is not used.

### WriteOutputConfigMultiAxis (output As Short, config As CML_OUTPUT_PIN_CONFIG, mask As Integer, axis As Short)

Description:

Sets the configuration for the specified output. Each of the amplifier's outputs can be configured to event status tracking mode or manual mode, as specified by the *config* parameter.

Parameters:

| | | |
|---|---|---|
| output | The output to configure. Outputs are numbered starting from 0. Check amplifier datasheet for the number of outputs available | Units: None |
| config | The function to be assigned to this output. | Units: None |
| mask | A 32-bit mask used to select which status bits the output should track. If the output is configured for manual mode, then the mask is not used. | Units: None |
| axis | The axis number this output is associated with (A=0, B=1, etc.) | Units: None |

### WriteOutputConfigExtMultiAxis (output As Short, config As CML_OUTPUT_PIN_CONFIG, param1 As Integer, param2 As Integer, axis As Short)

Description:

Sets the configuration for the specified output. Each of the amplifier's outputs can be configured to event status tracking mode, position triggered mode, or manual mode, as specified by the *config* parameter

Parameters:

| | | |
|---|---|---|
| output | The output to configure. Outputs are numbered starting from 0. Check amplifier datasheet for the number of outputs available | Units: None |
| config | The function to be assigned to this output | Units: None |
| param1 | The first 32-bit parameter that defines an output function (used for functions requiring 5 words of data). | Units: None |
| param2 | The second 32-bit parameter that defines an output function (used for functions requiring 5 words of data). | Units: None |
| axis | The axis number this output is associated with (A=0, B=1, etc.) | Units: None |

### CML_OUTPUT_PIN_CONFIG

OUTPUT_CONFIGURATION_EVENT_STATUS_LOW = 0

The output follows the amplifier's event status register and is active low.

| | |
|---|---|
| param1 | A 32-bit mask used to select which status bits the output should track. |
| param2 | Has no meaning. Set to zero. |

OUTPUT_CONFIGURATION_ EVENT_STATUS_HIGH = 256

The output follows the amplifier's event status register and is active high

| | |
|---|---|
| param1 | A 32-bit mask used to select which status bits the output should track. |
| param2 | Has no meaning. Set to zero. |

OUTPUT_CONFIGURATION_EVENT_LATCH_LOW = 1

The output follows the latched version of the amplifier's event status register and is active low

| | |
|---|---|
| param1 | A 32-bit mask used to select which status bits the output should track. |
| param2 | Has no meaning. Set to zero. |

OUTPUT_CONFIGURATION_EVENT_LATCH_HIGH = 257

The output follows the latched version of the amplifier's event status register and is active high

| | |
|---|---|
| param1 | A 32-bit mask used to select which status bits the output should track. |
| param2 | Has no meaning. Set to zero. |

OUTPUT_CONFIGURATION_MANUAL_LOW = 2

The output is manually controlled using Outputs property and is active low. This method does not use parameters; set all parameters to zero

OUTPUT_CONFIGURATION_MANUAL_HIGH = 258

The output is manually controlled using Outputs property, and is active high. This method does not use parameters; set all parameters to zero

OUTPUT_CONFIGURATION_TRAJECTORY_STATUS_LOW = 3

The output pin follows bits in the amplifier's trajectory status register and is active low

OUTPUT_CONFIGURATION_ TRAJECTORY_STATUS_HIGH = 259

The output pin follows bits in the amplifier's trajectory status register and is active high

| | |
|---|---|
| param1 | A 32-bit mask used to select which status bits the output should track. |
| param2 | Has no meaning. Set to zero. |

OUTPUT_CONFIGURATION_POSITION_WINDOW_LOW = 4

The output goes active low if the actual motor position is greater than param1 and less than param2

| | |
|---|---|
| param1 | Low edge of position trigger window. Units: Counts. |
| param2 | High edge of position trigger window. Units: Counts. |

OUTPUT_CONFIGURATION_POSITION_WINDOW_HIGH = 260

The output goes active high if the actual motor position is greater than param1 and less than param2

param1 Low edge of position trigger window. Units: Counts.

param2 High edge of position trigger window. Units: Counts.

OUTPUT_CONFIGURATION_MOTION_POSITIVE_LOW = 5

The output goes active low when the motor actual position crosses in the low-to-high direction through the point specified in param1. The pin stays active for amount of time specified in param2

param1 Trigger position. Units: Counts.

param2 Output active time. Units: milliseconds.

OUTPUT_CONFIGURATION_MOTION_POSITIVE_HIGH = 261

The output goes active high when the motor actual position crosses in the low-to-high direction through the point specified in param1. The pin stays active for amount of time specified in param2.

param1 Trigger position. Units: Counts.

param2 Output active time. Units: milliseconds.

OUTPUT_CONFIGURATION_MOTION_NEGATIVE_LOW = 6

The output goes active low when the motor actual position crosses in the high-to-low direction through the point specified in param1. The pin stays active for amount of time specified in param2

param1 Trigger position. Units: Counts.

param2 Output active time. Units: milliseconds.

OUTPUT_CONFIGURATION_MOTION_NEGATIVE_HIGH = 262

The output goes active high when the motor actual position crosses in the high-to-low direction through the point specified in param1. The pin stays active for amount of time specified in param2

param1 Trigger position. Units: Counts.

param2 Output active time. Units: milliseconds.

OUTPUT_CONFIGURATION_TRIG_AT_POSITION_LOW = 7

The output goes active low when the motor actual position crosses in any direction through the point specified in param1. The pin stays active for amount of time specified in param2

param1 Trigger position. Units: Counts.

param2 Output active time. Units: milliseconds.

OUTPUT_CONFIGURATION_TRIG_AT_POSITION_HIGH = 263

The output goes active high when the motor actual position crosses in any direction through the point specified in param1. The pin stays active for amount of time specified in param2

param1 Trigger position. Units: Counts.

param2 Output active time. Units: milliseconds.

OUTPUT_CONFIGURATION_PWM_SYNCH = 512
PWM Synchronization. Note: Valid only on Output 0. This method does not use parameters; set all parameters to zero

## Output Properties

### Outputs

Type: Integer

Description: Reads or writes the present states (active/inactive) of the programmable outputs. When this property is read, the current active/inactive state of all outputs is returned. Each output is represented by one bit in the returned value; bit 0 for output 1, bit 1 for output 2, etc. When this property is written, it is used to control the active/inactive state of any outputs that are configured to operate in manual mode. Writing a 1 to a bit causes the corresponding output to become active; writing a 0 causes the output to become inactive. Bits corresponding to outputs that are not configured in manual mode are ignored

Units: None

Default: None

# 4.19: Amplifier Events

## Methods

### CreateEvent (mask As CML_AMP_EVENT, condition As CML_EVENT_CONDITION) As EventObj

Description:

Creates an instance of EventObj, using specified parameters to monitor amplifier events.

Parameters:

| | | |
|---|---|---|
| mask | The bit-mapped value that indicates which events are to be monitored | Units: None |
| condition | The trigger condition for the events that will result in the event callback method being called (e.g. all events in the mask) | Units: None |

### CreateInputEvent (mask As Integer, condition As CML_EVENT_CONDITION) As EventObj

Description:

Reads the configuration for the specified output.

