ETAS

LABCAR-PINCONTROL V2.2

Benutzerhandbuch

Copyright

V2.2 R02 DE - 06.2016

Inhalt

1 Einführung

LABCAR-PINCONTROL V2.2 wird zusammen mit dem ES4440.1/.2 Compact Failure Simulation Module ausgeliefert. Das ES4440.1/.2 Compact Failure Simulation Module wird zur Echtzeit-Fehlersimulation bei Steuergeräten eingesetzt.

LABCAR-PINCONTROL V2.2 enthält eine Bedienoberfläche zur manuellen Steuerung und zur Konfiguration einer ES4440.1/.2 als auch einen COM-Controller für den automatisierten Betrieb.

Im Einzelnen können Sie mit LABCAR-PINCONTROL V2.2 insbesondere folgende Aufgaben durchführen:

- Manueller Test der Diagnosefunktionalität von Steuergeräten:
  - Erstellen und Verwalten von Failure Sets (ein Failure Set ist eine Gruppe von Steuergerätesignalen, z.B. von allen Lambdasondensignalen)
  - Einfache Auswahl eines einzelnen Signals zur Fehlersimulation
  - Setzen der Fehlerdauer und der Parameter für die Simulation von Wackelkontakten.
  - Aktivierung des Fehlers per Mausklick
- Konfiguration von ES4440.1/.2 Compact Failure Simulation Modules:
  - Spezifizieren von IP- und CAN-Adressen der eingesetzten Module
  - Konfiguration als Stand-Alone-, Master- oder Slave-System
  - Selbsttest und Sicherungstest der ES4440.1/.2
- COM-API zum Einsatz im automatisierten Testbetrieb

1.1 Über dieses Handbuch

Dieses Handbuch besteht aus den folgenden Kapiteln:

- „Einführung“ auf Seite 5

  Dieses Kapitel
- „Arbeiten mit LABCAR-PINCONTROL V2.2“ auf Seite 9

  In diesem Kapitel werden vorbereitende Konfigurationsaufgaben und die Bedienung von LABCAR-PINCONTROL V2.2 beschrieben.
- „API-Dokumentation“ auf Seite 33

  Diese Kapitel enthält Informationen zum automatisierten Betrieb des ES4440.1/.2 Compact Failure Simulation Module mit dem COM-Controller von LABCAR-PINCONTROL V2.2 oder über CAN.

1.1.1 Benutzerprofil

Dieses Handbuch richtet sich an Fachpersonal in den Bereichen Entwicklung und Test von Kfz-Steuergeräten. Fachwissen im Bereich Mess- und Steuergerätetechnik wird vorausgesetzt.

1.1.2 Umgang mit dem Handbuch

Darstellung von Information

Alle vom Anwender auszuführenden Tätigkeiten werden in einem sogenannten „Use-Case"-Format dargestellt. D.h., dass das zu erreichende Ziel zuerst in der Titelzeile kurz definiert wird, und die jeweiligen Schritte, die notwendig sind, um dieses Ziel zu erreichen, dann in einer Liste aufgeführt werden. Die Darstellung sieht wie folgt aus:

Zieldefinition

eventuelle Vorabinformation...

- Schritt 1

  eventuelle Erläuterung zu Schritt 1...
- Schritt 2

  eventuelle Erläuterung zu Schritt 2...
- Schritt 3

  eventuelle Erläuterung zu Schritt 3...

eventuelle abschließende Bemerkungen...

konkretes Beispiel:

Erstellen einer neuen Datei

Vor dem Erstellen einer neuen Datei darf keine andere geöffnet sein.

- Wählen Sie **Datei** → **Neu**.

  Die Dialogbox „Datei Erstellen" erscheint.
- Geben Sie den Namen für die Datei im Feld „Dateiname" ein.

  Der Dateiname darf nicht mehr als 8 Zeichen lang sein.
- Klicken Sie **OK**.

Die neue Datei wird erstellt und unter dem von ihnen angegebenen Namen abgelegt. Sie können nun mit der Datei arbeiten.

Typografische Konventionen

Folgende typografischen Konventionen werden verwendet:

Wählen Sie **Datei** → **Öffnen**.	Menübefehle werden fett/blau dargestellt.
Klicken Sie **OK**.	Schaltflächen werden fett/blau dargestellt.
Drücken Sie <EINGABE>.	Tastaturbefehle werden in spitzen Klammern in Kapitälchen dargestellt.

Das Dialogfenster „Datei öffnen“ erscheint.	Namen von Programmfenstern, Dialogfenstern, Feldern u.ä. werden in Anführungszeichen gesetzt.
Wählen Sie die Datei `setup.exe` aus.	Text in Auswahllisten, Programmcode, sowie Pfad- und Dateinamen werden in der Schriftart `Courier` dargestellt.
Eine Konvertierung zwischen den Datentypen logisch und arithmetisch ist *nicht* möglich.	Inhaltliche Hervorhebungen und neu eingeführte Begriffe werden *kursiv* gesetzt

2 Arbeiten mit LABCAR-PINCONTROL V2.2

In diesem Kapitel werden vorbereitende Konfigurationsaufgaben und die Bedienung von LABCAR-PINCONTROL V2.2 beschrieben.

- „LABCAR-PINCONTROL V2.2 konfigurieren“ auf Seite 9

  In diesem Abschnitt finden Sie eine Beschreibung vorbereitender Aufgaben.
- „Das Wire Harness File“ auf Seite 14

  In diesem Abschnitt wird beschrieben, wie Sie ein Wire Harness File erstellen.
- „LABCAR-PINCONTROL V2.2 bedienen“ auf Seite 19

  Dieser Abschnitt enthält eine Beschreibung aller Aktionen, die Sie in der Bedienoberfläche zur Simulation von Fehlern ausführen können.
- „Das Hauptmenü“ auf Seite 31

  Dieser Abschnitt enthält eine zusammenfassende Beschreibung des Hauptmenüs von LABCAR-PINCONTROL V2.2.

2.1 LABCAR-PINCONTROL V2.2 konfigurieren

Bevor Sie mit PINCONTROL arbeiten können, müssen Sie bestimmte Einstellungen bezüglich der Ethernetverbindungen und die eingesetzte Hardware betreffend vornehmen.

2.1.1 Konfiguration der Netzwerkkarte des Hostrechners

Bei der Netzwerkkarte, die zur Steuerung des ESES4440.1/.2 Compact Failure Simulation Module verwendet wird, müssen folgende Einstellungen vorgenommen werden.

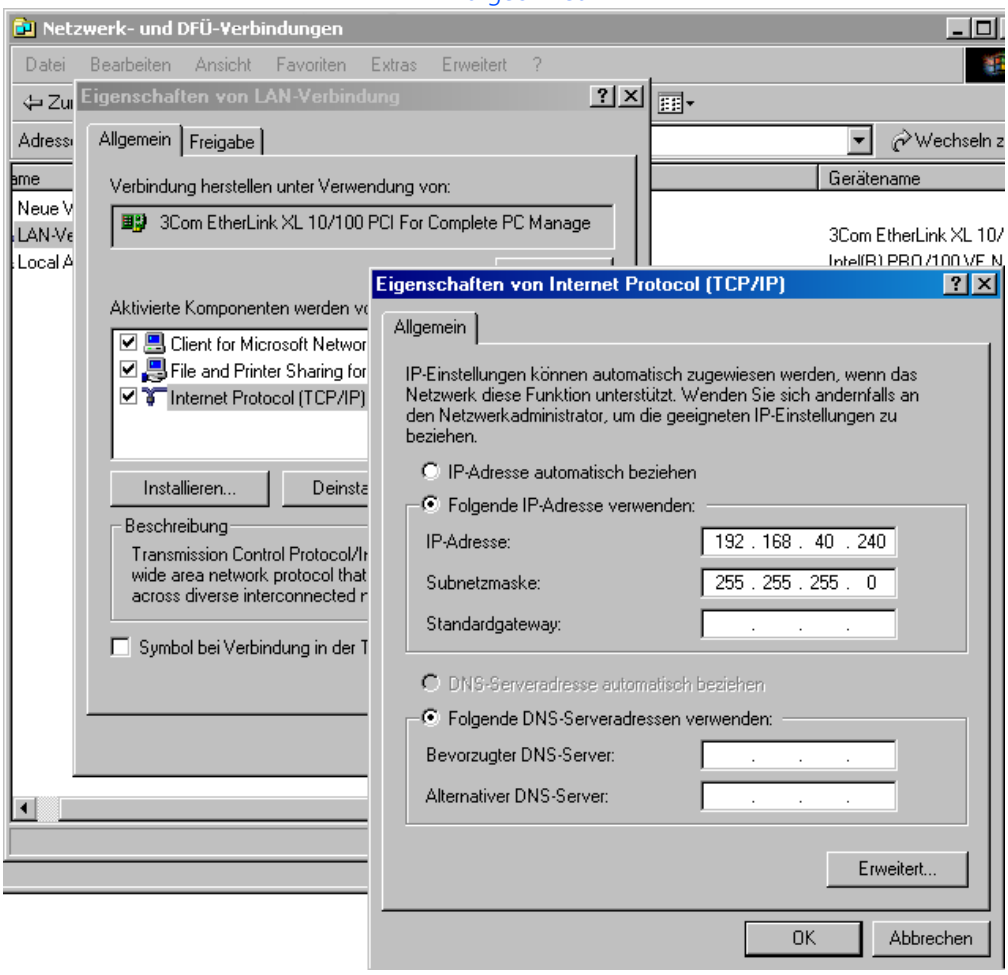
TCP/IP konfigurieren

- Wählen Sie im Windows Startmenü **Einstellungen** → **Systemsteuerung**.
- Doppelklicken Sie im Fenster der Systemsteuerung **Netzwerk- und DFÜ-Verbindungen**.
- Wählen Sie die gewünschte Verbindung/das gewünschte Gerät.
- Rechtsklicken Sie den Eintrag und wählen Sie **Eigenschaften**.

  Das Fenster „Eigenschaften von <*Name*>“ wird geöffnet.
- Wählen Sie die Komponente „Internet Protokoll (TCP/IP)“.

- Klicken Sie **Eigenschaften**.

  Das Fenster „Eigenschaften von Internet Protokoll“ wird geöffnet.



- Wählen Sie die in der Abbildung gezeigten Einstellungen:
  IP-Adresse: 192.168.40.240
  Subnetzmaske: 255.255.255.0

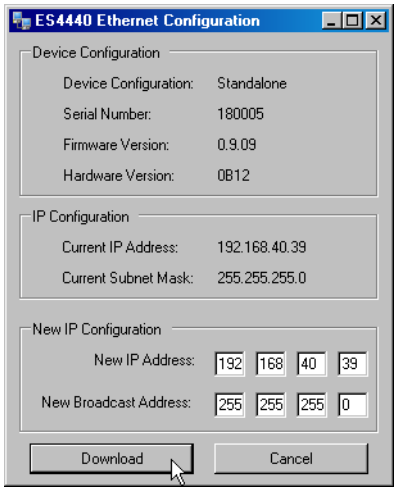
  Sie können aber auch jede andere gültige IP-Adresse verwenden.
- Schließen Sie alle Fenster mit **OK**.

2.1.2 Ethernet-Konfiguration

In diesem Abschnitt wird beschrieben, wie Sie den eingesetzten ES4440.1/.2 Compact Failure Simulation Modules Ethernetadressen zuweisen.

Etherneteinstellungen vornehmen

- Im Hauptmenü von LABCAR-PINCONTROL V2.2 wählen Sie **Tools** → **ES4440 Ethernet Configuration**.
- Sie erhalten eine Warnung, dass während der Konfiguration nur ein einziges Gerät (d.h. die zu konfigurierende ES4440.1/.2) eingeschaltet sein darf.
- Klicken Sie **OK**.
- Das Dialogfenster „ES4440 Ethernet Configuration“ wird geöffnet.



In diesem Fenster erhalten Sie Konfigurationsinformationen, inbesondere zur eingestellten IP-Adresse und Subnetzmaske.

- Im Feld „New IP Configuration“ können Sie neue Daten eingeben.
- Klicken Sie **Download**, um die geänderten Daten zur Hardware herunterzuladen.

oder

- Klicken Sie **Cancel**, um den Dialog ohne Änderungen zu verlassen.

2.1.3 ES4440.1 Systemkonfiguration

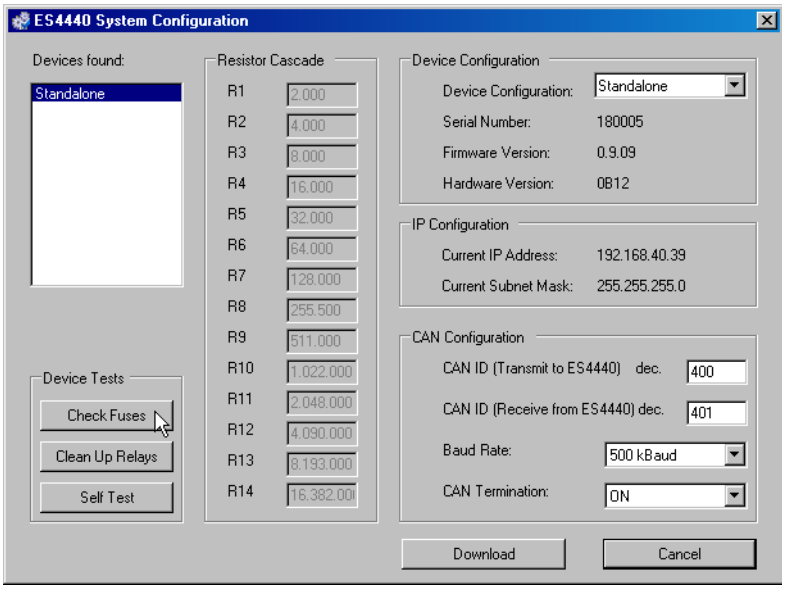
Die Systemkonfiguration besteht im Zuweisen des Status der einzelnen Hardware (Standalone, Master, Slave) und in der Konfiguration der CAN-Schnittstelle (nur wenn die Steuerung über CAN erfolgen soll).

Hinweis

Die in diesem Abschnitt beschriebene Systemkonfiguration ist nur möglich, wenn zur jeweiligen Hardware eine Ethernetverbindung besteht!

- Im Hauptmenü von LABCAR-PINCONTROL V2.2 wählen Sie **Tools** → **ES4440 System Configuration**.

  Das Dialogfenster „ES4440 System Configuration“ wird geöffnet.



