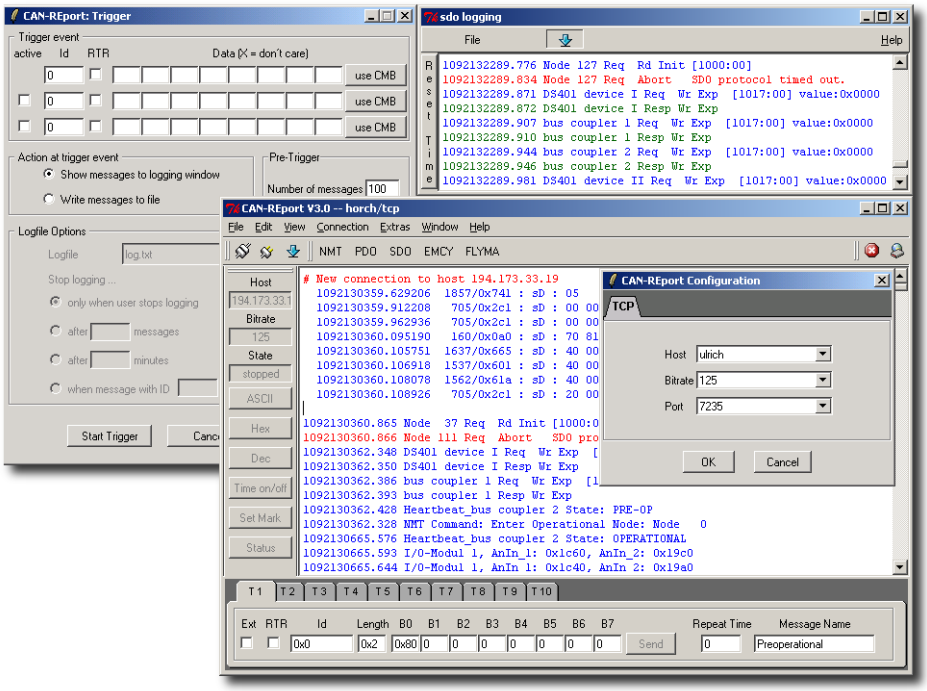
CAN-REport
Benutzerhandbuch



Ablehnungshinweis
Alle Rechte vorbehalten

Die von *port* GmbH gelieferten Programme, Baugruppen und Dokumentationen werden mit großer Sorgfalt erstellt und in unterschiedlichen Einsatzfällen getestet und geprüft.

port GmbH kann trotzdem keine Gewähr oder Haftung dafür übernehmen, dass die Software, die Baugruppe und die Dokumentation fehlerfrei bzw. für spezielle Zwecke geeignet ist.

Insbesondere Beschreibungen und technische Daten sind keine zugesicherten Eigenschaften im rechtlichen Sinne.

Für Folgeschäden, die aufgrund der Benutzung der Programme und Baugruppen auftreten, wird deshalb jede juristische Verantwortung oder Haftung ausgeschlossen.

port hat das Recht, Änderungen an den beschriebenen Produkten oder an der Dokumentation ohne vorherige Ankündigung vorzunehmen, wenn sie aus Gründen der Zuverlässigkeit oder Qualitätssicherung vorgenommen werden oder dem technischen Fortschritt dienen.

Für Hinweise auf eventuelle Fehler sind wir dankbar und bitten um eine Benachrichtigung.

Wir werden uns bemühen, derartige Hinweise schnellstmöglich zu überprüfen.

Copyright

Regensburger Straße 7
D-06132 Halle
Tel. +49 345 - 777 55 0
Fax. +49 345 - 777 55 20
E-Mail service@port.de
Internet http://www.port.de

Inhaltsverzeichnis

1. Übersicht

Der CAN-Analyzer *CAN-REport* von *port* ist ein leistungsfähiges und vielseitig einsetzbares Werkzeug für Entwicklung, Test und Wartung, zur Analyse und Inbetriebnahme von CAN-basierenden Netzwerken besonders DeviceNet und CANopen.

Die eingebaute Skriptfähigkeit erlaubt neben dem einfachen Aufzeichnen von Nachrichten eine universelle Verwendung bei Entwicklung, Test und Wartung, vor allem im Bereich der industriellen Automatisierungstechnik.

Die Trennung von Hardware-Interface (CAN-Anbindung) und Visualisierungs-Software erlaubt außerdem die Anwendung über TCP-IP Netzwerke.

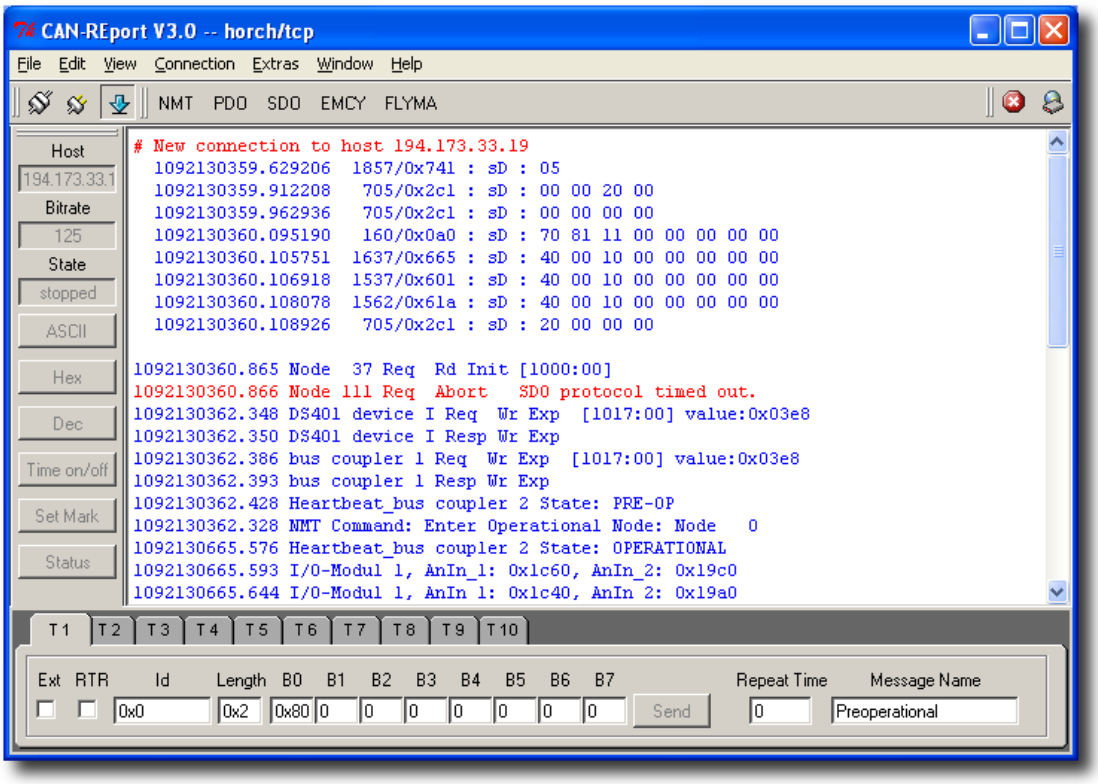


Bild 1, Analyzer Gesamtansicht

2. Installation

Die Installation umfasst:

- die Bedienoberfläche
- den horch Server und
- einen Layer 2 Treiber für das CAN-Interface.

Für die Installation sind die folgenden Schritte durchzuführen:

1. In Abhängigkeit des verwendeten CAN-Interfaces können vorbereitende Installationsschritte für den Layer 2 Treiber notwendig sein. Diese sind in der Datei **INSTALL** im horch-Verzeichnis auf der Installations-CD beschrieben.

    Bitte lesen Sie diese Datei vor der Installation.

2. setup.exe ausführen.

   • Full Installation: Es erfolgt eine menügeführte automatische Installation aller Softwarekomponenten einschließlich des Kopierens aller Handbücher.

   • Customized Installation: Es erfolgt durch Anklicken eine Auswahl der zu installierenden Softwarekomponenten. Für die Installation der Device Monitor Software sind die folgenden Komponenten zwingend erforderlich: *CAN-REport*, horch und Layer 2 Treiber.

3. Falls Sie ein Symbol für den horch-Server auf Ihrem Desktop angelegt haben, sind die Optionen im Programmaufruf des horch Server Ihrer Applikation anzupassen und zu erweitern. Eine Übersicht über alle verfügbaren Optionen, die der horch Server unterstützt, liefert die Hilfe:

   ```
   horch -h
   ```

3. Anwendung

In der Standardausstattung verfügt der CAN-Analyzer bereits über leistungsfähige Grundfunktionen, die viele Einsatzmöglichkeiten abdecken.

Dazu gehören neben der Online Beobachtung des Busverkehrs das Senden von einmaligen oder zyklischen Nachrichten, dem Senden von kompletten Nachrichtenfolgen sowie der Aufzeichnung von CAN-Nachrichten und die Speicherung des Mitschnitts in Log-Dateien. Neben dem Standard Text-Format wird auch die Speicherung im CSV-Format unterstützt, wodurch die Datenübernahme in Tabellenkalkulationen vereinfacht wird.