Parameters:

| | | |
|---|---|---|
| mask | A bit-mapped value that indicates which digital input pin is to be monitored. Each corresponds to one input pin; bit 0 for input 0, bit 1 for input 1, etc | Units: None |
| condition | The trigger condition for the events that will result in the event callback method being called (e.g. all events in the mask) | Units: None |

### CML_EVENT_CONDITION

CML_EVENT_ANY = 1
: Any event occurring

CML_EVENT_ALL = 2
: All the events are required

CML_EVENT_NONE = 3
: None of the events

# 4.20: Amplifier Trace

The trace system allows internal amplifier parameters to be sampled and stored at a specified interval. The stored data may later be downloaded for analysis. The typical sequence of steps involved to run the trace is as follows:

1. Set up the trace channels, sample period and trigger.
2. Start the trace.
3. Monitor the status until the trace has triggered and no longer running.
4. Read in the trace data.

The example, EX7_Trace, is provided with the installation of CMO. This example demonstrates the steps necessary to run the trace and save the trace data to a file.

## Methods

### ReadTraceStatus (status As CML_AMP_TRACE_STATUS, samplesCollected As Short, maxSamples As Short)

Description:

Read the status of the amplifier's trace system as a bit mapped value. For most tracing applications, only the first two bits are observed.

| Bit | Definition |
|---|---|
| 0 | Trace is running |
| 1 | Trace has triggered |
| 2 | Sampled mode |
| 3 | Trace will ignore initial delays |

A typical sequence is as follows:

1. The trace is started; bit 0 will be set to indicate that the trace is running.
2. When the trigger condition is met, bit 1 will be set.
3. Once the trigger occurs, the trace will start collecting data.
4. The trace is done collecting data; bit 0 will be cleared and the trace data can be read.

Parameters:

| | | |
|---|---|---|
| status | Information on whether the trace is currently running is returned in this parameter | Units: None |
| samplesCollected | The total number of trace samples collected is returned here | |
| maxSamples | The maximum number of trace samples that will fit in the internal buffer is returned here. This value will change depending on how many trace channels are active and which variables are selected. | |

### CML_AMP_TRACE_STATUS

TRACE_STATUS_RUNNING = 1
: Trace is currently collecting data.

TRACE_STATUS_TRIGGERED = 2
: Trace has been triggered

TRACE_STATUS_SAMPLED = 4
: Trace is currently in sampled mode

TRACE_STATUS_NODELAY = 8
: Trace is configured to ignore initial delays

### ReadTraceRefPeriod (ref Period As Integer)

Description:

Read-only. Read the fundamental period used with the amplifier's trace. The amplifier internally samples its trace channels at multiples of this time. For example, if the amplifier's reference period is 62500 nanoseconds, then setting the trace period to 10 would indicate that the amplifier should sample its internal variables every 625 μS.

Parameters:

| | | |
|---|---|---|
| refPeriod | The reference period is returned here. | Units: nS |

### WriteTracePeriod (tracePeriod As Short)

Description:

Set the trace period. The rate at which samples are read by the trace is the product of this value and the TraceRefPeriod.

Parameters:

| | | |
|---|---|---|
| tracePeriod | The trace period to be set | Units: multiple of TraceRefPeriod |

### ReadTracePeriod (tracePeriod As Short)

Description:

Set the trace period. The rate at which samples are read by the trace is the product of this value and the TraceRefPeriod.

Parameters:

| | | |
|---|---|---|
| tracePeriod | The trace period is returned here | Units: multiple of TraceRefPeriod |

### WriteTraceTrigger (type As CML_AMP_TRACE_TRIGGER, channel As Short, level As Integer, delay As Short)

Description:

Configure the trace trigger. The trigger resembles the trigger on an oscilloscope. It allows an event to be specified which will cause the trace to start collecting data. Most trigger types watch one of the trace channels and constantly compare its value to a level. The type of comparison made will depend on the type of trigger. For example, the trace can be triggered on the rising edge of a signal, on the falling edge, etc. The trigger also allows a delay value to be specified. The delay specifies the number trace periods to wait after the

trigger occurs to start collecting samples. The delay can also be negative, in which case the delay specifies the number of trace periods to collect data before the trigger occurs.

Parameters:

| | | |
|---|---|---|
| type | The trigger type | Units: None |
| channel | The trace channel to watch. This parameter defaults to 0 if not specified | Units: None |
| level | The trigger level. This parameter defaults to 0 if not specified | Units: Varies with trigger type and the trace channel variable |
| delay | The delay between the occurrence of the trigger and the start of data collection. | Units: trace periods |

### ReadTraceTrigger (type As CML_AMP_TRACE_TRIGGER, channel As Short, level As Integer, delay As Short)

Description:

Get the current configuration of the trace trigger.

Parameters:

| | | |
|---|---|---|
| type | The type of trigger to be used | Units: None |
| channel | Which channel to trigger on | Units: None |
| level | The trigger level | Units: Varies with trigger type and the trace channel variable |
| delay | The delay between the occurrence of the trigger and the start of data collection. Defaults to 0 if not specified | Units: trace periods |

### CML_AMP_TRACE_TRIGGER

TRACETRIG_NONE = 0

Trace trigger type none. The trace is triggered immediately on start

TRACETRIG_ABOVE = 256

Trigger as soon as the value on the selected variable is above the trigger level

TRACETRIG_BELOW = 512

Trigger as soon as the value on the selected variable is below the trigger level.

TRACETRIG_RISE = 768

Trigger when the value on the selected variable changes from below the trigger level to above it.

TRACETRIG_FALL = 1024

Trigger when the value on the selected variable changes from above the trigger level to below it

TRACETRIG_BITSET = 1280

Treat the trigger level as a bit mask which selects one or more bits on the selected trace variable. The trigger occurs as soon as any of the selected bits are set.

TRACETRIG_BITCLR = 1536

Treat the trigger level as a bit mask which selects one or more bits on the selected trace variable. The trigger occurs as soon as any of the selected bits are clear.

TRACETRIG_CHANGE = 1792

Trigger any time the selected trace variable value changes

TRACETRIG_EVENTSET = 2048

Treat the trigger level as a bit mask which selects one or more bits on the amplifier's event status register. The trigger occurs as any of the selected bits are set

TRACETRIG_EVENTCLR = 2304

Treat the trigger level as a bit mask which selects one or more bits on the amplifier's event status register. The trigger occurs as any of the selected bits are clear

TRACETRIG_FGEN_CYCLE = 2560

Trigger at the start of the next function generator cycle. This trigger type is only useful when running in function generator mode

TRACETRIG_NODELAY = 16384

If this bit is set, then the trigger is allowed to occur even if the trace setup delay has not yet occurred

TRACETRIG_SAMPLE = 32768

Only take a single sample for each trigger. Normally, the occurrence of the trigger causes the trace to begin sampling data and stop when the trace buffer is full.

### ReadTraceMaxChannel (maxChannels As Short)

Description:

Return the maximum number of trace channels supported by the amplifier.

Parameters:

| | | |
|---|---|---|
| maxChannels | The number of channels is returned here | Units: None |

### TraceStart ()

Description:

Start collecting trace data on the amplifier. The trace will automatically stop once the amplifier's internal trace buffer fills up.