  In diesem Fenster können Sie eine Reihe von Einstellungen vornehmen, die im Folgenden beschrieben werden.

Device konfigurieren

- Wählen Sie im Feld „Devices found“ diejenige ES4440.1/.2, für die die Einstellungen gelten sollen.
- Im Feld „Device Configuration“ können Sie der jeweiligen Karte ihren Status zuweisen (Standalone, Master, Slave1, ...,Slave14).
- Wenn die Konfiguration mit diesem Schritt vollständig ist, klicken Sie **Download**.

  Die Konfiguration wird durchgeführt und das Dialogfenster geschlossen.

- Andernfalls fahren Sie mit weiteren Einstellungen fort und beenden die Konfiguration anschließend wie eben beschrieben.

IP-Konfiguration einsehen

Im Feld „IP Configuration“ finden Sie IP-Adresse und Subnetzmaske der aktuell ausgewählten Hardware.

CAN-Schnittstelle konfigurieren

- Wenn Sie Ihre Hardware via CAN-Schnittstelle steuern wollen, müssen Sie im Feld „CAN Configuration“ die dafür erforderlichen Einstellungen vornehmen.
  - CAN ID (Transmit to ES4440) dec.

    Hier legen Sie die ID des jeweiligen Gerätes fest, die in einer Send-Botschaft für dieses Gerät enthalten sein muss.
  - CAN ID (Receive from ES4440) dec.

    Hier legen Sie die ID des jeweiligen Gerätes fest, die in einer Receive-Botschaft von diesem Gerät enthalten ist.
  - Baud Rate:

    Hier stellen Sie die Übertragungsrate ein - wählbar sind „500 kBaud“ oder „1 MBaud“.
  - CAN Termination:

    Hier können Sie festlegen, ob das jeweilige Gerät eine CAN-Terminierung besitzt oder nicht.

Informationen über die Widerstandskaskade einsehen

Im Feld „Resistor Cascade“ werden die tatsächlichen Widerstandswerte der internen Widerstandskaskade dargestellt.

Relais reinigen und Tests durchführen

- Eine Beschreibung der Aktionen, die Sie im Feld „Device Tests“ finden Sie unter „Reinigen der Relaiskontakte“ auf Seite 28, „Sicherungen testen“ auf Seite 29 und „Selbsttest durchführen“ auf Seite 30.

2.2 Das Wire Harness File

Das Wire Harness File ist ein wichtiger Bestandteil eines Projektes - es enthält eine Beschreibung, welche die Steuergerätesignale an welche Kanäle eines ES4440.1/.2 Compact Failure Simulation Module verbunden sind.

Für einen Steuergerätekanal muss damit ein Satz mit den folgenden Daten vorliegen:

- ECU Name
- Pin Number
- ES4440 Name
- Channel Number

Der Name des Pins (Pin Name) ist als erklärende Ergänzung zur Pin Number gedacht und muss nicht eindeutig sein.

Die folgende Abbildung enthält ein Beispiel:

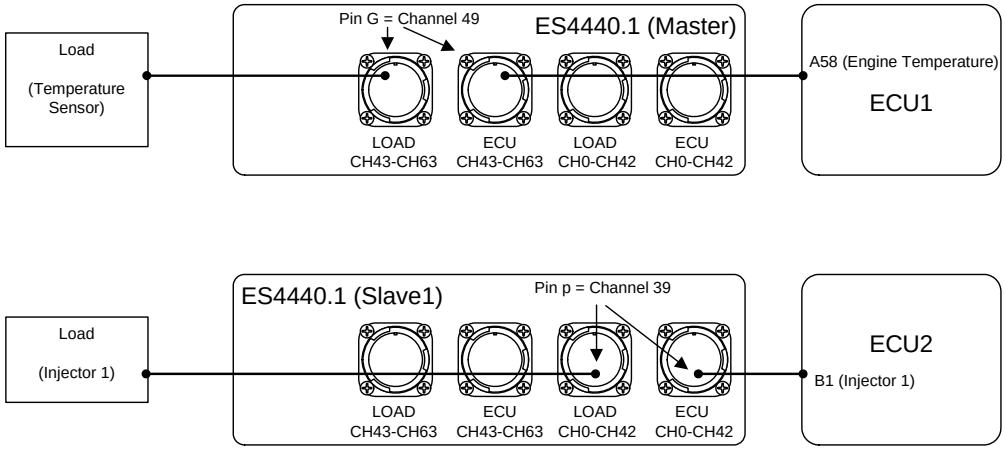


Abb. 2-1 Beispielhafter Anschluss von Steuergerätesignalen und Lasten

In der Abbildung sind zwei Steuergeräte an zwei ES4440.1/.2 angeschlossen:

- ECU Name: ECU1/ECU2
- Pin Number: A58/B1
- Pin Name: Engine Temperature/Injector 1
- ES4440 Name: Master/Slave1
- ES4440 Connector: ECU/LOAD CH43-CH63 / ECU/LOAD CH0-CH42
- ES4440 Pin: G = Channel 49 / p = Channel 39

2.2.1 Erstellung eines Wire Harness File

In diesem Abschnitt finden Sie ein Beispiel, wie Sie aus Kabelbaumdaten, die in einem Excel-Sheet abgelegt sind, ein Wire Harness File für Ihr LABCAR-PINCONTROL V2.2 Projekt erstellen.

- Unter Windows XP:

  `C:\Documents and Settings\All Users\ Anwendungsdaten\ETAS\LABCAR-PINCONTROL\2.1\Excel\ LABCAR-PINCONTROLV2.0_Example.xls`.

- Unter Windows Vista/Windows 7:

  `C:\ProgramData\ETAS\LABCAR-PINCONTROL\2.1\Excel\LABCAR-PINCONTROLV2.0_Example.xls`

Die Excel-Datei besteht aus 3 Tabellen:

- **WireHarnessData**

  In dieser Tabelle werden die Mappingdaten zwischen Ihren Steuergeräteanschlüssen und den ES4440-Anschlüssen definiert.

- **ES4440WireHarnessSignals**

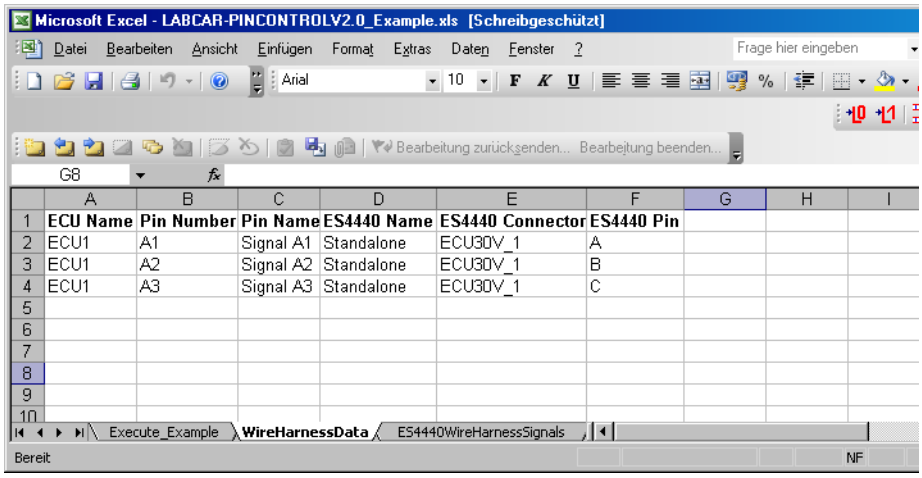
  Auf diesem Tabellenblatt werden bestimmte Einstellungen vorgenommen, damit die Daten in der Tabelle „WireHarnessData“ ausgewertet werden können.

- **Execute_Example**

  Diese Tabelle bietet die Möglichkeit, das Makro auszuführen und einen einfachen „Open Load“ Fehler an die ES4440 anzulegen (funktioniert nur mit diesem Beispiel)

Die Tabelle „WireHarnessData“

In dieser Tabelle muss die Zuordnung zwischen den Steuergeräteanschlüssen und ES4440-Anschlüssen festgelegt werden.



	A	B	C	D	E	F	G	H	I
1	**ECU Name**	**Pin Number**	**Pin Name**	**ES4440 Name**	**ES4440 Connector**	**ES4440 Pin**			
2	ECU1	A1	Signal A1	Standalone	ECU30V_1	A			
3	ECU1	A2	Signal A2	Standalone	ECU30V_1	B			
4	ECU1	A3	Signal A3	Standalone	ECU30V_1	C			
5									
6									
7									
8									
9									
10									

Die einzelnen Spalten enthalten folgende Informationen:

- **ECU Name**

  Der Name Ihrer ECU (muss eindeutig sein).

- **Pin Number**

  Der Pin der ECU (muss eindeutig sein, muss aber keine Zahl sein).

- **Pin Name**

  Mögliche weitere Bezeichnung des Pins, z.B. Name des Signals, das an diesem Pin anliegt (muss nicht eindeutig sein).

- **ES4440 Name**

  Der Name der ES4440, mit der der Pin verbunden ist. (mögliche Namen finden Sie in der Tabelle „ES4440WireHarnessSignals" unter „ES4440 Device Names“).

- **ES4440 Connector**

  Hier definieren Sie, über welchen Anschlussstecker das Signal zu der ES4440 gelangt (mögliche Namen finden Sie in der Tabelle „ES4440WireHarnessSignals“ unter „Connector/Pin/Channel/Electric Type“).

- **ES4440 Pin**

  Hier definieren Sie, mit welchem ES4440 Pin des oben definierten Steckers Ihr Steuergerätesignal verbunden ist (mögliche Namen finden Sie in Tabelle „ES4440WireHarnessSignals“ unter „Connector/Pin/Channel/Electric Type“).

Jede Zeile dieser Tabelle ergibt nun ein Mapping:

<**ECU Name**> + <**Pin Number**> (+ <**Pin Name**>) =
<**ES4440 Name**> + <**ES4440 Connector**> + <**ES4440 Pin**>

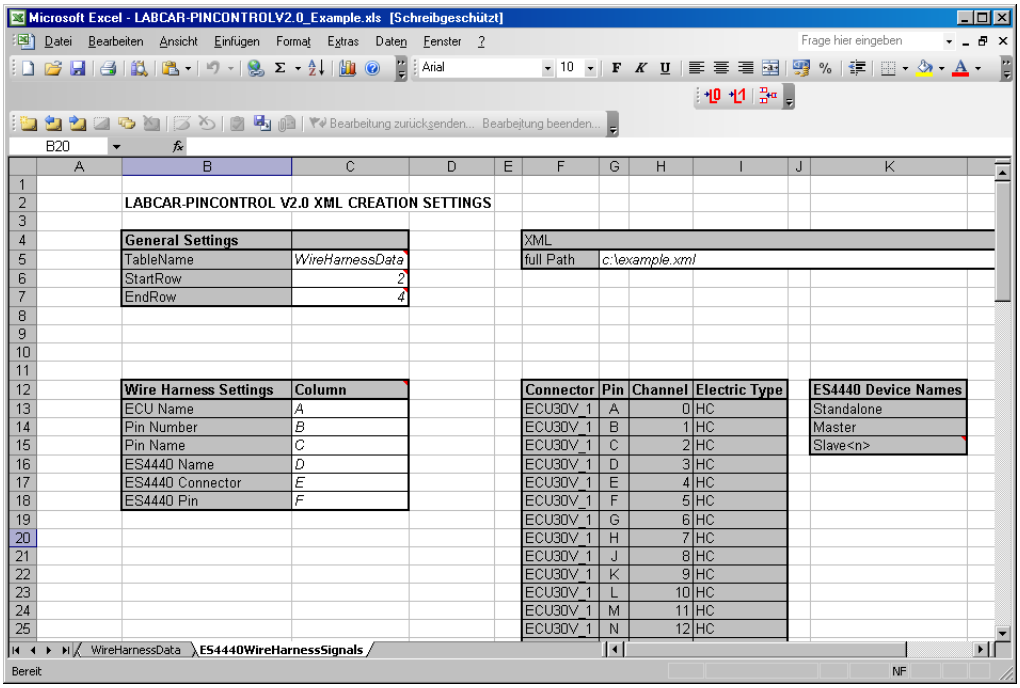
In dieser Darstellung wird auch deutlich, dass „Pin Name“ nicht eindeutig sein muss - er dient lediglich als zusätzliche Beschreibung zur „Pin Number“.

Wenn Sie nun alle Steuergeräte-Pins mit den zu verwendenden ES4440.1/.2 verbunden haben, so haben Sie Ihr Wire Harness File vollständig spezifiziert.

Nun müssen noch einige Einstellungen vorgenommen werden, damit Ihre Daten in die XML-Datei für LABCAR-PINCONTROL V2.2 transformiert werden können.

Anpassungen

Wählen Sie nun die Tabelle „ES4440WireHarnessSignals“ - diese Tabelle enthält alle Daten, die das Makro benötigt, um aus den vorher definierten Kabelbaumdaten die benötigte Datei zu erzeugen.



Diese Tabelle enthält folgende Informationen:

General Settings:

- **TableName**

  Hier definieren Sie, in welcher Tabelle sich die Daten „ECU Name“, „Pin Number“, „Pin Name“, „ES4440 Name“, „ES4440 Connector“, „ES4440 Pin“ befinden.

- **Start Row**

  Die erste auszuwertende Zeile der Tabelle „WireHarnessData“

- **End Row**

  Die letzte Zeile auszuwertende der Tabelle „WireHarnessData“

Wire Harness Settings:

Hier wird definiert, in welcher Spalte welche Information zu finden ist.

- **ECU Name**

  Die Spalte der Tabelle „WireHarnessData“, in der der Name des Steuergerätes definiert ist.

- **Pin Number**

  Die Spalte der Tabelle „WireHarnessData“, in der die Nummer des Steuergerätepins definiert ist.

- **ECU Pin Name**

  Die Spalte der Tabelle „WireHarnessData“, in der der ECU Pin Name definiert ist.

- **ES4440 Name**

  Die Spalte der Tabelle „WireHarnessData“, in der der Name der ES4440.1/.2 definiert ist.
  Die möglichen Namen sind in der Gruppe „ES4440 Device Names“ (s.u.) aufgeführt.

- **ES4440 Connector**

  Die Spalte der Tabelle „WireHarnessData“, in der der Anschluss der ES4440.1/.2 definiert ist. Die möglichen Anschlüsse sind in der Gruppe „Connector/Pin/Channel/Electric Type“ definiert.

