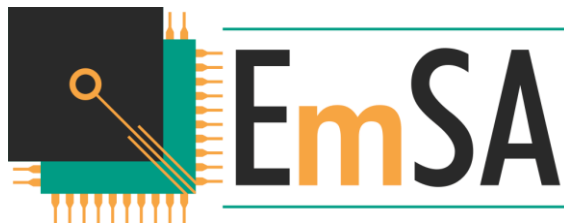


CANopen Magic User Manual

Manual Revision 3.01



Information in this document is subject to change without notice and does not represent a commitment on the part of the manufacturer. The software described in this document is furnished under license agreement or nondisclosure agreement and may be used or copied in accordance with the terms of the agreement. It is against the law to copy the software on any medium except as specifically allowed in the license or nondisclosure agreement. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than the software purchaser's use, without prior written permission.

Every effort was made to ensure the accuracy in this manual and to give appropriate credit to persons, companies and trademarks referenced herein.

© Embedded Systems Academy, Inc. 2002-2025, All Rights Reserved

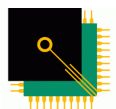
Microsoft® and Windows™ are trademarks or registered trademarks of Microsoft Corporation.

PC® is a registered trademark of International Business Machines Corporation.

CANopen® is a registered trademark of CAN in Automation User's Group.

ESAcademy® is a registered trademark of Embedded Systems Academy, Inc.

For support contact support@esacademy.com



EMBEDDED
SYSTEMS
ACADEMY

CANopen Magic was developed by Embedded Systems Academy. Embedded Systems Academy provides training and consulting services, specializing in CAN and CANopen. For more information visit

www.esacademy.com

Contents

Contents	3
About This Manual.....	5
Chapter 1 – Introduction	6
1.1 About CANopen	6
1.2 About CANopen Magic	6
1.3 Obtaining Compatible CAN Interfaces	7
Chapter 2 – Installation and Setup	8
2.1 Installation.....	8
Additional Step for PCAN Dongle Users	8
2.2 Setup	8
Chapter 3 – Evaluation Quick Start.....	10
3.1 Introduction	10
3.2 Connection.....	10
3.3 Trace.....	11
3.4 Start Simulated Node	11
3.5 Read From Node	12
3.6 Network Overview.....	13
3.7 PDO Configuration	14
3.8 Trace Export	14
3.9 Trace Analysis	15
Chapter 4 – Core Functionality	17
4.1 User Interface Overview	17
4.2 Network Management.....	19
4.3 Reading from a Node.....	19
4.4 Writing to a Node.....	20
4.5 Trace.....	22
4.6 Network Overview.....	23
4.7 Transmit CAN Messages.....	23
4.8 Preferences	24
Chapter 5 – Advanced Functionality.....	25
5.1 Trace Analysis, Filtering, Recording and Export.....	25
5.2 PDO Configuration	28
5.3 Network Description	29
5.4 Process Data Display	32
5.5 Layer Setting Services	33
5.6 Read/Write Node Configurations	34
5.7 SDO Channels	35
5.8 Scripting.....	36
5.9 CANopen Manager.....	36
5.10 Log Player	37
5.11 Device Profile Support	37
5.12 Node Setup	37
5.13 Node Selection.....	38
5.14 Network Diagrams.....	38
5.15 Node Object Dictionary	39
5.16 Trace Linking	40
5.17 Charting.....	41

Chapter 6 – Simulation	47
6.1 Overview	47
6.2 Adding Simulation Nodes	47
6.3 Process Data Display	48
6.4 Dynamic Object Dictionary Simulation	50
Chapter 7 – Command Line	51
7.1 Overview	51
7.2 Interrogation	51
7.3 Tasks	52
NMT	52
SDODOWNLOAD	53
SDOUPLOAD	53
7.4 Examples	54
7.5 CANopen Magic Pro Library	54
7.6 GUI Command Line	55
Chapter 8 – Trace Filtering Scripts	56
8.1 Overview	56
8.2 Editing	56
8.3 Debugging	56
8.4 Objects	56
Message	56
8.5 Functions	57
OnMessageReceiving	57
OnMessageReceived	57
8.6 Parameters	57
8.7 Example	57
Chapter 9 – Trace Interpreting Scripts	59
9.1 Overview	59
9.2 Editing	59
9.3 Debugging	59
9.4 Objects	59
Message	59
InterpretedCANMessage	60
9.5 Functions	60
OnMessagePreInterpret	60
OnMessageInterpret	61
9.6 Parameters	61
9.7 Example	61
Chapter 10 – General Purpose Scripts	63
10.1 Overview	63
10.2 Utility Functions	64
Read	64
ReadKey	64
App.SetExitCode	64
10.3 Namespaces and Classes	64
10.4 Function Declarations	66
10.5 Class Reference	66

About This Manual

This manual follows some set conventions with the aim of making it easier to read. The following conventions are used:

0x	Hexadecimal (base 16) values are prefixed with "0x".
<i>italic text</i>	Replace the text with the item it represents
[]	Items inside [and] are optional
a b	a OR b may be used
...	One or more items may go here.

This manual frequently uses CANopen terminology as defined by the CANopen standard CiA301 (see www.can-cia.org for more info). Readers that are not yet familiar with all the CANopen terms may want to consider reading a book like www.canopenbook.com or the official standard to update their knowledge on CANopen technology and terminology.

Chapter 1 – Introduction

1.1 About CANopen

CANopen is a higher layer protocol that runs on a CAN network. The CAN specification defines only the physical and data link layers in the ISO/OSI 7-layer Reference Model. This means that only the physical bus and the CAN message format is defined, but not how the CAN messages should be used. CANopen provides an open and standardized but customizable description of how to transfer data of different types between different CAN nodes. This allows off the shelf CANopen compliant nodes to be purchased and plugged into a network with the minimum of effort. It also can be used in place of an in-house proprietary higher layer protocol development.

The development of CANopen is supervised by the CAN in Automation User's Group and is being turned into an international standard. Use of CANopen does not require the payment of any royalties and the specification may be expanded or altered to suit if closed networks are being developed.

Typical applications for CANopen include:

- Commercial Vehicles
- Medical Equipment
- Maritime Electronics
- Building Automation
- Light Rail Systems

1.2 About CANopen Magic

CANopen Magic is a professional grade CANopen development tool, indispensable for assisting in the development and debugging of CANopen based networks. Using CANopen Magic, a CANopen network may be monitored, configured and analyzed and CANopen nodes can be tested and integrated. Timings of message transmissions can be observed, allowing comparison against network specifications. Nodes may be quickly and easily configured and test messages generated with a minimum of effort. New nodes may be simulated individually and as part of a network of real nodes and/or simulated nodes. Network traffic can be traced, logged and replayed.

There are three versions of the application available, each with different feature sets. Please visit canopenmagic.com for a comparison table. This manual covers the superset of features, so some sections may not apply to the version you are using.

1.3 Obtaining Compatible CAN Interfaces

CANopen Magic supports CAN interfaces from several vendors. At the time of writing the following are supported:

- ⤴ PEAK-System Technik CAN Interfaces
- ⤴ Kvaser CAN Interfaces
- ⤴ VSCOM CAN to Ethernet bridges

Embedded Systems Academy recommends the PCAN-USB Pro interface from PEAK as it provides timestamps with 1us accuracy in a metal casing with two CAN channels available.

Chapter 2 – Installation and Setup

2.1 Installation

To install the software run the installer and it will guide you through the steps. In addition you will need to install the drivers supplied with your CAN interface and the Microsoft .NET Framework 4.0 or later. If you don't have the .NET framework the installer will automatically download and install it for you.

When prompted choose to install the PEAK professional driver. Even if you are not using PEAK CAN interfaces this driver is still required.

You will need the activation code provided when you purchased the software, and this is entered into the installer when prompted.

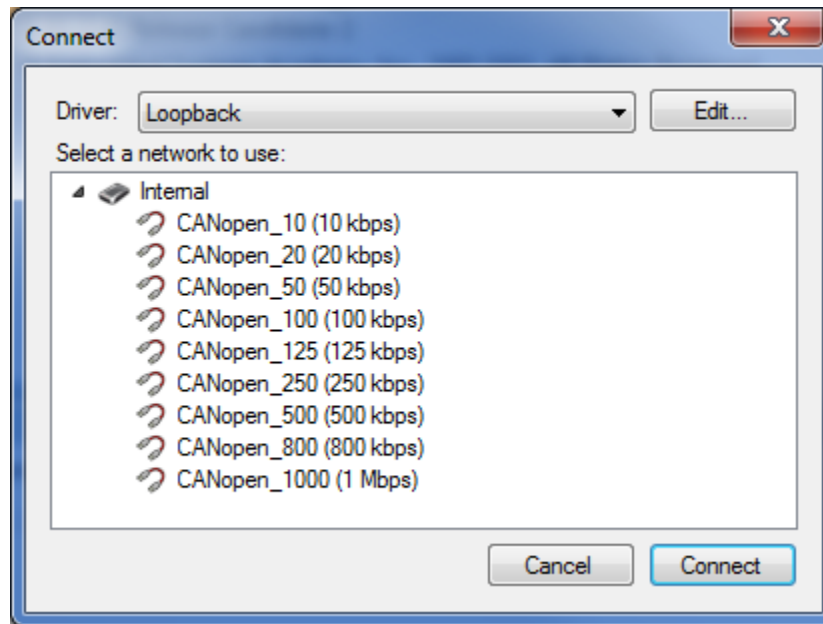
Additional Step for PCAN Dongle Users

Before using the PCAN Dongle interface with CANopen Magic, it must be removed from the setup window of the PCANView Dongle software. To do this complete the following steps:

- Start PCANView Dongle from the Start Menu
- Select the PCAN Dongle in the Available CAN Hardware section
- Click on "Delete"
- Click on "OK"
- Close PCANView Dongle

2.2 Setup

When you run CANopen Magic it is not connected to any CAN interface. Before the application can send or receive it must be connected. To do this click on the connection button on the toolbar or choose Connect... from the Network menu. A connection window will open.



Choose the CAN interface vendor from the drop down list and then choose the network or bus speed to use.

Loopback is a simulation CAN bus that allows you to use the functionality of the software without a CAN interface. This is useful when running simulated nodes or evaluating the software.

For some CAN interfaces the available networks or bus speeds can be edited. If this option is available then the Edit... button will be enabled. Click on the button to configure the CAN interface options.

Choose the network you wish to use and click on Connect.

If you are using a PEAK CAN interface then you can run PCAN-View, PCAN-Explorer or a custom application at the same time as CANopen Magic. Just choose the same CAN interface and network in all applications and they will be able to "see" each other's CAN messages.

Chapter 3 – Evaluation Quick Start

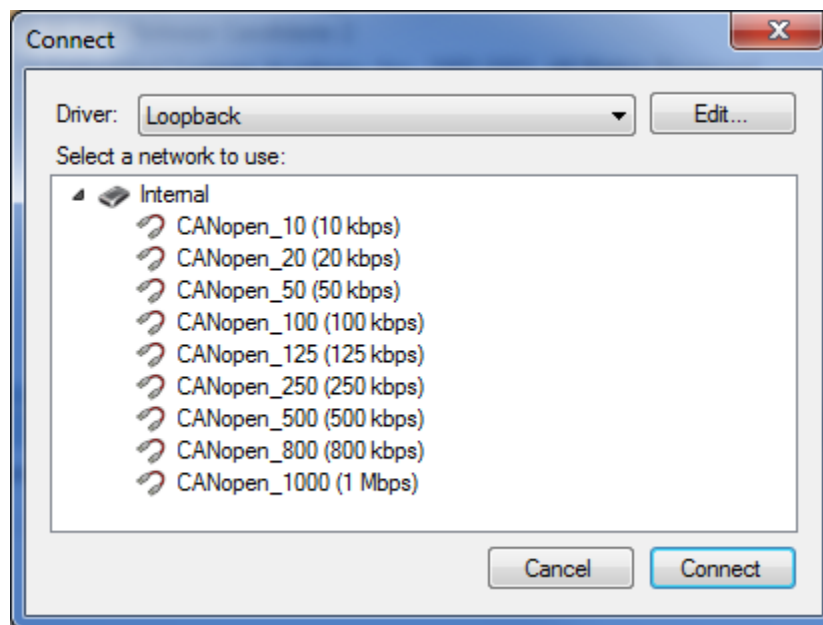
3.1 Introduction

This quick start guide introduces key features of software in a step by step manner and can be followed by users of the evaluation and ultimate versions. It can be followed by users of other versions if they have a real CANopen node.

3.2 Connection

When the application is started it is not connected to a CAN interface. For this quick start guide we will use the loopback interface, which is an internal CAN bus simulation.

Open the connection window by clicking on the Connect toolbar icon or choose Connect... from the Network menu.

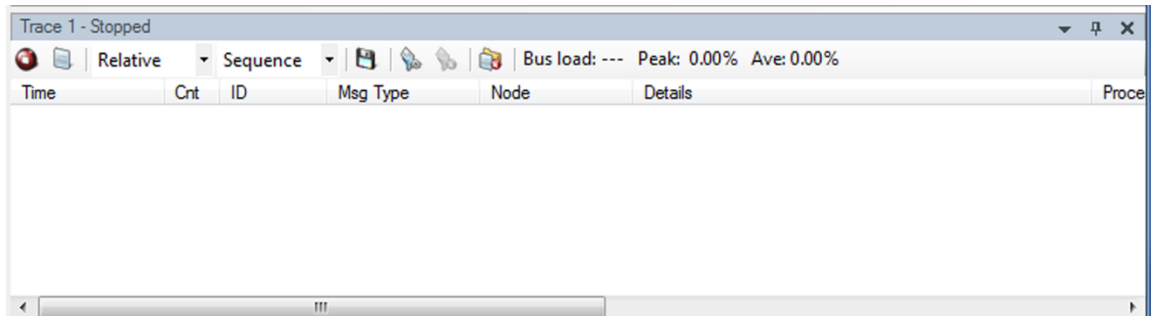


Choose Loopback for the driver and then CANopen_1000 for the network. Then click on Connect. The bottom of the main window will now show the current connection.