Parameters:

None

### TraceStop ()

Description:

Stop collecting trace data on the amplifier.

Parameters:

None

### ReadTraceData (traceDataArray As Integer, dataCount As Integer)

Description:

Upload any trace data captured in the amplifier. Trace data should only be uploaded when the trace has both triggered and stopped. Uploading data during data collection can cause corrupt data to be uploaded. The trace data is returned as an array of 32-bit integer values. The data for all active channels is contained within the trace data array. For example, if there are three active channels, then the trace data array will be formatted as shown below:

| Index 0 | Index 1 | Index 2 | Index 3 | Index 4 | Index 5 | Index 6 | Index7 | Index 8 |
|---|---|---|---|---|---|---|---|---|
| Chan 1 | Chan 2 | Chan 3 | Chan 1 | Chan 2 | Chan 3 | Chan 1 | Chan 2 | Chan 3 |

Parameters:

| | | |
|---|---|---|
| traceDataArray | An array where the trace data will be returned | Units: None |
| dataCount | On entry to this call, this parameter must hold the maximum number of samples to upload. Upon successful return, this parameter will contain the total number samples returned. | Units: None |

### WriteTraceChannel (channel As Short, traceVar CML_AMP_TRACE_VAR)

Description:

Set the trace variable associated with the specified channel.

Parameters:

| | | |
|---|---|---|
| channel | The trace channel that the variable will be assigned to (zero based). | Units: None |
| traceVar | The trace variable to sample | Units: None |

### WriteTraceChannel (channel As Short, traceVar CML_AMP_TRACE_VAR, axis As Integer)

Description:

Set the trace variable associated with the specified channel.

Parameters:

| | | |
|---|---|---|
| channel | The trace channel that the variable will be assigned to (zero based). | Units: None |
| traceVar | The trace variable to sample | Units: None |
| axis | The axis number this channel is associated with (A=0, B=1, etc.) | Units: None |

### ReadTraceChannel (channel As Short, traceVar CML_AMP_TRACE_VAR)

Description:

Read the trace variable associated with the specified channel.

Parameters:

| | | |
|---|---|---|
| channel | The trace channel to get (zero based) | Units: None |
| traceVar | The trace variable assigned to this channel will be returned here | Units: None |

### ReadTraceChannel (channel As Short, traceVar CML_AMP_TRACE_VAR, axis As Integer)

Description:

Read the trace variable associated with the specified channel.

Parameters:

| | | |
|---|---|---|
| channel | The trace channel to get (zero based) | Units: None |
| traceVar | The trace variable assigned to this channel will be returned here | Units: None |
| axis | The axis number this channel is associated with (A=0, B=1, etc.) | Units: None |

### CML_AMP_TRACE_VAR

TRACEVAR_CRNT_U = 3
: Actual current, U winding. Units: 0.01 A.

TRACEVAR_CRNT_V = 4
: Actual current, V winding. Units: 0.01 A

TRACEVAR_ANALOG_REF = 5
: Analog reference input. Units: mV

TRACEVAR_HIGH_VOLT = 6
: High voltage bus. Units: 0.1 V

TRACEVAR_CRNT_CMD = 7
: Commanded current (before limiting). Units: 0.01 A

TRACEVAR_CRNT_LIM = 8
: Commanded current (after limiting). Units: 0.01 A

TRACEVAR_CRNT_CMD_D = 9
: Commanded current, D axis. Units: 0.01 A

TRACEVAR_CRNT_CMD_Q = 10
: Commanded current, Q axis. Units: 0.01 A

TRACEVAR_CRNT_ACT_D = 13
: Actual current, calculated for D axis. Units: 0.01 A

TRACEVAR_CRNT_ACT_Q = 14
: Actual current, calculated for Q axis. Units: 0.01 A.

TRACEVAR_CRNT_ERR_D = 15
: Current loop error, D axis. Units: 0.01 A

TRACEVAR_CRNT_ERR_Q = 16
: Current loop error, Q axis. Units: 0.01 A

TRACEVAR_VOLT_D = 19
: Current loop output voltage, D axis. Units: 0.1 V

TRACEVAR_VOLT_Q = 20

Current loop output voltage, Q axis. Units: 0.1 V

TRACEVAR_VEL_MTR = 23

Motor velocity filtered. Units: 0.1 encoder counts / second

TRACEVAR_VLOOP_CMD = 24

Velocity loop commanded velocity (before limiting). Units: 0.1 encoder counts / second.

TRACEVAR_VLOOP_LIM = 25

Velocity loop commanded velocity (after limiting). Units: 0.1 encoder counts / second

TRACEVAR_VLOOP_ERR = 26

Velocity loop error. Units: 0.1 encoder counts / second

TRACEVAR_LOAD_POS = 28

Load encoder position. Units: encoder counts.

TRACEVAR_CMD_POS = 29

Commanded position from trajectory generator. Units: encoder counts

TRACEVAR_POS_ERR = 30

Position error. Units: encoder counts

TRACEVAR_MTR_POS = 31

Motor encoder position. Units: encoder counts

TRACEVAR_RAW_INPUTS = 33

Digital input pins (before debounce).

TRACEVAR_PHASE = 36

Motor phase angle. Units: 0.1 degree

TRACEVAR_TEMP = 37

Amplifier temperature. Units: degrees C

TRACEVAR_EVENTS = 38

Event status register.

TRACEVAR_EVENTLATCH = 39

Latched version of event status register

TRACEVAR_HALLS = 40

Hall sensor state

TRACEVAR_VEL_LOAD = 43

Load encoder velocity. Units: 0.1 encoder counts / second

TRACEVAR_CMD_VEL = 44

Commanded velocity from trajectory generator.
Units: 0.1 encoder counts / second

TRACEVAR_CMD_ACC = 45
Commanded acceleration from trajectory generator. Units: 10 encoder counts / second / second

TRACEVAR_ENC_SIN = 46
Analog encoder sine. Units: 0.1 mV.

TRACEVAR_ENC_COS = 47
Analog encoder cosine. Units: 0.1 mV

TRACEVAR_INPUTS = 48
Digital input pins (after debounce)

TRACEVAR_DEST_POS = 49
Destination position. Units: encoder counts

TRACEVAR_VEL_RAW = 50
Motor velocity, unfiltered. Units: 0.1 encoder counts / second

TRACEVAR_PASSIVE_ENC_POS = 51,
Passive encoder position

TRACEVAR_GAIN_SCHED_KEY = 52,
Gain scheduling key

TRACEVAR_POS_P_GAIN = 53,
Position loop proportional gain

TRACEVAR_VEL_P_GAIN = 54,
Velocity loop proportional gain

TRACEVAR_VEL_I_GAIN = 55,
Velocity loop integral gain

TRACEVAR_AMP_I2T_SUM = 56,
Amplifier's I2T sum

TRACEVAR_USER_I2T_SUM = 57,
User's I2T sum

TRACEVAR_ANALOG_ENC_INDEX = 59,
Analog encoder index pulse

TRACEVAR_COMMANDED_U = 60,
Commanded current U

TRACEVAR_COMMANDED_V = 61,
Commanded current V

TRACEVAR_CUR_OFFSET_CSP = 62,
Current offset, EtherCAT CSP mode

TRACEVAR_VEL_OFFSET_CSP = 63,
Velocity offset, EtherCAT CSP mode

TRACEVAR_RAW_ENCODER = 66
Raw encoder values

# 4.21: Other Methods and Properties

## Methods

### Reset ()

Description:

Resets the Amplifier and re-initializes the Amplifier Object.

