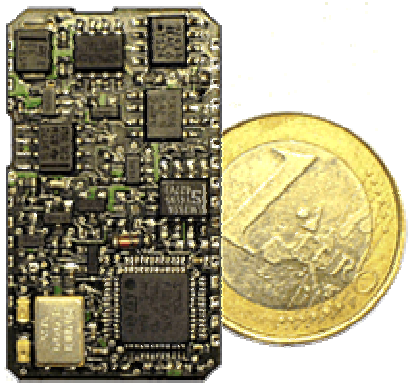
DXM1

DIAMEX®

OBD2-MODUL MIT SERIELLER SCHNITTSTELLE (Version 1.0)

Protokolle: ISO9141-2, ISO14230-2 (KWP2000), ISO15765-CAN, J-1850 PWM, J-1850 VPWM
Automatische Protokollerkennung oder Pre-Selektion des Protokolls
5 Baud Slow-Init und Fast-Init bei KWP2000
4 CAN-Protokolle (11/29 Bit, 250/500 kBaud)
Automatischer Wakeup bei ISO/KWP2000
Kurzschlußfester K-Line Ausgang
Pass-Through Fähigkeit
Keine externen Bauteile zur Verbindung mit der OBD2-Schnittstelle notwendig.
Multiframe-tauglich (Antworten von mehreren ECUs)
Serielle Schnittstelle mit 9.600 bis 250.000 Baud, kompatibel zu MAX3232 und FT232R
Bios-Update durch Bootloader möglich
Super schneller ARM Cortex-Controller mit 72 MHz Taktfrequenz
AT-Befehlssatz, hierdurch kompatibel zu vielen existierenden Programmen
Intelligente Protokoll-Scan-Funktion mit Speicherung des letzten Protokolls
2 Leuchtdioden für Connect und Datenfluss
Veränderbarer Identifikationstext zur Personalisierung des Chips
Stromversorgung 3,3 Volt für Controller
Stromversorgung 12 Volt aus OBD-Buchse
3,3 Volt Spannungsregler On-Board (für Versorgung aus 12 Volt)
Modulgröße 35 x 20mm
Made in Germany



Beschreibung

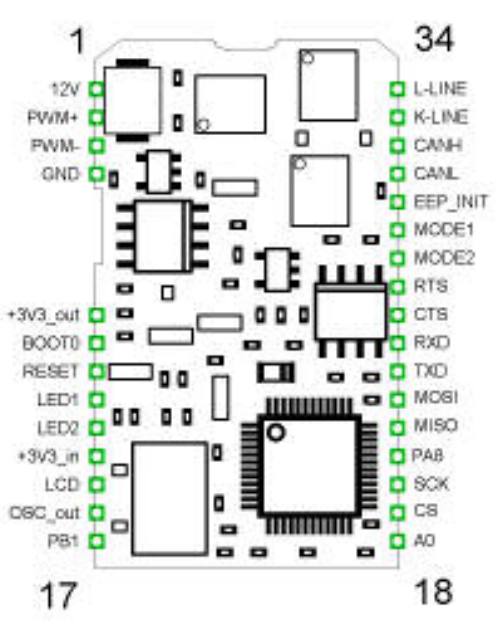
Die neue Innovation für industrielle und private On-Board-Diagnostik - klein, leistungsstark, flexibel und erstaunlich preiswert. Mit DXM dem lassen sich in kürzester Zeit eigene OBD2 Anwendungen erstellen, sowohl Soft- als auch Hardware. Das Modul beinhaltet die komplette Hardware für OBD2-Pkw-Diagnostik, inklusive der zugehörigen Protokolle und der zugehörigen Kommunikationsfirmware für die komfortable Verbindungsaufnahme via PC, Notebook oder Co-Prozessor. Die Verbindung zum Auswertungskomplex wird über die Standard TxRx-Schnittstelle zur Verfügung gestellt. Zum Aufbau einesfunktionierenden OBD2 -Interface benötigt man nur die folgenden Komponenten

- OBD2 Anschluss (via 9poligem SUB-D-Stecker)
- Shuttleboard zur Plazierung der elektronischen Bauelemente
- Serialwandler MAX3232 nebst fünf 100n Kondensatoren
- Schutzdiode + Siebelko 47µ, 35V

DXM ist das Resultat mehrerer Jahre OBD2-Interface-Entwicklung. Entstanden ist ein ultra-kompaktes, sehr einfach integrierbares SMD-Prozessormodul, das leicht zu programmieren und konfigurieren ist. Ausgestattet mit modernster 32-bit Prozessortechnologie und der darauf basierenden, leistungsfähigen Firmware, dient es als Basis für die verschiedensten Applikationen in Industrie- und Privatanwendungen. Komplett integrierte Hardware zum Sofortanschluss an die Kfz-Diagnoseschnittstelle eröffnet dabei völlig neue Perspektiven. Ingenieurs- und Entwicklungsleistungen sowie -kosten können eingespart werden. Wertvolle Entwicklungszeit, insbesondere die resourcen- und materialintensive Anpassung an unterschiedliche Einsatzgebiete, Toleranzen und Normabweichungen, können so elegant umgangen werden. Resultat ist ein preiswertes und schnell umgesetztes Endprodukt auf hohem technischem Niveau - ein genügsames, leistungsfähiges OBD2-Herzstück.

DXM - *solide, innovativ, zukunftssicher und preiswert.*

DXM1 Pinbelegung



PIN	Funktion
1	+12 Volt zur aus dem Fahrzeug zur Versorgung der OBD2-Schnittstellen
2	Ausgang PWM+/VPWM zum OBD2-Anschluß PIN 2
3	Ausgang PWM- zum OBD2-Anschluß PIN 10
4	Masseanschluss
5	Nicht vorhanden
6	Nicht vorhanden
7	Nicht vorhanden
8	Nicht vorhanden
9	3,3 Volt Ausgang, wird aus der 12 Volt Fahrzeugspannung gewonnen. Dieser Pin kann mit PIN15 verbunden werden, um das Modul mit 3,3 Volt Betriebsspannung zu versorgen. Maximale Belastbarkeit ca. 100mA
10	BOOT0, wird nur zur Programmierung des Moduls benötigt. Muss offen bleiben für den Normalbetrieb.
11	RESET des Moduls, wenn dieser Pin auf GND gelegt wird. Interner Pullup-Widerstand.
12	LED1, low aktiv
13	LED2, low aktiv
14	+3,3 Volt, Stromversorgung für den Mikrocontroller auf dem DXM1
15	LCD, Ausgang (für spätere Erweiterungen)
16	8 Mhz Takt Ausgang, zum Anschluss von externen Mikrocontrollern
17	PB1 (freier Portpin des Mikrocontrollers für spätere Erweiterungen)
18	A0, Ausgang (für spätere Erweiterungen)
19	CS, Chip-Select für die SPI-Schnittstelle (für spätere Erweiterungen)
20	SCK, SPI-Schnittstelle (für spätere Erweiterungen)
21	PA8 (freier Portpin des Mikrocontrollers für spätere Erweiterungen)
22	MISO, SPI-Schnittstelle (für spätere Erweiterungen)
23	MOSI, SPI-Schnittstelle (für spätere Erweiterungen)
24	TXD, Serielle Schnittstelle Ausgang
25	RXD, Serielle Schnittstelle Eingang
26	CTS, Serielle Schnittstelle, Clear to send, Eingang (muss nicht beschaltet werden)
27	RTS, Serielle Schnittstelle, Request to send, Ausgang (muss nicht beschaltet werden)
28	MODE2, wird für spätere Erweiterungen benötigt, interner Pullup
29	MODE1, wird für spätere Erweiterungen benötigt, interner Pullup
30	Initialisiert das EEPROM auf dem Modul nach RESET, interner Pullup
31	Ausgang CAN-L zum OBD2-Anschluß PIN 14
32	Ausgang CAN-H zum OBD2-Anschluß PIN 6
33	Ausgang K-LINE zum OBD2-Anschluß PIN 7
34	Ausgang L-LINE zum OBD2-Anschluß PIN 15