- **ES4440 Pin**

  Die Spalte der Tabelle „WireHarnessData“, in der der Anschlusspin definiert ist.
  „ES4440 Pin“ muss zum jeweiligen „ES4440 Connector“ passen. Die möglichen Pins sind in der Gruppe „Connector/Pin/Channel/Electric Type“ (s.u.) definiert.

XML:

- **Full Path**

  Hier können Sie Namen und Pfad des vom Makro zu erzeugenden Wire Harness Files definieren.

ES4440 Device Names:

Definition der Namen aller eingesetzter ES4440.1/.2

Connector/Pin/Channel/Electric Type:

Enthält die komplette Beschreibung aller Anschlüsse einer ES4440.1/.2.

Hinweis

Ändern Sie keine Felder mit grauem Hintergrund. Dies kann dazu führen, dass eine fehlerhafte oder gar keine Datei erzeugt wird.

Erstellung des Wire Harness Files

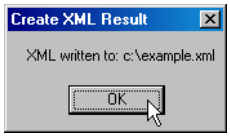
Nachdem alle Einstellungen vorgenommen worden sind, kann das Wire Harness File erzeugt werden.

- Wechsel Sie zur Tabelle „Execute_Example“.
- Klicken Sie die Schaltfläche **Create WireHarnessFile for ES4440**.

oder

- Wählen Sie **Extra** → **Macro** → **Macros...**.
- Wählen Sie das Makro „CreateXML“ und klicken Sie **Ausführen**.

  Die erfolgreiche Ausführung des Makros wird durch folgenden Dialog angezeigt:



2.3 LABCAR-PINCONTROL V2.2 bedienen

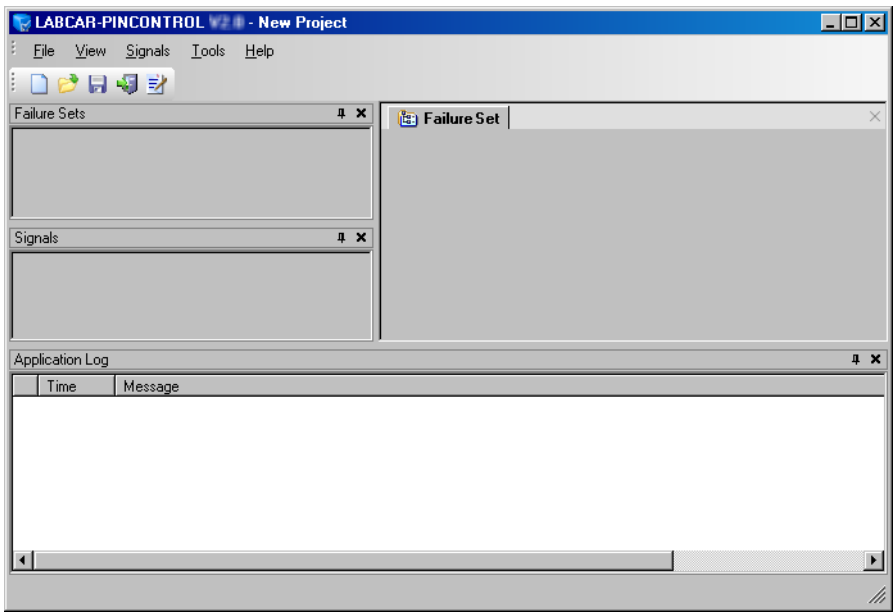
In diesem Abschnitt finden Sie eine Beschreibung aller wichtigen Schritte, um ein ES4440.1/.2 Compact Failure Simulation Module von der Bedienoberfläche von LABCAR-PINCONTROL V2.2 aus zu steuern.

Im Einzelnen finden Sie in diesem Abschnitt Beschreibungen der folgenden Benutzeraktionen:

LABCAR-PINCONTROL V2.2 starten

- Im Windows Startmenü wählen Sie **Programme** → **ETAS** → **LABCAR-PINCONTROL V2.2** → **PinControl.exe**.

  LABCAR-PINCONTROL V2.2 wird geöffnet.



Die Bedienoberfläche von LABCAR-PINCONTROL V2.2 besteht aus folgenden Teilen:

- Signals

  In diesem Fenster werden die Signale des projektspezifischen Wire Harness File dargestellt (siehe „Signale aus Wire Harness File importieren“ auf Seite 20).

- Failure Sets

  In diesem Fenster werden die vom Benutzer zu Failure Sets gruppierten Signale (mit zugewiesenen Fehler) dargestellt (siehe „Eine neues Failure Set erstellen“ auf Seite 21).

- Failure Set

  In diesem Bereich werden die Register mit den verschiedenen Fehlertypen angezeigt. Hier wird das aktuell gewählte Failure Set definiert, d.h. Fehler zu Signalen zugewiesen (siehe „Mehrfachfehler setzen“ auf Seite 24).

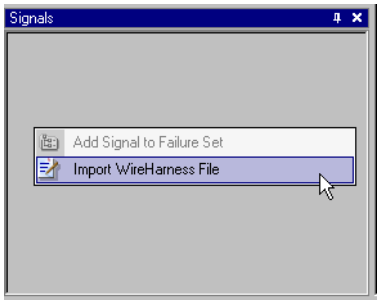
- Application Log

  In diesem Bereich werden Informationen und Meldungen ausgegeben.

Als ersten Schritt importieren Sie die Signale, indem Sie die Daten aus Ihrem Wire Harness File importieren.

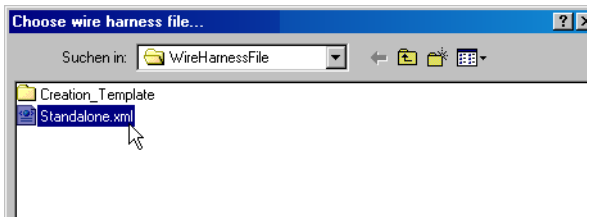
Signale aus Wire Harness File importieren

- Rechtsklicken Sie in das Fenster „Signals“.
- Aus dem Kontextmenü wählen Sie **Import WireHarness File**.

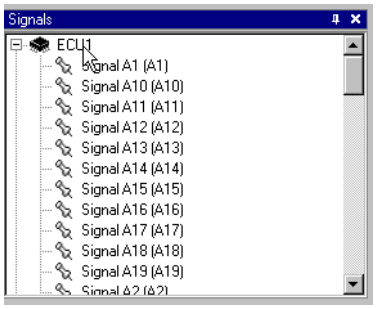


Ein Dateiauswahlfenster wird geöffnet.

- Wählen Sie die XML-Datei, die Ihre Kabelbaumspezifikation enthält.



Die Signale werden importiert und im Fenster „Signals“ angezeigt.

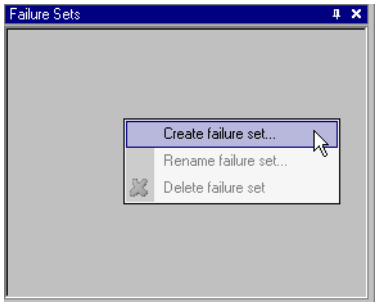


- Speichern Sie das Projekt mit **Save as**.

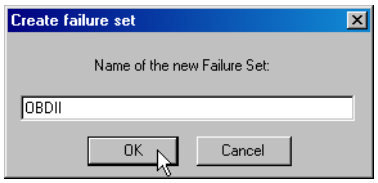
Jetzt liegen Ihnen alle Informationen vor, wie das Steuergerät mit dem ES4440.1/.2 Compact Failure Simulation Module verbunden ist. Als nächstes gruppieren Sie bestimmte, für Ihr aktuelles Projekt interessante Signale in einem (oder mehreren) Failure Set(s).

Eine neues Failure Set erstellen

- Rechtsklicken Sie in das Fenster „Failure Sets“.
- Aus dem Kontextmenü wählen Sie **Create failure set...**.



- Geben Sie einen Namen für das zu erstellende Failure Set an.

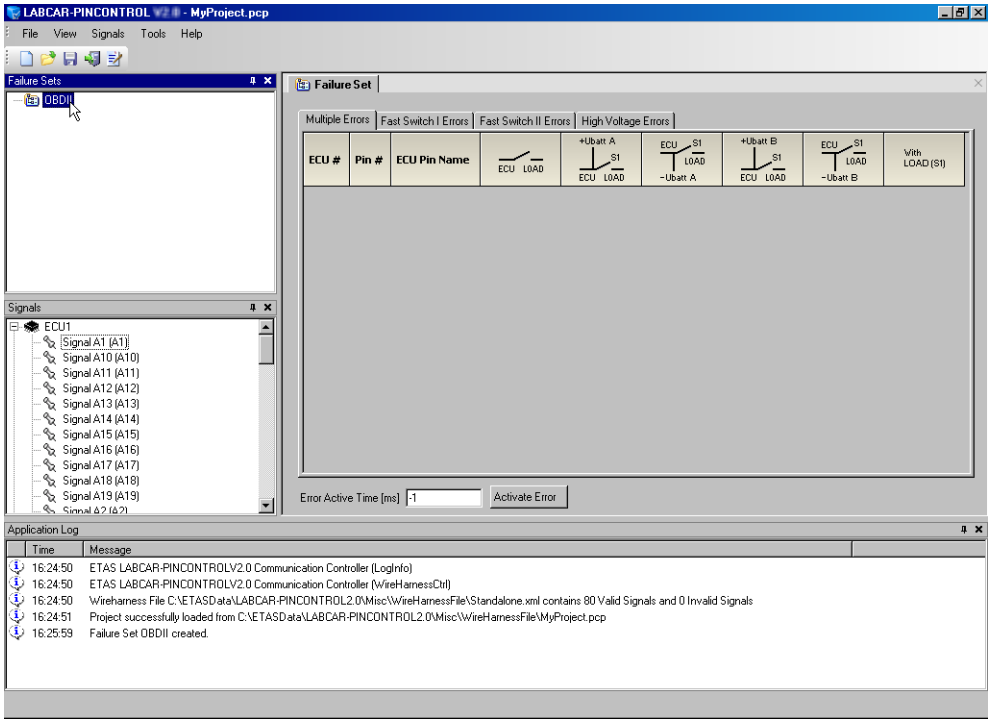


- Klicken Sie **OK**.

  Das Failure Set „OBDII“ wird erstellt.

- Klicken Sie das neu erstellte Failure Set an.

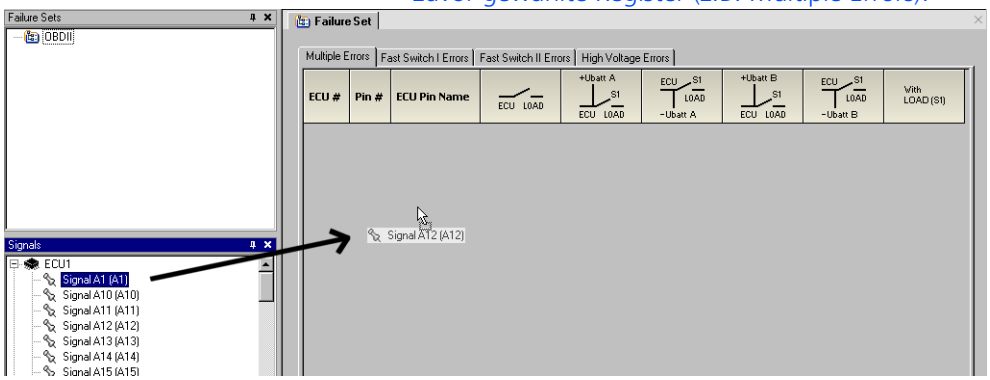
  Im rechten Teil der Bedienoberfläche werden jetzt (in verschiedenen Registern) alle Fehlerarten angezeigt, die verfügbar sind.



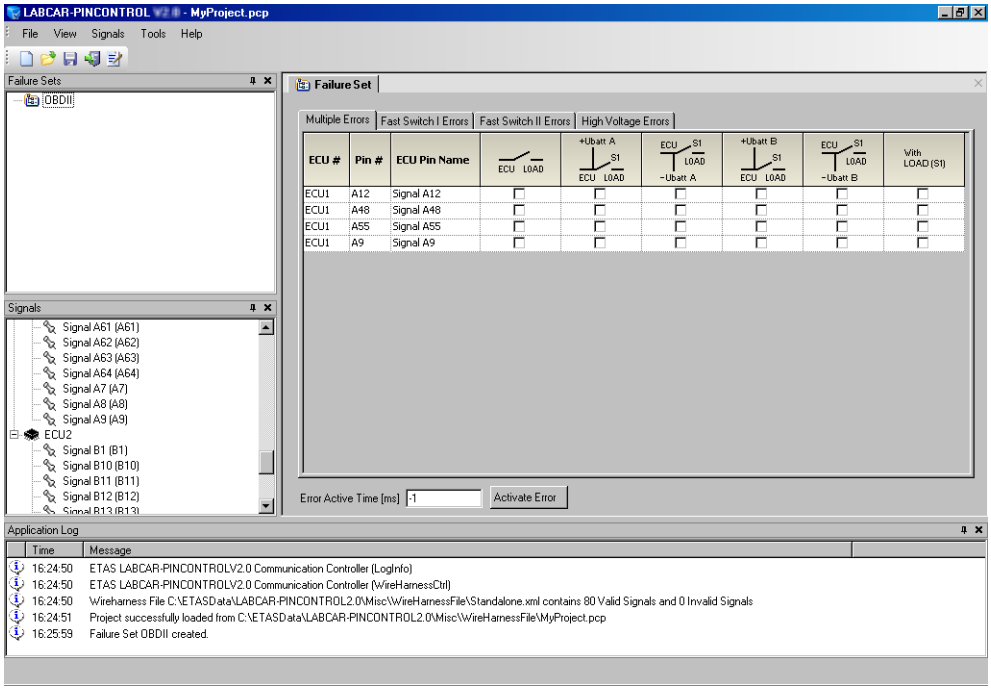
Sie werden jetzt das Failure Set „OBDII“ durch das Hinzufügen von Signalen vervollständigen.

Signal hinzufügen

- Wählen Sie im Fenster „Signals“ ein Signal.
- Ziehen Sie es (mit gedrückter Maustaste) in das zuvor gewählte Register (z.B. Multiple Errors).



- Verfahren Sie auf die selbe Weise mit weiteren Signalen.



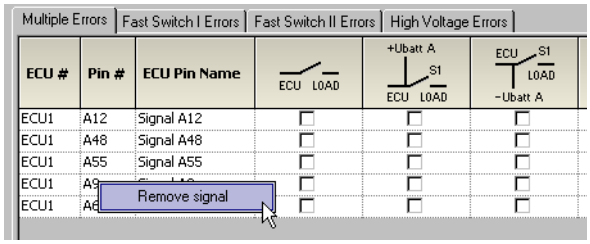
- Speichern Sie das Projekt.