Parameters:

None

### SDO_Dnld (index As Short, sub As Short, data As Object)

Description:

Downloads data to the amplifier via a CAN SDO transfer.

Parameters:

| | | |
|---|---|---|
| index | Index of a CANopen dictionary object | Units: None |
| sub | Sub-index of a CANopen dictionary object | Units: None |
| data | The data that is to be transferred. This data can be one of four types: 8-bit, 16-bit, 32-bit, or String | Units: None |

### SDO_Upld (index As Short, sub As Short, data As Object)

Description:

Uploads data from the amplifier via a CAN SDO transfer.

Parameters:

| | | |
|---|---|---|
| index | Index of a CANopen dictionary object | Units: None |
| sub | Sub-index of a CANopen dictionary object | Units: None |
| data | The data that is to be transferred. This data can be one of four types: 8-bit, 16-bit, 32-bit, or String | Units: None |

### SDO_DnldExt (index As Short, sub As Short, data As Byte, size As Integer)

Description:

Downloads data to the amplifier via a CAN SDO transfer.

Parameters:

| | | |
|---|---|---|
| index | The input to configure. Inputs are numbered starting from 0. Check amplifier data sheet for the number of inputs available | Units: None |
| time | The debounce time assigned to this input | Units: None |
| data | The data that is to be transferred | Units: None |
| size | The number of bytes of data to be downloaded | Units: None |

### SDO_UpldExt (index As Short, sub As Short, data As Byte, size As Integer)

Description:

Uploads data from the amplifier via a CAN SDO transfer.

Parameters:

| | | |
|---|---|---|
| index | The input to configure. Inputs are numbered starting from 0. Check amplifier data sheet for the number of inputs available | Units: None |
| time | The debounce time assigned to this input | Units: None |
| data | The data that is to be transferred | Units: None |
| size | On entry this gives the max number of bytes of data to be uploaded. On successful return this gives the actual number of bytes received | Units: None |

## Properties

### CountsPerUnit

Type: Double

Description: Adjustable number of encoder counts/user distance unit. The default value is 1.0 (user distance units are in encoder counts). Also controls velocity, acceleration, and jerk units. These units are always based on a time interval of seconds.

Units: None

Default: None

### AmpTemp

Type: Short

Description: Read-only. Get the current amplifier temperature

Units: degrees C

Default: None

### HighVoltage

Type: Short

Description: Read-only. Gets the high voltage bus voltage

Units: 0.1 V

Default: None

### RefVoltage

Type: Short

Description: Read-only. Gets the analog reference input voltage

Units: mV

Default: None

### AmpMode

Type: CML_AMP_MODE

Description: Read-only. The currently active amplifier mode of operation

Units: None

Default: None

### AmpModeWrite

Type: CML_AMP_MODE

Description: Change the amplifiers mode of operation

Units: None

Default: None

### CML_AMP_MODE

AMPMODE_SERVO_CAN_PROFILE = 7681

A true CANopen position mode. The CANopen network sends move commands to the amplifier, and the amplifier uses its internal trajectory generator to perform the moves. Conforms to the CANopen Device Profile for Motion Control (DSP-402) profile position mode

AMPMODE_SERVO_CAN_VELOCITY = 7683

In this mode the CANopen network commands target velocity values to the amplifier. The amplifier uses its programmed acceleration and deceleration values to ramp the velocity up/down to the target. Note that support for profile velocity mode was added in amplifier firmware version 3.06

AMPMODE_SERVO_CAN_TORQUE = 7684

In this mode, the network controller sends target torque values to the drive. When the drive is enabled, or the torque command is changed, the motor torque ramps to the new value at the rate programmed in the property Torque Slope. When the drive is halted, the torque ramps down at the same rate.

When using Profile Torque mode, the property HaltMode can be set to any mode except HALT_DISABLE, because HALT_DISABLE will disable the amplifier with no torque ramp.

If the torque target value is changed while the amplifier is enabled, the torque will ramp to the new target.

The units for torque target, demand, and actual are per thousand of the motor's rated torque. The units for torque slope are per thousand of the motor's rated torque per second.

The profile torque mode cannot be used with a stepper motor

AMPMODE_SERVO_CAN_HOMING = 7686

A true CANopen position mode. Used to home the motor (find the motor zero position) under CANopen control. Conforms to DSP-402 homing mode

AMPMODE_SERVO_CAN_PVT = 7687

A true CANopen position mode. In this mode the CANopen master calculates the motor trajectory and streams it over the CANopen network as a set of points that the amplifier interpolates between. This mode conforms to the CANopen device profile for motion control (DSP-402) interpolated position mode

AMPMODE_STEPPER_CAN_PROFILE = 10241

Same as AMPMODE_SERVO_CAN_PROFILE, but used with stepper capable amplifiers

AMPMODE_STEPPER_CAN_VELOCITY = 10243

Same as AMPMODE_SERVO_CAN_ VELOCITY, but used with stepper capable amplifiers

AMPMODE_STEPPER_CAN_HOMING = 10246

Same as AMPMODE_SERVO_CAN_ HOMING, but used with stepper capable amplifiers

AMPMODE_STEPPER_CAN_PVT = 10247

Same as AMPMODE_SERVO_CAN_PVT, but used with stepper capable amplifiers

# CHAPTER 5: LINKAGE

# 5.1: LinkageSettingsObj

## Overview

The Linkage Settings Object contains the settings for the LinkageObj. All of the properties have both read and write access. This object is passed in as a parameter in the InitializeExt method of the LinkageObj to customize the settings.

## Example:

1. Declare and create an instance of LinkageSettingsObj.

```
Dim LinkageSettings As LinkageSettingsObj
LinkageSettings = New LinkageSettingsObj()
```

2. Change one or more properties of the LinkageSettingsObj.

```
LinkageSettings.moveAckTimeout = 400
```

3. Call one of the Extended Initialization methods of the ampObj.

```
Linkage.InitializeExt(ampArray, LinkageSettings)
```

## Properties

### moveAckTimeout

| | |
|---|---|
| Type: | Short |
| Description: | Node guarding guard time. This property gives the node-guarding period for use with this node. This is the period between node guarding request messages sent by the master controller. |
| Units: | mS |
| Default: | 200 |

### haltOnPosWarn

| | |
|---|---|
| Type: | Boolean |
| Description: | When set to true, the linkage move will be halted when a position warning occurs. |
| Units: | none |
| Default: | false |

### haltOnVelWin

| | |
|---|---|
| Type: | Boolean |
| Description: | When set to true, the linkage move will be halted when the velocity is outside the velocity window. |
| Units: | none |
| Default: | false |

# 5.2: LinkageObj

## Overview

The Linkage Object allows the programmer to "link" a group of amplifiers to perform coordinated motion. A move using the Linkage Object will start moving all the linked amplifiers at the same time and end the move at the same time.