Mit Hilfe einer projektspezifischen Datenbasis kann für jedes CAN Telegramm eine anwenderspezifische Interpretation der Dateninhalte festgelegt werden.

Durch ergänzende Software-Module können erweiterte Funktionen zur Verfügung gestellt werden. Dies sind unter anderem die dienstabhängige Darstellung von Nachrichten in separaten Protokollfenstern für verschiedene Protokollfamilien. Bild 2 zeigt dies am Beispiel der SDO-Kommunikation für CANopen-basierende Systeme.

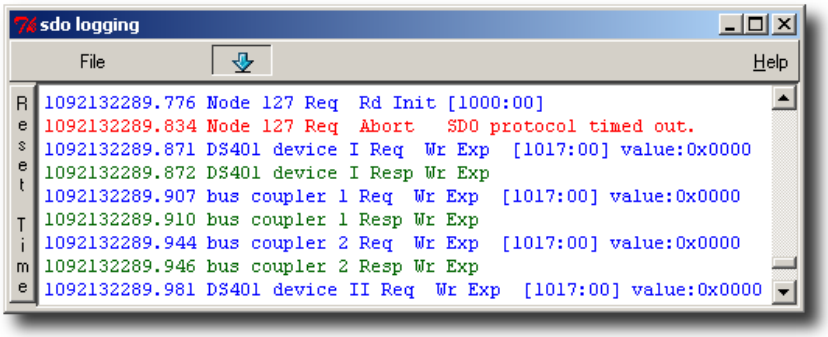


Bild 2, CANopen SDO Interpretation und Protokollierung

Man erkennt die Darstellung von Dienst und Knotennamen, z.B. "Req_Verriegelung", sowie die Anzeige von "index:subindex" in den **SDO Initiate** Anforderungen.

Mit Hilfe der integrierten Skriptsprache ist die Programmierung neuer Funktionen oder die individuelle Erweiterung des graphischen Interfaces möglich.

4. CAN-Schnittstellen

4.1. TCP/IP

Der CAN-Analyzer *CAN-REport* besteht aus einem Hardware-Interface mit CAN-Anbindung und der Visualisierungs-Software. Die Verbindung zwischen den beiden Komponenten erfolgt auf der Basis des Client/Server Modells über standardmäßige TCP-IP Schnittstellen. Diese Trennung gestattet, die gesamte Applikation über TCP-IP Netzwerke hinweg zu benutzen, so daß beide Komponenten auf unterschiedlichen Systemen eingesetzt werden können. Damit wird auch der Einsatz zur Fernwartung ohne zusätzliche oder modifizierte Software über Wählverbindungen oder das Internet möglich.

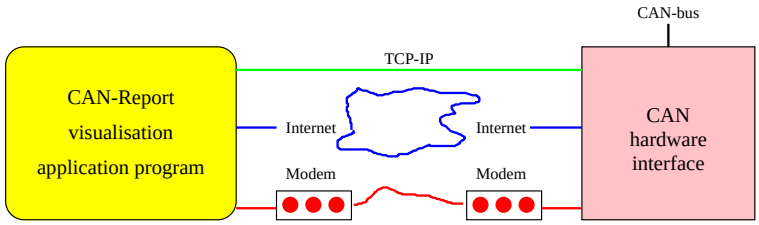


Bild 3, Kommunikationsprinzip

Als Hardwareplattformen stehen neben PC Einsteckkarten für ISA- und PCI-Bus, PC-104 und Parallelport auch externe Interfaces über V24-Schnittstellen oder Ethernetanschluß wie zum Beispiel das *port*-EtherCAN Modul zur Verfügung. Alle Hardwareinterfaces können unter den Betriebssystemen MS-Windows und LINUX eingesetzt werden.

4.1.1. CAN-Server

Der CAN-Server *horch* ist die Schnittstelle von TCP/IP zu CAN. Eine genaue Beschreibung finden sie in dem Handbuch zu *horch*.

4.1.2. Client - *CAN-REport*

Die Visualisierungs- und Bediensoftware ist in der für viele Plattformen erhältlichen Skriptsprache *Tcl/TK* programmiert. Ein Einsatz ist somit auf nahezu beliebigen Systemen möglich, für die eine *Tcl/TK* Implementierung verfügbar ist. Damit können neben MS-Windows basierenden Systemen auch UNIX oder Macintosh Rechner zur Bedienung des Hardwareinterfaces genutzt werden.

4.2. Serielle Schnittstellen

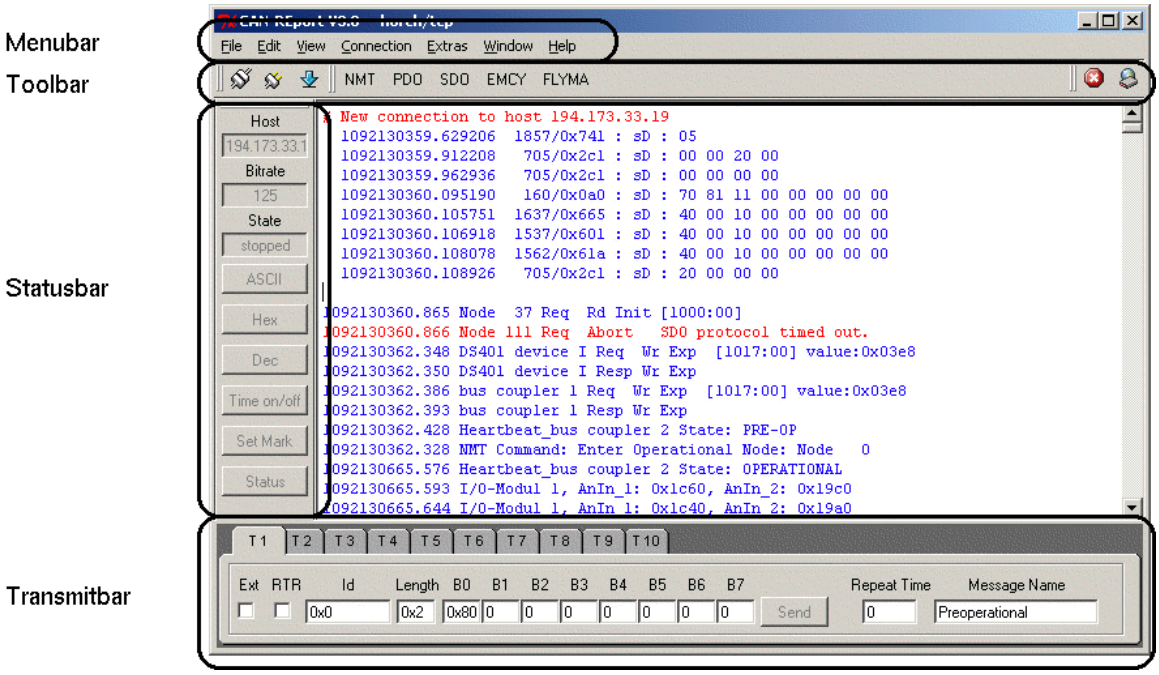
Neben den Client-Server-Verbindungen unterstüzt der *CAN-REport* ebenfalls Direktverbindungen mit CAN-RS232-Interface-Baugruppen, wie:

- CAN232
- CANview

Zur Verwendung dieser seriellen CAN-RS232-Schnittstellen ist das jeweilige Gerät und die benutzte serielle Schnittstelle in den *CAN-Interface* im Verbindungsmenü auszuwählen.

Für weitere Hinweise zur Konfiguration siehe Absatz Konfiguration serieller Schnittstellen

5. Grafische Bedienoberfläche



5.1. Menüleiste

5.1.1. Das Datei-Menü

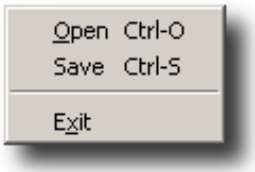


Bild 4, Das File Menu

Open

Eine früher gespeicherte Log-Datei öffnen. Wenn die Option *CANopen Extension* aktiviert ist, wird der Inhalt nach dem CANopen-Protokoll interpretiert.

Save

Speichert den Inhalt des Hauptfensters in eine Datei.

Exit

Beendet *CAN-REport*.

5.1.2. Das Edit-Menü

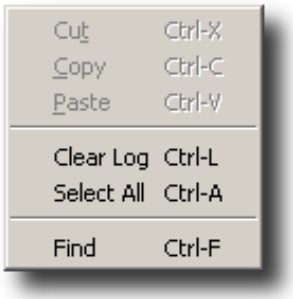


Bild 5, Das Edit-Menü

Cut
: Den markierten Text ausschneiden.

Copy
: Kopiert den markierten Text.

Paste
: Fügt Text aus der Zwischenablage in das Hauptfenster ein.

Clear Log
: Löscht den Inhalt des Hauptfensters.

Select All
: Markiert den gesamten Inhalt des Hauptfensters.

Find
: Öffnet einen Suchdialog zur Suche nach Textstellen.

