PCANopen Magic Pro
User Manual

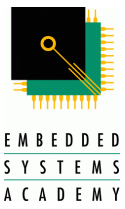
Manual Revision 1.23

PEAK System Technik

For support contact support@esacademy.com

For the latest news on PCANopen Magic Pro visit PEAK-System Technik at

www.peak-system.com



PCANopen Magic Pro was developed for PEAK-System Technik by Embedded Systems Academy. Embedded Systems Academy provides training and consulting services, specializing in CAN, CANopen and Embedded Internetworking. For more information visit

www.esacademy.com

Contents

Contents 3
About This Manual 6
Chapter 1 – Introduction 7
1.1 About CANopen 7
1.2 About PCANopen Magic Pro 7
1.3 PCANopen Magic Pro Features 7
1.4 Obtaining Compatible CAN Interfaces 10
Chapter 2 – Installation and Setup 11
2.1 Installation 11
Minimum Requirements 11
Installation Overview 11
Install CAN Interface Driver 11
Install PCANopen Magic Pro 11
Install CAN Professional Driver 12
Activate PCANopen Magic Pro 12
Additional Step For PCAN Dongle Users 12
2.2 Setup 12
Chapter 3 – Basic Functionality 15
3.1 Node Selection 15
3.2 Window Access 16
3.3 Trace Recording 16
Start, Stop and Clear 17
Filtering 17
Continuous and Static Recording 18
Absolute and Relative Timestamps 19
Trace Recording Export 19
Message Ordering 19
3.4 SDO Access 20
SDO Upload 20
SDO Download 22
3.5 Network Management 24
3.6 Projects 25
3.7 Reset the CAN Interface 25
3.8 Changing the Baud Rate 25
Chapter 4 – Advanced Functionality 26
4.1 Network Overview 26
4.2 PDO Configuration 27
Enabled 28
CAN ID 28
Name 28
RTR 28
Trans Type and Sync 28
Inhibit 29
Event 29
Map Num 29
Map 1 – Map *n* 29
4.3 Message Transmission 30

Editing a Message Configuration 31
Enabling and Disabling Messages 32
Immediate Transmission 32
4.4 Generate and Restore Node Configurations 33
4.5 Process Data Display 33
4.6 Layer Setting Services 35
Standard LSS 36
MicroLSS 37
4.7 Trace Filtering 37
Application Object 38
Trace Object 39
Chapter 5 – Configuration 40
5.1 Symbolic Support 40
5.2 Network Description 40
Creating the Files 40
Using the Files 41
5.3 Nodes 41
5.4 Messages 42
5.5 Process Data 43
5.6 Default Network Description File 44
5.7 Symbol Loading Order 45
5.8 Preferences 45
Chapter 6 – Command Line 47
6.1 Syntax 47
6.2 Directives 48
NMT 48
SDODOWNLOAD 49
SDOUPLOAD 49
HARDWARE 50
QUIET 51
ODWRITE 51
ODREAD 52
SDOTIMEOUT 53
Appendix A – CXL Reference 54
A.1 File Format 54
Header and Footer 54
XML Tag 54
CXL Tag - Version 54
CXL Tag - Author 54
CXL Tag - Date 54
Number Bases 55
Case 55
Formatting 55
A.2 objectdictionary 55
nodeid 56
entry 56
A.3 network 59
message 59
abortcode 60

errorcode 60
node 61
devicetype 63
vendor 64
A.4 processdata 66
data 66
A.5 comments 69
A.6 Example File 69

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".
italictext	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 DS301 (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 standarized 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 proprietory 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 PCANopen Magic Pro

PCANopen Magic Pro is a professional grade CANopen development tool, indispensable for assisting in the development and debugging of CANopen based networks. Using PCANopen Magic Pro, a CANopen network may be monitored, configured and analyzed. Timings of message transmissions can be observed, allowing comparrison against network specifications. Nodes may be quickly and easily configured and test messages generated with a minimum of effort.

PCANopen Magic Pro is the professional version of PCANopen Magic, also available from PEAK-System Technik. Although there is a small overlap in the features and functionality between the regular and Pro versions, the Pro version contains a large number of enhancements and improvements over the regular version, making it more suitable for professional development assistance.

1.3 PCANopen Magic Pro Features

The following is a list of features in PCANopen Magic Pro. The list is not exhaustive by any means, but does give a good overview of the abilities of PCANopen Magic Pro.

- SDO Upload (read from a node)
  - Display of read data in ASCII♦
  - Display of read data in decimal Integer♦, Unsigned integer♦ and Real
  - Display of read data in Hexadecimal♦
  - Display of read data in TIME OF DAY and TIME DIFFERENCE formats
  - Store read data in a file♦
  - Automatic selection of the most appropriate data type for the entry
  - Manual entry selection♦
  - Entry selection from a configurable Object Dictionary list
- SDO Download (write to a node)
  - Enter data to write in ASCII♦
  - Enter data to write in decimal Integer♦, Unsigned integer♦ and Real
  - Enter data to write in Hexadecimal♦
  - Enter data to write in TIME OF DAY and TIME DIFFERENCE formats
  - Data to write read from a file♦
  - Automatic selection of the most appropriate data type for the entry
  - Manual entry selection♦
  - Entry selection from a configurable Object Dictionary list
- Network Management of all nodes or a single node♦
- Network Scan for available nodes
- Can connect to the same CAN network as other PEAK development tools at the same time
- Automatically remembers window positions and settings, hardware and network configurations
- Transmission of up to 20 configurable messages
  - Select default connection set IDs
  - Manually enter IDs♦
  - Select user defined message IDs
  - Transmit periodically with a customizable period♦
  - Transmit on a keypress♦
  - Transmit on reception of a specific message ID♦
  - Configurable message contents and length♦
  - Able to transmit Remote Transmission Request messages♦
  - Display overview of all messages configured♦
  - Specify descriptive names for messages
- Traffic, transmit and receive indicator lights♦
- Network Management toolbar buttons in every window
- Network Description and Configuration
  - Specify names for nodes
  - Specify Electronic Datasheets for nodes
  - Specify names for messages
- Save Node configurations in Device Configuration Files
- Load Node configurations from Device Configuration Files
- Save all Node configurations in Network Configuration Files
- Load all Node configurations from Network Configuration Files

- Network Description Files
  - Written in XML based CANopen eXchange Language and describe the network
  - Symbolic information from the files are used throughout the windows to allow selection of messages, nodes, etc. by name rather than by ID
  - Allows description of customized node Object Dictionaries
- Network Overview Display
  - Automatically scans the network for nodes♦
  - Provides quick overview of available nodes and basic information♦
  - Continually displays current node state♦ and last error
  - Set the heartbeat producer time of all nodes at once
- Configuration of Layer Setting Services Nodes
  - Automatic detection of single nodes
  - Configuration of Node IDs
  - Configuration of bit timings
- Quick PDO Configuration
  - Automatically scans for PDOs defined on a node
  - Quick and easy configuration of PDO communication parameters
  - Quick and easy configuration of PDO mapping parameters
  - Select default connection set IDs for PDOs
  - Manually enter IDs for PDOs
  - Select user defined IDs for PDOs
  - One click enable and disable a PDO
  - One click enable and disable all PDOs defined on a node
- Configurable Trace Display
  - Stop and start for trace review♦
  - Stores 13000+ messages (0.5 seconds @ 1Mbps with 100% bus load)♦
  - CANopen interpretation of messages allowing easy viewing of network activity♦
  - User defined message names displayed
  - Filtering of messages displayed♦
  - Continuous mode showing each message on the bus♦
  - Static mode showing the last message for each ID on the bus♦
  - Absolute timestamps accurate to 1us
  - Relative timestamps accurate to 1us showing time between consecutive messages
  - Export trace in ascending CSV format for further analysis in Excel, etc.
  - Export trace in descending CSV format for further analysis in Excel, etc.
  - Display of message contents in hexadecimal♦
  - Display of message contents in decimal signed and unsigned
  - Display of message contents in ASCII♦
  - Display of raw message contents
  - Display of error frames with an option to hide them
- Configurable Process Data Display
  - Show any number of process data items
  - Updated on detection of PDOs on the network
  - Configurable graphical displays of data in multiple formats
  - Able to add labels and graphics describing the data being viewed
  - Display graphs of analog data

- Projects
  - Save and load settings stored in project files
  - Start the application with a project on the command line
- Works with all PEAK-System Technik CAN interfaces♦
- Compatible tools can be executed from the user interface
- Monitoring of the last emergency transmitted by the currently selected node♦
- Command line interface for batch programming and end of production line testing♦
- Can run at the same time, on the same CAN network as other compatible tools, including PCANExplorer and PCANView
- Configurable SDO and LSS timeouts
- Reset the CAN interface

♦ This feature is available in the regular PCANopen Magic

1.4 Obtaining Compatible CAN Interfaces

As mentioned in the feature list, all PEAK-System Technik CAN interfaces are supported. Visit www.peak-system.com to locate the nearest distributor.
Note that the regular PCANopen Magic supports the SYSTEC CAN-USB Interface, however PCANopen Magic Pro does not.

Chapter 2 – Installation and Setup

2.1 Installation

Minimum Requirements

The following is a list of the recommended minimum requirements for running PCANopen Magic Pro.

- Pentium III 866MHz
- Windows 95
- Pointing device (mouse, trackball, etc.)
- 5Mb of disk space
- 64Mb of RAM

Installation Overview

Installation is a four part process. The following steps must be completed to install PCANopen Magic Pro:

- Install CAN Interface Driver
- Install PCANopen Magic Pro
- Install CAN Professional Driver
- Activate PCANopen Magic Pro

If you have already installed PCANopen Magic Pro and you are upgrading to a new version, then you normally only need to install PCANopen Magic Pro itself. I.e. the drivers only need to be uninstalled and reinstalled if they have been updated.

The PCANopen Magic Pro installation wizard can combine steps two to four into one easy step.

Install CAN Interface Driver

The CAN driver for the interface is supplied with the interface itself on disk or CD. Alternatively, drivers may be downloaded from the PEAK System Technik web site at www.peak-system.com.

Follow the instructions supplied with the interface to install.

Install PCANopen Magic Pro

To install PCANopen Magic Pro, simply run the installation executable. An installation wizard will guide you through the steps necessary to install the software. When the activation page is shown, enter the activation code you were supplied with from the place you purchased your copy of PCANopen Magic Pro. Enter your name and company name. If you did not receive an activation code, contact the place where you purchased the software.

At one point in the installation wizard you will be given the option of also installing the CAN Professional Driver. If you choose to do this (the CAN Professional driver must be installed to use PCANopen Magic Pro) then a separate installation wizard for the driver will automatically run when PCANopen Magic Pro has finished installing.

Install CAN Professional Driver

The best way to install the CAN Professional driver is to select the option to install it during installation of PCANopen Magic Pro.

Follow the prompts in the installation wizard to install the driver.

Note that the CAN Professional driver installation is separate from the PCANopen Magic Pro installation. Uninstalling PCANopen Magic Pro will not uninstall the CAN Professional driver. Instead you must seperately uninstall the CAN Professional driver if you wish to remove it.

Activate PCANopen Magic Pro

Each copy of PCANopen Magic Pro needs a valid activation code in order to run. An activation code usually consists of around 14 letter and numbers. If you did not receive an activation code with your copy of PCANopen Magic Pro, then contact the place where you purchased your copy and ask for one.

To activate your copy, enter the activation code during the PCANopen Magic Pro installation when prompted. If you are installing over a previous version then you may find that the code has already been filled in for you.

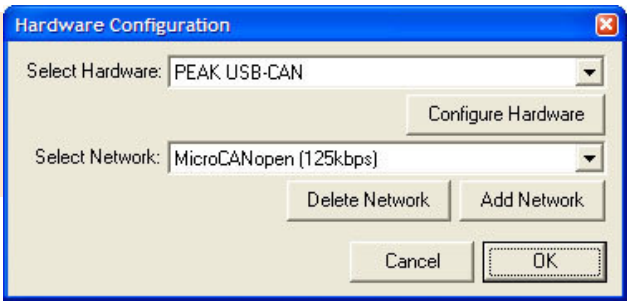
Additional Step For PCAN Dongle Users

Before using the PCAN Dongle interface with PCANopen Magic Pro, 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

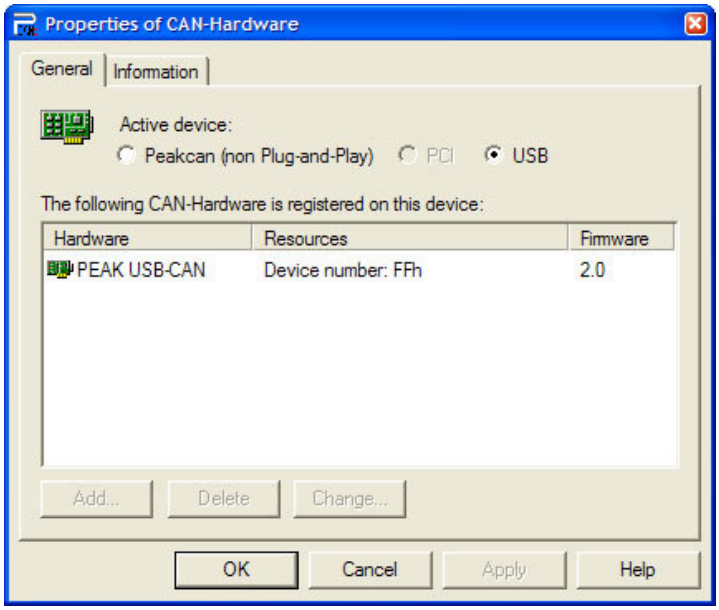
2.2 Setup

Each time you run PCANopen Magic Pro, the hardware configuration window will open.



1. Hardware Configuration Window

Clicking on "Configure Hardware" will display a window that allows configuration of the CAN interface to use.

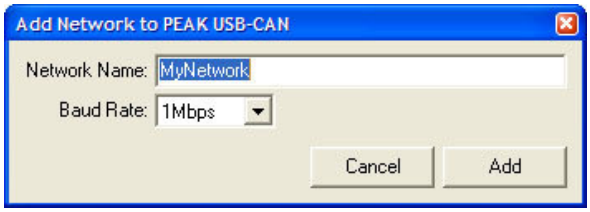


2. CAN Interface Properties Window

Select the category of your interface, for example, PCI for PCI cards, USB for USB interfaces and Peakcan for parallel port (printer port) dongles. Some hardware, such as parallel port dongles must be added by clicking on the "Add" button and entering the settings that apply to your PC.

Make sure your CAN interface is highlighted and click the "OK" button. The Hardware Configuration window should now show your selected interface. In addition no hardware interface may be selected for purely internal CAN bus operation.

Once a CAN interface has been selected, the bottom part of the Hardware Configuration window will contain a drop-down list of currently available networks. For each network the CAN baud rate is shown. If you do not see a network for the baud rate you wish to use, then click on the "Add Network" button. The Add Network window will be displayed.