## Methods

### Initialize (ampArray As AmpObj)

Description:

Initializes the Linkage object with the array of amp objects passed in as a parameter. These amp objects will be linked together upon successful initialization.

Parameters:

| | | |
|---|---|---|
| ampArray | Array of one or more AmpObj (which have already been initialized) | Units: None |

### InitializeExt (ampArray As AmpObj, linakeSettings as LinkageSettingsObj)

Description:

Initializes the Linkage object with the array of amp objects and the linkage settings passed in as parameters. The amp objects in the ampArray will be linked together upon successful initialization.

Parameters:

| | | |
|---|---|---|
| ampArray | Array of one or more AmpObj (which have already been initialized) | Units: None |
| LinkageSettings | Array of one or more AmpObj (which have already been initialized) | Units: None |

### MoveTo (positionArray As Double)

Description:

Performs a multi-axis move to the positions specified by an array containing one position per axis.

Parameters:

| | | |
|---|---|---|
| positionArray | Contains the target positions for each axis | Units: Double |

### ReadMoveLimits (vel As Double, acc As Double, dec As Double, jrk As Double)

Description:

Reads the limits for a move.

Parameters:

| | | |
|---|---|---|
| vel | Velocity limit | Units: User defined units/second |
| acc | Acceleration limit | Units: User-defined units/second$^2$ |

| | | |
|---|---|---|
| dec | Deceleration limit | Units: User-defined units/second$^2$ |
| jrk | Jerk limit (maximum rate of change of acceleration) | Units: User-defined units/second$^3$ |

### SetMoveLimits (vel As Double, acc As Double, dec As Double, jrk As Double)

Description:

Sets the limits for the move.

Parameters:

| | | |
|---|---|---|
| vel | Velocity limit | Units: User defined units/second |
| acc | Acceleration limit | Units: User-defined units/second$^2$ |
| dec | Deceleration limit | Units: User-defined units/second$^2$ |
| jrk | Jerk limit (maximum rate of change of acceleration) | Units: User-defined units/second$^3$ |
| ampArray | Array of one or more AmpObj (which have already been initialized) | Units: None |

### TrajectoryInitialize (positions As Double, velocities As Double, times As Integer, lowWater As Integer)

Description:

Initializes and starts a PVT (Position-Velocity-Time) trajectory move on a Linkage Object. The linked amplifiers will queue up the PVT segments and find the best-fit curve for each set of three PVT segments.

Parameters:

| | | |
|---|---|---|
| Positions | A two dimensional array of positions declared as numOfSegments, numOfAxis | Units: Counts |
| Velocities | A two dimensional array of velocities declared as numOfSegments, numOfAxis | Units: User defined units/second |
| Times | A single dimensional array of delta time values representing times from 1 to 255 milliseconds. A time value of zero indicates to the amplifier that the trajectory is complete. The length of this array, as of the position and velocity arrays, must be equal to the number of segments | Units: mS |
| lowWater | This is the level of PVT segments left in the Copley Motion Object buffer on the PC at which CMO generates an event requesting more PVT segments. This number must be less than the number of segments | Units: None |

### TrajectoryAdd (positions As Double, velocities As Double, times As Integer, lowWater As Integer)

Description:

This method adds PVT segments to the CMO PVT buffer waiting to be sent to the amplifier. (Note: this buffer is used in addition to the 32-deep PVT buffer on the amplifier.) This method is typically used within the handler for the TrajectoryEventNotify event handler such that new PVT segments can be sent to the amplifier when the CMO PVT trajectory generator reaches the lowWater level.

Parameters:

| | | |
|---|---|---|
| Positions | A two dimensional array of positions declared as numOfSegments, numOfAxis | Units: Counts |
| Velocities | A two dimensional array of velocities declared as numOfSegments, numOfAxis | Units: User defined units/second |
| Times | A single dimensional array of delta time values representing times from 1 to 255 milliseconds. A time value of zero indicates to the amplifier that the trajectory is complete. The length of this array, as of the position and velocity arrays, must be equal to the number of segments | Units: mS |
| lowWater | This is the level of PVT segments left in the Copley Motion Object buffer on the PC at which CMO generates an event requesting more PVT segments. This number must be less than the number of segments | Units: None |

### WaitMoveDone (timeout As Long)

Description:

Wait until the multi axis move is complete. This method is blocking. When called, it will not return until either the event occurs, the timeout expires, a fault occurs, or a move is aborted. If a timeout occurs, CMO will report the timeout by throwing an exception.

Parameters:

| | | |
|---|---|---|
| timeout | The timeout for the wait. If < 0, then wait indefinitely | Units: mS |

### HaltMove ()

Description:

Halt the current move. The exact type of halt can be programmed individually for each axis using the AmpObj property HaltMode.

Parameters:

None

### CreateEvent (mask As CML_LINK_EVENT, condition As CML_EVENT_CONDITION) As EventObj

Description:

Creates an instance of the EventObj that monitors amplifier events and sets them up using the specified parameters.

Parameters:

| | | |
|---|---|---|
| mask | A bit-mapped value that indicates which events are to be | Units: None |

| | |
|---|---|
| | monitored |
| condition | The trigger condition for the events that will result in the event callback method being called (e.g. all events in the mask). See |
| eventObject | The EventObj instance created by this method |

### CML_LINK_EVENT

| Value | Bit | Description |
|---|---|---|
| LINKEVENT_MOVEDONE | 0 | Set when all amplifiers attached to this linkage have finished their moves and have settled in to position at the end of the move. Cleared when a new move is started on any amplifier. |
| LINKEVENT_TRJDONE | 1 | Set when all amplifiers attached to the linkage have finished their moves, but have not yet settled into position at the end of the move. Cleared when a new move is started on any amplifier. |
| LINKEVENT_NODEGUARD | 2 | A node guarding (or heartbeat) error has occurred. This indicates that one of the amplifiers failed to respond within the expected amount of time for either a heartbeat or node-guarding message. |
| LINKEVENT_FAULT | 4 | A latching fault has occurred on one of the amplifiers attached to this linkage. |
| LINKEVENT_ERROR | 5 | A non-latching error has occurred on one of the amplifiers. |
| LINKEVENT_POSWARN | 6 | One of the amplifiers is reporting a position-warning event. |
| LINKEVENT_POSWIN | 7 | One of the amplifiers is reporting a position window event. |
| LINKEVENT_VELWIN | 8 | One of the amplifiers is reporting a velocity window event. |
| LINKEVENT_DISABLED | 9 | One of the amplifiers is currently disabled. |
| LINKEVENT_POSLIM | 10 | The positive limit switch of one or more amplifier is currently active. |
| LINKEVENT_NEGLIM | 11 | The negative limit switch of one or more amplifier is currently active. |
| LINKEVENT_SOFTLIM_POS | 12 | The positive software limit of one or more amplifier is currently active. |
| LINKEVENT_SOFTLIM_NEG | 13 | The negative software limit of one or more amplifier is currently active. |
| LINKEVENT_QUICKSTOP | 14 | One of the linkage amplifiers is presently performing a quick stop sequence or is holding in quick stop mode. The amplifier must be disabled to clear this. |
| LINKEVENT_ABORT | 15 | One or more amplifier aborted the last profile without finishing. |
| LINKEVENT_LOWWATER | 31 | The active PVT profile is at or below the low water mark and needs more data points. |

