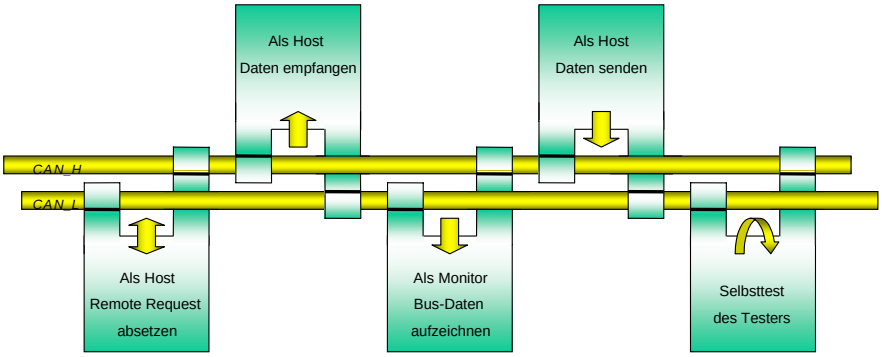
CAN - Bus - Tester

USB - HID



Version 7.0

1. Der CAN-Bus-Tester und seine Installation

Der CAN-Bus-Tester besteht aus einem Hardwareteil und der zugehörigen Software, die mit allen Windows-Betriebsystemen lauffähig ist, da die USB-HID-Schnittstelle benutzt wird. Es wird also kein Treiber benötigt.

Der CAN-Tester kann folgende Aufgaben übernehmen:

Kommunikationsobjekt Senden	→	Tester arbeitet als Host
Kommunikationsobjekt Empfangen	→	Tester empfängt Daten vom Bus
Kommunikationsobjekt Anfordern	→	Tester fordert ein ″Remote Request″ an
Monitoring	→	alle Bus-Daten werden vom Tester protokolliert
Selbsttest	→	Hardware-Test und Funktionsdemo

Nach Installation der Software steht unter dem CAN-Tester-Verzeichnis auch das Sprach-, Doku- und Records-Verzeichnis zur Verfügung. Diese sind notwendig, damit der Tester an die jeweilige Landessprache bzw. Hilfe angepasst werden kann.

Beim Anschließen des Testers ruft das Windows-Betriebssystem den eigenen USB-Treiber auf (HidUsb) und konfiguriert die Schnittstelle. Außerdem wird damit in der Registry unter HKEY_LOCALMACHINE\Enum\USB (bei Windows 98) bzw. HKEY_LOCALMACHINE\SYSTEM\CurrentControlSet\Enum\USB (ab Windows 2000) der Treiber eingetragen.

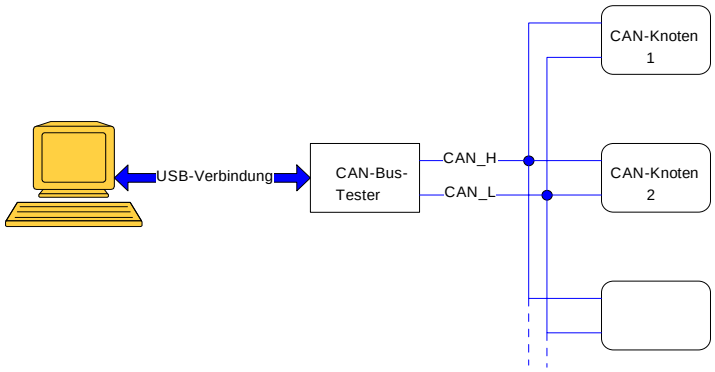
Die Versorgungsspannung für den Tester wird durch die USB-Verbindung bereitgestellt.

Vor dem Start der CAN-Tester-Software sollte der PC auf seine Einstellungen bezüglich ″Bildschirmschoner″ und ″Power down″ geprüft werden. Diese Einstellungen sollten abgeschaltet sein, da sonst Laufzeitfehler entstehen können. Bei unnatürlicher Farbdarstellung sollten Sie die Farbpalette Ihres Monitors auf ″True Color″ und die Auflösung auf 1024x768 Bildpunkte einstellen.

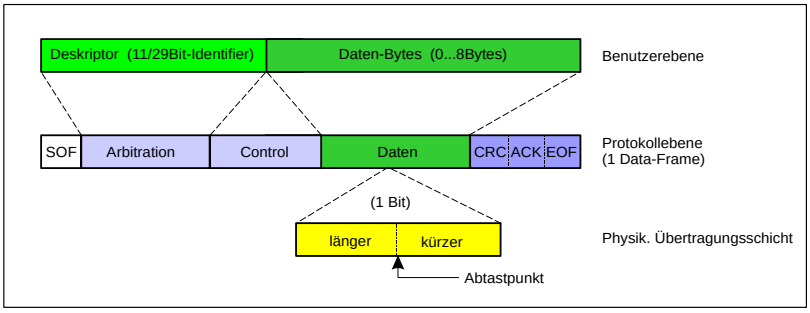
2. Allgemeines

Die Funktion des CAN-Bus-Testers basiert auf dem Prinzip des CAN-Protokolls, bei dem vorausgesetzt wird, dass die Anzahl der auszutauschenden Nachrichten im Netz begrenzt ist und somit jedes Objekt über eine Nummer – dem Identifier **"ID"** – eindeutig identifiziert werden kann. Er kann als Adresse des Busteilnehmers verstanden werden. Möglich sind 2032 unterschiedliche ID's.
Bei CAN gibt es also keine Teilnehmer-Adressierung, sondern jeder Busteilnehmer bestimmt über einen Filtermechanismus im Buskontroller, welches Objekt er übernehmen will. Auf den Bus aufgeschaltete Nachrichten können von jedem Busteilnehmer empfangen werden, die den entsprechenden Objektidentifier als Empfangsobjekt eingetragen haben.

Die Verbindung zwischen den Teilnehmern wird mit 2 Leitungen – dem high-level-Signal CAN_H und low-level-Signal CAN_L – hergestellt.



Das CAN-Protokoll definiert, wie eine Nachricht (Message) bitseriell über den CAN-Datenbus übertragen wird. Entsprechend dem ISO/OSI –7-Schichtenmodell zur Kommunikation werden von der Benutzerebene bis zur physikalischen Übertragungsschicht Zusatzinformationen (Protokollelemente) per Hardware dem CAN-Protokoll hinzugefügt. Dieses Prinzip erlaubt den sendenden und den empfangenden CAN-Controllern, gemeinsam die korrekte und fehlerfreie Nachrichtenübertragung zu sichern.



Auf der Benutzerebene besteht eine CAN-Nachricht aus dem Deskriptor (CAN 2.0B = 29 Bit) und aus 0...8 Datenbytes. Dabei definiert der Deskriptor, von welcher Art die folgenden Datenbytes sind (inhaltsbezogene Adressierung). Mit Hilfe dieses Deskriptors können dann

alle empfangenden CAN-Controller – jeder für sich – entscheiden, ob sie diese Daten speichern wollen oder nicht.

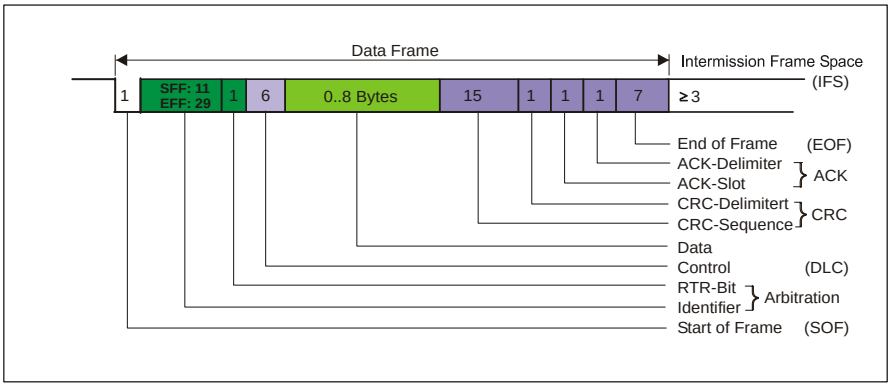
Die Protokollebene umfasst neben Arbitration und Control weitere Elemente, wie ″Start of Frame″, ″CRC-Feld″ und ″End of Frame″. Dabei dient die im CRC-Feld (Cyclic Redundancy Code) enthaltene Prüfsumme der Erkennung von Fehlern, womit eine Restfehlerwahrscheinlichkeit von $3x10^{-5}$ erreicht wird. Zusammen mit den weiteren Mechanismen zur Erkennung von Fehlern wird damit eine gesamte Restfehlerwahrscheinlichkeit von

Nachrichtenfehlerrate x $4,6 \cdot 10^{-11}$

erreicht. Diese Werte sind besser als bei ähnlichen Protokollen.

Ebenfalls auf der Protokollebene wird die ***bitweise Arbitration*** durchgeführt. Hierbei bewirbt sich ein sendebereites Objekt über das Arbitrierungsfeld (ID mit RTR-Bit zur Kennzeichnung einer Remote-Anforderung) um die Sendeberechtigung. Über die ID ist dem Objekt eine Priorität für den Buszugriff zugeordnet. Beim gleichzeitigem Sendeversuch mehrerer CAN-Controller lässt sich damit sicherstellen, dass die Bewerber mit höherwertigem Identifier (niedrigerer Priorität) ihre Bewerbung aufgeben, sobald der von ihnen aufgeschaltete Pegel durch einen anderen Teilnehmer ″überstimmt″ wurde. Diese Methode garantiert die Übertragung der Botschaft mit höchster Priorität an alle Stationen.