3. Add Network Window

Enter a name for the network and select a baud rate. Click on the "Add" button. Your new network should now be displayed in the Network Configuration window.

PCANopen Magic Pro automatically remembers networks you add. If you wish to delete a network, then simply select it and click on the "Delete Network" button.

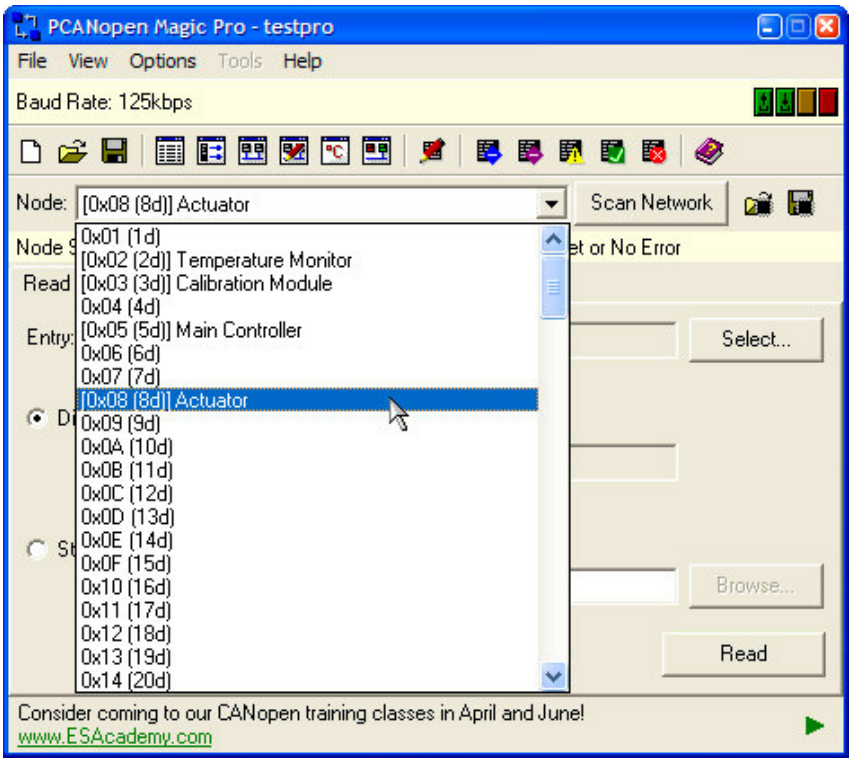
Once you have completed the Hardware Configuration, click on the "OK" button to start using PCANopen Magic Pro.

If you wish to use other PEAK Professional development tools at the same time as PCANopen Magic Pro, then you will need to select the same network in all the applications.

Chapter 3 – Basic Functionality

3.1 Node Selection

All of the features in the main window apply or can apply to the currently selected node. To select a specific node, simply choose it from the drop down list at the top of the main window. Once selected, SDO Upload, SDO Download and Network Management messages set to a single node will all operate on the selected node.



4. Node Selection

Initially the drop-down list contains all 127 possible nodes, however often specific Node IDs may not be known or to hand. Clicking on the "Scan Network" button will scan the network for currently operating nodes and reduce the list to shown only those nodes that were found. If no nodes were found then the list will be empty.

In order for a node to be found using the Network Scan it must implement the mandatory Object Dictionary entry 1000H – Device Type.

If a name has been given to a node then the name will be displayed in the drop-down list alongside the Node ID. See the Configuration chapter for details on how to give names to nodes.

Below the node selection area there is a cream colored band that shows basic information about the currently selected node. If a heartbeat or node guarding message has been transmitted by the device then the state the node reported will be displayed. In addition, if the node has transmitted any emergency messages then the last emergency reported will also be displayed. This information is only updated when the relevant messages are seen on the CAN bus by PCANopen Magic Pro, and is reset whenever a new node is selected.

The selected node may also be optionally used to filter the trace display to show only messages that are coming from or going to the selected node.

3.2 Window Access

The user interface of PCANopen Magic Pro consists of the main window and a set of additional monitoring and configuration windows. Some or all of these additional windows may be open at the same time to provide a wide range of information. In order to ease navigation around the windows, each one contains a set of toolbar buttons that will open the other windows. Hover the pointer over the icons to learn which icon to use for a specific window.

The windows may also be opened by choosing the relevant item on the View menu in the main window.

The positions and sizes of the windows, along with the column sizes in the windows are remembered automatically when PCANopen Magic Pro is closed.

3.3 Trace Recording

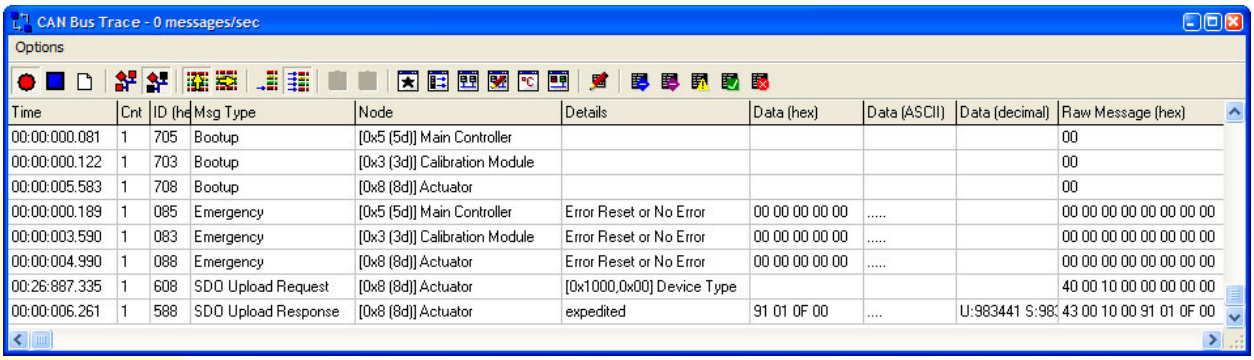
The Trace window shows the current messages on the bus as they appear in a highly configurable format. The default configuration is for each message to be displayed as it is seen, with the newest message being inserted at the bottom of the trace. The trace may be configured using the toolbar buttons or the items on the Options menu.

The settings are automatically remembered when PCANopen Magic Pro is closed.

Approximately the last 13000 messages detected on the bus may be stored in the trace recording for review and export.

The title bar of the Trace window while running shows the number of messages per second detected on the bus.

If a message has been given a name, then the name will be shown in the trace window. See the Configuration chapter to learn how to name messages.



Time	Cnt	ID (he	Msg Type	Node	Details	Data (hex)	Data (ASCII)	Data (decimal)	Raw Message (hex)
00:00:000.081	1	705	Bootup	[0x5 (5d)] Main Controller					00
00:00:000.122	1	703	Bootup	[0x3 (3d)] Calibration Module					00
00:00:005.583	1	708	Bootup	[0x8 (8d)] Actuator					00
00:00:000.189	1	085	Emergency	[0x5 (5d)] Main Controller	Error Reset or No Error	00 00 00 00 00	.....		00 00 00 00 00 00 00 00
00:00:003.590	1	083	Emergency	[0x3 (3d)] Calibration Module	Error Reset or No Error	00 00 00 00 00	.....		00 00 00 00 00 00 00 00
00:00:004.990	1	088	Emergency	[0x8 (8d)] Actuator	Error Reset or No Error	00 00 00 00 00	.....		00 00 00 00 00 00 00 00
00:26:887.335	1	608	SDO Upload Request	[0x8 (8d)] Actuator	[0x1000,0x00] Device Type				40 00 10 00 00 00 00 00
00:00:006.261	1	588	SDO Upload Response	[0x8 (8d)] Actuator	expedited	91 01 0F 00	....	U:983441 S:98	43 00 10 00 91 01 0F 00

5. Trace Window

Start, Stop and Clear

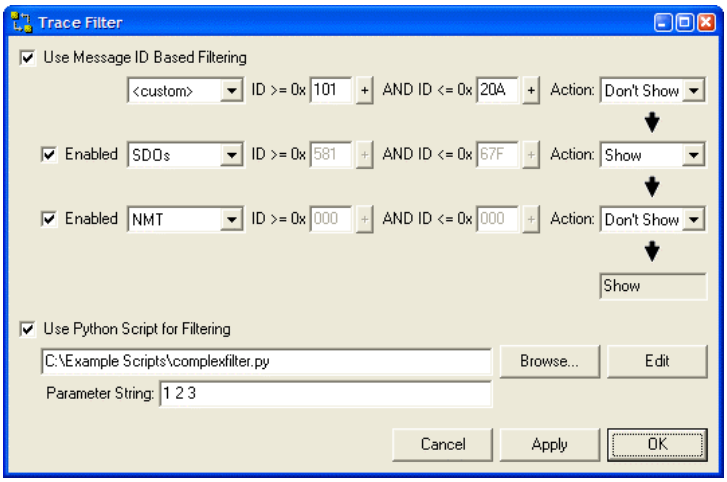
The trace recording may be started or stopped at any time. For performance reasons it is not possible to view all of the messages recorded while the trace recording is running. To view all recording messages, stop the trace. Once stopped a scrollbar will appear allowing the trace to be scrolled back. Once stopped the trace recording may be exported in CSV format.

The trace may be cleared at any time without stopping, simply by clicking on the Clear button or choosing "Clear Trace" from the Options menu.

Filtering

The trace recording may be set to either show all messages, or only show messages sent to or from the currently selected node. The currently selected node is the node chosen in the main window.

In addition more advanced filtering is possible by choosing "Configure Filter…" from the Filter menu.



6. Trace Filter Window

To filter based on a range of message IDs, check the option "Use Message ID Based Filtering". Select either a range of messages to filter (SDOs, PDOs, etc.) or choose a custom range. When a custom range is selected the start and end IDs of the range can be entered, either by typing the IDs into the boxes or clicking on the "+" buttons and choosing the IDs.

When a message matches one of the enable message ID ranges, the message can be either shown or not shown. If a message does not match the first range then it will be compared with the second range. If it does not match the second range then it will be compared with the third range. If a message does not match any ranges then it will be shown in the trace window.

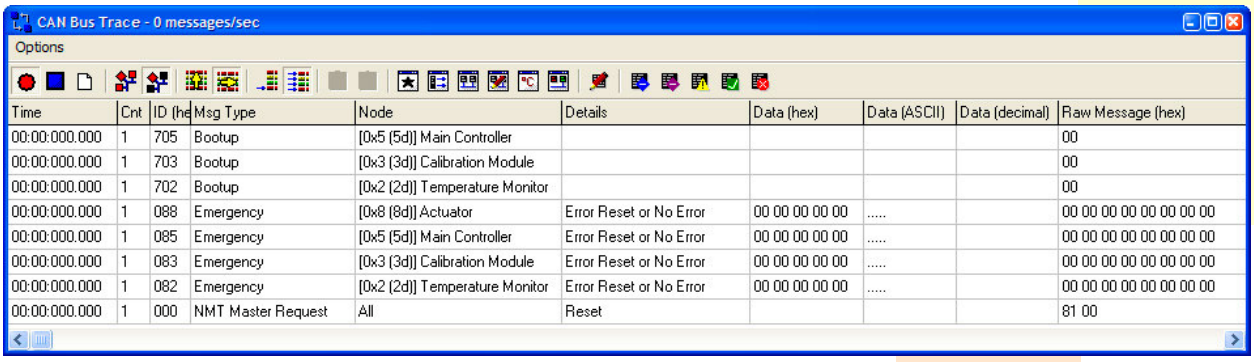
More complex filtering is possible using a Python script. For details of this option please refer to the trace filtering section in Chapter 4.

Continuous and Static Recording

The trace recording may operate in either Continuous mode or Static mode and may be swtiched between either mode at any time. Note that the entire trace recording is lost when modes are switched.

In Continuous mode, each message is shown in the trace recording as it is detected, with the newest message inserted at the bottom of the window. The timestamps shown for each message are either relative to the start of the trace recording or relative to the previous message received, depending on which timestamp mode the trace is in.
The Continuous mode is useful for analyzing sequences of messages and the time delays between any two messages.

In Static mode, each unique message ID is shown in the trace recording, with one line per message ID. As messages are detected, they overwrite in the trace recording any older messages with the same ID. The trace display is sorted into message ID order. The timestamps shown for each message ID are either relative to the start of the trace recording or relative to the previous message received with the same ID, depending on which timestamp mode the trace is in.
The Static mode is useful for analyzing periodic or synchronous transmissions and the time delays between transmissions.



Time	Cnt	ID (he	Msg Type	Node	Details	Data (hex)	Data (ASCII)	Data (decimal)	Raw Message (hex)
00:00:000.000	1	705	Bootup	[0x5 (5d)] Main Controller					00
00:00:000.000	1	703	Bootup	[0x3 (3d)] Calibration Module					00
00:00:000.000	1	702	Bootup	[0x2 (2d)] Temperature Monitor					00
00:00:000.000	1	088	Emergency	[0x8 (8d)] Actuator	Error Reset or No Error	00 00 00 00 00	.....		00 00 00 00 00 00 00 00
00:00:000.000	1	085	Emergency	[0x5 (5d)] Main Controller	Error Reset or No Error	00 00 00 00 00	.....		00 00 00 00 00 00 00 00
00:00:000.000	1	083	Emergency	[0x3 (3d)] Calibration Module	Error Reset or No Error	00 00 00 00 00	.....		00 00 00 00 00 00 00 00
00:00:000.000	1	082	Emergency	[0x2 (2d)] Temperature Monitor	Error Reset or No Error	00 00 00 00 00	.....		00 00 00 00 00 00 00 00
00:00:000.000	1	000	NMT Master Request	All	Reset				81 00

7. Static Trace Recording

Absolute and Relative Timestamps

Each message in the trace recording has a timestamp with an accuracy of 1us (PCI, ISA and Parallel Port interfaces) or 42us (USB interface). The timestamps may be displayed in Absolute mode or Relative mode.

In Absolute mode the timestamp is the time the message was received since the start of the trace recording.

In Relative mode the timestamp will be the time since the last message was received. If the trace is in Continuous mode then this will be relative to the last message seen on the bus. If the trace is in Static mode, then this will be relative to the last message seen on the bus with the same message ID.

Technical footnote: On rare occasions while in Absolute mode, a message may be received and timestamped by the CAN driver while the trace was stopped, but by the time the message was made available to PCANopen Magic Pro the trace was started by the user. In this situation the timestamp of the message will actually be a small negative value. In order to reduce user confusion, these messages are adjusted to have a timestamp of zero. Due to a limitation in Windows Performance Counters, once every 10-20 minutes while at 100% bus load and using a baud rate of 1Mbps, there may be observed a 2.7 second jump in the timestamps. This should be taken into account when relying heavily on the timestamp information for debugging.