# CHAPTER 6: THE EVENT OBJECT

## Overview

The eventObj allows an application program to be event-driven by having a function called when an event occurs in the amplifier. This eliminates the need for polling for the event. The eventObj is created by calling the CreateEvent method for: AmpObj, LinkageObj, and IOObj. The recommended steps for using the EventObj are as follows:

**1** Declare an EventObj variable:

```
// C#
eventObj xAxisEventObj;
```

```
'VB
Friend WithEvents YAxisEventObj As eventObj
```

**2** Create the event:

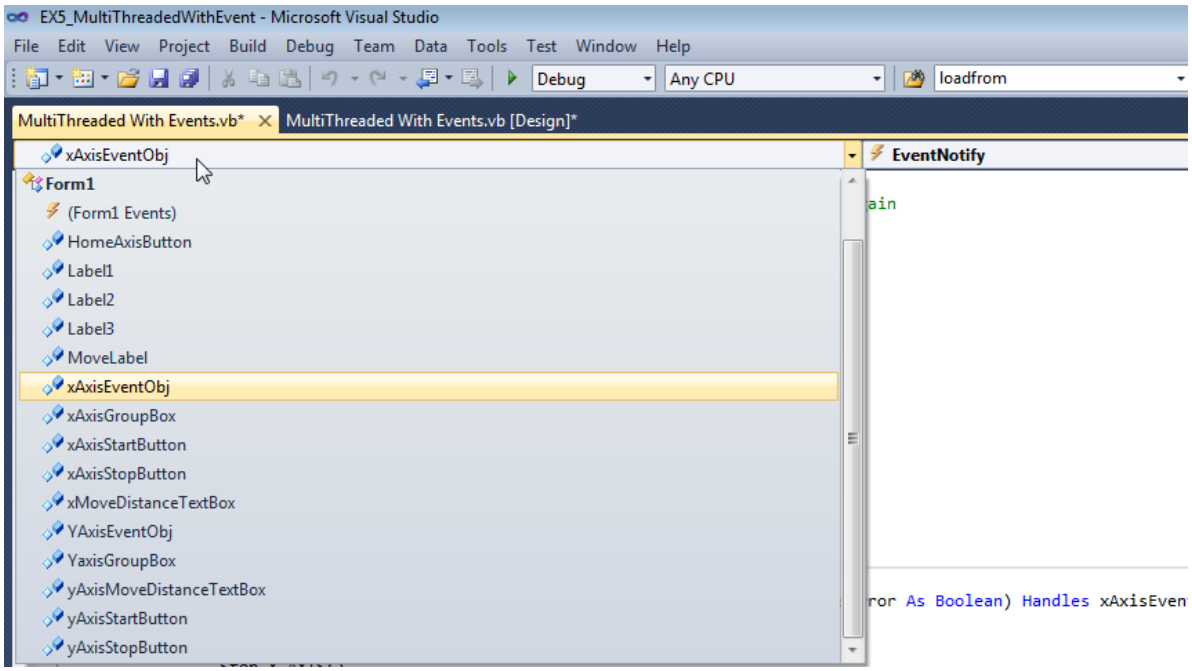
```
// C#
xAxisEventObj = AmpX.CreateEvent(CML_AMP_EVENT.AMPEVENT_MOVE_DONE,
CML_EVENT_CONDITION.CML_EVENT_ANY);
```

```
'VB
xAxisEventObj = AmpX.CreateEvent(CML_AMP_EVENT.AMPEVENT_MOVE_DONE,
CML_EVENT_CONDITION.CML_EVENT_ANY)
```

**3** Register the callback method with the eventObj.

```
// C#
xAxisEventObj.EventNotify += new eventObj.EventHandler(xAxisEventObj_EventNotify);
```

```
' VB
' In order to associate the callback method with the eventObj, select the eventObj
' variable from the variable list in Visual Studio as shown below.  Then, select
' EventNotify from the list on the right.  This will create the callback method.
```




**4** Start the eventObj:

```
' C# and VB
```

```
xAxisEventObj.Start(False, 50000)
```

5 Implement the callback method to handle the event in a manner appropriate with the application.

## Methods

### Start (repeats As Boolean, timeout As Long)

Description:

Starts the event monitor.

Parameters:

| | | |
|---|---|---|
| repeats | Set to true to set up the event monitor to perform a callback each time the event occurs until the event monitor is stopped. Set to false to set up the event monitor to perform a callback on a one-time basis. When set up for repeating events, the event condition must go away, then come back for the event callback to occur again | Units: None |
| timeout | The timeout for the wait. If < 0, then wait indefinitely. Units: milliseconds. If the timeout expires before the event occurs, then the callback routine will be called with its second parameter (hasError) set to true | |

### Stop ()

Description:

Stops the event monitor.

Parameters:

None

### Wait (timeout As Long)

Description:

Wait on the event. This method is blocking. When called, it will not return until either the event occurs, or the timeout expires. If a timeout occurs, CMO will report the timeout in the form of a COM compatible error object.

Parameters:

| | | |
|---|---|---|
| timeout | The timeout for the wait. If < 0, then wait indefinitely | Units: mS |

## Callback

### EventNotify (match As CML_AMP_EVENT, timeout As Boolean)

Description:

Returns the contents of the register that was set up to trigger the event. The timeout variable will be true if the timeout period expired.

Parameters:

| | | |
|---|---|---|
| match | The contents of the register that was set up to trigger the | Units: None |

| | |
|---|---|
| | event |
| timeout | *True* if a timeout or error occurred, *False* otherwise. Should be checked for an error condition before processing the event handling code |

# CHAPTER 7: THE I/O OBJECT

## Overview

The functions described here support I/O devices that comply to the CiA profile DS-401: CANopen Device Profile for Generic I/O Modules.

## Methods

### Initialize (canOpenObj As CANopenObj, nodeId As Integer)

Description:

Initializes the I/O device with the CANOpenObj and the specified node ID.

Parameters:

| | | |
|---|---|---|
| canOpenObj | An instance of a CanOpenObj that has already been initialized | Units: None |
| nodeid | The node ID of the I/O module | Units: None |

### InitializeExt (canOpenObj As CANopenObj, nodeId As Integer, IOSettingsObj As IOSettings)

Description:

Initializes the I/O device with the CANOpenObj and the specified node ID. Also, through the IOsettingsObj parameter, allows the CAN network settings for an I/O module to be set at initialization time. This is necessary if PDO mapping is to be turned off for a particular I/O module.

Parameters:

| | | |
|---|---|---|
| canOpenObj | An instance of a CanOpenObj that has already been initialized | Units: None |
| nodeid | The node ID of the I/O module | Units: None |
| IOsettingsObj | Allows the CAN network settings for an I/O module to be set at initialization time | Units: None |

### CreateEvent (mask As CML_IOMODULE_EVENTS, condition As CML_EVENT_CONDITION) As EventObj

Description:

Creates an instance of the EventObj that monitors I/0 events and sets them up using the specified parameters.