  Das Failure Set ist jetzt Bestandteil Ihres Projekts.

  Bei Bedarf können Sie jetzt weitere Failure Sets auf die eben beschriebene Art erstellen.

Signal entfernen

- Wenn Sie ein irrtümlich hinzugefügtes Signal wieder aus dem Failure Set entfernen wollen, klicken Sie mit der rechten Maustaste auf dieses.
- Wählen Sie **Remove Signal**.

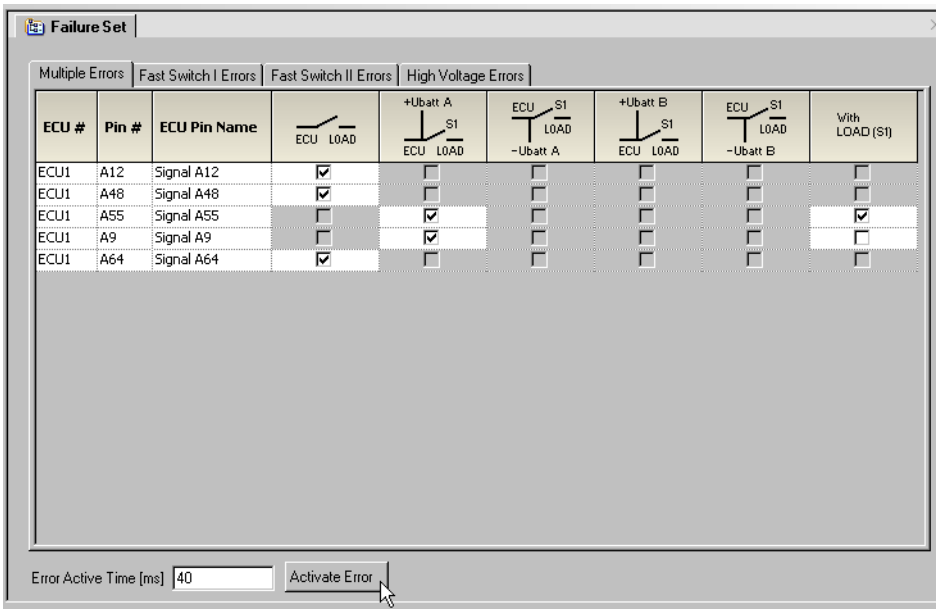


Das Signal wird aus Ihrem Failure Set entfernt.

Um jetzt für bestimmte Signale des Failure Sets Mehrfachfehler[1] zu setzen, gehen Sie wie folgt vor.

Mehrfachfehler setzen

- Um für die Signale „Signal A12“, „Signal A48“ und Signal A64“ eine Leitungsunterbrechung zu simulieren, wählen Sie die jeweiligen Kästchen.
- Für das Signal „Signal A55“ setzen Sie den Fehler „Kurzschluss gegen +UBatt_A“. Da bei diesem Fehler die Last angeschlossen bleiben soll, markieren Sie das entsprechende Kästchen in der Spalte „With Load (S1)“.
- Zuletzt wählen Sie für das Signal „Signal A9“ ebenfalls einen Kurzschluss gegen +UBatt_A.

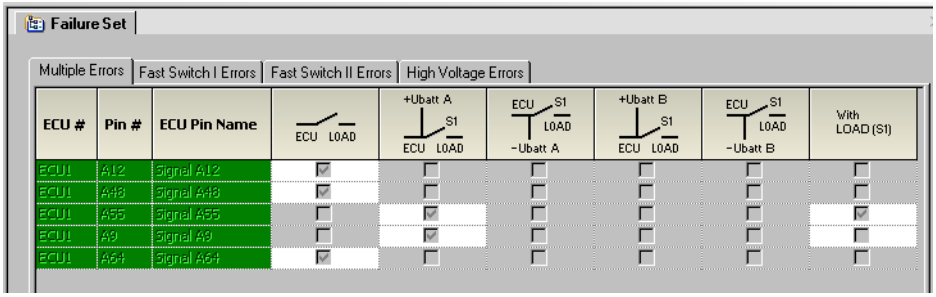


1. Information über die verschiedenen Fehlerarten finden Sie im Benutzerhandbuch des ES4440.1/.2 Compact Failure Simulation Module.

- Tragen Sie unten im Register die Zeit ein, für die die Fehler aktiv sein sollen.

  „-1“ bedeutet, dass die Fehler so lange anliegen, bis erneut **Activate Error** gedrückt wird.
- Klicken Sie **Activate Error**.

  Die Fehler werden aktiviert - während ihrer Ausführung werden die Signalbezeichnungen grün hinterlegt.



Einen Wackelkontakt simulieren

Bei Fehlern in den Registern „Fast Switch I Errors“ und „Fast Switch II Errors“ ist auch die Simulation von Wackelkontakten möglich, d.h. ein bestimmter Fehler liegt nicht für einen definierten Zeitraum an, sondern wird mit einer bestimmen Frequenz und Tastverhältnis geschaltet.

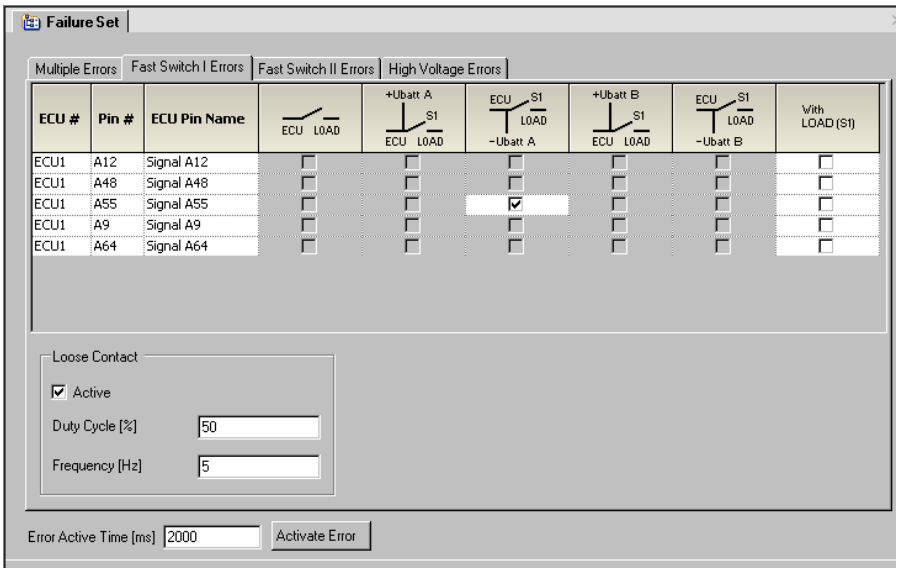
- Wechseln Sie z.B. in das Register „Fast Switch I Errors“.

Hinweis

Beim Wechsel in anderes Register werden die im aktuellen Register ausgewählten Fehler wieder gelöscht!

- Wählen Sie einen Fehler aus.
- Aktivieren Sie im Feld „Loose Contact“ die Option „Activate“.
- Geben Sie unter „Duty Cycle“ das gewünschte Tastverhältnis ein.

- Geben Sie unter „Frequency“ die gewünschte Frequenz ein.



- Geben Sie unter „Error Active Time“ die gewünschte Dauer der Wackelkontaktsimulation an.
- Zur Ausführung des Fehlers klicken Sie **Activate Error**.

Einen Kontaktwiderstand simulieren

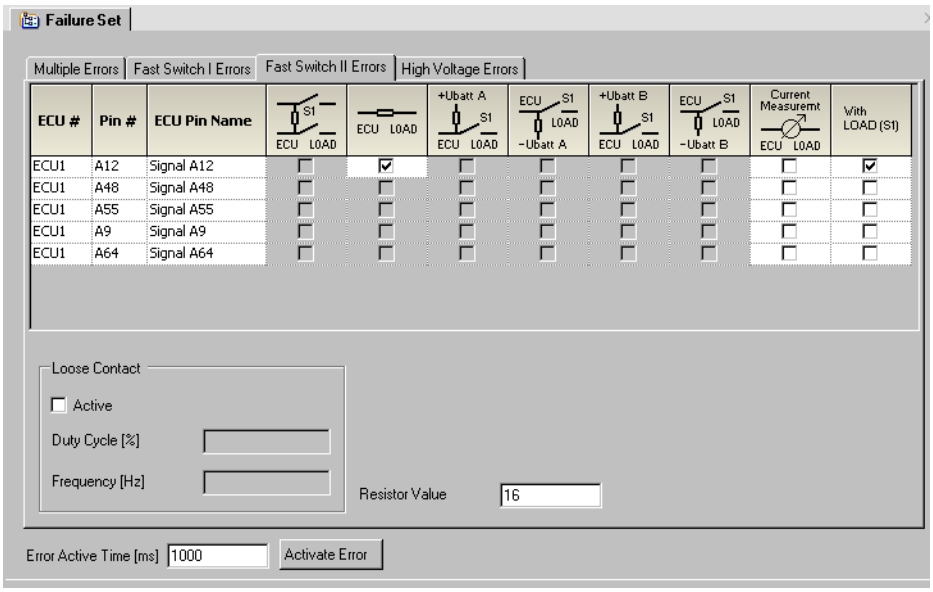
Die Fehler mit Kontaktwiderständen (Inline, Pin-to-Pin, Leckströme zu Batteriespannungen) finden Sie im Register „Fast Switch II Errors“.

- Wählen Sie in diesem Register einen Fehler aus (z.B. einen Leitungswiderstand).

Hinweis

Beachten Sie, dass Sie einem Pin-to-Pin-Fehler zwei Signale zuweisen müssen!

- Geben Sie im Feld „Resistor Value“ den gewünschten Widerstand ein.



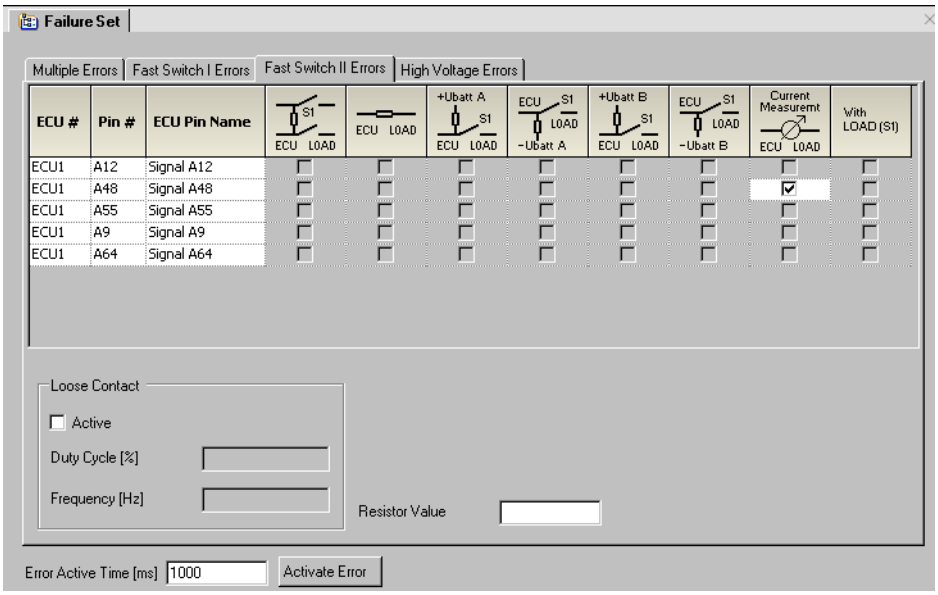
- Geben Sie unter „Error Active Time“ die gewünschte Dauer der Fehlersimulation an.
- Zur Ausführung des Fehlers klicken Sie **Activate Error**.

Strom messen

Im Register „Fast Switch II Errors“ gibt es die Möglichkeit, den Strom zu messen, der in einer Signalleitung fließt, indem dieses Signal über die Frontplattenanschlüsse „Current“ geführt wird.

- Wählen Sie das entsprechende Signal.

- Geben Sie unter „Error Active Time“ die gewünschte Dauer der Messung ein.



- Zur Ausführung des Fehlers klicken Sie **Activate Error**.
- Das gewählte Signal wird jetzt über die Anschlüsse „Current“ auf der Frontplatte des ES4440.1/.2 Compact Failure Simulation Module geführt, an denen dann der über diesen Kanal fließende Strom gemessen werden kann.

Reinigen der Relaiskontakte

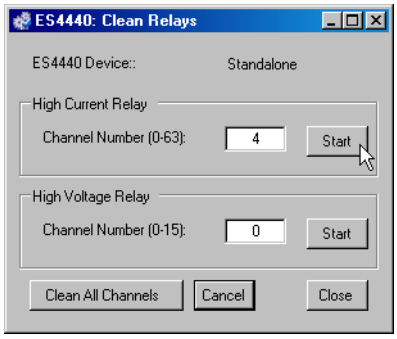
Zur Reinigung oxidierter Relaiskontakte gehen Sie wie folgt vor:

- Wählen Sie **Tools** → **ES4440 System Configuration**.

  Das Dialogfenster „ES4440 System Configuration“ wird geöffnet.
- Wählen Sie im Feld „Devices found“ die Hardware, auf die sich die nachfolgende Aktion beziehen soll.
- Im Feld „Device Tests“ klicken Sie **Clean Up Relays**.

  Es erscheint eine Warnmeldung, dass Sie vor Durchführung der Reinigung alle Steckanschlüsse von Typ „LOAD“ und „ECU“ entfernen sollten.

- Im folgenden Fenster können Sie nun Relais einzelner Kanäle auswählen oder auch alle reinigen



.

Hinweis

Da bei der Reinigung jedes Relais zehnmal geschaltet wird, kann eine Reinigung aller Relais einige Minuten in Abspruch nehmen!

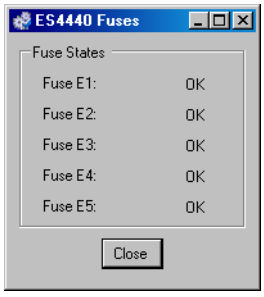
Sicherungen testen

Zu Testen der fünf Sicherungen der Fehler-Rails gehen Sie wie folgt vor:

- Wählen Sie **Tools** → **ES4440 System Configuration**.