Die physikalische Übertragungsschicht erlaubt die Einstellung einer für die Applikation geeigneten Datenrate von 1kBit/s bis 1Mbit/s. Der CAN-Bus ist ein synchroner Bus, was insbesondere aus der bitweisen Arbitration resultiert. Um die Synchronisation zu erreichen, passen die CAN-Controller ihre individuell getakteten Bitströme einander an, verlängern oder verkürzen einzelne Bits und erreichen damit automatisch die Synchronisation.

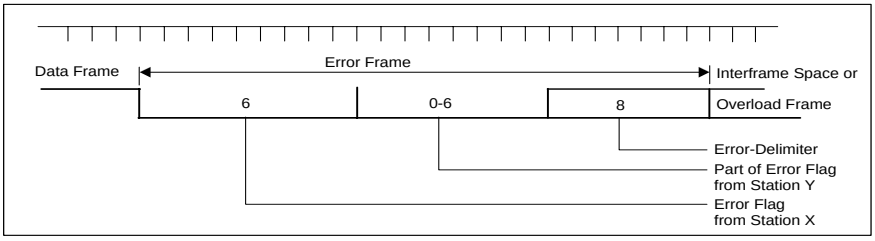


Dieses Bild zeigt die typische Struktur eines Datentelegramms, den Data Frame mit seinen 7 Feldern. Er enthält die eigentliche Information, also Identifier (Name der Botschaft) und die zugehörigen Daten. Zusätzlich sind im Bitstrom Informationen zur Resynchronisation der Stationen enthalten, die jeweils mit eigenen Taktgeneratoren frei laufen. Die Start- und Ende-Information kennzeichnet die Dauer eines Telegramms. Die einzelnen Felder des Data Frame sind:

Bezeichner	Eigenschaften
Startbit = SOF (1 Bit = low)	Es kennzeichnet den Telegrammanfang. Nach einer Idle-Zeit auf dem Bus dient die neg. Flanke des Startbits zur Phasensynchronisation der einzelnen, sonst frei laufenden Stationen.
Identifier = ID (11/29 Bit)	Er kennzeichnet Namen und Priorität der Botschaft; je kleiner der Wert, desto höher die Priorität. Im Standard Frame Format (SFF) hat er 11 Bit und im Extended Frame Format (EFF) 29 Bit.
RTR-Bit (1 Bit)	Es kennzeichnet ein Telegramm, das keine Daten enthält, sondern nur den Sender auffordert, ein Telegramm mit aktuellen Daten abzusenden.
Controll (6 Bit)	Es enthält die Länge der nachfolgenden Daten
Data (0...8 Byte)	Enthält die Daten des Telegramms
CRC-Field (16 Bit)	Enthält den Fehlercode aller vorangegangenen Stellen; dient nur zur Erkennung.
ACK-Field (2 Bit)	Alle Knoten, die eine Botschaft über ihren Bus-Handler als korrekt empfangen haben, quittieren dies durch Senden eines Low-Signals im ACK-Slot. Fehlt das Quittiersignal, dann erkennt der Sender dies als Fehler.
EOF (7 Bit= high)	Kennzeichnet das Ende eines Telegramms
Interframe space (≥3 Bit)	Kennzeichnet den Zeitraum für das Übertragen einer korrekt empfangenen Botschaft im Botschaftsempfangspuffer.

Der Fehlerzustand eines CAN-Controllers wird durch Empfangs- und Sendefehlerzähler bestimmt. Diese Zähler werden bei einem erfolgreichen Transfer dekrementiert und im Fehlerfall inkrementiert. Je nach Zählerstand nimmt der Controller einen unterschiedlichen Fehlerstatus ein (Fehler aktiv, Fehler passiv, Abschaltung vom Bus). Ist der Fehlerzähler größer 255, dann wird der Controller vom Bus getrennt.

Im Fehlerfall werden weitere Protokollelemente (z.B. Error Frames) erzeugt, so dass Fehler automatisch – und für die Benutzerebene transparent – behoben werden können. Die grundlegende Struktur eines Fehlertelegramms ist im folgenden Bild zu sehen.

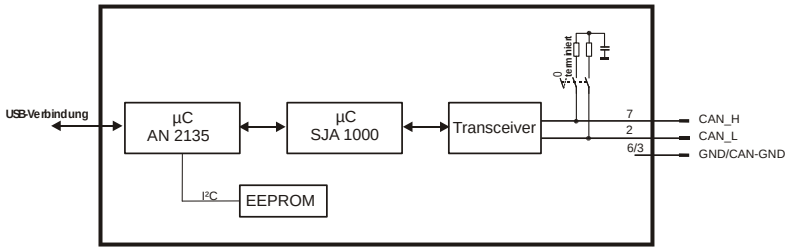


Sobald ein Busteilnehmer einen Fehler in einer Botschaft entdeckt, sendet er ein solches Fehlertelegramm. Es ist so angelegt, dass mit einer Folge von 6 dominanten Bits jeder andere aktuelle Buswert überschrieben wird und durch die bewusste Verletzung der Stuff-

Bit-Regel die Erkennung des Fehlerrahmens durch jeden anderen Busteilnehmer möglich wird.
Als Folge eines Fehlerrahmens, der eigentlich nur 6 dominante Bitstellen aufweist, können andere Busteilnehmer eine Verletzung der Stuff-Regel entdecken und ihrerseits erst daraufhin einen Fehlerrahmen aussenden. Dieser Effekt führt zu einer möglichen Verlängerung der Sequenz bis zu 12 dominanten Bits. Der Error-Delimiter (Fehlerbegrenzer) besteht aus einer Folge von 8 rezessiven Bits und hat das gleiche Format, wie ein ″Overload Delimiter″ [1][3].

3. Hardware

Das Blockschaltbild des CAN-Bus-Testers - auf der Basis des Mikrokontrollers AN2135 und SJA1000 - ist in dem nachfolgenden Bild zu sehen. Dabei übernimmt der AN2135 die USB 1.2-Kommunikation und der SJA1000 die CAN 2.0-Anbindung. Die Datenrate der USB-Verbindung ist 12MBit/s (full speed mode) und beim Übertragungsverfahren wird der Bulk-Transfer benutzt, der beim Senden und Empfang jeweils 3 Byte transferiert. Das dafür nötige Controller-Programm (Firmware) wird beim Starten des Testers vom EEPROM in den Arbeitsspeicher des AN2135 geladen.



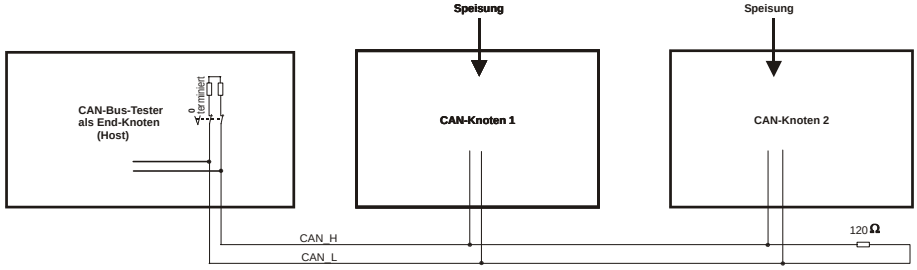
Die Spannungsversorgung wird durch die USB-Verbindung bereitgestellt.

Der SJA1000 kann mit dem CAN 2.0A- oder CAN 2.0B-Protokoll arbeiten und hat eine max. Übertragungsrate von 1MBit/s. Mit ihm ist es möglich, den Datentransfer sowohl im standardmäßigen 11-Bit-Mode (SFF) wie auch im erweiterten 29-Bit-Mode (EFF) durchzuführen. Philips bezeichnet ihn als *"**PeliCan-Mode**"*. Der CAN-Identifier kann in dem jeweiligen Mode folgende Werte annehmen:

- ID im Standard Frame Format (SFF): 0... 7FFh
- ID im Extended Frame Format (EFF): 0... 1FFFFFFFh.

Die Bus-Verbindung wird über einen 9pol. D-Sub-Stecker hergestellt, bei dem GND und CAN-GND an Pin 6 und 3 geführt sind. Das Signal CAN_H liegt an Pin 7 und CAN_L an Pin 2.

Zur universellen CAN-Bus-Anpassung ist der Tester mit einem Schiebeschalter ausgestattet, der es erlaubt, die Leitung nach dem ISO 11898 Standard abzuschließen (terminieren); d.h. den Tester als End-Knoten im Netz zu benutzen. In diesem Fall wird also das Bus-Ende mit einem gesplittetem 120Ω-Widerstand überbrückt. Der zusätzliche Kondensator bewirkt einen optimalen HF-Abschluß [2].