Trace Recording Export

While the trace recording is stopped it may be exported in CSV format for processing or display in other applications. The export may be generated in ascending or descending format.

When displaying the CSV files in Excel it was found that Excel wanted to automatically convert some of the timestamps and message IDs to a date or real format, and would therefore display them incorrectly. To avoid this, timestamps and message IDs are prefixed in the file with a ''' character. This may be easily stripped out by any custom processing applications.

Message Ordering

In order to ensure that the order of messages in the trace recording exactly matches the order of the messages on the CAN bus, messages transmitted by PCANopen Magic Pro are not simply inserted into the trace recording. Instead the messages are transmitted onto the bus and then detected by the receive portion of PCANopen Magic Pro just like any other message. I.e. there is no direct connection in PCANopen Magic Pro between the transmit code and the receive code or between the transmit code and the trace recording code. This feature helps to ensure that the trace recording always shows messages in order, which can be cruicial in debugging work.

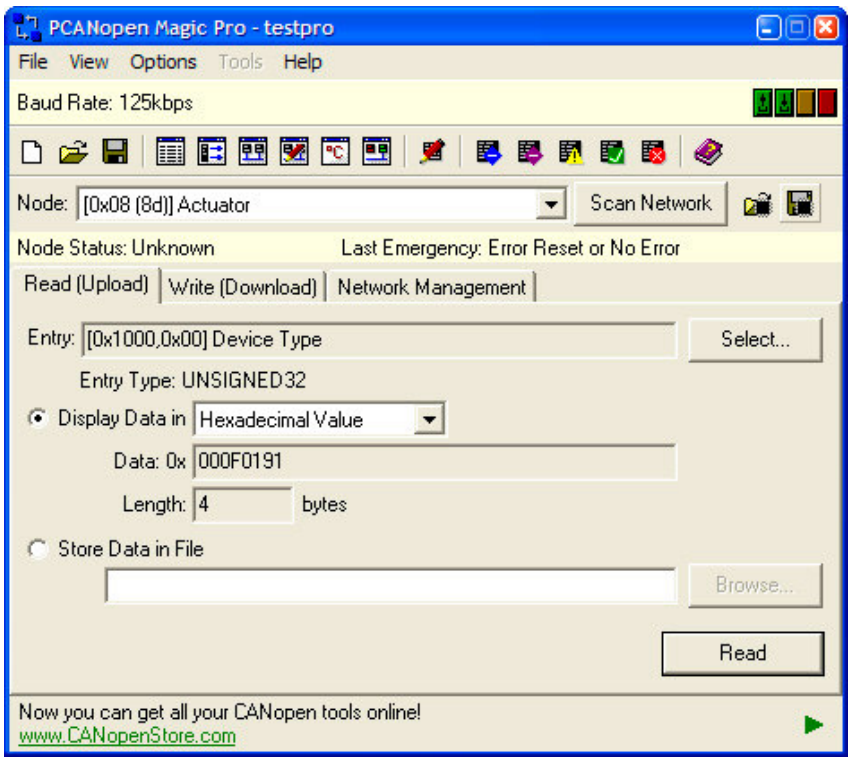
3.4 SDO Access

Individual SDO Uploads and SDO Downloads may be performed in the main window. The lower part of the main window is divided up into sections, with the first two being the SDO access sections. Clicking on the SDO Upload or SDO Download tabs will display the settings.

The SDO accesses are always performed on the currently selected node, which is selected using the drop-down list in the main window.

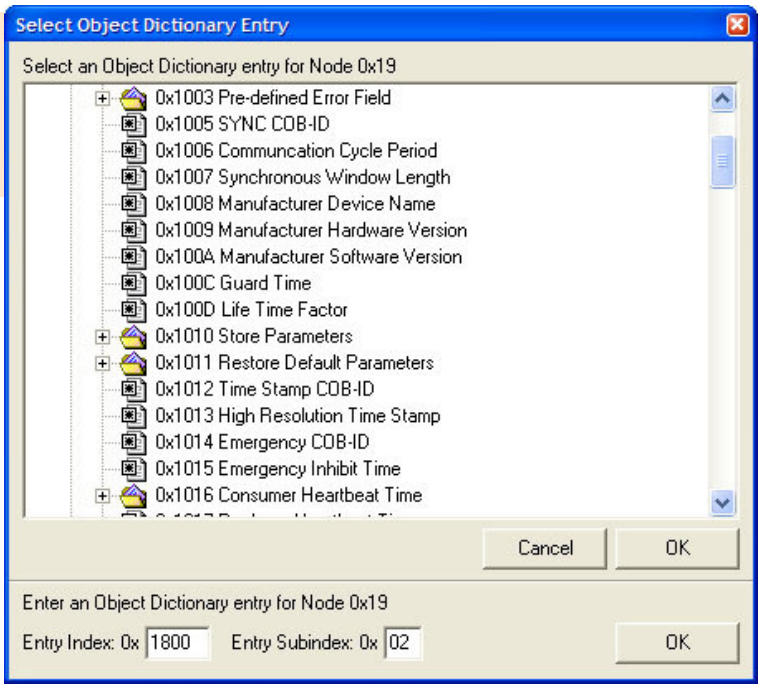
SDO Upload

The SDO Upload section displays the Object Dictionary entry to read followed by the read options.



8. Main Window – SDO Upload Section

Clicking on the "Select…" button will open the Entry Selection Window, allowing an Object Dictionary entry to be selected.



9. Entry Selection Window

The top part of the Entry Selection Window shows the Object Dictionary entries for the currently selected node. Simply double-click on the desired entry to read or select it and click on the "OK" button.

The Object Dictionary of a node may be defined by either specifiying an Electronic Datasheet for the node or by defining the Object Dictionary for the node in a Network Description file. See the Configuration chapter for details.

If a node does not have an Object Dictionary defined, the a default Object Dictionary will be shown.

The bottom part of the Entry Selection Window allows the Index and Subindex of the entry to be entered. This is useful if you wish to quickly access an entry that is not shown in the top part of the window.

When using this window, ensure that you click on the relevant "OK" button for the section you are using.

Once the entry to read has been selected, the choice can be made to either display the read data or store it in a file.

When displaying the read data, the data type may be selected. All the standard CANopen data types are supported. When an entry is selected the closest matching data type will be automatically selected.

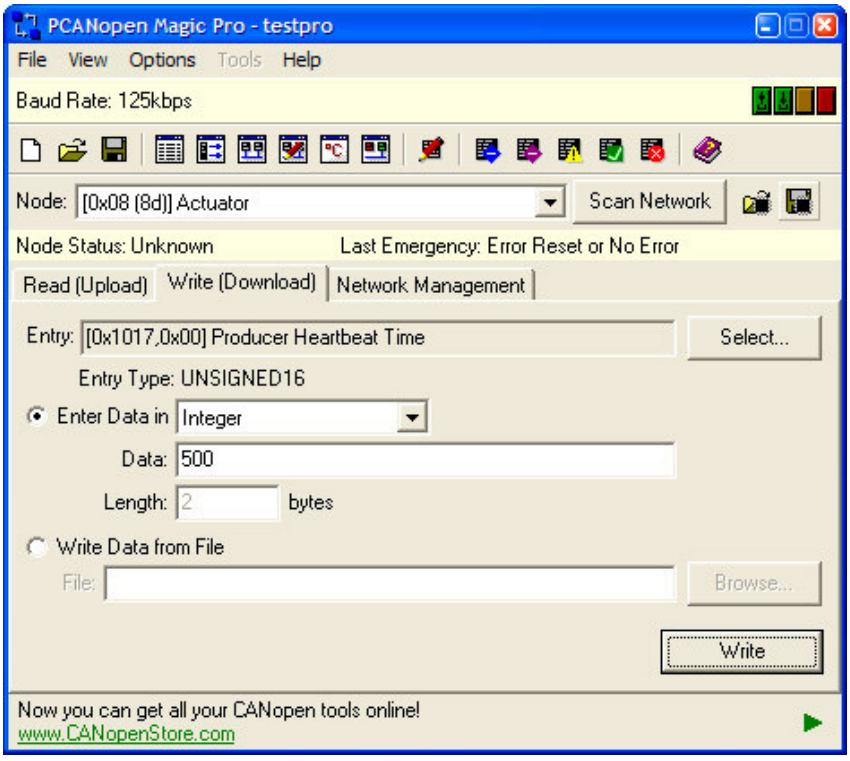
To store the read data in a file, click on the "Browse…" button and select a location to save to.

Clicking on the "Read" button will perform the read. The SDO messages used to implement the read may be viewed in the trace window.

If the option to display the read data was selected, and the size of the read data was larger than 16 bytes, then the first 16 bytes will be displayed along with a button to view more. Clicking on the button will open a display window that will show all the read data.

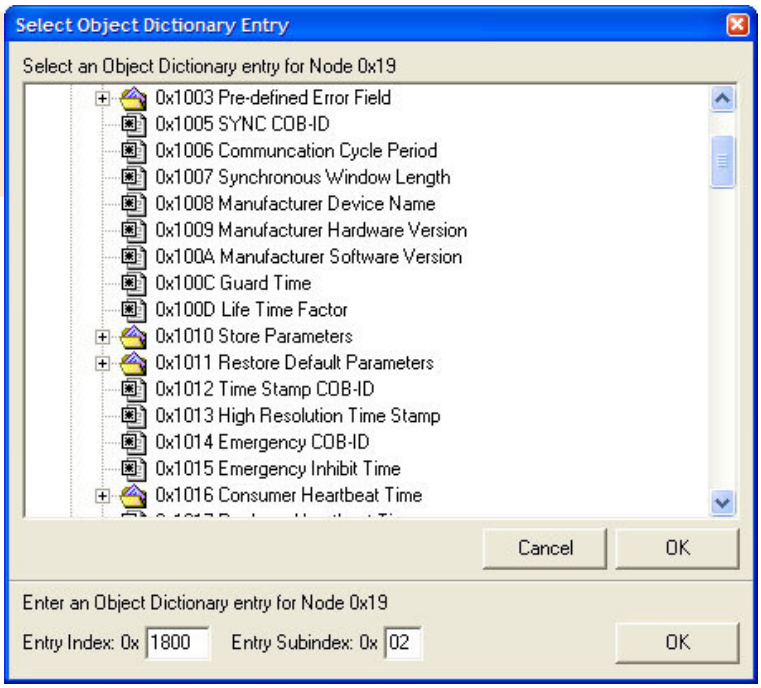
SDO Download

The SDO Download section displays the Object Dictionary entry to write to followed by the write options.



10. Main Window – SDO Download Section

Clicking on the "Select…" button will open the Entry Selection Window, allowing an Object Dictionary entry to be selected.



11. Entry Selection Window

The top part of the Entry Selection Window shows the Object Dictionary entries for the currently selected node. Simply double-click on the desired entry to write to or select it and click on the "OK" button.

The Object Dictionary of a node may be defined by either specifiying an Electronic Datasheet for the node or by defining the Object Dictionary for the node in a Network Description file. See the Configuration chapter for details.

If a node does not have an Object Dictionary defined, then a default Object Dictionary will be shown.

The bottom part of the Entry Selection Window allows the Index and Subindex of the entry to be entered. This is useful if you wish to quickly access an entry that is not shown in the top part of the window.

When using this window, ensure that you click on the relevant "OK" button for the section you are using.

Once the entry to write to has been selected, the choice can be made to either enter the data directly or read it from a file.

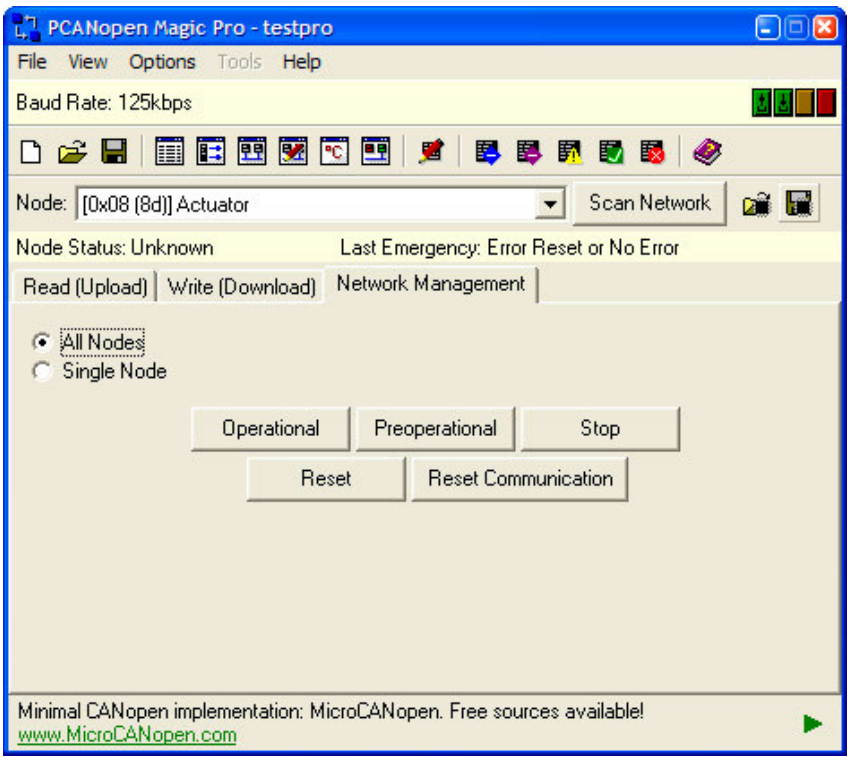
When entering the data directly, the data type to enter the data in may be selected. All the standard CANopen data types are supported. When an entry is selected the closest matching data type for the entry is automatically selected along with the correct length in bytes. Enter the data in the relevant format. For the TIME OF DAY and TIME DIFFERENCE types, a hint on the correct format is displayed next to the box where the data is entered.

To read the data from a file, click on the "Browse…" button and select the file to use.

Clicking on the "Write" button will perform the write. The SDO messages used to implement the write may be viewed in the trace window.

3.5 Network Management

Network Management may be performed in the Main window. The lower part of the Main window is divided up into sections, with one section being for Network Management. Click on the Network Management tab to access the feature.



12. Main Window – Network Management Section

Network Management messages may be transmitted immediately by clicking on the buttons. The messages may be sent to all nodes or a single node. When Single Node is selected, the messages are sent to the currently selected node, which is selected using the drop-down list in the main window.

The Network Management messages may be viewed in the Trace window.

3.6 Projects

All settings may be saved into project files and reloaded at a later date, making it easy to share settings with other users and work on multiple projects at once. Three toolbar buttons and entries on the File menu allow a new project to be created with default settings, a project to be opened and a project to be saved. If creating a new project or opening a project and there are unsaved settings, then a confirmation will be requested to ensure settings are not accidentally lost.

The name of the current project is shown in the title bar of the main window.