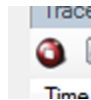


3.3 Trace

The trace window should already be displayed at the bottom of the main window. If it isn't choose Trace... from the Add menu.



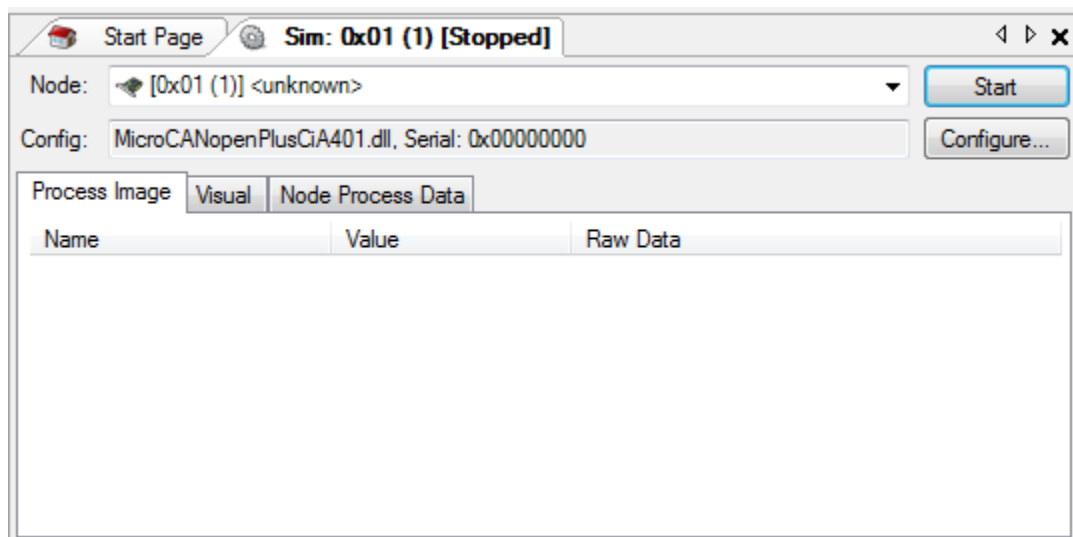
Start the trace window recording by clicking on the record button



The trace window will now show the messages on the CAN bus.

3.4 Start Simulated Node

Add a simulation node by clicking on the simulation node toolbar icon or choosing New Simulation Node... from the Simulation menu. This will open a simulation node window which defaults to simulating an I/O node based on the MicroCANopen Plus communication stack.



Click on the Start button to start the simulated node. At this point the Process Image tab will show the contents of the object dictionary and the trace window will show the bootup messages from node id 0x01.

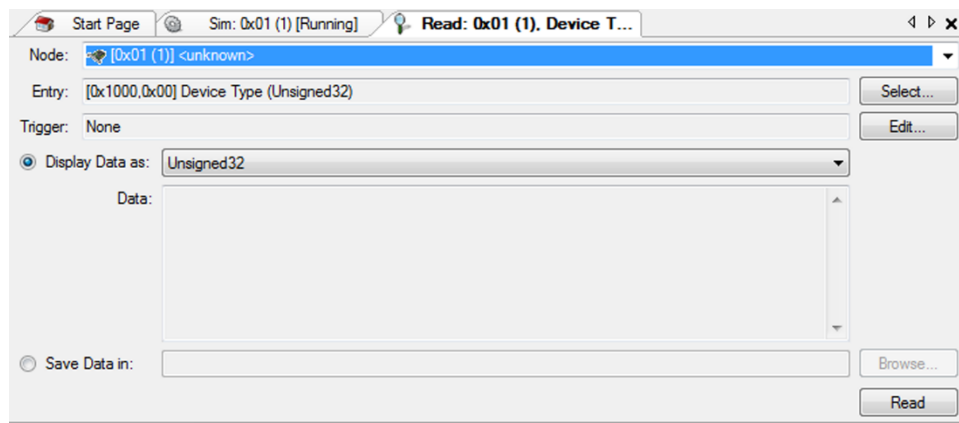
Trace 1 - Running

Relative Sequence Bus load: 0.00% Peak: 0.06% Ave: 0.01%

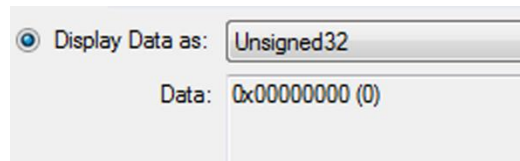
Time	Cnt	ID	Msg Type	Node	Details
0:00:01:40.6822...	1	0x701	Bootup	[0x01 (1)] <unkno...	
0:00:00:00.0040...	1	0x081	Emergency	[0x01 (1)] <unkno...	[0x0000 (0)] Error reset or no error

3.5 Read From Node

Open a read window by clicking on the read toolbar icon or choosing Read From Node... from the Add menu. A read window will open. The default configuration is to read the device type of node id 0x01.



Click on the read button to read the device type of the node. The read window will show the 32-bit device type value in hexadecimal and the trace window will show the expedited SDO access.



0:00:03:15.8608...	1	0x701	Bootup	[0x01 (1)] <unkno...	
0:00:00:00.0010...	1	0x081	Emergency	[0x01 (1)] <unkno...	[0x0000 (0)] Error reset or no error
0:00:02:07.6048...	1	0x601	SDO Initiate Uplo...	[0x01 (1)] <unkno...	[0x1000,0x00] Device Type
0:00:00:00.0010...	1	0x581	SDO Upload Res...	[0x01 (1)] <unkno...	expedited

3.6 Network Overview

Open the network overview window by choosing Network Overview... from the View menu. Click on the Scan Network button to look for nodes on the network. Our single simulation node will be found.

Node	NMT Status	Device Type	Error Register	Vendor ID	Product Code	Revision Number	Serial Number	Last Emergency
[0x01 (1)] <unkno...	Unknown	[0x0000,0x00] ...	[0x00000004] (4)	[0x01455341 (21...	[0x11223344] (28...	[0x22334455] (57...	[0x00000000] (0)	Unknown

The NMT status will be shown as unknown because this feature relies on seeing a bootup, heartbeat or node guarding messages. We can turn heartbeats on by clicking on the Set Heartbeats button.

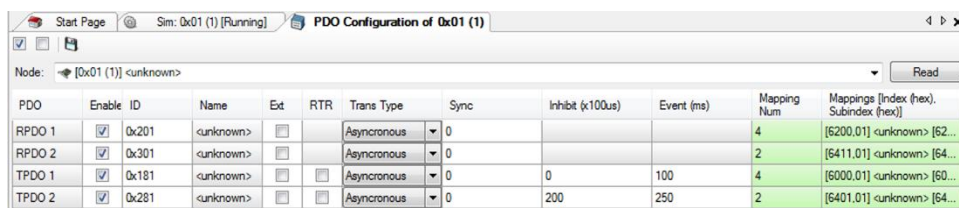
You will now see a heartbeat message being transmitted once a second from the node showing that it is in pre-operational state. The network overview window will also show the node as being in pre-operational state.

	NMT Status	Dev
1kno...	Pre-operational	[0x0

3.7 PDO Configuration

Open the PDO configuration window by clicking on the PDO toolbar icon or choosing PDO Configuration... from the Add menu. The window will open.

Click on the Read button to read the current PDO configuration of the node. The table will fill with entries as a result of interrogating the node. The trace window will show a large number of SDO transfers as the node was read.



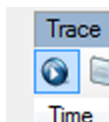
PDO	Enable	ID	Name	Ext	RTR	Trans Type	Sync	Inhibit (x100us)	Event (ms)	Mapping Num	Mappings [Index (hex), Subindex (hex)]
RPDO 1	<input checked="" type="checkbox"/>	0x201	<unknown>	<input type="checkbox"/>		Asynchronous	0			4	[6200.01] <unknown> [62...
RPDO 2	<input checked="" type="checkbox"/>	0x301	<unknown>	<input type="checkbox"/>		Asynchronous	0			2	[6411.01] <unknown> [64...
TPDO 1	<input checked="" type="checkbox"/>	0x181	<unknown>	<input type="checkbox"/>	<input type="checkbox"/>	Asynchronous	0	0	100	4	[6000.01] <unknown> [60...
TPDO 2	<input checked="" type="checkbox"/>	0x281	<unknown>	<input type="checkbox"/>	<input type="checkbox"/>	Asynchronous	0	200	250	2	[6401.01] <unknown> [64...

The PDO settings can be changed by clicking in the cells of the table.

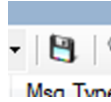
3.8 Trace Export

This section shows you how to export the contents of the trace window to a CSV file. This is useful when sharing debugging information with colleagues, engineers at other companies or documenting node and network testing.

Stop the trace recording by clicking on the stop button. For performance reasons the export feature is only enabled when the trace recording is stopped. For other options please see the advanced functionality chapter of this manual.



Click on the export button in the toolbar of the trace window.



Choose a location to save the CSV file.

Hint: If process data has been configured it will appear in the export (and trace window) looking like this:

Temperature: 0x1122 (4386)

To extract the decimal value for graphing, etc. use the following Excel formula:

`=MID(J6,SEARCH("(",J6)+1,SEARCH(")",J6)-SEARCH("(",J6)-1)`

3.9 Trace Analysis

The final step in this quick start guide shows you how to analyze the contents of the trace buffer.

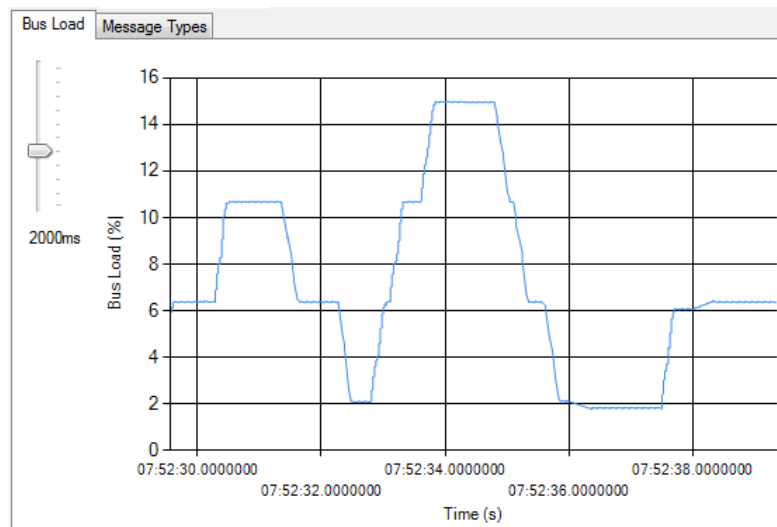
The trace recording should be stopped already (from the previous step). If it isn't stop it now by clicking on the stop button in the trace window toolbar. Analysis of the trace buffer can only take place when recording is stopped.

Open the Trace Analysis window by choosing Trace Analysis... from the View menu. The window will open.

At the top of the window choose the trace buffer to analyze. If you have one trace window open then this should be "Trace 1".

Clicking on the tabs will show different representations of the messages in the trace buffer. Use the controls under each tab to adjust the view as desired.

The output of the analysis can be exported as an image by clicking on the export button in the toolbar of the trace analysis window.



The bus load graph uses absolute timestamps. If the trace window is also set to absolute timestamps then the two windows can be cross-referenced.

Calculation of the bus load can vary depending on the period that the calculation is performed over. The default is 2000ms. Drag the slider to adjust the period and the graph will be updated to reflect the new analysis. This can be useful for observing long-term trends and short-term spikes.

Move the pointer over the graph and click somewhere on the line. The trace window will scroll to show the message that was transmitted on the bus at that moment in time. Also notice that the graph will show the exact time and busload at that point on the line.

With the pointer over the graph use the mouse wheel to zoom in and out for a more detailed view. View of the entire graph can always be obtained by clicking on the zoom reset button.

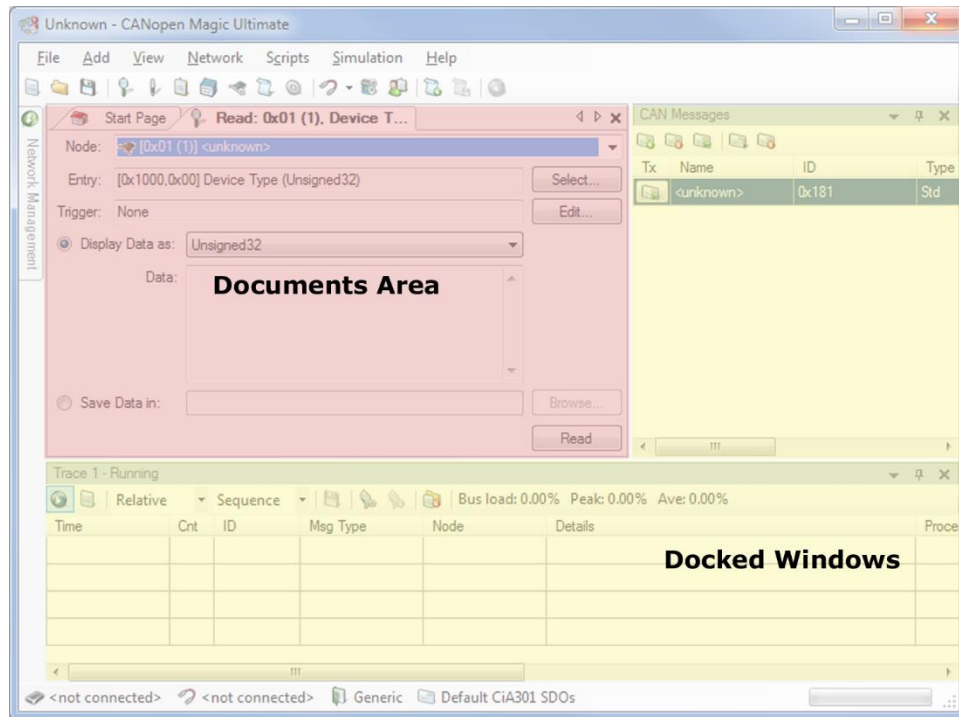


The trace analysis window is updated whenever the trace recording is stopped.

Chapter 4 – Core Functionality

4.1 User Interface Overview

The user interface is divided into four areas, menus/toolbars, status bar, documents and docked windows.



