Handbuch

H/S-System

Stand 09.2010

Handbuch H-/S-System 3. Auflage

Veröffentlich durch
DATCOM protelematik GmbH
D-63628 Bad Soden-Salmünster
Telefon: +49 (6056) 20972 – 0
Fax: +49 (6056) 20972 – 69
E-Mail: info@protelematik.de
Internet: http://www.protelematik.de/

Inhalt

Einleitung

In diesem Handbuch erfahren sie, wie das H-System und das S-System benutzt und programmiert werden kann. Um einen ersten Eindruck der beiden Systeme zu vermitteln, wird in Kapitel 1 ein einfaches Skript erstellt, auf ein System übertragen und ausgeführt.

Nachdem sie die Beschreibung der Programmiersprache, versehen mit vielen einfachen Beispielen, in Kapitel 2 verstanden haben, sind sie in der Lage komplexere Steuerungsaufgaben zu lösen. Damit sie die Besonderheiten im Unterschied zwischen H-System und S-System einfacher erkennen, wurde an den erforderlichen Stellen ein Hinweis eingefügt (leicht erkennbar durch „ℍ“ und „𝕊“). Ebenso sind häufig auftretende Probleme gut gekennzeichnet.

Im Folgenden Kapitel 3 wird auf die Kommunikation über GSM und GPRS eingegangen. Dabei wird beschrieben wie eine Verbindung hergestellt werden kann, um Dateien zu übertragen und wie die Kommunikation mit SMS und GPRS-Datenpaket zu einer Zentrale umgesetzt werden kann. Im Anschluss daran erklärt das Kapitel 4 die Erfassung der Position mittels GPS. Das GPS liefert nicht nur die aktuelle Position, sondern noch weitere Informationen, wie zum Beispiel die Uhrzeit, die in verschiedenen Variablen der Skriptsprache zur Verfügung stehen.

Kapitel 5 beschreibt die Bedienelemente, die an beiden Systemen genutzt werden können. Dabei zeigt dieses Kapitel wie Sie zum Beispiel einzelne Menüs anpassen und anzeigen können.

Das H-System kann, gegenüber dem S-System, durch diverse Hardware-Module erweitert werden, zum Beispiel durch ein Akku-/Lademodul für den Einsatz ohne eine feste Stromversorgung. Auf diese Module wird in Kapitel 6 eingegangen.

Das letzte Kapitel (Nummer 7) vermittelt ein Teil der Erfahrungen die wir mit unseren Systemen gesammelt haben. Diese Beispiele können als Anregung für ihre Ideen angesehen werden.

Wir wünschen ihnen viel Erfolg beim realisieren von Kundenwünsche und stehen ihnen bei Fragen gerne zur Verfügung.

Ihr **DATCOM protelematik** Team

Kapitel 1: Einführung

Dieses Kapitel vermittelt, wie ein Skript erstellt wird und wie es auf ein System übertragen werden kann. Eine Schritt-für-Schritt-Anweisung zeigt Ihnen die Konfiguration der erforderlichen Programme und begleitet Sie mit den wichtigsten Informationen. Sind alle Einstellungen gemacht und wurde das Beispiel erfolgreich übertragen, steht dem Erstellen von eignen Lösungen nichts mehr im Wege.

Erstellung eines Skripts

Das Skript kann in einen einfachen Editor, etwa Notepad, geschrieben werden und muss mit dem Namen „script.txt“ gespeichert werden. Dieser Name ist unbedingt einzuhalten, denn das H-/S-System sucht diese Datei und beginnt diese Zeilenweise zu interpretieren.

H Beachten Sie, dass der Dateiname klein geschrieben ist. Einige Betriebssysteme ändern den Dateiname automatisch so, das der erste Buchstabe groß geschrieben wird.

Als Vorlage für unser erstes Skript dient folgendes Beispiel:

```
set com1 stdio 115200
while (1)
  set output "Hallo Welt!"
wend
```

Die Aufgabe von diesem Skript ist es, den Text “Hallo Welt!” auf der seriellen Schnittstelle 1 (Bei dem H-System „Com1“ und bei dem S-System „Data1“) auszugeben. Auf die einzelnen Anweisungen wird in den später folgenden Kapiteln eingegangen.

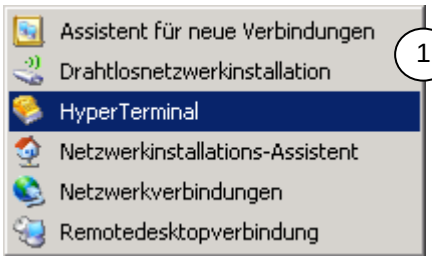
Übertragung im Terminalmodus

Ist das System mit der ersten seriellen Schnittstelle (H-System: Com1, S-System: Data1) zugänglich, so kann über spezielles Verbindungskabel[1] das Skript übertragen werden.

S Beachten Sie: Das Programmierkabel darf nicht in die Buchse mit der Aufschrift „MIC/SPK“ gesteckt werden. Wird das S-System so in Betrieb genommen, kann es zur Zerstörung des Systems kommen.

Für die Übertragung wird ein Terminalprogramm, wie etwa Hyperterminal, das im Lieferumfang von Microsoft Windows enthalten ist, benötigt. Im Folgenden wird gezeigt wie die Software eingerichtet werden muss:

[1] Programmierkabel für Skript (S-Box) ist mit der Artikelnummer 4004 erhältlich, für die H-Box mit der Artikelnummer 2359



Hyperterminal aus dem Menü "Kommunikation" im Zubehör-Menü starten.

Selbstdefinierter Name für diese Verbindung vergeben.

*Schnittstelle des PC wählen, die mit dem System verbunden ist. Anschließend mit **OK** bestätigen.*

.

*Stellen sie die Einstellungen wie im Bild dargestellt ein und bestätigen sie die Eingabe mit **OK***

Hyperterminal ist jetzt mit dem System verbunden. Durch betätigen der **Eingabetaste** kann das Anmelden an dem System ausgelöst werden.

H S Wird auf dem System bereits ein Skript abgearbeitet und nutzt das Skript ein Zubehörgerät oder, wie in unserem Beispiel, ein seriellen Dialog, so muss vor dem Betätigen der Eingabetaste **Strg+Z** gedrückt werden. Gegebenfalls müssen sie die Tastenkombination **Strg+Z** gefolgt von **Eingabetaste** mehrfach betätigen, da das System versucht das vermeintlich angeschlossene Endgerät über ein Protokoll anzusprechen.

Das System meldet sich mit folgender Meldung:

```
    ******************************************
    *                                        *
    *          D A T C O M  - Terminal       *
    *                                        *
    * Version: xxxx x.xx xxx  x xxxx  xx:xx:xx *
    *                                        *
    ******************************************

login as root, user or none

login:
```

Geben sie hier **root** ein und benutzen sie als Passwort **1111**. Bereiten sie das System für die Übertragung durch Eingabe des Befehls "upload" vor. Anschließend senden sie die Datei wie im Folgenden gezeigt:

*Wählen sie aus dem Menü **Übertragen** den Menüpunkt **Datei senden** aus.*

*Öffnen sie mit **Durchsuchen** das zu übertrage Skript und starten sie die Übertragung über **senden**.*

```
any more files to upload? (y/n) >
```

*Beantworten sie diese Frage mit **n** wenn sie keine weitere Datei übertragen wollen.*

Das Skript wird direkt nach der Übertragung abgearbeitet. Wenn sie das System mit **quit** verlassen, können sie den wiederholenden Text "Hallo Welt" sehen.

Um einfache Skriptfehler zu erkennen, kann im Terminalmodus der Befehl cli benutzt werden:

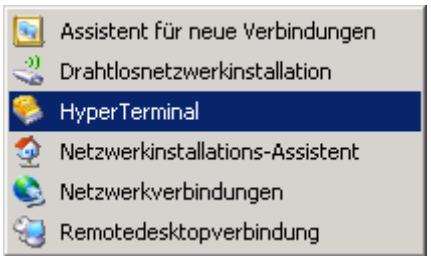
```
root > cli
No script error !
```

Wenn Fehler im Skript vorliegen erscheint anstelle von „No script error !“ eine entsprechende Fehlermeldung.

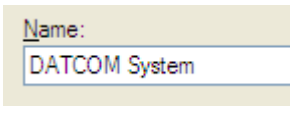
Beachten sie hierbei, dass nur durchlaufene Befehle auf Richtigkeit überprüft werden.

Übertragung im Terminalmodus über GSM

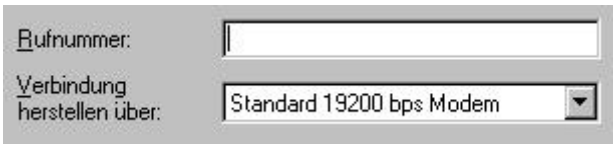
Ist das System nicht direkt zugänglich kann eine Verbindung auch über das GSM-Modul hergestellt werden. Dazu wird ein handelsübliches Modem benötigt. Die Installation des Modems muss dem entsprechenden Handbuch entnommen werden. Nach der Installation muss Hyper-Terminal wie folgt eingestellt werden:



Hyperterminal aus dem Menü "Kommunikation" im Zubehör-Menü starten.



Selbstdefinierter Name für diese Verbindung vergeben.



*Schnittstelle des PC wählen, an der das Modem angeschlossen ist. Anschließend mit **OK** bestätigen.*

*Dann **Ändern** betätigen und im nächsten Schritt **Konfigurieren** betätigen*

*Bei der Übertragungsrate muss **9600** ausgewählt und mit **OK** übernommen werden. Auch das nächste Fenster mit **OK** bestätigen. Im nachfolgenden Fenster auf **Wählen** klicken.*

Anschließend kann, genauso wie bei einer direkten Verbindung, das System bedient werden.

Kapitel 2: Skript Grundlagen

Um fehlerfreie Skripte zu schreiben, ist es notwendig die "Rechtsschreibung" und die "Grammatik" verstanden zu haben. Damit beschäftigt sich der erste Teil von diesem Kapitel. Im Anschluss daran folgt die Beschreibung der Variablen (Gleitkommazahlen, Ganze Zahlen, Zeichenketten) und allgemeinen Funktionen. Ein wichtiger Punkt ist hier auch die Kontrollstrukturen (if, while), mit deren Hilfe Wiederholungen und bedingtes Ausführen realisiert werden können.
Der zweite Teil des Kapitels wird unter anderem auf die interne Echt-Zeit-Uhr, die Datenbank und die Ein-/Ausgänge eingegangen.

Syntax

Der Interpreter akzeptiert nur Bezeichner die klein geschrieben sind. Generell gilt für Bezeichner: Es dürfen keine Zahlen am Anfang stehen und Parameter die Funktionen übergeben werden, müssen klar durch ein Leerzeichen voneinander getrennt sein (alternativ darf hier auch ein Tabulator benutzt werden). Zum Beispiel:

Ungültig	Gültig
0mem	mem0
send dattext"Test"	send dattext "Test"
SetPort0	setport0

Der Interpreter liest Zeilenweise das Skript, das bedeutet es darf in jeder Zeile nur ein Befehl oder eine Kontrollfunktion stehen. Ein anderer Aspekt der sich hieraus ergibt, ist das Skripte mit sehr vielen Zeilen eine lange Durchlaufzeit haben. Lösung zu diesem Problem schafft der Befehl run (Seite 24).

Kommentare

Ein Kommentar ist ein erläuternder Text, den der Programmierer seinem Programm als Verständnishilfe mitgibt. So ein Kommentar wird durch das voranstellen des Schlüsselwort rem eingeleitet (Ausnahme bildet die Gleitkommazahl, siehe Seite 12). Als Kommentar gilt der Bereich von rem bis zum Ende der Zeile:

```
rem *** Kommentar ***
rem setport2 öffnet die Tür
```

Variablen

Für Berechnungen oder zum Formulieren von Bedingungen stehen dem System sechs Arten von Variablen zur Verfügung:

- Ganzzahlvariablen (enthalten z.B. 5 oder 11)
- Extravariablen (enthält z.B. 145555 oder 279286)
- Gleitkommavariablen (enthalten z.B. 7,3 oder 39,4)
- Variablen für Zeichenketten (enthalten z.B. "Test" oder "Service")
- Zählvariablen (enthalten z.B. 5 oder 10)
- Bereichsvariablen (enthalten z.B. (50,1233|6,0453|51,3|6,23)) (Eine Beschreibung über Bereichsvariablen befindet sich auf Seite 60)
- Zeitgebervariablen (enthalten z.B. 10 oder 0)

Generell werden Zahlen in englischer Schreibweise angegeben, d.h. anstelle eines "," wird ein "." benutzt. Durch ein "h" kann eine hexadezimale Zahl angegeben werden.

Alle Variablen weisen einen Index auf (z.B. mem**x**, var**x**). Dieser Index kann durch eine Berechnung oder durch eine andere Variable ersetzt werden:

```
set mem(4+3) 10 rem Bedeutet: set mem7 10
set counter(mem1) 0
```

Dadurch kann zum Beispiel ein Feld (Tabelle) von Variablen verwaltet werden:

```
set mem(0 + (0*2)) 65 rem Spalte 0 Zeile 0 auf 65
set mem(1 + (2*2)) 65 rem Spalte 1 Zeile 2 auf 13
```

Alle Variablen (mit Ausnahme der Zeitgebervariablen) werden durch eine Batterie im Speicher, auch nach Abschalten des Systems, beibehalten. Es gibt auch die Möglichkeit alle Variablen (und auch die Datenbank) durch den Befehl backup in die Datei backup.bin schreiben zu lassen und diese durch restore wieder einzulesen:

```
rem Skriptanfang
restore
...
rem Skriptende
backup
```

Der Befehl backup darf nicht innerhalb einer Schleife benutzt werden, da ständiges schreiben in den Speicher zu einem erhöhten Verbrauch des Speichers führt. Nutzen sie den Befehl vorzugsweise am Ende des Skripts bevor das System in den Stand-By-Zustand wechselt.

Variablen können Ergebnisse aus mathematischen Berechnungen enthalten:

Skript	Beschreibung	Beispiel	Ergebnis
+	Addition	`set mem1 (2+3)`	5
-	Subtraktion	`set mem1 (5-1)`	4
*	Multiplikation	`set mem1 (2*3)`	6
/	Division	`var0 = 20 / 2`	10
sin	Sinus	`var0 = sin 3 * 2`	0,282240
cos	Kosinus	`var0 = cos 1 * 3`	1,620907
tan	Tangens	`var0 = tan 3`	-0,142547
sqr	Wuzel	`var0 = sqr 9`	3
arcsin	Arkusinus	`var0 = arcsin -1 * 2`	-3,141593
arctan	Arkustangens	`var0 = arctan 3`	1,2499046
arccos	Arkuskosinus	`var0 = arccos 0 * 2`	3,141593

Es existieren keinerlei Rechenregel und Prioritäten, so dass nur durch Einklammern die Auswertungsreihenfolge festgelegt wird.