Kommunikation mit dem Controller

Die Kommunikation mit dem DXM1-Modul geschieht über die serielle Schnittstelle. Die Datenrate ist im Auslieferzustand fest auf 9600 Baud, 8 Datenbits, 1 Stoppbit (9600, 8N1) eingestellt. Diese Geschwindigkeit reicht für die relativ geringe Anzahl der zu übertragenen Daten völlig aus. Optional kann die Baudrate per AT-Befehl oder über den Bootloader bis auf 250.000 Baud umgestellt werden. Die Handshakeleitungen RTS (Request To Send) und CTS (Clear To Send) können zur Datenflusssteuerung benutzt werden, sie können jedoch auch unbeschaltet bleiben, wenn sichergestellt ist, dass der Ein- oder Ausgangsbuffer nicht überläuft.

Nach Reset des DXM-1-Moduls sollte folgende Meldung über die serielle Schnittstelle ausgegeben werden:

```
DIAMEX DXM1 v1.0

>
```

Hinweis: Diese Meldung variiert je nach Version und kann zusätzlich per AT-Befehl geändert werden.

Welche Meldung auch erscheint, sie zeigt an, daß die Kommunikation des Interface funktioniert. Das Zeichen „>“ bedeutet, daß das Interface bereit ist, Befehle zu empfangen. Das Interface unterscheidet nun zwei verschiedene Kommandogruppen:

1. Interne Befehle zur Konfiguration und Initialisierung des Interface-Controllers. Alle diese Befehle beginnen mit den Zeichen „AT“, dieses wurde von den Steuerbefehlen bei Modems übernommen und bedeutet „ATtention, Achtung“.
2. Daten, die an den OBD2-Bus für das Steuergerät des Fahrzeugs weitergeleitet werden. Alle diese Befehle werden als hexadezimale Zahlen übermittelt, es dürfen deshalb nur ASCII-Zeichen 0-9 und A-F paarweise eingegeben werden.

Alle eingegebenen Befehle müssen mit einem Zeilenende-Zeichen (Carriage Return, Dez. 13, Hex $0D) abgeschlossen werden. Leerzeichen oder Tabulatoren werden automatisch herausgefiltert, Klein- und Großschreibung wird nicht unterschieden. In den folgenden Beispielen muß jede Eingabe mit dem Zeilenende-Zeichen abgeschlossen werden, es wird nicht extra angegeben.

Beispiele:

`at dp`
Wird intern nach `ATDP` gewandelt
`A T Z`
Wird intern nach `ATZ` gewandelt
`01 1c`
Wird intern nach `011C` gewandelt

Falls das Zeilenende-Zeichen CR ausbleibt, wird der Befehl automatisch nach 5 Sekunden abgebrochen und ein „?“ wird als Fehlermeldung ausgegeben. Befehle, die der Controller nicht versteht oder Falscheingaben bei Hex-Werten, werden ebenfalls mit einem „?“ als Fehlermeldung quittiert.

OBD2-Befehle müssen immer mit geraden Anzahl von Hex-Zeichen eingegeben werden:

Beispiel:
`0100`
oder
`01 00`

Eine ungerade Anzahl von Zeichen erzeugt eine Fehlermeldung.

Einige AT-Befehle benötigen als zusätzliche Parameter einen oder mehrere Hex-Zeichen. Die genaue Anzahl entnehmen Sie bitte der Befehlsliste. Eine falsche Anzahl der Parameter wird ebenfalls mit einer Fehlermeldung quittiert.

Der AT-Befehlssatz des DXM1-Moduls

>ATZ

Dieser Befehl bewirkt einen sofortigen Abbruch aller laufenden Funktionen und einen Warmstart des Controllers. Alle Parameter werden in den Grundzustand gesetzt und der Selbsttest (durch blinken der LEDs angezeigt) wird durchgeführt. Zum Schluß wird der Ident-Text ausgegeben.

Ausgabe:
```
Diamex DXM1 v1.0

>
```

Hinweis: Der Text ist abhängig von der Bios-Version und kann mit Hilfe von AT-Befehlen geändert werden.

>ATWS

Dieser Befehl bewirkt einen sofortigen Abbruch aller laufenden Funktionen und einen Warmstart des Controllers. Alle Parameter werden in den Grundzustand gesetzt. Zum Schluß wird der Ident-Text ausgegeben. Dieser Befehl hat dieselbe Wirkung wie ATZ, wird jedoch wesentlich schneller ausgeführt, da der Selbsttest übersprungen wird.

Ausgabe:
```
Diamex DXM1 v1.0

>
```

Hinweis: Der Text ist abhängig von der Bios-Version und kann mit Hilfe von AT-Befehlen geändert werden.

>ATI

Hiermit wird nur der Identifikationstext ausgegeben, ohne einen Warmstart durchzuführen.
Laufende Funktionen, wie das automatische Wakeup bei ISO oder KWP2000, werden nicht abgebrochen.

Ausgabe:
```
Diamex DXM1 v1.0

>
```

>ATD

Alle Parameter werden in den Grundzustand wie nach einem Kalt- oder Warmstart versetzt.

Ausgabe:
```
OK

>
```

>ATE0
>ATE1

Dieser Befehl schaltet das serielle Echo ein (1) oder aus (0). Alle Daten, die über die serielle Schnittstelle empfangen werden, werden bei eingeschaltetem Echo wieder zum PC geschickt.

Ausgabe:
```
OK

>
```

>ATL0
>ATL1

Über diesen Befehl wird die Übermittlung des Linefeed-Zeichens am Ende der Zeile ein- (1) oder ausgeschaltet (0). Alle zum PC gesendeten Antworten sind in der Regel nur mit einem Carriage-Return abgeschlossen (13 Dez., $0D Hex). Mit eingeschaltetem Linefeed wird hinter jedem Carriage-Return noch das Linefeed-Zeichen (10 Dez, $0A Hex) ausgegeben. Besonders wenn man mit einem Terminalprogramm die Befehl von Hand übermittelt, ist es recht sinnvoll, den Linefeed eingeschaltet zu lassen, da sonst alle Antworten in einer Zeile dargestellt werden und neue Antworten die Alten überschreiben.

Ausgabe:
```
OK

>
```

>ATM0
>ATM1

Schaltet die Memory-Funktion des letzten Protokolls ein bzw. aus.

Wenn häufig dasselbe OBD2-Protokoll verwendet wird, kann es sinnvoll sein, dieses als Voreinstellung im EEPROM abzuspeichern. Hierzu schalten Sie die Memory-Funktion mit ATM1 ein, dann stellen Sie die Verbindung mit dem Fahrzeugsteuergerät her, so daß ein Protokoll aktiviert wird, zum Schluß schalten Sie die Memory-Funktion mit ATM0 wieder aus. Ab sofort wird das aktuelle Protokoll als Standard benutzt.