All windows in the application are divided into two types – document or dockable. Documents appear in the central document area of the user interface and can be switched between using the tabs above each document. Dockable windows are attached (docked) to one side of the main window and can be moved outside of the main window if desired.

Document windows can be reordered by holding down the left mouse button on the tab and dragging it left or right.

If too many documents are open then the tabs can be scrolled to view them all by clicking on the arrows to the right of the tabs.

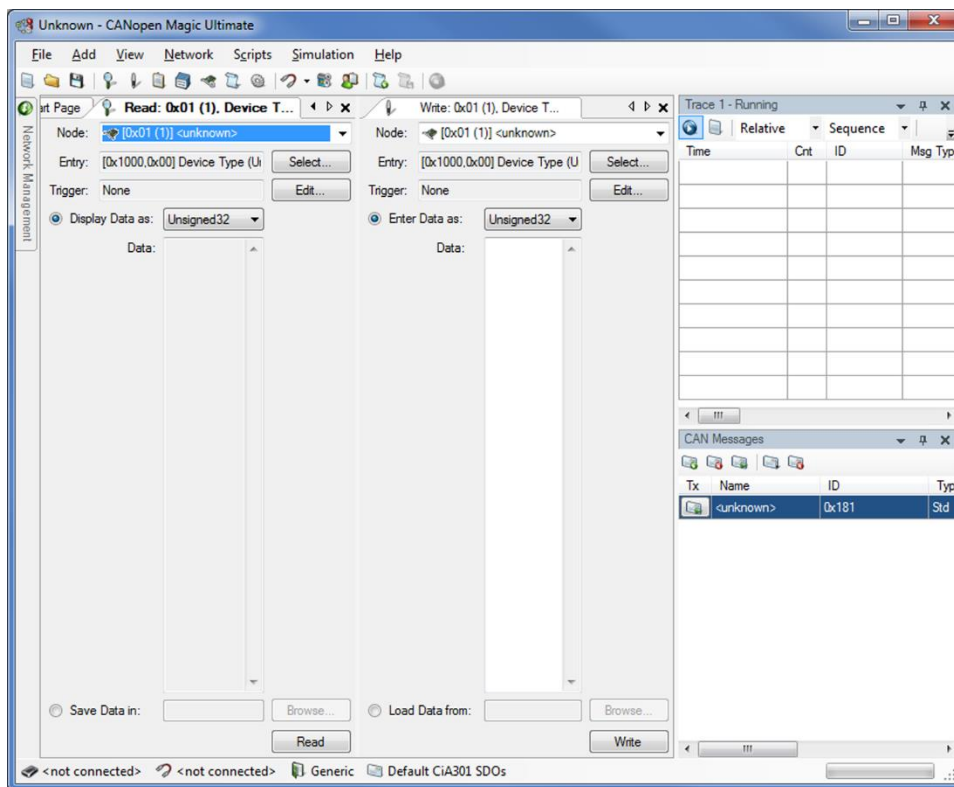
Some documents, such as read and write to a node, can be cloned. This creates a duplicate of the document with the same settings. To clone a document select the document and choose Clone Selected Tab from the Add menu. Alternatively right-click on the tab and choose Clone This Tab.

The document area can be subdivided creating subgroups of tabs. To do this hold down the left mouse button over a tab and drag towards the center of the main window. A group of small blue symbols will appear. Move the pointer over one of these symbols and release the mouse button.

The dockable windows have a large amount of flexibility in how the user interface can be arranged. For example hold down the left mouse button with the pointer over a docked window title bar. Drag towards the center of the main window and a group of small blue symbols will appear. Move the pointer over one of these symbols and release the mouse button. The window will then appear in the new position.

If you release the mouse button when the pointer is elsewhere then the window will be split out of the main window. It can then be placed on a second monitor for example.

The following screenshot gives a taste of how flexible the user interface can be by positioning read and write windows side-by-side and moving the trace window out of the way.



Dockable windows can be hidden. When the application is started the network management window is also opened but it is hidden on the left side and shown as a tab.

Click on the network management tab to pop open the window. Click elsewhere to hide it again.

When the window is open you can interact with it just like any other window. Clicking on the small pin icon at the top left of the window causes the window to stay open, and it can then be docked like any other dockable window.



Clicking on the pin icon in any dockable window hides it to one side.

The default user interface layout can be restored by choosing New Project from the File menu or clicking on the new project toolbar icon.

Any table column can be hidden by right-clicking on the column headers in a table. A menu will appear allowing selection of which columns should be shown. One column must always be shown.

Some tabbed windows can be cloned. This creates another copy of the window with the same settings. To clone a window right-click on the tab and choose Clone this Tab or choose Clone Current Tab from the Add menu.

4.2 Network Management

When creating a new project the network management window is opened and hidden on the left side of the main window. Click on it's tab to expand the window. If you have closed the window then it can be reopened by choosing Network Management... from the View menu.

This window allows you to send NMT messages to nodes on the network. Choose a specific node from the drop-down list or select the All Nodes option. Next click on the button corresponding to the NMT command you wish to send.

4.3 Reading from a Node

To view the read window choose Read From Node... from the Add menu or click on the read toolbar icon.

The read window allows you to access the object dictionary of a node using SDOs. If possible expedited SDOs will be used, otherwise segmented or block transfers will take place depending on the application preferences.

Choose a node to access from the drop-down list.

Next click on the Select... button to choose an object dictionary entry. If the entry you want to access is not shown then you can manually enter the index and subindex at the bottom. The contents of this window can be customized by specifying an EDS or DCF for the node in the Network Description window. If you are not sure what the type of the custom entry is then choose Domain.

Choose a suitable display data type. If unsure choose Domain. This defaults to match the type of the object dictionary entry you have selected. After reading the data the type can be changed to view the data in different ways.

Finally click on the Read button to perform the read. The read data will be displayed.

If the operation takes too long to complete it can be canceled by clicking on the cancel button in the toolbar or choosing Cancel Current Task from the Network menu.

By choosing the Save Data option the read data can be stored in a binary file. Click on the Browse... button to choose a file.

It is possible to start the read using a trigger. A trigger is a condition that causes an operation (such as reading from a node) to start. Triggers should be used with caution as they can cause heavy bus load and bursts of back to back messages. To configure the trigger click on the Edit... button.

There are several trigger options available and more than one option can be used at the same time. Click the checkboxes to enable the options and then configure them appropriately. Once the trigger window is closed it will take immediate effect. To disable the triggers go back to the edit window.

4.4 Writing to a Node

To view the write window choose Write To Node... from the Add menu or click on the write toolbar icon.

The write window allows you to access the object dictionary of a node using SDOs. If possible expedited SDOs will be used, otherwise segmented or block transfers will take place depending on the application preferences.

Choose a node to access from the drop-down list.

Next click on the Select... button to choose an object dictionary entry. If the entry you want to access is not shown then you can manually enter the index and subindex at the bottom. The contents of this window can be customized by specifying an EDS or DCF for the node in the Network Description window. If you are not sure what the type of the custom entry is then choose Domain.

Choose a suitable entry data type. If unsure choose Domain. This defaults to match the type of the object dictionary entry you have selected.

Enter the data into the window in a format that is suitable for the data type you have chosen. A table at the end of this section shows examples for formatting that is supported.

Finally click on the Write button to perform the write.

If the operation takes too long to complete it can be canceled by clicking on the cancel button in the toolbar or choosing Cancel Current Task from the Network menu.

By choosing the Load Data option the data can be read from a binary file. Click on the Browse... button to choose a file.

It is possible to start the write using a trigger. A trigger is a condition that causes an operation (such as writing to a node) to start. Triggers should be used with caution as they can cause heavy bus load and bursts of back to back messages. To configure the trigger click on the Edit... button.

There are several trigger options available and more than one option can be used at the same time. Click the checkboxes to enable the options and then configure them appropriately. Once the trigger window is closed it will take immediate effect. To disable the triggers go back to the edit window.

Data Type	Example User Input	Notes
Boolean	true false	Case insensitive
Unsigned8, Unsigned16, etc. Real32, Real64	12345 0xA7B1 1.3526 10.6E-2	0x means hexadecimal
Visible Char	A q	
Octet String	1A 2B 3C 4D	Always hexadecimal but without the 0x prefix. Spaces are optional
Visible String Unicode String	Richard of York gave battle in vain Richard of York gave battle in vain	Each character is stored in one byte Each character is stored in two bytes
Time of Day	03/02/2020 14:32:21.362	ISO8601 or RFC1123 formats are supported
Time Difference Domain	3.14:32:21.362 1A 2B 3C 4D	d.hh:mm:ss.iii Always hexadecimal but without the 0x prefix. Spaces are optional
Heartbeat Consumer	1000 0x5A 1000 ms for node 0x5A	Two values in decimal or hexadecimal, space separated, with optional text before, between or after. The first value is milliseconds, the second is node ID
Four State	Off On Failure NotAvailable	

4.5 Trace

The trace window is your view onto the CAN bus and is essential for all development and debugging tasks related to CANopen. To view the trace window choose Trace... from the Add menu or click on the trace window toolbar icon.

The trace window has two modes, stopped and running. Initially the window is stopped and if there are no messages then it will appear blank. Click on the start button in the window to start recording.

When recording any messages that appear on the CAN bus are shown in the window. For performance reasons scrolling to view old messages is not allowed while recording.

During recording the window shows the current, peak and average bus loads. These are estimations based on looking at messages on the bus in one second time slices. The peak and average values are reset each time the trace recording is started.

To stop the recording click on the stop button. When stopped the vertical scroll bar is enabled and all messages in the trace buffer can be viewed. To clear the trace at any time click on the clear button.

Message timestamps have two modes, relative and absolute, and the mode may be chosen at any time by using the drop-down list. In absolute mode the timestamp of each message is measured from a single common starting point. In relative mode the timestamp shows the time since the last message on the bus.

The trace display has two modes, sequence and fixed, and the mode may be chosen at any time by using the drop-down list. Changing the display mode clears the trace window. In sequence mode each message is shown individually. When a new message is placed onto the bus it appears at the bottom of the trace window and older messages are shuffled up. In fixed mode each row corresponds to a specific message identifier. When a new message is placed onto the bus and the identifier has not been seen before then it will appear on a new row in the trace window. However if the identifier has been seen before then it will overwrite the previous message with that identifier. The Cnt column shows the number of messages with that identifier which have been received.

The following screen shot shows the trace window configured for fixed mode and how it would look after a NMT reset and a series of SDO transfers:

Trace 1 - Running					
Absolute		Fixed	Bus load: 0.00% Peak: 6.28% Ave: 0.06%		
Time	Cnt	ID	Msg Type	Node	Details
0:00:05:23.6758...	1	0x701	Bootup	[0x01 (1)] <unknown>	
0:00:05:19.5748...	38	0x601	SDO Initiate Uplo...	[0x01 (1)] <unknown>	[0x1802,0x00] Transmit PDO Parameter 3 - Highest Subinde...
0:00:05:19.5768...	38	0x581	SDO Abort by Se...	[0x01 (1)] <unknown>	[0x06020000 (100794368)] Object does not exist in the obje...
0:00:05:23.6778...	1	0x081	Emergency	[0x01 (1)] <unknown>	[0x0000 (0)] Error reset or no error
0:00:05:23.1728...	1	0x000	NMT Master Req...	[0x01 (1)] <unknown>	Reset

You can see that there has been one NMT producer request, one emergency message and one bootup message. However you can also see there have been 38 transmit SDOs and 38 receive SDOs. Only the contents of the last message for each identifier are shown. Also note that the identifiers are ordered with the lowest at the bottom of the trace window and the highest at the top.

4.6 Network Overview

The network overview window displays a summary of all nodes on the network. To view the window choose Network Overview... from the View menu.

Click on the Scan Network button to find all nodes on the network. For each node the device type and identity information will be read. The NMT status will show unknown until the application sees a bootup, heartbeat or node guarding message for the node.

When the scan is completed an attempt is made to match each node up with nodes defined in the network description. If a match is found the node id in the network description is updated.

The heartbeat producer time for all nodes can be set in one go using this window. Enter a time into the box and click on the Set Heartbeats button.

Some nodes support saving the communication parameters into non-volatile memory. By clicking on the Store Configs button each node in the list will be sent the save command. Optionally the timestamps of the save can be written to a configuration manager on the network.

4.7 Transmit CAN Messages

The CAN messages window allows plain CAN messages to be created and transmitted, with optional transmission triggers. This is typically used for protocols running alongside CANopen or for transmitting PDOs or test messages. To view the window choose CAN Messages... from the View menu.

To create a new message click on the add icon on the toolbar in the CAN messages window. Enter the details of the message and click on the OK button to create it.

By entering a name for the message this name will appear where appropriate throughout the application, including in the trace window. Even if the message will not be manually transmitted using this window, it can be very useful to give names to messages when analyzing and debugging a network.

If the timestamp option is selected then the message data will contain a six-byte Time of Day value each time it is transmitted. Typically this is used in conjunction with the Timestamp message identifier 0x100.

Enabling trigger options allows the message to be transmitted under certain conditions, for example at a predictable rate, when a function key is pressed or when a particular message is seen on the CAN bus.

Once a message has been created it can be transmitted manually by clicking on the button next to the message entry in the window. There is also a toolbar icon to transmit the currently selected message.

The message can be edited by right-clicking on the message and choosing Edit... from the menu, or by clicking on the toolbar icon with the message selected.

There are toolbar icons to delete messages and transmit all defined messages.

Defined messages can be placed into groups. A default group is always present. To create a new group click on the Add Group toolbar button in the CAN Messages window. The new group can be renamed by right-clicking on the tab and entering a new name.

Messages can be cut, copied and pasted between groups. Select the message to cut or copy and then click on the toolbar icon for the relevant operation. Switch to a different group and click on the paste toolbar button to paste the message.

4.8 Preferences

View the preferences window by choosing Preferences... from the View menu or by clicking on the preferences toolbar icon.

The options shown in the preferences window vary according to the version of the application. Most of the options are self-explanatory however a few will be described here.

When transferring more than four bytes to or from a node's object dictionary SDO segmented mode will be used as nodes must support it. For more than 28 bytes it is more efficient to use SDO block mode, which some nodes will support. To enable block mode check the Use SDO block transfers option. The parameters for the block mode can be configured.