5.1.3. Das View-Menü

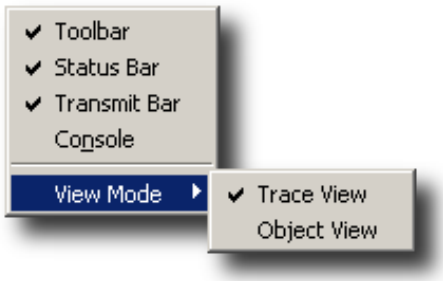


Bild 6, Das View Menü

Toolbar
: Schaltet die Anzeige der Werkzeugleiste ein bzw. aus

Statusbar
: Schaltet die Anzeige der Statusleiste ein bzw. aus

Transmitbar
: Schaltet die Anzeige der Statusleiste ein bzw. aus

Console
: Öffnet die Tcl/Tk-Konsole am unteren Fensterrand des Hauptfensters. Mit der Konsole kann der Funktionsumfang des *CAN-REport* durch eigene Tcl/Tk-Skripte erweitert und an die eigenen Bedürfnisse angepaßt werden. Siehe Absatz *CAN-REport* erweitern.

5.1.3.1. Ansichtsmodus

Der *CAN-REport* verfügt über einen Ablaufverfolgungs- (Trace) und einen Objektansichtsmodus.

Trace view
: Alle CAN-Telegramme werden in das Hauptfenster geschrieben, wie sie empfangen wurden. Die Ansicht zeigt: Zeitstempel, CAN-ID, Typ, Daten.

Object view
: Jedes CAN-Telegramm wird in einer Zeile angezeigt. Sobald ein CAN-Telegramm mit dem selben CAN-Identifier empfangen wird, wird die Zeile aktualisiert. In dieser Ansicht können 192 CAN-Telegramme angezeigt werden Diese Ansicht zeigt: die CAN-Id, Typ, Daten, Periode des Telegramms und Anzahl der empfangenen CAN-Telegramm pro CAN-ID.

5.1.4. Das Verbindungsmenü

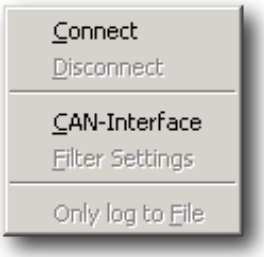


Bild 7, Das Verbindungsmenü

Connect
: Verbindung mit dem CAN-Server *horch* aufbauen. Die Angaben, wie Servername, IP-Port und CAN-Bitrate, werden unter dem Menüpunkt *CAN-Interface* eingeben.

Disconnect
: Verbindung mit dem Server trennen.

CAN-Interface

Öffnet einen Dialog zur Konfiguration der TCP/IP Verbindung zum CAN-Server. Es werden nur die CAN-Schnittstellen angezeigt, für die ein entsprechender Treiber installiert wurde.

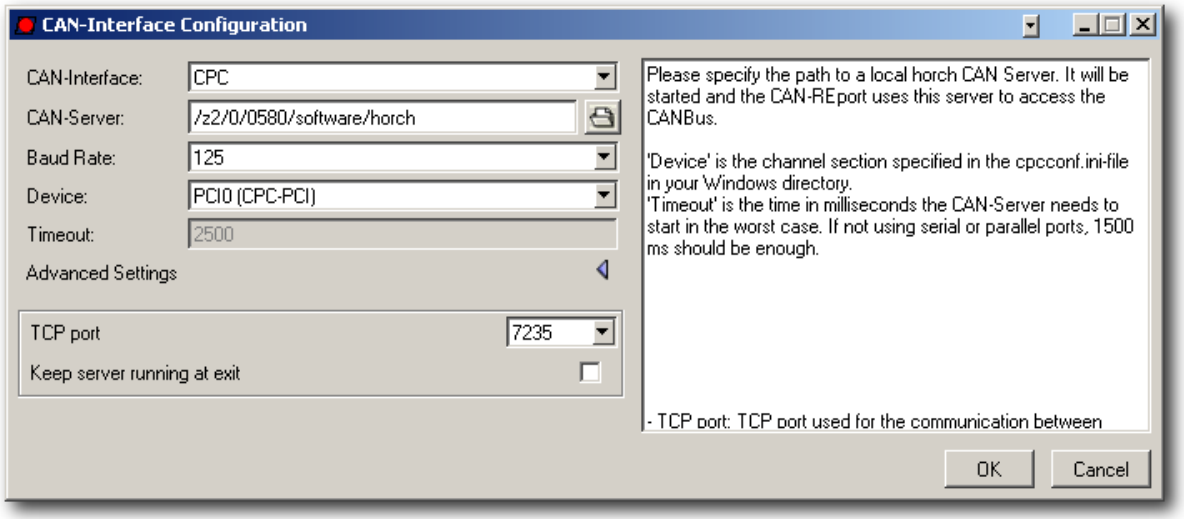


Bild 8, Konfigurationsdialog für die CAN-Schnittstelle

Filter Settings

Öffnet einen Dialog mit einer graphischen Eingabemaske für die Akzeptanz-Masken Register des CAN-Controllers. Obwohl dieser Menüpunkt angezeigt wird, können die Register bei einigen CAN-Schnittstellenkarten nicht verändert werden. Siehe Absatz Filter.

Only Log to File

CAN-Telegramme werden in einer Log-Datei gespeichert und nicht im Hauptfenster angezeigt.

5.1.5. Das Extras-Menü

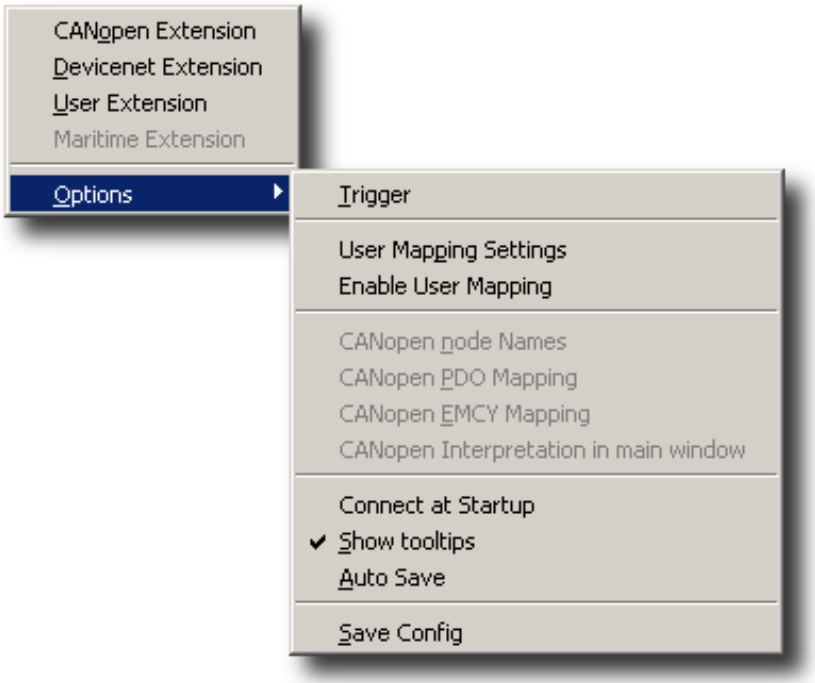


Bild 9, Das Extras-Menü

CANopen Extension

Lädt die CANopen-Erweiterung. Diese Option erweitert die Symbolleiste um je einen Knopf für die CANopen-Dienste SDO, PDO, NMT, EMCY, FLYMA. Auf Knopfdruck öffnet sich ein Fenster des jeweiligen Dienstes. In diesem Fenster werden nur CANopen-Nachrichten für diesen Dienst angezeigt.

DeviceNet Extension

Diese Option erweitert die Symbolleiste um einen Knopf für ein DeviceNet-Protokoll Fenster. In diesem Fenster wird die Interpretierung des empfangenen CAN-Telegramms angezeigt.

User Extension

Lädt die Datei <application.tcl> und fügt der Symbolleiste einen Schaltknopf hinzu. Die Datei enthält eine Tcl-Funktion für die anwenderspezifische Interpretation von CAN-Telegrammen. Ein Beispiel liegt bei.

Maritime Extension

Lädt die Maritime-Erweiterung. Sie wird benutzt, um zwei CAN-Linien gleichzeitig zu beobachten. Dazu müssen zwei Verbindungen zu 2 CAN-Servern aufgebaut werden. Wenn die Erweiterung aktiviert wird, wird die aktuelle Verbindung geschlossen. Der Verbindungsdialog (CAN-Interface) verfügt nun über weitere Eingabefelder, um die Verbindung zum zweiten CAN-Server angeben zu können.

Im Hauptfenster werden die CAN-Telegramme nebeneinander dargestellt. Die Sendeleiste (Transmitbar) wird geändert. Die Auswahlkästchen für Extended und RTR-Telegramme werden benutzt, um die Linie auszuwählen.

5.1.5.1. Optionen

Trigger
: Öffnet ein Fenster zur Trigger-Konfiguration (siehe Absatz Trigger).

User Mapping Settings
: Öffnet ein Konfigurationsfenster, in dem die Interpretation von CAN-Telegrammen festgelegt werden kann. Der *CAN-REport* interpretiert dann eintreffende Nachrichten. Siehe Interpretation von CAN-Telegrammen.

Enable User Mapping
: Aktiviert und Deaktiviert die Anwender-Interpretation von CAN-Telegrammen.

CANopen Node Names
: Öffnet einen Dialog, in dem Knotennummern Namen zugewiesen werden können. Wenn die *CANopen Extension* aktiviert ist, werden anstatt der Knotennummer die Namen angezeigt.