Ist die Memory-Funktion eingeschaltet, können auch die Parameter ATEx, ATLx , ATHx und SBx abgespeichert werden. Nach der Speicherung sollte diese Funktion unbedingt wieder mit ATM0 abgeschaltet werden um unkontrolliertes beschreiben des EEPROM zu verhindern.

Die Memory-Funktion wird auch durch ATZ und ATWS deaktiviert.

>ATH0
>ATH1

Mit diesem Befehl kann eingestellt werden, ob bei OBD2-Antworten der Header und das Checksummenbyte mit ausgegeben werden soll.

Ausgabe:
Mit ATH0:
```
>0100
41 00 E8 19 30 12
```

Mit ATH1:
```
>0100
48 6B 10 41 00 E8 19 30 12 47
```
48, 68, 10 sind die 3 Headerbytes
47 ist das Checksummenbyte

Ausgabe:
```
OK

>
```
Da beim CAN-Protokoll keine Header- und Checksummenbytes existieren, wird statt dessen die CAN-ID übermittelt. Nähere Informationen hierzu finden Sie in der Beschreibung zum CAN-Protokoll später in diesem Dokument.

>ATBD

Die Abkürzung von „**B**uffer **D**ump“ bewirkt, den internen OBD2-Empfangsspeicher auszugeben. Gültige Daten stehen hier jedoch nur, wenn zuvor ein OBD2-Befehl ausgeführt wurde.

Beispiel:
```
>0100
41 00 E8 19 30 12

>ATBD
0A 48 6B 10 41 00 E8 19 30 12 47 00 00
```

Das erste Byte gibt die Anzahl der gültigen Zeichen im Speicher an. In diesem Fall handelt es sich um 10 Zeichen (Hex 0A). Das letzte Byte ist in diesem Fall ungültig und kann einen beliebigen Wert beinhalten.

>ATB

Anzeige des OBD2-Empfangsbuffers. Dieser Befehl unterscheidet sich vom vorhergehenden ATBD, dass hier der komplette Inhalt des Empfangsbuffers angezeigt wird, anstatt nur die ersten maximal 12 Zeichen. Dies ist vor allem bei Multiframe- und Multi-ECU-Antworten erforderlich.

Beispiel (Abfrage DTC mit 4 Fehlercodes):
```
>03
43 01 15 02 30 03 50
43 04 60 00 00 00 00

>ATB
16 02
C7 F1 10 43 01 15 02 30 03 50 A6
C7 F1 10 43 04 50 00 00 00 00 6F
```

In der ersten Antwortzeile ist die Gesamtanzahl der im Buffer befindlichen Bytes angegeben (16) und die Anzahl der Antwortframes (02). Darunter die Inhalte der der Frame-Buffer mit Header- und Checksummenbytes.

>ATSR xx

Eingabe der RX-ECU-Filteradresse für OBD2-Antworten.
Falls bei einem „Functional Request“ mehrere ECU antworten, kann durch setzen des RX-Filters das gewünschte ausgefiltert werden.

Beispiel (KWP2000 mit eingeschalteten Header):
```
>0100
C6 F1 10 41 00 B8 7B B0 10 FB
C6 F1 18 41 00 08 28 00 00 40
```

Hier antworten 2 verschiedene Steuergeräte. Sollen nur die Antworten des 2. Steuergerätes (18) ausgefiltert werden, kann dies durch Eingabe des Befehls `ATSR18`
erzwungen werden.

Bei KWP2000 wird immer das 3. Byte (Target-Address) als Filteradresse benutzt. Bei ISO9141, PWM und VPW wird abhängig vom Requesttyp das 2. oder 3. Byte benutzt. Beim „Physical Request“ wird das 3. Byte, beim „Functional Request“ wird das 2. Byte + 1 benutzt.

Beispiel (ISO9141 Functional Request):
```
Request Header    68 6A F1
Antwort           48 6B 10
```

Die Filteradresse kann nur durch Eingabe von ATZ, ATWS oder ATD ausgeschaltet werden.

Dieser Parameter hat keine Funktion beim CAN-Protokoll.

>ATN

Anzeige des aktuell benutzten Protokolls als Hexwert. F0..F9

Beispiel:
```
>ATN
F2

>
```

F0: Kein Protokoll aktiv
F1: PWM-Protokoll
F2: VPWM-Protokoll
F3: ISO9141-Protokoll
F4: KWP2000-Protokoll (5 Baud Init)
F5: KWP2000-Protokoll (Fast Init)
F6: CAN-Protokoll 11Bit-ID, 500kBaud
F7: CAN-Protokoll 29Bit-ID, 500kBaud
F8: CAN-Protokoll 11Bit-ID, 250kBaud
F9: CAN-Protokoll 29Bit-ID, 250kBaud

Eine Ausgabe von F0 bedeutet, daß zur Zeit kein Protokoll benutzt wird (zum Beispiel nach ATZ).

>ATDP

Anzeige des aktuell benutzten Protokolls im Klartext.

Beispiel:
```
>ATDP
ISO 9141-2

>
```

Hier die Liste alle möglichen Ausgaben:

NOT CONNECTED
SAE J1850 / PWM
SAE J1850 / VPWM
ISO 9141-2
ISO 14230-4, KWP2000 (5 Baud Init)
ISO 14230-4, KWP2000 (Fast Init)
ISO 15765-4, CAN (11/500)
ISO 15765-4, CAN (29/500)
ISO 15765-4, CAN (11/250)
ISO 15765-4, CAN (29/250)

>ATK

Anzeige des aktuellen Keywords bei ISO9141 und KWP2000.

Beispiel:
```
>ATK
8F E9

>
```

Wenn keine Verbindung zum Fahrzeug besteht oder ein Protokoll benutzt wird, das keine Keywords unterstützt, ist die Ausgabe ungültig.

>ATKW0
>ATKW1

Bei einem Slow-Init wird anhand des Keywords zwischen den Protokollen 3 (ISO9141) und 4 (KWP2000) unterschieden. Diese automatische Erkennung kann mit ATKW0 abgeschaltet werden. Dies ist nur zu Experimentierzwecken erforderlich. Für eine sichere Erkennung des Protokolls muss die Erkennung eingeschaltet werden.

Ausgabe:
`OK`

`>`

Voreinstellung: ATKW1

>ATP[A]x

Manuelle Einstellung des aktuellen Protokolls oder der Automatischen Protokoll-Suchfunktion.

Hiermit kann ein Protokoll fest voreingestellt werden. Es wird kein anderes Protokoll gesucht, sondern mit
`UNABLE TO CONNECT`
abgebrochen, wenn das Steuergerät nicht antwortet.

ATP1 PWM
ATP2 VPWM
ATP3 ISO9141-2
ATP4 KWP2000 5 Baud Init
ATP5 KWP2000 Fast Init
ATP6 CAN 11/500
ATP7 CAN 29/500
ATP8 CAN 11/250
ATP9 CAN 29/250

Wenn statt dessen `ATPAx` (x = Protokollnummer) eingegeben wird, durchsucht der Controller automatisch alle anderen Protokolle, wenn das Aktuelle nicht gefunden wird.

Mit `ATP0` oder `ATPA0` (identisch) wird die automatische Suchfunktion aktiviert. Es ist kein Protokoll voreingestellt und es werden nach einem Neustart alle Protokolle durchsucht, bis ein passendes gefunden ist.