When project files are saved they may contain references to other files, for example EDS/DCFs or images. The default is for these paths to remain relative to the project file. This is useful when moving the project to a different PC or a different folder and the referred files are moved with it.

Sometimes it may be useful to keep the referred files in a central location and only move the project file. To enable this check the option Use absolute paths under the Project tab and re-save the project file.

Chapter 5 – Advanced Functionality

5.1 Trace Analysis, Filtering, Recording and Export

When trace recording is stopped and absolute timestamps are showing the origin or zero for the timestamps is either when the PC was started or when the application was started depending on which CAN interface driver is being used. It is possible to set the origin to any message in the recording. This allows the time between two messages to be easily viewed.

To set the timestamp origin to a specific message right click on the message in the recording window and choose Set Timestamp Zero Point from the popup menu. The message will be highlighted in green and its absolute timestamp will be set to zero. Messages older will have a negative timestamp and messages newer will have a positive timestamp.

Sometimes it is useful to manually transmit a message that is shown in the trace window. To do this right click on the message when recording is stopped and choose Add to Defined CAN Messages. The CAN Messages window will open and the message will be copied there. It can now be manually transmitted, periodically transmitted or a transmission trigger created.

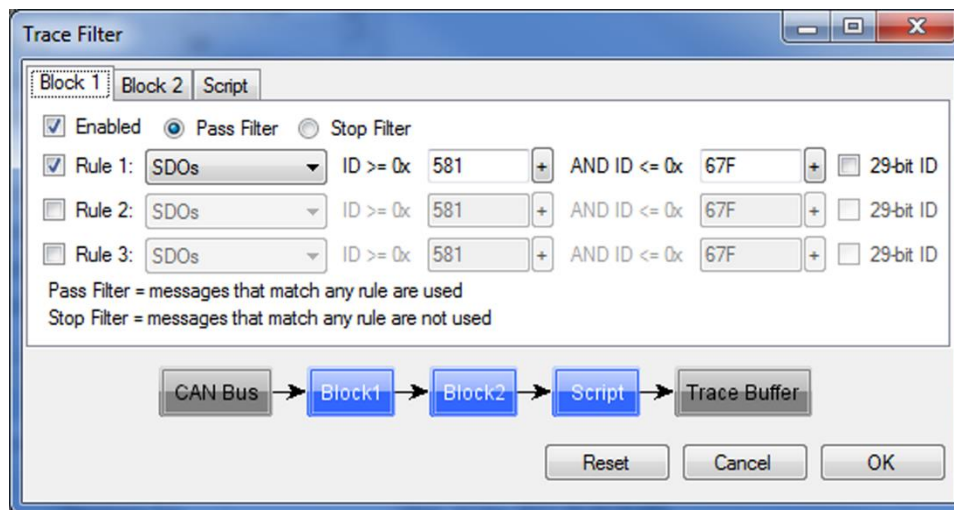
The trace buffer can be exported to a CSV log file for documentation or further analysis in a spreadsheet. The export functionality is enabled when trace recording is stopped. To export the trace click on the export toolbar button in the trace window and choose a location to save to.

The trace window supports long-term recording. This saves messages to a CSV log file when they are received while the trace window is recording. To enable click on the continuous recording toolbar button in the window and select a location to save to. To disable click again on the continuous recording toolbar button.

Note that the CSV log files saved from the trace window can be replayed using the log player. See the log player section of this chapter for details.

Each trace window features its own filter. This allows selection of which messages should be displayed. If there are multiple trace windows open then each one can have a different filter. For example one can show SDOs and another can show PDOs. The export and continuous recording will save the output of the filter.

To configure click on the configure filter toolbar icon in the window and the edit window will be displayed. While recording click on the enable/disable trace filtering toolbar button to turn the filtering on and off.



The filter is divided into three sections, each of which can be individually enabled. When a section is disabled it is not involved in the filtering. The diagram at the bottom of the window shows the relationship between the sections.

Messages are received from the CAN bus and passed to block 1. If they are allowed through the filtering in block 1 they are passed to block 2. If they are allowed through block 2 then they are passed to the script. If they are allowed through the script then the messages are shown in the trace window.

Block 1 and block 2 are functionally identical. Each block can be configured as a pass filter or a stop filter. A pass filter allows any message through that matches one of the rules. A stop filter blocks any message that matches one of the rules. Up to three rules can be enabled and consist of a range of message identifiers. Predefined ranges of identifiers can be chosen from the drop-down lists.

For example to show only PDOs:

1. Enable block 1
2. Set block 1 to a pass filter
3. Enable block 1, rule 1
4. Choose PDOs in the drop-down list for block 1, rule 1

To show all SDOs except for the SDOs belonging to node 0x01:

1. Enable block 1
2. Set block 1 to a pass filter
3. Enable block 1, rule 1
4. Choose SDOs in the drop-down list for block 1, rule 1
5. Enable block 2
6. Set block 2 to a stop filter
7. Enable block 2, rule 1
8. Enter 0x581 in both boxes for block 2, rule 1
9. Enable block 2, rule 2

10. Enter 0x601 in both boxes for block 2, rule 2

A python script can be used to implement sophisticated filtering. The application contains a built in script editor which can be opened by choosing New Script from the Scripts menu or clicking on the New Script toolbar icon. Here is an example script:

```
# Test trace filter script

import sys

print "Parameters = %s" % Parameters

def OnMessageReceiving(Message):
    if Message.Id.Id == 0x000:
        return False
    else:
        return True

def OnMessageReceived(Message):
    if Message.Id.Id == 0x100:
        print "Saw timestamp message"
```

Two functions must be defined, OnMessageReceiving and OnMessageReceived and are called as messages are passed to the script section of the filter.

OnMessageReceiving is called first and performs the actual filtering. Returning False will stop the message and it won't be shown in the trace window. Returning True will allow the message through. In this example NMT messages are not allowed through the filter.

OnMessageReceived is called second and only for those messages that were not filtered out. It allows optional post processing to be performed. In this example a message is printed to the script console when a timestamp message is seen.

Print statements send output to the script console, which can be opened by choosing Script Console from the Scripts menu. This is useful for debugging. Note that each trace window defines two trace buffers which use the same filter, one for display and one for continuous recording. This means that the script is executed twice for every message received, therefore any messages printed to the script console will appear twice.

It is recommended to use tabs for indentation as any mistakes with the indentation in a python script can cause it to function incorrectly.

If the script is modified while using the script trace filtering feature, click on the enable/disable filtering button in the trace window twice to use the new script.

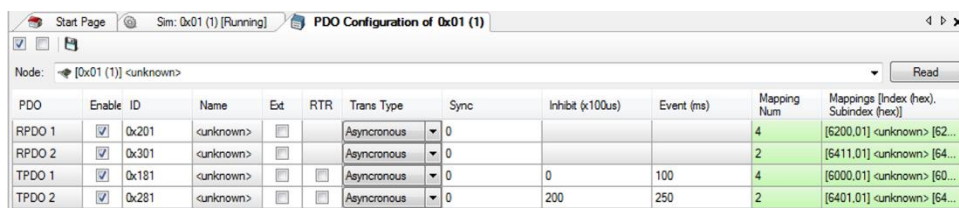
Please refer to the scripting chapter for full details on what is supported in the trace filtering scripts.

5.2 PDO Configuration

To view the PDO configuration window choose PDO Configuration... from the Add menu or click on the PDO configuration toolbar icon.

This window allows you to read and modify the PDOs defined in a node.

Choose a node to access from the drop-down list. Click on the Read button to read the PDOs from the node. The results will be displayed in a table.



PDO	Enable	ID	Name	Ext	RTR	Trans Type	Sync	Inhibit (x100us)	Event (ms)	Mapping Num	Mappings [Index (hex), Subindex (hex)]
RPDO 1	<input checked="" type="checkbox"/>	0x201	<unknown>	<input type="checkbox"/>		Asynchronous	0			4	[6200.01] <unknown> [62...
RPDO 2	<input checked="" type="checkbox"/>	0x301	<unknown>	<input type="checkbox"/>		Asynchronous	0			2	[6411.01] <unknown> [64...
TPDO 1	<input checked="" type="checkbox"/>	0x181	<unknown>	<input type="checkbox"/>		Asynchronous	0	0	100	4	[6000.01] <unknown> [60...
TPDO 2	<input checked="" type="checkbox"/>	0x281	<unknown>	<input type="checkbox"/>		Asynchronous	0	200	250	2	[6401.01] <unknown> [64...

The contents of the table can be exported to a CSV file by clicking on the export icon at the top of the PDO configuration window. This is useful for documenting nodes or sharing configurations with colleagues.

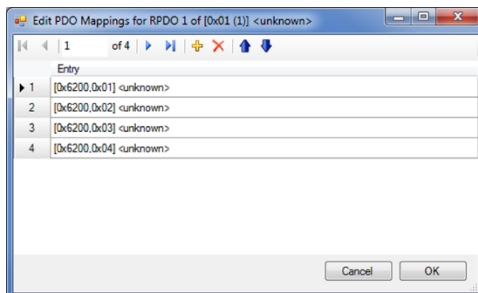
The configurations of the PDOs can be changed by clicking on cells in the table. Note that some nodes may not support changing all of the options and will generate abort errors when an attempt to make changes is made. Any cell that is gray cannot be changed.

To enable or disable a PDO click on the checkbox in the Enable column. To enable or disable all PDOs in one go click on the checkbox icons in the toolbar of the PDO configuration window.

Names are given to PDOs by defining them in the CAN messages window. See the section later in this chapter for details.

Disabled data mappings are shown in read. Enabled data mappings are shown in green.

If the node supports changing the data mappings then click on the mappings cell to edit it.



Click on the yellow plus symbol to add a new mapping entry. Select a mapping entry and then click on the red "X" to delete it. Click on the blue up and down arrows to reorder the mapping entries. Click on a mapping entry to choose a different object dictionary entry.

When OK is clicked the application will attempt to write the new PDO mappings to the node.

5.3 Network Description

The network description is a central place where aspects of the CANopen network can be defined or described. By adding information about the network the user interface of the application is improved. For example by giving a name and EDS/DCF to a node the name will appear throughout the application and custom object dictionary entries can be easily chosen when needed.

The network description window can be opened by choosing Edit Network Description... from the Network menu or clicking on the Network Description toolbar icon.

The network description is divided into the following sections:

- ⤴ Nodes
- ⤴ Network process data
- ⤴ Abort codes
- ⤴ Error codes
- ⤴ Device types
- ⤴ Vendors

The CANopen standard defines a set of abort codes, error codes and device types and these values are built into the application. In addition a set of basic CANopen vendors is also built into the application. The network description window allows you to override or add to these lists of definitions.

For example to add a new abort code:

1. Click on the Abort Codes tab
2. Click on the yellow "+" icon to add a new entry
3. Click in the Abort Code box and enter a new 32-bit value in decimal or hexadecimal (prefix with "0x")
4. Click in the Name box and enter a name

Once added if a node transmits the custom abort code then the name of the code will show up in the trace window and error messages.

To delete the custom abort code click on the row header so a small black triangle is shown then click on the red "X" icon to delete it.

Editing of error codes, device types and vendors works in the same way.

Defining nodes is essential for medium or large network as it makes it easier to determine which node is which without relying on node identifiers. It is not required to define a node in the network description in order to access it, but it does make it easier. To define a node:

1. Click on the Nodes tab
2. Click on the yellow "+" icon to add a new node
3. Choose a node id from the drop-down list
4. Click in the Name box and enter a name
5. Click in the EDS/DCF box and choose a EDS/DCF for the node

To delete the node click on the row header so a small black triangle is shown then click on the red "X" icon to delete it.

If the node is an LSS node and does not have a node id, then choose one of the "LSS Consumer n" options from the node id drop-down list. If an EDS/DCF is specified and it has a complete and unique identity object defined ([1018,01] to [1018,04]) then it is possible for the node id to be assigned automatically. This happens when either the node is simulated and is given a node id through the LSS process, or a network scan in the Network Overview window discovers the node.

Instead of specifying an EDS file it is possible to specify a CANopen Architect file. These files define multiple EDSs, for example an entire network.

If a CANopen Architect file is used then the name of the node must match the name of the desired EDS inside the file. The name is case-insensitive. For example if the file defines an EDS with the name "Pressure" then the node must be given the same name.

Within CANopen Architect it is also possible to define multiple configurations for an EDS. To use a specific configuration the node must be given a name of the form *EDSName-ConfigurationName*. For example if the EDS "Pressure" has the configuration "Test" then the node name must be "Pressure-Test". The configuration name is case-insensitive. Omitting a configuration name means that all the entries in the EDS will be available.

To define more than one node using the same EDS in a CANopen Architect file the node must be given a name of the form *NodeName (EDSName)* or *NodeName (EDSName-ConfigurationName)*.

Process Data is the real-time data being transferred on the network. For example a temperature monitoring node might periodically transmit a PDO containing the current temperature. By working with the Process Data section of the Network Description it is possible to display these values in the application.

To illustrate how this works we are going to create a fake PDO containing a couple of different temperature values and then display them in the trace window.

Open the CAN messages window and create the following two messages:

- ⤴ Identifier = 0x181, length = 2, data = 0x00, 0x00, transmit on keypress F1
- ⤴ Identifier = 0x181, length = 2, data = 0xFF, 0x03, transmit on keypress F2

Connect to a network, start the trace recording and press F1 then F2. You will see two TPDOs from node 1 transmitted. The first contains the value zero. The second contains the value 1023. This might come from a 10-bit ADC for example.

In the Network Description window complete the following steps:

1. Click on the Network Process Data tab
2. Click on the yellow "+" icon to add a new entry
3. Click in the Name box and enter "Temperature"
4. Click in the PDO Message ID box and enter "0x181"
5. Make sure Start Bit is set to zero
6. Choose "Unsigned16" as the type from the drop-down list. This will set the bit size to 16.
7. Keep the Pre Offset and Post Offset at zero.
8. Click on the Scale Factor box and enter "0.195503"
9. Click in the Units box and enter "F"
10. Click elsewhere in the window to ensure the changes are used

If you now look at the Process Data column in the trace window it will show the temperature contained in the PDOs. Using the scale factor we can see that the maximum raw value of 1023 corresponds to a real value of 200 degrees Fahrenheit.