The application can be started on the command line with a specific project, simply by adding the path to the project as follows.

```
C:\>PCANopenMagicPro.exe E:\test\testproject.pcm
```

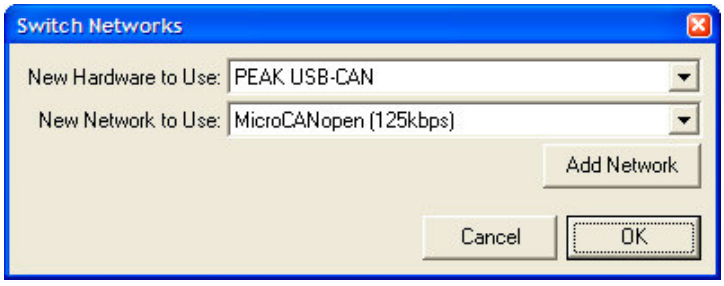
3.7 Reset the CAN Interface

If PCANopen Magic Pro is used at a different baud rate to a node, it is possible for the CAN interface to become stuck constantly transmitting error frames. In this type of situation it is useful to be able to reset the CAN interface.

To reset the CAN interface choose "Reset CAN Interface" from the Options menu.

3.8 Changing the Baud Rate

PCANopen Magic Pro uses the concept of networks. Each network has a different baud rate and any network can be used at any time. In order to switch baud rates the network must be changed.



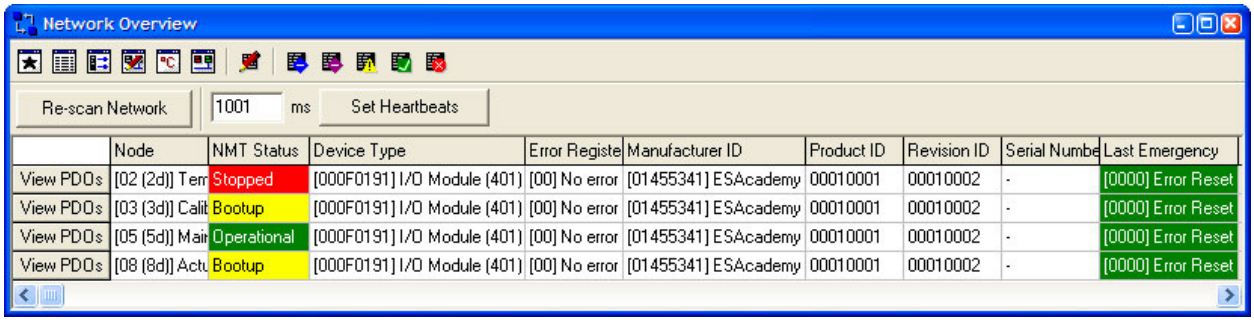
13. Switch Networks Window

To change to a new network, choose "Switch Networks…" from the Options menu. Select the CAN interface to use and the network. To add a network click on the "Add Network" button. When "OK" is clicked PCANopen Magic Pro will immedately switch to the selected CAN interface and network.

Chapter 4 – Advanced Functionality

4.1 Network Overview

The Network Overview window displays a brief overview of all the nodes found on the bus. When the window is opened the network is immediately scanned for nodes. The scan attempts to access Object Dictionary entry 1000H of each possible node, therefore each node must implement this mandatory entry if it is to detected and displayed in the Network Overview window. Once detected additional information about the node is read and displayed, such as the Identity entry at 1018H. The details of nodes found are displayed on separate lines. The network may be re-scanned at any time by clicking on the "Re-scan Network" button.



14. Network Overview Window

There are two different types of information displayed in the Network Overview window for each node:

- Static Data – this is data that is read in only when the network is scanned for nodes. The network may be re-scanned at any time by clicking on the "Re-scan Network" button.
- Real Time Data – this is data that is continually updated based on messages detected on the bus. Real Time Data is only shown for nodes that have been found during a network scan.

Real Time data that is displayed includes:

- NMT Status of the node – this is updated when bootup, heartbeat or node guarding messages are transmitted by the node. The status is also color coded, using Green for Operational, Red for stopped, etc.
- Last Emergency of the node – this is updated when the node transmits an emergency message. The Network Overview window shows details of the emergency.

The Network Overview window also allows the setting of the heartbeat producer time for all nodes at once. Simply enter the desired time into the box and click on "Set Heartbeats". A high speed expedited write will be performed for all nodes on the network.

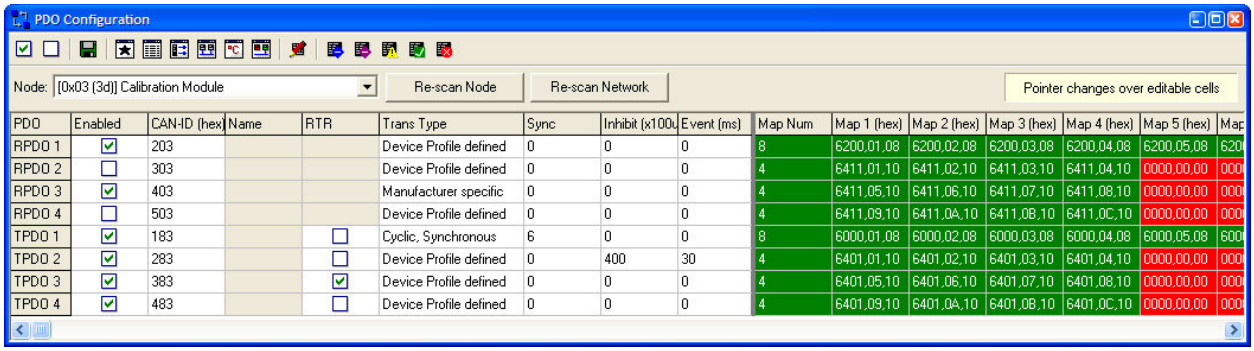
4.2 PDO Configuration

The changing of PDO Configurarations in a node that supports changes is always a multi-step process. To make any change first the PDO must be disabled. Once the change is made the PDO may be re-enabled. If a change is made to the mapping of data into the PDO, then the number of valid mapping entries must first be set to zero before the change can be made. Each step requires a specific Object Dictionary entry to be accessed and either read or written. In addition, the steps must be completed in a specific order.

Most (if not all) CANopen development tools simply allow you to read and write to Object Dictionary entries that the user selects, much like the main window of PCANopen Magic Pro works. In order to configure PDOs the user must know:

- The steps to complete for the desired change in configuration
- The values to write in each step and therefore the meaning of specific bits for some of the values.

The PDO Configuration window simplifies the process of configuring PDOs and reduces the user interface to little more than a point and click system. Upon opening the window the network is automatically scanned for nodes. The network may be re-scanned at any time by clicking on the "Re-scan Network" button. All detected nodes will be shown in the drop-down list. Choosing a node will scan that node for currently implemented PDOs. The node may be re-scanned for PDOs at any time by clicking on the "Re-scan Node" button. Once scanned the found PDOs will be displayed in a configuration table, with each PDO occupying one row. Scanning of RPDOs stops at the first RPDO not found in the Object Dictionary of the node. The same is true for TPDOs.



PDO	Enabled	CAN-ID (hex)	Name	RTR	Trans Type	Sync	Inhibit (x100u	Event (ms)	Map Num	Map 1 (hex)	Map 2 (hex)	Map 3 (hex)	Map 4 (hex)	Map 5 (hex)	Map
RPDO 1	☑	203			Device Profile defined	0	0	0	8	6200,01,08	6200,02,08	6200,03,08	6200,04,08	6200,05,08	620
RPDO 2	☐	303			Device Profile defined	0	0	0	4	6411,01,10	6411,02,10	6411,03,10	6411,04,10	0000,00,00	000
RPDO 3	☑	403			Manufacturer specific	0	0	0	4	6411,05,10	6411,06,10	6411,07,10	6411,08,10	0000,00,00	000
RPDO 4	☐	503			Device Profile defined	0	0	0	4	6411,09,10	6411,0A,10	6411,0B,10	6411,0C,10	0000,00,00	000
TPDO 1	☑	183		☐	Cyclic, Synchronous	6	0	0	8	6000,01,08	6000,02,08	6000,03,08	6000,04,08	6000,05,08	600
TPDO 2	☑	283		☐	Device Profile defined	0	400	30	4	6401,01,10	6401,02,10	6401,03,10	6401,04,10	0000,00,00	000
TPDO 3	☑	383		☑	Device Profile defined	0	0	0	4	6401,05,10	6401,06,10	6401,07,10	6401,08,10	0000,00,00	000
TPDO 4	☑	483		☐	Device Profile defined	0	0	0	4	6401,09,10	6401,0A,10	6401,0B,10	6401,0C,10	0000,00,00	000

15. PDO Configuration Window

Any item in any box may be edited at any time. The pointer changes shape when moved over boxes to indicate the type of editing that may be performed. Once changed the PDO will be automatically configured using multiple SDO accesses. If a change is made to a PDO that is enabled, then the PDO will not be re-enabled after making the change, instead it

must be manually re-enabled. This allows multiple changes to be made to a PDO without the risk of a partially configured PDO being used by the node.

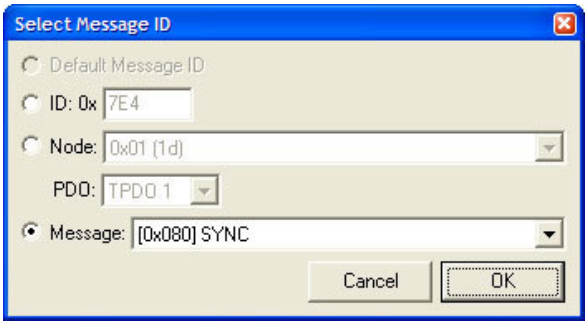
Each column is described:

Enabled

Click on the checkbox to either enable (check) or disable (uncheck) the PDO. Two checkbox buttons are provided on the toolbar to enable or disable all PDOs at once.

CAN ID

Click to change the ID currently be used for the PDO. When the box is clicked on a small button will appear in the box containing a "+" sign. Click on the button to confirm that you wish to change the ID of the PDO. The Select ID dialog window will open allowing selection of a new ID using various methods. The default ID for the PDO may selected along with default IDs for PDOs of other nodes. Also the ID may be entered manually or selected from a list of defined messages (see the Configuration chapter for details on how to define messages).



16. Select ID Dialog Window

Name

This column displays a name for the PDO if it has one. See the Configuration chapter for details on how to define messages.

RTR

Click to enable (check) or disable (uncheck) the RTR flag on the PDO. This option is only available for Transmit PDOs and normally allows their transmission to be requested by other node with an RTR request.

Trans Type and Sync

These two columns allow the selection of the PDO transmission type. Choose the desired type from the drop-down list, which appears when the transmission type box is clicked. If "Cyclic, synchronous" is selected then enter the number of SYNC messages into the Sync column. Changing the value in the Sync column will select the "Cyclic, synchronous"

transmission type automatically if the value entered is within the desired range. For all other transmission types the Sync column should be zero.

Inhibit

Enter the inhibit time of the PDO in multiples of 100µs.

Event

Enter the event time of the PDO in milliseconds.

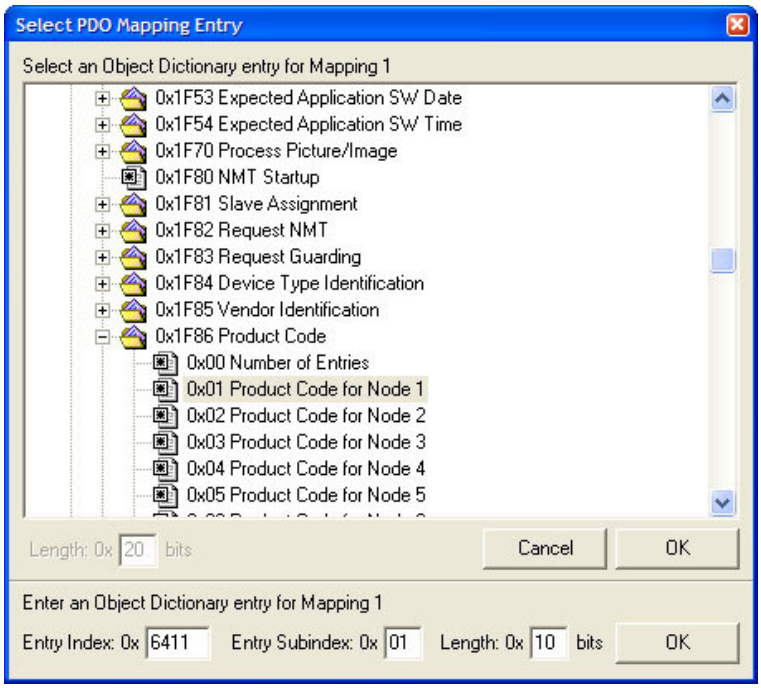
Map Num

Clicking on the box allows selection of the number of mapping entries from a drop-down list. Selecting a value of zero disables the entire mapping. Once selected the mapping entries themselves, which are displayed on the right, might change colors. Green mapping entries are entries that will be used if the PDO is enabled. Red mapping entries are entries that will be ignored if the PDO is enabled.

Map 1 – Map *n*

Each box shows a mapping entry for the PDO. Map 1 is the first mapping entry, Map 2 is the second, and so on. The Object Dictonary entry for the data contained in the mapping is shown, along with the length of the entry and the name of the data (if available – see the Configuration chapter for details on defining the Object Dictionary of a node).

To change a mapping entry first click on the box. A small button containing a "+" sign will appear in the box. Click on the button to confirm you wish to change the entry and the Select PDO Mapping Entry Dialog Window will appear.



17. Select PDO Mapping Entry Dialog Window

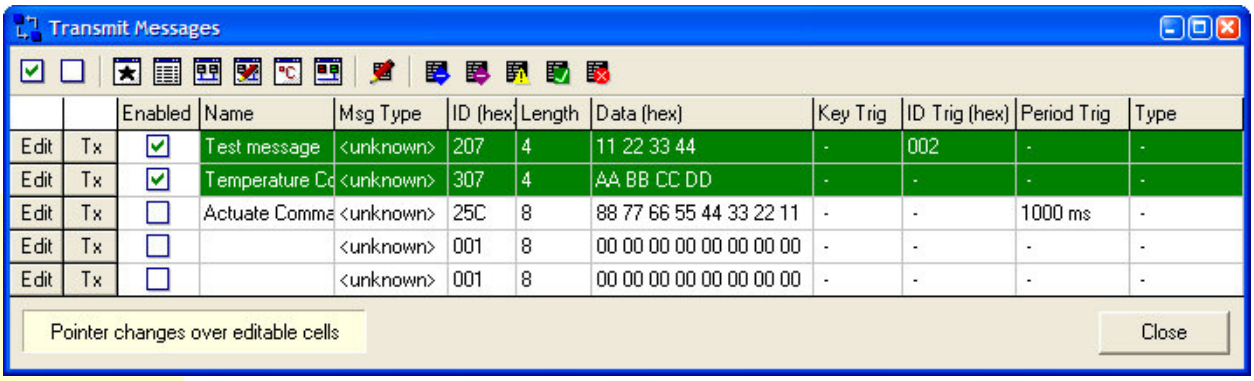
Either select an entry from the node's Object Dictionary in the top part of the window, or manually enter the Object Dictionary entry Index, Subndex and Length in the bottom part of the window.