Vorsicht!
Wenn ein festes Protokoll ohne Auto-Suchfunktion voreingestellt ist, wird auch kein anderes Protokoll gesucht, wenn das Steuergerät im Fahrzeug nicht auf das Eingestellte antwortet. In diesem Fall bitte die Suchfunktion mit `ATP0` oder `ATPAx` (x = aktuelles Protokoll) aktivieren.

Ist die Memory-Funktion mit `ATM1` eingeschaltet, wird die Änderung der Einstellung sofort ins EEPROM abgespeichert und bei Neustart wieder verwendet.
Durch Eingabe von `ATP` ohne zusätzliche Parameter wird der eingestellte Modus angezeigt. In diesem Fall wird keine Änderung durchgeführt.

Mehr Informationen hierzu im Abschnitt „OBD2-Protokolle“.

Im Auslieferzustand ist ATP0 voreingestellt. Es werden alle Protokolle durchsucht.

>ATV

Eine Übersicht aller veränderbaren Parameter des DXM1 wird angezeigt.

Beispiel:

```
>ATV
E1 L1 H0 M0 KW1 SW19 SR00 SB00 (9600)
  HEADER: 00 00 00
  WAKEUP:
CA1 CC1
CAN-BAUD: 00000000
CAN-TXID: 00000000
CAN-RXID: 00000000
CAN-MASK: 00000000
CAN-FLOW: 00000000
FLOWDATA: 00 00 00 00
```

E1 = Echo ein(1), aus(0)
L1 = Linefeed ein(1), aus(0)
H0 = Header ein(1), aus(0)
M0 = Memory ein(1), aus(0)
KW1 = Protokoll-Erkennung per Keyword
ein(1), aus(0)
SW19 = ISO/KWP-Wakeup 0x19 * 100ms
SR00 = Response-Byte 00 = auto
SB00 = Serielle Baudrate 00 = 9600
HEADER = ISO/KWP/PWM/VPWM Headerbytes
WAKEUP = ISO/KWP Wakeup-Kommando
CA1 = CAN Auto-Format ein(1), aus(0)
CC1 = CAN Flow-Control ein(1), aus(0)
CAN-BAUD = Baudratenbytes CAN-Controller
CAN-TXID = CAN-Sende-ID
CAN-RXID = CAN-Empfangs-ID
CAN-MASK = CAN-Empfangs-Maske
CAN-FLOW = CAN-Flow-Control-ID
FLOWDATA = CAN-Flow-Message-Daten

>ATSB x

Baudrate der seriellen Schnittstelle einstellen. Im Auslieferzustand oder nach Reinitialisierung des EEPROM-Speichers ist die Baudrate auf 9600 eingestellt. Eine geänderte Baudrate ist erst nach erfolgtem `ATZ` aktiviert. Die neue Baudrate kann fest im EEPROM abgespeichert werden, wenn der Speicher zuvor mit `ATM1` freigegeben wird.

Beispiel:

```
>ATSB4
115200
OK
>ATZ
```

Hier wird die Baudrate auf 115200 eingestellt. Bitte darauf achten, dass nach `ATZ` die Baudrate auf der seriellen Schnittstelle vom steuernden Rechner/Controller ebenfalls geändert wird, da ansonsten kein Befehl mehr möglich ist. Sollte die eingestellte Baudrate im EEPROM abgespeichert und unbekannt sein, hilft nur, alle möglichen Baudraten durchzuprobieren, oder den EEPROM-Speicher zu reinitialisieren. Danach sind wieder 9600 Baud voreingestellt. Wird die geänderte Baudrate nicht im EEPROM abgespeichert, wird nach einem RESET bzw. nach Trennen der Stromversorgung wieder die im EEPROM gespeicherte Baudrate aktiviert.

Liste der möglichen Baudraten:

```
ATSB0 = 9600 Baud (Standardwert)
ATSB1 = 19200 Baud
ATSB2 = 38400 Baud
ATSB3 = 57600 Baud
ATSB4 = 115200 Baud
ATSB5 = 125000 Baud
ATSB6 = 250000 Baud
```

>ATSH xx yy zz

Manuelles Setzen der Headerbytes für ISO9141, KWP2000, PWM und VPWM Protokolle.

Es werden 3 Hexwerte erwartet:
xx = Priority/Type-Byte
yy = Target-Address
zz = Source-Address„

Den genauen Aufbau der 3 Bytes entnehmen Sie bitte den SAE-Protokoll-Spezifikationen.

Bei KWP2000 wird die Längenangabe im 1. Byte (xx) automatisch angepaßt.

Wenn das Protokoll mit ATPx festgelegt ist, können die Headerbytes bereits vor dem 1. Verbindungsaufbau geändert werden. Ist die Auto-Suchfunktion aktiviert, werden für den Verbindungsaufbau immer die Standardwerte benutzt, hier können die Headerbytes nur nach erfolgtem Connect geändert werden.

Standardwerte:

ISO9141-2:	`68 6A F1`
KWP2000:	`Cx 33 F1` (x = Längenangabe)
PWM:	`61 6A F1`
VPWM:	`68 6A F1`

Die Headerbytes können durch Eingabe von ATZ, ATWS oder ATD auf die Standardwerte zurückgesetzt werden.

Dieser Parameter hat keine Funktion beim CAN-Protokoll.

>ATWM xx yy zz aa [bb] [cc]

Manuelles Setzen der Wakeup-Message-Bytes für ISO9141 und KWP2000.

Es werden 4-6 Hexwerte erwartet:
xx = Priority/Type-Byte (mit Längenangabe bei KWP)
yy = Target-Address
zz = Source-Address
aa,bb,cc = 1-3 Befehlsbytes

Bei KWP2000 muß das 1. Byte die richtige Längenangabe enthalten.

Beispiel (ISO9141-2):
```
>ATWM 68 6A F1 03
OK
```

Beispiel (KWP2000):
```
>ATWM C2 33 F1 01 04
OK
```

Standardwerte:
ISO9141-2: `68 6A F1 01 00`
KWP2000: `C1 33 F1 3E`

Die Wakeup-Message-Bytes können durch Eingabe von ATZ, ATWS oder ATD auf die Standardwerte zurückgesetzt werden.

Dieser Parameter hat keine Funktion beim PWM, VPWM und CAN-Protokoll.

>**ATSW xx**

Mit diesem Befehl kann die Zeitdauer zwischen den automatischen Wakeup-Befehlen bei einer bestehenden ISO9141 oder KWP2000 Verbindung eingestellt werden.

ISO9141 und KWP2000 Fahrzeugcontroller erwarten einen regelmäßigen Datenverkehr zum angeschlossenen OBD2-Interface. Sollte dieser längere Zeit ausbleiben (laut SAE-Norm 5000ms), wird die Verbindung getrennt und muß durch einen erneuten Init wieder hergestellt werden.

Dieser Befehl erwartet die Zeitdauer als 2-stelligen HEX-Code. Die Zeitdauer zwischen den Wakeup-Befehlen ergibt sich aus dem Hex-Wert * 100 Millisekunden.

Beispiel:
```
>ATSW1E
OK

>
```

Hier wird eine Wakeup-Zeit von $1E (30) * 100ms = 3,0 Sekunden eingestellt.
Der Standardwert wird mit ATZ, ATWS oder ATD auf $19 (25) eingestellt, was 2,5 Sekunden entspricht. Wird als Wert 0 eingetragen, ist der automatische Wakeup abgeschaltet und die Verbindung wird nach 5 Sekunden getrennt, wenn keine OBD2-Kommandos übertragen werden.