Um die EMV-Abstrahlung auf langen Leitungen zu minimieren, ist es außerdem möglich, mit verschiedenen Übertragungsraten zu arbeiten (siehe Optionen). Auf jeden Fall sollte bei hoher Baudrate und langen Leitungen geschirmte Bus-Kabel benutzt werden, die folgende Längen annehmen können [2]:

Baudrate [kBit/s]	Länge [m]
1000	30
500	100
100	500

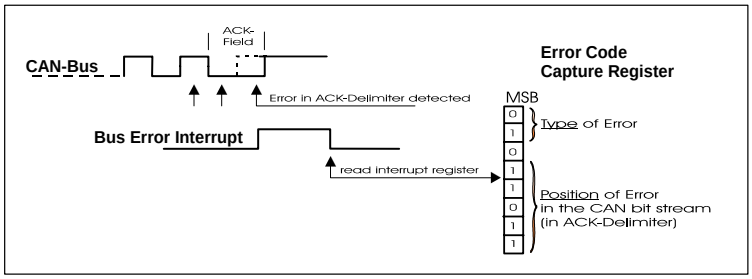
Letztendlich ist es durch den im CAN-Tester verwendeten Tranceiver möglich, bis zu 110 Knoten im Netz anzuschließen. Seine Ausgänge sind gegen Kurzschluß geschützt und stellen eine differentielle Verbindung zum Bus her [2]. Die Anschlüsse sind für Bus-Teilnehmer bis 12V Nennspannung ausgelegt. Bei 24V-Systemen müsste man den Transceiver austauschen.

Eine besonders hervorzuhebende Eigenschaft des µC SJA1000 und somit die des CAN-BUS-Testers ist, dass mit ihm gemäß den CAN 2.0B-Spezifikationen eine Vielzahl von Fehlerereignissen aufgezeigt und analysiert werden können. Die entsprechenden Register heißen:

- Error Code Capture Register (ECC)
- Arbitration Lost Capture Register (ALC)
- Fehlerzähler TX und RX.

Sobald der µC in einer Botschaft einen Fehler entdeckt, wird er im entsprechenden Register gespeichert und durch die zyklische Tester-Programmabfrage sichtbar gemacht.

Das **ECC**- Register informiert also über Typ und Position der angefallenen Fehler. Unterschieden wird nach vier Arten: Bit-, Form-, Stuff- und andere Fehler. Wie ein Fehler während einer Übertragung oder Empfang aufgespürt wird, ist in dem nachfolgenden Bild dargestellt.



Ergänzt wird die Diagnose mit dem **ALC**-Register, in dem die genaue Position eines Fehlers während der Arbitration bestimmt werden kann.

Die Fehlerzähler **TX** und **RX** zählen beim Senden und Empfangen und lassen damit Rückschlüsse auf die Übertragungsqualität zu. Der TX-Zähler erhöht sich um 8 bei jedem Übertragungsfehler und der RX-Zähler um 1 bei jedem empfangenen Fehler [3]. Wenn der Zähler 256 erreicht, ist die Teilnahme am CAN-Bus AUS (Bus-Off). Der Tester ist dann nicht mehr im CAN-Netz; er kann in diesem Fall aber mit Hilfe der ″Initialisierung″ in den aktiven Zustand zurückkehren.

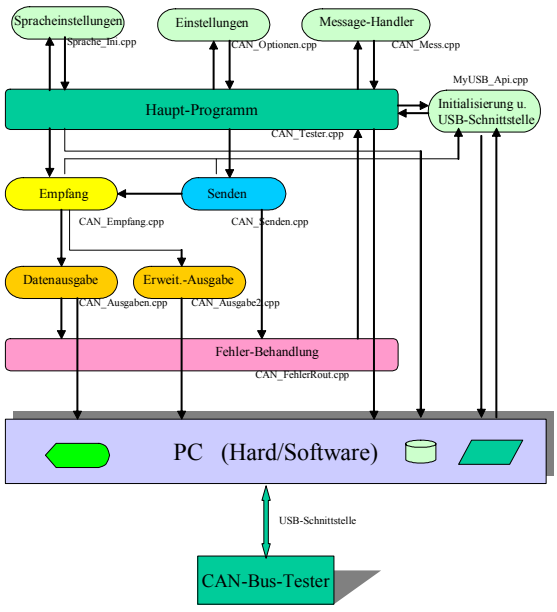
4. Software

Das Programm des CAN-Bus-Testers hat eine leicht zu handhabende Windows-Oberfläche, die als leistungsfähige 32-Bit-Software durchgehend in C++ programmiert ist. Mit den graphischen Objekten haben Sie sofort alles im Griff. So können Sie z.B. von Ihrer ganz speziellen Anwendung die Empfangs- oder Sende-Daten abspeichern, wenn der entsprechende Fensterteil fokussiert ist (durch weiße Schrift von Senden bzw. Empfangen erkenntlich). Das Einlesen bezieht sich nur auf die Sendedaten.

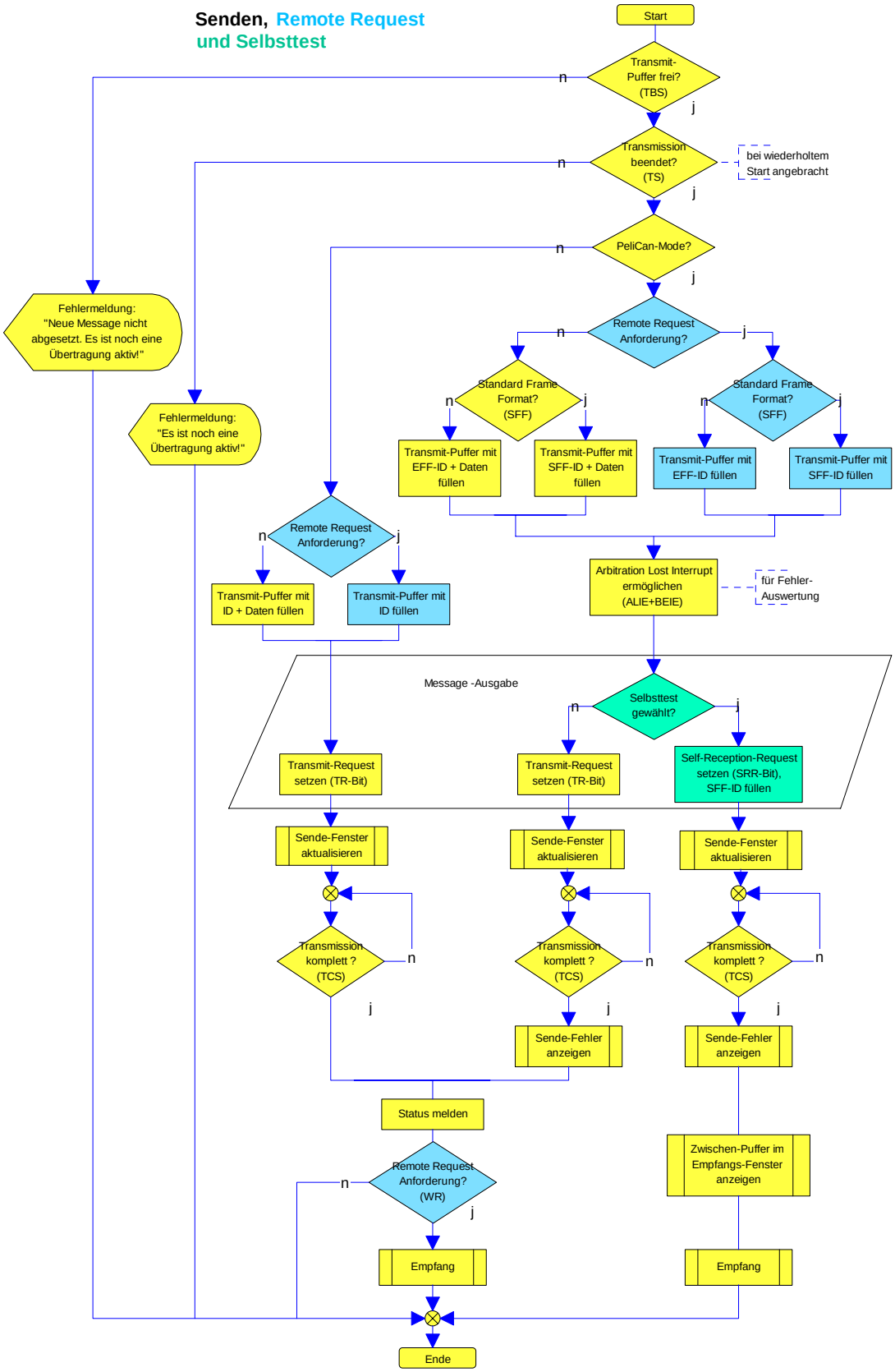
Bei der Erstellung der Software stand im Vordergrund, dass alle erdenklichen Testmöglichkeiten am CAN-Bus abgedeckt werden. Somit ist für jedes Modul ein Flussdiagramm vorhanden, in dem neben den Fehlermeldungen auch die einzelnen Testschritte nachvollzogen werden können. Damit ist es jederzeit möglich, Störungen bei einem Datentransfer – egal in welcher Richtung – genau zu lokalisieren.

Mit dem Toolbutton ″Sprache″ können Sie das Tester-Programm sogar an die englische, französische, italienische oder spanische Sprache anpassen, sofern der jeweilige Text-File im Verzeichnis ″Language″ vorhanden ist. Zur Zeit ist nur die deutsche und englische Version beigestellt. Mit Hilfe eines einfachen Editors können Sie jedoch durch Kopieren und Ersetzen des Textes nach dem Gleichheitszeichen, die anderen Sprach-Files (francais.lng, italiano.lng und espaniol.lng) ergänzen.