Once a new entry has been selected the PDO will be reconfigured and the PDO Configuration window will update to reflect the current mappings.

4.3 Message Transmission

PCANopen Magic Pro can be configured to transmit up to 20 CAN messages. The CAN messages may have any message ID, length and contents desired. The messages are transmitted when a trigger event occurs, and several different triggers may be used, either individually or combined.

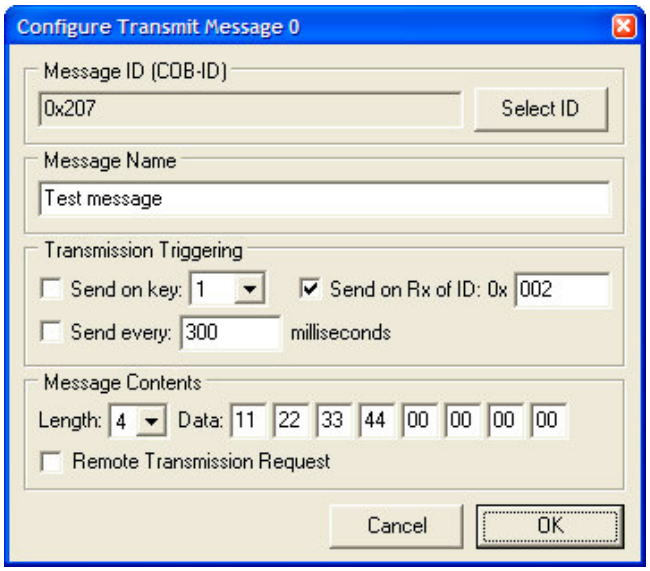
The Transmit Messages window contains an overview of all currently configured messages. Each message is shown on one line and is highlighted in green if currently enabled. When the pointer is moved over the table it changes shape to indicate which boxes may be edited by clicking.



18. Transmit Messages Window

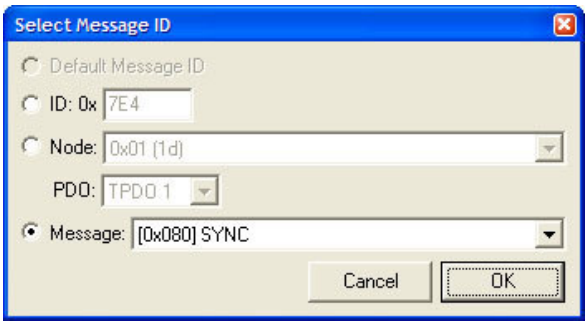
Editing a Message Configuration

Clicking on the "Edit" button, shown at the start of each row, will allow the message configuration to be edited. The Configure Message dialog window will be displayed.



19. Configure Message Dialog Window

To select the ID to be used for the message, click on the "Select ID" button. This will open the Select ID dialog window. The default ID for the PDO may selected along with default IDs for PDOs of other nodes. Also the ID may be entered manually or selected from a list of defined messages (see the Configuration chapter for details on how to define messages).



20. Select ID Dialog Window

The following methods of triggering the transmission of the message are available:

- Send when a specific key is pressed. Check to enable the trigger and select the desired key from the drop-down list. Pressing that key while using PCANopen Magic Pro will trigger the transmission of the message.
- Send when a specific message is detected on the bus. Check to enable the trigger and enter the desired ID of the message to detect.
- Periodic transmission. Check to enable the trigger and enter the delay between transmissions in milliseconds.

Each message may use one or more triggers. If no triggers are used then the message may only be transmitted by clicking on the "Tx" button in the Transmit Messages window.

Select the number of bytes of data the message contains from the drop-down list and enter the message contents in hexadecimal. If the message should have the RTR flag set then check the Remote Transmission Request option.

Each message may be given a descriptive name to help differentiate between the messages in the transmit list.

Enabling and Disabling Messages

Messages may be enabled by clicking on the checkbox in the Enabled column. In the toolbar at the top of the window there are two checkbox type buttons that may be used to enable and disable all messages in one go.

Immediate Transmission

Clicking on the "Tx" button, shown towards the start of each row, will cause that message to be immediately transmitted onto the bus.

4.4 Generate and Restore Node Configurations

If an Electronic Datasheet has been specified for a node, then the node's current configuration may be generated by PCANopen Magic Pro into a Device Configuration File and restored at a later time.

To generate the configuration of the currently selected Node, choose "Generate DCF from Node" from the File menu. Select a location for the DCF file to be created and click on "OK". The node will be read and the current value of Object Dictionary entries stored in the DCF file.

If a node does not have an Electronic Datasheet associated with it in the Network Configuration, then it will not be possible for the configuration to be generated.

Object Dictionary entries which are write only are not read. Also, Object Dictionary entries which are marked as "Refuse Read on Scan" will not be read.

If the node implements a DOMAIN entry, then the data from that entry will be stored in a separate file according to the Device Configuration File specification. The filename of the binary file created will be constructed as follows:

> DCF File Name followed by entry index in hexadecimal followed by "sub" followed by the subindex in hexadecimal followed by "Data.bin".

For example, if Object Dictionay entry 2000H, subindex 03H was a DOMAIN, and the DCF file being created was called "foo.dcf", then the data file created would be called "foo2000sub3Data.bin".

To restore the configuration of a Node from a DCF file, first select the node from the drop down list in the main window. Next choose "Restore DCF to Node" from the File menu. The configuration will be read in and written to the node using SDOs. Object Dictionary entries which are read only will not be written. Also, Object Dictionary entries which are marked as "Refuse Write on Download" will not be written.

The generate and restore features may also be accessed from the two buttons to the right of the "Scan Network" button in the main window.

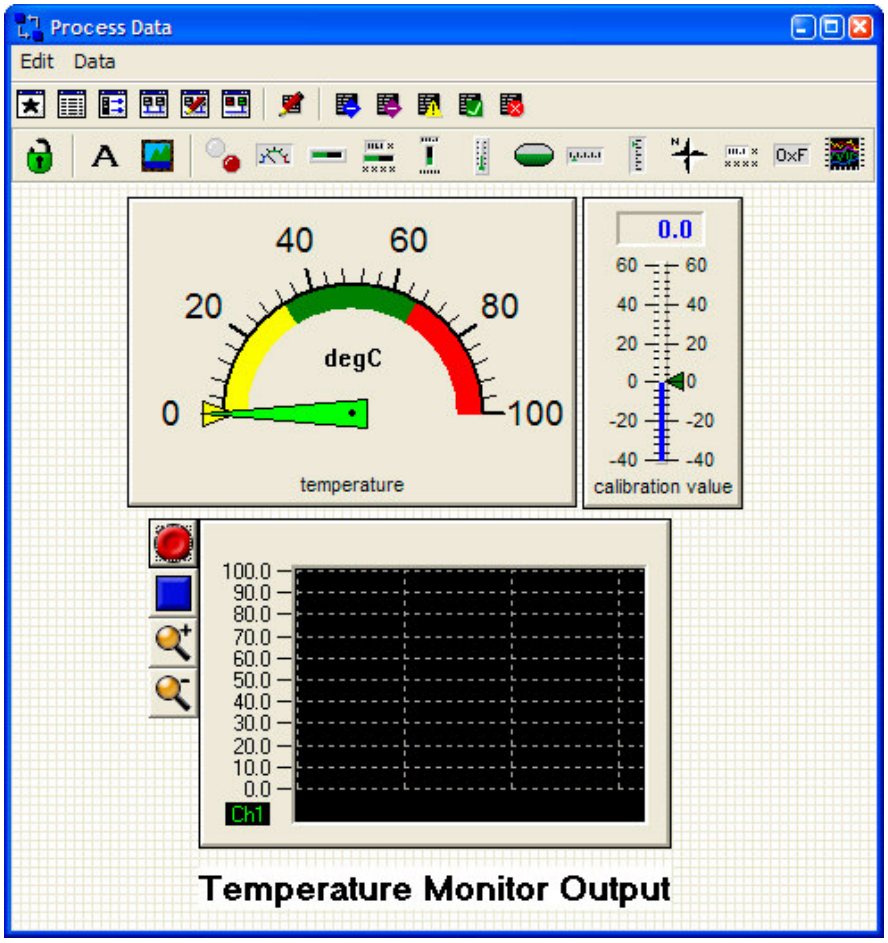
It is also possible generate and restore DCFs for all nodes on the network at once. To do this choose "Restore NCF to Network" or "Generate NCF from Network" from the File menu. Networks are stored in a Network Description File, which is a list of DCFs seperated by "[DCF *nodeid*]" where *nodeid* is the ID of the node in hexadecimal prefixed with "0x".

4.5 Process Data Display

The display of actual process data along with units provides an powerful and easy method for debugging nodes on the network and observing their behavior. By showing the process

data, a level of abstraction above the CANopen level is achieved, allowing the ability to focus on node functionality and operation.

The Process Data window shows a representation of defined process data entries, using graphical displays called "controls", and may be opened by clicking on the process data toolbar button or choosing "Process Data..." from the View menu.



21. Process Data Window

Process data entries are defined either in the Network Description File or under the "Process Data" tab in the Network Configuration window.

Graphical controls may be created anywhere in the window, and mutliple controls may be used, even for the same process data entry.

To create a control click on the toolbar button for that control and click in the window at the position the control should appear.

To select a control simply click on it. A yellow and black border will appear. Moving the pointer over the border and clicking and dragging allows controls to be moved. Moving the

pointer over the black box at the bottom right corner of the control and clicking and dragging allows the control to be resized.

To copy a control, right click over the control and choose "Copy". Click in the window where the copied control should appear, right click and choose "Paste".

To delete a control, right click over the control and choose "Delete".

The copy, control and delete functions are also available on the Edit menu and apply to the currently selected control.

To restore a resized control to the original size, right click on the control and choose "Restore to Default Size".

To configure a control either right click over the control and choose "Configure…", or select the control and choose "Configure…" from the Edit menu.

The configuration window allows various aspects of the control to be changed. Most of these are self explanatory. To select a process data entry for the control, click on the "Process Data" tab and select the entry from the drop-down list. The "Manage Process Data Entries" button provides a short cut to the relevent section of the Network Configuration window.

Once a process data entry has been associated with a control, it will begin showing the current value immediately. Initially this will be zero for unseen data. The Process Data window is periodically updated.

The grid may be turned on or off and the size of the grid may be changed. This is achieved by changing the PCANopen Magic Pro preferences (choose "Preferences…" from the Options menu).

Text labels may be added to the window, just like controls. They do not show data and are ment for labelling purposed. The font, size and text may be changed.

Images may be added to the window, and are similar to labels in that they do not show data but are used for labelling purposes.

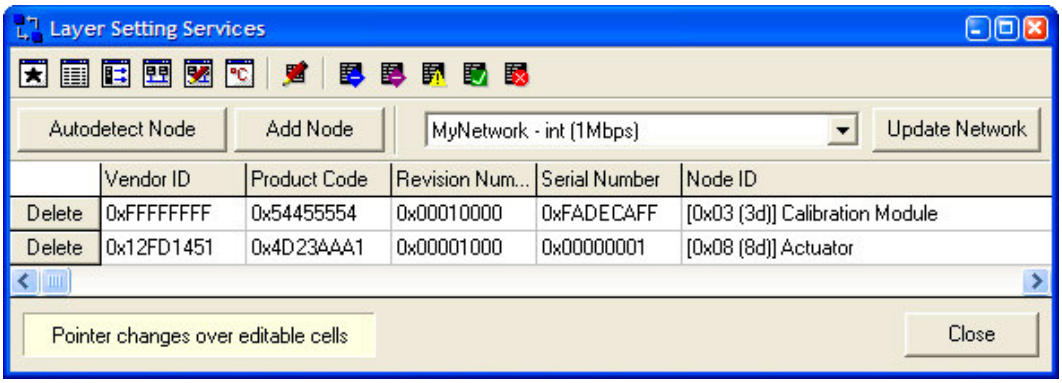
The window may be toggled between locked and unlocked by clicking on the padlock toolbar button. In the unlocked state controls may be selected, configured, resized and moved. In the locked state no changes can take place and the grid is hidden.

4.6 Layer Setting Services

Layer Setting Services (LSS) is an aspect of CANopen that can be optionally implemented on nodes to provide configurable Node IDs and baud rates. When an LSS node is started it waits for commands to configure it before booting up. PCANopen Magic Pro provides an LSS window that allows Node IDs and baud rates of LSS nodes to be configured.

The LSS window is available by clicking on the LSS toolbar button or choosing "Layer Setting Services…" from the View menu.

Standard LSS



22. Layer Setting Services Window

The window shows a list of LSS enabled nodes (called LSS slaves) known to be on the network. Nodes may be added to the list by clicking on "Add Node" and manually entering the identification information of the node. Alternatively single nodes may be automatically discovered by clicking on "Autodetect Node". This feature can only detect one node at a time and that node must be the only LSS slave on the network. If there is more than one LSS slave then disconnect then all and connect each one in turn to auto detect it. Note that LSS slaves start up using 125kbps, so ensure that a network with 125kbps baud rate is selected before attempting to detect any nodes.

To remove a node from the list click on that node's "Delete" button.

To set the Node ID of a node, click in the Node ID box for that node and choose from the drop-down list. The mouse pointer changes shape when placed over editable boxes in tables.

The network currently in use is shown at the top of the window. If no change is made then the bit timing of the LSS slaves will not be changed. To program the LSS slaves to use a different baud rate, simply selected the desired network from the drop-down list.

To assign the Node IDs to the LSS slaves and configure the bit timing, click on "Update Network". PCANopen Magic Pro will switch to the new network along with the LSS slaves and bootup messages for the slaves will appear on the bus. Because LSS slaves start up using 12kbps, ensure that a network with 125kbps baud rate is currently being used before clicking on "Update Network".

MicroLSS

MicroLSS is a specialized version of LSS that allows scanning of the network for LSS slaves quicker thsn standard LSS. In addition it provides optimizations that may be performed to further reduce the scan time.

MicroLSS does not support changing the baudrate of LSS slaves. MicroLSS can only be used with nodes that support it. Nodes supporting standard LSS will not be configurable using MicroLSS.

To access MicroLSS, click on the MicroLSS tab.

MicroLSS scans for nodes and configures them in one step. Nodes are configured using consecutive node IDs. Enter the first node ID to use and the timeout period, which is used to wait for responses from the nodes.

Click on "Scan and Configure MicroLSS Nodes" to start the process.