>**ATCA0**
>**ATCA1**

Mit diesem Befehl kann die automatische Formatierung der gesendeten und empfangenen CAN-Datenpakete ein- bzw. ausgeschaltet werden.

Ein Ausschalten der automatischen Formatierung ist nur sinnvoll, wenn der DXM-Controller in CAN-Bussen betrieben werden soll, die nicht OBD2-kompatibel sind.

Ist die Formatierung ausgeschaltet, müssen immer alle gewünschten Bytes (max. 8) des CAN-Datenpaketes eingegeben werden, bei eingeschalteter Formatierung wird automatisch das Längenbyte hinzugefügt und es dürfen max. 7 Bytes zum Senden eingegeben werden.

Beispiel:

Mit Auto-Format an:
```
>0100
```
wird umgewandelt nach:
```
02010000000000000
```

Im OBD2-Modus werden immer Datenpakete mit 8 Bytes versendet. Folgender Befehl ist gleichbedeutend mit obigem Beispiel, jedoch bei ausgeschaltetem Auto-Format:
```
>0201000000000000
```

Wird statt dessen nur
```
>020100
```
eingegeben, wird nur ein Datenpaket mit 3 Bytes gesendet, das nicht OBD2-konform ist.

Die Einstellung der Auto-Formatierung beeinflußt ebenfalls die Ausgabe der empfangenen Antworten auf dem CAN-Bus.

Beispiel:

Mit Auto-Format aus werden alle 8 Bytes des CAN-Frame ausgegeben. Es wird keine Auswertung der empfangenen Daten vorgenommen:

```
06 41 00 B8 7B B0 10 00
```

Mit Auto-Format an wird das empfangende OBD2-Datenpaket ausgewertet. Das 1. Byte zeigt in diesem Fall an, daß 6 gültige Bytes folgen. Damit wird folgendes ausgegeben:

```
41 00 B8 7B B0 10
```

In Multiframe-Antworten ist die Ausgabe der Datenpakete mit eingeschalteter Formatierung an das Format der ELM-Controller angepaßt.

Beispiel, mit Auto-Format an.
Abfrage der Fehlercodes mit 4 Fehlern:
```
>03
00A
0: 43 04 01 15 02 30
1: 03 50 04 60 00 00 00
```

Die 1. Zeile zeigt an, daß die Antwort aus 00A (hex) = 10 gültigen Bytes besteht. Jede weitere Zeile beginnt mit einem Zähler mit anschließendem Doppelpunkt, der die Reihenfolge der empfangenen Datenpakete angibt. Es wird von 0 bis F (hex) gezählt und beginnt dann wieder bei 0, wenn noch mehr Pakete übertragen werden müssen.

Mit ausgeschalteter Formatierung sieht die Antwort folgendermaßen aus:

```
10 0A 43 04 01 15 02 30
21 03 50 04 60 00 00 00
```

Hier müssen PCI- und Längenbyte von der PC-Software ausgewertet werden.

Mehr Infos zum OBD2-Datenformat finden Sie in den SAE J1979 – Spezifikationen.

Standardeinstellung: Ein (ATCA1).

>**ATCC0**
>**ATCC1**

In OBD2-Systemen müssen bei Multiframe-Antworten so genannte Flow-Control Nachrichten vom Tester gesendet werden, die dem Steuergerät anzeigen, daß nachfolgende Pakete akzeptiert werden. Dieses wird durch den DXM-Controller automatisch vorgenommen.

Soll der Controller in CAN-Systemen eingesetzt werden, die nicht OBD2-konform sind, kann es sinnvoll sein, diese automatischen Flow-Control Nachrichten auszuschalten.

Beispiel:
```
>ATCC0
OK

>
```

Standardeinstellung: Ein (ATCC1).

>**ATCD xx**

Flow-Control Datenpakete, die bei Multiframe-Antworten gesendet werden müssen, enthalten neben einem Statusbyte (FS), einem Blockgrößenbyte (BS) auch ein Byte für die Zeitdauer (ST), die zwischen den nachfolgenden Antwortpaketen eingefügt werden soll, damit der CAN-Controller Zeit hat, diese Daten zu verarbeiten. Mit diesem Befehl kann die Zeitdauer verändert werden.

Beispiel:
```
>ATCD 7F
OK

>
```

Stellt die Zeitdauer auf den maximalen Wert von 127ms ein.

Dieser Befehl ist nur zu Experimentierzwecken vorhanden und kann zum Testen von Simulatoren verwendet werden. In der Regel muß die Einstellung nicht verändert werden.

Standardwert: 0A (10 ms)

>ATCI xxx
>ATCI xxxxxxxx

Die zu sendende CAN-ID wird mit diesem Befehl gesetzt. Dies kann sinnvoll sein, wenn der DXM Controller in nicht OBD2-konformen CAN-Systemen eingesetzt wird.

Standardwerte:

11 Bit ID	7DF
29 Bit ID	18 DB 33 F1

Beispiele:

```
>ATCI 7E0
OK

>
>ATCI 18 DA 10 F1
OK

>
```

Bitte darauf achten, daß durch die Anzahl der eingegebenen Zeichen bestimmt wird, ob die 11Bit oder die 29Bit ID verändert werden soll. Für die 11Bit ID müssen immer 3 Hex-Zeichen eingegeben für die 29Bit ID müssen immer 8 Hex-Zeichen eingegeben werden.

Durch Eingabe von ATWS, ATZ oder ATD werden die ID auf die Standardwerte zurückgesetzt.

>ATCF xxx
>ATCF xxxxxxxx

CAN-RX-Filter setzen. Wenn zu viele Daten auf dem CAN-Bus übertragen werden, kann es passieren, daß der Controller-interner Puffer überläuft wenn die Daten nicht rechtzeitig zum PC übertragen werden können. In diesem Fall sollten aus den empfangenen Daten die gewünschten ausgefiltert werden. Der Filter wird zusammen mit der RX-Maske verwendet, die mit dem Befehl ATCM verändert werden kann.

Standardwerte:

11 Bit Filter	7E8
29 Bit Filter	18 DA F1 00

Beispiele:

```
>ATCF 7E0
OK

>
>ATCI 18 DA F1 10
OK

>
```

Bitte darauf achten, daß durch die Anzahl der eingegebenen Zeichen bestimmt wird, ob der 11Bit oder der 29Bit Filter verändert werden soll. Für den 11Bit Filter müssen immer 3 Hex-Zeichen eingegeben für den 29Bit Filter müssen immer 8 Hex-Zeichen eingegeben werden.

Durch Eingabe von ATWS, ATZ oder ATD werden die Filter auf die Standardwerte zurückgesetzt.

>ATCM xxx

>ATCM xxxxxxxx

CAN-RX-Maske setzen. In Verbindung mit dem RX-Filter (ATCF) kann die Maske dazu benutzt werden, einzelne oder eine Gruppe von Daten auszufiltern.

Standardwerte:

11 Bit Maske	7F8
29 Bit Maske	1F FF FF 00

Beispiele:

```
>ATCM 7F0
OK

>

>ATCM 1F FF 00 00
OK

>
```

In Verbindung mit dem Filter gibt ein 1-Bit in der Maske an, ob die ankommende Nachricht mit dem Filter verglichen werden soll. Wenn das Maskenbit 0 ist, wird dieses Bit als „ok“ angenommen.