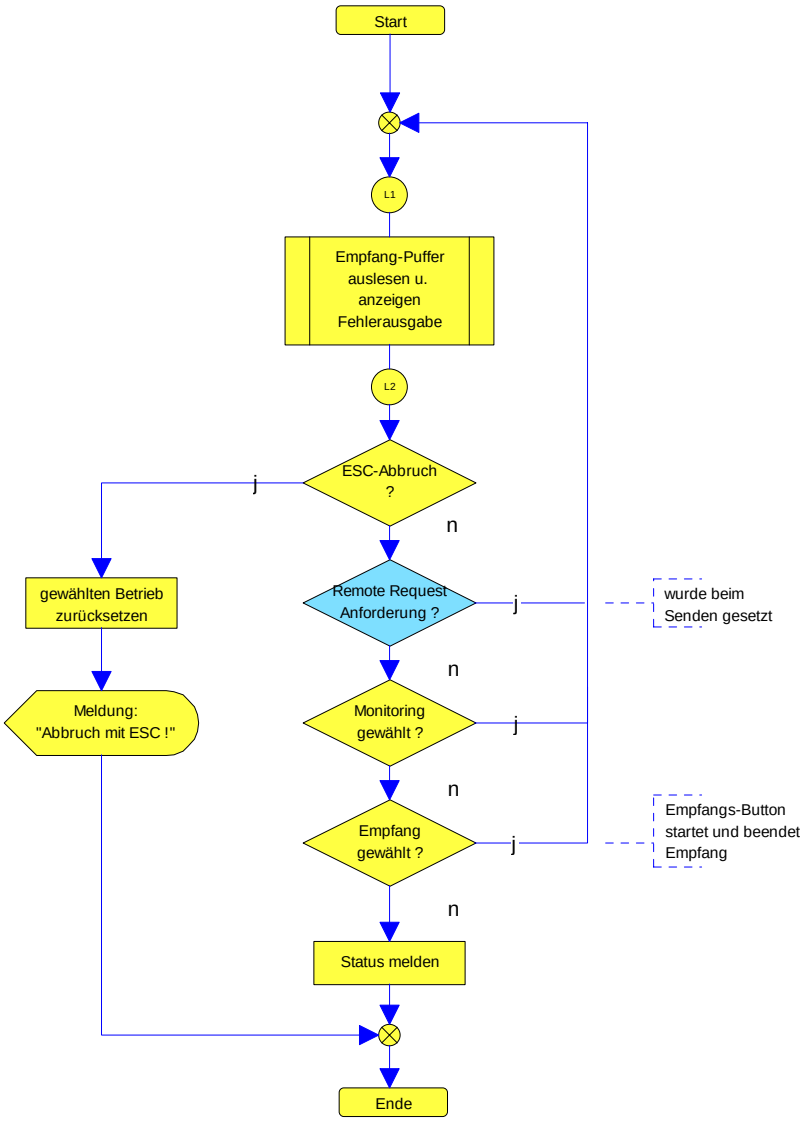
Modulstruktur des CAN-Tester 6.0 (USB)

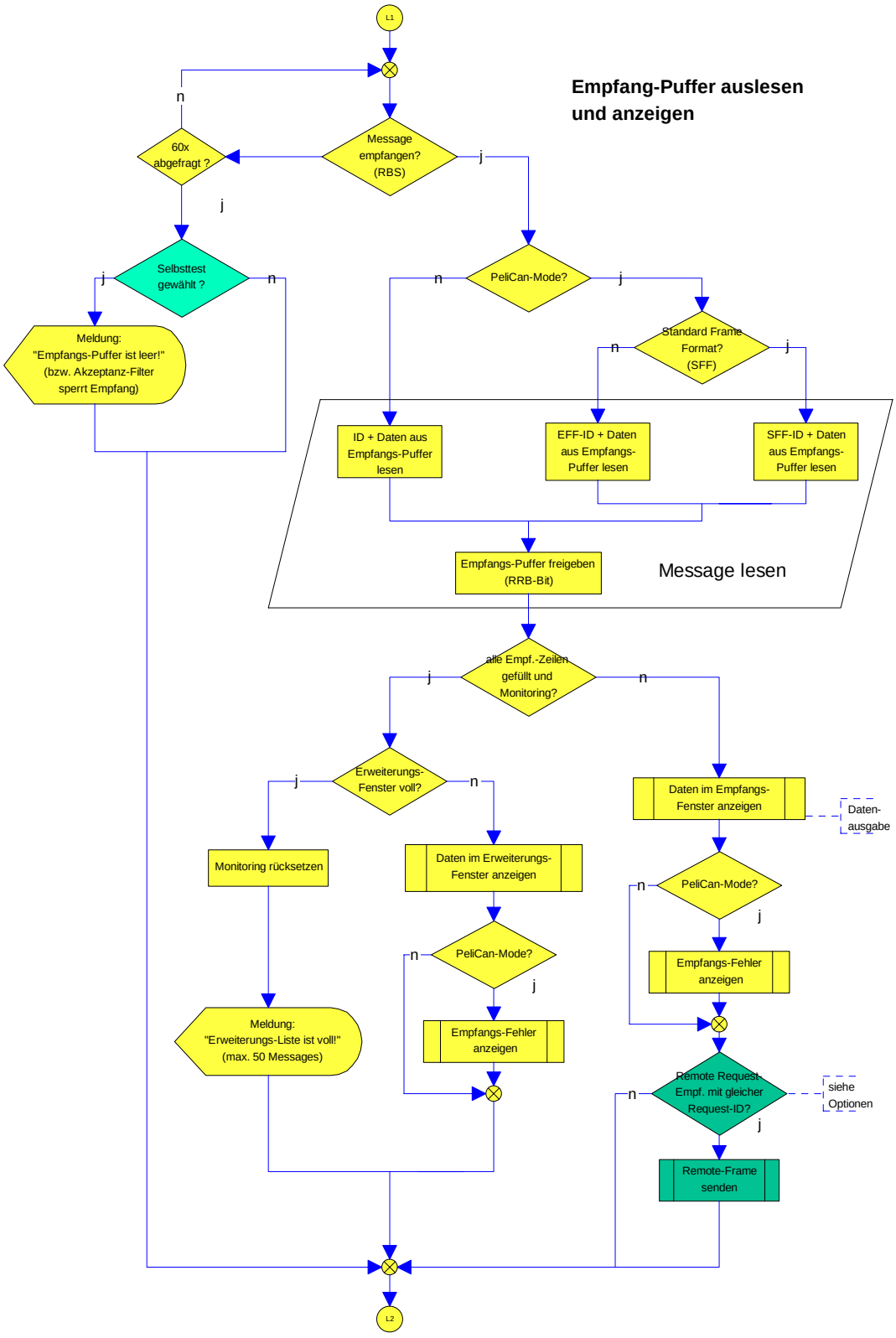


Senden, Remote Request und Selbsttest



Empfang

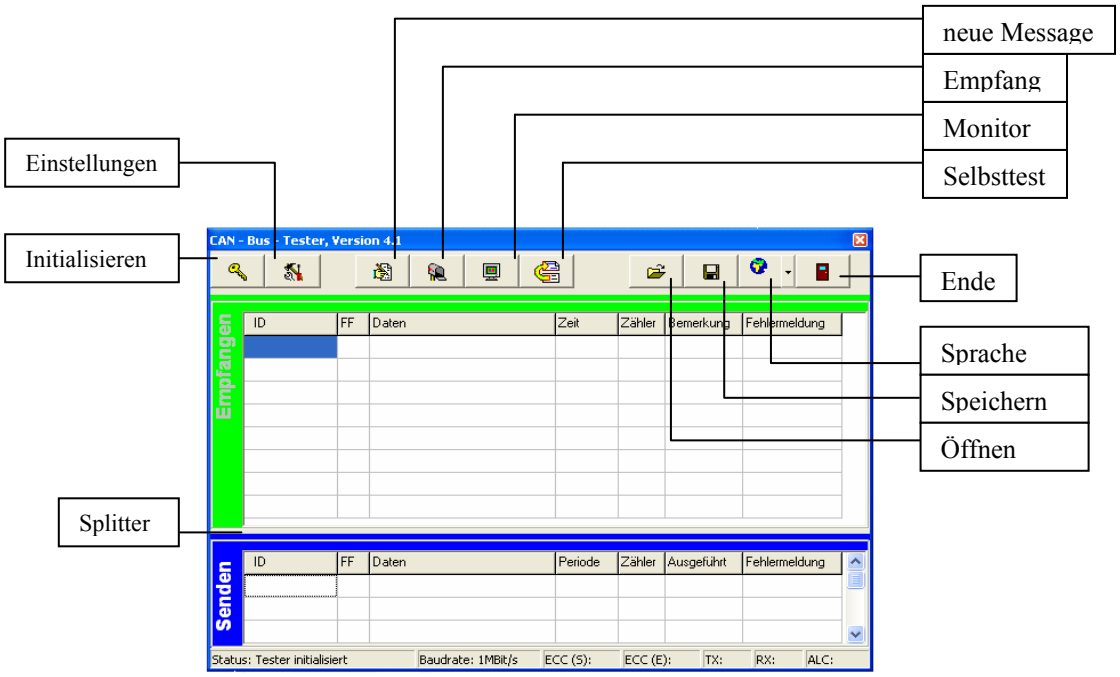

Start
L1
Empfang-Puffer auslesen u. anzeigen Fehlerausgabe
L2
ESC-Abbruch ?
j
n
gewählten Betrieb zurücksetzen
Meldung: "Abbruch mit ESC !"
Remote Request Anforderung ?
j
wurde beim Senden gesetzt
n
Monitoring gewählt ?
j
n
Empfang gewählt ?
j
Empfangs-Button startet und beendet Empfang
n
Status melden
Ende