  Das Dialogfenster „ES4440 System Configuration“ wird geöffnet.
- Wählen Sie im Feld „Devices found“ die Hardware, auf die sich die nachfolgende Aktion beziehen soll.
- Im Feld „Device Tests“ klicken Sie **Check Fuses**.

  Der Sicherungstest wird duchgeführt und das Ergebnis angezeigt.



Hinweis

Die Position der Sicherungen im Gerät und deren Spezifikation ist im Handbuch zum ES4440.1/.2 Compact Failure Simulation Module beschrieben.

Selbsttest durchführen

Bei einem Selbsttest werden PLD, der CAN-Controller und das EEPROM auf ihre Integrität überprüft. Um einen Selbsttest durchzuführen gehen Sie wie folgt vor:

- Wählen Sie **Tools** → **ES4440 System Configuration**.

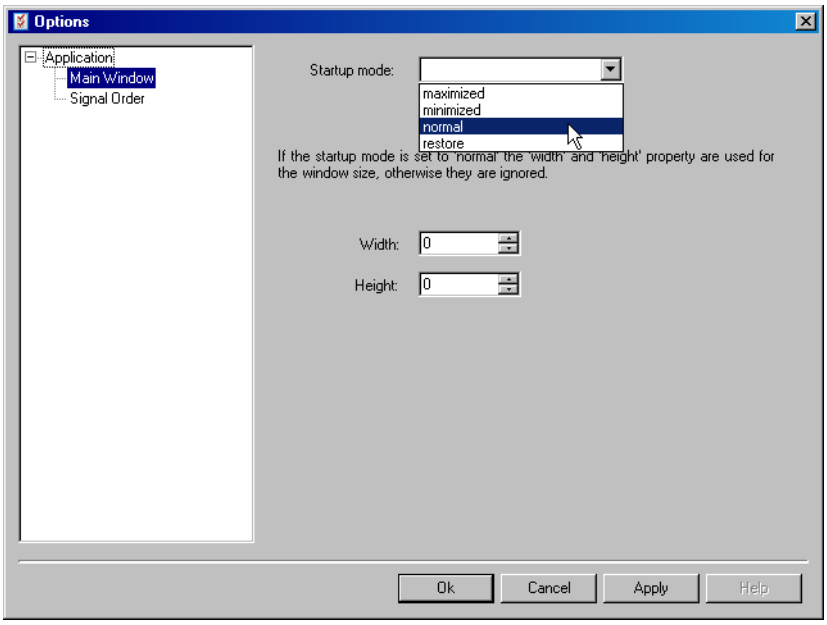
  Das Dialogfenster „ES4440 System Configuration“ wird geöffnet.
- Wählen Sie im Feld „Devices found“ die Hardware, auf die sich die nachfolgende Aktion beziehen soll.
- Im Feld „Device Tests“ klicken Sie **Self Test**.

  Der Selbsttest wird durchgeführt und das Ergebnis im Log-Fenster angezeigt.

Darstellungsoptionen festlegen

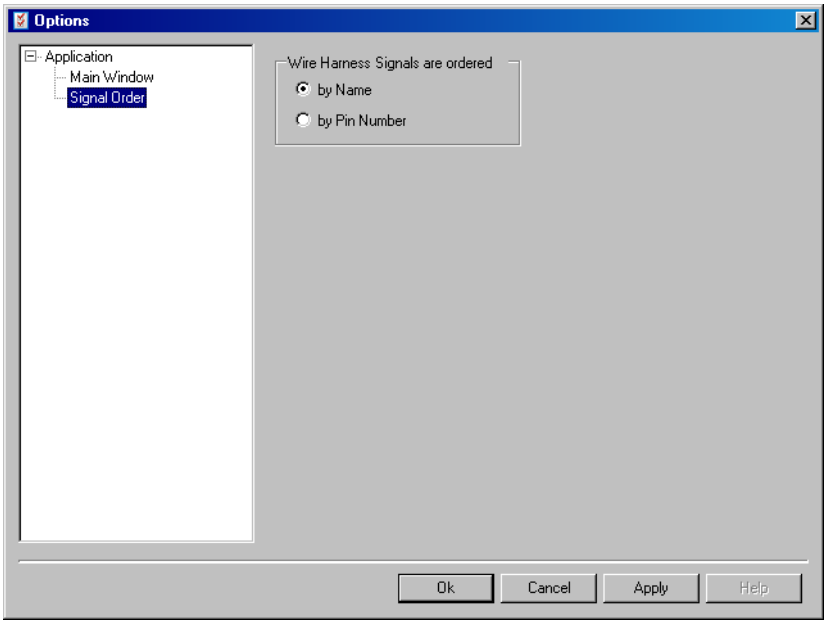
Sie können die Darstellung der Bedienoberfläche von LABCAR-PINCONTROL V2.2 beim Programmstart als auch die Sortierung der Signale im Fenster „Signals“ beeinflussen.

- Wählen Sie **Tools** → **Options...**.
- Im linken Teil des Fensters „Options“ wählen Sie „Main window“.
- Wählen Sie unter „Startup Mode“ die gewünschte Darstellung des Hauptfensters.



- Wenn Sie die Option „normal“ wählen, können Sie bei „Width“ und „Height“ die Breite und die Höhe des Fensters festlegen.

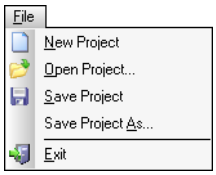
- Mit der Option „Signal Order“ können Sie festlegen, ob die Signale (im Fenster „Signals“) nach ihrem Namen oder nach „Pin Number“ geordnet werden.



2.4 Das Hauptmenü

In diesem Abschnitt finden Sie eine Beschreibung des Hauptmenüs von PINCONTROL.

Das Menü „File“



- **File → New Project**

  Erstellt ein neues Projekt

- **File → Open Project...**

  Öffnet ein zuvor gespeichertes Projekt

- **File → Save Project**

  Speichert das aktuell geladene Projekt. Neben Konfigurationsdaten werden insbesondere die Daten des Failure Sets (Pin Name, Pin Number und ECU Name) gespeichert.

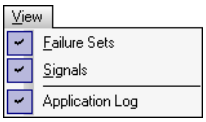
- **File → Save Project As...**

  Speichert das Projekt unter einem anderen Namen

- **File → Exit**

  Beendet LABCAR-PINCONTROL V2.2

Das Menü „View“



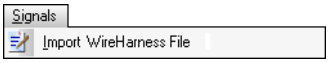
- **View → Failure Sets**

  Anzeigen/Verbergen des Bereiches, in dem die Failure Sets des Projektes dargestellt werden
- **View → Signals**

  Anzeigen/Verbergen des Bereiches, in dem die Signale des ins Projekt importierten Wire Harness Files dargestellt werden.
- **View → Application Log**

  Anzeigen/Verbergen des Log-Fensters

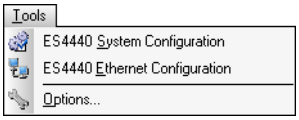
Das Menü „Signals“



- **Signals → Import WireHarness File**

  Ermöglicht die Auswahl eines Wire Harness Files

Das Menü „Tools“



- **Tools → ES4440 System Configuration**

  Öffnet das Dialogfenster, in dem die Systemeinstellungen vorgenommen werden
- **Tools → ES4440 Ethernet Configuration**

  Öffnet das Dialogfenster zur Konfiguration der Ethernetschnittstelle
- **Tools → Options...**

  Öffnet das Dialogfenster zur Definition von Darstellungsoptionen

Das Menü „Help“

- **Help → About**

  Öffnet ein Fenster mit Versionsinformationen
- **Help → Support**

  Öffnet ein Fenster mit ETAS Kontaktadressen

3 API-Dokumentation

Diese Kapitel enthält Informationen zum automatisierten Betrieb des ES4440.1/.2 Compact Failure Simulation Module mit dem COM-Controller von LABCAR-PINCONTROL V2.2 oder über CAN.

Im Einzelnen enthält dieses Kapitel folgende Informationen:

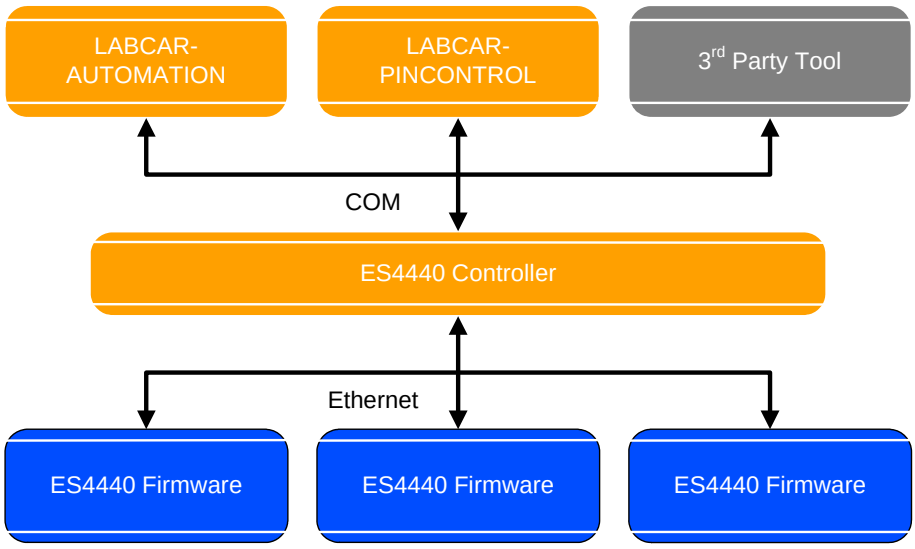
3.1 Einführung

Das ES4440.1/.2 Compact Failure Simulation Module kann sowohl über Ethernet als auch über CAN gesteuert werden. Das Ethernet-Protokoll wird von dem mitgelieferten COM-Controller unterstützt. Das CAN-Protokoll ist dazu gedacht, um ein Echtzeit-Simulationstarget über eine CAN-Karte direkt mit einer oder mehreren ES4440.1/.2 zu verbinden.

Diese API-Beschreibung beinhaltet die Methodenaufrufe des COM-Controllers und die Datenstruktur der CAN-Botschaften.

3.1.1 Aufgaben des COM-Controllers

Die folgende Abbildung zeigt die Arbeitsweise des COM-Controllers.



Eingangsgrößen des COM-Controllers sind der Steuergerätename und Steuergeräteanschluss. Mit Hilfe des Wire Harness Files weist der COM-Controller diesen Größen eine ES4440.1/.2 und die Nummer des jeweiligen Anschlusspins zu.

Desweiteren sorgt der COM-Controller für die Übersetzung der COM-Methoden auf das Ethernet-Protokoll. Bei Master/Slave-Konfiguration übernimmt der COM-Controller zudem die Verteilung der Befehle auf die richtigen ES4440-Module.

3.1.2 Einsatz der CAN-API

Die CAN-Botschaften werden direkt zur jeweiligen ES4440.1/.2 geschickt. Da die ES4440.1/.2-Module nicht miteinander kommunizieren können, muss der Benutzer Sorge tragen für

- das Mapping vom Steuergerätename und -pins zu den ES4440-Modulen und
- die Verteilung der Befehle an die ES4440-Module im Master/Slave-Betrieb.

3.1.3 Dateninhalte der COM- und der CAN-API

Die Dateninhalte der COM-API und der CAN-API sind bis auf eine Ausnahme gleich. Die Ausnahme besteht darin, dass der COM-Controller als Inputdaten Steuergerätename und Pinnamen erwartet und die CAN-API Pin Nummern der ES4440-Module.

3.2 Konfigurationen und Reihenfolge der CAN-Botschaften

In diesem Abschnitt wird die Verteilung der CAN-Botschaften an die jeweils verwendeten ES4440-Module beschrieben. Im Folgenden werden alle Fehlerfälle aufgeführt und gezeigt, in welcher Reihenfolge die CAN-Botschaften an die verschiedenen ES4440-Module geschickt werden müssen.

Generell gelten dabei folgende Regeln:

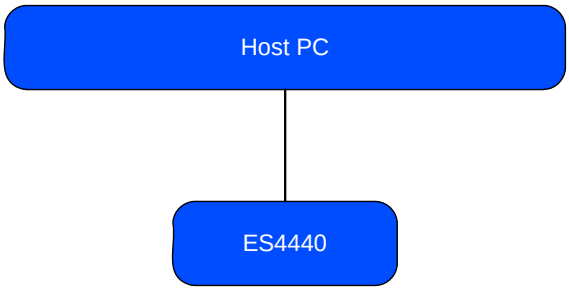
- Die Aktivierung der Relaisfehler geschieht immer durch das Master-Modul. Damit wird gewährleistet, dass der Fehler von allen ES4440-Modulen gleichzeitig geschaltet wird.
- Für Fehler, die mit MOSFETs realisiert werden, wird die Aktivierung an dem Modul vorgenommen, an dem auch der Fehler konfiguriert wurde.
- Sobald an einem ES4440-Modul ein Fehler konfiguriert wurde, benötigt dieses auch einen Reset-Befehl, damit der Fehler wieder aufgehoben wird.
- Pin-to-Pin-Fehler stellen eine gewisse Ausnahme dar - diese werden gesondert behandelt.

Hinweis

Die dargestellte Szenarien mit Master/Slave-Konfiguration sind beispielhaft - wenn eine Befehlsausführung auf „Slave 1" dargestellt wird, könnte dies auch der Master oder ein anderer Slave sein!

3.2.1 Einfachfehler in Stand-Alone-Anwendungen

Die in diesem Abschnitt beschriebenen Programmabläufe gelten für Einfachfehler, die auf einem ES4440.1/.2 Compact Failure Simulation Module ausgeführt werden.