Hier ist darauf zu achten wenn man eine Ganzzahlvariable (mem) benutzt, dann wird nur der ganzzahlige Anteil zurückgegeben. Soll eine Zahl gerundet werden so muss lediglich 0,5 hinzu addiert werden:

```
set mem0 (0.3 + 0.5)
```

Der Befehl *set* erwartet einen Namen einer Variable gefolgt von **einem** Wert, so würde die Zeile "set mem0 2 + 4 + 5" mem0 auf 2 setzen. Um dennoch eine vollständige Berechnung ausführen zu können, müssen Berechnungen in Klammer gesetzt werden: "set mem0 (2 + 4 + 5)"

Zu den mathematischen Funktionen kommen noch die logischen Verknüpfungen hinzu:

Skript	Beschreibung	Beispiel	Ergebnis
not	Negieren	`set mem0 (not 1)` `set mem0 (not 0)`	0 1

&|and	Und-Verknüpfung	set mem0 (0 & 0) set mem0 (0 & 1) set mem0 (1 & 0) set mem0 (1 and 1)	0 0 0 1
or	Oder-Verknüpfung	set mem0 (0 or 0) set mem0 (0 or 1) set mem0 (1 or 0) set mem0 (1 or 1)	0 1 1 1
xor	Exklusiv-Oder-Verknüpfung	set mem0 (0 xor 0) set mem0 (0 xor 1) set mem0 (1 xor 0) set mem0 (1 xor 1)	0 1 1 0

Ganzzahlvariablen

Die Ganzzahlvariablen werden mit mem**x** angegeben. Dabei steht x für eine Zahl zwischen 0 und 99. Somit können 100 Ganzzahlvariablen genutzt werden. Diese Variablen haben einen Wertebereich von -32768 bis 32767 und werden mit dem Befehl set auf einen bestimmt Wert gesetzt:

```
set mem0 17
set mem29 0
```

Wird einer Ganzzahlvariable eine Zeichenkette zugewiesen, so wird die Zeichenkette nach einer Zahl durchsucht:

```
set mem0 "Sollwert 50"
```

Die Angabe der Zahl kann auch in Hexadezimal erfolgen, dazu wird der Zahl einfach ein „h“ vorangestellt:

```
set mem0 h34 rem Hexadezimal 34 = Dezimal 52
```

Extravariablen

Extravariablen sind erweiterte Ganzzahlvariable mit einem größeren Wertebereich.
Die Extravariablen werden mit exvar**x** angegeben. Dabei steht x für eine Zahl zwischen 0 und 9. Somit können 10 Extravariablen genutzt werden. Diese Variablen haben einen Wertebereich von -2147483648 bis 2147483648 und werden mit dem Befehl set auf einen bestimmt Wert gesetzt:

```
set exvar0 17
set exvar9 0
```

Wird einer Ganzzahlvariable eine Zeichenkette zugewiesen, so wird die Zeichenkette nach einer Zahl durchsucht:

```
set extvar0 "Sollwert 50"
```

Die Angabe der Zahl kann auch in Hexadezimal erfolgen, dazu wird der Zahl einfach ein „h“ vorangestellt:

```
set mem0 h34 rem Hexadezimal 34 = Dezimal 52
```

Gleitkommavariablen

Die Zuweisung einer Gleitkommazahl geschieht durch ein Gleichheitszeichen:

```
var0 = 34.12
var0 = mem1 * sin 30 /12
```

Im Gegensatz zu den Mem-Variablen wird der Wert nicht durch set zugewiesen und es darf nach den Werten kein rem folgen. Diese Variable hat einen Wertebereich von ca. $1*10^{-37}$ bis $1*10^{38}$ und bei Berechnungen oder Ausgaben werden 6 Nachkommastellen benutzt

Variablen für Zeichenkette

Die Zeichenkette besteht aus aneinander gereihten Zeichen und können den Variablen mit dem Namen "string" zugewiesen werden:

```
set string0 "Test"
```

Der Text muss in zwei Anführungszeichen gesetzt werden und kann Sonderzeichen oder Leerzeichen enthalten. Das System bietet 40 Zeichenkettenvariablen mit einer maximalen Länge von 400 Zeichen an. Für Zeichenketten stehen dem System noch weitere Funktionen zur Verfügung:

Skript	Beschreibung	Beispiel	Ergebnis

add	Verbindet mehrere Variablen und fügt diese an einer Zeichenkette an.	`set mem1 25`
		`set string0 "Notruf"`
		`add string0 " 112 "`
		`add string0 mem1`
			string0: "Notruf 112 25"

replace	Ersetzt alle vorkommenden Zeichen durch ein anderes Zeichen.	`Set string0 "Alles ist super"` `replace "s" "z" string0`	string0: "Allez izt zuper"
chr	Erzeugt ein Zeichen mit entsprechendem ASCII-Wert.	`set string0 "Das ist ein: "` `add string0 chr 65`	string0: "Das ist ein: A"
fill	Füllt eine Zeichenkette mit einem Zeichen (fill zufüllend zeichen anzahl)	`set string0 "Test:"` `fill string0 "A" 5`	string0: "Test:AAAAA"
mid	Liefert einen Ausschnitt von einer Zeichenkette. (mid string start anzahl)	`set string0 "Kennung ABC35"` `set string1 mid string0 9 3`	string1: "ABC"
len	Liefert die Anzahl der Zeichen einer Zeichenkette.	`set string0 "Tür geöffnet"` `set mem0 len string0`	mem0: 12
byte2hex	Liefert den Hexa-dezimalwert einer 1Byte Variablen.	`set mem1 16` `set string0 byte2hex mem1`	string0: h10
word2hex	Liefert den Hexa-dezimalwert einer 2Byte Variablen.	`set mem2 34798` `set string0 byte2hex mem2`	string0: h87ee
long2hex	Liefert den Hexa-dezimalwert einer 4Byte Variablen.	`set exvar1 1646113119` `set string0 byte2hex exvar1`	string0: h621db15f

Zu der Funktion chr befindet sich im Anhang A (Seite 94) eine ASCII-Tabelle.

Zählvariablen

Es wurden spezielle Variablen für das Zählen eingefügt. Der Vorteil bei der Nutzung von Zählvariablen ist das die Verwendung schon aus der Bezeichnung der Variable hervorgeht und durch zwei zusätzlichen Funktionen die das erhöhen und vermindern der Variablen leichter erkennbar machen. Der Index geht von 0 bis 9, d.h. es gibt 10 Zählvariablen. Der Wertebereich geht von -32768 bis 32767. Die Zählvariablen werden wie folgt benutzt:

```
set counter0 0          rem Zählvariable setzen
set counter(mem1) 200   rem Zählvariable mit mem-Index setzen
inccounter0             rem Erhöhen der Zählvariable 0
deccounter0             rem Vermindern der Zählvariable 0
```

Zeitgebervariablen

Diese Variablen können auf einen Wert gesetzt werden, der jede Sekunde um 1 vermindert wird:

```
set timer0 15          rem Zeitgebervariable setzen
```

Der Wertebereich geht, genau wie bei Ganzzahlvariablen, von -32768 bis 32767.

Wird die vergangene Zeit benötigt, also ein hochzählende Zeitgebervariable, dann kann das Entwurfsmuster „Zeit messen“ auf Seite 89 genutzt werden.

Durch die Nutzung der Variable systics kann erfasst werden, wie viele Sekunden seit anlegen der Spannung am H-/S-System Vergang ist:

```
var0 = systics          rem Zeit erfassen
```

Kontrollstrukturen

Um Abhängigkeiten zu realisieren bietet das System zwei Kontrollstrukturen. Beide benutzen für die Entscheidung ob eine Wiederholung stattfindet oder ein Block ausgeführt wird, eine Bedingung. Diese Bedingung setzt sich aus mathematischen und logischen Ausdrücken zusammen und muss damit sie als Wahr gilt eine Zahl größer als 0 sein. Solche Bedingungen könnten so aussehen:

Vorbedingung	Bedingung	Ergebnis	Beschreibung
`set mem0 2`	`mem0=2`	Wahr	Die Bedingung ist ergibt 1 da 2=2 und somit ist diese Bedingung wahr.
`set mem0 2`	`mem0>34`	Falsch	mem0 ist nicht größer als 34
`var0 = 3.14`	`var0 < 4.345`	Wahr	
`set mem0 9`	`mem9 & 8`	Wahr	In binärer Schreibweise: 1001 (9) & 1000 (8) 1000 (8) Da das Ergebnis größer als 0 ist, ist diese Aussage wahr
`Eingang 1 aktiviert und:` `set mem0 0`	`in1 and (mem0=0)`	Wahr	Eingänge werden durch 0 oder 1 repräsentiert, daraus ergibt sich: 1 und (1) also Wahr

Bei dem vergleich von zwei Zeichenkette wird generell nur die Anzahl von Zeichen der rechten Zeichenkette mit der linken Zeichenkette verglichen:
if ("Hallo"="Hallo123“) → Ist falsch
if ("Hallo123"="Hallo") → Ist wahr

Wiederholungen (while...wend)

Die While-Schleife wiederholt einen Block von Anweisungen solange bis die angegebene Bedingung (für den Aufbau der Bedingung siehe Seite 11) nicht mehr erfüllt wird. Ist schon zu beginn die Bedingung nicht erfüllt, wird der Abweisungsblock ignoriert und das Skript interpretiert Anweisungen erst wieder ab dem Befehl wend. Zum Beispiel:

```
set counter0 0
while (counter0<20) rem Wiederholen solange Zähler kleiner 20
  set mem(counter0) 0 rem mem(counter0) auf 0 setzen
  inccounter0 rem Zähler erhöhen
wend
```

H S Der Befehl wend (nur das letzte wend bei mehrfachen Schleifen) setzt die Flanken der Eingänge (z.B. \in1) zurück (Siehe Eingänge und Ausgänge für H-System auf Seite 16 und auf Seite 19 für das S-System)

Mit der Variablen cycletime kann die Zeit, die benötigt wird um eine Schleife abzuarbeiten, ermittelt werden. Die Variable misst die Zeit zwischen den letzten beiden aufrufen:

```
cycletime
set mem1 cycletime
```

Bedingter Block (if...else...endif)

Durch die if-Anweisung kann die Ausführung von Blöcken von einer Bedingung (für den Aufbau siehe Seite 11) abhängig gemacht werden:

```
if (in1)
   set string1 "Tür ist zu"
else
   set string1 "Tür ist auf"
endif
```

Entsprechenden dem Eingang wird hier dem string1 der Zustand der Tür in Klartexte zugewiesen.

Der else-Teil ist optional und ist nicht zwingend erforderlich. Das obige Beispiel kann auch in zwei if-Anweisungen gepackt werden:

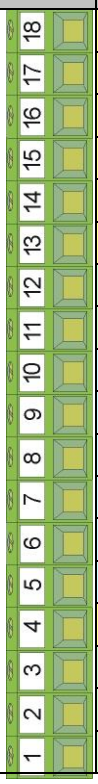
```
if (in1) rem Eingang 1 ist aktiv
   set string1 "Tür ist zu"
endif
if (not in1) rem Eingang 1 ist inaktiv
   set string1 "Tür ist auf"
endif
```

Eingänge und Ausgänge

Die Systeme bieten Ausgänge zum Steuern und Eingänge zum Erfassen von Zustanden. Bei der S-Box gibt es an der grünen Klemmleiste sieben Eingänge (davon können vier analoge Signale auswerten und zwei Frequenzen messen) und vier Ausgänge. Generell bietet die H-Box sechs Ausgänge, zwei analoge Eingänge und sieben digitale Eingänge, jedoch kann das Zusatzmodul "Ein/Ausgabe-Erweiterung" (siehe Seite 81) genutzt werden um weitere Ein-/ und Ausgänge zur realisieren.

Eingänge und Ausgänge der H-Box

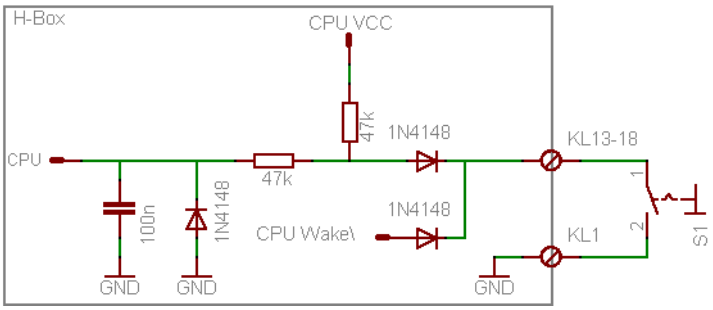
Leiste	Name	Technische Daten	Beschreibung
18	Digital Eingang 6	Nach GND	Beendet nach anlegen von GND den Stand-By-Modus des Systems. Tachosignal kann an Eingang 5 und Eingang 6 angelegt werden (1-8kHz).
17	Digital Eingang 5	Nach GND	
16	Digital Eingang 4	Nach GND	
15	Digital Eingang 3	Nach GND	
14	Digital Eingang 2	Nach GND	
13	Digital Eingang 1	Nach GND	
12	Analog Eingang 4	0-5V	Erfasst Werte von 0V bis 5V.
11	Analog Eingang 3	0-5V	
10	Digital Ausgang 6	max. 30mA	Alle Eingänge sind "open collector" Eingänge und werden nach GND geschaltet. Benutzen sie nur GND, die auch an Klemme 1 anliegt[2].
9	Digital Ausgang 5	max. 30mA	
8	Digital Ausgang 4	max. 30mA	
7	Digital Ausgang 3	max. 30mA	
6	Digital Ausgang 2	max. 30mA	
5	Digital Ausgang 1	max. 30mA	
4	On/Off (Eingang 7)		Siehe digitale Eingänge
3	UBsw	max. 750mA[3]	Beachten Sie di e Anmerkung
2	UB+	max. 28,8V	
1	GND		



Sind die angegebenen maximal Ströme für eine Anwendung zu niedrig, dann kann zur Verstärkung der H3A3-Adapter (Seite 82) benutzt werden.

[2] Durch anlegen einer fremden GND kann es zu Rückspannungen auf den anderen Eingängen kommen und führt unter Umständen zur Zerstörung der angeschlossenen Hardware.
[3] 750mA können ab dem Auslieferungsmonat 01/2006 genutzt werden. Alle bisherigen Systeme können nur mit 300mA belastet werden.

Die Beschaltung der Eingänge kann wie folgt aussehen:



Durch anlegen von GND an einem beliebigen Eingang wird das System wieder aktiviert falls sich das System im Stand-By-Modus befindet. Die Eingänge können durch die Funktion in wie folgt abgefragt werden:

```
if (in1=1)
   set string1 "Signal aktiv"
endif
if (in1=0)
   set string1 "Signal inaktiv"
endif
if (/in1)
   set string1 "Signal von aktiv nach inaktiv gewechselt"
endif
if (\in1)
   set string1 "Signal von inaktiv nach aktiv gewechselt"
endif
```

Treten Veränderungen an den Eingängen auf, dann werden diese Änderungen zu erst zwischengespeichert und bei erreichen des letzen wend-Befehls in die Variablen /in und \in geschrieben. Alternativ kann auch über die Anweisung clredge die neuen Zustände auf /in und \in geschrieben werden:

```
while (swofftime<20)
  run "in.txt"
  set timer0 10

  while (timer0)
    clredge
    run "in.txt"
  wend
wend
```

HS

Soll eine Schleife realisiert werden, in der Eingänge verarbeitet werden, so muss folgendes geachtet werden:

- Verarbeiten sie die Eingänge vor Eintritt in die Schleife
- clredge muss innerhalb der Schleife **vor** dem verarbeiten der Eingänge stehen

Der digitale Eingang sieben entspricht dem „On/Off“ Eingang (Zündung) und kann entsprechend dem oben genannten Schema genutzt werden:

```
if (in7=1) rem Zündung ist eingeschaltet
   setport1
endif
```

Um alle Eingänge auf einmal zu erfassen kann die Variable ina benutzt werden:

```
set mem0 ina
while (1=1)
  if (not (ina=mem0)) rem Haben sich die Eingänge geändert?
    rem *** Auf Änderungen reagieren ***
  endif
wend
```

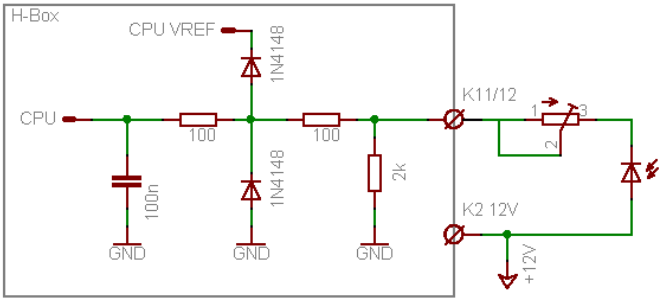
Die beiden digitalen Eingänge 5 und 6 können durch einen gesonderten Befehl Frequenzen messen:

```
if (frq5>5) rem Wenn die Frequenz größer als 5Mhz
   set mem1 1
endif
```

Soll durch die Frequenz z.B. ein Tachosignal erfasst werden, so können alternativ die beiden Befehle prescaler und pulsecntr genutzt werden:

```
set prescaler5 20 rem Tacho: 1m bedeutet hier 20 Impulse
set pulsecntr5 0  rem pulsecntr=frq/prescaler
while (1=1)
   if (pulsecntr5>2000) rem Mehr als 2km gefahren
     rem *** Meldung ausgeben ***
   endif
wend
```

Die analogen Eingänge können wie im folgenden Beispiel beschaltet werden:



Erfasst werden kann der analoge Wert durch folgende Befehle:

```
set mem1 adc4 rem Analog Eingang (ohne Entprellung)
set mem2 ain4 rem Analog Eingang (mit Entprellung)
```

Die Variable ainX wurde mit einer Trägheit versehen um z.B. Füllstände in einem Fahrzeug messen zu können.