Bitte darauf achten, daß durch die Anzahl der eingegebenen Zeichen bestimmt wird, ob die 11Bit oder die 29Bit Maske verändert werden soll. Für die 11Bit Maske müssen immer 3 Hex-Zeichen eingegeben für die 29Bit Maske müssen immer 8 Hex-Zeichen eingegeben werden.

Durch Eingabe von ATWS, ATZ oder ATD werden die Masken auf die Standardwerte zurückgesetzt,

>AT!00

Ausgabe der Seriennummer des DXM-Controllers. Alle DXM-Controller besitzen eine einzigartige nicht veränderbare Seriennummer. Sie kann mit diesem Befehl abgefragt werden.

Beispiel:

```
>AT!00
X123456789

>
```

>AT!01

Ausgabe des Controllertyps und der Bios-Versionsnummer. Da der Identifikations-String veränderbar ist, ist keine eindeutige Identifizierung des Controllertyps und der Bios-Version über den ATI Befehl möglich. Aus diesem Grund wurde dieser Befehl eingefügt, dessen Ausgabe unveränderlich ist und immer den richtigen Controller-Typ anzeigt.

Beispiel:

```
>AT!01
DXM1-10

>
```

Obiges Beispiel zeigt die Ausgabe eines DXM-1 Controllers mit Bios-Version 1.0

>AT!10

Messung der 12 Volt Fahrzeugspannung.

Beispiel:

```
>AT!10
12.5V

>
```

Weitere Befehle, speziell für die Änderung des Identifikationstextes und zum internen Test des DXM-Interfaces bei der Herstellung werden hier nicht aufgeführt. Interessenten können diese Informationen unter Angabe des Verwendungszwecks beim Entwickler der DXM-Controller per E-Mail anfordern.

Direkteingabe eines OBD2-Befehls

Zur Eingabe eines OBD2-Befehls werden nur die für diesen Befehl notwendigen Daten im Hexcode übergeben. Sollte zuvor noch keine Verbindung mit dem OBD2-Bus im Fahrzeug aufgebaut worden sein, wird dies bei Eingabe des ersten Befehls einmalig vorgenommen. Die verschiedenen Protokolle werden abhängig von der Protokoll Einstellung abgefragt (siehe Abschnitt „OBD2-Protokolle“

Sobald der Fahrzeugcontroller eine Antwort bei Anfrage mit einem der Protokolle liefert, wird der weitere Test abgebrochen und das erkannte Protokoll auch für alle weiteren Anfragen benutzt. Die Dauer bis zur Ausführung des ersten Befehls dauert somit maximal 2,5 Sekunden, bedingt durch die lange Antwortzeit beim Slow-Init. VPWM, PWM und CAN wird innerhalb 1 Sekunde erkannt.

Beispiel:
Ausgabe bei ISO9141 oder KWP2000 Connect.

```
>0100
SEARCHING...
41 00 E8 19 30 12

>

>0100
41 00 E8 19 30 12
```

Beispiel:
Ausgabe bei VPWM, PWM oder CAN Connect.

```
>0100
41 00 E8 19 30 12


>
```

Übersicht der AT-Befehle des DXM1

Allgemeine Befehle:

ATD	Alle Werte zurücksetzen
ATE0	Echo aus
ATE1	Echo ein (standard)
ATI	Identifizierungstext ausgeben
ATL0	Linefeed aus (standard)
ATL1	Linefeed ein
ATV	Variable Parameter anzeigen
ATWS	Warmstart
ATZ	Reset
AT!00	Ausgabe der Seriennummer
AT!01	Ausgabe der Controller-Kennung
AT!10	Messung der Fahrzeugspannung
ATSB x	Serielle Baudrate einstellen

OBD2 Befehle:

ATBD	Buffer Dump
ATB	Anzeige des Empfangsbuffers
ATDP	Aktuelles Protokoll im Klartext ausgeben
ATH0	Header aus (standard)
ATH1	Header ein
ATM0	Protokoll-Speicher aus (standard)
ATM1	Protokoll-Speicher ein
ATN	Aktuelles Protokoll als Nummer ausgeben
ATP	Aktuelle Protokollkonfiguration anzeigen
ATP x	Protokoll x voreinstellen
ATP Ax	Protokoll x voreinstellen, mit Auto-Suchfunktion
ATP0	Immer Protokoll Auto-Suchfunktion benutzen
ATSH xx yy zz	Header definieren
ATSR xx	RX-Filteradresse einstellen

ISO/KWP2000-spezifische Befehle:

ATSW xx	Wakeup Intervall-Zeit einstellen
ATWM xx yy zz aa [bb] [cc]	Wakup Message setzen
ATK	Anzeige des aktuellen Keywords
ATKW1	Automatische Erkennung des Protokolls einschalten (standard)
ATKW0	Automatische Erkennung des Protokolls ausschalten

CAN-spezifische Befehle:

ATCA0	CAN Autoformat aus
ATCA1	CAN Autoformat ein (standard)
ATCC0	CAN Flow Control aus
ATCC1	CAN Flow Control ein (standard)
ATCD xx	CAN Flow Delay setzen
ATCF xxx	CAN RX Message Filter 11 Bit
ATCF xxxxxxxx	CAN RX Message Filter 29 Bit
ATCI xxx	CAN TX Message Identifier 11 Bit
ATCI xxxxxxxx	CAN TX Message Identifier 29 Bit
ATCM xxx	CAN RX Message Mask 11 Bit
ATCM xxxxxxxx	CAN RX Message Mask 29 Bit

OBD2-Protokolle

Das DXM1-Modul unterstützt alle derzeit existierenden OBD2-Protokolle.

Modus	Protokoll
1	J-1850 PWM (41.6kBaud)
2	J-1850 VPWM (10.4kBaud)
3	ISO 9141-2 (5 Baud Init)
4	ISO 14230-4 KWP2000 (5 Baud Init)
5	ISO 14230-4 KWP2000 (Fast Init)
6	ISO 15765-4 CAN 11Bit-ID, 500 kBaud
7	ISO 15765-4 CAN 29Bit-ID, 500 kBaud
8	ISO 15765-4 CAN 11Bit-ID, 250 kBaud
9	ISO 15765-4 CAN 29Bit-ID, 250 kBaud

Im Auslieferzustand ist das DXM1-Modul so konfiguriert, daß automatisch alle Protokolle durchsucht werden, bis ein passendes gefunden ist `(ATP0)`. Sollte kein passendes Protokoll gefunden werden, wird folgender Text ausgegeben:

```
UNABLE TO CONNECT
```

Die Protokolle werden in folgender Reihenfolge getestet:

1	J-1850 PWM
2	J-1850 VPWM
3	5: KWP2000 (Fast Init)
4	3+4: ISO/KWP2000 (5 Baud Init)
5	6: CAN 11/500
6	7: CAN 29/500
7	8: CAN 11/250
8	9: CAN 29/250

Sobald ein passendes Protokoll gefunden wird, wird der Suchvorgang abgebrochen und das gefundene Protokoll ab sofort bei allen weiteren Befehlen benutzt. Das aktuelle Protokoll kann mit dem Befehl `ATDP` angezeigt werden.

```
>ATDP
ISO9141-4

>
```

Gleichzeitig wird das gefundene Protokoll als Suchstart benutzt, wenn die Verbindung zuvor mit `ATWS` oder `ATZ` geschlossen wurde. Dies kann mit dem Befehl `ATP` angezeigt werden.

```
>ATP
AUTO 3 = ISO9141-4

>
```

Ist die Protokoll Memory Funktion aktiviert `(ATM1)`, wird das aktuelle Protokoll im EEPROM des Moduls abgespeichert und bleibt auch nach Trennung der Stromversorgung erhalten.