Details	Process Data	Data (Hex)
Default: TPDO 1 of Node [0x01 (1)] <unknown>	Temperature:0.000F	00 00
Default: TPDO 1 of Node [0x01 (1)] <unknown>	Temperature:200.000F	FF 03

Note that the process data column is included in exports of the trace recording, which is ideal for documentation, debugging and analysis of the network.

The start bit corresponds to a bit in the PDO. PDOs can contain up to 64 bits of data so this value can be in the range zero to 63.

The diagram at the bottom of the Network Description window shows how the offsets and scaling factor are applied. Raw data is received from the CAN bus and the pre offset is added. Next the value is multiplied by the scaling factor. Finally the post offset is added to obtain the real value to be shown in the trace window.

If nodes have been defined along with their EDS or DCFs then by clicking on the Automatically Add button process data definitions are automatically generated by reading the EDS/DCF of the nodes. This is useful for complex nodes with large numbers of PDOs.

The Enum column for process data definitions allows display of custom text for specific values. This feature is not available if scaling or pre/post offsets are used.

An enum has the following format:

Value=text;value=text;value=text

Value and text pairs are separated by semicolons. Values can be in decimal or hexadecimal. Here is an example:

5=Operational;0x7F=Bootup

This will display the text if the process data has the specified values, rather than just showing the value.

5.4 Process Data Display

The Process Data window shows real time data in graphical formats, such as meters and graphs. To use this window the process data must have been previously described in the Network Description window.

To open the window choose Process Data... from the View menu.

The window has two modes, locked and unlocked. When locked the graphical controls cannot be edited. When unlocked the controls can be edited. The mode is toggled by clicking on the padlock icon on the toolbar in the window. Initially the window is locked so click on the icon to unlock it.

When the window is unlocked a grid will be shown. Controls will snap to the grid when moved or resized. To change the size of the grid choose a new setting from the drop-down list in the toolbar in the window. Choosing None disables the grid and snapping feature.

To add a new control click on one of the toolbar icon and then click anywhere on the grid. The control will be added.

To move a control drag it around with the left mouse button pressed. To move multiple controls at once select them while pressing the Ctrl key. Once they are selected click and drag one of the selected controls.

To resize a control click on it to show a highlighted border and then drag the border in any direction.

To delete a control right-click over the control and choose Delete from the menu.

Controls generally should not be overlapped however the order of overlapping can be modified by right-clicking on a control and choosing Bring to Front or Send to Back.

Once a control has been added it can be customized and associated with process data that has been defined in the network description. To edit a control right-click over it and choose

Edit... The features enabled in the edit window depend on the type of control and are self-explanatory.

The update of the controls in the window with process data can be enabled or disabled. This is useful to allow pausing of the display for analysis. To enable or disable click on the run/stop icon in the toolbar in the window.

The following short tutorial follows on from the steps in the Network Description section, which described how to create a PDO containing a temperature value and display the value in the trace window.

1. Unlock the Process Data window by clicking on the padlock icon
2. Click on the meter toolbar icon then click on the grid to add a meter
3. Resize the meter so it is a size you like
4. Right click on the meter and choose Edit... from the menu
5. Click on the Process Data tab and from the drop-down list choose "Temperature (0x181, Unsigned16)"
6. Click on the Scale 2 tab
7. Set the start value to zero, end value to 200, Maj Interval to 20 and Min Interval to 2.
8. Change the end of the red range from 100 to 200. The color can be changed if desired by clicking on the red area.
9. Click on OK
10. Lock the window by clicking on the padlock icon.
11. Ensure the window is "running". If it is not click on the start process update toolbar icon in the window
12. Press F1 and F2 to see the meter change value.



5.5 Layer Setting Services

There are two different types of Layer Setting Services (LSS) supported in the application, standard and FastScan. To view the LSS window choose Layer Setting Services... from the View menu.

Detecting all LSS consumers on a network using standard LSS is slow. LSS consumers can therefore be manually entered or detected one at a time.

To manually enter a standard LSS consumer:

1. Click on the Standard LSS tab
2. Click on the yellow "+" button to add a new consumer
3. Click in the boxes in turn and enter decimal or hexadecimal (prefixed with "0x") values that match the consumer's identity
4. Choose an ID to be assigned to the consumer from the drop-down list

To detect a single standard LSS consumer make sure that only one unconfigured LSS consumer is on the network and click on the Autodetect button. The found consumer will be added to the table. Choose an ID to be assigned to the consumer from the drop-down list.

The network selection choice will be set to the current network. If the consumers should be reconfigured for a different bitrate then choose the new network from the drop-down list.

To assign node IDs to the consumers and optionally reconfigure bitrates click on the Config button or the Config and Store button. The Config and Store button instructs the consumers to store the ID and bitrate in non-volatile memory.

If you choose to reconfigure the bitrates of the consumers then the application will switch to the new bitrate at the end of the procedure. This option should only be used if the network consists of only unconfigured LSS consumers.

FastScan LSS offers improvements over the standard version, including faster discovery of LSS consumers on the network.

To detect and configure all FastScan LSS consumers in one go choose a starting node ID and click on Scan and Config.

5.6 Read/Write Node Configurations

The object dictionary of a node can be saved into a Device Configuration File (DCF). Conversely a DCF can be used to write a configuration to a node. The application supports both scenarios.

To view the Node Configuration window choose Node Configuration... from the View menu.

Before a DCF can be generated the EDS or DCF of the node must be specified in the network description. This tells the application what object dictionary entries the node supports.

To save the configuration of a node into a DCF choose the node from the drop-down list and then click on Save DCF...

To update the configuration of a node using a DCF choose the node from the drop-down list and then click on Load DCF...

The application also supports the concept of Network Description Files (NCF). A NCF is a single text-based file that contains multiple DCFs one after the other. The structure of an NCF containing the DCFs of nodes 0x01 and 0x02 would be as follows:

```
[DCF 0x01]
; DCF for node 0x01 goes here

[DCF 0x02]
; DCF for node 0x02 goes here
```

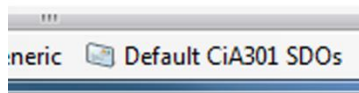
NCFs can be easily manually created using Notepad.

To generate a NCF click on the Save NCF... button. All nodes on the network must have an EDS or DCF defined in the Network Description. To write a NCF to nodes on the network click on the Load NCF... button.

The area at the bottom of the window shows status and error messages.

5.7 SDO Channels

The default configuration is to use the standard SDO channels for accessing SDO servers, which is 0x580 + node ID and 0x600 + node ID. The current SDO configuration is shown in the status bar at the bottom of the main window.



The default configuration is suitable for the vast majority of networks but sometimes other SDO channel configurations are needed. The application allows the following choices:

- ⤴ Request SDO channels from an SDO manager
- ⤴ CiA447 Car Add-on Devices
- ⤴ Custom SDO channels

To view the SDO Channels window choose SDO Channels... from the View menu.

If the network features a node which acts as the SDO manager then no other application or node on the network can access object dictionaries without first requesting a free SDO channel from the SDO manager first. To use this mode choose Request SDOs from SDO Manager from the drop-down list. In order for the application to perform the request it must appear as a minimal SDO server on the network. To configure the minimal SDO server open the Preferences window (Preferences... on the View menu) and click on the SDO Manager tab.

Some device profiles implement “fully-meshed” communication. This assigns an SDO channel to every possible combination of node pairs. For example there is an SDO channel for node 0x01 (the client) to access the object dictionary in node 0x02 (the server) and another channel for the reverse situation. This means that the application must act as one of the clients and use the set of SDO channels assigned to that client.

The CiA447 Car Add-on Devices device profile is one such profile. When using a CiA447 network one of the CiA447 SDO channel configurations must be chosen by using the drop-down list.

Any pair of message identifiers can be used as a custom SDO channel. To define a custom SDO channel:

1. Click on the yellow “+” icon to add a new custom SDO channel
2. Click in the Name box and enter a name for the channel
3. Click in the SDO Request ID and SDO Response ID boxes and enter message identifiers
4. Choose the custom SDO channel from the drop-down list

5.8 Scripting

Python scripts can be created, opened and saved in the application using the built in editor. The scripts are used in the application to provide more flexibility in specific areas, for example the trace filtering.

To create a new script choose New Script from the Scripts menu. An editor will open.

A script that has been modified and not saved has a “*” shown after it's name.

To save the current script choose Save Script or Save Script As... from the Scripts menu.

To open a script choose Open Script... from the Scripts menu.

It is recommended to use tabs for indentation as any mistakes with the indentation in a python script can cause it to function incorrectly.

Scripts can execute the print statement to generate output. The output appears in the script console window, which can be opened by choosing Script Console... from the Scripts menu. Care should be taken when using this feature as it requires significant execution time and may slow the application down.

For details of what objects are available in the scripting environment please see the scripting chapter in this manual.

5.9 CANopen Manager

The CANopen Manager window shows information about a CANopen manager on the network. To view the window choose CANopen Manager... from the View menu. Upon

opening the application will attempt to read the configuration of a CANopen manager at node ID 0x7E. To read a different manager choose a node ID from the drop-down list and click on the Re-Read Manager button.

5.10 Log Player

Previously saved trace recording can be replayed. This is useful for quickly generating background traffic to be used during testing. Trace recordings can be generated by exporting from the trace window or using the continuous recording feature, also in the trace window. Trace recordings are saved as CSV files.

To view the Log Player window choose Log Player... from the View menu.

Click on the Browse... button to choose a trace recording. Click on the play, fast forward and rewind buttons to control the replaying.

5.11 Device Profile Support

Some device profiles require CANopen messages to be interpreted or used in different ways. To choose a device profile to use open the Preferences window (choose Preferences... from the View menu), click on the CANopen Protocol tab and choose a device profile from the drop-down list.

If the device profile you are using is not shown choose Generic.

5.12 Node Setup

The node setup window allows basic configuration of a node. To view the window choose Node Setup... from the Add menu.

To read the configuration of a node choose it from the drop-down list and click on the Read button.

To set the message identifiers and heartbeat producer times enter the desired values and click on the Write buttons.

If the node supports the heartbeat consumer functionality then entries will appear in the table. Choosing a node from the drop-down list and enter a time will cause the new configuration to be immediately written to the node.

If the node supports the emergency consumer functionality then entries will appear in the table. Click in the table cells to enter the configuration, which will be immediately written to the node.

5.13 Node Selection

Many windows in the application feature a drop-down list of nodes, allowing you to choose a specific node to access. The default is to show all 127 nodes however this list can be customized in two ways.

Right click over the drop-down list to view a menu of options.

Choosing Show Named Nodes will show only those nodes with names. To give a name to a node add it to the network description as described earlier in this chapter.

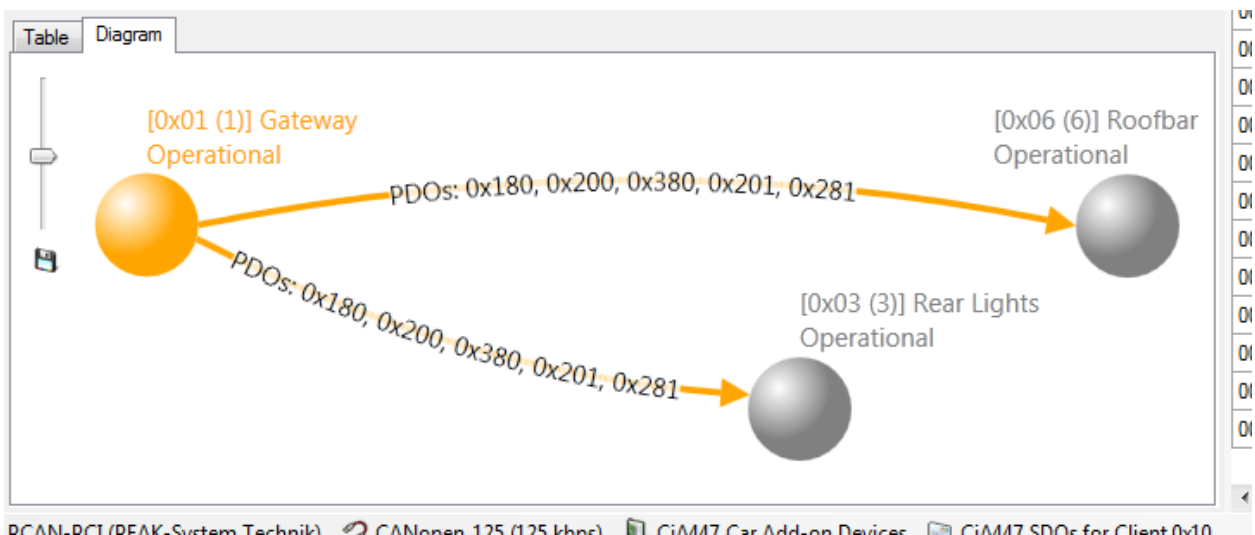
Choosing Show Recently Used Nodes will show those nodes that you have been using. Every time a node is selected from a drop-down list it is added to the recently used list, which is shared throughout the application. To clear the list choose Clear Recently Used Nodes from the same menu.

In the drop-down list an icon is shown next to each node entry. The icon reflects the current mode of the list. For example a star indicates that it is showing recently used nodes.

5.14 Network Diagrams

By scanning a network in the Network Overview window the application can build up diagrams of relationships in the network. These are useful to confirm network design and for documentation of systems.

To generate a diagram choose Network Overview... from the View menu. Click on the Diagram tab and then click on the Scan Network button.



Each node on the network is represented by a sphere. PDOs are represented by arrows, which show the direction the process data is being transmitted. The thicker the arrow the more PDOs are represented by it.

Clicking on a node highlights it in orange. It also highlights all PDOs transmitted by that node.

The name of each node and its current NMT status are shown. The NMT status dynamically changes in response to bootup and heartbeat messages observed on the network.

Drag the slider to change the scale of the diagram. A smaller scale is more useful for dense networks.

Click anywhere on the diagram and then use the mouse wheel to zoom in and out to the location where the pointer is.