Parameters:

| | | |
|---|---|---|
| mask | A bit-mapped value that indicates which events are to be monitored | Units: None |
| condition | Trigger condition for the events that will result in the callback method being called (e.g. all events in the mask) | Units: None |

### CML_IOMODULE_EVENTS

IOEVENT_AIN_PDO0 = 0x10000

Trigger when any of the first 4 analog inputs generates an event.

IOEVENT_AIN_PDO1 = 0x20000

Trigger when any of the second 4 analog inputs generates an event

IOEVENT_AIN_PDO2 = 0x40000

Trigger when any of the third 4 analog inputs generates an event

IOEVENT_DIN_PDO0 = 0x0001

Trigger when first 64 digital inputs change state.

### SDO_Dnld (index As Integer, sub As Integer, data As Object)

Description:

Downloads data to the IO module via a CAN SDO transfer.

Parameters:

| | | |
|---|---|---|
| index | Index of a CANopen dictionary object | Units: None |
| sub | Sub-index of a CANopen dictionary object | Units: None |
| data | The data that is to be transferred. This data can be one of four types: 8-bit, 16-bit, 32-bit, or String | Units: None |

### SDO_Upld (index As Integer, sub As Integer, data As Object)

Description:

Uploads data from the IO module via a CAN SDO transfer.

Parameters:

| | | |
|---|---|---|
| index | Index of a CANopen dictionary object | Units: None |
| sub | Sub-index of a CANopen dictionary object | Units: None |
| data | The data that is to be transferred. This data can be one of four types: 8-bit, 16-bit, 32-bit, or String | Units: None |

### SDO_DnldExt (index As Integer, sub As Integer, data As Byte, size As Integer)

Description:

Downloads data to the amplifier via a CAN SDO transfer.

Parameters:

| | | |
|---|---|---|
| index | Index of a CANopen dictionary object. | Units: None |
| sub | Sub-index of a CANopen dictionary object | Units: None |
| data | The data that is to be transferred. This data is an array of bytes | Units: None |
| size | The number of bytes of data to be downloaded | Units: None |

### SDO_UpldExt (index As Integer, sub As Integer, data As Byte, size As Integer)

Description:

Uploads data from the amplifier via a CAN SDO transfer.

Parameters:

| | | |
|---|---|---|
| index | Index of a CANopen dictionary object | Units: None |

| | | |
|---|---|---|
| sub | Sub-index of a CANopen dictionary object | Units: None |
| data | The data that is to be transferred. This data is an array of bytes | Units: None |
| size | On entry this gives the max number of bytes of data to be uploaded. On successful return this gives the actual number of bytes received | Units: None |

## ioSettingsObj

### Properties

#### useStandardDinPDO

Type: Boolean
Description: Use the standard digital input PDO object
Units: None
Default: true

#### UseStandardDoutPDO

Type: Boolean
Description: Use the standard digital output PDO object
Units: None
Default: true

#### UseStandardAinPDO

Type: Boolean
Description: Use the standard analog input PDO object
Units: None
Default: true

#### UseStandardAoutPDO

Type: Boolean
Description: Use the standard analog output PDO object
Units: None
Default: true

#### heartBeatPeriod

Type: Short
Description: Configures the heartbeat period used by this IO module to transmit its heartbeat message. If this property is set to zero, then the heartbeat protocol is disabled on this module
Units: mS
Default: 0

### heartbeatTimeout

| | |
|---|---|
| Type: | Short |
| Description: | Additional time to wait before generating a heartbeat error |
| Units: | mS |
| Default: | 0 |

### guardTime

| | |
|---|---|
| Type: | Short |
| Description: | This object gives the time between node-guarding requests that are sent from the network master to this IO module. The IO module will respond to each request with a node-guarding message indicating the internal state of the IO module. If the IO module has not received a node-guarding request within the time period defined by the product of the guard time and the lifeFactor, the IO module will treat this lack of communication as a fault condition |
| Units: | mS |
| Default: | 0 |

### lifeFactor

| | |
|---|---|
| Type: | Short |
| Description: | This property gives a multiple of the guardTime parameter. The IO module expects to receive a node-guarding request within the time period defined by the product of the guard time and the lifetime factor. If the IO module has not received a node-guarding request within this time period, it treats this condition as a fault |
| Units: | None |
| Default: | 3 |

# 7.1: Analog Inputs

## Methods

### Ain16Read (channel As Integer, value As Integer, viaSDO As Boolean)

Description:

Reads a 16-bit analog input.

Parameters:

| | | |
|---|---|---|
| channel | The analog input channel ID | Units: None |
| value | The analog input value read | Units: None |
| viaSDO | If True, read inputs using SDO transfer. If False (default), use most recently received PDO data, if this input is mapped to a transmit PDO and the PDO is active | Units: None |

### AinTrigTypeRead (channel As Integer, trigger As CML_IO_AIN_TRIG_TYPE)

### AinTrigTypeWrite (channel As Integer, trigger As CML_IO_AIN_TRIG_TYPE)

Description:

Reads/writes the analog input trigger type associated with input channel. Use this command to set/get the type of event associated with an analog input.

Parameters:

| | | |
|---|---|---|
| channel | The analog input channel ID | Units: None |
| trigger | The analog input trigger type associated with input channel | Units: None |

### CML_IO_AIN_TRIG_TYPE

IOAINTRIG_UPPER_LIM = 1
: Input above upper limit

IOAINTRIG_LOWER_LIM = 2
: Input below lower limit

IOAINTRIG_UDELTA = 4
: Input changed by more than the unsigned delta amount

IOAINTRIG_NDELTA = 8
: Input reduced by more than the negative delta amount

IOAINTRIG_PDELTA = 16
: Input increased by more than the positive delta

### Ain16LowerLimitRead (channel As Integer, limit As Integer)

### Ain16LowerLimitWrite (channel As Integer, limit As Integer)

Description:

Reads/writes the analog input lower limit value as a 16-bit integer. The lower limit defines the value at which an interrupt will be generated if it is enabled.

Parameters:

| | | |
|---|---|---|
| channel | The analog input channel ID | Units: None |
| limit | The analog input lower limit value | Units: None |

### Ain16NegativeDeltaRead (channel As Integer, delta As Integer)

### Ain16NegativeDeltaWrite (channel As Integer, delta As Integer)

Description:

Reads/writes the analog input negative delta value as a 16-bit integer. The negative delta defines the amount of change at which an interrupt will be generated if it is enabled.

Parameters:

| | | |
|---|---|---|
| channel | The analog input channel ID | Units: None |
| delta | The analog input negative delta value | Units: None |

### Ain16PositiveDeltaRead (channel As Integer, delta As Integer)

### Ain16PositiveDeltaWrite (channel As Integer, delta As Integer)

Description:

Reads/writes the analog input positive delta value as a 16-bit integer. The positive delta defines the amount of change at which an interrupt will be generated if it is enabled.

Parameters:

| | | |
|---|---|---|
| channel | The analog input channel ID | Units: None |
| delta | The analog input positive delta value | Units: None |

### Ain16UnsignedDeltaRead (channel As Integer, delta As Integer)

### Ain16UnsignedDeltaWrite (channel As Integer, delta As Integer)

Description:

Reads/writes the analog input unsigned delta value as a 16-bit integer. The unsigned delta defines the amount of change at which an interrupt will be generated if it is enabled.