Programmablauf für Fehler, die mit Relais erzeugt werden

1. **Fehlerkonfiguration**

   Hochstromfehler:

   - `Open_Load`

     oder

   - `ShortCut_xUBATTy_20A`

     oder

   - `Pin2PinFirstChWithoutLoad` / `Pin2PinSecondChWithoutLoad`

   Hochspannungsfehler:

   - `Open_Load_400V`

     oder

   - `ShortCut_xUBATTy_400V`

     oder

   - `Pin_2_Pin_400V`

2. **Fehleraktivierung**

   - `Activate_relay`

3. **Zurücksetzen des Fehlers**

   - `Reset_all_errors`

Programmablauf für Fehler, die mit MOSFETs realisiert werden

1. **Fehlerkonfiguration**

   - `Open_Load_realtime`

     oder

   - `ShortCut_xUBATTy_20A_realtime`

     oder

   - `Pin2PinFirstChRelatimeWithLoad`

     oder

- `Pin2PinSecondChRealtimeWithLoad`

  oder

- `RInline_realtime`

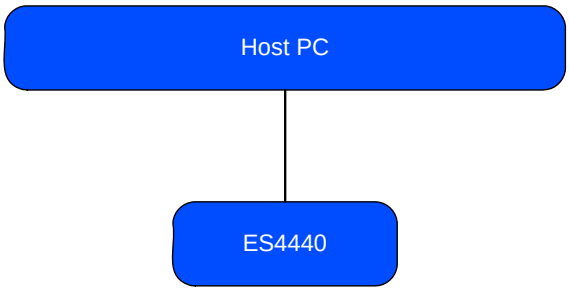
  oder

- `Pullup_Pulldown_xUBATTy_20A_realtime`

2. **Fehleraktivierung**
   - `Activate_realtime_switch`
3. **Zurücksetzen des Fehlers**
   - `Reset_all_errors`

3.2.2 Mehrfachfehler in Stand-Alone-Anwendungen

Die in diesem Abschnitt beschriebenen Programmabläufe gelten für Mehrfachfehler, die auf einem ES4440.1/.2 Compact Failure Simulation Module ausgeführt werden.



Programmablauf für Fehler, die mit Relais erzeugt werden

1. **Fehlerkonfiguration**

   (max. 10 Befehle werden der Reihe nach an die ES4440 verschickt)

   - `Open_Load`

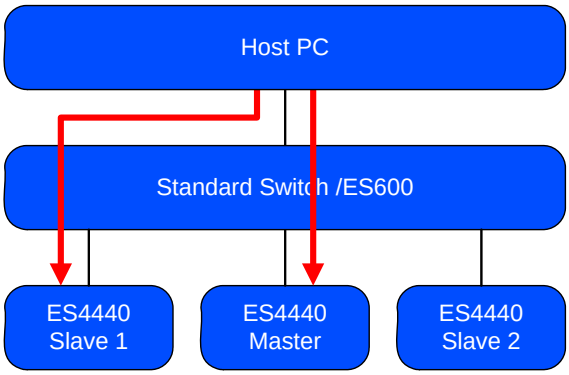
     und/oder

   - `ShortCut_xUBATTy_20A`
2. **Fehleraktivierung**
   - `Activate_relay`
3. **Zurücksetzen der Fehler**
   - `Reset_all_errors`

3.2.3 Einfachfehler in Master/Slave Anwendungen

Die in diesem Abschnitt beschriebenen Programmabläufe gelten für Einfachfehler, die auf einem Master/Slave-System ausgeführt werden.

Programmablauf für Fehler, die mit Relais erzeugt werden



1. **Fehlerkonfiguration (auf Slave 1)**

   Max. 10 Hochstromfehler:

   – `Open_Load`

   und/oder

   – `ShortCut_xUBATTy_20A`

2. **Fehleraktivierung (auf Master)**

   – `Activate_relay`

3. **Zurücksetzen der Fehler (auf Slave 1) ***

   – `Reset_all_errors`

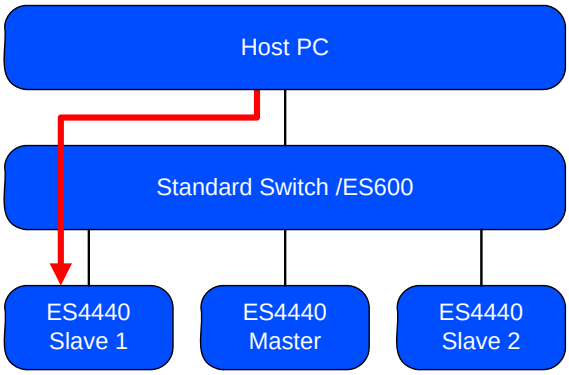
4. **Zurücksetzen der Fehler (auf Master) ****

   – `Reset_all_errors`

* Der Befehl „Zurücksetzen des Fehlers“ auf einem Slave wird zunächst nur gespeichert.

** Der Befehl „Zurücksetzen des Fehlers“ auf dem Master bewirkt ein synchrones Zurücksetzen der Fehler auf dem Master und allen Slaves, die diesen Befehl zuvor gespeichert haben.

Programmablauf für Fehler, die mit MOSFETs erzeugt werden



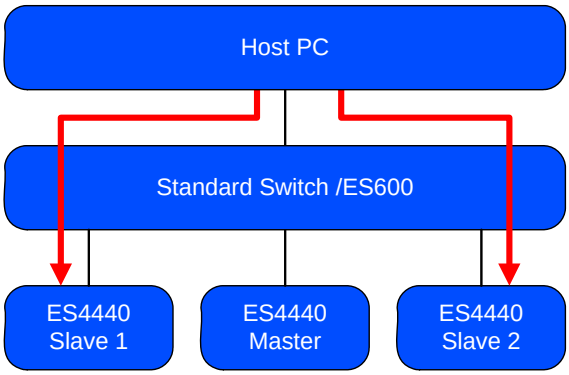
1. **Fehlerkonfiguration**
   - `Open_Load_realtime`

     oder
   - `ShortCut_xUBATTy_20A_realtime`

     oder
   - `RInline_realtime`

     oder
   - `Pullup_Pulldown_xUBATTy_20A_realtime`
2. **Fehleraktivierung**
   - `Activate_realtime_switch`
3. **Zurücksetzen des Fehlers**
   - `Reset_all_errors`

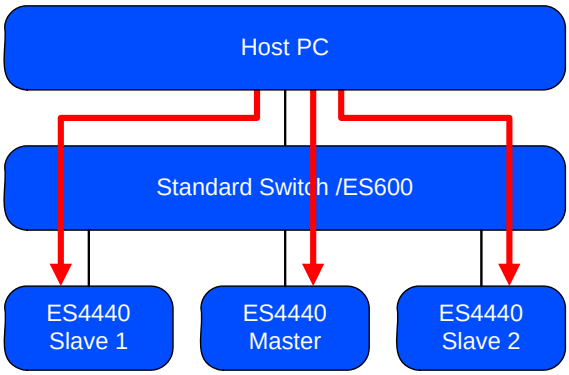
Sonderfall: Pin-to-Pin-Fehler mit Last



1. **Fehlerkonfiguration für ersten Pin (auf Slave1)**
   - `Pin2PinFirstChRelatimeWithLoad`
2. **Fehlerkonfiguration für zweiten Pin (auf Slave 2)**
   - `Pin2PinSecondChRealtimeWithLoad`

3. **Fehleraktivierung (auf Slave1)**
   - `Activate_realtime_switch`
4. **Zurücksetzen des Fehlers (auf Slave1)**
   - `Reset_all_errors`
5. **Zurücksetzen des Fehlers (auf Slave 2)**
   - `Reset_all_errors`

Sonderfall: Pin-to-Pin ohne Last



1. **Fehlerkonfiguration für ersten Pin (auf Slave 1)**
   - `Pin2PinFirstChWithoutLoad`
2. **Fehlerkonfiguration für zweiten Pin (auf Slave 2)**
   - `Pin2PinSecondChWithoutLoad`
3. **Fehleraktivierung (auf Master)**
   - `Activate_relay`
4. **Zurücksetzen des Fehlers (auf Slave 1)**
   - `Reset_all_errors`
5. **Zurücksetzen des Fehlers (auf Slave 2)**
   - `Reset_all_errors`
6. **Zurücksetzen des Fehlers (auf Master)**
   - `Reset_all_errors`

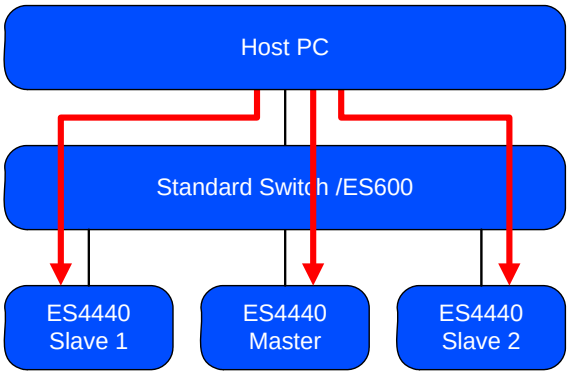
Hinweis

Dieser Fehler wird mit Relais geschaltet - es befindet sich keine Sicherung im Fehlerpfad zwischen Pin 1 und Pin 2!

3.2.4 Mehrfachfehler in Master/Slave-Anwendungen

Die in diesem Abschnitt beschriebenen Programmabläufe gelten für Mehrfachfehler, die auf einem Master/Slave-System ausgeführt werden.

Programmablauf für Fehler, die mit Relais erzeugt werden



1. **Fehlerkonfiguration (auf Slave 1)**
   Max. 10 Hochstromfehler:
   - `Open_Load`
     und/oder
   - `ShortCut_xUBATTy_20A`
2. **Fehlerkonfiguration (auf Slave 2)**
   Max. 10 Hochstromfehler:
   - `Open_Load`
     und/oder
   - `ShortCut_xUBATTy_20A`
3. **Fehleraktivierung (auf Master)**
   - `Activate_relay`
4. **Zurücksetzen des Fehlers (auf Slave1)**
   - `Reset_all_errors`
5. **Zurücksetzen des Fehlers (auf Slave 2)**
   - `Reset_all_errors`
6. **Zurücksetzen des Fehlers (auf Master)**
   - `Reset_all_errors`

3.3 Initialisierung

3.3.1 Initialisierung des COM-Controllers

Die Initialisierungsprozedur des COM-Controllers sieht wie folgt aus:

```
Dim returnValue As Integer
Dim WireharnessPath As String
'access CommCtrlAccess
Set ctrla = CreateObject
   ("ETAS.PTS.PINCONTROLV2.CommCtrl.CommCtrlAccess")
'using CommCtrlAccess class you can try to access the
singleton instance
Set ctrl = ctrla.CommCtrlInstance
returnValue = ctrl.InitErrorSimulationUsingFile(Wire-
harnessPath)
```

Die Exit-Procedure hat folgende Form

```
'free CommCtrl Singleton instance
    Call ctrla.FreeCommCtrlInstance
```

Hinweis

Der COM-Controller kann jeweils nur von einer Anwendung verwendet werden!

3.3.2 Initialisierung für CAN

Für CAN ist keine Initialisierung notwendig.

3.4 Detaillierte Beschreibung der Befehle

In diesem Abschnitt finden Sie die vollständige Syntaxbeschreibung aller Befehle.

Im Einzelnen sind dies:

3.4.1 Allgemeine Befehlsstruktur

Im Folgenden finden Sie eine Beschreibung der allgemeinen Struktur der CAN-Send- und CAN-Receive-Botschaften.

CAN-Send-Botschaft

***COM_command_name*(*arguments*)**		
1. Byte	Command ID	*Wert*
2. Byte	Parameter 0	*Wert*
3. Byte	Parameter 1	*Wert*
4. Byte	Parameter 2	*Wert*
5. Byte	Parameter 3	*Wert*
6. Byte	Parameter 4	*Wert*
7. Byte	Parameter 5	*Wert*
8. Byte	Parameter 6	*Wert*

Tab. 3-1 Struktur einer Send-Botschaft

CAN-Receive-Botschaft

Antwort		
1. Byte	Command ID	*Wert*
2. Byte	Parameter 0	*Wert*
3. Byte	Parameter 1	*Wert*
4. Byte	Parameter 2	*Wert*
5. Byte	Parameter 3	*Wert*
6. Byte	Parameter 4	*Wert*
7. Byte	Parameter 5	*Wert*
8. Byte	Parameter 6	*Wert*

Tab. 3-2 Struktur der Receive-Botschaft

Hinweis

Um zu vermeiden, dass der Empfangspuffer überläuft, dürfen CAN-Botschaften zur Steuerung der ES4440.1/.2 nur im Single-Shot-Betrieb gesendet werden!

3.4.2 Definitionen für alle Funktionen

Die Informationen in diesem Abschnitt sind für alle Befehle gültig.

Kanalnummern

Bei den Hochstromkanälen werden die Kanäle von 0 - 63 abgezählt, bei den Hochspannungskanälen von 0 - 15.

Definition for Parameter 1 der CAN-Send-Botschaft:

Der Parameter 1 (3.Byte) hat bei allen Funktionen denselben Aufbau:

Bit 0	**load**	load = 1: Fehlersimulation mit Last load = 0: Fehlersimulation ohne Last
Bit 1 **Bit 2** **Bit 3**	**xUBatty**	xUBatty = 0: +UBatt_A xUBatty = 1: -UBatt_A xUBatty = 2: +UBatt_B xUBatty = 3: -UBatt_B xUBatty = 4:+UBatt_C xUBatty = 5: -UBatt_C
Bit 4	**current**	current = 0: Strommessung aus current = 1: Strommessung an
Bit 5	**set**	set = 1: Fehler gesetzt set = 0: Fehler zurückgesetzt
Bit 6	**duration_flag**	duration_flag = 0: Fehler liegt unendlich (d.h. bis zum Reset) an. duration_flag = 1: Fehlerdauer wird von „duration_time“ festgelegt.
Bit 7	nicht verwendet	

3.4.3 Fehlercodes

Fehlercodes werden im Byte 8 der Receive-Message übermittelt – beim API-Befehl `GetLastErrorSimulationAnswer()` werden Fehlercodes ebenfalls in Byte 8 übergeben.