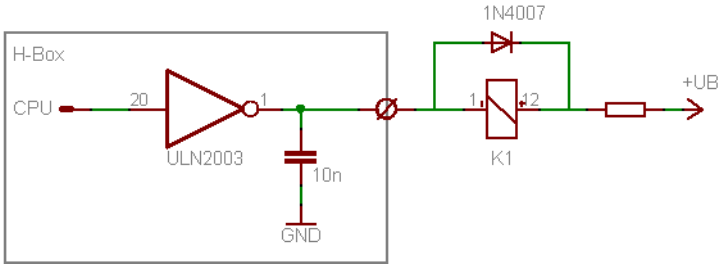
Durch abfragen der Variable ain6 kann die interne Spannung des Systems gemessen werden:

```
var0 = ain6 * 0.012 rem Ermittelt die interne Spannung
```

Dabei ist zu beachten, dass der Faktor 0,012 bei jedem System neu errechnet werden muss, da sich bereits kleinste Unterschiede in der Hardware sich auf die Erfassung auswirken. Sie benötigen für die Berechnung die gemessene Spannung (Vmes) und den momentanen Wert von ain6 (Awert, kann im Dialogbetrieb [Seite 7] durch die Anweisung *tele* abgefragt werden). Die Formel lautet wie folgt:

$$\text{Faktor} = \frac{\text{Vmes}}{\text{AWert}}$$

Die Ausgänge können wie folgt beschaltet werden:



Wird ein Relais benutzt, so muss eine Diode parallel zum Relais (Sperrrichtung siehe Schaltung) eingebaut werden, damit die Induktionsspannung beim abschaltet des Realais abgebaut wird.

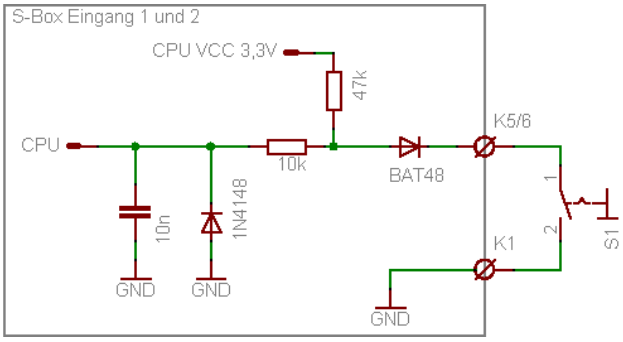
Für die Ausgänge existieren die beiden Anweisungen setport und clrport. Die Anweisung setport legt auf einen angegebenen Ausgang GND. Durch die Anweisung setport wird GND durchgeschaltet, entsprechend wird mit der Anweisung clrport der Ausgang geöffnet (Open-Collector):

```
setport1 rem Setzt den Ausgang 1
clrport2 rem Setzt den Ausgang 2 zurück
```

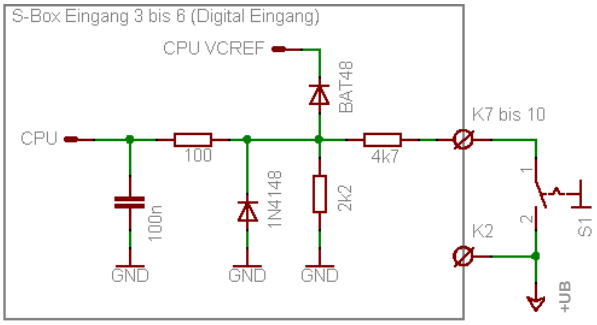
Eingänge und Ausgänge der S-Box

Leiste	Name	Technische Daten	Beschreibung
14	Digital Ausgang 4	max. 30mA	„open collector“ (nach GND geschaltet)[4]. Benutzen sie nur GND, die auch an Klemme 1 anliegt.
13	Digital Ausgang 3	max. 30mA	
12	Digital Ausgang 2	max. 30mA	
11	Digital Ausgang 1	max. 30mA	
10	A/D Eingang 6	Nach +UB	Die Eingänge von 3 bis 6 können als analoge oder digitale Eingänge genutzt werden. Werden die Eingänge als digitale Eingänge verwendet, muss nach +UB geschaltet werden. Bei Verwendung als analoge Eingänge, muss eine Spannung von 0V bis 5V angelegt werden.
9	A/D Eingang 5	Nach +UB	
8	A/D Eingang 4	Nach +UB	
7	A/D Eingang 3	Nach +UB	
6	Eingang 2	Nach GND	Digitaler Eingang. Tachosignal kann an Eingang 1 und Eingang 2 angelegt werden (1-1,2kHz).
5	Eingang 1	Nach GND	
4	Zündung (Eingang 7)	Nach +UB	Reaktivierung wenn im Stand-By-Modus
3	+UBsw	max. 1A	
2	+UB	max. 28,8V	
1	GND		

Die Beschaltung der beiden Eingänge 1 und 2 kann wie folgt aussehen:



Für die Nutzung der Eingänge 3 bis 6 als digitale Eingänge gilt folgende Beschaltung:



Die Eingänge können durch die Funktion in wie folgt abgefragt werden

```
if (in1=1)
   set string1 "Signal aktiv"
endif
```

[4] Durch anlegen einer fremden GND kann es zu Rückspannungen auf den anderen Eingängen kommen und führt unter Umständen zur Zerstörung der angeschlossenen Hardware.

```
if (in1=0)
   set string1 "Signal inaktiv"
endif
if (/in1)
   set string1 "Signal von aktiv nach inaktiv gewechselt"
endif
if (\in1)
   set string1 "Signal von inaktiv nach aktiv gewechselt"
endif
```

Treten Veränderungen an den Eingängen auf, dann werden diese Änderungen zu erst zwischengespeichert und bei erreichen des letzen wend-Befehls in die Variablen /in und \in geschrieben. Alternativ kann auch über die Anweisung clredge die neuen Zustände auf /in und \in geschrieben werden:

```
while (swofftime<20)
  run "in.txt"
  set timer0 10

  while (timer0)
    clredge
    run "in.txt"
  wend
wend
```

§ Soll eine Schleife realisiert werden, in der Eingänge verarbeitet werden, so muss folgendes geachtet werden:

- Verarbeiten sie die Eingänge vor Eintritt in die Schleife
- clredge muss innerhalb der Schleife **vor** dem verarbeiten der Eingänge stehen

Der digitale Eingang sieben entspricht dem „Ignition“ Eingang (Zündung) und kann entsprechend dem oben genannten Schema genutzt werden.

Um alle Eingänge auf einmal zu erfassen kann die Variable ina benutzt werden:

```
set mem0 ina
while (1=1)
  if (not (ina=mem0)) rem Haben sich die Eingänge geändert?
    rem *** Auf Änderungen reagieren ***
  endif
wend
```

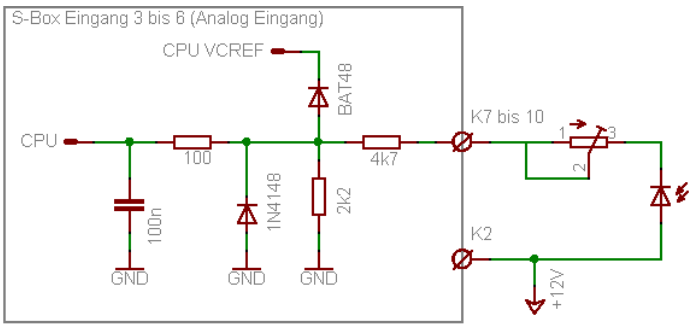
Die beiden digitalen Eingänge 1 und 2 können durch einen gesonderten Befehl Frequenzen messen:

```
if (frq5>5) rem Frq 5 = In 1 = gemessene Frequenz an In 1
   set mem1 1
endif
if (frq6>10) rem Frq 6 = In 2 = gemessene Frequenz an In 2
   set mem1 1
endif
```

Soll durch die Frequenz z.B. ein Tachosignal erfasst werden, so können alternativ die beiden Befehle prescaler und pulsecntr genutzt werden:

```
set prescaler1 20 rem Tacho: 1m bedeutet hier 20 Impulse
set pulsecntr1 0  rem pulsecntr=frq/prescaler
while (1=1)
   if (pulsecntr1>2000) rem Mehr als 2km gefahren
     rem *** Meldung ausgeben ***
   endif
wend
```

Die Eingänge 3 bis 6 können ebenfalls als analoge Eingänge verwendet werden und können wie folgt beschaltet werden:



Erfasst werden kann der analoge Eingang durch folgende Befehle:

```
set mem1  adc4 rem Analog Eingang (ohne Entprellung)
set mem2  ain4 rem Analog Eingang (mit Entprellung)
```

Die Variable ainX wurde mit einer Trägheit versehen um z.B. Füllstände in einem Fahrzeug messen zu können.

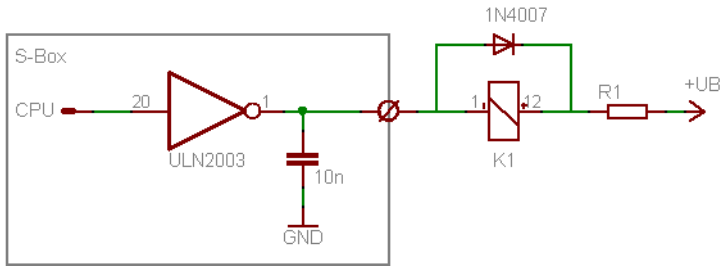
Durch abfragen der Variable ain2 kann die interne Spannung des Systems gemessen werden:

```
var0 = ain2 * 0.012 rem Ermittelt die interne Spannung
```

Dabei ist zu beachten, dass der Faktor 0,012 bei jedem System neu errechnet werden muss, da sich kleinste Unterschiede in der Hardware sich auf die Erfassung auswirken. Sie benötigen für die Berechnung die gemessene Spannung (Vmes) und den momentanen Wert von ain2 (Awert, kann im Dialogbetrieb [Seite 7] durch die Anweisung *tele* abgefragt werden). Die Formel lautet wie folgt:

$$\text{Faktor} = \frac{\text{Vmes}}{\text{AWert}}$$

Die Ausgänge können wie folgt beschaltet werden:



Wird ein Relais benutzt, so muss eine Diode parallel zum Relais (Sperrrichtung siehe Schaltung) eingebaut werden, damit die Induktionsspannung beim abschaltet des Realais abgebaut wird.

Für die Ausgänge existieren die beiden Anweisungen setport und clrport. Die Anweisung setport legt auf einen angegebenen Ausgang GND. Durch die Anweisung setport wird GND durchgeschaltet, entsprechend wird mit der Anweisung clrport der Ausgang geöffnet (Open-Collector):

```
setport1 rem Setzt den Ausgang 1
clrport2 rem Setzt den Ausgang 2 zurück
```

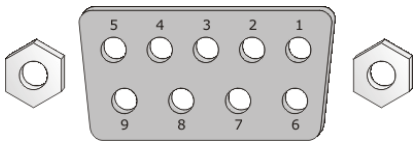
Serielle Schnittstellen

Das H-System besitzt eine Schnittstelle auf die im Skript mit der Variable „com1“ zugegriffen werden kann. Um dem System bekannt zugeben, welches Endgerät sich an dieser Schnittstelle befindet wird einfach der Variable „com1“ das entsprechende Gerät zugewiesen:

```
set com1 stdio 9600 rem Com1/Data1 auf 9600 Baud setzen
set com1 tabo3      rem Tabo3/4
```

Für das H-System steht ein Erweiterungsmodul zu Verfügung, durch das zwei weitere Schnittstellen realisiert werden können (siehe Seite 83).

COM 1

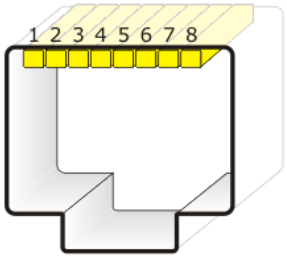


1	+UBSw	max. 750mA
2	Tx (PC→H-System)	
3	Rx (PC→H-System)	
4	Reserviert	
5	GND	
6	Reserviert	
7	+UB	
8	On/Off (Zündung)	
9	Progmode	

Das S-System besitzt zwei Schnittstellen um Endgeräte anzusprechen. Dabei bezeichnet die Variable „com1“ die erste Schnittstelle und die Variable „com3“ die zweite Schnittstelle

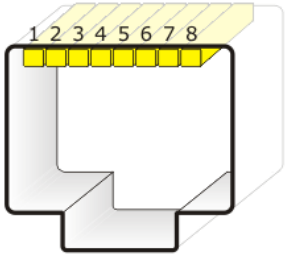
Beachten Sie, dass die zweite Schnittstelle nicht, wie erwartend, com2 sondern com3 als Zugriffsvariable verwendet.

Data 1 (Com 1)



1	+UB	Bordspannung
2	GND	
3	Zündung	
4	Progmode	
5	Service	
6	Tx (Gerät→H-System)	
7	Rx (Gerät →H-System)	
8	+UBSw	max. 750mA

Data 2 (Com 3)



1	+UB	Bordspannung
2	GND	
3	Zündung	
4	RS485A	
5	RS485B	
6	Tx (PC→H-System)	
7	Rx (PC→H-System)	
8	+UBSw	max. 750mA

Unterbrechungen und Skriptmodule

In einigen Fällen muss eine festgelegte Zeit gewartet werden bis z.B. Messwerte erfasst werden können. Dazu bieten die beiden Systeme den Befehl wait:

```
setport0 rem Ausgang aktivieren
wait 250 rem 250ms warten
clrport0 rem Ausgang wieder deaktivieren
```

Dadurch können Impulse von wenigen Millisekunden realisiert werden.

Für eine Teststellung oder wenn ein Skript die seriellen Schnittstellen umschalten soll, muss ein Neustart der Box durchgeführt werden. Dazu kann der Befehl reboot an einer beliebigen Stelle im Skript genutzt werden:

```
if (mem1=1)
  set com1 tabo3
else
  set com1 dt1
endif

if (/in1)
  set mem1 1
  reboot
endif
if (/in2)
  set mem1 0
  reboot
endif
```

Der Ablauf ist nicht nur auf die Datei „script.txt“ beschränkt. Es können weitere Dateien auf dem System abgelegt werden, die durch eine Skriptanweisung (run) Ausgeführt werden können. Dafür wird die Ausführung der Datei „script.txt“ unterbrochen, die Zeilen der angegeben Datei interpretiert und anschließend an der unterbrochenen Stelle der Datei „script.txt“ weiter gearbeitet. Dazu ein Beispiel:

```
rem *** script.txt ***
while (1=1)
  if (in2=1) rem Anhänger wurde angehängt
    run "contain.txt"
  else
    run "nocontai.txt"
  endif
wend
```

```
rem *** contain.txt ***
if (in1=1) rem Wenn Türöffner betätig
  setport1 rem Tür öffnen
else
  clrport1 rem Tür schließen
endif
```

```
rem *** nocontai.txt ***
if (in1=1) rem Wenn Türöffner betätig
  rem *** Meldung: Alarm ****
endif
```

Echtzeituhr

Die Systeme verfügen über eine interne Echtzeituhr mit Kalenderfunktionen. Dadurch können z.B. Meldungen mit einem Zeitstempel versehen werden oder Steueraktionen zu einem festgelegten Zeitpunkt ausgeführt werden. Dabei stehen diese Funktionen nicht nur dem Skript-Interpreter zur Verfügung sondern werden auch im System selbst genutzt. So ist zum Beispiel das Datum einer Datei auf dem System abhängig von der Uhr. Um die Echtzeituhr auch für Systeme zu nutzen, die Länderübergreifend eingesetzt werden, kann die Uhrzeit auch vom GPS-Empfänger genutzt werden (Seite 61).