CANopen PDO Mapping
: Öffnet einen Dialog, in dem festgelegt werden kann, wie der Inhalt von PDO-Nachrichten dargestellt werden soll. Einem PDO werden eine COB-ID, ein symbolischer Name und paarweise Datentyp und symbolischer Name der Daten zugewiesen. Siehe Interpretation von CAN-Telegrammen.

CANopen EMCY Mapping
: Öffnet einen Dialog, in dem festgelegt werden kann, wie der Inhalt von EMCY-Nachrichten dargestellt werden soll. Einem EMCY werden eine COB-ID, ein symbolischer Name und paarweise Datentyp und symbolischer Name der Daten zugewiesen. Siehe Interpretation von CAN-Telegrammen.

CANopen interpretation in main window
: Die Interpretation der CAN-Telegramme nach dem CANopen-Protokoll wird auch im Hauptfenster vorgenommen.

Connect at Startup
: Diese Option steuert das Startverhalten. Ist sie aktiviert, versucht der *CAN-REport* beim nächsten Neustart eine Verbindung zum CAN-Server aufzubauen.

Tooltips
: Aktiviert Tooltips für alle Eingabeelemente.

Auto Save
: Wenn dieser Menüpunkt aktiviert ist, werden die Konfigurationsdaten automatisch gespeichert bevor der *CAN-REport* beendet wird.

5.1.6. Das Window-Menü

Bild 10, Das Window Menü

Clear All
: Löscht den Inhalt des Hauptfensters und den Inhalt von Fenstern, die von Erweiterungen geöffnet werden.

Cascade
: Plaziert die Erweiterungsfenster in einer Kaskade neben dem Hauptfenster.

Tile Vertical
: Plaziert die Erweiterungsfenster vertikal neben dem Hauptfenster.

Tile Horizontal
: Plaziert die Erweiterungsfenster horizontal unter dem Hauptfenster.

Hide All
: Schließt die Erweiterungsfenster.

Show All
: Öffnet die Erweiterungsfenster.

5.1.7. Das Help-Menü

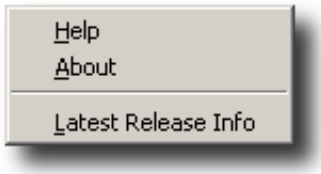


Bild 11, Das Hilfe Menü

Help
: Bietet eine kurze Hilfe zu *CAN-REport*.

About
: Urheber- und Lizenzinformation. Zusätzlich werden auch die vorhandenen Tcl/Tk Pakete angezeigt.

Latest Release Info

Prüft, ob eine neue Version des *CAN-REport* vorhanden ist. Dafür wird eine TCP/IP-Verbindung zu *http://www.port.de* aufgebaut.

5.2. Symbolleiste

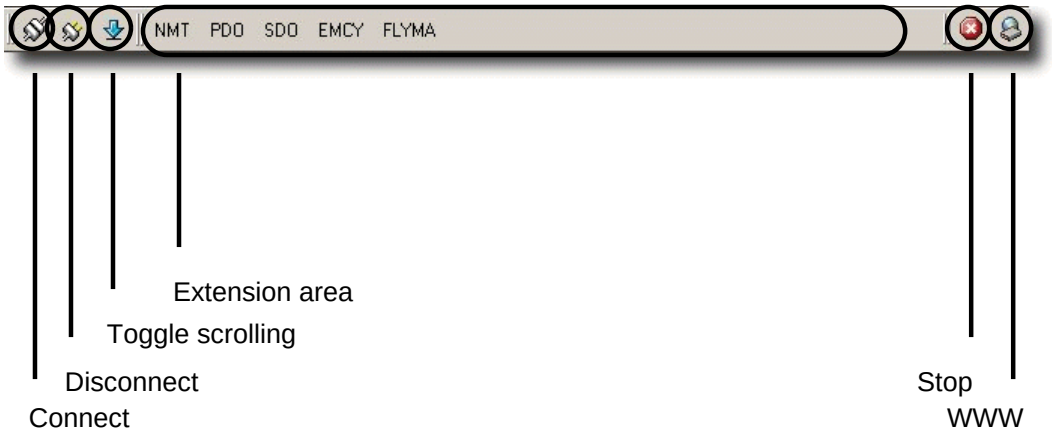


Figure 12, Symbolleiste

Die Symbolleiste bietet schnellen Zugriff auf die Menüpunkte "Connect" und "Disconnect". Der dritte Schaltknopf mit dem nach unten gerichteten Pfeil schaltet das Mitrollen des Laufbalkens im Protokollfenster an bzw. aus. Der "Stop"-Schaltknopf ermöglicht es Anwender-Skripte abzubrechen. Weiteres siehe Absatz *CAN-REport* erweitern. Mit dem "www"-Schaltknopf prüft *CAN-REport*, ob eine neue Version vorhanden ist. Dazu wird eine Internetverbindung aufgebaut.

5.3. Statusleiste

Bild 13, Statusleiste

Die Statusleiste zeigt Informationen über die Verbindung zum *horch*-Server an. Zusätzlich enthält sie Schaltknöpfe, um die Anzeige der CAN-Telegramme zu verändern. Mit den Knöpfen "ASCII", "Hex" und "Dec" werden die CAN-Daten in dem jeweiligen

Format dargestellt. Der Schaltknopf "Toggle Time" schaltet die Zeitanzeige im *CAN-REport* an bzw. aus. Die Zeit wird in dem Format "Sekunden.Mikrosekunden" angegeben. Die Angabe von Mikrosekunden ist treiberabhängig. Der Linux-Treiber *can4linux* ermöglicht die Anzeige von Millisekunden statt Mikrosekunden. "Set Mark" schreibt eine Trennzeile in das Protokollfenster.
Der Statusknopf öffnet ein Fenster, daß Informationen über die verwendete Hardware beziehungsweise den verwendeten Treiber gibt. Bild 14 zeigt die Informationen von dem CAN-Server, der auf Linux läuft. Die Windows-Treiber unterstützen nicht alle hier dargestellten Informationen, so daß einige Werte mit 0 angezeigt werden.

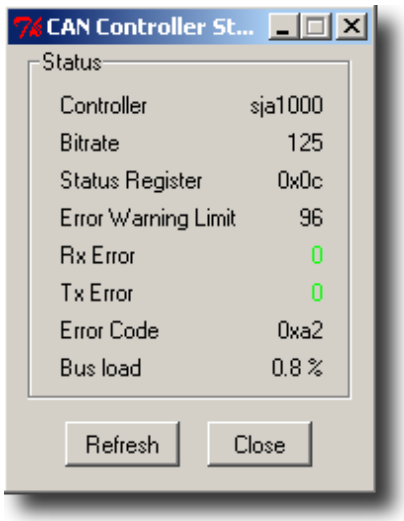


Bild 14, Statusfenster

5.4. Sende-Symbolleiste

CAN-REport ermöglicht es auch CAN-Telegramme zu senden. Über die Sende-Symbolleiste am unteren Rand werden CAN-Id und Daten eingegeben und mit dem Schaltknopf "Send" gesendet. Die Anzahl der angezeigten Register kann in der Konfigurationsdatei <canreport.rc> folgendermassen angegeben werden:

```
set ST(txch) 12 ; #put any number here
```

Diese Angaben werden in der Konfigurationsdatei gespeichert und bei erneutem Start des *CAN-REport* wieder geladen. Wenn "AutoSave" aktiviert wurde, speichert *CAN-REport* die Daten automatisch beim Verlassen des Programms. Die Daten eines Telegramms können in den drei Zahlensystemen dezimal, hexadezimal und oktal angegeben werden. Dezimalen Zahlen wird ein "0x" und oktalen Zahlen eine "0" vorangestellt. Mit Hilfe der "Repeat Time" kann eine Wiederholzeit in ms angegeben werden. Nach der angegebenen Zeit sendet *CAN-REport* dieses CAN-Telegramm wieder. Optional kann einem CAN-Telegramm ein symbolischer Name zugeordnet werden. Das gesendete CAN-Telegramm

wird nicht im Protokollfenster angezeigt. Es erscheint lediglich eine Nachricht, daß ein Telegramm gesendet wurde. Bei periodischen Telegrammen erscheint keine Ausschrift.

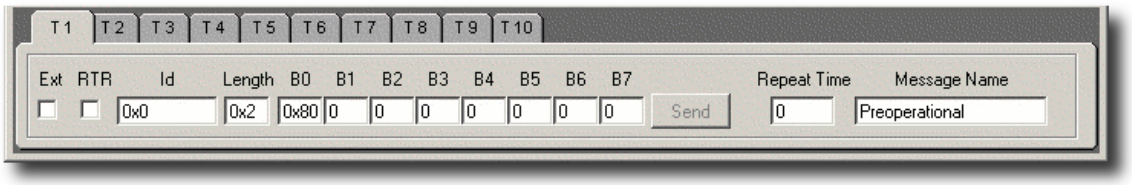


Bild 15, Sende-Symbolleiste

6. Interpretation von CAN-Telegrammen

Dieser Dialog dient der Konfiguration von Anwender-, PDO und EMCY-Interpretation. Die PDO und EMCY-Interpretation ist nur verfügbar, wenn der Menüpunkt *CANopen Extension* aktiviert ist. Das PDO/EMCY-Mapping wird verwendet, um CAN-Telegramme mit einem Identifier aus den PDO- (0x181 - 0x57F) und EMCY-Bereich (0x81 - 0xFF) anwenderspezifisch zu interpretieren. Die Anwenderinterpretation kann für alle anderen CAN-Telegramme genutzt werden.