Empfang-Puffer auslesen
und anzeigen
L1
Message empfangen? (RBS)
60x abgefragt ?
Selbsttest gewählt ?
Meldung: "Empfangs-Puffer ist leer!" (bzw. Akzeptanz-Filter sperrt Empfang)
PeliCan-Mode?
Standard Frame Format? (SFF)
ID + Daten aus Empfangs-Puffer lesen
EFF-ID + Daten aus Empfangs-Puffer lesen
SFF-ID + Daten aus Empfangs-Puffer lesen
Empfangs-Puffer freigeben (RRB-Bit)
Message lesen
alle Empf.-Zeilen gefüllt und Monitoring?
Erweiterungs-Fenster voll?
Monitoring rücksetzen
Meldung: "Erweiterungs-Liste ist voll!" (max. 50 Messages)
Daten im Erweiterungs-Fenster anzeigen
PeliCan-Mode?
Empfangs-Fehler anzeigen
Daten im Empfangs-Fenster anzeigen
Daten-ausgabe
PeliCan-Mode?
Empfangs-Fehler anzeigen
Remote Request-Empf. mit gleicher Request-ID?
siehe Optionen
Remote-Frame senden
L2

5. Das Hauptfenster

Das Hauptfenster des CAN-Bus-Testers, das im wesentlichen aus Empfangs- und Sende-Fenstern besteht, aber eine Vielzahl von Funktionen hat, macht den Tester zum universalen Werkzeug:

1. Die Toolleiste mit den Funktionstasten

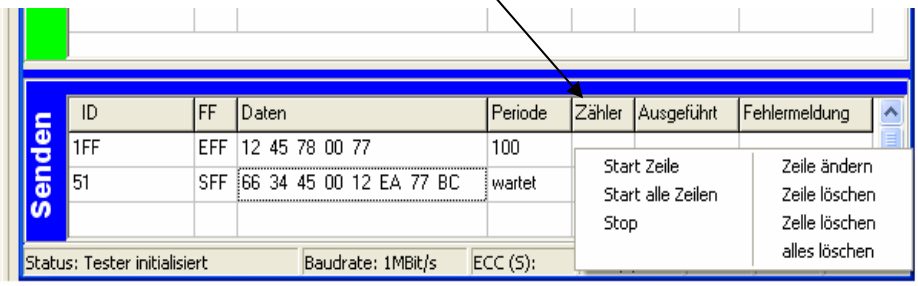


2. Im Empfangs- und Sende-Fenster kann mit der rechten Maustaste ein Untermenü aufgerufen werden. So können z.B. markierte Zellen gelöscht werden, wenn die Fehlerwiederholung geprüft werden soll.
3. Mit dem Fensterteiler (Splitter) können Sie das untere Fenster verschieben.
4. In der Status-Leiste werden neben der Tester-Bereitschaft und –Einstellung auch die auftretenden Übertragungs- bzw. Empfangs-Fehler angezeigt (ECC, ALC, TX und RX). Die Details dazu finden Sie im Hardware-Kapitel bzw. [3].
5. Die Grundfunktionen des CAN-Bus-Testers sind mit den Eigenschaften des CAN-Bausteins in fünf Betriebsarten gegliedert:

   - ein Objekt senden,
   - ein Objekt empfangen,
   - ein Remote-Objekt anfordern (Remote Request),
   - Monitoring
   - Selbsttest.

Senden

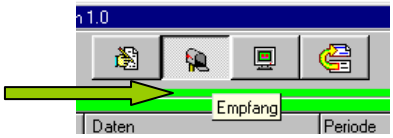
Vor dem Senden muß zuerst ein Kommunikationsobjekt mit Hilfe des "Message-Handlers" (s.S.19) aufbereitet und in den Sendebereich des Hauptfensters übergeben werden. Erst danach startet man mit der rechten Maustaste die Übertragung der Message (von der markierten Zeile). Alles weitere – Busarbitrierung, Formatierung, Fehlersicherung, Überwachung und Datensendung – wickelt die Datenübertragungs-Steuerung ab. Eine Wiederholung der Datensendung erfolgt nur durch erneutes Starten oder zyklisch, wenn eine Periode vorgegeben ist. Nach dem Stop einer zyklischen Übertragung können Sie einen erneuten Start erst durchführen, wenn zuvor die Zähler-Zelle gelöscht wurde. Ansonsten bleibt der manuelle Start erhalten.
Beim Senden *aller Zeilen* werden die Datenpakete nacheinander übertragen, wobei die Perioden-Eingabe ignoriert wird.



Achtung: Der periodische Start darf nur mit einer Message erfolgen!

Als Ergebnis wird der Status rückgemeldet. Im PeliCan-Mode erscheinen im Fehlerfall zusätzlich die übergeordnete Fehlerbezeichnung in der Fehlerspalte, der ECC-Code und der Inhalt vom ALC-Register in der Statuszeile. Die Entschlüsselung dafür finden Sie im Anhang A1 und A2 bzw. im Hardware-Kapitel.

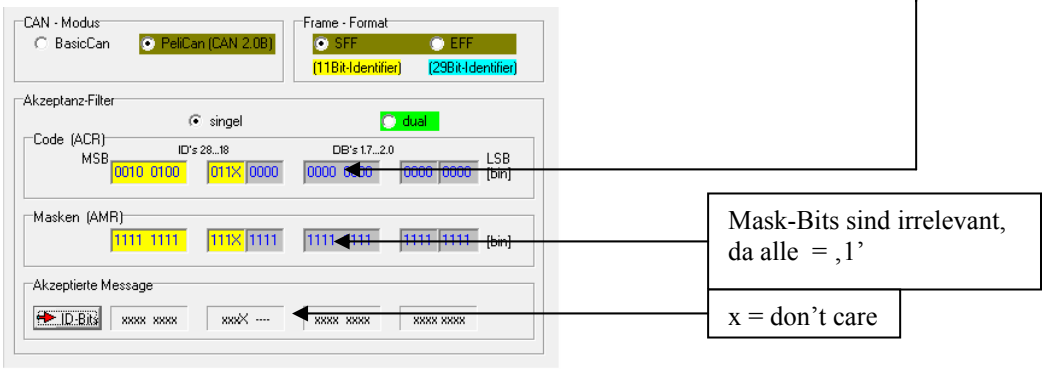
Empfang



Den Empfang leitet man mit der Betätigung des Toolbutton ein, wobei alle im Empfangs-Puffer vorhandenen Messages wiederholt ausgelesen werden. In dieser Betriebsart werden all die Telegramme vom CAN-Bus dargestellt, die durch das Akzeptanz-Filter freigegeben sind (siehe Filtereinstellung in Optionen). Dabei ist es egal, ob der Tester die Einstellung BasicCAN- oder PeliCAN-Mode hat. Der Tester ermittelt das Frame Format (FF) selbst; Standard- (SFF) oder Extended-Format (EFF). EFF-Messages können nur im PeliCAN-Mode empfangen werden.
Ein auf dem Bus aufgeschaltetes Datenobjekt wird also in den internen Puffer des µC nur übernommen und im Empfangs-Fenster ausgegeben, wenn dessen ID-Bits vom Akzeptanz-Filter toleriert werden.

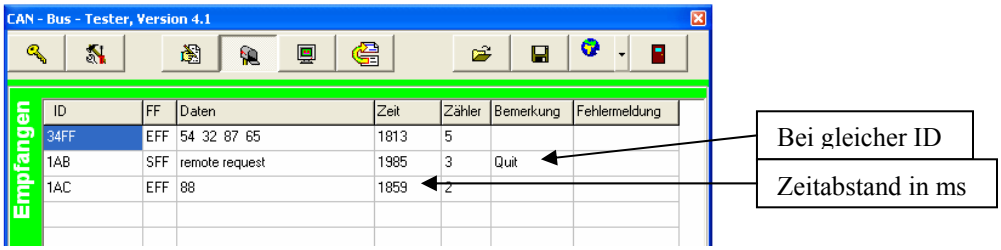
Beispiel :
Wenn eine SFF-Message empfangen wird, dann wird dessen 11-Bit-Identifier mit den Masken- und Code-Bits verglichen (ID-Bit 0...10 + RTR gemäß ALC-Register, siehe Anhang A1).

Remote-ID: 0x123 → ID-Bit10... 0 : 0010 0100 011
RTR-Bit: *x*



Mit dieser Einstellung werden alle SFF-ID's akzeptiert, da alle Mask-Bits = 1

Empfängt der CAN-Bus-Tester einen *Remote-Frame* (besteht nur aus ID+RTR-Bit), dann wird bei Übereinstimmung des empfangenen Identifiers mit der Tester-ID als Antwort die gleiche ID zurückgesendet (siehe Kapitel Optionen) und in der Empfangs-Bemerkung mit "Quit" bestätigt.