Zwei Zeichenketten liefern das aktuelle Datum und die Uhrzeit:

```
set com1 stdio 115200
while (1=1)
```

```
    rem Wochentag Tag.Monat.Jahr (ww tt.mm.jj)
    set output strdate
    add chr 13 chr 10

    rem Stunde:Minuten:Sekunden(hh:mm:ss)
    set output strtime
    add chr 13 chr 10
  wend
```

Werden nur Teile des Datums, zum Beispiel nur die Minuten, benötigt, so können auch die folgenden Variablen genutzt werden:

```
set mem1 date  rem Tag (1 bis 31)
set mem1 month rem Monat (1 bis 12)
set mem1 year  rem Jahr (jjjj)
set mem1 hour  rem Stunde (0 bis 23)
set mem1 min   rem Minute (0 bis 59)
set mem1 sec   rem Sekunde (0 bis 59)
set mem1 day   rem Wochentag (0=Montag bis 6=Sonntag)
```

Diese Variablen können auch Gesetzt werden, zum Beispiel:

```
set mem1 12
set date mem1  rem Tag auf den 12. setzen
set month 2    rem Monat auf Februar
```

Befindet sich das System im Stand-By-Modus (siehe 28) kann ein Termin für das Einschalten gesetzt werden:

```
set houralarm 15  rem Stunde des Einschaltens
set minalarm 30   rem Minute des Einschaltens
set secalarm 10   rem Sekunde des Einschaltens
set dayalarm 2    rem Tag des Einschaltens (siehe "day")
```

Werden Angaben weggelassen so werden nur die angegeben Werte beachtet, d.h. wenn z.B. minalarm auf 15 gesetzt wird, dann wird immer wenn die Minutenzahl 15 erreicht das System eingeschaltet.

Eine weitere Möglichkeit ist die Nutzung der Variable clockmode:

```
set clockmode 1 rem Im Minutentakt einschalten
set clockmode 2 rem Im Stundentakt einschalten
set clockmode 3 rem Im Tagestakt einschalten
```

Datenbank

Die Datenbank des S- und H-Systems bietet eine einfach Möglichkeit, um mit einer höheren Anzahl von Datensätzen umzugehen. Als Grundfunktionen wurde das Einfügen, Verändern, Sortieren und Löschen realisiert. Die Datenbank kann zum Beispiel zur Speicherung von Aufträge und Meldungen, die durch die Zentrale übermittelt wurden, genutzt werden. Eine Selektierung der Datensätze kann auch durch komplexe Bedingungen gemacht werden und durch eine anschließende Sortierung in die richtige Reihenfolge gebracht werden.

Werden Daten in die Datenbank geschrieben, so könnte die Tabelle folgendes Aussehen aufweisen:

	Spalte 1	**Spalte 2**	**Spalte 3**	**Spalte ...**
Zeile 1	Meldung	Maier	ID 12	...
Zeile 2	Auftrag	Schäfer	ID 9765	...
Zeile 3	Auftrag	Schneider	ID 538	...
Zeile ...	...	...	...	...

Die einzelne Spalten werden durch Semikolon getrennt. Datensätze werden durch den Befehl insert in die Datenbank eingefügt:

```
delete all
insert "Meldung;Maier;ID 12"
insert "Auftrag;Schäfer;ID 9765"
insert "Auftrag;Schneider;ID 538"
```

Die Zeichenkette darf bis zu 400 Zeichen lang sein und beliebig viele Spalten beinhalten. Beachten Sie hierbei, dass bei einen späteren Ändern der Spalteninhalte nur maximal so viele Zeichen gesetzt werden können, wie die ursprüngliche Anzahl der Zeichen (im Fall von „Auftrag“ können nur Zeichenketten mit einer Länge von 7 Zeichen zugewiesen werden). Werden Zeichenketten mit mehr Zeichen übergeben, so wird diese Abgeschnitten.

Tipp: Sollen die Zellen später geändert werden, dann einfach beim Einfügen genügend Leerzeichen anhängen.

Bevor Daten in die Datenbank geschrieben werden können oder wenn Zeilen aus der Tabelle nicht mehr benötigt werden, so können diese über folgende Anweisungen gelöscht bzw. die Datenbank initialisiert werden:

```
delete rem Löscht alle ausgewählte Zeilen
delete row rem Löscht nur die aktuelle Zeile
delete all rem Löscht die gesamte Tabelle
```

Die Datenbank nutzt intern eine Verweisvariable um sich zu merken, welche Zeile aktuell aktiv ist. Um auf die einzelnen Zellen zuzugreifen werden die Variablen mit der Bezeichnung cell genutzt:

```
rem Wurden gerade die oben genannten Daten in die Datenbank
rem eingefügt, so steht die Verweisvariable auf der ersten
rem Zeile
set string0 cell3 rem string0 enthält "ID 12”
set mem0 len cell2 rem mem0 = 5, da "Maier” 5 Zeichen hat
set cell3 "id0" rem Setz neue Zeichenkette
```

Die Länge der zu setzenden Zeichenkette darf nicht länger sein als der originale Inhalt der Zelle, ansonsten wird die neue Zeichenkette abgeschnitten.

Um die interne Verweisvariable auf die nächste oder vorherige Zeile zu setzen können folgende Anweisungen benutzt werden:

```
next rem Wählt die nächste Zeile (im Beispiel Zeile2)
prev rem Wählt die vorherige Zeile (im Beispiel Zeile1)
next rem Nochmals die nächste Zeile (im Beispiel Zeile2)
```

Jetzt steht die interne Verweisvariable auf der Zeile 2 und zum Beispiel durch Zugriff auf cell2 erhält man die Zeichenkette „Schäfer“.

Um zu prüfen, ob das Ende oder der Anfang der Datenbank erreicht ist, können diese beiden Variablen genutzt werden:

```
if (bot) rem Am Anfang der Datenbank
  rem *** Taster für "Zurück" ausblenden
endif
if (eot) rem Am Ende der Datenbank
  rem *** Taster für "Vorwärts" ausblenden
endif
```

Die Variable dbindex liefert die aktuelle Zeilennummer:

```
if (dbindex<5) rem Befinden wir uns vor Zeile 5?
  rem *** Meldung das die Top 5 verlassen wurden
endif
```

Sind mehrere Verschiedene Informationen in der Datenbank enthalten, so kann durch den Befehl select mehrere Zeilen durch Angabe von verschiedenen Kriterien ausgewählt werden. Alle Zeilen die nicht mit den Kriterien übereinstimmen werden bis zu einem neuem select-Befehl verborgen gehalten. Innerhalb der Bedingung für die Auswahl können reguläre Ausdrücke genutzt werden und die cell-Variablen. Dabei sind die cell-Variablen die Angaben der zu testenden Zeile. Ein Beispiel mit den bereits oben eingefügten Zeilen[5]:

```
select (cell1="Auftrag") rem Nur Aufträge auswählen
```

Die Tabelle sieht dann wie folgt aus (nicht ausgewählte Zeilen sind nicht nutzbar):

	Spalte 1	Spalte 2	Spalte 3
	Meldung	Maier	ID 12
Zeile 1	Auftrag	Schäfer	ID 9765
Zeile 2	Auftrag	Schneider	ID 538

Soll die ganze Tabelle wieder genutzt werden, muss der select-Befehl mit einer immer wahren Bedingung ausgeführt werden:

```
select (1) rem Alle Zeilen auswählen
```

Die Anzahl wie viele Zeilen ausgewählt wurden, kann durch die Variable selected ermittelt werden. So kann zum Beispiel die Aussage gemacht werden, welcher Auftrag von wie vielen gerade angezeigt wird:

```
set string0 dbindex rem Aktuelle Zeile
add string0 " von " selected
```

Eine Sortierung der Tabelle kann durch die Anweisung sort erreicht werden:

```
sort cell1
sort cell2
```

CSV Datenbank (Nur lesen)

Um auf größere Datenmenge aus dem Skript heraus zugreifen zu können, wurde die Möglichkeit geschaffen, selektiv Daten aus einer Datei zu lesen, die im CSV-Format (Spalten sind durch Semikolon getrennt) vorliegt. Zum Beispiel liegt die Datei „personen.csv“ mit folgenden Daten auf dem System:

```
Peter;Fahrer;id21;Gruppe 2
Frank;Service;id39;Gruppe1
```

[5] Beachten sie das Bedingungen die aus mehreren Verknüpfungen bestehen, immer eine umgebene Klammer beinhalt müssen

Timo;Fahrer;id4;Gruppe 2
Markus;Anwalt;id60;Gruppe 2

Bevor die Daten genutzt werden können, muss die Datei geöffnet werden:

```
csvopen "personen.csv" rem Öffnet die Datei "personen.csv"
```

Nachdem die Datei geöffnet wurde, kann ein Filter genutzt werden um eine Selektion der zu lesenden Zeilen vorzunehmen:

```
csvfilter "an" rem Filtern nach "an"
```

Um eine Auszuwahl zu treffen muß der Befehl "csvnext" ausgeführt werden.

Hier ist zu beachten, dass die Zeichenkette die an den Filter übergeben wird, nur aus Kleinbuchstaben besteht. Die Filterung wird auf alle Spalten angewendet, z.B. nach "an" gefiltert ergibt:

Frank;Service;id39;Gruppe1
Markus;Anwalt;id60;Gruppe 2

Ein Zugriff auf die aktuelle Zeile erfolgt, genau wie bei der Datenbank, über die Variablen cell**x**:

```
set string0 cell1 rem Legt die erste Zelle in string0
```

Die Zellen können nur gelesen, nicht aber geschrieben werden.
Die nächste Zeile kann durch die folgende Anweisung ausgewählt werden:

```
csvnext rem Selektiert die nächste Zeile
```

Das Ende der CSV Datenbank wird durch leere Zellen angezeigt. Ist das Ende der CSV Datenbank erreicht so muss die Datenbank wieder geschlossen werden:

```
csvclose rem Datei schließen
```

Ein- und Abschaltung H/S-Box

Um eine Abschaltung des Systems durchzuführen, wurde die Variable swofftime und swoffsec eingeführt. Sind alle Bedingungen wie im Beispiel erfüllt, wurde kein eingehender Anruf erkannt und kein Neustart des Systems durchgeführt wird die Variable swofftime jede Minute und die Variable swoffsec jede Sekunde erhöht. Ist der angegebene Parameter erreicht, schaltet sich die Box ab.

Durch Aktivierung der Eingänge D In1-D In6, „Ignition“-Eingang sowie durch Nutzung der Echtzeituhr (siehe Seite 25) kann die Box erneut aktiviert werden.

Durch Aktivierung des „Ignition“-Eingangs (Seite 20) und durch Nutzung der Echtzeituhr (siehe Seite 25) kann die Box erneut aktiviert werden.

```
while (swofftime<16)
  rem ** Anweisungsblock **
wend
rem ** Diese Anweisungen werden erst nach einer Leerlauf-
rem zeit von 15 Minuten abgearbeitet
```

Energiesparfunktionen

Die beiden Systeme können durch eine Batterie oder einem Akku unabhängig vom Stromnetz betrieben werden. Um die Akkuleistung effizient nutzen zu können können einzelne Komponenten in der Box deaktiviert werden.

Sind an dem System zusätzliche Geräte verbaut (zum Beispiel Kartenleser oder DATCOM 640) benötigt das komplette System wesentlich mehr Strom. Wenn es die Anwendung zulässt, können diese Geräte durch das System abgeschaltet werden:

```
clrport0 rem Schaltet die Geräte ab
setport0 rem Schaltet die Geräte an
```

Durch Nutzung der Variable gpspower kann der GPS Empfänger in seiner Leistungsaufnahme verändert werden. Das setzen dieser Variable auf 0 schaltet den Empfänger komplett aus:

```
set timer0 0 rem Zeitgeber auf 0 setzen
set mem0 0   rem Status "GPS ist aus" setzen
while (1=1)
  if (timer0=0) rem Ist Zeit abgelaufen?
    if (mem0=0) rem War GPS abgeschaltet?
      set gpspower 100 rem Ja, dann GPS mit 100% Leistung
      set mem0 1       rem Zustand "GPS wurde eingeschaltet"
    else
      set gpspower 0 rem Nein, dann GPS mit 0% Leistung
      set mem0 1     rem Zustand "GPS wurde abgeschaltet"
    endif
    set timer0 300  rem In 5 Minuten Zustand ändern
  endif
wend
```

Beachten Sie hierbei, dass es unter Umständen sehr lange dauern kann bis gültige GPS-Daten empfangen werden, wenn von gpspower 0 auf gpspower 100 gewechselt wird.

Sicherheit

Um den Zugriff auf das System nur für berechtigte Personen zuzulassen, erwartet das System bei einem Zugriff einen Benutzername (root oder user) und bei „root“ zusätzlich ein Passwort. Das Passwort kann im Skript gesetzt werden:

```
set rootpwd "4711" rem Setzt Passwort für Benutzer "root"
```

Der Benutzer „root“ hat einen kompletten Zugriff auf das System, der Benutzer „user“ hingegen hat nur einen beschränkten Zugriff. Die Einschränkungen sind wie folgt:

Thema	root	User
Datum/Uhrzeit ändern	●	●
Status abfrage	●	●
Dateizugriff	●	
Variablen anzeigen/verändern	●	
GPS/GSM Daten abfragen	●	●
SMS senden/anzeigen	●	●
Telemetriezustand abfragen	●	●

Wird das System mit GSM Funktionen genutzt kann durch eine entsprechende Anweisung (siehe „dest“ auf Seite 36) verhindert werden, dass fremde Zentrale Daten an das System schicken.

Manche Anwendungen erfordern das SMS von einem Mobilfunkgeräte empfangen und durch das System entsprechend Interpretiert werden muss. Dabei kann durch eine spezielle Variable die Rufnummer des Senders ausgelesen werden. Somit kann sichergestellt werden, dass keine fremde Person in die Steuerung eingreifen kann (siehe „origin“ auf Seite 38).

Protokolldatei

Das System kann für eine zyklische Datenerfassung genutzt werden. Dafür kann auf dem System eine Protokolldatei angelegt werden, in die beliebige Werte (Ganzzahl, Gleitkommavariablen, Text, Zeitstempel und GPS) abgelegt werden können. Die Größe der Protokolldatei wird begrenzt durch den freien nichtflüchtigen Speicher (H-Box: Flash; S-Box: MMC). Um Daten zu erfassen, muss eine Protokoll-Datei geöffnet werden:

```
logstart rem Öffnet die Datei "datcom.log"
logstart "log.txt" rem Öffnet die Datei "log.txt"
```

Danach können Angaben nach dem Schema *log typ id wert* wie folgt abgelegt werden:

```
log word 50 100 rem Legt die Zahl 100 mit der ID 50 ab
log float 13000 3.14 rem Legt die Zahl 3.14 mit der
                     rem ID 13000 ab
log long 5000 64000 rem Legt die Zahl 64000 mit der
                    rem ID 5000 ab

log text 10000 "GPS" rem Legt "GPS" mit der ID 10000 ab
log time 58000 rem Legt die Zeit mit der ID 58000 ab
log gps rem Legt die aktuelle Position ab
```

Die ID kennzeichnet einen Eintrag für ein Auswertungsprogramm um zum Beispiel anzuzeigen, dass der Eintrag den Zustand der Fahrertür beinhaltet. Die ID-Angabe ist abhängig vom verwendet Typ. Für jeden Typ wurde ein ID-Bereich eingerichtet:

Typ	ID-Bereich
word	0 bis 4095
long	4096 bis 8191
float	12288 bis 16383
text	81922 bis 12287
time	57344 bis 61439
gps	Keine ID (Fix auf 61441)

Für die Ganzzahlvariablen (mem, counter, timer) muss der Typ word verwendet werden und Gleitkommazahlen (var) werden durch float repräsentiert. Der Typ text kann für Zeichenketten (string) benutzt werden.