Die Daten können als vorzeichenbehaftete und vorzeichenlose Zahlen dargestellt werden. Für jedes Datum kann ein eigener Text und das Datenformat "Big Endian" oder "Little Endian" angegeben werden. Die Konfiguration wird mittels zwei Dialogen durchgeführt. Der erste Dialog zeigt eine Übersicht aller konfigurierten CAN-Telegramme. Im zweiten Dialog wird die Interpretation des CAN-Telegramms festgelegt.

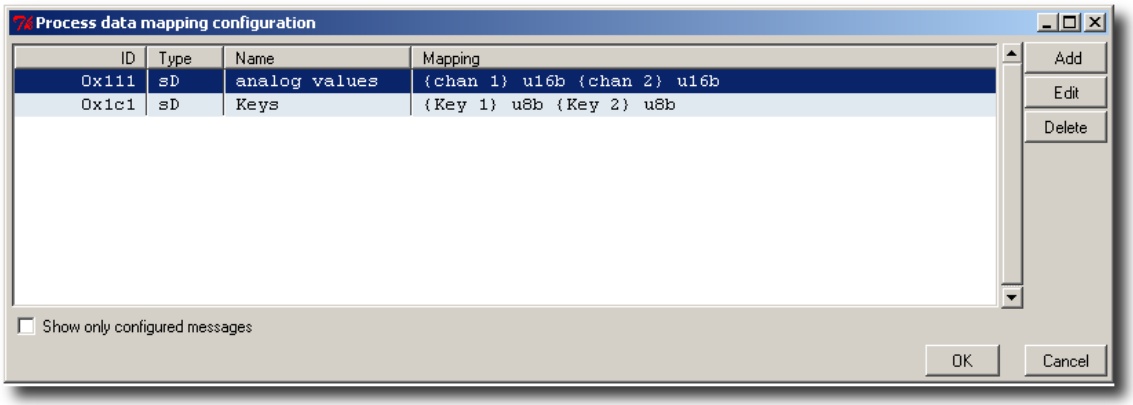


Bild 16, PDO/EMCY-Mapping Dialog

Der Dialog hat Schaltflächen, um CAN-Telegramme hinzuzufügen (add), zu bearbeiten (edit) und zu löschen (delete). Mit den Schaltflächen Hinzufügen und editieren erscheint der Konfigurationsdialog.

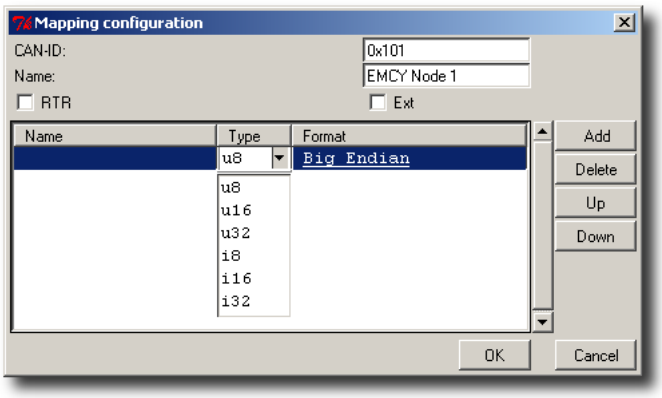


Bild 17, CAN-Telegramm Konfigurationsdialog

Hinweis: Bei EMCY-Telegrammen sind die ersten 3 Bytes des CAN-Telegramms als u16 und u8 vordefiniert. Diese können nicht verändert werden und werden nicht angezeigt.

7. CANopen-Erweiterung

7.1. Knotennamen vergeben

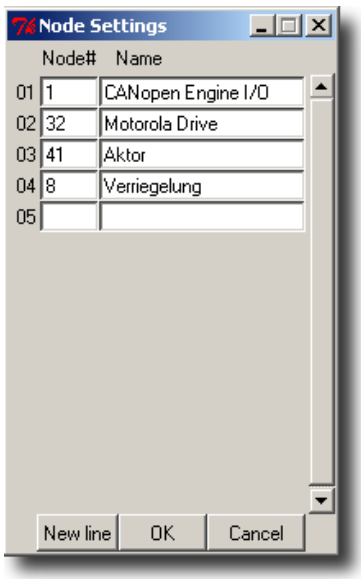


Bild 18, CANopen Konfiguration von Knotennamen

Dieser Dialog ist nur verfügbar, wenn der Menüpunkt *CANopen Extension* aktiviert ist. Mit diesem Dialog können CANopen Geräte symbolische Namen zugewiesen werden. Diese werden dann in den Protokoll-Fenstern für die CANopen-Dienste anstatt der Knotennummer des Gerätes verwendet. Bild 19 zeigt ein Beispiel.

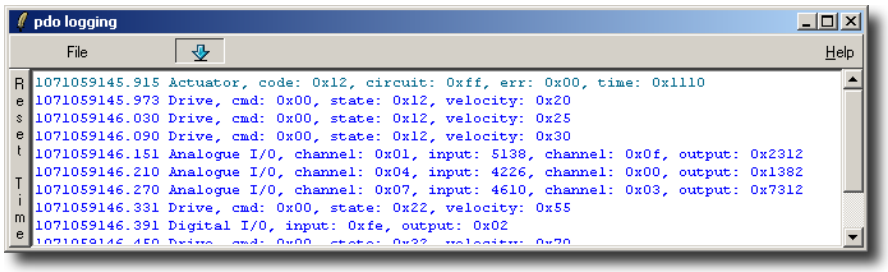


Bild 19, CANopen PDO Interpretation und Protokollierung

7.2. "Dienstspezifische Interpretation"

Die CANopen Erweiterung bietet die dienstbezogene Interpretation der CAN-Telegramme in separaten Fenstern für die Dienste: NMT, SDO, PDO, EMCY and Flying Master (FLYMA).
Die Interpretation für PDO und EMCY kann wie das User-Mapping an die Anwendung angepaßt werden.

8. Trigger

Mit der Trigger-Funktionalität, kann *CAN-REport* auf bestimmte CAN-Telegramme warten und nach Eintreffen des gewünschten Telegramms die Protokollierung weiter fortsetzen. Der Eingabedialog besteht aus den drei Teilen

Trigger Event	-	Hier wird die CAN-Id und der Inhalt des CAN-Telegramms angegeben, auf das gewartet wird. Der "CANopen Message Builder" (CMB) hilft ein CAN-Telegramm zu erstellen, das dem CANopen-Protokoll entspricht.
Action at trigger event	-	Auswahl der Aktion nachdem das angegebene CAN-Telegramm auftrat. Die CAN-Telegramme können im Hauptfenster angezeigt werden oder in eine Datei gespeichert werden.
Logfile options	-	Wenn in eine Log-Datei gepeichert wird, kann hier eine Abbruchbedingung für das Mitschneiden angegeben werden.

Tabelle 1, Einteilung des Triggerdialogs

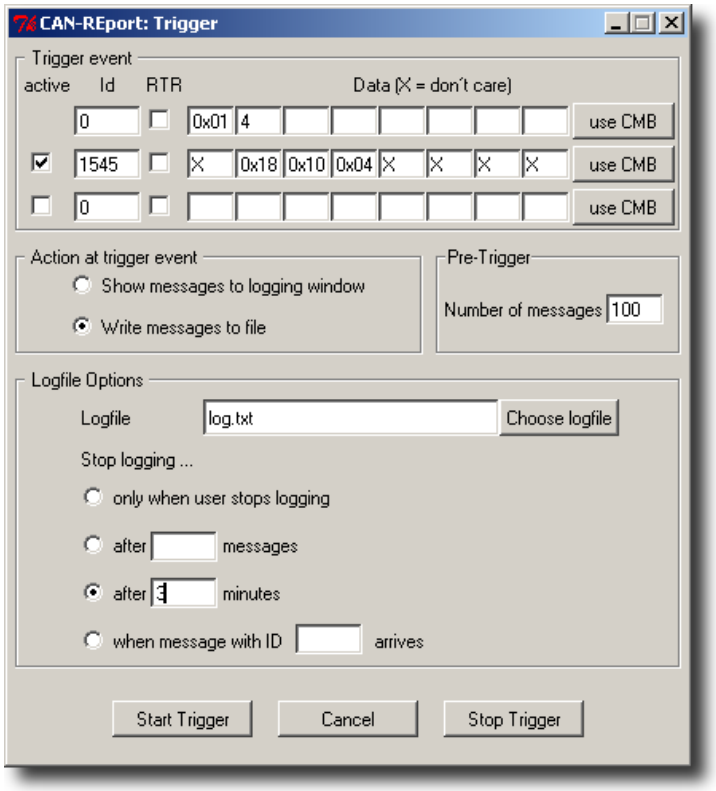


Bild 20, Triggerdialog

Der Vorgang der Triggerung wird mit dem Schaltknopf "Start Trigger" gestartet. Während des Vorgangs bleibt der Dialog stehen und die Aufzeichnung im Hauptfenster wird angehalten.