Die max. Anzahl der aufzuzeichnenden Messages mit unterschiedlichen ID's ist acht.
Messages mit gleicher ID werden in der gleichen Zeile von Nachfolgenden überschrieben.
Wenn Sie alle Messages mit gleicher ID aufzeichnen wollen, dann sollten Sie die Betriebsart ″Monitor″ wählen.

Bei Empfangsfehlern erscheint im PeliCan-Mode die übergeordnete Bezeichnung des Fehlers in der Fehlerspalte und zusätzlich der ECC- und ALC-Code in der Statuszeile. Die Entschlüsselung dafür finden Sie im Anhang A1 und A2 bzw. in der „Hardware-Beschreibung".

Beendet wird der Empfang durch Rücksetzen des Empfangs-Button (erneutes Anklicken), oder alternativ mit der ESC-Taste.

Remote Request

Bei der Anforderung eines Remote-Objekts (z.B. für Systemüberwachung oder Diagnose) sendet der anfordernde Knoten zuerst einen Identifier über den CAN-Bus.

z.B. ID: 1ABh .

Danach wartet er auf die Antwort eines anderen Knoten, von dem diese ID verwaltet wird (er müsste die ID: 1ABh haben); d.h. nach der Anforderung (Senden) wird im Programm automatisch auf Dauerempfang geschaltet und die Rückmeldung erwartet. Der antwortende Knoten muß in diesem Fall also auf Empfang stehen.

Die Handhabung ist identisch mit dem normalen Senden. Als Ergänzung gibt es hier jedoch die Möglichkeit, die zyklische Anforderung mit der ESC-Taste abzubrechen, falls keine Rückmeldung kommt. Dies tritt dann auf, wenn Sende- und Remote-ID nicht übereinstimmen (siehe Empfang).

In dieser Betriebsart werden sowohl beim Senden, wie auch beim Empfang, keine Daten übertragen, sondern nur der Identifier (ID).

Achtung: Alle Teilnehmer am CAN-Bus müssen mit der gleichen Baudrate arbeiten (siehe Kapitel Optionen) !

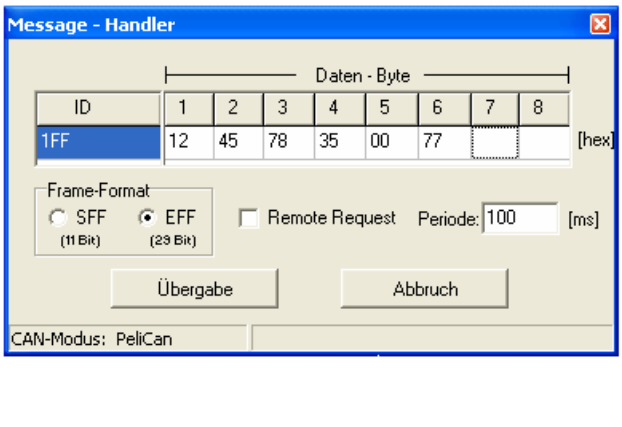
6. Message-Handler

Durch Betätigen des Button ″neue Message″ können neue Daten und Einstellungen für das Senden an das Hauptfenster übergeben oder eine Fern-Nachfrage - ***Remote Request*** – vorbereitet werden.

Die Eingabe des CAN-Identifiers (ID) und die Daten-Bytes müssen dabei als Hex-Wert erfolgen. Außerdem ist darauf zu achten, dass der ID im PeliCan-Mode bei

- SFF: 0 bis 7FFh und bei
- EFF: 0 bis 1FFFFFFFh

annehmen kann. Im BasicCan-Mode ist nur das SFF möglich!



Die Mode-Einstellung für die Message – die in der Statusleiste zu sehen ist – wird beim CAN-Bus-Tester im Kapitel ″Einstellungen″ vorgenommen. Das Frame Format (FF) können Sie aber zusätzlich auch hier anpassen.

Wenn bei der Eingabe nicht alle Daten-Bytes gefüllt werden, dann werden sie bei der Übertragung ins Hauptfenster linksbündig aneinandergereiht. Sind also Nullen erwünscht, dann müssen sie explizit eingefügt werden.

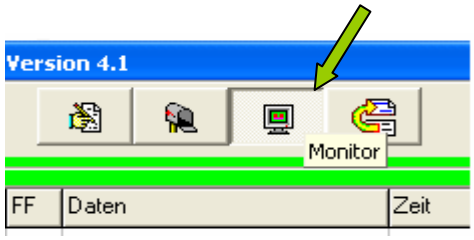
Um eine zyklische Übertragung der Daten zu erreichen, müssen Sie deren Periode bestimmen.

Wollen Sie nur ein ″Remote Request″ starten, dann ist neben dem Eintrag der ID die CheckBox zu markieren und wenn zusätzlich eine zyklische Anforderung gewünscht wird, die Periode einzutragen. Da in diesem Fall keine Daten berücksichtigt werden, ist dessen Eingabe unterbunden bzw. gelöscht.

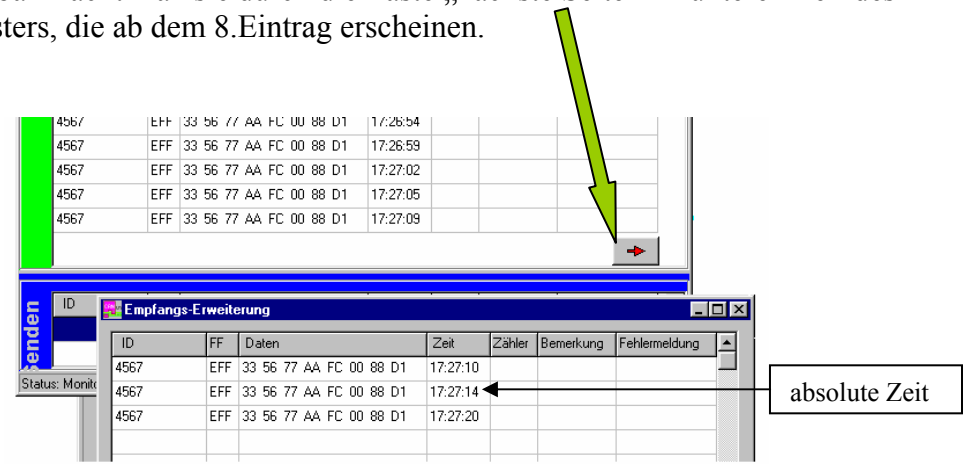
7. Als Monitor

Im Monitor-Betrieb – der nur im PeliCan-Mode möglich ist - zeichnet der Tester den gesamten Datentransfer des CAN-Busses auf, wobei es egal ist, wer sendet und wer empfängt. Zu diesem Zweck wird das Mode-Register des µC mit der Einstellung ″list only″ initialisiert. Als passiver Monitor wird dann

- der Empfangs-Puffer zyklisch ausgelesen, wenn Objekte auf dem Bus gesichtet wurden,
- kein Acknowledge-Signal generiert,
- kein Überladen des Empfangs-Puffers berücksichtigt,
- die Akzeptanz-Filter AMR und ACR aktiviert,
- die Fehler-Analyse arbeitet wie im normalen Betrieb,
- der Abbruch durch erneutes Betätigen der Monitor-Taste, oder alternativ mit der ESC-Taste vollzogen.



Im Monitor-Betrieb werden die ersten 8 Messages – genau wie beim normalen Empfang – im Empfangsfenster sofort dargestellt und alle weiteren (50 Messages) in einem weiteren Fenster. Sichtbar macht man sie durch die Taste „nächste Seite“ im unteren Teil des Empfangsfensters, die ab dem 8.Eintrag erscheinen.



Anders als beim Empfang, wird hier also jede Messages aufgezeichnet und die mit gleicher ID auch nicht überschrieben. Gelöscht werden die Einträge mit der rechten Maustaste.

Anmerkung:
Wenn Sie das Monitoring benutzt haben und den Tester nicht spannungslos machen, dann sollten Sie ihn anschließend neu initialisieren. Andernfalls kann es zu unerklärlichen Fehlern kommen.