4.7 Trace Filtering

Trace filtering may be achieve by simple message ID based filtering or more complex script based filtering. Message ID filtering allows specific ranges or types of messages to be excluded from the trace window. Script based filtering can act like triggers to start and stop showing messages when specific conditions occur.

For a description of the message ID filtering please see the section on the trace window in Chapter 3.

To configure the trace filtering choose "Configure Filter…" from the Filter menu in the trace window. To enable script based filtering check "Use Python Script For Filtering" and select a script to use. The script will start executing when the "OK" button is clicked, and will stop executing when "Use Python Script For Filtering" is unchecked and the "OK" button is again clicked.

When the script is executing all received messages will be processed by the script to determine if each message should be stored in the trace buffer or not.

The scripting language used is Python and all standard libraries are available for use. Because all received messages pass through the script, it is possible to use the standard libraries to log the messages, transmit them to a server on the internet, etc.

An optional string may be passed to the script to configure it. For example it may be desirable to pass a node ID to the script.

The application defines two objects for use in the script, "Trace" which represents the trace functionality and "Application" which represents the application as a whole.

Application Object

The application object defines the following methods and properties:

SetLoadScriptCallback

This method defines which function should be called when the script is first loaded. It is called when the user chooses the script and clicks on the "OK" button in the filter configuration window to start using the script.
Possible uses of this function are to perform initialization, open a log file, etc.

Example:

```
import Application;

def LoadScript():
	// perform some initialiation
	return;

Application.SetLoadScriptCallback(LoadScript);
```

SetUnloadScriptCallback

This method defines which function should be called when the script is unloaded. It is called when the user disabled script based filtering and clicks on the "OK" button in the filter configuration window.
Possible uses of this function are to perform any cleanup, close a log file, etc.

Example:

```
import Application;

def UnloadScript():
	// perform cleanup
	return;

Application.SetUnloadScriptCallback(UnloadScript);
```

Parameter

This property contains the string passed to the script when it was started. The string is specified in the filter configuration window.

Example:

```
logfile.write("%s\n" % Application.Parameter);
```

Trace Object

The trace object defines the following methods and properties:

SetMessageCallback

This method defines which function should be caled when a message is received by the application. The function is called once per message.
If the function returns a "1" then the message is shown in the trace window. If the function returns a "0" then the message is not shown in the trace window.

Example:

```
import Trace;

def ProcessMessage():
	// process a message here
	// show message in trace window
	return 1;

Trace.SetMessageCallback(ProcessMessage);
```

Message Object

The message object describes a single CAN message. It can be accessed inside the message callback function. The object contains details about the CAN message, and consists of the following properties:

ID	The message ID
Millis	The message timestamp in milliseconds
Micros	The fractional part of the timestamp 0 – 99 microseconds
DLC	The nubmer of data bytes in the message
Data[]	An array of eight bytes of data

Example:

```
msg = Trace.Message();
if msg.ID == 0x123:
	return 0;
if msg.Data[4] != 0x4C:
	return 0;
return 1;
```

Chapter 5 – Configuration

5.1 Symbolic Support

Throughout PCANopen Magic Pro there is support for user-defined symbolic information. This information includes:

- Defining names for nodes
- Defining Object Dictionaries for nodes
- Defining names for messages
- Defining names for Device Types
- Defining names for node emergencies

This information is displayed when available and relevant. Some examples of where it is used:

- When selecting an Object Dictionary entry to read or write, the currently defined Object Dictionary of the selected node is displayed. Each node may have it's own unique Object Dictionary. If a node does not have an Object Dictionary defned, then a default one is used.
- In the Network Overview window the names for emergencies, device types and nodes are displayed.
- In the trace recording, names for nodes, Object Dictionary entries, emergencies and messages are displayed.

By defining symbolic information for a network, the network becomes easier to work with. No longer do nodes need to be known by a number, but instead descriptive names related to their function may be given. Important messages may be easily identified. Debugging becomes easier due to the trace recording showing customized information.

To define symbolic information the Network Configuration dialog window is used. The window is accessed by choosing Configure Network from the Options menu. The window is divided up into sections.

5.2 Network Description

Network Description Files are a method of quickly and easily describing all the symbolic information for a network. Once created PCANopen Magic Pro can be told to use a specific Network Description File. This system allows network descriptions to be stored in files and easily shared between Engineers, Emailed, etc.

Creating the Files

Network Description Files are written in a language called CANopen eXchange Language, or CXL for short. This is a language that was developed to exchange CANopen network descriptions easily between different development tools. It was designed with the aim of

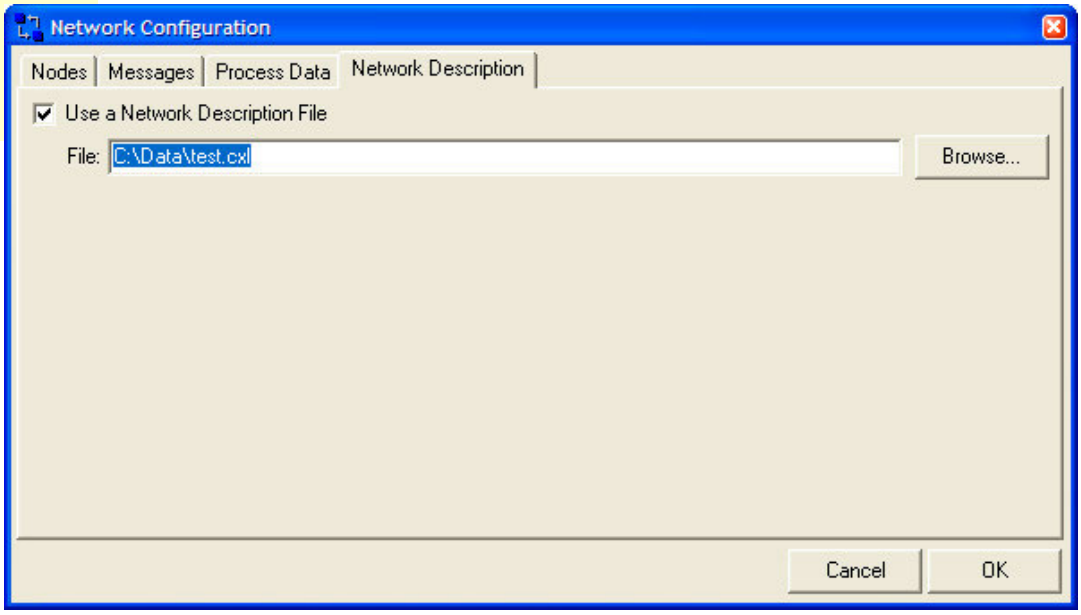
providing an easy to use, human-readable environment. CXL is based on XML and looks similar to HTML at first glance. All the files are pure ASCII and therefore editable in any application that supports such files, for example, Notepad.

Each file may define Object Dictionaries for multiple nodes, along with all the symbolic information types that PCANopen Magic Pro supports.

Appendix A contains the complete file format and an example file.

Using the Files

Click on the "Network Description" tab to access the Network Description section.



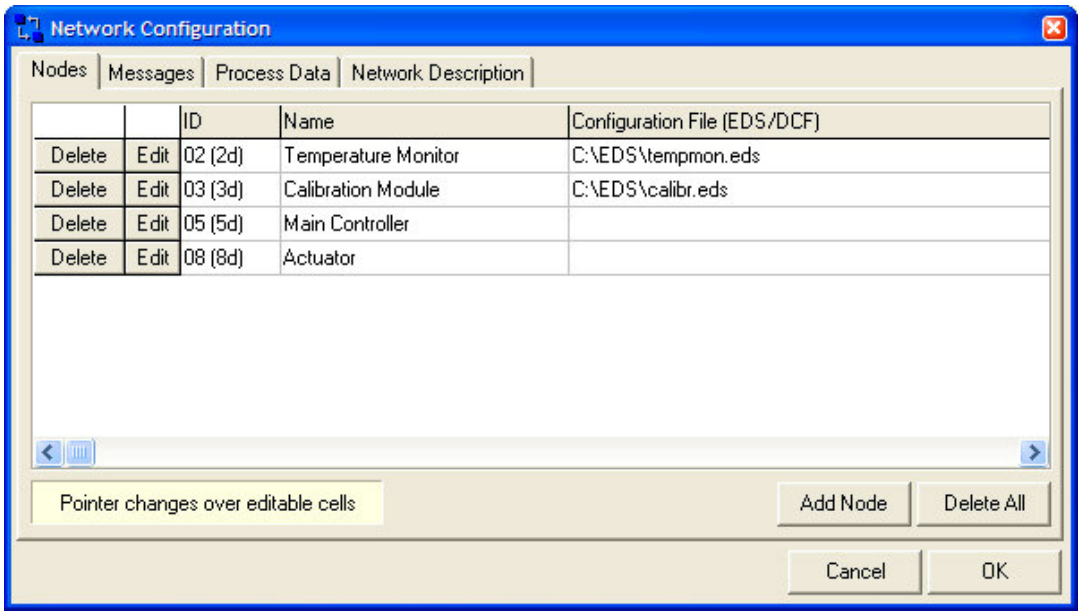
23. Network Configuration Dialog Window – Network Description Section

Check the option to use a Network Description File and click on the "Browse…" button to select one. Click on the "OK" button to use it. If there are any syntax errors in the file, PCANopen Magic Pro will generate an error message.

Network Description Files may be changed at any time simply by selecting a different one. It is not necessary to restart PCANopen Magic Pro.

5.3 Nodes

Clicking on the "Nodes" tab accesses the Nodes configuration section where the nodes on the network may be described.



24. Network Configuation Dialog Window – Nodes Section

To add a node, click on the "Add Node" button. For each node an ID, name and location of an Electronic Datasheet (EDS) or Device Configuration File (DCF) may be specified. If an EDS or DCF is specified then the Object Dictionary for the node will be known to PCANopen Magic Pro. Mutliple nodes with the same configuration but different IDs may be added at once.

To edit a node, click on the "Edit" button for that node. The dialog window is the same as the one used to add a node, and all parameters may be changed.

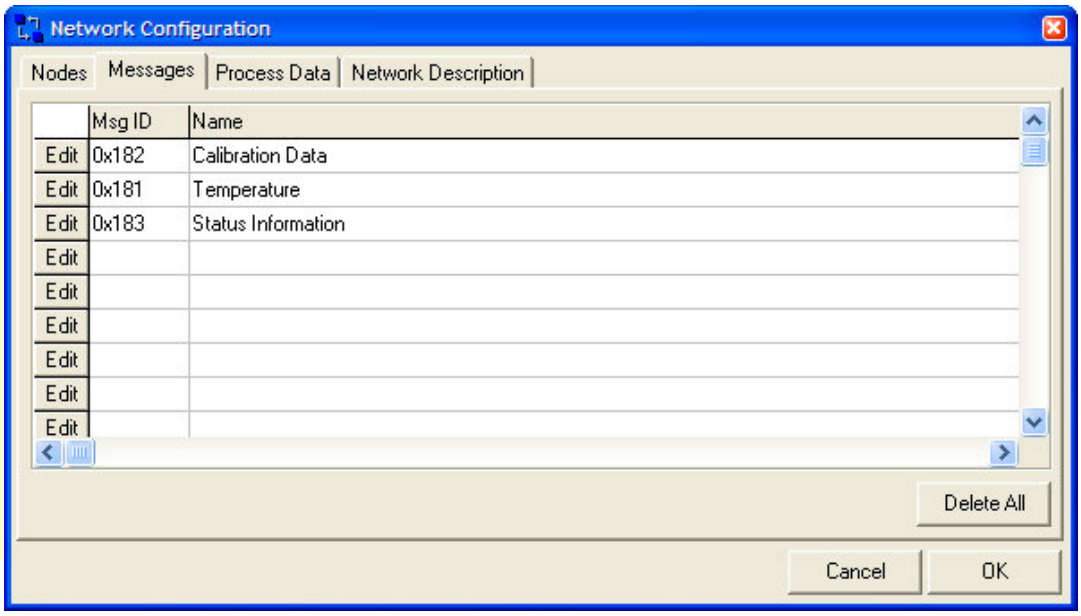
To delete a node, click on the "Delete" button for that node. To delete all nodes, click on the "Delete All" button below the table.

To change the name of a node without having to open the edit node dialog window, simply click on the current name in the table and enter a new name. The pointer changes shape when moved over boxes that can be edited.

The paths to EDS files may be relative or absolute. If there is currently no project then the relative paths are relative to the application executable. If there is a project, then the paths are relative to the project file.

5.4 Messages

Clicking on the "Messages" tab accesses the Messages configuration section where the messages on the network may be described.

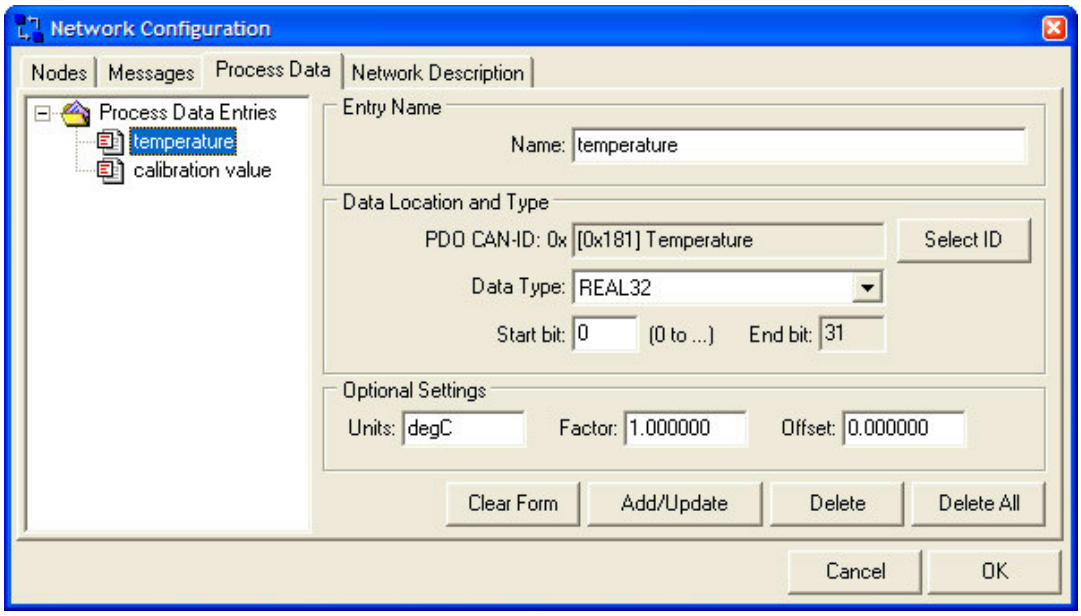


25. Network Configuration Dialog Window – Messages Section

To describe a message click on an Edit button and enter the ID and name of the message. Rows may be used in any order. Click on "Clear All" to clear the configuration of all messages.