Es empfiehlt sich vor jedem Eintrag ein Zeitstempel abzulegen, so das eine spätere zeitliche Einordnung erfolgen kann-

Die Struktur der Datei ist im Aufbau gleich wie die H-Block-Struktur ohne Start- und Endzeichen. Die gesamte Struktur legen wir Ihnen auf Anfrage gern offen.

Da ein häufiges schreiben auf den nichtflüchtigen Speicher nicht empfohlen wird, werden die protokollierten Daten erst im Speicher gehalten bis ein größere Block an Daten erfasst wurde und dann erst auf den nichtflüchtigen Speicher abgelegt. Um Datenverlust zu vermeiden, muss bevor die H-Box in den Stand-By-Modus wechselt, die Datei geschlossen werden:

```
logend rem Schließt die Protokoll-Datei
```

Dateisystem

Die Systeme bietet einige Funktionen um Änderungen am Dateisystem auszuführen (H-Box: Flash; S-Box: MMC). Der freie Speicher kann durch folgende Anweisung ermittelt werden:

```
if (diskfree<5) rem Speicher weniger als 10 Byte?
  rem *** Alarmmeldung senden ***
endif
```

H S Beachten sie das die Angabe von diskfree ist in 2-Byte-Schritten erfolgt.

Werden z.B. mehrere Protokolldateien benutzt so kann über die Anweisung files Dateien umbenannt werden:

```
filerename "datcom.log" "log.txt" rem datcom.log nach log.txt

set string0 "log"
add string0 month ".txt" rem z.B. month = 2
filerename "datcom.log" string0 rem datcom.log nach log2.txt
```

Sind mehrere Datei vorhanden die nicht mehr benötigt werden so kann über die folgende Anweisung eine Datei gelöscht werden:

```
filedelete "log5_05.txt"
```

Auf dem System kann eine Datei mit eigenen Inhalten gefüllt werden. Dazu muss eine entsprechende Datei geöffnet werden:

```
fsopenw "daten.txt"
```

S Beachten Sie, dass der Dateiname im S-System automatisch in Großbuchstaben umgewandelt wird.

Mit der Anweisung fsput können Zeichenketten in die Datei abgelegt werden:

```
fsput "Erster Text"
fsput "Zweiter Text"
```

Damit die Daten endgültig auf den Flash geschrieben werden, muss die Datei geschlossen werden:

```
fsclose
```

Um eine Datei zu löschen wird der Befehl fsdel benötigt :

```
fsdel "test.txt"
```

Auf dem gleichen Weg kann eine Datei gelesen werden (das öffnen muss jedoch mir fsopenr geschehen):

```
fsopenr "daten.txt"
fsget string0
while (len string0>0)
  rem Gelesener Block verarbeiten
  fsget string0
wend
fsclose
```

Beachten Sie, dass der Dateiname im S-System automatisch in Großbuchstaben umgewandelt wird.

Auf dem Speicher kann über die Anweisung filefind nach **einer Datei** gesucht werden, dabei ist zu beachten, dass nur nach Angabe des vollständigen Dateinamens oder nach Angabe eines Teils des Dateinamens mit führendem „*“ genutzt werden kann. Die letzte Angabe bedeutet das nach allen Dateien mit dem Text nach „*“ am Ende ihres Dateinamens haben, gesucht wird:

```
set string0 filefind "*.txt"
```

Beachten Sie, dass der Dateiname im S-System automatisch in Großbuchstaben umgewandelt wird.

Besteht eine GPRS-Verbindung so können die beiden Systeme auch Dateien an das GPRS-Gateway senden:

```
filesend "datcom.log"
```

Tachosignal

Bei dem H-System verfügt der digitale Eingang 5 und Eingang 6 (Frequenzen von 1Hz bis 8000Hz) über die Eigenschaft, ein Tachosignal auf einen Kilometerstand umzusetzen. Analog dazu gilt für das S-System der Eingang 1 und Eingang 2 (Frequenzen von 1Hz bis 1200Hz). Um bei der Erfassung der gefahrene Kilometer eine bestmögliche Genauigkeit zu erreichen, muss eine Kalibrierung zwischen der Frequenz und der gefahrenen Kilometer durchgeführt werden.

Eine der beiden Möglichkeiten ist es, den GPS-Empfänger für das Erfassen der zurückgelegten Entfernung zu nutzen:

```
calibrate auto
```

Nach einer Entfernung von 1 Kilometer wird die automatische Berechnung für das Tachosignal beendet und der Faktor (Frequenz zu Kilometer) in der Variable knumber abgelegt. Diese Variable kann zu Testzwecken ausgelesen werden:

```
set mem0 knumber rem Verhältnis von Frequenz zu km
```

Wenn kein GPS-Empfänger Verwendet werden kann, so kann eine manuelle Kalibrierung wie folgt stattfinden:

```
calibrate start
rem *** 1km fahren ***
calibrate stopp
```

Genutzt wird diese Technik auch wenn eine sehr genaue Angabe des Tachos benötigt wird, da der GPS-Empfänger eine Ungenauigkeit durch den Betreiber aufweist. Nach der Kalibrierung kann der aktuelle Tachostand gesetzt werden:

```
set kmvalue 12000
```

Diese Variable wird jetzt entsprechenden den gefahrenen Kilometer erhöht.

Um das Verhältnis zwischen Frequenz und Tachosignal nicht bei jedem Neustart des Systems erfassen zu müssen, kann die Variable knumber in einer Datei abgelegt und erneut gelesen werden:

```
rem *** Faktor erfassen ***
```

```
write "config.xml"
read "config.xml"
```

Odometer

Mit der Variablen odometer kann über das GPS-Signal, bei eingeschalterter Zündung, der Kilometerstand ermittelt werden:

```
set odometer 1500
set string0 odometer
```

Benutzerspezifische serielle Endgeräte

An die seriellen Schnittstelle des Systems (bei S-System Data 1 und Data 2, bei H-System Com1 mit Erweiterung auch Com2 und Com3) können benutzerspezifische serielle Endgeräte angeschlossen werden. Diese Restriktionen sind bei der Kommunikation zwischen Endgerät und dem System zu beachten:

- In der Übertragung vom Endgerät zum System darf kein Zeichen mit dem Ascii-Wert 26 (Escape) auftreten
- Die Übertragung läuft mit 8 Datenbits und 1 Stoppbit

Um die serielle Schnittstelle auf eine bestimmte Übertragungsgeschwindigkeit zu setzen, kann folgende Anweisung benutzt werden:

```
set com1 stdio 9600 rem Com1/Data1 auf 9600 Baud setzen
```

Als Baudrate kann 1200, 2400, 4800, 9600, 19200, 38400 und 115200 genutzt werden.

Um Daten zu senden und zu empfangen können die beiden Variablen input und ouput verwendet werden:

```
if ("login:"=input) rem "login:" wurde empfangen
  set input ""      rem Empfangene Zeichen löschen
  set output "root" rem Zeichenkette "root" senden
  set output chr 13 rem Sende Zeilenumbruch
  set output chr 10 rem Sende "Zum Anfang der Zeile"
endif
```

Es können alle Zeichenkettenfunktionen auf die Variablen angewendet werden. Es ist lediglich zu beachten das Output nur beschrieben und nicht ausgelesen werden kann.

H S Wird ein benutzerspezifisches Gerät auf Com1/Data1 benutzt, normalerweise ist hier der Terminalbetrieb des Systems, kann durch betätigen von [Esc], [Strg+z] gefolgt von [Enter] wieder in den Terminalbetrieb zurückgekehrt werden.

Durch Nutzung der Variable getchar kann ein einzelnes Zeichen aus den empfangen Daten gelesen und verarbeitet werden.

```
if (getchar="a")
  rem *** Zeichen verarbeiten ***
endif
```

H S Beachten sie, dass die Variable getchar das gelesen Zeichen aus den empfangenen Daten löscht.

Terminalmodus auf Com2/Com3/Data2

In manchen Fällen kann die Com 1 nicht für den Terminalmodus genutzt werden, zum Beispiel bei Nutzung von Tabo 3/4 das exklusiv an Com 1 betrieben werden muss. Um dennoch den Terminalmodus nutzen zu können, kann folgende Anweisung genutzt werden:

```
set com2 term9600
set com3 term115200
```

Beachten Sie hierbei, dass die Data 2 Schnittstelle intern mit der Variable com3 angesproche n wird. Die Übertragungsgeschwindigkeit kann auf 9600 oder 115200 Baud gesetzt werden.

H-/S-System Firmwareversion

Durch die Variable version kann die Versionsnummer des H-/S-Systems ermittelt werden:

```
if (version = "S1SCR605")
   rem Nutzung von speziellen Befehle der Firmware S1SCR605
endif
```

Kapitel 3: GSM/GPRS

Die Kommunikation der beiden Systeme mit der Zentrale basieren auf GSM und GPRS. Die Verbindung zu der Zentralensoftware findet durch ein eigenes gesichertes DATCOM-Protokoll statt, es besteht aber auch die Möglichkeit einfache SMS zu senden um zum Beispiel eine Alarmmeldung an ein Mobiltelefon zu senden. Das System stellt einen Web-Server und ein FTP-Server zur Verfügung, so dass mit nur geringem Aufwand auf das Dateisystem des Systems oder auf eine benutzerfreundliche Internetseite zugegriffen werden kann.

Aufbau des GSM-Modems

Das GSM/GPRS-Modem (AC45) besitzt einen SMP Anschluss für die GSM-Antenne. Dieser wird in der H/S-Box auf einen FME Anschluss nach aussen umgeleitet. Die SIM-Kartenhalterung ist nicht im Modem enthalten und befindet sich auf der Hauptplatine (S-Box) bzw. auf der Zusatzplatine (H-Box). Um eine einwandfreien Nutzung des GSM/GPRS-Moduls zu garantieren beachten sie die Einbau- und Sicherheitshinweise auf Seite 93.

GSM Datenübertragung

Die Kommunikation über das GMS-Netz findet über SMS-Meldungen statt. Vorraussetzung dafür ist, dass dem System bekannt ist, welcher Service-Center (Vermittlungsstelle für SMS) genutzt werden soll:

```
set smsc +491710760000 rem SMSC für D1
set smsc +491722270000 rem SMSC für D2
set smsc +491770610000 rem SMSC für E-Plus
set smsc +491760000443 rem SMSC für Viag-Intercom
```

Generell sollten die SIM-Karten mit einem PIN versehen werden. Das System nimmt als Standardeinstellung den Pin „1111“ an. Soll ein anderer Pin genutzt werden, dann kann folgende Variable verwendet werden:

```
set pin "3490" rem Setzt den Pin 3490
```

H/S Beachten Sie, dass die SIM-Karte eventuell nach falscher Eingabe gesperrt ist (meistens nach dreimalig falscher Eingabe). Zur falschen Eingabe zählt auch, dass im Gerät der PIN auf einen anderen Wert ausser 1111 steht.

Der Vorteil bei der Nutzung des DATCOM-Protokolls liegt darin, das bei jeder SMS die aktuelle Position und den Zustand der Ein- und Ausgänge des Grundsystems (ohne Ein-/ Ausgabeerweiterung) mit gesendet werden kann. Damit diese sicherheitsrelevanten Daten nicht von unbefugten Zentralen abgerufen werden können, muss die Zentrale im Skript definiert sein:

```
set       dest1       +49123456789       rem       Erste       Zentrale
...
set dest4 +49987654321 rem Vierte Zentrale
```

Um zu überprüfen ob das GSM-Modul bereits im GSM-Netz eingebucht ist, kann folgende Variable verwendet werden.

```
if (registered)
  rem *** Anweisungen, wenn GSM empfang ***
endif
```

Die aktuelle Qualität des Empfangs kann so ausgelesen werden:

```
if (quality<10)
  rem *** Anweisungen, wenn sehr schlechter Empfang ***
endif
```

Dabei steht der Wert 0 für sehr schlechten Empfang und der Wert 32 für einen idealen Empfang.
Ob sich das System gerade im Sendevorgang befindet kann durch die Variable transmit geprüft werden:

```
if (not transmit) rem Nicht am Senden?
  var0=ain1/0.025 rem z.B. Spannung messen
endif
```

Der Provider des GSM-Netzes kann durch die Variable operator ermittelt werden:

```
if ("D1-Telekom"=operator)
  rem *** Im D1-Netz ***
endif
```

Auf der SIM-Karte ist im Normalfall die eigene Rufnummer hinterlegt, die durch ownnumber abgefragt werden kann. Beachtet sie, dass einige Mobilfunkbetreiber diese Funktionalität nicht zur Verfügung stellen.

```
set string0 "Meine Nummer ist "
add string0 ownnumber
```

Die DATCOM-SMS beinhaltet neben einem Text oder einer Zahl auch noch welchen Zweck die Meldung erfüllen soll (kann durch DATCOM-Fleet bestimmten Ereignissen zugeordnet werden).

- *daton*: Gerät Eingeschaltet
- *datoff*: Gerät ausgeschaltet
- *dattele*: Telemetriedaten, d.h. Eingänge von Basissystem
- *dattemp*: Temperaturdaten (siehe Seite 39)
- *datgps*: Nur Fahrzeugposition
- *dattext*: Textmeldung
- *datalarm:* Alarmmeldung
- *dathint*: Hinweismeldung
- *datkey*: Statusmeldung
- *datcardin*: Karte in Kartenleser

Hierbei ist zu beachten, dass sowohl daton, datoff, dattele, und datgps kein Text und keine Zahl übergeben wird. Wird bei den restlichen Anweisungen ein Text übergeben, so darf er maximal 130 Zeichen lang sein.
Beispiele für die Übermittlung:

```
send datalarm +49123456789 "Alarm"
send datgps string0
send cardin +49123456789 "Karte gesteckt"
```

Der Parameter datkey verlangt neben der Zentralennummer und einem Text auch eine Statusnummer. Dieser Statusnummer können durch DATCOM-Fleet bestimmte Ereignisse zugeteilt werden.

```
send datkey +49123456789 h30 "Taste 0"
```

Eine Standard SMS kann bis zu 160 Zeichen enthalten und wird wie folgt versendet:

```
send stdtext +49123456789 "Alarm im Auto" rem oder mit
send stdsms +49123456789 "Alarm im Auto"
```

Wie oft versucht wird eine SMS z.B. bei einer schlechten GSM-Verbindung zu versenden, lässt sich über die Variable smsretry einstellen (Grundeinstellung ist 5 Sendeversuche):

```
set smsretry 10 rem 10 mal versuchen eine SMS zu versenden
```

Ein empfangen einer Standard SMS kann durch die Variable recstdtext geprüft werden. Die Variable recstdtext zeigt an ob eine Standard SMS empfangen wurde:

```
if (recstdtext) rem Standard SMS empfangen?
  send dattext +491234567489 "Meldung"
endif
```

Beachten Sie, dass die Variable recstdtext automatisch nach dem Abfragen zurückgesetzt wird.

Auf den empfangenen Text kann über die Variable stdtext zugegriffen werden:

```
if (recstdtext) rem Standard SMS empfangen?
  if ("Alarm"=stdtext) rem Steht "Alarm" in der SMS ?
    setport1
  endif
endif
```

Analog dazu kann auch eine DATCOM-SMS ausgewertet werden:

```
if (recdattext) rem DATCOM SMS von Zentrale empfangen?
  if ("Alarm"=dattext) rem Steht "Alarm" in der SMS ?
    setport1
  endif
endif
```

DATCOM-SMS werden nur von der Zentrale (oder mehreren Zentralen), die zuvor durch setzen der „dest“-Variable bekannt gegeben wurde, ausgewertet. Dadurch ist sichergestellt, dass die Meldungen die durch dritte übermittelt wurden, nicht interpretiert werden. Bei Standard SMS kann zum gleichen Zweck die Rufnummer des Senders abgefragt werden:

```
if (recstdtext)
  if ("+49123456789"=origin)
    setport1
  endif
endif
```

Durch die DATCOM-Fleet-Software können Zustände in Form von einzelnen Bits versendet werden. Die funktionale Aktion hat frei im Skript anhand der indizierten Variable rxbit definiert werden:

```
if (rxbit1) rem Wurde Schaltausgang 1 gesetzt?
  setport1
endif
```

Um gezielt auf Veränderungen einzelner rxbits zu reagieren, kann folgende Anweisungen genutzt werden:

```
if (/rxbit1) rem Schaltausgang 1 wurde gesetzt
  setport1
endif
if (\rxbit2) rem Schaltausgang 2 wurde zurück gesetzt
  rem *** Anweisungen ***
endif
```

Mit der Variable recrxbit kann abgefragt werden, ob eine SMS zum steuern der RX-Bits empfangen wurde:

```
if (recrxbit) rem SMS zum Steuern der RX-Bits empfangen
  rem *** RX-Bits auswerten ***
endif
```

recrxbit wird direkt beim Abfragen zurückgesetzt, so dass z.B. nur innerhalb einer if-Anweisung auf recrxbit reagiert werden kann.