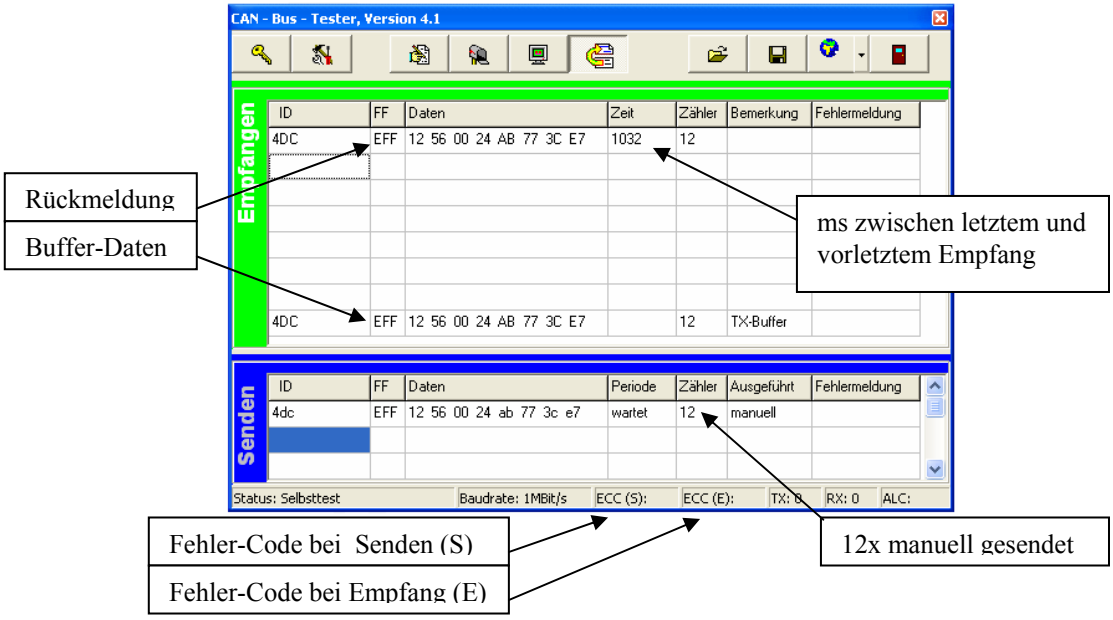
8. Der Selbsttest

Die Funktionen des lokalen Selbsttestes sind eine Kombination aus:

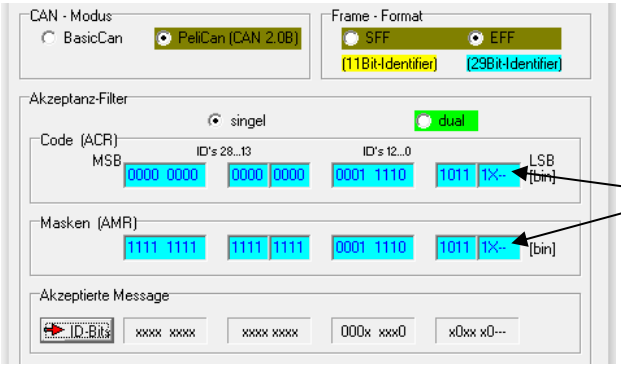
- Dateneingabe in den CAN-Controller
- Message senden
- Empfangssimulation,

die nur im PeliCan-Mode benutzt werden können und ohne Acknowledge-Bit anderer Knoten arbeiten. Mit dieser Einstellung haben Sie die Möglichkeit, die Kommunikation zwischen PC und CAN-Bus-Tester und sogar den CAN-Bus mit seiner Termination zu testen. Außerdem besteht hier die hervorragende Gelegenheit, die Anwendung der Akzeptanz-Filter auszu-probieren (siehe Optionen).
Der Ablauf ist genauso, als wenn Sie ein Datenobjekt absenden, nur dass zuvor zusätzlich die Selbsttest-Taste betätigt sein muß.

Bei fehlerfreiem Ablauf erscheinen nach dem Start im Empfangsfenster zur Bestätigung alle rückgelesenen (gesendete) Messages und in der 8.Zeile zusätzlich die TX-Buffer-Daten des µC's (Zwischen-Buffer). Hiermit können Sie also den normalen Sende/Empfangs-Ablauf, die Remote Request-Funktion und den zyklischen Betrieb anschaulich ablaufen lassen. Zur genaueren Betrachtung sollten Sie die Funktionen ″Senden″, ″Empfang″ und ″Anforderung eines Remote-Objekts″ zu Hilfe nehmen.



Die Anwendung des Akzeptanz-Filters beim erweitertem Frame Format (EFF) ist analog wie beim standard Format (SFF) .

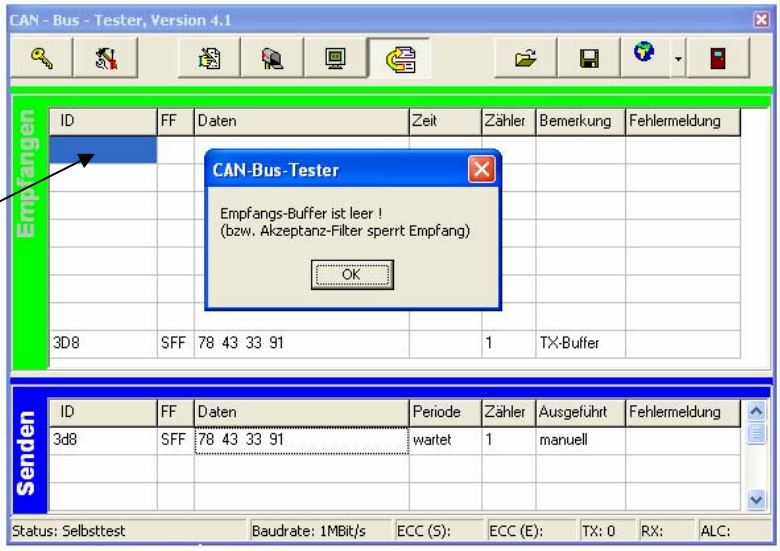


Liegt eine EFF-Einstellung vor, dann werden 29-Bit-Identifier und das RTR-Bit verglichen (ID-Bit0...28 + RTR gemäß Anhang A1).

ID = 0x3D7

Wenn Sie aber die nicht zulässige ID = 3D8h wählen würden, käme keine Rückmeldung, sondern eine Fehlermeldung.

Rückmeldung fehlt



Bei der zyklischen Ausführung sollten Sie die Periode nicht zu klein wählen (typ. 100ms), da einige PCs es dann nicht schaffen, alle Anweisungen ordnungsgemäß auszuführen (siehe auch Besonderes).

Anmerkung:
Wenn Sie den Selbsttest benutzt haben und den Tester nicht spannungslos machen, dann sollten Sie ihn anschließend neu initialisieren. Andernfalls kann es zu unerklärlichen Fehlern kommen.

9. Einstellungen als Optionen

A) Übertragungsrate

Um den Tester in unterschiedlichen Netzen benutzen zu können, lässt sich die Übertragungsrate entsprechend anpassen,d.h. alle miteinander verbundene Geräte müssen auf die gleiche Baudrate eingestellt sein!

B) Filter

Für den *Remote Request* – Betrieb besteht die Möglichkeit, einen Identifier frei zu gestalten, der durch Doppelklick in das Code-Objekt (ACR) übergeben wird.

Der CAN-Bus-Tester kann sowohl im **BasicCan**-Mode sowie im **PeliCan**-Mode (CAN 2.0B-Protokoll) mit seinen erweiterten Eigenschaften, dem standard Frame Format (SFF) oder dem extended Frame Format betrieben werden.
Die Wahl des Frame Format's ist nur für die ID-Akzeptanz ausschlaggebend, was bedeutet, es sind entweder 11Bit oder 29Bit beim Filterobjekt bzw. Maske einstellbar.
Im BasicCan-Mode kann nur mit dem standard Frame Formate gearbeitet werden; die EFF-Wahl ist also unterbunden (Grauschaltung).

Mit Hilfe der Akzeptanz-Filter ist es also möglich, diejenigen Empfangs-Objekte zu bestimmen, die vom Tester akzeptiert werden sollen. Das bedeutet, die empfangenen Daten werden bitweise mit dem Filter-Register verglichen und das Masken-Register bestimmt die Position, wo der Vergleich relevant sein soll (″0″ = relevant, ″1″ = not).

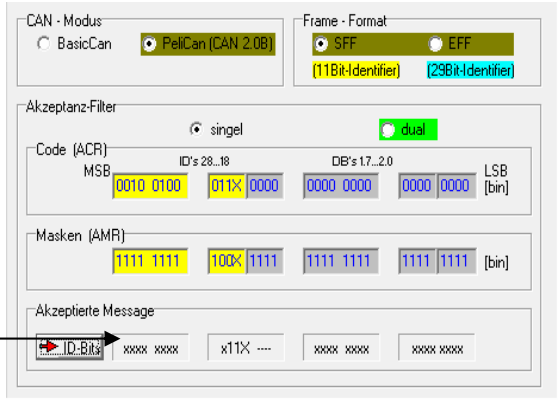
Vereinfacht: ist die Remode-ID (=ACR) z.B. 0x123, die Maske 0x03 negiert und die Message-ID = 0x123, dann wird die Message akzeptiert !

Wie die einzelnen Filter-Beziehungen funktionieren, können Sie durch Betätigen der Taste ″ID-Bits″ erfahren und im ***Selbsttest*** erproben. Weitergehende ausführliche Informationen finden Sie in den Applikationen [1][3].

Ist das Filter-Objekt: 0x123 und SFF aktiv, dann werden 11 Bits verglichen (ID-Bit 0...10).