5.5 Process Data

Clicking on the "Process Data" tab accesses the process data configuration section where the real time data on the network may be described.



26. Network Configuration Dialog Window – Process Data Section

Currently defined entries are shown on the left. To add an entry fill in the details and click on "Add/Update". To edit an entry click on the entry, change the settings and click on "Add/Update". To delete an entry click on the entry and then click on "Delete". To delete all the currently defined entries, simply click on "Delete All".

Each name must be unique to that entry, otherwise the currently existing entry with that name will be updated instead of a new entry added.

The Data Location and Type section describes the location of the data in a specific message that may appear on the network.

Units may be added and are shown alongside the value in the Process Data window. The factor and offset allow conversion of values from the raw data in a CANopen message to real world data. The raw value is multipled by the factor before the offset is added.

5.6 Default Network Description File

In the PCANopen Magic Pro installation folder is a default Network Description File called default.cxl. This file contains the default Object Dictionary that is used when a node does not have an Object Dictionary defined, along with basic CANopen definitions. The file may be customized if needed, however PCANopen Magic Pro must be restarted before any changes take effect.

If the Object Dictionary of a node is defined in a Network Description file, either by specifying the entries directly or by specifying an Electronic Datasheet File, then the default Object Dictionary is not used for that node.

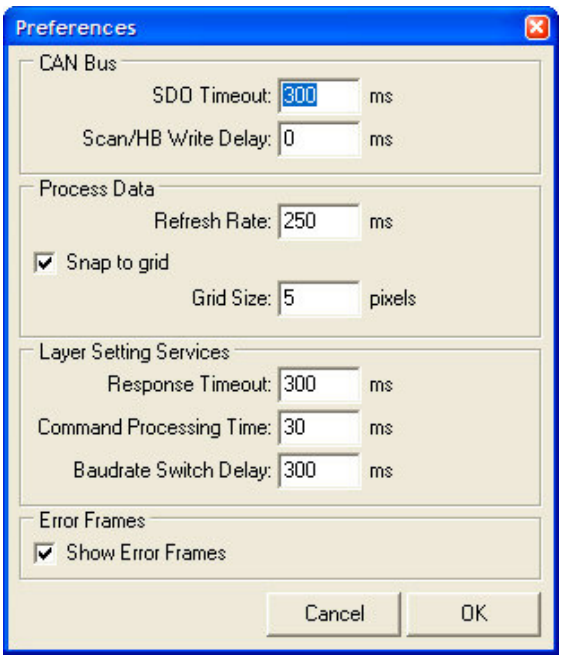
5.7 Symbol Loading Order

Because the symbolic information is loaded into PCANopen Magic Pro from multiple sources, and may overwrite any previously loaded information, there is a specific order to the loading, which is as follows:

- Default Network Description File
- User Network Description File
- Network Configuration (Nodes, Messages and Process Data)

5.8 Preferences

Various timeouts used by PCANopen Magic Pro may be configured by using the Preferences dialog window. This is accessed by choosing Preferences… from the Options menu.



27. Preferences Window

Network scans for nodes and writing of the heartbeat producer time to all nodes is performed by producing bursts of SDO expedited reads and writes. This burst can be too fast for some nodes to handle, therefore the Scan/HB Write Delay is the delay to insert between each message produced by PCANopen Magic Pro. By increasing this delay the bursts of messages can be slowed down, allowing slower nodes time to process and filter the messages.

The Layer Setting Services section provides several configurable timings. The response timeout is the time to wait for a response from an LSS node when it has been set a

confirmed command. If a response is not received within this period then the procedure is stopped.
The command processing time is the time to wait for an LSS node to process an unconfirmed command. If the commands are received too quickly the LSS node may not be able to process them all, but will not offer an indication that a failure has occurred.
The baudrate switch delay is used when switching the baud rates of all nodes on the network, and must be at least the longest command processing time of any node. Refer to the LSS specification for a full description of this parameter.

The rest of the preferences are self explanatory.

Chapter 6 – Command Line

6.1 Syntax

The command line has the following syntax:

PCOMAGICPRO [*directives*]

Where:

directives	space separated list of directives or @*commandfile*
commandfile	An ASCII file containing a space separated or newline separated list of directives

There are two types of directives:

Configuration	configure the hardware and CAN/CANopen functionality
Operation	an operation to be performed on the CANopen network

Directives may appear in any order in the list, however the operation directives are processed in the order listed. For example the following command line:

```
PCOMAGICPRO NMT(0x14, START) SDODOWNLOAD(23, 0x5FFF, 0x01, ..\foobar.bin)
```

Transmits a network management message before performing an SDO Download. However the following command line:

```
PCOMAGICPRO SDODOWNLOAD(23, 0x5FFF, 0x01, ..\foobar.bin) NMT(0x14, START)
```

Performs the SDO Download and then transmits a network management message.

Some directives may appear more than once on a command line, allowing complex operations to be performed. For example:

```
PCOMAGICPRO NMT(0x14, PREOPERATIONAL) SDODOWNLOAD(0x14, 0x5FFF, 0x01,
..\foobar.bin) NMT(0x14, START) SDOUPLOAD(0x18, 0x2000, 0x00, c:\config.txt)
```

Places node 14H in Preoperational mode, downloads a binary file to Object Dictionary entry 5FFFH subindex 01H, starts the node, then uploads ASCII data from Object Dictionary entry 2000H subindex 00H of node 18H.

Directive names are case insensitive. Whitespace is ignored except when it is between directives, where it is required and inside hardware and network names. All directives are optional. When a directive is omitted the default setting for that directive is used.

Paths may contain spaces. Paths may also contain newlines, carriage returns and tabs, however these characters are converted to spaces before the path is used, allowing word wrapping at the spaces in paths.

6.2 Directives

All numeric values passed to directives may be passed in decimal or hexadecimal unless otherwise specified. Hexadecimal values are either prefixed by "0x" or have the suffix "H" or "h". Without counting the prefix or suffix, there must be an even number of characters in a hexadecimal value.

NMT

Description: Transmits a network management message

More than one allowed: Yes

Type: Operation

Syntax: NMT (*nodeid*, *command*)

Where:

nodeid	The ID of the Node to receive the message or zero to transmit the message to all nodes.
command	The command to send to the node(s). One of:

START
STOP
PREOPERATIONAL
RESET
RESETCOMM

Output: NMT message sent to *nodeid*
Or: NMT message send failed: *reason* (*nodeid*)

nodeid	The ID of the node receiving the message in hexadecimal with no prefix or suffix
reason	The reason for the failure

Examples:

```
NMT(0x14, START)
NMT(2AH, STOP)
```

SDODOWNLOAD

Description: Writes data to a node. Either a file to write data from or hex data directly may be specified. If the Hex data is four bytes or less it must be entered into the directive in big-endian format for human readability. If the Hex data is more than four bytes then it must be entered in little-endian format.

More than one allowed: Yes

Type: Operation

Syntax: SDODOWNLOAD(*nodeid*, *index*, *subindex*, *path* | *data*)

Where:

nodeid	The ID of the node to write to
index	The Object Dictionary index to write to
subindex	The Object Dictionary entry subindex to write to
path	Path to the file to take the data to write from
data	Hex data to write

Output: SDO Download complete
Or: SDO Download failed: *reason*

Where:

reason	The reason for the failure

Examples:

```
SDODOWNLOAD(0x14, 0x2000, 0x1A, C:\shared\test.hex)
SDODOWNLOAD(23, 0x5FFF, 0x01, ..\foobar.bin)
SDODOWNLOAD(23, 0x5FFF, 0x01, 0x11223344)
SDODOWNLOAD(0x14, 0x2000, 0x1A, 0x33)
SDODOWNLOAS(0x07, 0x20AA, 0x00, 0xA24D2A5A2B5C3D8A)
```

SDOUPLOAD

Description: Reads data from a node

More than one allowed: Yes

Type: Operation

Syntax: SDOUPLOAD(*nodeid*, *index*, *subindex*, *path*)

Where:

nodeid	The ID of the node to read from
index	The Object Dictionary index to read from
subindex	The Object Dictionary entry subindex to read from
path	Path to the file to store the read data

Output: SDO Upload complete
Or: SDO Upload failed: *reason*

Where:

reason	The reason for the failure

Examples:

```
SDOUPLOAD(0x14, 0x2000, 0x1A, C:\shared\test.hex)
SDOUPLOAD(23, 0x5FFF, 0x01, ..\foobar.bin);
```

HARDWARE

Description: Selects and configures the CAN hardware interface

More than one allowed: No

Type: Configuration

Syntax: HARDWARE(*hardware*, *network*)

Where:

hardware	The name of the hardware interface to use. This must match exactly, for example, "PEAK USB-CAN".
network	The name of the network to use. This must match exactly the name of an existing network for the hardware interface selected. For example "MyNetwork".

Output: Hardware initialized
Or: Hardware initialization failed: *reason*

Where:

reason The reason hardware initialization failed

Default: HARDWARE(PEAK USB-CAN, MyNetwork)

Examples:

```
HARDWARE(PEAK USB-CAN, Test Network 5)
```

QUIET

Description: Outputs information about the processing of each directive to an ASCII output file rather than the standard output, where a program can process it.

More than one allowed: No

Type: Configuration

Syntax: QUIET(*path*)

Where:

path Path of the output file to generate.

Output: None.
Or: Output file creation failed (*path*)

Where:

path The path of the output file to generate as passed to the QUIET directive.

Default: Output is sent to the standard output.

Examples:

```
QUIET(test.txt)
QUIET(..\test.txt)
QUIET(C:\work\test.txt)
```

ODWRITE

Description: Writes data to a node. The same as the SDODOWNLOAD directive. Either a file to write data from or hex data directly may be specified. If the Hex data is four bytes or less it must be entered into the directive in big-endian format for human readability. If the Hex data is more than four bytes then it must be entered in little-endian format.

More than one allowed: Yes

Type: Operation

Syntax: ODWRITE(*nodeid*, *index*, *subindex*, *path* | *data*)

Where:

nodeid	The ID of the node to write to
index	The Object Dictionary index to write to
subindex	The Object Dictionary entry subindex to write to
path	Path to the file to take the data to write from
data	Hex data to write

Output: OD Write complete
Or: OD Write failed: *reason*

Where:

reason	The reason for the failure

Examples:

```
ODWRITE (0x14, 0x2000, 0x1A, C:\shared\test.hex)
ODWRITE (23, 0x5FFF, 0x01, ..\foobar.bin)
ODWRITE (23, 0x5FFF, 0x01, 0x11223344)
ODWRITE (0x14, 0x2000, 0x1A, 0x33)
ODWRITE (0x07, 0x20AA, 0x00, 0xA24D2A5A2B5C3D8A)
```

ODREAD

Description: Reads data from a node. The same as the SDOUPLOAD directive.

More than one allowed: Yes

Type: Operation

Syntax: ODREAD(*nodeid*, *index*, *subindex*, *path*)

Where:

nodeid	The ID of the node to read from
index	The Object Dictionary index to read from
subindex	The Object Dictionary entry subindex to read from
path	Path to the file to store the read data

Output: OD Read complete
Or: OD Read failed: *reason*

Where:

reason The reason for the failure

Examples:

```
ODREAD (0x14, 0x2000, 0x1A, C:\shared\test.hex)
ODREAD (23, 0x5FFF, 0x01, ..\foobar.bin);
```

SDOTIMEOUT

Description: Sets the timeout of the SDO protocol in milliseconds

More than one allowed: No

Type: Configuration

Syntax: SDOTIMEOUT(*timeout*)

Where:

timeout The timeout to use in milliseconds

Output: None

Examples:

```
SDOTIMEOUT(300)
```

Appendix A – CXL Reference

This appendix describes the CXL language used for the Network Description Files.

A.1 File Format

Header and Footer

CXL files are in plain ASCII format. Each file begins with a header:

```
<?xml version="1.0" encoding="UTF-8"?>
<cxl version="1.00" author="Joe Bloggs" date="08-Jul-2004">
```

and ends with a footer:

```
</cxl>
```

Between the header and footer are the top level tags. The top level tags specify sub level tags, which may also in turn specify lower level tags. Refer to the following sections for a full description of each top-level tag.

XML Tag

The XML tag must appear exactly as shown. Even though the encoding is UTF-8, only latin characters are currently supported.

CXL Tag - Version

The version field is mandatory and specifies the CXL file format version. Currently only version 1.00 is defined.

CXL Tag - Author

The author field is optional. The value does not have a defined format.

CXL Tag - Date

The date field is optional. The value has the format:

dd-mmm-yyyy

where *dd* is the day, *mmm* is the first three letters of the month and *yyyy* is the year.

Number Bases

All hexadecimal values in CXL are prefixed with "0x".

Case

Tag names are case sensitive and must appear as they are described in this appendix. The data between the tags is not case-sensitive.

Formatting

Whitespace (spaces, tabs, newlines, carriage returns) are ignored between end and start tags. For example the following are the same:

```
<message>
  <id>0x81</id>
  <name>Node 1 Emergency</name>
</message>
```

```
<message><id>0x81</id><name>Node 1 Emergency</name></message>
```

But whitespace in the data itself is not ignored, so the following are not the same:

```
<message>
  <id>0x81</id>
  <name>Node 1 Emergency</name>
</message>
```

```
<message>
  <id>0x81</id>
  <name>Node 1
  Emergency</name>
</message>
```

A.2 objectdictionary

Defines an Object Dictionary. There may be multiple objectdictionary tags in a symbols file. The subtags are:

- nodeid
  Describes the ID of the node that implements the Object Dictionary. Optional.
- entry
  Describes an Object Dictionary entry

nodeid

If the Object Dictionary relates to a specific node then the nodeid subtag may be included. It specifies the ID of the node in hexadecimal. If it is omitted, then the Object Dictionary entries relate to all nodes.

Example:

```
<nodeid>0x45</nodeid>
```

objectdictionary examples:

```
<objectdictionary>
  <nodeid>0x40</nodeid>
  <entry>
    …
  </entry>
  <entry>
  …
  </entry>
</objectdictionary>

<objectdictionary>
  <entry>
  …
  </entry>
  <entry>
  …
  </entry>
</objectdictionary>
```

entry

Describes a single entry in the Object Dictionary and has the following subtags:

- index
  Describes the index of the entry
- name
  Describes the name of the entry
- subentry
  Describes the subentries in the entry

index

Contains the value in hexadecimal of the index of the entry.

Example:

```
<index>0x1000</index>
```

name
Contains the name associated with the entire Object Dictionary entry.

Example:

```
<name>Device Type</name>
```

subentry
Contains the following subtags that define the subentry:

- subindex
  Describes the subindex of the subentry.
- name
  Describes the name of the subentry. Optional.
- type
  Describes the type of the subentry.

subindex
Contains the value in hexadecimal of the subindex of the subentry.