8.1. Festlegen eines Trigger CAN-Telegramms

Es können bis zu drei CAN-Telegramme angegeben werden. Das zweite und dritte CAN-Telegramm wird durch den "active"-Schalter aktiviert. Das erste Telegramm ist bei Verwendung des Triggers standardmäßig aktiviert. Die Eingabe in die Felder CAN-ID und Data kann in den Zahlensystemen dezimal, hexadezimal und octal geschehen. Hexadezimalen Zahlen wird ein "0x" und oktalen Zahlen wir eine "0" vorangestellt. Ein leeres Eingabefeld stellt das Ende eines CAN-Telegramms dar. Die Länge des Telegramms wird dann berechnet. Ein Byte kann als "don’t care"-Byte angegeben werden, indem statt einem Zahlenwert ein "X" in ein Datenfeld eingegeben wird. Diese Bytes werden bei dem Vergleich der Telegramme nicht berücksichtigt.

Beispiele:

Bild 21, Trigger Beispiel 1

CAN-Telegramm mit 8 Bytes Datenlänge. Byte 4 bis 7 sind als "don’t care" markiert. Die Datenbytes sind in unterschiedlichen Zahlensystemen angegeben.

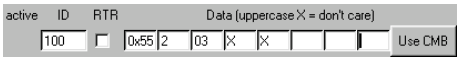


Bild 22, Trigger Beispiel 2

CAN-Telegramm mit 5 Bytes Datenlänge. Byte 4 und 5 sind als "don’t care" markiert. Die Datenbytes sind in unterschiedlichen Zahlensystemen angegeben.

Bild 23, Trigger Beispiel 3

Es wird nur auf die Telegrammnummer getriggert. Der Inhalt und die Länge des Telegramms werden nicht ausgewertet.

8.2. Festlegen eines Trigger Remote-Request CAN-Telegramms

Ein Remote Request Telegramm wird in gleicher weise eingegeben wie ein Datentelegramm. Die Länge eines RTR-Telegramms wird durch die Angabe von "dont’t care" Einträgen angegeben. Beispiel:

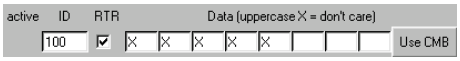


Bild 24, Trigger Beispiel 4

Der *CAN-REport* triggert auf ein RTR-Telegramm mit der Telegrammnummer 0x64 und der Länge 5.

8.3. CANopen Message Builder

Der CANopen Message Builder erleichtert die Erstellung von Inhalten für CAN-Telegrammen für den Trigger. Er bietet Eingabemasken für die CANopen-Dienste NMT, PDO, SDO, HEARTBEAT und EMCY. Bild 25 zeigt die Eingabemaske für den NMT-Dienst. Der CANopen Message Builder ist nur mit der CANopen Erweiterung verfügbar.

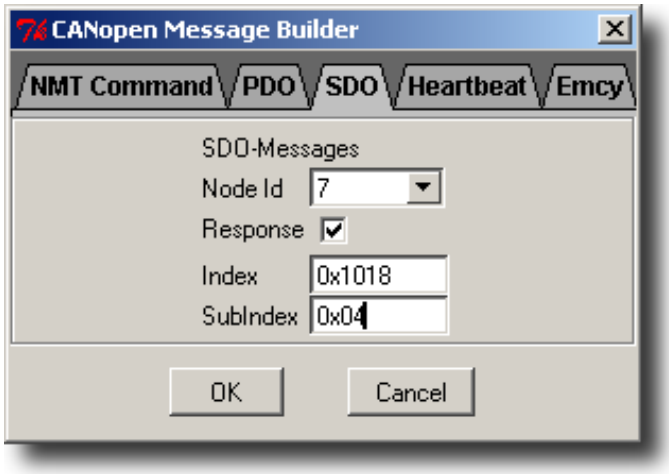


Bild 25, Eingabedialog für den NMT-Dienst des NMT-Dienstes

9. Filter

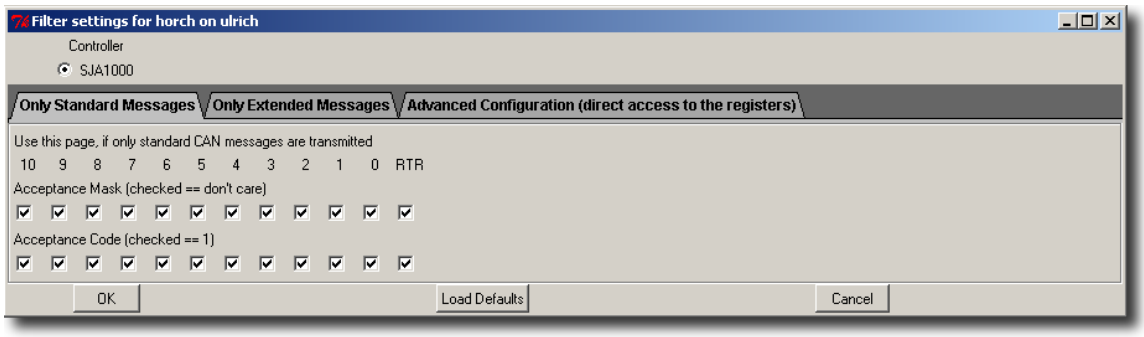


Bild 26, Dialog zur Filtereinstellung

Der Filterdialog bietet den Zugriff auf die Register der Akzeptanzmaske im CAN-Controller SJA1000. Die Verwendung von einer Akzeptanzmaske im CAN-Controller verringert die Interruptlast. Es sind drei Eingabemöglichkeiten vorhanden:

1. Eingabe der Akzeptanzmaske für CAN 2.0 A beginnend mit dem MSB auf der linken Seite.
2. Eingabe der Akzeptanzmaske für CAN 2.0 B beginnend mit dem MSB auf der linken Seite.
3. Eingabe der Akzeptanzmaske wie im Register des CAN-Controllers

Diese Fähigkeit ist abhängig von der verwendeten CAN-Schnittstelle in ihrem Rechner. Zur Zeit unterstützen die horch Server für *can4linux* und *CPC (SJA1000)* das Setzen der Akzeptanz-Register. Die anderen Hardware-Interfaces (siehe Einschränkungen) erlauben

diese Einstellung nicht.

Achtung: Obwohl die beiden niederwertigsten Bits der CAN-Telegrammnummer verändert werden können, werden diese beiden Bits bei Filterung vom SJA1000 nicht beachtet.

10. Programmieren mit *CAN-REport*

10.1. *CAN-REport* erweitern

Mit Hilfe der integrierten Konsole stehen nicht nur Geräteentwicklern komfortable Kommandos zum interaktiven Zugriff auf den CAN Bus zur Verfügung, die in eigenen Testskripten genutzt werden können. Dazu gehören Kommandos wie **`wr`** zum Senden von CAN Nachrichten oder **`wait`** zum Synchronisieren auf CAN Nachrichten am Bus.

Overview of user function

wait {id timeout}	- *wait for CAN message with id or until timeout*
wait2 {id type timeout}	- *wait for CAN message with id and type or until timeout*
wr {args}	- *send CAN message with standard identifier*
wrx {args}	- *send CAN message with extended identifier*

Detailed function description

wait {id timeout}

wait for CAN message with id or until timeout

this function is deprecated please use wait2

Arguments

id - can be an integer, or hex (0x...) or "any"
timeout - in ms, defaults to 10000 == 10s

Results

String with message
String: "wait timeout"

wait2 {id type timeout}

wait for CAN message with id and type or until timeout

Arguments

id - can be an integer, or hex (0x...) or "any"
type - any combination of s,x,D,R
s - standard; x extended
D - data; R - RTR
timeout - in ms, defaults to 10000 == 10s

Results

String with message
String: "wait timeout"

Example

```
    wait2 12 R   ; wait 12s for RTR ID of any format s
or x
    wait2 10 sR  ; wait 10s for standard RTR ID
    wait2 any    ; wait for the next message
```

wr {args}

send CAN message with standard identifier

Arguments

args - a CAN message. It is specified in the format
id [r] data-bytes

Results

nothing

Example

```
    wr 0x620 r 0x30 0xff
    wr 0x20    0x69 0x55
```

wrx {args}

send CAN message with extended identifier

Arguments

args - a CAN message. It is specified in the format
id [r] data-bytes

Results

nothing

Example

```
wrx 0x820 r 0x30 0xff
wrx 0x20    0x69 0x55
```

Das folgende Bild zeigt ein Beispiel für eine interaktiv eingegebene Kommandofolge:

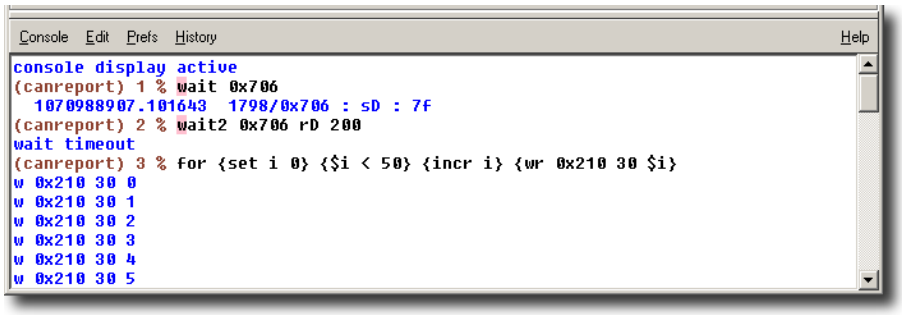


Bild 27, Interaktive Kommandoeingabe

Damit kann man die zugehörigen Nachrichten am Bus beobachten:

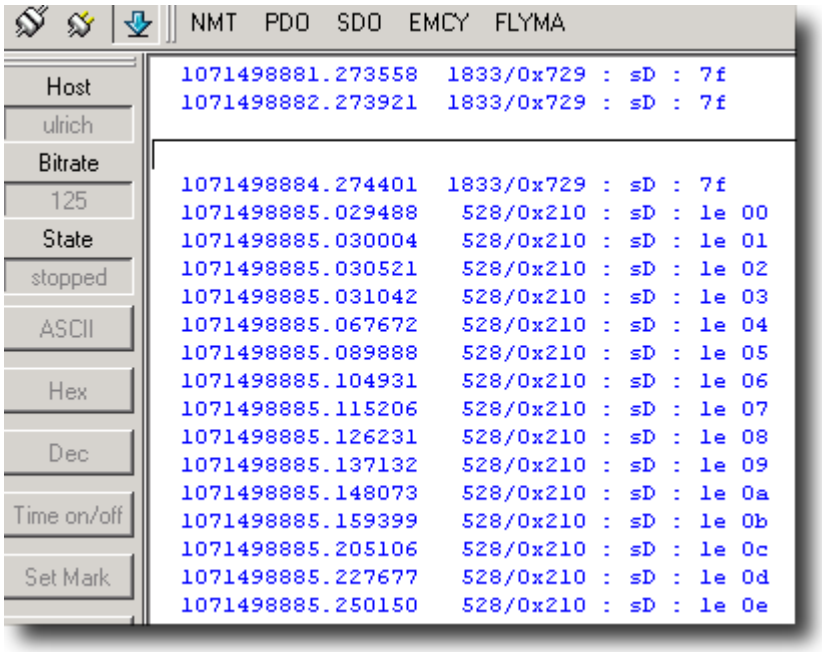


Bild 28, Ergebnis der Kommandos aus Bild 4

Die Kommandos können zu Sequenzen oder Prozeduren zusammengefaßt werden. Es stehen somit alle Sprachmittel moderner Hochsprachen zur Verfügung.

Vor allem für Inbetriebnahme und Fehleranalyse ist eine möglichst genaue Zeitauflösung der am Bus gesendeten Telegramme notwendig. Die vom CAN-Analyzer dargestellte Sende- bzw. Empfangszeit wird dabei nur von der eingesetzten Hardware beeinflußt. Beim Einsatz von CAN-Interfaces unter LINUX können sogar Zeitauflösungen im µs Bereich erreicht werden.

Die vorhandenen Protokollierungsfunktionen gestatten den gesamten Testablauf, aber auch die verschiedenen Logging-Fenster Inhalte in separaten Dateien zu sichern. Weiterhin besteht die Möglichkeit CAN Nachrichten direkt in eine Datei zu schreiben, ohne sie auf dem Bildschirm anzuzeigen. Diese Dateien können dann nachträglich durch die CANopen- oder benutzerdefinierte Erweiterungen interpretiert werden. Dies ist besonders bei hohen Buslasten sinnvoll.

10.2. Interpretation von CAN-Telegrammen

Der *CAN-REport* erlaubt die Interpretation von CAN-Telegrammen und deren Inhalt. Speziell für das Anwendungsprotokoll CANopen wurde die Interpretation schon vorgenommen. Die Taste "User Extension" in dem File-Menü lädt die Datei <application.tcl> und es wird ein Taster "User" in der Werkzeugleiste angelegt. Diese Datei muß im Arbeitsverzeichnis vorhanden sein. Bei Betätigung des Tasters erscheint ein neues Fenster. Der Anwender hat die Möglichkeit die empfangenen Telegramme auszuwerten und die Ergebnisse in diesem Fenster darzustellen. Der Widget-Pfad zu diesem Fenster lautet `.usr`. Mit dem *CAN-REport* wurde eine Vorlage mitgeliefert, wie eine Interpretation aussehen kann. Diese Vorlage befindet sich in der Datei application.tcl Diese enthält die Prozedur `application`. Diese Prozedur wird beim Eintreffen eines CAN-Telegramms aufgerufen.

```
proc application { msg } {

}
```

Diese Prozedur erwartet den Parameter `msg`. Er enthält das komplette Telegramm mit Zeitstempel, Telegrammnummer, Datentyp und Daten. Die Felder sind durch Leerzeichen getrennt.

```
      11.123  256/0x100 : sD : 00 01 02 03 04 05 06 07
#     ^       ^           ^    ^
#       time  id dec/hex type data 1..8 (if available)
```

Es wurden bereits Prozeduren vorbereitet, um die einzelnen Daten aus der Zeile zu erhalten.

Overview of user function

InsertText {win string tag chan}	- *Insert text <string> into user window*
get_can_id {msg}	- *extract CAN message identifier from message string*
get_data {msg}	- *return the data of a the message*
get_dlc {msg}	- *return the data length code of the message*
get_time {msg}	- *return time of message if time was set*
get_type {msg}	- *return the type of the message*

is_std {msg} *- check if the message is a standard CAN message*

Detailed function description

InsertText {win string tag chan}

Insert text <string> into user window

Arguments

win - widget-path .usr
string - Text to insert into the text-widget
tag - tag, that defines the font
the tags stdout and stderr are already defined

Results

Inserts a text in the user window

Example

```
InsertText .usr "hello world!" stdout
InsertText .usr "$msg" stdout
```

get_can_id {msg}

extract CAN message identifier from message string

Arguments

msg the standard message string provided for the procedure application {}

Results

the CAN-Id as a decimal value

get_data {msg}

return the data of a the message

Arguments

msg the standard message string provided for the procedure application {}

Results

list containing the databyes of the CAN message
in case it is an RTR message an empty string is returned:

get_dlc {msg}

return the data length code of the message

Arguments

msg the standard message string provided for the procedure application {}

Results

an integer value between 0 and 8 for a valid msg
an empty string for an invalid msg

get_time {msg}

return time of message if time was set

Return time information if time was set.
When time wasn't set the an empty string is returned

Arguments

msg the standard message string provided for the procedure application {}

Results

time - when time is on
empty string - when time is off

get_type {msg}

return the type of the message

Arguments

msg the standard message string provided for the procedure application {}

Results

D- Data frame
R- Remote frame

is_std {msg}

check if the message is a standard CAN message

Arguments msg the standard message string provided for the procedure application {}

Results

1 if the message is a standard CAN-Message
0 if the message is a extended CAN-Message

10.2.1. Beispiel

```
#The procedure application receives the message and interprets it.
#It’s an example for an user-defined interpretation of CAN-Messages.
#

proc application { msg } {

    set tag stdout
    set msg ""

    # get id
    set id [get_can_id $msg]
    if { $id eq "" } {
     return
    }

    if { $id == 0x201 } {
     # extract data
     set data [get_data $msg]

     set len [get_dlc $msg]
     for { set i 0 } { $i < $len } { incr i } {
         set byte$i [lindex $data $i]
     }
     if { $byte0 == 0x55 } {
         set msg "Start processing"
     }
    }

    if { $id == 0x281 } {
     # extract data
     set data [get_data $msg]

     set len [get_dlc $msg]
     for { set i 0 } { $i < $len } { incr i } {
         set byte$i [lindex $data $i]
     }
     if { $byte0 == 0x1F } {
         set msg "zero limit switch reached"
     }
    }
```

```
    InsertText .usr $msg $tag
    return
}
```

10.3. Die Bedienoberfläche erweitern

Es besteht die Möglichkeit die Bedienoberfläche zu erweitern oder an individuelle Gegebenheiten anzupassen. Dabei können neben der Funktionalität der Scriptsprache auch die Elemente der grafischen Oberfläche genutzt werden, um Test- oder Steuerapplikationen zu erstellen. Mit den vielfältigen Protokollierungsmöglichkeiten sind Testabläufe somit problemlos auch über das Netzwerk hinweg automatisierbar.

port GmbH ist gern bereit, die Bedienoberfläche des Analyzers für bestimmte Protokollfamilien oder an spezielle Kundengegebenheiten für Service- und Fertigungsaufgaben anzupassen.