Wird ein zweites Mal einen setzen der RX-Bits durch die Zentrale geschickt und es sollen die gleichen RX-Bits gesetzt werden, so wird diese SMS verworfen.

Mit der Anweisung store können bis zu vier Messwerte erfasst werden und in einen Zwischenspeicher abgelegt werden. Die Größe des Zwischenspeichers entspricht einer SMS, sobald der Speicher voll ist, wird dieser Speicher als SMS an die DATCOM Fleet Software übertragen. Dort können die vier Kanäle ausgewertet und angezeigt werden:

```
store var1 "Fühler2 defekt" ain1 ain2
```

GSM-Modul ausschalten und neustarten

Bei einigen Anwendungen kann es erforderlich sein Energie zu sparen (zum Beispiel bei Übersee-Transport). Wenn auf eine zeitnahe Positionsmeldung verzichtet werden kann ist es möglich das GSM-Modul aus bzw. einzuschalten:

```
set gsmpower 0   rem GSM-Modul ist ausgeschaltet
set gsmpower 100 rem GSM-Modul ist eingeschaltet
```

Unter Umständen kann es auch nötig werden das Modul neu zu starten.

```
gsmreboot
```

Übertragung in H-Block-Struktur

Die Kommunikation mit eigenen GSM-Geräten oder Softwareapplikationen kann auch über SMS auf der Basis der H-Block-Struktur erfolgen.

Der erste Schritt beim versenden von H-Block-Daten besteht darin, Werte in einen Zwischenspeicher zu schreiben:

```
smswrite word 1 256
smswrite text 10000 "Türen geöffnet"
```

Der Aufbau von smswrite ist identisch mit tcpwrite (siehe Seite 53).
Der Zwischenspeicher kann durch die Anweisung smssend an ein Endgerät versendet werden:

```
smssend +49017012345678
```

Ist der Zwischenspeicher voll weil zum Beispiel kein Netz empfangen wird, so kann durch die Anweisung smsclear diese Zwischenspeicher gelöscht werden:

```
smsclear
```

Um empfangene Daten auszuwerten muss smsrec zyklisch aufgerufen werden. Solange Daten vorhanden sind, wird die Variable smsread auf 1 gesetzt. Nach einmaligen Abfragen der Variable smsrec wird automatisch der nächste Datensatz gewählt:

```
smsrec
while (smsread)
  rem Diese Schleife wird für jeden empfangene Datensatz durchgeführt
wend
```

Die empfangenen Daten können mit den beiden Variablen smstype und smsdata ausgewertet werden:

```
smsrec
while (smsread)
  if (smstype=100)
    set string1 smsdata rem Enthält die empfangenen Daten
  endif
wend
```

Die funktionale Zuordnung der Typen kann frei entschieden werden, es ist jedoch zu empfehlen sich an die Vereinbarung auf Seite 55 zu halten, um einen späteren wechseln zu GPRS zu vereinfachen.

Externe GSM Verbindung über MC35i

Das GSM-Modem MC35i wird über eine serielle Schnittstelle[6] mit dem System verbunden. Nach dem einstellen des richtigen Protokolls auf der entsprechenden seriellen Schnittstelle, findet die Kommunikation zum GSM-Netz ausschließlich über das MC35i statt:

```
set com1 gsm
```

Externe GSM Verbindung über Nokia 6090/810

Das Nokia 6090/810 ist ein Mobiltelefon für den Einbau in Fahrzeugen und findet häufig Verwendung in Transportunternehmen. Um die Kommunikation über die serielle Schnittstelle ein Nokia 6090/810 zu betreiben, muss folgende Anweisung ausgeführt werden:

```
set com1 nokia
```

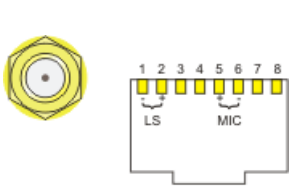
GSM Sprache

Das im System verbaute GSM-Modul bietet die Möglichkeit eine Sprachverbindung zu anderen Teilnehmer im Mobilnetz oder im Festnetz aufzubauen. Um eine Sprachverbindung von bzw. zu dem System herzustellen, muss das System im GSM-Netz eingebucht sein (sichtbar durch die LED „Netz EB“ bei dem H-System und durch die LED „SIM/Netz“ bei dem S-System). Durch das Skript kann über die Variable registered (siehe Seite 36) abgefragt werden ob das System im GSM-Netz eingebucht ist.

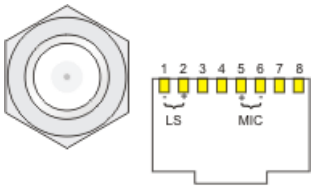
Folgende Anschlussbelegung beschreibt den Anschluss für einen Telefonhörer:

H-System	**S-System**

[6] Benötigt das Verbindungskabel Artikelnrummer 4052



GPS MIC/SPK



GSM MIC/SPK

Zum Aufbau einer Sprachverbindung kann die Anweisung call benutzt werden. Die call-Anweisung erwartet eine Telefonnummer die mit einem Ländercode beginnt und optional eine Zeit (Sekunden) wie lange der Anruf ausgeführt werden soll:

```
call +49123456789
call +49123456789 100
call string0 120
```

Nach einem call kann mit den folgenden Systemvariablen der Zustand der Verbindung erfasst werden:

```
if (refused)
  rem *** Gegenstelle hat Anrufe nicht angenommen ***
endif
if (answered)
  rem *** Gegenstelle hat Anrufe angenommen ***
endif
if (busy)
  rem *** Gegenstelle ist besetzt ***
endif
if (online)
  rem *** Sprachverbindung besteht ***
endif
```

Um einen Anruf zu erkennen kann die Variable ring genutzt werden. Die Variable ring wird parallel zum bekannten akustischen Signal gesetzt und zurückgesetzt, d.h. während des Anrufs wechselt ring zwischen 1 und 0.

```
rem Wenn an Ausgang 1 ein akustischer Signalgeber
rem angeschlossen ist, wird hier ein Klingeln erzeugt
if (ring)
  setport1
else
  clrport1
endif
```

Ein Anruf kann durch die Anweisung answer entgegengenommen werden:

```
if (ring)
  answer
endif
```

Um eine bestehende Verbindung, unabhängig ob das System die Verbindung aufgebaut hat oder angerufen wurde, zu beenden kann die Anweisung hangup benutzt werden:

```
if (mem1=1) rem Ist mem1 auf 1 gesetzt?
  hangup    rem Auflegen
  set mem1 0
endif
```

Wird bei einem Anruf an das System die eigene Rufnummer mit übergeben, so kann die Rufnummer durch die Variable clip abgefragt werden:

```
if (ring)
```

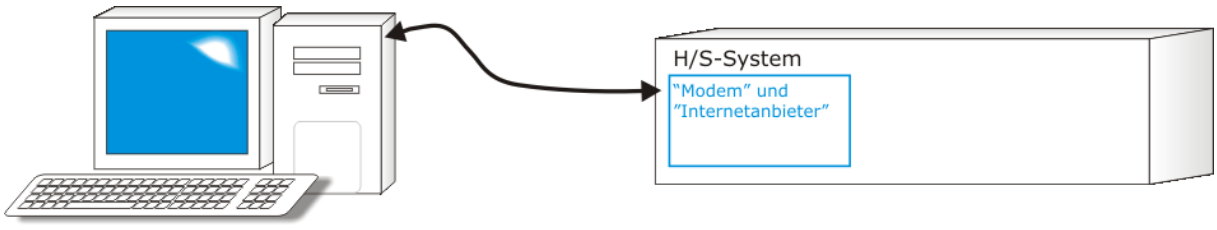
```
    if (clip="+49123456789")
      answer
    endif
  endif
```

TCP/IP und GPRS Datenübertragung

Das System bietet einen Web-Server für HTML-Seite und einen FTP-Server für die Übertragung von Dateien an. Dadurch ist es Möglich mit jedem Internetfähigen Computer Dateien auf das System zu übertragen oder detaillierte Informationen über den Zustand des Systems zu erfragen. Dabei kann direkt über ein Verbindungskabel[7], über ein Modem oder über eine stationäre H-/S-Box eine Verbindung zum System aufgebaut werden.

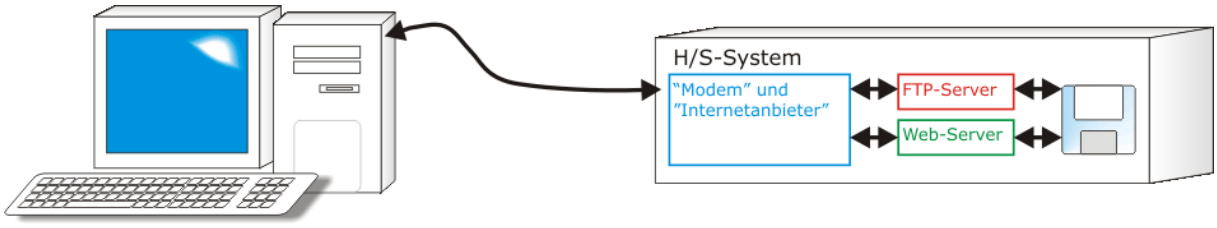
Direkte TCP/IP Verbindung im Überblick

Die Verbindung zu dem H-/S-System gestaltet sich wie der Verbindungsaufbau zu einem Internetanbieter. Der PC nutzt das System wie ein Modem und wählt eine beliebige Nummer, danach bestätigt das System die Verbindung und Benutzername/Kennwort wird geprüft.



PC nutzt DFÜ-Verbindung

Das System vergibt an den PC die IP-Adresse 111.111.111.112, das H-/S-System selbst hat die IP 111.111.111.111. Durch einen FTP-Client kann jetzt auf den FTP-Server des Systems, der den Speicher des Systems widerspiegelt, zugegriffen werden. Alternativ kann auch über einen Internetbrowser auf den Web-Server zugegriffen werden.



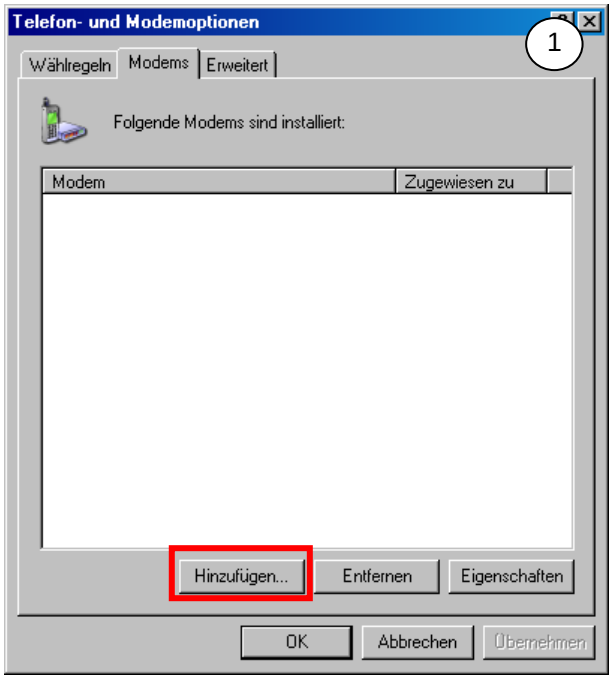
PC nutzt DFÜ-Verbindung

[7] Artikel 2359 für die H-System und Artikel 4004 für das S-System

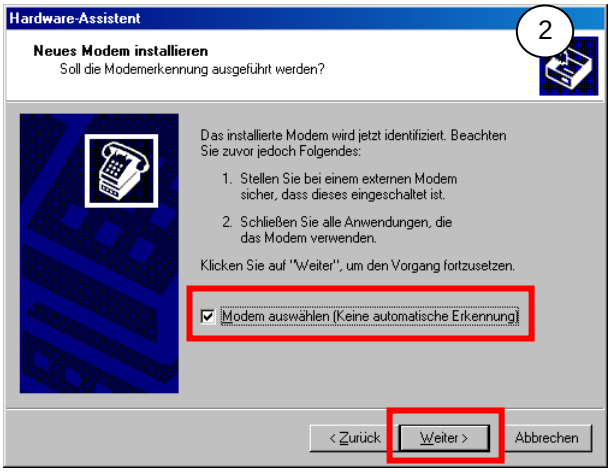
Direkte TCP/IP Verbindung Einrichten

Modem einrichten

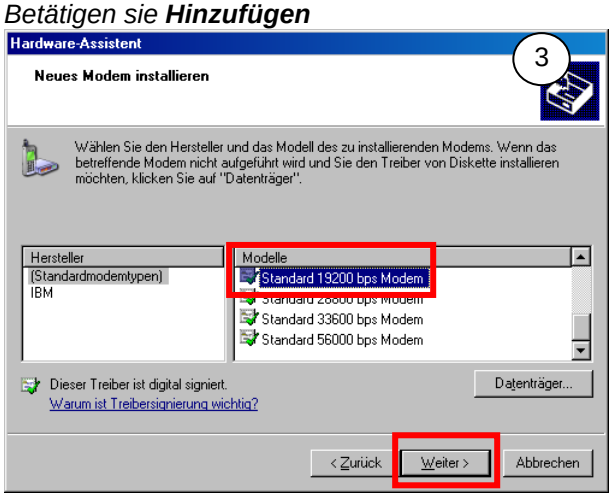
Im ersten Schritt muss ein Modem über **Systemsteuerung**, **Telefon- und Modemoptionen** eingerichtet werden:



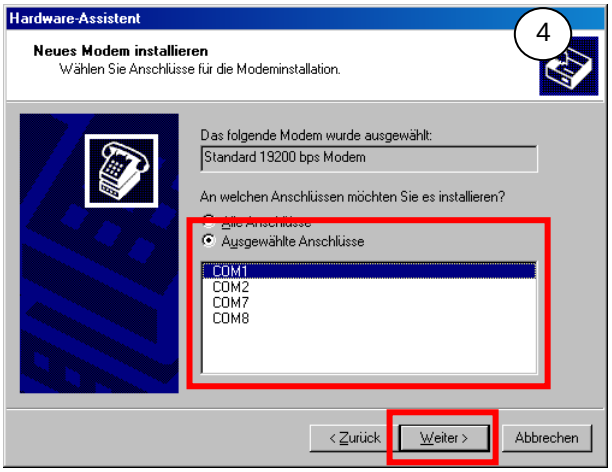
*Betätigen sie **Hinzufügen***



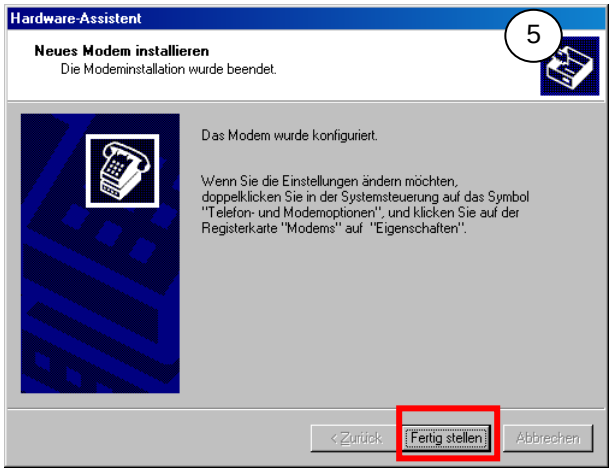
***Modem auswählen** markieren und **Weiter** betätigen*