Ergebnis	Bedeutung
0x0	Befehl OK
0x21	Falscher Parameter für Slave-Adresse (> 16)
0x22	Unbekannter Befehl
0x23	Falscher Datentyp beim Schreiben des Flash
0x24	Falscher Parameter beim LED-Test
0x25	Zu große Zahl in IP-Adresse (muss < 256 sein)
0x26	Falscher Parameter für CAN Baudrate
0x27	Falscher Parameter für CAN-Terminierung
0x28	Falscher Parameter für CAN-Identifiertyp
0x29	Parameter für Kanal der Kaskade zu groß (muss < 15 sein)
0x2a	Falscher Parameter für Widerstandskaskade
0x2b	nicht verwendet
0x2c	Falsche Adresse bei Flash-Lesezugriff (gültige Werte < 513)
0x2d	Falsche Datenlänge bei Flash-Lesezugriff (gültige Werte < 17)
0x2e	Falsche Adresse bei Flash-Schreibzugriff (gültige Werte < 513)
0x2f	Falsche Datenlänge bei Flash-Schreibzugriff (gültige Werte < 17)
0x30	PLD-Fehler
0x31	Fehler EEPROM-Prüfsumme
0x32	CAN-Controller lässt sich nicht ansprechen
0x41	Simulationsbefehl gibt Plausibilitätsfehler zurück
0x42	Referenzrelais wurde nicht erkannt
0x43	„duration_time“ hat nicht den Wert „0xffff“, obwohl Fehler unbefristet anliegen soll.
0x44	Simulationsbefehl wurde nicht erkannt
0x45	Interner Hardwarefehler PLD: Befehl konnte nicht richtig geschaltet werden
0x46	Wert von „duration time“ ausserhalb des gültigen Bereichs (1 bis 5000 oder 0xFFFF)
0x47	Letzte Fehlersimulation ist noch aktiv, mit `Reset_all_errors` löschen
0x48	Max. Anzahl Relais erreicht
0x49	Fehler mit MultiErrorFlag
0x4a	Angegebene Kanalnummer außerhalb des gültigen Bereichs
0x4b	Frequenz oder Tastverhältnis außerhalb des gültigen Bereichs

Ergebnis	Bedeutung
0x4c	Systemtemperatur > 60 °C
0x4d	Temperatur der Widerstandskaskade > 60 °C
0x4e	Temperatur MOSFET > 60 °C
0x4f	Fühler für Systemtemperatur defekt
0x50	Fühler für Temperatur der Widerstandskaskade defekt
0x51	Fühler für Temperatur der MOSFETs defekt
0x52	Railspannung nicht korrekt (möglicher Kurzschluss)
0x53	Ungültiger Widerstandswert

3.4.4 Der IDN-Befehl

Dieser Befehl dient zur Identifikation der jeweiligen ES4440.1/.2.

Befehl für COM-Controller

```
int ctrl.CommandIDN(string deviceKey);
```

Mögliche Werte von „deviceKey“: Standalone, Master, Slave1, ..., Slave14

Rückgabewert vom COM-Controller

```
byte[] Answer =
          ctrl.GetLastSystemConfigurationAnswer();
```

CAN-Send-Botschaft

IDN-Befehl		
1. Byte	Command ID	0x0
2. Byte	Parameter 0	nicht verwendet
3. Byte	Parameter 1	nicht verwendet
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	nicht verwendet

CAN-Receive-Botschaft

Antwort		
1. Byte	Command ID	0x0
2. Byte	Parameter 0	Device Config Byte 1
3. Byte	Parameter 1	Device Config Byte 0
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	command result

Werte für „Device Config“:

- Standalone: 255
- Master: 0
- Slave 1 ... Slave 14: 1...14

3.4.5 Open_Load

Unterbricht eine Leitung zwischen Steuergerät und Last. Dieser Fehler wird mit Relais geschaltet - bis zu zehn Fehler können gleichzeitig geschaltet werden. Der Wert von „channels left“ in der Befehlsantwort gibt an, wieviele Kanäle noch für weitere Fehler zur Verfügung stehen.

Befehl für COM-Controller

```
int ctrl.OpenLoad(string ecu, string ecuPin,
                        int durationType, int set);
```

Rückgabewert vom COM-Controller

```
byte[] Answer = ctrl.GetLastErrorSimulationAnswer();
```

CAN-Send-Botschaft

Open_Load(channel_nr, duration_flag, set)		
1. Byte	Command ID	0x1
2. Byte	Parameter 0	ES4440 channel number
3. Byte	Parameter 1	set, duration_flag (siehe Seite 44)
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	nicht verwendet

CAN-Receive-Botschaft

Antwort		
1. Byte	Command ID	0x1
2. Byte	Parameter 0	ES4440 channel number
3. Byte	Parameter 1	channels left
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	command result

3.4.6 Open_Load_realtime

Unterbricht eine Leitung zwischen Steuergerät und Last. Dieser Fehler wird mit MOSFETs geschaltet und ist nur als Einfachfehler realisierbar.

Befehl für COM-Controller

```
int ctrl.OpenLoad_RealTime(string ecu,
                string ecuPin, int durationType);
```

Rückgabewert vom COM-Controller

```
byte[] Answer = ctrl.GetLastErrorSimulationAnswer();
```

Aufbau der CAN-Botschaft

Open_Load_realtime (channel_nr, duration_flag)		
1. Byte	Command ID	0x2
2. Byte	Parameter 0	ES4440 channel number
3. Byte	Parameter 1	duration_flag (siehe Seite 44)
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	nicht verwendet

CAN-Receive-Botschaft

Antwort		
1. Byte	Command ID	0x2
2. Byte	Parameter 0	ES4440 channel number
3. Byte	Parameter 1	nicht verwendet
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	command result

3.4.7 ShortCut_xUBATTy_20A

Erzeugt bei einem Hochstromkanal einen Kurzschluss einer Leitung gegen eine Batteriespannung. Dieser Fehler wird mit Relais geschaltet und ist mehrfach realisierbar.

Befehl für COM-Controller

```
int ctrl.Shortcut_xUBATTy_20A(string ecu,
            string ecuPin, int load, int xUBATTy,
                        int durationType, int set);
```

Rückgabewert vom COM-Controller

```
byte[] Answer = ctrl.GetLastErrorSimulationAnswer();
```

CAN-Send-Botschaft

ShortCut_xUBATTy_20A (channel_nr, load, xUBATTy, duration_flag, set)		
1. Byte	Command ID	0x3
2. Byte	Parameter 0	ES4440 channel number
3. Byte	Parameter 1	load, xUBatty, set, duration_flag (siehe Seite 44)
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	nicht verwendet

CAN-Receive-Botschaft

Antwort		
1. Byte	Command ID	0x3
2. Byte	Parameter 0	ES4440 channel number
3. Byte	Parameter 1	channels left
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	command result

3.4.8 ShortCut_xUBATTy_20A_realtime

Erzeugt bei einem Hochstromkanal einen Kurzschluss einer Leitung gegen eine Batteriespannung. Dieser Fehler wird mit MOSFETs geschaltet und ist daher nur einfach realisierbar.

Befehl für COM-Controller

```
int ctrl.Shortcut_xUBATTy_20A_RealTime
        (string ecu, string ecuPin, int load,
                  int xUBATTy, int durationType);
```

Rückgabewert vom COM-Controller

```
byte[] Answer = ctrl.GetLastErrorSimulationAnswer();
```

CAN-Send-Botschaft

ShortCut_xUBATTy_20A_realtime (channel_nr, load, xUBATTy, duration_flag)		
1. Byte	Command ID	0x4
2. Byte	Parameter 0	ES4440 channel number
3. Byte	Parameter 1	load, xUBatty, duration_flag (siehe Seite 44)
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	nicht verwendet

CAN-Receive-Botschaft

Antwort		
1. Byte	Command ID	0x4
2. Byte	Parameter 0	ES4440 channel number
3. Byte	Parameter 1	nicht verwendet
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	command result

3.4.9 Pin2PinFirstChWithoutLoad

Definiert die erste Leitung für einen Kurzschluss zwischen zwei Leitungen - die zweite Leitung wird mit dem Befehl „Pin2PinSecondChannelWithoutLoad“ definiert (siehe Seite 53).

Hinweis

Dieser Fehler wird mit Relais geschaltet und ohne Last und ohne Widerstand realisiert. Es befindet sich keine Sicherung zwischen Pin 1 und Pin 2!

Befehl für COM-Controller

```
int ctrl.Pin2PinChannel1_WithoutLoad(string ecu,
                 string ecuPin, int durationType);
```

Rückgabewert vom COM-Controller

```
byte[] Answer = ctrl.GetLastErrorSimulationAnswer();
```

CAN-Send-Botschaft

Pin2PinFirstChWithoutLoad (channel_nr1, duration_flag)		
1. Byte	Command ID	0x5
2. Byte	Parameter 0	ES4440 channel number 1
3. Byte	Parameter 1	duration_flag (siehe Seite 44)
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	nicht verwendet

CAN-Receive-Botschaft

Antwort		
1. Byte	Command ID	0x5
2. Byte	Parameter 0	ES4440 channel number 1
3. Byte	Parameter 1	nicht verwendet
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	command result

3.4.10 Pin2PinSecondChannelWithoutLoad

Definiert die zweite Leitung für einen Kurzschluss zwischen zwei Leitungen.

Hinweis

Dieser Fehler wird mit Relais geschaltet und ohne Last und ohne Widerstand realisiert. Es befindet sich keine Sicherung zwischen Pin 1 und Pin 2!

Befehl für COM-Controller

```
int ctrl.Pin2PinChannel2_WithoutLoad(string ecu,
                  string ecuPin, int durationType);
```

Rückgabewert vom COM-Controller

```
byte[] Answer = ctrl.GetLastErrorSimulationAnswer();
```

CAN-Send-Botschaft

Pin2PinSecondChannelWithoutLoad (channel_nr1, duration_flag)		
1. Byte	Command ID	0x6
2. Byte	Parameter 0	ES4440 channel number 2
3. Byte	Parameter 1	duration_flag (siehe Seite 44)
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	nicht verwendet

CAN-Receive-Botschaft

Antwort		
1. Byte	Command ID	0x6
2. Byte	Parameter 0	ES4440 channel number 2
3. Byte	Parameter 1	nicht verwendet
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	command result

3.4.11 Pin2PinFirstChRealtimeWithLoad

Definiert die erste Leitung für einen Kurzschluss zwischen zwei Leitungen. Dieser Fehler ermöglicht die Simulation mit Last und endlichem Widerstand zwischen den beiden Leitungen.

Dieser Fehler wird mit MOSFETs geschaltet.

Befehl für COM-Controller

```
int ctrl.Pin2PinChannel1_RealTime_WithLoad
     (string ecu, string ecuPin, int resistor,
                  int current, int durationType);
```

Rückgabewert vom COM-Controller

```
byte[] Answer = ctrl.GetLastErrorSimulationAnswer();
```

CAN-Send-Botschaft

Pin2PinFirstChRealtimeWithLoad (channel_nr1, duration_flag)		
1. Byte	Command ID	0x7
2. Byte	Parameter 0	ES4440 channel number 1
3. Byte	Parameter 1	current, duration_flag (siehe Seite 44)
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	Widerstand Byte 0
6. Byte	Parameter 4	Widerstand Byte 1
7. Byte	Parameter 5	Widerstand Byte 2
8. Byte	Parameter 6	Widerstand Byte 3

CAN-Receive-Botschaft

Antwort		
1. Byte	Command ID	0x7
2. Byte	Parameter 0	ES4440 channel number 1
3. Byte	Parameter 1	nicht verwendet
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	command result

3.4.12 Pin2PinSecondChRealtimeWithLoad

Definiert die zweite Leitung für einen Kurzschluss zwischen zwei Leitungen. Dieser Fehler ermöglicht die Simulation mit Last und endlichem Widerstand zwischen den beiden Leitungen.

Dieser Fehler wird mit MOSFETs geschaltet.

Befehl für COM-Controller

```
int ctrl.Pin2PinChannel2_RealTime_WithLoad
     (string ecu, string ecuPin, int durationType);
```

Rückgabewert vom COM-Controller

```
byte[] Answer = ctrl.GetLastErrorSimulationAnswer();
```

CAN-Send-Botschaft

Pin2PinSecondChRealtimeWithLoad (channel_nr2, duration_flag)		
1. Byte	Command ID	0x8
2. Byte	Parameter 0	ES4440 channel number 2
3. Byte	Parameter 1	duration_flag (siehe Seite 44)
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	nicht verwendet

CAN-Receive-Botschaft

Antwort		
1. Byte	Command ID	0x8
2. Byte	Parameter 0	ES4440 channel number 2
3. Byte	Parameter 1	nicht verwendet
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	nicht verwendet

3.4.13 RInline_realtime

Schaltet einen Leitungswiderstand mit MOSFETs.

Befehl für COM-Controller

```
int ctrl.RInline_RealTime(string ecu1,
            string ecuPin1, int resistor,
                   int current, int durationType);
```

Rückgabewert vom COM-Controller

```
byte[] Answer = ctrl.GetLastErrorSimulationAnswer();
```

CAN-Send-Botschaft

RInline_realtime (channel_nr1, resistor, duration_flag, current)		
1. Byte	Command ID	0x9
2. Byte	Parameter 0	ES4440 channel number
3. Byte	Parameter 1	current, duration_flag (siehe Seite 44)
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	Widerstand Byte 0 (LSB) *
6. Byte	Parameter 4	Widerstand Byte 1 *
7. Byte	Parameter 5	Widerstand Byte 2 *
8. Byte	Parameter 6	Widerstand Byte 3 (MSB) *

* Die Byte-Order entspricht dem Motorola-Format:

```
Byte n    (LSB)
Byte n+1
Byte n+2
Byte n+3  (MSB)
```

Ein Widerstandswert von 0x1234 hat also folgende Byte-Order:

```
Byte n    0x34 (LSB)
Byte n+1  0x12
Byte n+2
Byte n+3       (MSB)
```

CAN-Receive-Botschaft

Antwort		
1. Byte	Command ID	0x9
2. Byte	Parameter 0	ES4440 channel number
3. Byte	Parameter 1	nicht verwendet
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	command result

3.4.14 Pullup_Pulldown_xUBATTy_20A_realtime

Legt einen Kanal über einen Widerstand an eine Batteriespannung (Pull-Up/Pull-Down).

Befehl für COM-Controller

```
int ctrl.PullUp_PullDown_20A_RealTime
   (string ecu, string ecuPin, int load, int xUBATTy,
     int resistor, int current, int durationType);
```

Rückgabewert vom COM-Controller

```
byte[] Answer = ctrl.GetLastErrorSimulationAnswer();
```

CAN-Send-Botschaft

Pullup_Pulldown_xUBATTy_20A_realtime (channel_nr, load, xUBATTy, resistor, duration_flag, current)		
1. Byte	Command ID	0xB
2. Byte	Parameter 0	ES4440 channel number
3. Byte	Parameter 1	xUBatty, current, duration_flag, load (siehe Seite 44)
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	Widerstand Byte 0 (LSB) *
6. Byte	Parameter 4	Widerstand Byte 1 *
7. Byte	Parameter 5	Widerstand Byte 2 *
8. Byte	Parameter 6	Widerstand Byte 3 (MSB) *

* Zur Darstellung der Bytes siehe „RInline_realtime“ auf Seite 56.

CAN-Receive-Botschaft

Antwort		
1. Byte	Command ID	0xB
2. Byte	Parameter 0	ES4440 channel number
3. Byte	Parameter 1	nicht verwendet
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	command result

3.4.15 Open_Load_400V

Unterbricht eine Hochspannungsleitung zwischen Steuergerät und Last.
Dieser Fehler wird mit Relais geschaltet und ist nur als Einfachfehler möglich.