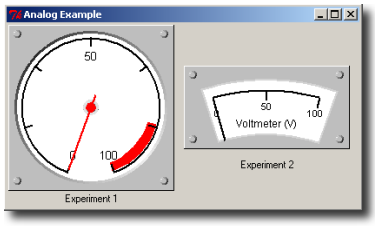


Bild 29, Beispiel Analog

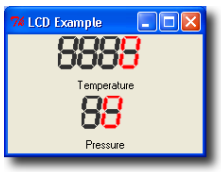


Bild 30, Beispiel LCD

Die beigefügten Beispiele benutzen den "User Extension"-Mechnismus des *CAN-REport*. Deshalb muss vor Benutzung eines Beispiels, der Inhalt des Beispielverzeichnisses in das Verzeichnis vom *CAN-REport* kopiert werden.
Wenn die Datei application.tcl von einem Beispiel über File→Open geladen wird, wird nur die graphische Oberfläche angezeigt. Die Graphik wird bei eintreffenden CAN-Telegrammen nicht aktualisiert.

Demo-Modus

Im Demo-Modus läuft in der Console ein Skript ab und es werden die Beispiele aus dem Verzeichnis examples geöffnet

10.4. CANopen-Funktionen

Die CANopen Erweiterung bietet eine Grundfunktionalität, um auf das Objectverzeichnis eines CANopen SDO-Server zuzugreifen. Dazu arbeitet der *CAN-REport* wie ein CANopen SDO-Client. Die Lese- und Schreibfunktion benutzten den Expetited-SDO-Transfer. Mit dem Expetited-SDO-Transfer können bis zu 4 Bytes übertragen werden.

Overview of user function

r {idx sidx dt}	*- read data with expetited SDO transer*
setRemoteID {nid}	*- set remote Id for SDO transfers*
set_sdo_timeout {val}	*- set timeout for SDO-Transfer*
w {idx sidx dt val}	*- write data with expetited SDO transer*

Detailed function description

r {idx sidx dt}

read data with expetited SDO transer

This function is only available with the
CANopen extension.
The node id has to be set previously

Arguments

idx - Index of the server object
sidx - SubIndex of the server object
dt - Datatype of the value

Results

"ERROR ..." - Error, reason is given
value - Success

setRemoteID {nid}

set remote Id for SDO transfers

This function is only available with the CANopen extension.
The initial value is 0x20.

Arguments

nid - new node id

Results

0 - Error
1 - Success

set_sdo_timeout {val}

set timeout for SDO-Transfer

This function is only available with the CANopen extension.
The initial value is 1000 ms.

Arguments

val - new timeout (ms)

Results

w {idx sidx dt val}

write data with expetited SDO transer

This function is only available with the
CANopen extension.
The node id has to be set previously

Arguments

idx - Index of the server object
sidx - SubIndex of the server object
dt - Datatype of the value
val - value itself

Results

"ERROR ..." - Error, reason is given
"OK" - Success

11. Konfiguration

11.1. Konfigurationsdateien

Die Konfiguration für *CAN-REport* wird in den Dateien <canreport.rc> und <co_cfg.rc> gespeichert. Diese Dateien müssen sich im Arbeitsverzeichnis von *CAN-REport* befinden. In den Dateien werden Kommentare mit dem Zeichen '#' am Anfang gekennzeichnet. Die Anweisungen setzen Variablen, die das Verhalten bzw. eine Option steuern. Die meisten Optionen können über Maskeneingaben gesetzt werden. Die folgende Tabelle zeigt die Optionen, die nicht über eine Maskeneingabe eingestellt werden können.

Variablenname	Werte	Beschreibung
ST(showSplash)	0,1	aktiviert/deaktiviert die Anzeige des Startbildes
ST(txch)	0 - 20	Anzahl der Senderegister in der Sendeleiste
ST(line1,font-XXX)	Zeichensatz	Zeichensatz für das CAN-Telegramm xxx
ST(line1,font-XXX,foreground)	Farbe	Schriftfarbe, der für den Zeichensatz des CAN-Telegramms xxx benutzt wird
ST(line1,font-XXX,background)	Farbe	Schrifthintergrund, der für den Zeichensatz des CAN-Telegramms xxx benutzt wird

XXX kann folgende Werte annehmen: stderr, stdout, usr; stdout wird im Hauptfenster verwendet. Die CANopen Erweiterung definiert für jeden Dienst weitere Farben. Diese Werte werden in der Datei <co_cfg.rc> gespeichert. XXX kann folgende Werte annehmen: nmtcmd, nmtstate, lssreq, lssresp, pdo, sync, srdo, sdoreq, sdoresp, sdoerr, emcy, flyma.

Die Änderung der Hintergrundfarbe setzt die Markierungsfarbe auf denselben Wert.

11.1.1. Ausschnitt einer Konfigurationsdatei

```
set ST(line1,font-stdout) {-*-Courier-Medium-*-*-12-*-*-*-*-*-*-*-*-*}
set ST(line1,font-stdout,foreground) blue
set ST(line1,font-stdout,background)

set ST(line1,font-stderr) "Helvetica 18 bold"
set ST(line1,font-stderr,foreground) #00FF00
set ST(line1,font-stderr,background)
```

11.2. Konfiguration serieller Schnittstellen

Die jeweilige Hardware kann im nachfolgenden Konfigurationsdialog ausgewählt werden. Unter *Serial Port* wird die serielle Schnittstelle gewählt und *Bitrate* bestimmt die CAN-Baudrate.

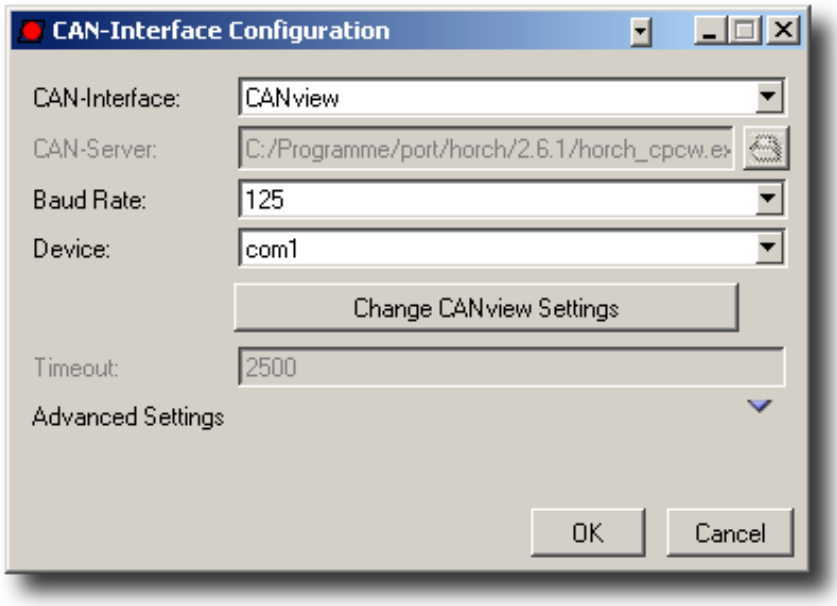


Bild 31, Konfigurationsdialog

11.2.1. CAN232 - Konfiguration

Vor Beginn der Konfiguration muss sichergestellt werden, dass das RS232-Modul mit Spannung versorgt ist und über die richtige RS232-Schnittstelle an den PC angeschlossen ist. Nach Betätigung der Schaltfläche *Configure CAN232* versucht der *CAN-REport* mit dem CAN232-Modul eine Verbindung herzustellen. Es werden die Baudraten 230400, 115200, 57600, 38400, 19200, 9600, 2400 getestet. Antwortet das CAN232-Modul mit einer bestimmten RS232-Baudrate, so wird diese angezeigt und ein Dialog zur Änderung der Baudrate wird geöffnet.

Hinweis: Der Baudratensuchvorgang wird mit Windows 98 nicht genutzt. Die Ausgangsbaudrate muss von Hand eingestellt werden.

11.2.2. CANview - Konfiguration

Nach Betätigung der Schaltfläche *Change CANview Settings* erscheint ein Dialogfenster zur Konfiguration der CANview-Einstellungen.

Bitte beachten Sie, dass dabei **nicht** das CANview konfiguriert wird, sondern die Konfiguration des Gerätes dem *CAN-REport* mitgeteilt werden muss. Zur Änderung der Geräteeinstellungen nutzen Sie bitte das mitgelieferte Konfigurationswerkzeug *RM Device Configurator*.

12. Systemanforderung

Die Bedienoberfläche des *CAN-REport* ist ausführbar auf allen Systemen, für die eine *Tcl/TK* Portierung verfügbar ist. Dazu gehören unter anderem PC's mit Microsoft Windows 9X/2000/NT oder diverse UNIX/LINUX Systeme und Macintosh.

Betriebssystem:	Windows 9X/NT/2000, LINUX, Max-OS
Prozessor:	Pentium und höher
RAM:	32 MByte
Festplattenspeicher:	5 MByte

13. Bestellinformation

0580/10 CAN Analyzer *CAN-REport*
0580/11 CANopen Erweiterung
0580/12 DeviceNet Erweiterung
0580/00 CAN Analyzer *CAN-REport* (Demo Version)
0580/03 CAN Analyzer *CAN-REport* (Evaluation Version)

Lieferung erfolgt immer mit ausgewählter Hardware. Nach Absprache kann eine Lieferung ohne Hardware erfolgen.

Einschränkungen

Folgende Kombinationen sind lieferfähig:

Hardware	Win32	LINUX	Mac
AT-CAN-MINI (port GmbH)		x	
Level-X (I+ME)	x		
EtherCAN (port GmbH)	x	x	x
CPC (EMS-Wünsche)	x	x	