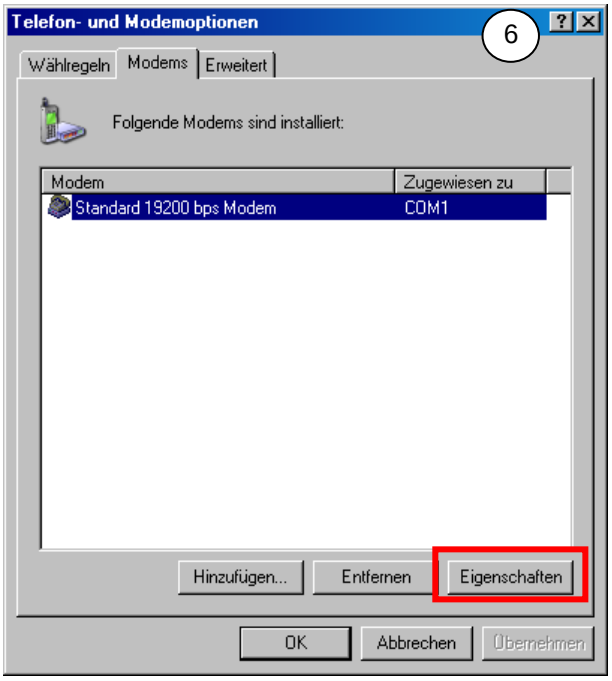
***Standard 19200 bps Modem** auswählen und **Weiter** betätigen*



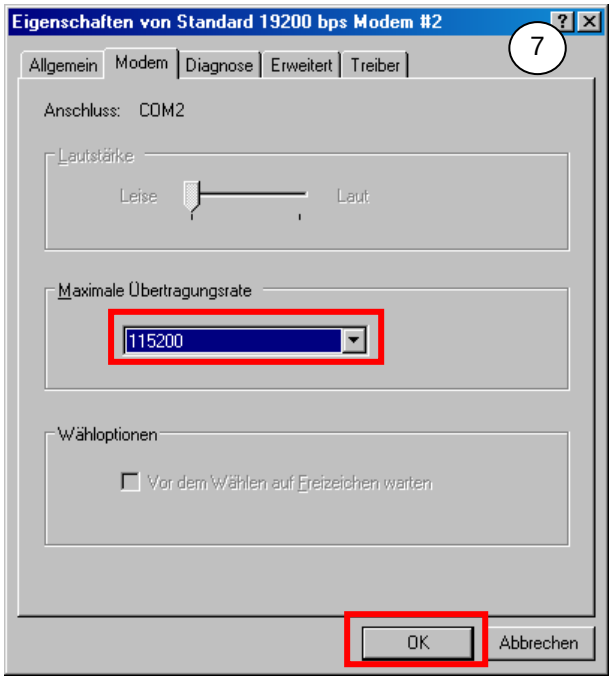
*Wählen sie die Schnittstelle an der das H-/S-System angeschlossen ist und betätigen sie **Weiter***



*Betätigen sie **Fertig stellen***



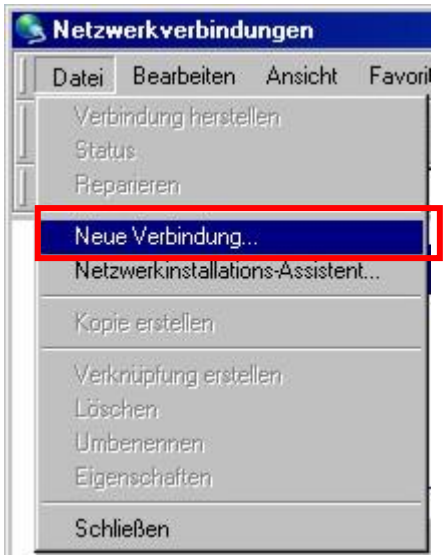
*Betätigen sie **Eigenschaften***



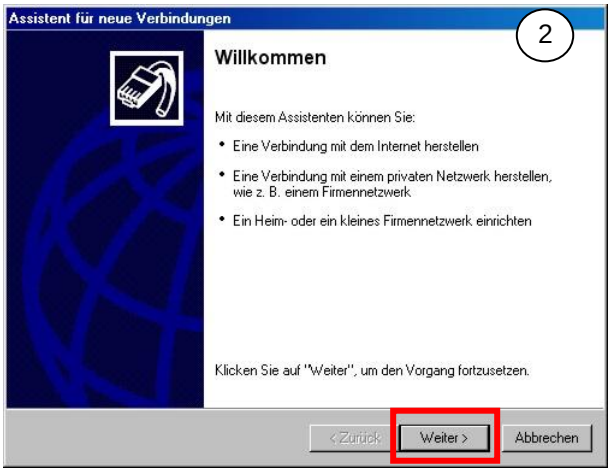
*In der Registerkarte **Modem** die **Maximale Übertragungsrate** auf **115200** einstellen und mit **OK** bestätigen*

DFÜ-Verbindung einrichten

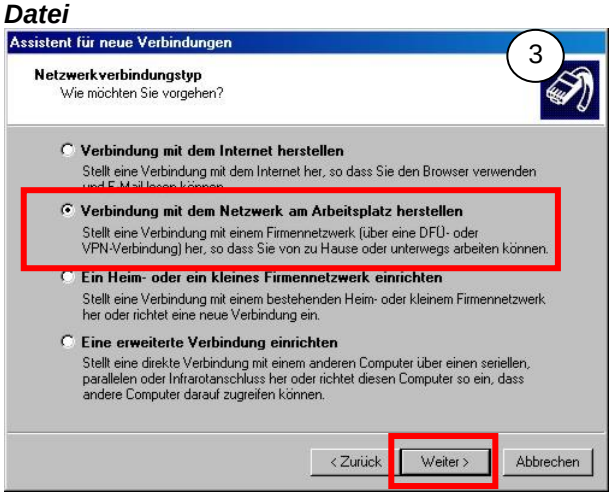
Im zweiten Schritt muss eine DFÜ-Verbindung unter **Systemsteuerung**, **Netzwerkverbindungen** zur Einwahl in das System eingerichtet werden:



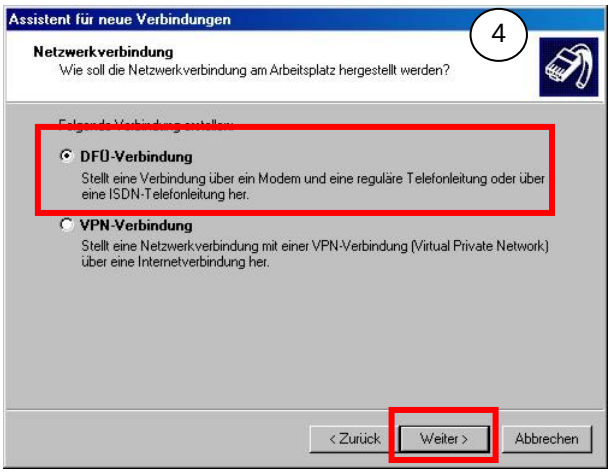
*Wählen sie **Neue Verbindung** aus dem Menü **Datei***



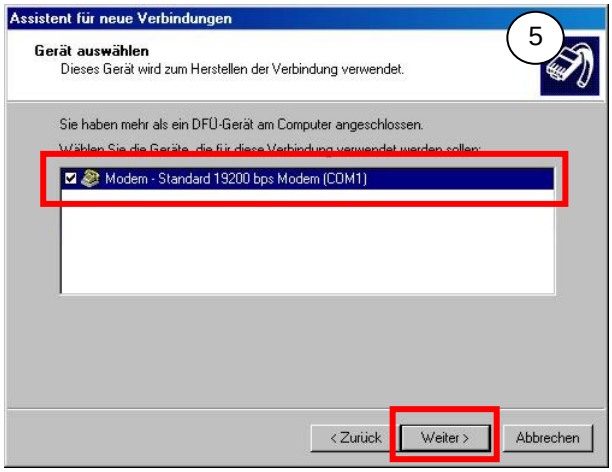
*Betätigen sie **Weiter***



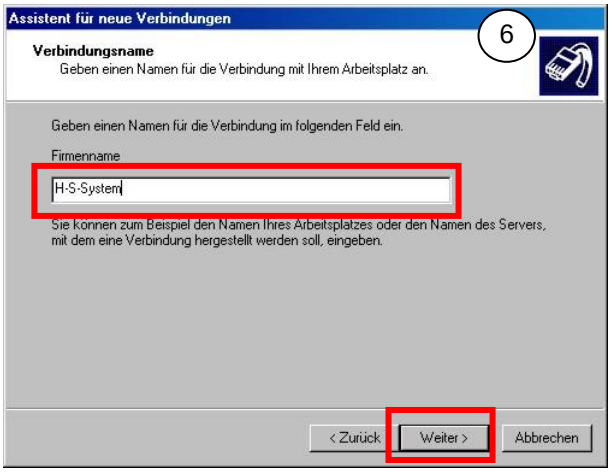
*Wählen sie **Verbindung mit dem Netzwerk am Arbeitsplatz herstellen** und betätigen sie **Weiter***



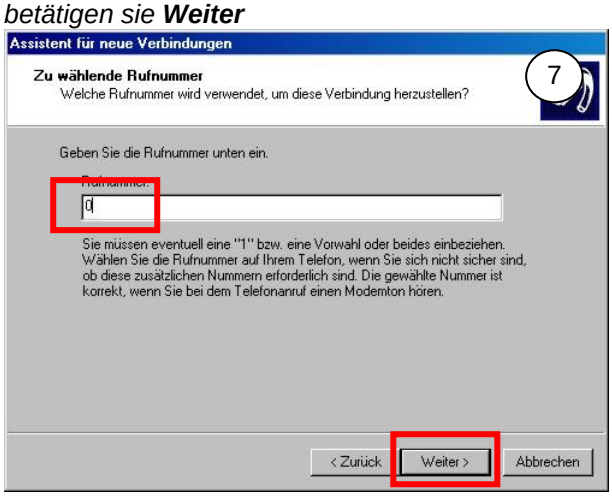
Wählen sie **DFÜ-Verbindung** und betätigen sie **Weiter**



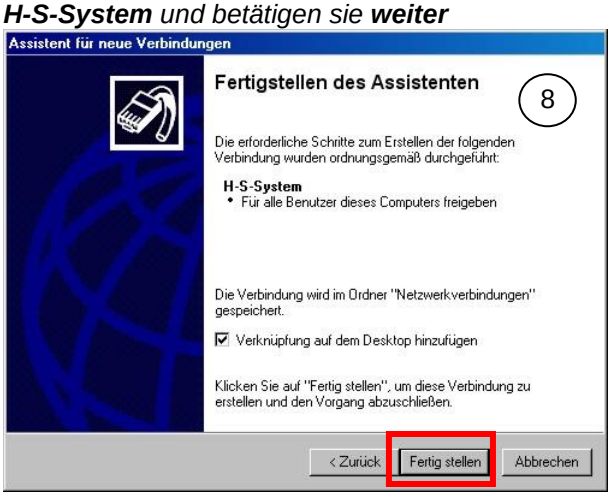
*Wählen sie **Modem – Standard 19200...** und betätigen sie **Weiter***



*Geben sie den Namen für die Verbindung ein z.B. **H-S-System** und betätigen sie **weiter***

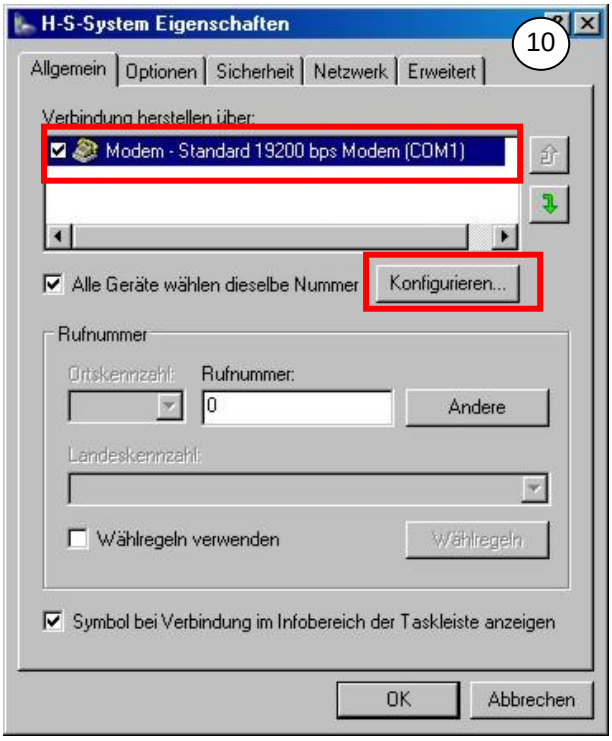


*Geben sie als **Rufnummer** die Zahl **0** ein und betätigen sie **Weiter***

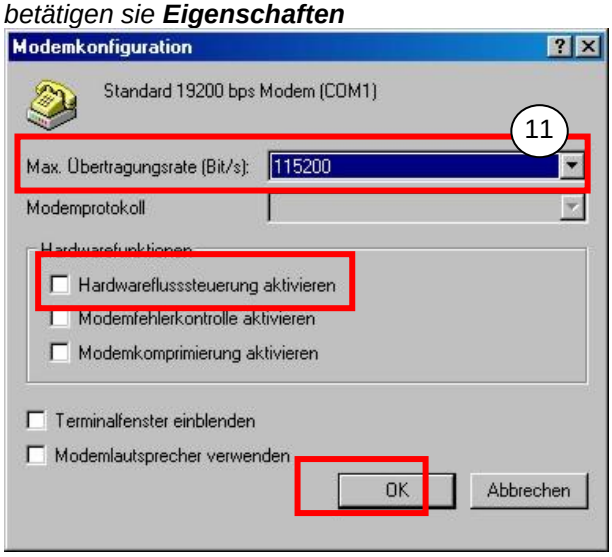


*Aktivieren sie **Verknüpfung auf dem Desktop hinzufügen** und betätigen sie **Fertig stellen***

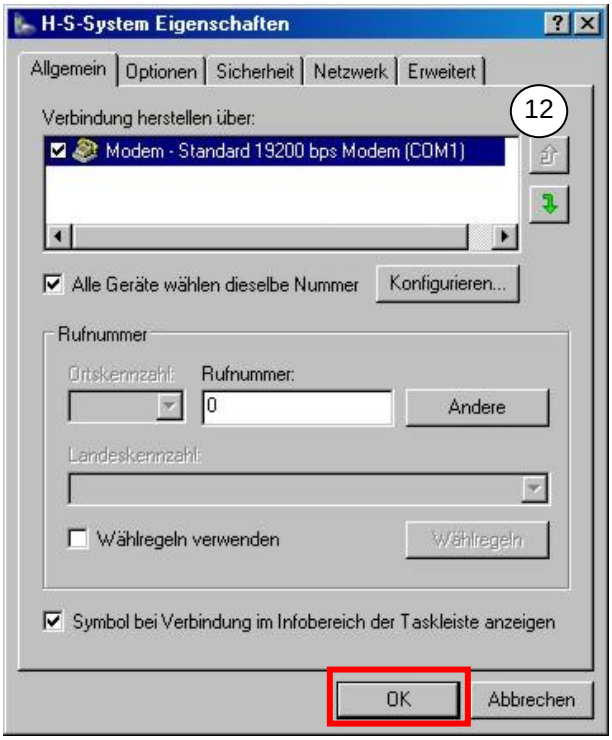
*Geben sie als **Benutzername** „**datcom**“ und als **Kennwort** „**1111**“ ein, aktivieren sie **Benutzername und Kennwort speichern für** und betätigen sie **Eigenschaften***



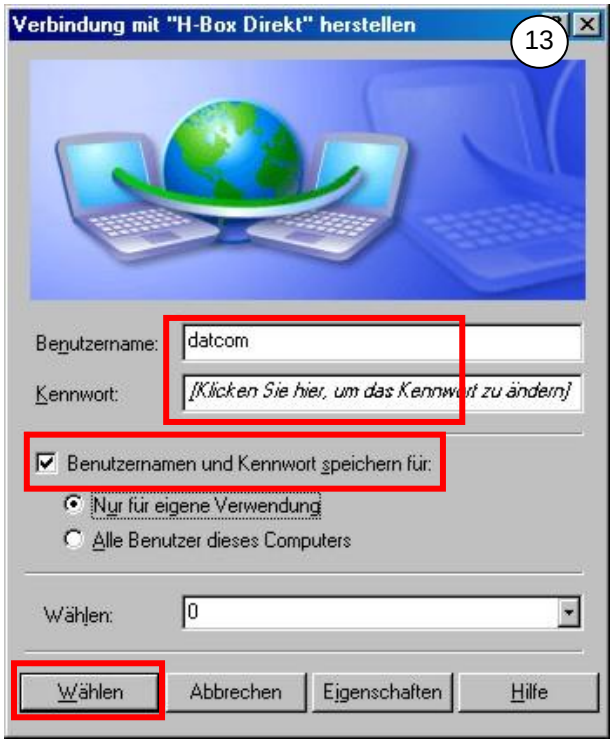
Markieren sie **Modem – Standard 19200...** und betätigen sie **Konfiguration**



Stelle sie bei **Max. Übertragungsrate** die Geschwindigkeit von **115200** ein, deaktivieren sie **Hardwareflusssteuerung aktivieren** und bestätigen sie mit **OK**.