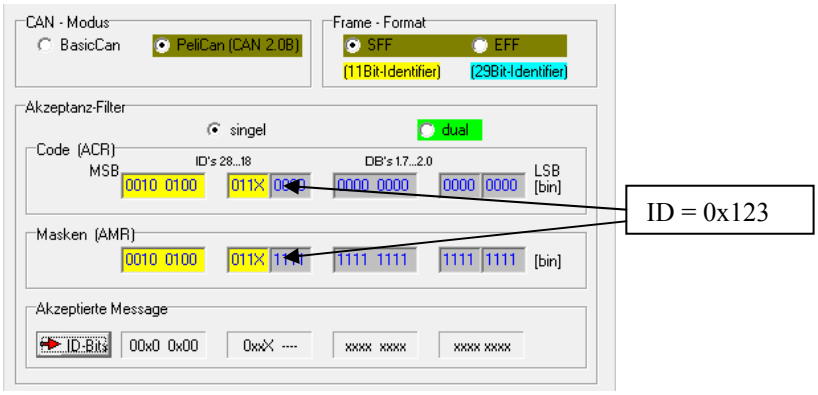
ID-Bit 10...0 (0x123) = 0010 0100 011X →

akzeptierte Message-Möglichkeiten

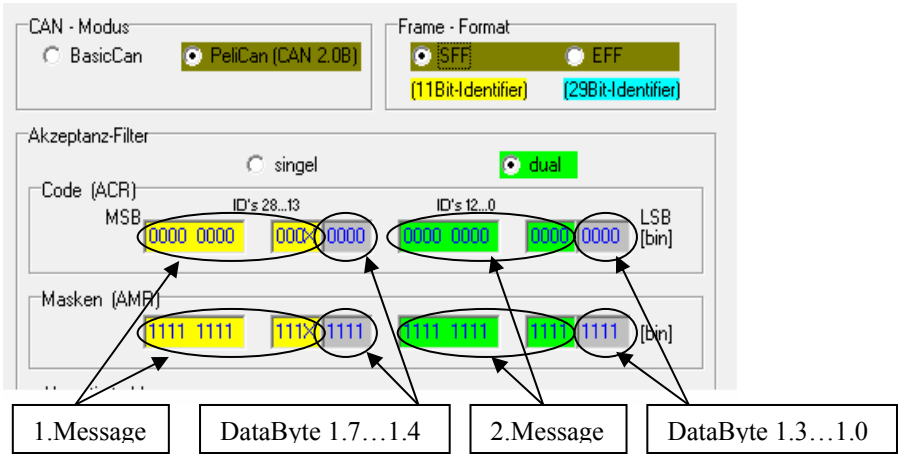


Bei dieser Konfiguration werden alle Messages akzeptiert, die am Ende ihrer ID eine 3 haben! Anders bei der nachfolgenden Einstellung.

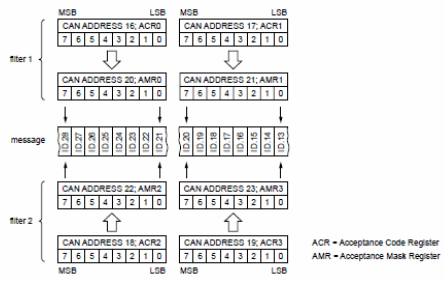
Hier wird z.B. die SFF-Message mit der ID = 0x123 akzeptiert und alle bei denen das „x“ vom Akzeptanz-Feld deckungsgleich ist.



Während im BasicCan-Mode nur ein Masken- und Code-Byte gelten, stehen im PeliCan-Mode vier Register und ein ″single long Filter″ mit 32 Bit oder zwei ″short dual Filter″ (jedes mit 16 Bit) zur Verfügung.



Mit der obigen Einstellung werden die 2 Filter-Anforderungen und zusätzlich das 1.Daten-Byte akzeptiert.



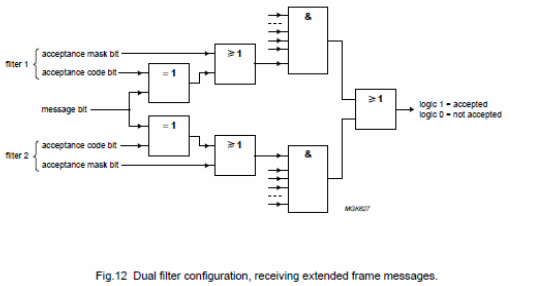


Fig.12 Dual filter configuration, receiving extended frame messages.

10. Initialisieren

Die Initialisierung ist erforderlich, wenn die Datenübertragung auf dem Bus gestört ist oder der Tester durch Fehlerüberlauf (Counter = 256) den Zustand „Bus-Off“ annimmt (siehe Besonderes).

Durch die Initialisierung wird die Steuerung (Tester) neu gestartet.

11. Besonderes

Hilfe bei der Behandlung von Meldungen:

1. Beim Auftreten der Status-Meldung: ″Überlauf des Fehler-Zählers″ sollten Sie den Tester neu initialisieren, womit auch ein Hardware-Reset erfolgt, der das „Error Warning Limit Register“ löscht. Ohne Reset bleibt die Fehlermeldung erhalten.

2. ″Bus-Off″ bedeutet, der Can-Bus-Tester ist vom Bus getrennt.

3. Die Fehlermeldung: ″Neue Message nicht abgesetzt. Es ist noch eine Übertragung aktiv ! ″ bedeutet, dass die Übertragung nicht mit ACK quittiert wurde (ECC(S): 19 = Acknowledge slot; Anhang A2). Die Übertragung bleibt solange aktiv, bis neu initialisiert wird.

4. Die Fehlermeldung: ″USB-Schreiben ist fehlerhaft″ bzw. ″USB-Lesen ist fehlerhaft″ erscheint, wenn die USB-Verbindung unterbrochen ist. Sie müssen dann alle nachfolgenden Meldungen quittieren, um das Tester-Programm neu benutzen zu können.

5. Bei einigen PC’s kann es vorkommen, dass der periodische Selbsttest oder Senden nicht gestoppt werden kann. In diesem Fall war die Periode zu klein gewählt worden. Die Lese- bzw. Sende-Routine, die im Grunde sehr lang ist, verhindert neue Tasten-Ausführungen und es bleibt dann nur noch der Task-Abbruch.

12. Anhang

A1) Interpretation des ALC-Registers (Arbitration Lost Capture)

Hiermit lässt sich die genaue Position im Daten-Frame auffinden, wo das jeweilige Bit verloren ging.

Nr.	ALC [hex]	Arbitration lost in Identifier-Bit
0	00	ID.28
1	01	ID.27
2	02	ID.26
3	03	ID.25
4	04	ID.24
5	05	ID.23
6	06	ID.22
7	07	ID.21
8	08	ID.20
9	09	ID.19
10	0A	ID.18
11	0B	SRR
12	0C	IDE
13	0D	ID.17
14	0E	ID.16
15	0F	ID.15
16	10	ID.14
17	11	ID.13
18	12	ID.12
19	13	ID.11
20	14	ID.10
21	15	ID.9
22	16	ID.8
23	17	ID.7
24	18	ID.6
25	19	ID.5
26	1A	ID.4
27	1B	ID.3
28	1C	ID.2
29	1D	ID.1
30	1E	ID.0
31	1F	RTR

RTR-Bit bei SFF

Kennung für ID-Erweiterung (=EFF)

nur im EFF

Details siehe in [1] und [3].

A2) Interpretation des ECC-Codes *(Error Code Capture)*

Die Reihenfolge entspricht der Datenübertragung, wie sie in der ″arbitration lost bit number″ (ALC) zu sehen ist. Der ECC-Code gibt also den Bereich eines Daten-Frame's an, in dem der Fehler auftritt.

ECC [hex]	**Funktionen**
03	Start of frame
02	ID.28 … ID.21
06	ID.20 … ID.18
04	Bit SRTR
05	Bit IDE
07	ID.17 … ID.13
0F	ID.12 … ID.5
0E	ID.4 … ID.0
0C	Bit RTR
0D	Reserved Bit 1
09	Reserved Bit 0
0B	Data length code
0A	Data field
08	CRC sequence
18	CRC delimiter
19	Acknowledge slot
1B	Acknowledge delimiter
1A	End of frame
12	Intermission
11	Active error flag
16	Passive error flag
13	Tolerate dominant bits
17	Error delimiter
1C	Overload flag

13. Technische Daten

Der CAN-Tester

Anwendung	Nach ISO 11898 (high-speed) für 5 und 12V Systeme
Betriebsspannung durch USB	=5V und =3,3V
Stromaufnahme	26,5mA typ. , 30mA max.
Ein/Ausgangssignale	max. +18V, diff. 1,5Vmin.
Arbeitsfrequenz	16MHz
CAN-Datentransfer	einstellbar 100kBit/s...1MBit/s
USB-Datentransfer	12MBit/s (full speed)
Temperaturbereich	-40 ... +85°C
CAN-Protokoll	2.0A und 2.0B
CAN-Klemmspannung	-8 … +18V max.
CAN_H und CAN_L	differentieller Pegel
CAN-Bus-Länge	typ. 100m bei 500kBit/s
Terminator wählbar	120Ω
PC-Anschluß	USB - B
CAN-Bus-Schnittstelle	9pol. Sub-D-Stecker
PC-Betriebssystem	Windows 98SE, Me, 2000, XP, Win7, 8, 10
Monitor-Einstellungen	1024x768 Bildpunkte True Colour (24Bit) kleine Schriftarten
Abmessungen LxBxH	80x61x21mm
Gewicht	ca. 57g

CE

Dieses Gerät entspricht dem EMV-Gesetz (EG-Richtlinie 89/336/EWG) sowie der Niederspannungsrichtlinie (73/23 EWG)

Literatur:

[1] Data SHEET, IC 18, SJA 1000 Stand-alone CAN controller, von Philips Semiconductors.
[2] Application Note AN96116, PCA 82C250, von Philips Semiconductors.
[3] Application Note AN97076, SJA 1000 Stand-alone CAN controller, von Philips Semiconductors.