Parameters:

| | | |
|---|---|---|
| channel | The analog input channel ID | Units: None |
| Delta | The analog input unsigned delta value | Units: None |

### Ain16UpperLimitRead (channel As Integer, limit As Integer)

### Ain16UpperLimitWrite (channel As Integer, limit As Integer)

Description:

Reads/writes the analog input upper limit value as a 16-bit integer. The upper limit defines the value at which an interrupt will be generated if it is enabled.

Parameters:

| | | |
|---|---|---|
| channel | The analog input channel ID | Units: None |
| Limit | The analog input upper limit value | Units: None |

## Properties

### AinIntEnable

| | |
|---|---|
| Type: | Boolean |
| Description: | Current setting of the global interrupt enable for analog inputs |
| Units: | None |
| Default: | False |

# 7.2: Analog Outputs

## Methods

### Aout16Write (channel As Integer, value As Integer, viaSDO As Boolean)

Description:

Writes to a 16-bit analog output.

Parameters:

| | | |
|---|---|---|
| channel | The analog input channel ID | Units: None |
| value | The value to write | Units: None |
| viaSDO | If true, the outputs will be written using SDO messages. If false (default), then a PDO will be used if possible | Units: None |

### AoutErrModeRead (channel As Integer, mode As Boolean)

### AoutErrModeWrite (channel As Integer, mode As Boolean)

Description:

Reads/writes the analog output error mode. If the error mode is True, then the analog output will change its value to the programmed error value in the case of a device failure. If False, a device failure will not cause a change in the analog output value.

Parameters:

| | | |
|---|---|---|
| channel | The analog output channel ID | Units: None |
| mode | The analog output error mode | Units: None |

### Aout16ErrorValueRead (channel As Integer, error As Integer)

### Aout16ErrorValueWrite (channel As Integer, error As Integer)

Description:

Reads/writes the analog out error value.

Parameters:

| | | |
|---|---|---|
| channel | The analog input channel ID | Units: None |
| error | The analog output error value | Units: None |

# 7.3: Digital Inputs

## Methods

### Din8Read (group As Integer, value As Integer, viaSDO As Boolean)

Description:

Reads a group of 8 digital inputs.

Parameters:

| | | |
|---|---|---|
| group | Identifies which group of 8 to read | Units: None |

| | | |
|---|---|---|
| value | The value of the input | Units: None |
| viaSDO | If true, read inputs using the SDO transfer. If false (default) use the most recently received PDO data if this input group is mapped to a transmit PDO and the PDO is active | Units: None |

### Din8MaskAnyRead (group As Integer, mask As Integer)

### Din8MaskAnyWrite (group As Integer, mask As Integer)

Description:

Reads/writes the 'any transition' interrupt mask setting for a group of 8 digital inputs. For each input in the group, a value of 1 enables interrupts on any change, and a value of 0 disables the interrupt.

Parameters:

| | | |
|---|---|---|
| group | Identifies which group of 8 inputs to read/write | Units: None |
| mask | The 'any transition' interrupt mask | Units: None |

### Din8MaskHigh2LowRead (group As Integer, mask As Integer)

### Din8MaskHigh2LowWrite (group As Integer, mask As Integer)

Description:

Reads/writes the 'high to low' interrupt mask setting for a group of 8 digital inputs. For each input in the group, a value of 1 enables interrupts on a high to low transition, and a value of 0 disables the interrupt.

Parameters:

| | | |
|---|---|---|
| group | Identifies which group of 8 inputs to read/write | Units: None |
| mask | The 'high to low' interrupt mask | Units: None |

### Din8MaskLow2HighRead (group As Integer, mask As Integer)

### Din8MaskLow2HighWrite (group As Integer, mask As Integer)

Description:

Reads/writes the 'low to high' interrupt mask setting for a group of 8 digital inputs. For each input in the group, a value of 1 enables interrupts on a low to high transition, and a value of 0 disables the interrupt.

Parameters:

| | | |
|---|---|---|
| group | Identifies which group of 8 inputs to read/write | Units: None |
| mask | The 'low to high' interrupt mask | Units: None |

# Properties

### DinIntEnable

| | |
|---|---|
| Type: | Boolean |
| Description: | Current setting of the global interrupt enable of digital inputs |
| Units: | None |

Default: False

# 7.4: Digital Outputs

## Methods

### Dout8Write (group As Integer, value As Integer, viaSDO As Boolean)

Description:

Writes a group of 8 digital outputs.

Parameters:

| | | |
|---|---|---|
| group | Identifies which group of outputs to write | Units: None |
| value | Value to write to group | Units: None |
| viaSDO | If true, outputs are written using SDO message. If false (default), a PDO is used if possible | Units: None |

### Dout8ErrModeRead (group As Integer, mode As Integer)

### Dout8ErrModeWrite (group As Integer, mode As Integer)

Description:

Reads/writes the current error mode setting of a group of 8 digital outputs. For each output in the group, a value of 1 will cause the output to take its programmed error value on a device failure. Setting the mode to 0 will cause the output to hold its programmed value on failure.

Parameters:

| | | |
|---|---|---|
| group | Identifies the group of outputs to read/write | Units: None |
| mode | The current error mode setting of a group of 8 digital outputs | Units: None |

### Dout8ErrValueRead (group As Integer, error As Integer)

### Dout8ErrValueWrite (group As Integer, error As Integer)

Description:

Reads/writes the current error value setting for a group of 8 digital outputs. Error values define the state of the output if a device failure occurs. The error value will only be set for those output pins that have an error mode set to 1. Those with error mode set to zero will not be changed by a device failure.

Parameters:

| | | |
|---|---|---|
| group | Identifies the group of outputs to read/write | Units: None |
| mode | The current error value setting for a group of 8 digital outputs | Units: None |

# CHAPTER 8: COPLEYMOTIONLIBRARY OBJECT

## Properties

### VersionString

Type: String

Description: The version number of Copley Motion Libraries (CML) used by CMO.

Units: None

Default: None

### DebugLevel

Type: Integer

Description: Debug message level. Setting this property greater than zero results in debug messages being written to a log file (see table below). The value set for DebugLevel will result in that level, plus all lower levels being logged. Therefore, if DebugLevel is set to 3, then levels 3, 2, and 1 are logged. Setting this property to zero will result in the log file being closed.

| Debug Level | Description |
|---|---|
| 0 | Debug logging is disabled |
| 1 | Log serious errors only |
| 2 | Log warning messages and errors |
| 3 | Log debugging info |
| 4 | Not defined |
| 5 | Log most CAN messages (some common messages are filtered out |
| 6 | Log all CAN messages |
| 99 | Log everything |

Units: None

Default: 0 (no messages)

### MaxLogSize

Type: Integer

Description: Maximum log file size. Once the log file exceeds MaxLogSize, it is renamed *logfilename*.bak, and a new log file is started. Old backup log files are overwritten.

Units: None

Default: 1,000,000 bytes

### LogFileName

Type: String

Description: Name of the debug message log file. This file is used to log debug messages. The file will be created (or truncated if it already exists) when the

first message is written to the file. Note that the debug level must be set > 0 for any messages to be written.

Units: None

Default: "cml.log"

CMO Programmer's Guide
P/N 16-01041
Revision 01
June 2015