Example:

```
<subindex>0x00</subindex>
```

name
Contains the name of the subentry. This name is appended to the name of the entry when displayed in PCANopen Magic Professional. If the name subelement is omitted, then no name is appended to the name of the entry. Omitting the name is therefore only useful if the entry has one subentry.

Example:

```
<name>Number of Entries</name>
```

type
Contains the CANopen data type of the subentry. It must be one of the following values (case is ignored):

- BOOLEAN
- INTEGER8
- INTEGER16
- INTEGER24
- INTEGER32
- INTEGER40
- INTEGER48
- INTEGER56
- INTEGER64
- UNSIGNED8

- UNSIGNED16
- UNSIGNED24
- UNSIGNED32
- UNSIGNED40
- UNSIGNED48
- UNSIGNED56
- UNSIGNED64
- REAL32
- REAL64
- VISIBLE_STRING
- OCTET_STRING
- UNICODE_STRING
- TIME_OF_DAY
- TIME_DIFFERENCE
- DOMAIN

Example:

```
<type>UNSIGNED32</type>
```

Examples

The following are examples of an entry:

```
<entry>
  <index>0x1000</entry>
  <name>Device Type</name>
  <subentry>
    <subindex>0x00</subindex>
    <type>UNSIGNED32</type>
  </subentry>
</entry>

<entry>
  <index>0x2000</index>
  <name>Cursor Position</name>
  <subentry>
    <subindex>0x00</subindex>
    <name>Number of Entries</name>
    <type>UNSIGNED8</type>
  </subentry>
  <subentry>
    <subindex>0x01</subindex>
    <name>X</name>
    <type>REAL32</type>
  </subentry>
  <subentry>
    <subindex>0x02</subindex>
    <name>Y</name>
    <type>REAL32</type>
```

```
  </subentry>
</entry>
```

A.3 network

Defines network wide information. There may be multiple network elements in a symbols file. The subtags are:

- message
  Describes a message identifier
- abortcode
  Describes an SDO abort code
- errorcode
  Describes an error code
- node
  Describes a node on the network
- devicetype
  Describes a device type
- vendor
  Describes vendors of CANopen nodes

message

Describes a single message and has the following subtags:

- id
  Describes the COB-ID of the message
- name
  Describes the name of the message

id

Describes the COB-ID of the message in hexadecimal.

Example:

```
<id>0x80</id>
```

name

Describes the name of the message. I.e. the name to associate with the ID.

Example:

```
<name>SYNC</name>
```

Examples

The following are examples of message identifier definitions:

```
<message>
  <id>0x80</id>
  <name>SYNC</name>
</message>
<message>
  <id>0x81</id>
  <name>Node 1 Emergency</name>
</message>
```

abortcode

Describes a single abort code and has the following subtags:

- code
  Describes the abort code.
- name
  Describes the name of the code.

code

Describes the code in hexadecimal.

Example:

```
<code>0x05030000</code>
```

name

Describes the name of the abort code. I.e. the name to associate with the abort code.

Example:

```
<name>Toggle bit not alternated</name>
```

Examples

The following are examples of abort code definitions:

```
<abortcode>
  <code>0x05030000</code>
  <name>Toggle bit not alternated </name>
</abortcode>
```

errorcode

Describes a single error code and has the following subtags:

- code
  Describes the error code.
- name
  Describes the name of the code.

code
Describes the code in hexadecimal.

Example:

```
<code>0x2000</code>
```

name
Describes the name of the error code. I.e. the name to associate with the error code.

Example:

```
<name>Current</name>
```

Examples
The following are examples of error code definitions:

```
<errorcode>
  <code>0x2000</code>
  <name>Current</name>
</errorcode>
```

node
Describes a single node and has the following subtags:

- id
  The ID of the node.
- name
  Describes the name of the node.
- eds
  Describes the Electronic Datasheet for the node.
- simsetupfile
  Describes the setup file for the simulated node. Optional.
- simdll
  Describes the name of the simulation DLL. Optional.
- simio
  Describes the simulation IO file to use when node is simulated. Optional.
- simproduct
  Describes the predefined product to simulate. Optional.

id
Describes the ID of the node in hexadecimal.

Example:

```
<id>0x7F</id>
```

name
Describes the name of the node. I.e. the name to associate with the node.

Example:

```
<name>NMT Master</name>
```

eds
Describes the electronic datasheet of a node by specifying the path to the file. The path may be absolute or relative to the symbol file.

Example:

```
<eds>..\eds\mynode.eds</eds>
```

simsetupfile
Either contains the name of a standard setup file located in the Simulation\Setup Files subfolder (minus the .txt extension), or contains the absolute path to a specific setup file to use. Optional and when used it is assumed that the node is simulated, however it is only used with PCANopen Magic ProDS.

Example:

```
<simsetupfile>Generic IO</simsetupfile>
```

simdll
Describes the name of the DLL to be used to simulate the node. Optional and when used it is assumed that the node is simulated, however it is only used with PCANopen Magic ProDS. The name must the name of a simulation DLL in the Simulation\DLLs subfolder. May also be an absolute a path to a specific simulation DLL.

Example:

```
<simdll>MicroCANopen.dll</simdll>
```

simio
Describes the name of the IO file to be used to provide controls when the node is simulated. Optional. Only used with PCANopen Magic ProDS. The name must be the name of a file in the Simulation\IO Files subfolder minus the extension (.sim) or the absolute path to a specific simulation file.

Example:

```
<simio>PCAN_MM_Dig1</simio>
```

simproduct
Describes the product that the node should function like when simulated. Products are defined in the Simulation\Products subfolder, and are a shorthand method of specifying the simsetupfile, simdll and simio parameters. Optional and usually used instead of simsetupfile, simdll and simio. When used it is assumed the node is simulated, however it is only used with PCANopen Magic ProDS.

Example:

```
<simproduct>PEAK MicroMod</simproduct>
```

Examples
The following are examples of node definitions:

```
<node>
  <id>0x01</id>
  <name>Motor Controller</name>
</node>
<node>
  <id>0x7F</id>
  <name>NMT Master</name>
  <eds>..\eds\nmtmaster.eds</eds>
</node>
```

devicetype
Describes a single device type and has the following subtags:

- profile
  The device profile number.
- name
  The name of the device profile.
- Bit
  Describes a single bit in the upper 16 bits of the device type. Optional.

profile
Describes the device profile number.

Example:

```
<profile>0x0191</profile>
```

name
Describes the name of the device profile. I.e. the name to associate with the device profile.

Example:

```
<name>I/O Module</name>
```

bit
Describes a bit in the upper 16 bits of the device type, and has the following subtags:

- position
  The bit position in the range 16 – 31.
- set
  Text to display when the bit is set.
- unset
  Text to display when the but is not set.

position
Defines the position of the bit being described. Must be in the range 16 – 31.

Example:

```
<position>16</position>
```

set
Defines the text to display when the bit is set. May be blank.

Example:

```
<set>Digital inputs</set>
```

unset
Defines the text to display when the bit is unset. May be blank.

Example:

```
<unset>No digital inputs</unset>
```

Examples
The following are examples of device type definitions:

```
<devicetype>
  <profile>0x0191</profile>
  <name>I/O Module</name>
  <bit>
    <position>16</position>
    <set>Digital inputs</set>
    <unset></unset>
  </bit>
</devicetype>
```

vendor
Describes a single CANopen node vendor, and has the following subelements:

- id
  The Manufacturer ID (24-bits).
- name
  The vendor's name
- department
  Describes a department for the vendor (8-bits). Optional.

id

Defines the manufacturer ID for the vendor. This is the lower 24 bits of the Vendor ID.

Example:

```
<id>0x455341</id>
```

name

Defines a name for the vendor.

Example:

```
<name>ESAcademy</name>
```

department

Defines a single department for the vendor. There may be up to 256 department entries for a single vendor. The following subelements are defined:

- id
  The ID of the department.
- name
  The name of the department.

id

Defines the ID for the department. This is the upper 8 bits of the Vendor ID.

Example:

```
<id>0x01</id>
```

name

Defines a name for the department.

Example:

```
<name>USA</name>
```

Examples

The following is an example of a vendor declaration:

```
<vendor>
  <id>0x455341</id>
  <name>ESAcademy</name>
  <department><id>0x00</id><name>Extenal</name></department>
  <department><id>0x01</id><name>USA</name></department>
  <department><id>0x02</id><name>Germany</name></department>
  <department><id>0x03</id><name>ESS Germany</name></department>
</vendor>
```

A.4 processdata

Defines process data that can appear on the bus. This section is used to create the Process Data window. There may be multiple processdata elements in a symbols file. The subtags are:

- data
  Defines a single item of process data

data

Defines a single item of process data and has the following subtags:

- name
  The name of the process data
- id
  The COB-ID of the PDO that contains the process data
- startbit
  The starting bit of the process data
- endbit
  The ending bit of the process data
- type
  The type of the process data
- units
  The units of the data. Optional.
- factor
  The scaling factor to apply to the value of the data. Optional.
- offset
  The offset to apply to the value of the data. Optional.

name

The name of the process data. This will be the name displayed in PCANopen Magic Pro.

Example:

```
<name>Temperature</name>
```

id
The message ID of the PDO that contains the process data in hexadecimal.

Example:

```
<id>0x180</id>
```

startbit
The starting bit where the process data can be found inside the PDO's data (0-indexed).

Example:

```
<startbit>8</startbit>
```

endbit
The ending bit where the process data can be found inside the PDO's data (0-indexed).

Example:

```
<endbit>15</endbit>
```

type
The CANopen type of the data. It must be one of the following values (case is ignored):

BOOLEAN
INTEGER8
INTEGER16
INTEGER24
INTEGER32
INTEGER40
INTEGER48
INTEGER56
INTEGER64
UNSIGNED8
UNSIGNED16
UNSIGNED24
UNSIGNED32
UNSIGNED40
UNSIGNED48
UNSIGNED56
UNSIGNED64
REAL32
REAL64
VISIBLE_STRING
OCTET_STRING
UNICODE_STRING
TIME_OF_DAY
TIME_DIFFERENCE

DOMAIN

units

The units for the data. For example, degrees, celcius, flashes, rpm, etc.

Example:

```
<units>RPM</units>
```

offset

An offset applied to the value of the data after the scaling factor has been applied. Allows the data to be given specific ranges. May be a real number. Optional.

value = (value x factor) + offset

The offset is not used for the following types:

BOOLEAN
VISIBLE_STRING
OCTET_STRING
UNICODE_STRING
TIME_OF_DAY
TIME_DIFFERENCE
DOMAIN

Example:

```
<offset>6.252</offset>
```

factor

A scaling factor applied to the value of the data before the offset has been applied. May be a real number. Optional.

value = (value x factor) + offset

The scaling factor is not used for the following types:

BOOLEAN
VISIBLE_STRING
OCTET_STRING
UNICODE_STRING
TIME_OF_DAY
TIME_DIFFERENCE
DOMAIN

Example:

```
<factor>-45.32</factor>
```

Examples
The following are examples of Process Data declarations:

```
<data>
  <name>Temperature</name>
  <id>0x180</id>
  <startbit>0</startbit>
  <endbit>31</endbit>
  <type>REAL32</type>
  <units>degrees C</units>
</data>
<data>
  <name>Data Size</name>
  <id>0x204</id>
  <startbit>8</startbit>
  <endbit>15</endbit>
  <type>UNSIGNED8</type>
  <units>bytes</units>
</data>
```

A.5 comments

Allows comments to be inserted into the file. All contents inside the comments tags are ignored and may be arbitary.

Example:

```
<comments>
Revision 1.00
History: Added new node definition, AA, 06-Jul-2005
</comments>
```

A.6 Example File

The following is an example Network Description File:

```
<?xml version="1.0" encoding="UTF-8"?>
<cxl version="1.00" author="AAyre" date="26-Jan-2004">

  <objectdictionary>
    <entry>
      <index>0x1000</index>
      <name>Device Type</name>
      <subentry>
        <subindex>0x00</subindex>
        <type>UNSIGNED32</type>
```

```
    </subentry>
  </entry>
  <entry>
    <index>0x2000</index>
    <name>Cursor Position</name>
    <subentry>
      <subindex>0x00</subindex>
      <name>Number of Entries</name>
      <type>UNSIGNED8</type>
    </subentry>
    <subentry>
      <subindex>0x01</subindex>
      <name>X</name>
      <type>REAL32</type>
    </subentry>
    <subentry>
      <subindex>0x02</subindex>
      <name>Y</name>
      <type>REAL32</type>
    </subentry>
  </entry>
</objectdictionary>

<objectdictionary>
  <nodeid>0x21</nodeid>
  <entry>
    <index>0x2500</index>
    <name>Temperature</name>
    <subentry>
      <subindex>0x00</subindex>
      <type>REAL32</type>
    </subentry>
  </entry>
</objectdictionary>

<network>
  <message>
    <id>0x80</id>
    <name>SYNC</name>
  </message>
  <message>
    <id>0x181</id>
    <name>Environment</name>
  </message>
  <abortcode>
    <code>0x05030000</code>
    <name>Toggle bit not alternated</name>
  </abortcode>
  <errorcode>
```

```
    <code>0x2000</code>
    <name>Current</name>
  </errorcode>
  <node>
    <id>0x21</id>
    <name>Environment Monitor</name>
  </node>
  <node>
    <id>0x7F</id>
    <name>NMT Master</name>
  </node>
  <devicetype>
    <profile>0x0191</profile>
    <name>I/O</name>
    <bit>
      <position>16</position>
      <set>Digital inputs</set>
      <unset></unset>
    </bit>
  </devicetype>
  <vendor>
    <id>0x455341</id>
    <name>ESAcademy</name>
    <department><id>0x00</id><name>Extenal</name></department>
    <department><id>0x01</id><name>USA</name></department>
    <department><id>0x02</id><name>Germany</name></department>
    <department><id>0x03</id><name>ESS Germany</name></department>
  </vendor>
 </network>

 <processdata>
  <data>
    <name>Temperature</name>
    <id>0x180</id>
    <startbit>0</startbit>
    <endbit>31</endbit>
    <type>REAL32</type>
    <units>degrees C</units>
  </data>
 </processdata>

</cxl>
```

This example symbols file defines the following:

- There are two nodes on the network, Environment Monitor (ID 0x21) and NMT Master (ID 0x7F).
- The PDO with COB-ID 0x181 contains environmental information, and it's first four bytes contain the temperature in degrees C.

- All nodes define the Device Type at OD entry 0x1000 and a Cursor Position at 0x2000.
- The Environment Monitor node also defines Temperature at OD entry 0x2500.
- The SYNC message has the COB-ID 0x80.
- A Device Type of 0x00000191 is an I/O node. When bit 16 is set, the device supports digital inputs.
- The vendor ESAcademy is defined with four departments within the vendor.
- Values are defined for error and abort codes.