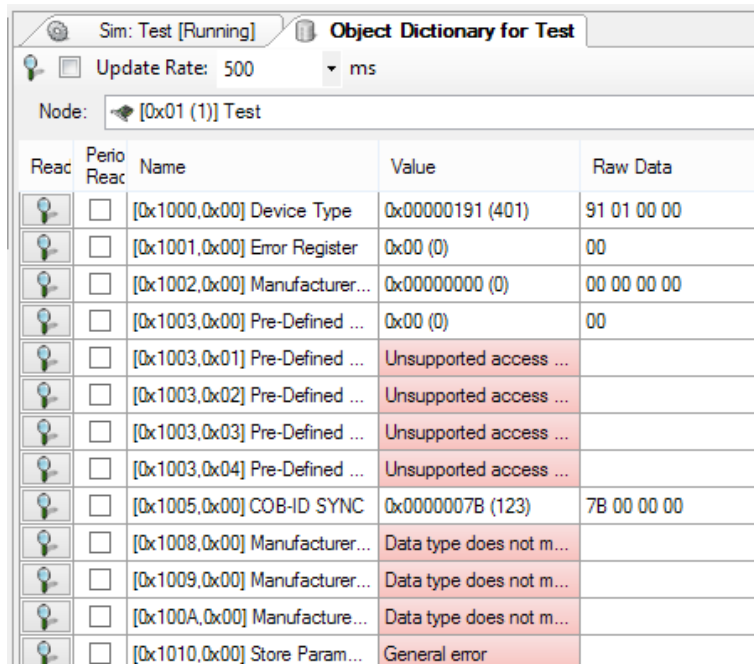
Nodes can be repositioned by clicking and dragging them around the diagram. This is helpful for dense networks.

To save an image (PNG, GIF, JPEG or BMP) of the diagram click on the save icon. This saves an image which can be printed at 300DPI, suitable for documents and publications.

At any time click on the Scan Network button to rescan and regenerate the diagram.

5.15 Node Object Dictionary

The Node Object Dictionary window shows an overview of the contents of the object dictionary. It supports reading all entries, reading specific entries, automatically reading entries at a fixed rate and writing to entries.



Reac	Perio Reac	Name	Value	Raw Data
	<input type="checkbox"/>	[0x1000,0x00] Device Type	0x00000191 (401)	91 01 00 00
	<input type="checkbox"/>	[0x1001,0x00] Error Register	0x00 (0)	00
	<input type="checkbox"/>	[0x1002,0x00] Manufacturer...	0x00000000 (0)	00 00 00 00
	<input type="checkbox"/>	[0x1003,0x00] Pre-Defined ...	0x00 (0)	00
	<input type="checkbox"/>	[0x1003,0x01] Pre-Defined ...	Unsupported access ...	
	<input type="checkbox"/>	[0x1003,0x02] Pre-Defined ...	Unsupported access ...	
	<input type="checkbox"/>	[0x1003,0x03] Pre-Defined ...	Unsupported access ...	
	<input type="checkbox"/>	[0x1003,0x04] Pre-Defined ...	Unsupported access ...	
	<input type="checkbox"/>	[0x1005,0x00] COB-ID SYNC	0x0000007B (123)	7B 00 00 00
	<input type="checkbox"/>	[0x1008,0x00] Manufacturer...	Data type does not m...	
	<input type="checkbox"/>	[0x1009,0x00] Manufacturer...	Data type does not m...	
	<input type="checkbox"/>	[0x100A,0x00] Manufacture...	Data type does not m...	
	<input type="checkbox"/>	[0x1010,0x00] Store Param...	General error	

To open the window click on the Node OD button on the toolbar or choose Node Object Dictionary... from the Add menu.

To use the window first go to the Network Description and specify the EDS or DCF for the node. If you do not do this then a generic object dictionary will be used which may not match the node.

When the window is opened it will scan the selected node and read in the values of all the object dictionary entries defined in the EDS or DCF that are not marked as write-only. Choosing a different node from the drop down list will cause a scan of the new node.

Click on the read button in the toolbar to re-read all object dictionary entries.

Click on the read button at the start of a row to read-read only that specific object dictionary entry.

To continually read an entry check the box in the second column for the entry. Multiple entries may be continually read. This will cause the application to continually make SDO accesses to the node. The rate at which the reading takes place can be adjusted using the update rate drop down list in the toolbar.

Any entry which was successfully read or is write-only can be written to. Click in the value column for the entry and enter a new value. Values can be entered in decimal, hexadecimal or real formats. For example:

- 0x45FD
- -45
- 2.832

When the cursor leaves the table cell or Enter is pressed the application will attempt to write the new value to the node. If it succeeds the entered value will be displayed in the table. If it fails then the value in the node is shown.

The window maintains a history of written values. After a new value has been entered and written to the node a write icon will appear to the right of the value. Click on the icon to access the last 10 values written to the entry. The list is specific to the node ID and is saved to project files.

To clear all input history click on the toolbar icon.

Note that the window can only show entries of a known size. For example `VISIBLE_STRING` and `DOMAIN` types have an unknown size and therefore cannot be shown. Instead use the Read Node window to view the contents of those types of entries.

5.16 Trace Linking

Trace window linking allows display of messages from multiple CAN buses to be shown together.

To link trace windows:

- Start two or more instances of CANopen Magic
- In the trace window that is to send messages drag the link toolbar icon to the destination trace window
- The sender trace window will show "[Shared]" in its title bar, and the destination trace window will show "[1 client]" or similar in its title bar
- When a message is displayed that came from another trace window its details column will show "[from <projectname>]".

Once linked any messages that pass through the sender trace window's filter will be displayed in the destination trace window.

To break the link click on the break link toolbar icon in either trace window.

Linked trace windows also link start, stop, clear and message highlight functionality.

When selecting a message in one window the nearest and second-nearest messages will be highlighted in the other window. The second-nearest will be highlighted in a lighter color.

Clicking on the appropriate toolbar buttons in a linked trace window will also perform that action in all trace windows that are linked together.

Some notes:

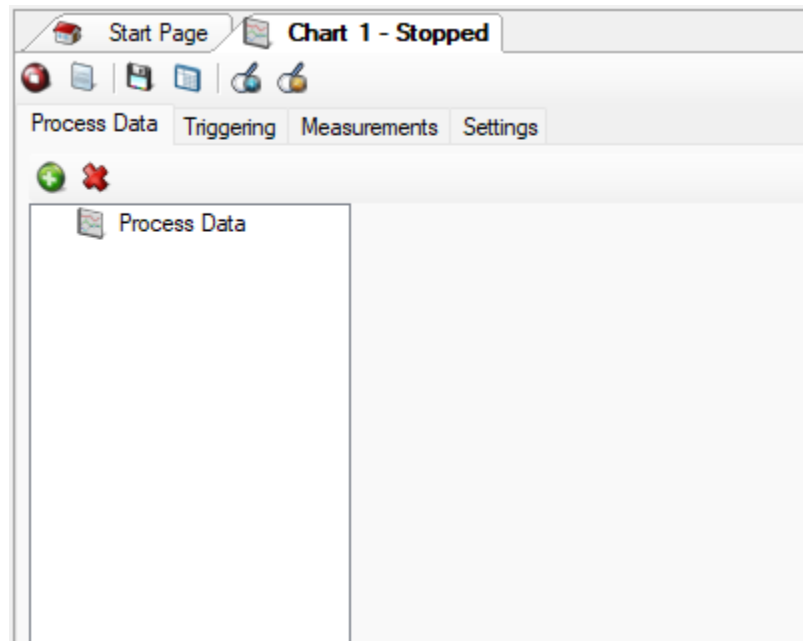
- Messages must pass through the sender's filter and the destination's filter to be displayed
- Avoid creating loops, e.g. trace A -> trace B -> trace C -> trace A
- Timestamps will only be correct if all messages come from the same type of CAN interface, e.g. all from PEAK interfaces.
- If the sender project is saved under a new name the name displayed in the destination trace window details column will automatically change to match
- Each trace window can have up to four senders linked to it
- If a trace window is a sender/destination then it cannot also be a destination/sender

5.17 Charting

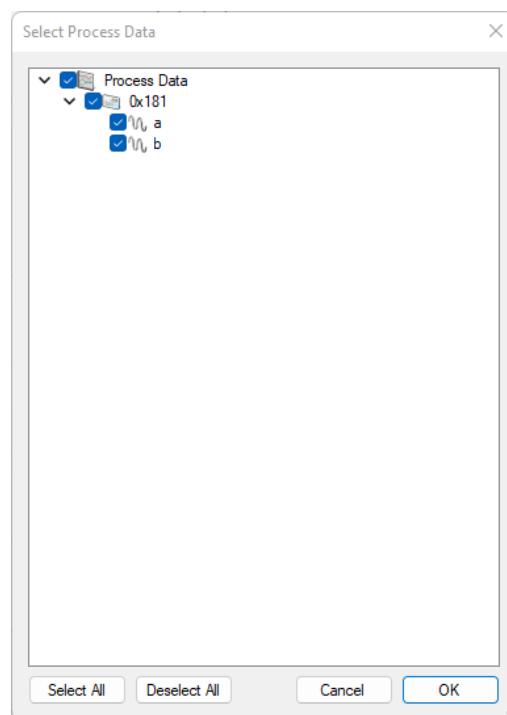
The charting interface can take process data and plot it on a chart for export as an image or for analysis.

Measurements on the data can be performed.

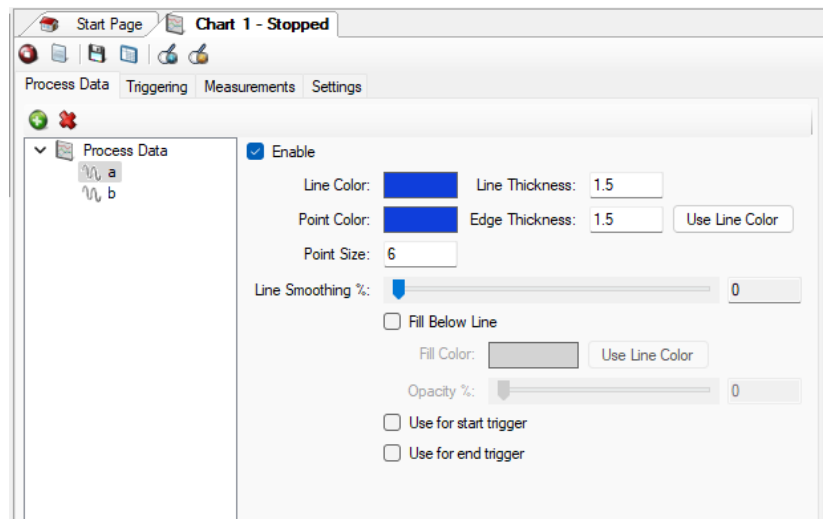
To open the charting interface click on the Chart toolbar icon.



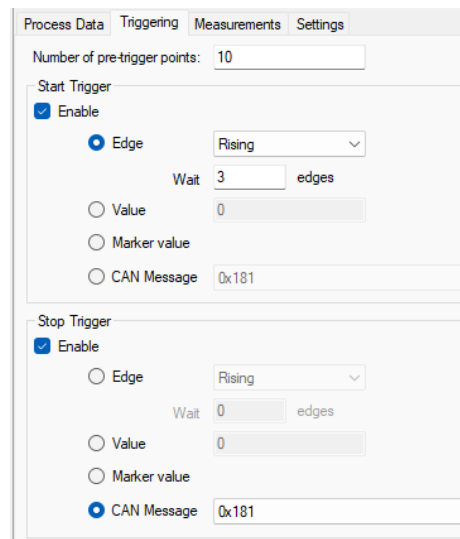
Under the Process Data tab click on the green “+” button to choose the process data to show on the chart.



Once chosen click on the process data to view the options for that data. The enable checkbox will cause the data to be shown or disabled.



It is possible to start and stop recording of data on trigger events. Click on the Triggering tab to view the options.



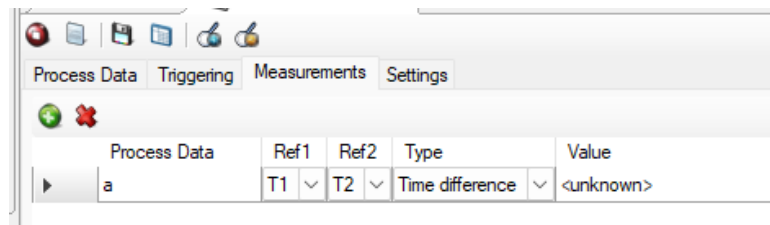
For example recording can start when a specific number of rising edges have occurred and stop when a specific CAN message appears on the bus.

Click on the record button to start the chart recording and if configured, look for the start trigger.



Click on the stop button to stop recording (or wait for the stop trigger) and to allow measurements to take place.

To make measurements click on the Measurements tab. Click on the green "+" button to add a new measurement.



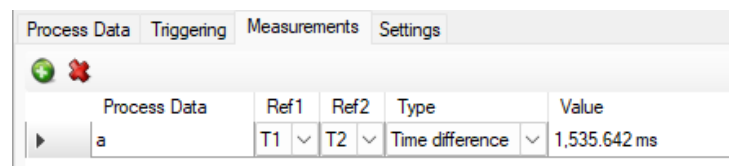
Measurements are made between two reference points, called Ref1 and Ref2. The reference points are determined by measurement lines added to the chart.

To add a line right-click on the chart and choose Show Vertical Measurement Line -> T1. Then do the same for T2.



Measurement lines can be moved by clicking and dragging on them, either on the circle or the line itself. If a measurement line is dragged off the side of the chart it is removed.

Once added the measurements using the lines becomes functional.



Choose the measurement type from the drop-down list.

Some measurements need to know the threshold between logic low and logic high, for example the duty cycle. To specify the threshold the logical level marker must be added to the chart.

Right-click on the chart and choose Show Logical Level Marker.

The marker appears as a grey horizontal line with a diamond. Just like the measurement lines it can be moved by clicking and dragging.



Here is an example of a measurement based on the screenshot above, using the marker to determine the number of rising edges between T1 and T2.