Befehl für COM-Controller

```
int ctrl.OpenLoad_400V(string ecu, string ecuPin,
                       int durationType, int set);
```

Rückgabewert vom COM-Controller

```
byte[] Answer = ctrl.GetLastErrorSimulationAnswer();
```

CAN-Send-Botschaft

Open_Load_400V (channel_nr, set, duration_flag)		
1. Byte	Command ID	0xD
2. Byte	Parameter 0	ES4440 channel number
3. Byte	Parameter 1	duration_flag, set (siehe Seite 44)
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	nicht verwendet

CAN-Receive-Botschaft

Antwort		
1. Byte	Command ID	0xD
2. Byte	Parameter 0	ES4440 channel number
3. Byte	Parameter 1	nicht verwendet
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	command result

3.4.16 ShortCut_xUBATTy_400V

Mit diesem Befehl schliessen Sie einen Hochspannungskanal mit einer Batteriespannung kurz.Dieser Fehler wird mit Relais geschaltet und ist nur als Einfachfehler und ohne Last möglich.

Befehl für COM-Controller

```
int ctrl.ShortCut_xUBATTy_400V
        (string ecu, string ecuPin, int xUBATTy,
                        int durationType, int set);
```

Rückgabewert vom COM-Controller

```
byte[] Answer = ctrl.GetLastErrorSimulationAnswer();
```

CAN-Send-Botschaft

ShortCut_xUBATTy_400V (channel_nr, xUBATTy, set, duration_flag)		
1. Byte	Command ID	0xE
2. Byte	Parameter 0	ES4440 channel number
3. Byte	Parameter 1	xUBATTy, duration_flag, set (siehe Seite 44)
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	nicht verwendet

CAN-Receive-Botschaft

Antwort		
1. Byte	Command ID	0xE
2. Byte	Parameter 0	ES4440 channel number
3. Byte	Parameter 1	nicht verwendet
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	command result

3.4.17 ShortCut_xUBATTy_400V_Ex

Mit diesem Befehl schliessen Sie einen Hochspannungskanal mit einer Batteriespannung kurz. Dieser Fehler wird mit Relais geschaltet und ist nur als Einfachfehler möglich. Er kann mit oder ohne Last geschaltet werden.

Befehl für COM-Controller

```
int ctrl.ShortCut_xUBATTy_400V_Ex
        (string ecu, string ecuPin, int xUBATTy,
                  int load, int durationType, int set);
```

Rückgabewert vom COM-Controller

```
byte[] Answer = ctrl.GetLastErrorSimulationAnswer();
```

CAN-Send-Botschaft

ShortCut_xUBATTy_400V (channel_nr, load, xUBATTy, set, duration_flag)		
1. Byte	Command ID	0xE
2. Byte	Parameter 0	ES4440 channel number
3. Byte	Parameter 1	load, xUBATTy, duration_flag, set (siehe Seite 44)
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	nicht verwendet

CAN-Receive-Botschaft

Antwort		
1. Byte	Command ID	0xE
2. Byte	Parameter 0	ES4440 channel number
3. Byte	Parameter 1	nicht verwendet
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	command result

3.4.18 Pin_2_Pin_400V

Erzeugt einen Kurzschluss zwischen zwei Hochspannungskanälen (ohne Last).

Dieser Fehler wird mit Relais geschaltet und ist nur als Einfachfehler und ohne Last möglich.

Befehl für COM-Controller

```
int ctrl.Pin2Pin_400V
     (string ecu1, string ecuPin1, string ecu2,
               string ecuPin2, int durationType);
```

Rückgabewert vom COM-Controller

```
byte[] Answer = ctrl.GetLastErrorSimulationAnswer();
```

CAN-Send-Botschaft

Pin_2_Pin_400V (channel_nr1, channel_nr2, duration_flag)		
1. Byte	Command ID	0xF
2. Byte	Parameter 0	ES4440 channel number 1
3. Byte	Parameter 1	duration_flag (siehe Seite 44)
4. Byte	Parameter 2	ES4440 channel number 2
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	nicht verwendet

CAN-Receive-Botschaft

Antwort		
1. Byte	Command ID	0xF
2. Byte	Parameter 0	ES4440 channel number 1
3. Byte	Parameter 1	nicht verwendet
4. Byte	Parameter 2	ES4440 channel number 2
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	command result

3.4.19 Pin_2_Pin_400V_Ex

Erzeugt einen Kurzschluss zwischen zwei Hochspannungskanälen (ohne Last). Dieser Fehler wird mit Relais geschaltet und ist nur als Einfachfehler möglich. Er kann mit oder ohne Last geschaltet werden.

Befehl für COM-Controller

```
int ctrl.Pin2Pin_400V_Ex
     (string ecu1, string ecuPin1, string ecu2,
          string ecuPin2, int load, int durationType);
```

Rückgabewert vom COM-Controller

```
byte[] Answer = ctrl.GetLastErrorSimulationAnswer();
```

CAN-Send-Botschaft

Pin_2_Pin_400V_Ex (channel_nr1, channel_nr2, load, duration_flag)		
1. Byte	Command ID	0xF
2. Byte	Parameter 0	ES4440 channel number 1
3. Byte	Parameter 1	load, duration_flag (siehe Seite 44)
4. Byte	Parameter 2	ES4440 channel number 2
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	nicht verwendet

CAN-Receive-Botschaft

Antwort		
1. Byte	Command ID	0xF
2. Byte	Parameter 0	ES4440 channel number 1
3. Byte	Parameter 1	nicht verwendet
4. Byte	Parameter 2	ES4440 channel number 2
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	command result

3.4.20 Reset_all_errors

Mit diesem Befehl werden alle Fehler auf einem ES4440.1/.2 Compact Failure Simulation Module zurückgesetzt.

Befehl für COM-Controller

```
int ctrl.ResetAllErrors();
```

Rückgabewert vom COM-Controller

```
byte[] Answer = ctrl.GetLastErrorSimulationAnswer();
```

CAN-Send-Botschaft

Reset_all_errors ()		
1. Byte	Command ID	0x10
2. Byte	Parameter 0	nicht verwendet
3. Byte	Parameter 1	nicht verwendet
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	nicht verwendet

CAN-Receive-Botschaft

Antwort		
1. Byte	Command ID	0x10
2. Byte	Parameter 0	nicht verwendet
3. Byte	Parameter 1	nicht verwendet
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	command result

3.4.21 Activate_relay

Mit diesem Befehl wird das Relais für eine bestimmte Zeit geschlossen.

Wenn im vorhergehenen Fehlerbefehl „duration_flag“ (Bit 6 im 3. Byte) gesetzt ist (= 1), kann „duration_time“ zwischen 20 ms und 5 s gewählt werden. Ist „duration_flag“ = 0, muss „duration_time“ = -1 oder = 65535 (0xFFFF) gewählt werden.

Hinweis

Beim Setzen von Mehrfachfehlern müssen die Parameter „duration_flag“ aller Fehler denselben Wert besitzen!

In der Befehlsantwort werden die am Referenzrelais gemessenen Schaltzeiten übermittelt.

Befehl für COM-Controller

```
int ctrl.ActivateRelay(int durationTime);
```

Rückgabewert vom COM-Controller

```
byte[] Answer = ctrl.GetLastErrorSimulationAnswer();
```

CAN-Send-Botschaft

Activate_relay (duration_ time)		
1. Byte	Command ID	0x12
2. Byte	Parameter 0	nicht verwendet
3. Byte	Parameter 1	duration_time in ms (Byte 0)
4. Byte	Parameter 2	duration_time in ms (Byte 1)
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	nicht verwendet

Der Parameter „duration_time“ kann in Schritten von 20 ms spezifiziert werden - der kleinstmögliche Wert beträgt 20 ms, der größtmögliche 5000 ms.

Endlosfehler werden mit dem Wert 0xFFFF erzeugt.

Der Parameter „channel_type“ kann folgende Werte annehmen:

- channel_type = 0: Hochstromkanal
- channel_type = 1: Hochspannungskanal

CAN-Receive-Botschaft

Antwort		
1. Byte	Command ID	0x12
2. Byte	Parameter 0	delay_time0 NO-Kontakt* 20 A schliesst (in 100 µs Schritten)
3. Byte	Parameter 1	delay_time1 NO-Kontakt* 20 A schliesst (in 100 µs Schritten)
4. Byte	Parameter 2	delay_time0 NC-Kontakt** 20 A öffnet (in 100 µs Schritten)
5. Byte	Parameter 3	delay_time1 NC-Kontakt** 20 A öffnet (in 100 µs Schritten)
6. Byte	Parameter 4	delay_time0 NC-Kontakt** 400 V schliesst (in 100 µs Schritten)
7. Byte	Parameter 5	NC-Kontakt** 400 V schliesst (in 100 µs Schritten)
8. Byte	Parameter 6	command result

* NO = normally open = Schließer
** NC= normally closed = Öffner

3.4.22 Activate_realtime_switch

Mit diesem Befehl wird ein mit MOSFETs geschalteter Fehler für eine bestimmte Zeit geschlossen.

Wenn im vorhergehenen Fehlerbefehl „duration_flag“ (Bit 6 im 3. Byte) gesetzt ist (= 1), kann „duration_time“ zwischen 1 ms und 5 s gewählt werden. Ist „duration_flag“ = 0, muss „duration_time“ = -1 oder = 65535 (0xFFFF) gewählt werden.

Befehl für COM-Controller

```
int ctrl.ActivateRealTimeSwitch(int mode,
     int durationTime, int dutyCycle, int frequency);
```

Rückgabewert vom COM-Controller

```
byte[] Answer = ctrl.GetLastErrorSimulationAnswer();
```

CAN-Send-Botschaft

Activate_realtime_switch (mode, duration_time, dutycycle, frequency)		
1. Byte	Command ID	0x13
2. Byte	Parameter 0	mode
3. Byte	Parameter 1	duration_time (Byte 0)
4. Byte	Parameter 2	duration_time (Byte 1)
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	Tastverhältnis bei Wackelkontakt
7. Byte	Parameter 5	Frequenz bei Wackelkontakt (Byte 0)
8. Byte	Parameter 6	Frequenz bei Wackelkontakt (Byte 1)

Der Parameter „duration_time“ kann in Schritten von 1 ms spezifiziert werden - der kleinstmögliche Wert beträgt 1 ms, der grösstmögliche 5000 ms. Endlosfehler werden mit dem Wert 0xFFFF erzeugt.

Zur Deaktivierung der Wackelkontaktsimulation müssen die Parameter 4, 5 und 6 auf den Wert 0xFF gesetzt werden.

Der Parameter „mode“ kann folgende Werte annehmen:

- mode = 0: Statischer Fehler, dessen Dauer durch „duration_time“ definiert ist.
- mode = 1: Wackelkontaktsimulation

  Für Tastverhältnis und Frequenz gelten folgende Einschränkungen: 1 - 99% bei 3 Hz bis 100 Hz, 50% bei 2 Hz

CAN-Receive-Botschaft

Antwort		
1. Byte	Command ID	0x13
2. Byte	Parameter 0	mode
3. Byte	Parameter 1	duration_time (Byte 0)
4. Byte	Parameter 2	duration_time (Byte 1)
5. Byte	Parameter 3	duration_time (Byte 2)
6. Byte	Parameter 4	duration_time (Byte 3)
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	command result

3.4.23 Test fuses

Ermittelt den Zustand der Sicherungen für die Fehler-Rails.

Befehl für COM-Controller

```
int ctrl.TestFuses(string deviceKey ,
                                out int[] fuses);
```

Der Zustand der Sicherungen wird im Array `fuses` gespeichert: In `fuses[n]` ist der Zustand der Sicherung n+1 abgelegt (n=0 ... 4). Ein Wert von 1 bedeutet „Sicherung OK“, ein Wert von 0 bedeutet „Sicherung defekt“.

Mögliche Werte für „deviceKey“: Standalone, Master, Slave1, …, Slave14

Rückgabewert vom COM-Controller

```
byte[] Answer = ctrl.GetLastErrorSimulationAnswer();
```

CAN-Send-Botschaft

1. Byte	Command ID	0x14
2. Byte	Parameter 0	nicht verwendet
3. Byte	Parameter 1	nicht verwendet
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	nicht verwendet

CAN-Receive-Botschaft

Antwort		
1. Byte	Command ID	0x14
2. Byte	Parameter 0	Fuse_status
3. Byte	Parameter 1	nicht verwendet
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	command result

Der Zustand der Sicherung wird über das jeweilige Bit ermittelt (0 = Sicherung defekt, 1 = Sicherung OK):

MSB							LSB
Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
-	-	-	E5	E1	E3	E4	E2

3.4.24 CurrentMeasurement

Führt einen Kanal zur Messung des Stromes über die „Current"-Anschlüsse auf der Frontplatte des ES4440.1/.2 Compact Failure Simulation Module.

Befehl für COM-Controller

```
int ctrl.CurrentMeasurement
          (string ecu, string ecuPin, byte set);
```

Rückgabewert vom COM-Controller

```
byte[] Answer = ctrl.GetLastErrorSimulationAnswer();
```

CAN-Send-Botschaft

CurrentMeasurement (channel_nr)		
1. Byte	Command ID	0x15
2. Byte	Parameter 0	channel_nr
3. Byte	Parameter 1	nicht verwendet
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	nicht verwendet

CAN-Receive-Botschaft

Antwort		
1. Byte	Command ID	0x15
2. Byte	Parameter 0	channel_nr
3. Byte	Parameter 1	nicht verwendet
4. Byte	Parameter 2	nicht verwendet
5. Byte	Parameter 3	nicht verwendet
6. Byte	Parameter 4	nicht verwendet
7. Byte	Parameter 5	nicht verwendet
8. Byte	Parameter 6	command result

4 ETAS Kontaktinformation

ETAS Hauptsitz

ETAS GmbH

Borsigstraße 14 Telefon: +49 711 3423-0

70469 Stuttgart Telefax: +49 711 3423-2106

Deutschland WWW: www.etas.com

ETAS Regionalgesellschaften und Technischer Support

Informationen zu Ihrem lokalen Vertrieb und zu Ihrem lokalen Technischen Support bzw. den Produkt-Hotlines finden Sie im Internet:

ETAS Regionalgesellschaften WWW: www.etas.com/de/contact.php

ETAS Technischer Support WWW: www.etas.com/de/hotlines.php

Index