Nach erneutem Anlegen der Stromversorgung wird der Inhalt des EEPROM ausgelesen und das gespeicherte Protokoll als erstes für den erneuten Suchvorgang benutzt. Wurde das gespeicherte Protokoll nicht gefunden, werden die Protokolle in der zuvor angegebenen Reihenfolge getestet, dabei wird jedoch das bereits als negativ getestete Protokoll übersprungen.

Soll der Suchvorgang gezielt mit einem gewünschten Protokoll beginnen, kann dies auch manuell eingegeben werden:

```
>ATPA5
AUTO 5 = ISO 14230-4 KWP2000 Fast Init

>
```

Hier wird der Suchvorgang mit dem Protokoll 5 begonnen. Wird das Protokoll nicht gefunden, werden die Protokolle in der Reihenfolge der Suchliste getestet.

Soll das DXM1-Modul immer nur ein bestimmtes Protokoll benutzen, kann dies durch folgenden Befehl festgelegt werden:

```
>ATP5
5 = ISO 14230-4 KWP2000 Fast Init

>
```

Hier wird nur noch das Protokoll 5 benutzt. Wird das Protokoll nicht gefunden, wird die Suche abgebrochen und `UNABLE TO CONNECT` oder eine andere Fehlermeldung ausgegeben. Ist die Protokoll Memory Funktion eingeschaltet, wird dieses Protokoll im EEPROM abgespeichert und auch nach Trennung und erneutem Anlegen der Stromversorgung wieder benutzt. Deshalb Achtung! Wenn keine Verbindung aufgebaut werden kann, zunächst mit `ATP` schauen, welches Protokoll voreingestellt ist.

Fehlermeldungen

Viele der im Folgenden aufgeführten Fehlermeldungen treten nur während des ersten Verbindungsaufbaus auf und erscheinen nur, wenn ein festes Protokoll mit `ATPx` voreingestellt ist. Im automatischen Protokoll-Suchmodus werden die Fehlermeldungen unterdrückt um mit dem Test des nächsten Protokolls fortzufahren.

?

Syntax-Fehler. Wird ausgegeben, wenn der AT-Befehl nicht existiert oder wenn zum Befehl gehörende Parameter fehlen oder falsch eingegeben wurden.

BUS BUSY

Tritt bei PWM und VPWM auf, wenn ein Befehl innerhalb der vorgeschriebenen Zeitspanne nicht gesendet werden konnte.

CAN ERROR

Fehler im CAN-Protokoll. Da das CAN-Protokoll relativ sicher gegen Übertragungsfehler auf dem Bus ist, wird diese Meldung nur ausgegeben, wenn die Verbindung zum CAN-Bus getrennt wird.

NO DATA

Wird ausgegeben, wenn ein OBD2-Befehl nicht innerhalb der vorgeschriebenen Zeitspanne beantwortet wurde, zum Beispiel weil er im Steuergerät nicht unterstützt wird.

UNABLE TO CONNECT

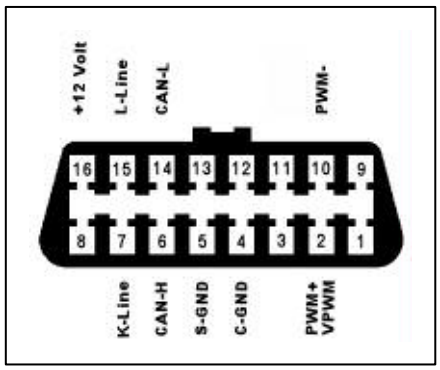
Erscheint, wenn keine Verbindung zu einem Steuergerät aufgebaut werden konnte.

ERROR (xx)

Ein unspezifizierter Fehler ist aufgetreten.

OBD2-MODUL MIT SERIELLER SCHNITTSTELLE (Version 1.0)

Die OBD2-Anschlußbuchse



Alle neu hergestellten Fahrzeuge müssen über solch eine 16-polige Buchse nach OBD2-Norm verfügen. Die Buchse muß sich in 1 Meter Umkreis vom Fahrersitz befinden. Obwohl die Bauform genormt ist, benutzen die verschiedenen Fahrzeughersteller verschiedene Übertragungsprotokolle.
Genormt ist jedoch, welches Protokoll an welchem Pin der Buchse anliegt. In den meisten Fahrzeugen sind nur die Kontakte der 16-poligen Buchse vorhanden, die auch Signale führen. Oft sind zusätzliche (nicht genormte) Kontakte an Pins vorhanden und auch mit dem Steuergerät verbunden, die zur internen Diagnose der Fahrzeughersteller dienen. Da es für diese Kontakte keine offiziellen Dokumentationen gibt, sollte man diese möglichst auch immer unbelegt lassen.

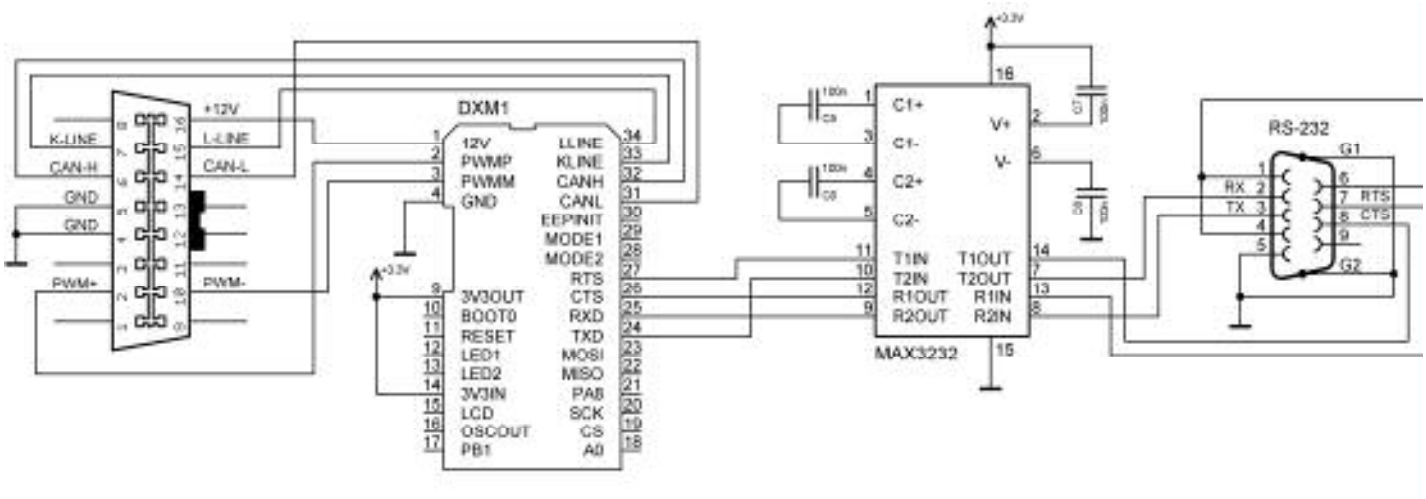
Folgende Kontakte sind immer vorhanden:

PIN 4	Fahrzeug-Chassis-Masse	(Pin 4 + 5 sind meist verbunden)
PIN 5	Signal-Masse	
PIN 16	Stromversorgung + 12 Volt	

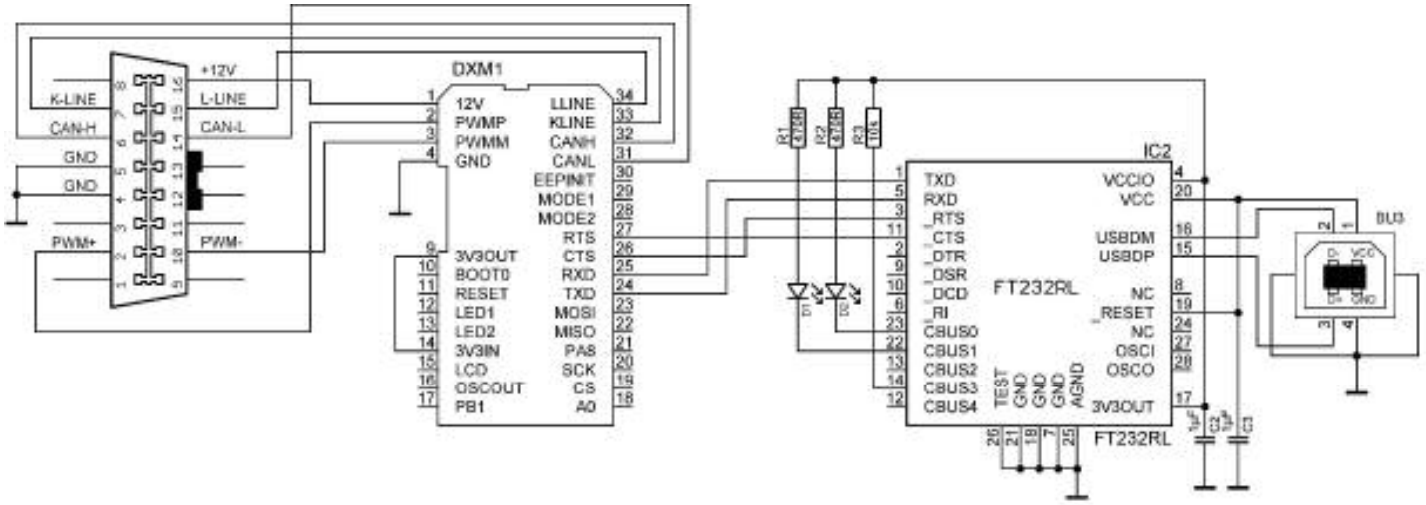
Folgende Kontakte sind je nach verwendetem Protokoll vorhanden und belegt:

PIN 7	ISO9141 / ISO14230 K-Line	(Ein- und Ausgang)
PIN 15	ISO9141 / ISO14230 L-Line	(Eingang)
PIN 2	J1850, PWM+	(Differentialsignal Bi-Direktional)
PIN 10	J1850, PWM-	(Differentialsignal Bi-Direktional)
PIN 2	J1850, VPWM	(Bi- Direktional)
PIN 6	ISO15765, CAN-H	(Differentialsignal Bi-Direktional)
PIN 14	ISO15765, CAN-L	(Differentialsignal Bi-Direktional)

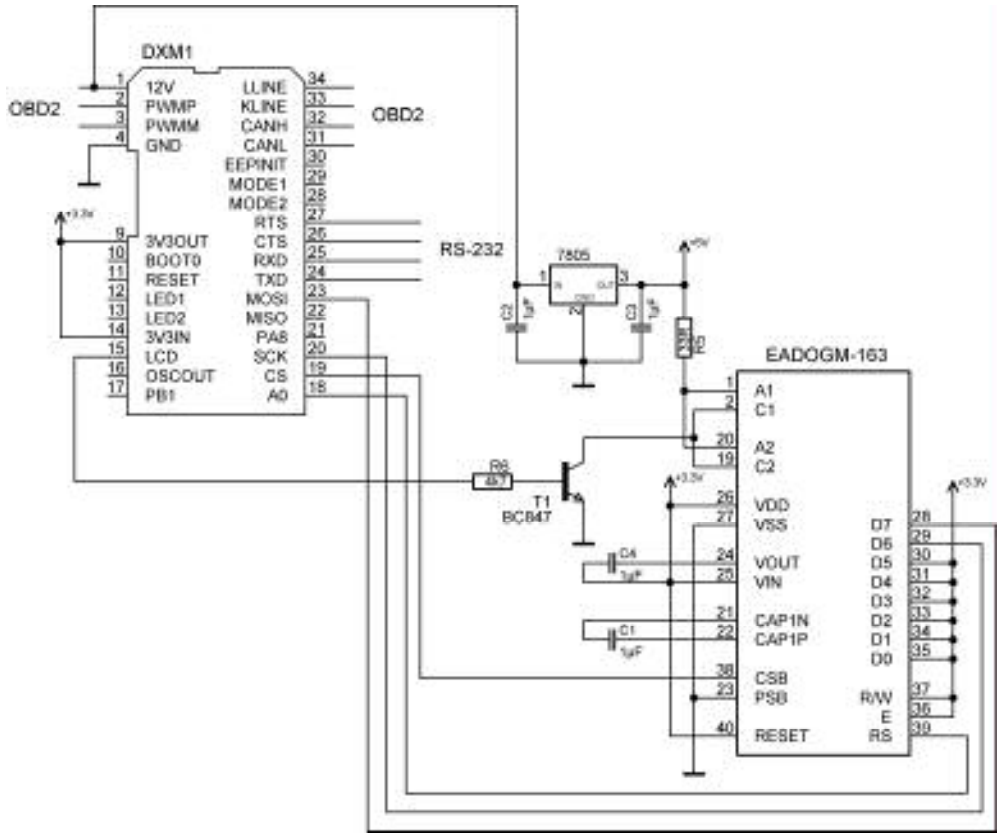
Schaltplan: DXM1 mit MAX3232-Seriell-Wandler



Schaltplan: DXM1 mit FT232R-USB-Wandler



Schaltplan: LC-Display am DXM1



Anschluss eines LC-Displays (3 x 16 Zeichen) an den SPI-Port des DXM1.

Über Transistor T1 wird die Hintergrundbeleuchtung des Displays eingeschaltet. Die Stromversorgung des Displays kann aus der 3,3 Volt Versorgung des DXM1 erfolgen. Für die Hintergrundbeleuchtung ist jedoch ein eigener Spannungsregler erforderlich, da die Stromaufnahme zu hoch für den 3,3 Volt Regler auf dem DXM1 ist.

Hinweise

© Erwin Reuß; Folker Stange. Nutzung und Weitergabe dieser Informationen auch Auszugsweise nur mit Erlaubnis der Copyright-Inhaber.

Änderungen am Bios und Befehlssatz des Controllers vorbehalten.

DIAMEX® ist eingetragenes Warenzeichen.

Alle Markennamen, Warenzeichen und eingetragenen Warenzeichen sind Eigentum Ihrer rechtmäßigen Eigentümer und dienen hier nur der Beschreibung.

Haftungshinweis

Der Hersteller übernimmt keine Haftung für Schäden die durch Anwendung der DIAMEX Interface und der Diagnose-Software entstehen könnten.

Kontakt, Forum, Software, Updates

http://www.dxm.obd-diag.net

info@diamex.de