Process Data					
Inggerring					
measurements					
Settings					
Process Data					
Ref1 Ref2 Type Value					
a		T1	T2	Rising edges	2

Under the settings tab some options may be chosen that affect the operation of the chart.

The chart can automatically determine the Y axis scale based on the incoming data but if desired, a specific range can be specified.

During recording the chart will fill a scrolling buffer and then start to scroll the data. The size of this buffer is defined by the Scroll Window Size. Even though data may move out of the scrolling buffer it will still be shown when recording is stopped.

The refresh period defines how often the chart is updated during recording. This affects the speed of the scrolling.

The chart can be zoomed by using the mouse wheel. The zoom will be centered on the location of the mouse pointer. Position the pointer on a measurement line to zoom at the line.

There are also zoom buttons on the toolbar. These zoom in and out of the center of the chart.

The chart can be exported as an image using the toolbar button.

The process data displayed on the chart can be exported in CSV format for further processing and analysis in a spreadsheet, such as Excel.

The Ultimate version is able to open up to 10 charting windows at the same time.

Chapter 6 – Simulation

6.1 Overview

The simulation system is a sophisticated method of creating and testing CANopen nodes. The system runs multiple copies of the MicroCANopen Plus CANopen stack at full speed allowing for accurate testing. A network can consist of a mixture of simulated and real nodes communicating with each other.

The suggested development process is as follows:

- ⤴ Compile the CANopen stack into the nodes needed for the network
- ⤴ Simulate and test
- ⤴ Retarget nodes for a microcontroller
- ⤴ Final testing on hardware

The CANopen stack is compiled into a DLL with an API that can be used by the application to execute it. Some example simulation nodes are provided. Purchasing MicroCANopen Plus allows custom nodes to be developed.

6.2 Adding Simulation Nodes

To add a simulation node choose New Simulation Node... from the Simulation menu. The simulation node window will open.

Choose a node ID to use for the node or choose "LSS Consumer n" if the node does not have a node ID.

For LSS nodes each simulation node should have a unique choice, such as LSS Consumer 1 or LSS Consumer 2.

Click on the Configure... button to choose a different simulation node DLL and serial number, if desired.

The contents of non-volatile memory are saved to the project file for each node. Checking the option to restore the NVOL contents will result in the last used non-volatile memory contents (if available) being restored before the simulated node is started. The restore happens on each start of the simulated node.

To start the node click on the Start button. To stop the node click on the Stop button.

If the node does not use Layer Setting Service then a boot up message will be transmitted onto the CAN bus. The features of the application can then be used to access the node.

If the node does use Layer Setting Services then once the node has been assigned an identifier the user interface will update to reflect it. The features of the application can then be used to access the node.

Once running the Process Image tab in the simulation node window will show the contents of the object dictionary entries containing read/write data.

6.3 Process Data Display

The state of process data (real time data) inside the node while it is running can be displayed. To define new process data entries click on the Node Process Data tab in the simulation node window.

The default MicroCANopenPlusCiA401.dll simulation node uses a counter to continually increment a 16-bit value stored in the object dictionary entry [0x6401,0x01]. This can be seen by scrolling down the object dictionary table under the Process Image tab. The following steps will illustrate how to add a process data entry for this value.

1. Click on the Node Process Data tab
2. Click on the yellow "+" icon to add a new entry
3. Click in the Name box and enter "Analog Input"
4. Click in the Entry box and choose entry [0x6401,0x01]
5. Keep Start Bit at zero.
6. Choose Unsigned16 for the type. The bit size will change to 16.
7. Keep the Pre Offset and Post Offset at zero.
8. Enter "0.015259" into the Scale Factor box.

Next click on the Visual tab.

This section of the window has two modes, locked and unlocked. When locked the graphical controls cannot be edited. When unlocked the controls can be edited. The mode is toggled by clicking on the padlock icon on the toolbar in the window. Initially the window is locked so click on the icon to unlock it.

When the window is unlocked a grid will be shown. Controls will snap to the grid when moved or resized. To change the size of the grid choose a new setting from the drop-down list in the toolbar in the window. Choosing None disables the grid and snapping feature.

To add a new control click on one of the toolbar icon and then click anywhere on the grid. The control will be added.

To move a control drag it around with the left mouse button pressed. To move multiple controls at once select them while pressing the Ctrl key. Once they are selected click and drag one of the selected controls.

To resize a control click on it to show a highlighted border and then drag the border in any direction.

To delete a control right-click over the control and choose Delete from the menu.

Controls generally should not be overlapped however the order of overlapping can be modified by right-clicking on a control and choosing Bring to Front or Send to Back.

Once a control has been added it can be customized and associated with process data that has been defined under the Node Process Data tab. To edit a control right-click over it and choose Edit... The features enabled in the edit window depend on the type of control and are self-explanatory.

The update of the controls in the window with process data can be enabled or disabled. This is useful to allow pausing of the display for analysis. To enable or disable click on the run/stop icon in the toolbar in the window.

The following short tutorial follows on from the previous steps where a process data item for the analog input was created:

1. Unlock the Visual section by clicking on the padlock icon
2. Click on the meter toolbar icon then click on the grid to add a meter
3. Resize the meter so it is a size you like
4. Right click on the meter and choose Edit... from the menu
5. Click on the Process Data tab and from the drop-down list choose "[0x6401,0x01] Analog Input (Unsigned16)"
6. Click on the Scale 2 tab
7. Set the start value to zero, end value to 1000, Maj Interval to 200 and Min Interval to 20.
8. Change the end of the red range from 100 to 1000. The color can be changed if desired by clicking on the red area.
9. Click on OK
10. Lock the window by clicking on the padlock icon.
11. Ensure the window is "running". If it is not click on the start process update toolbar icon in the window

The needle on the meter will slowly increase from zero to 1000 and then swing back to zero when the 16-bit counter wraps around.



It is also possible to simulate inputs. The Visual section contains a toggle switch which can be associated with boolean process data, and a dial which can be spun around.

It is possible to automatically define process data entries for a simulated node. It is first necessary to specify an EDS or DCF for the node and start execution at least once so the process image is known. Next click on the Automatically Add button under the Process Data tab. The process image of the node is cross-referenced with the EDS/DCF and the entries are created. It will still be necessary to define offsets, scaling and units where needed.

6.4 Dynamic Object Dictionary Simulation

Some simulation nodes have the ability to dynamically construct an object dictionary directly from an EDS/DCF. This is useful for rapid prototyping of nodes. To use this feature:

1. In the Simulation Node window click on Configure...
2. Click on Browse... to choose a simulation DLL
3. Go to My Documents\CANopen Magic Ultimate\SimulationNodes or My Documents\CANopen Magic Evaluation\SimulationNodes
4. Choose MicroCANopenPlusXOD.dll
5. Click on Browse... to choose the EDS/DCF to use for the simulation
6. Check the box "Convert EDS into node configuration"
7. Click on OK

Start the simulation node. There will be a pause while the EDS/DCF is converted into an object dictionary that the simulation node can use, then the simulation will start.

Click on the process image tab to view the object dictionary in the node and verify your custom entries are there.

If for some reason the EDS/DCF cannot be converted a minimal object dictionary will be used instead.

Chapter 7 – Command Line

7.1 Overview

Some CANopen Magic functionality is available via a command line interface, allowing usage from batch files or custom applications.

The command line interface is called using CANopenMagicCL.exe and has the following command format:

```
canopenmagiccl.exe [options]
```

where [options] is a space separated list of options to use.

Options have a long format and sometimes a short format. Long formats start with two hyphens. Short formats start with a single hyphen and use one letter. For example the following two options are the same:

```
--network="CANopen_1000"  
-n="CANopen_1000"
```

Please note that you must provide on a single command line all the necessary options for the application to connect and use the network. Each time you run the application it does not remember how it was last used.

To get help use the --help option:

```
canopenmagiccl.exe --help
```

The order of options is important. For example using:

```
--nmt="0,reset" --nmt="1,operation"
```

will first reset all nodes and then place node 0x01 in operational state. If these two options were reversed then they would be executed in the reversed order producing a different result.

7.2 Interrogation

In order to use the command line interface it must select a specific driver, CAN interface and network, all of what are defined by their names. It is possible to read out the supported driver, interface and network names using the command line application. For example to show the drivers currently available:

```
canopenmagiccl.exe --listdrivers
```

To choose the loopback driver and then list the available CAN interfaces for that driver:

```
canopenmagiccl.exe --driver="Loopback" --listinterfaces
```

To choose the loopback driver and the internal CAN interface and then list the available networks:

```
canopenmagiccl.exe --driver="Loopback" --interface="Internal" --listnetworks
```

Finally to connect to the loopback driver, internal CAN interface and 1Mbps network then send an NMT command:

```
canopenmagiccl.exe --driver="Loopback" --interface="Internal" --  
network="CANopen_1000" --nmt=0,reset"
```

This can be shortened to:

```
canopenmagiccl.exe -d="Loopback" -i="Internal" -n="CANopen_1000" --nmt=0,reset"
```

7.3 Tasks

Tasks are options that may appear more than once on a command line and cause something to happen on the CAN bus. This section describes each supported task in turn.

NMT

The NMT task sends out NMT messages to the network and has the following format:

```
--nmt="nodeid,state"
```

nodeid is a node ID or zero to choose all nodes on the network. It can be a decimal number or a hexadecimal number with a "0x" prefix.

state can be one of the following:

- ✧ operational
- ✧ preoperational
- ✧ stop
- ✧ reset
- ✧ resetcomm

Examples:

```
--nmt="0,reset"  
--nmt="3,operational"  
--nmt="0x5F,stop"
```

SDODOWNLOAD

This task writes data to a node and has the following format:

```
--sdodownload="nodeid,index,subindex,data,length"  
--write="nodeid,index,subindex,data,length"  
-w="nodeid,index,subindex,data,length"
```

nodeid is the ID of the node to write to. It can be a decimal number or a hexadecimal number with a "0x" prefix.

index is the 16-bit object dictionary entry index. It can be a decimal number or a hexadecimal number with a "0x" prefix.

subindex is the 8-bit object dictionary entry subindex. It can be a decimal number or a hexadecimal number with a "0x" prefix.

data is a path and name of a binary file containing data to write, or a decimal or a hexadecimal number with a "0x" prefix. If a decimal or hexadecimal value then the maximum size is 64-bits of data.

length is the length of data to write in bytes. It can be a decimal number or a hexadecimal number with a "0x" prefix. If a file is being written to the node then set this value to zero.

Examples:

```
--sdodownload="0x01,0x1017,0x00,1000,2"  
-w="0x5B,0x2000,0x01,..../testdata/myfile.bin,0"
```

SDOUPLOAD

This task reads data from a node and stores it into a file. It has the following format:

```
--sdoupload="nodeid,index,subindex,filename"  
--read="nodeid,index,subindex,filename"  
-r="nodeid,index,subindex,filename"
```

nodeid is the ID of the node to read from. It can be a decimal number or a hexadecimal number with a "0x" prefix.

index is the 16-bit object dictionary entry index. It can be a decimal number or a hexadecimal number with a "0x" prefix.

subindex is the 8-bit object dictionary entry subindex. It can be a decimal number or a hexadecimal number with a "0x" prefix.

filename is the path and name of the file to store the read data in.

Examples:

```
--sdoupload="0x01,0x1000,0x00,devicetype.bin"  
-r="0x5B,0x2000,0x00,../testdata/myfile.bin"
```

7.4 Examples

Here are some examples of complete command lines.

Reset all nodes, read device type of node 0x01 and node 0x4B:

```
canopenmagiccl.exe -d="Loopback" -i="Internal" -n="CANopen_1000" --nmt=0,reset" -  
r="0x01,0x1000,0x00,node1_devicetype.bin" -  
r="0x4B,0x1000,0x00,node4b_devicetype.bin"
```

Make all nodes operational and set the heartbeat time for node 15 to 500ms:

```
canopenmagiccl.exe -d="Loopback" -i="Internal" -n="CANopen_1000" --  
nmt=0,operational" -w="15,0x1017,0,500"
```

7.5 CANopen Magic Pro Library

For developing your own CANopen applications there are two choices:

- ⤴ Call the command line interface from your application
- ⤴ Use the CANopen Magic Pro library (available separately)

The command line interface is included with CANopen Magic and is easy to use, however it has some limitations when it comes to developing custom CANopen applications:

- ⤴ Results of operations are human-friendly and not so easy to automatically parse
- ⤴ Each execution requires reconnecting to the CAN interface which has a delay involved
- ⤴ The functionality is basic and is intended for simple production-line tasks
- ⤴ Windows only

The CANopen Magic Pro library is a set of C or .NET libraries providing a large amount of CANopen functionality already developed and tested. It can run on Windows or Linux (with SocketCAN). Applications developed with the library can have sophisticated user interfaces constructed with better error detecting and reporting.

For more information see: <http://www.canopenstore.com/pip/canopen-magic-pro-dll.html>

7.6 GUI Command Line

The GUI version of CANopen Magic has a basic command line, allowing for projects to be opened and scripts to be run. This is useful for automated testing of nodes using the full scripting environment.

See also `App.SetExitCode()` in the scripting chapter.

Opening a project:

```
CANopenMagicUltimate.exe --project Foo.cox
```

Opening a script:

```
CANopenMagicUltimate.exe --script Bar.py
```

Opening and running a script:

```
CANopenMagicUltimate.exe --runscript Bar.py
```

Opening a project, opening a script, running the script and closing the application when the script exits:

```
CANopenMagicUltimate.exe --project Foo.bar --runscript Bar.py --runscript --  
closeonscriptexit
```

If you wish to wait for a script to exit on the command line or in a batch file use the Windows start command:

```
start /WAIT CANopenMagicUltimate.exe --project Foo.bar --runscript Bar.py --runscript --  
closeonscriptexit  
echo %errorlevel%
```

Chapter 8 – Trace Filtering Scripts

8.1 Overview

The trace filter script environment is a very simplified Python environment for the sole purpose of providing more sophisticated trace filtering.

When a trace filter script is loaded or reloaded it is immediately executed. This means that global variables will be initialized. After loading the application calls functions in the script depending on the script's purpose. The sections of this manual that describe features which use scripts detail the names of functions that are called and how they can be used.

8.2 Editing

Python scripts can be created, opened and saved in the application using the built in editor.

To create a new script choose New Script from the Scripts menu. An editor will open.

A script that has been modified and not saved has a "*" shown after it's name.

To save the current script choose Save Script or Save Script As... from the Scripts menu.

To open a script choose Open Script... from the Scripts menu.

It is recommended to use tabs for indentation as any mistakes with the indentation in a python script can cause it to function incorrectly.

8.3 Debugging

If a script uses the print statement then the output will be displayed in the script console. To view the console choose Script Console... from the Scripts menu. Using the print statement can affect performance of the entire application and should therefore be used with care.

8.4 Objects

Message

This object is used to describe a CAN message, for example it is passed to the functions involved in trace filtering.

Fields:

- ✧ Length = length of data in bytes
- ✧ Flags = message flags, one or more of: None, Rtr, ErrorFrame

- ⤴ Data = byte array
- ⤴ Timestamp = date and time of message
- ⤴ Id = message identifier
- ⤴ Name = name of message

The message identifier field Id has the following fields:

- ⤴ Id = message identifier value
- ⤴ Ext = true if using 29-bit identifier, false if using 11-bit identifier

For example to access the identifier value of a message use:

```
message.Id.Id
```

8.5 Functions

OnMessageReceiving

Called when a CAN message has been detected on the bus. Passed is the CAN message. If the message should be placed into the trace buffer then return True. If the message should be ignored (i.e. filtered out) then return False.

OnMessageReceived

Called when a CAN message has been placed into the trace buffer. Passed is the CAN message that was received.

This function is not called for messages that were filtered out (either by using the block filters or by the script function OnMessageReceiving).

8.6 Parameters

When specifying the trace filter script to use there is an optional text field for parameters. The contents of this field are passed to the script in a string called Parameters. The value of the Parameters string is set whenever the script is first executed after loading the script or modifying the script.

8.7 Example

Here is a simple example of a trace filtering script. Print statements send their output to the script console window.

```
# Example trace filter script

# show the parameters passed to the script, if any
print "Parameters = %s" % Parameters

# called when message has been detected on bus
```

```
# determine if message should be filtered out or not
# return false to filter out
def OnMessageReceiving(Message):
    if Message.Id.Ext:
        print "Checking message: 0x%.8X" % Message.Id.Id
    else:
        print "Checking message: 0x%.3X" % Message.Id.Id

    if Message.Id.Id == 0x001:
        return False

    return True

# called when message has been received (i.e. not filtered out)
# can be used for custom message interpretation in the script
# console window
def OnMessageReceived(Message):
    if Message.Id.Ext:
        print "Rx message: 0x%.8X" % Message.Id.Id
    else:
        print "Rx message: 0x%.3X" % Message.Id.Id
```

Chapter 9 – Trace Interpreting Scripts

9.1 Overview

The trace interpreting script environment is a very simplified Python environment for the sole purpose of providing more sophisticated message interpretation in the trace window.

When a trace interpreting script is loaded or reloaded it is immediately executed. This means that global variables will be initialized. After loading the application calls functions in the script depending on the script's purpose. The sections of this manual that describe features which use scripts detail the names of functions that are called and how they can be used.

9.2 Editing

Python scripts can be created, opened and saved in the application using the built in editor.

To create a new script choose New Script from the Scripts menu. An editor will open.

A script that has been modified and not saved has a "*" shown after it's name.

To save the current script choose Save Script or Save Script As... from the Scripts menu.

To open a script choose Open Script... from the Scripts menu.

It is recommended to use tabs for indentation as any mistakes with the indentation in a python script can cause it to function incorrectly.

9.3 Debugging

If a script uses the print statement then the output will be displayed in the script console. To view the console choose Script Console... from the Scripts menu. Using the print statement can affect performance of the entire application and should therefore be used with care.

9.4 Objects

Message

This object is used to describe a CAN message, for example it is passed to the function involved in trace interpretation.

Fields:

- Length = length of data in bytes
- Flags = message flags, one or more of: None, Rtr, ErrorFrame
- Data = byte array
- Timestamp = date and time of message
- Id = message identifier
- Name = name of message

The message identifier field Id has the following fields:

- Id = message identifier value
- Ext = true if using 29-bit identifier, false if using 11-bit identifier

For example to access the identifier value of a message use:

```
message.Id.Id
```

InterpretedCANMessage

This object is used to describe a CAN message that has been interpreted. It is used for creating the display in the trace window. It is passed to the function involved in trace interpretation.

Fields:

- Time = timestamp of message as a string
- Count = number of messages of this ID as a string
- Id = message identifier as a string
- MessageType = type of message as a string
- Node = Node that transmitted the message as a string
- Details = further description of the message as a string
- ProcessData = description of the process data in the message as a string
- DataHex = hexadecimal interpretation of the message data as a string
- DataASCII = ASCII interpretation of the message data as a string
- DataDecimal = interpretation of the message data as decimal values as a string
- RawMessage = the data bytes for the entire message as a string

9.5 Functions

OnMessagePreInterpret

Called when a CAN message is about to be interpreted. Passed is the CAN message.

Use this function to manipulate a message before it is passed to the built-in interpreter. For example the message identifier could be changed.

It is recommended not to change the contents of the Time, Count and RawMessage fields, as doing so is likely to make the trace window output confusing.

Use with care as this function can easily create confusing output to the user.

OnMessageInterpret

Called when a CAN message needs to be interpreted. Passed is the CAN message and an interpretation object.

When this function is called the string fields of the interpretation object have already been filled in with the standard CANopen interpretation. Inside this function the script can either add to the existing contents of one or more fields or replace the contents of one or more fields.

It is recommended not to change the contents of the Time, Count and RawMessage fields, as doing so is likely to make the trace window output confusing.

If you wish to be able to play back exported log files that have been interpreted using a script then the following requirements must be met:

- Time field must not be changed
- Id field must not be changed
- RawMessage field must not be changed
- The MessageType field must contain the following strings when applicable. The order they appear and the case used is not important. "Ext" means a 29-bit identifier:
 - Ext
 - RTR
 - Error Frame

The names of the fields directly correspond to the columns in the trace window.

Avoid using the python 'print' function inside this function as it could lead to performance problems.

9.6 Parameters

When specifying the trace interpreting script to use there is an optional text field for parameters. The contents of this field are passed to the script in a string called Parameters. The value of the Parameters string is set whenever the script is first executed after loading the script or modifying the script.

9.7 Example

Here is a simple example of a trace interpreting script. Print statements send their output to the script console window.

```
# Example message interpreter script

# show the parameters passed to the script, if any
# all printed output goes to the script console window
```

```
print "Parameters = %s" % Parameters

# called when a CAN message needs to be interpreted
def OnMessageInterpret(Message, Interpretation):

    # set the string-based fields of the Interpretation object
    # to reflect what the CAN message means
    # note that these fields have already been filled in with the
    # standard CANopen interpretation by the application and therefore
    # you can also add on to the existing contents as well

    # it is possible to change the Time, Count and RawMessage fields as well
    # but it is not recommended as the display will become confusing

    # warning: avoid using 'print' inside this function as it will be called a lot
    # potentially causing performance problems

    Interpretation.Id = Message.Id.ToString();
    Interpretation.MessageType = "A";
    Interpretation.Node = "B";
    Interpretation.Details = "C";
    Interpretation.ProcessData = "D";
    Interpretation.DataHex = "E";
    Interpretation.DataASCII = "F";
    Interpretation.DataDecimal = "G";

# called before a CAN message is interpreted
# allows the raw message to be modified
def OnMessagePreInterpret(Message):
    if Message.Data[0] == 0x01:
        Message.Data[0] = 0x55
```

Chapter 10 – General Purpose Scripts

10.1 Overview

The scripting system allows for the development and testing of Python-based scripts that can interact with the user, CAN bus and the application. It is an ideal platform for rapid prototyping of ideas and test harnesses.

There are two distinct ways that python code can be executed inside CANopen Magic – by running a script or entering commands manually into a console.

To create a new script choose New Script... from the Scripts menu or click on the New Script toolbar button. A window will open showing two tabs, one for the script editor and one for the console.

Clicking on the Execute Script button will immediately execute the script and show the output in the console area. It is not necessary to save the script in order to execute it. You can also type "run" into the console to execute the script.

The environment supports Python 2.7 and all the standard libraries are available for use. However it is not guaranteed that any particular standard library feature is functional inside the CANopen Magic environment.

To use the console click at the prompt ">>>" and enter a command. For example:

```
a = App.AbortCode(0x12345678, "My abort")
print a
App.Instances.NetworkDescription.AddAbortCode(a)
```

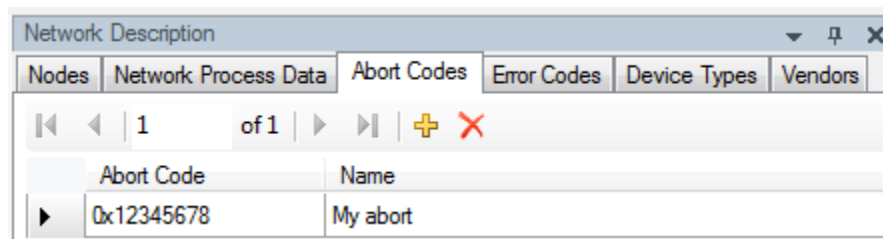
```
>>>a = App.AbortCode(0x12345678, "My abort")

>>>print a

[0x12345678 (305419896)] My abort
>>>App.Instances.NetworkDescription.AddAbortCode(a)

>>>|
```

Now view the Network Description and click on the Abort Codes tab.



10.2 Utility Functions

CANopen Magic provides some utility functions that are available to all scripts to help with user interaction and various other aspects not catered for by the standard Python libraries.

Read

The Read function gets a line of text from the user. Here is an example:

```
Text = Read()
print Text
```

If you run this script it will wait for the user to enter some text then press enter. The text will then be displayed.

ReadKey

The ReadKey function gets a single keypress from the user. This is useful for constructing menus. It can be used as follows:

```
Key = ReadKey()
print key
```

Note that this script will print the key pressed twice. That is because user input is also shown on the screen.

App.SetExitCode

This function sets the return code for the executable. This can be used in conjunction with running a script from the command line to pass a result code to a batch file. Here is an example:

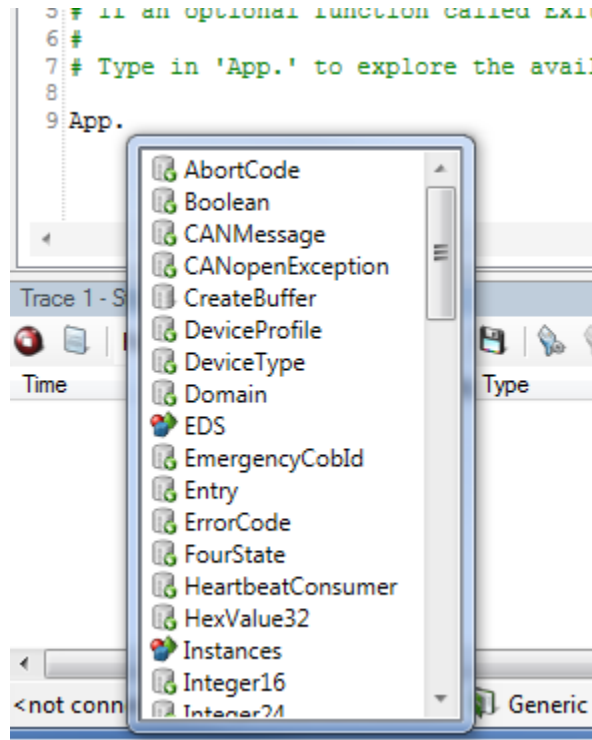
```
App.SetExitCode(5)
```

10.3 Namespaces and Classes

Access to CANopen Magic from Python is via a set of classes. For example an abort code is a class containing a 32-bit value and a name. The collection of classes are described in the rest of this chapter. All classes can be found in the "App" namespace. For example:


```
a = App.AbortCode(0x12345678, "My abort")
```

Typing "App." Into the script editor will bring up a list of available classes.



When the application starts it creates its own instances of some of these classes and makes them available to scripts. These instances can be found in the App.Instances namespace. For example:

```
App.Instances.NetworkDescription.AddAbortCode(a)
```

This adds an abort code to the network description used by the application. If you opened the Network Description window after executing this statement then you would see your abort code.

Executing the following:

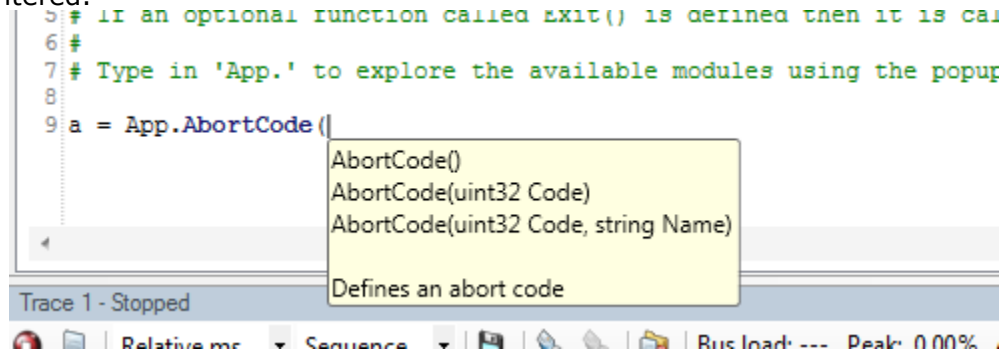
```
desc = App.NetworkDescription(True)
```

creates a new network description instance, but this has nothing to do with the copy that the application is using internally.

In summary: there is no need to create your own instances of classes that can already be found in the App.Instances namespace.

10.4 Function Declarations

When a "(" is entered into the script editor it will display a list of possible parameters that can be entered.



This example shows that there are three ways to create a new abort code – no parameters, 32-bit code and 32-bit code with name.

10.5 Class Reference

For a complete reference of all the classes available, including enumerations, properties and functions, please see CANopenMagicPythonRef.pdf which is included with versions of CANopen Magic that support scripting.