*Betätigen sie **OK***



Verbinden sie jetzt das H-/S-System mit der Stromversorgung (wenn bereits die Stromversorgung anliegen, dann unterbrechen sie diese kurz) und betätigen sie ***Wählen***

Nach dem wählen besteht eine Verbindung zwischen PC und dem H-/S-System.

Modem TCP/IP Verbindung einrichten

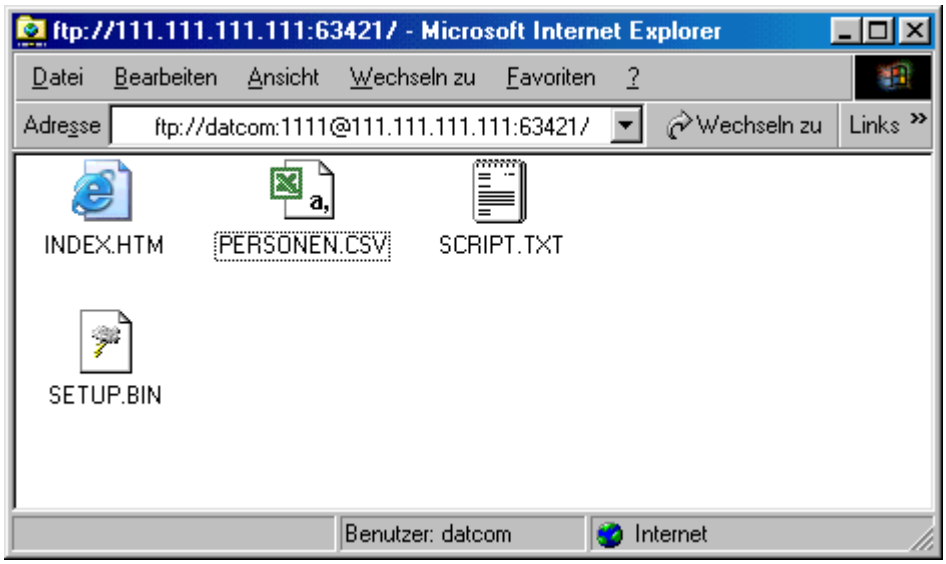
Richten Sie das Modem, wie im Handbuch oder Installationsbeschreibung des Modems beschrieben wird, ein und erstellen sie eine DFÜ-Verbindung genau wie bei einer direkten Verbindung (siehe Seite 45), nur das sie anstelle des Standard Modem ihr installiertes Modem wählen. Bei der Anwahl verwenden sie anstelle der „0“ die Rufnummer des H-/S-Systems.

Zugriff auf den FTP-Server

Um die Datei auf dem H-/S-System zu ändern kann eine herkömmliche FTP-Software z.B. SmartFTP oder psFTP genutzt werden. Die benötigten Verbindungsdaten sind:

- IP-Adresse: 111.111.111.111
- Port: 63421
- Benutzername: datcom
- Passwort: 1111

Es ist generell Möglich den Internet Explorer zu verwenden jedoch kann über diese Software kein Firmware update und auch keine Formatierung durchgeführt werden:



Erlaubt Ihre Software freie Befehle an einen FTP-Server zu senden, so können Sie zusätzlich folgende Funktionen ausführen:

- Firmware Update
  Benutzen Sie den Befehl „UPDATE firmware.bin“ um eine neue Firmware auf das System zu übertragen. Zuvor muss eine entsprechende Firmware auf das System übertragen werden („firmware.bin“ ist stellvertretend für Ihre neue Firmware)

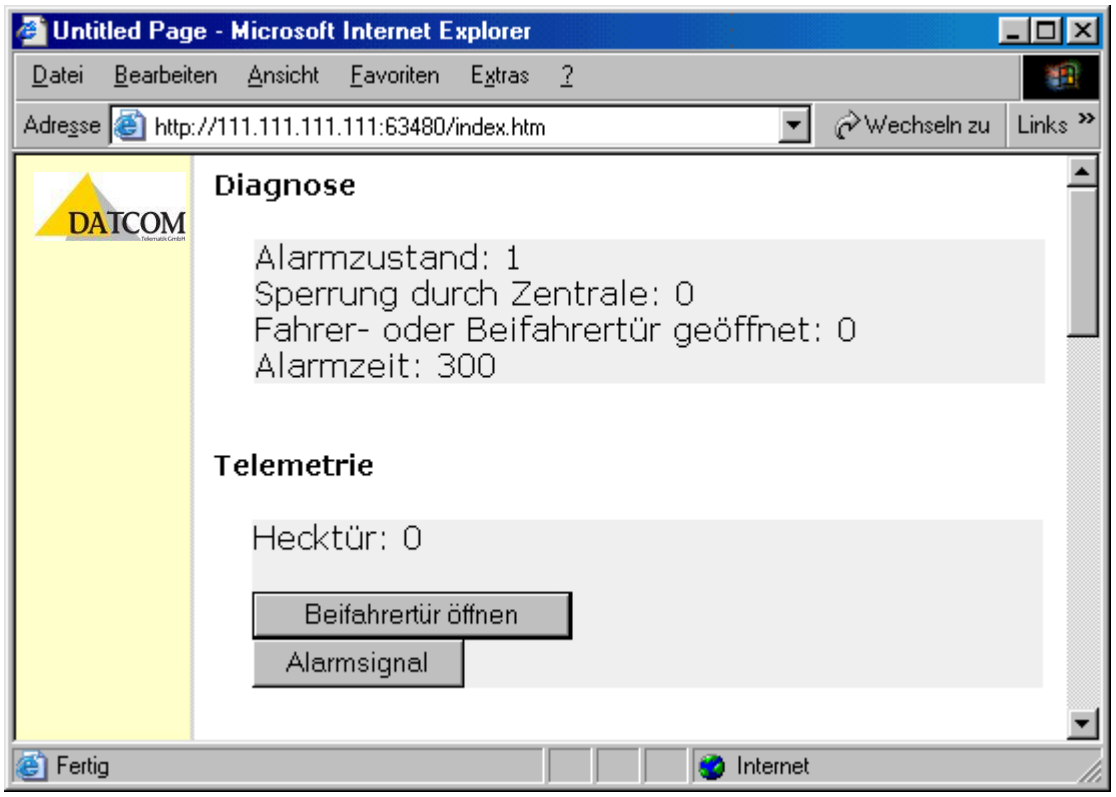
- Formatieren
  Durch den Befehl „FORMAT“ kann der nichtflüchtige Speicher (H-System: Flash; S-System: MMC) formatiert werden.

Durch die Variable ftpport ist es Möglich einen anderen frei wählbaren Port für den FTP-Server festzulegen:

```
set ftpport 2000 rem Im Browser: ftp://111.111.111.111:2000
set ftpport 21 rem Standard-Port für FTP
```

Zugriff auf Web-Server

Befinden sich HTML-Dateien mit der Endung „.htm“ auf dem System so können diese über den Web-Server abgerufen werden:



Die HTML-Datei können mit SSI-Befehle (Server Side Include) Angaben über Variablen und Eingänge gemacht werden. Dafür werden die SSI-Befehle, die in der HTML-Datei vorhanden sind, durch den Wert der entsprechenden Variablen ersetzt und an den Internet Browser gesendet. Folgendes Beispiel zeigt wie die SSI-Befehle in der HTML-Datei eingesetzt werden:

```
<html>
<body>
Inhalt von String0: <!--#echo var="STRING0" -->
</body>
</html>
```

Achten Sie darauf, dass die Angabe des Variablennamens in Großbuchstaben erfolgt.

Folgende SSI-Befehle können genutzt werden (als Beispiel wurde immer der Index 0 benutzt, es können hier natürlich alle zur Verfügung stehen Variablen eines Typs genutzt werden):

Befehl	Beschreibung
`<!--#echo var="STRING0" -->`	Gibt die jeweilige Zeichenkette zurück.
`<!--#echo var="VAR0" -->`	Gibt die jeweilige Gleitkommazahl zurück.
`<!--#echo var="MEM0" -->`	Gibt die jeweilige Variable zurück.
`<!--#echo var="TIME" -->`	Gibt die aktuelle Zeit des Systems zurück.
`<!--#echo var="DATE" -->`	Gibt das aktuelle Datum zurück.
`<!--#echo var="TIMER0" -->`	Gibt den Inhalt des jeweiligen Zeitgebers zurück.
`<!--#echo var="COUNTER0" -->`	Gibt den jeweiligen Zähler zurück.

<!--#echo var="CLIERR" -->	Gibt einen bestehenden Fehler im Skript zurück
<!--#echo var="ERRMSG" -->	Gibt einen bestehenden Fehler im Skript zurück
<!--#echo var="GPSLON" -->	Gibt den Längengrad zurück
<!--#echo var="GPSLAT" -->	Gibt den Breitengrad zurück
<!--#echo var="IN1" -->	Gibt den Eingang zurück (0 oder 1)

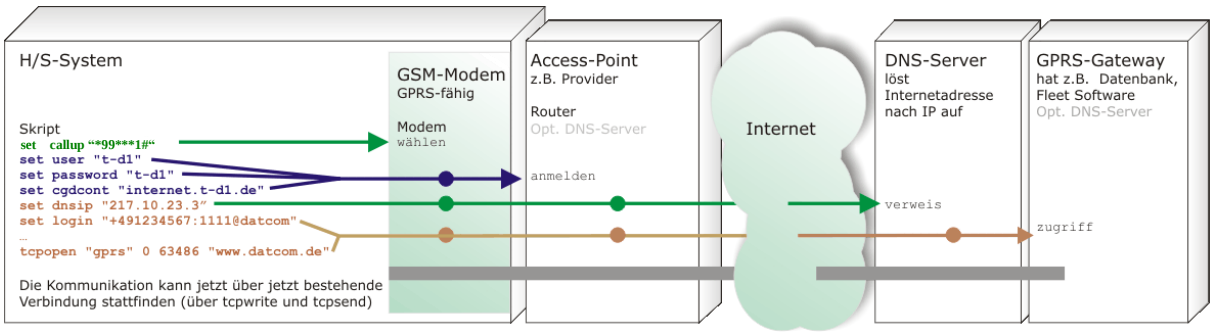
Durch die Variable httpport ist es Möglich einen anderen frei wählbaren Port für den Web-Server festzulegen:

```
set httpport 1500 rem Im Browser: http://111.111.111.111:1500
set httpport 80 rem Standard-Port für HTTP
```

GPRS Datenübertragung

Genau wie bei der Kommunikation über GSM können Daten auch über GPRS versendet werden. Der Vorteil bei GPRS liegt dabei in den angepassten Kostenmodellen der Mobilfunkbetreiber. Durch diese Technologie können auch Zeichenketten über 160 Zeichen (maximale Anzahl von Zeichen bei einer Standard SMS) hinaus weltweit übertragen werden.

Bevor jedoch eine Übertragung stattfinden kann, muss die Verbindung zum GPRS-Server aufgebaut werden. Um leichter das Verständnis über den Verbindungsaufbau zu erlangen betrachten sie bitte folgende Skizze:



Die Verbindung zum GPRS-Gateway stellt sich im Skript folgendermaßen da:

```
set callup "*99***1#" rem Rufnummer des GPRS-Anbieters
```

Bei callup wird die Einwahlnummer im Siemens-Format „*Rufnummer#“ angegeben. Im nächsten Schritt muss sich das System bei dem Access-Point, das kann ihr Mobilfunkbetreiber sein oder aber auch ein VPN-Server, angemeldet werden:

```
set user "t-d1" rem Gibt den Benutzernamen an
set password "t-d1" rem Gibt das Passwort an
set cgdcont "internet.t-d1.de" rem Access-Point
```

Mit dem Befehl coding kann festgelegt werden ob die zu übertragenden Daten verschlüsselt werden. Hierzu muss die Gegenstelle die Daten auch wieder entschlüsseln:

```
set coding 1 rem Datenübertragung verschlüsselt
set coding 0 rem Datenübertragung unverschlüsselt
```

Die Daten werden ihnen durch den Mobilfunkbetreiber bereitgestellt. Damit Servernamen (URL) in IP-Adressen umgesetzt werden können, wird ein DNS-Server benötigt. Bietet der Mobilfunkbetreiber keinen DNS-Server an oder soll innerhalb eines VPN gearbeitet werden, so muss der entsprechende DNS-Server gesetzt werden:

```
set dnsip "217.3.42.2"
```

Für die Anmeldung am GPRS-Gateway wird eine Zugangszeichenkette benötigt, die sich aus einer Kennung, normalerweise die Rufnummer der SIM-Karte, aus einem Passwort und einem Benutzernamen zusammensetzt. Das Format der Zeichenkette sieht folgendermaßen aus: „rufnummer:passwort@benutzer“.

```
set login "+49123456789:1111@datcom"
```

Die eigene Rufnummer kann auch durch die Variable ownnumber ermittelt werden (beachten sie hierzu Seite 37).

Gelingt es nicht eine Verbindung zum GPRS aufzubauen, so wird nach einer definierten Wartezeit ein erneuter Versuch unternommen. Bei jedem scheitern wird diese Zeit erhöht. Die Wartezeit kann durch die folgende Variable gesetzt werden:

```
set callupto 5 rem Wartezeit auf 5 Minuten setzen
```

Die GPRS-Verbindung kann durch die Anweisung gprsclose beendet werden:

```
gprsclose
```

Die Kommunikation mit dem GPRS-Gateway wird durch tcpopen eröffnet, dabei muss die Geräteschnittstelle „gprs“[8], den lokalen Port, den Port des GPRS-Gateway und die IP bzw. Adresse des GPRS-Gateway:

```
tcpopen "gprs" 0 63586 "www.datcom.de"
tcpopen "gprs" 0 63586 "217.7.100.188" rem Alternativ
```

Das H-/S-System fällt zyklisch vom GPRS-Betrieb in den GSM-Betrieb zurück. Durch die Anweisung tcpopen wird nach einer Zeit wieder in den GPRS-Betrieb gewechselt.

H S Beachten Sie, dass die Kommunikation zyklisch geöffnet werden muss, daher wird empfohlen mindestens einmal pro Skriptdurchlauf ein tcpopen durchzuführen.

Um zu erfahren ob das System sich gerade im GPRS-Modus (oder im GSM-Modus) befindet, kann die Variable gprsactive abgefragt werden:

```
if (gprsactive)
  rem *** System im GPRS-Modus ***
endif
```

Durch abfragen der Variable loggedin kann ermittelt werden, ob das System sich an dem GPRS-Gateway angemeldet hat:

```
if (loggedin )
  rem *** Verbunden mit GPRS-Gatway ***
endif
```

Das Versenden von Daten über GPRS erfolgt über die beiden Anweisungen tcpwrite und tcpsend. Die Anweisung tcpwrite schreibt einen Datenblock in einen Zwischenspeicher:

```
tcpwrite word 0 100 rem 100 in Zwischenspeicher
tcpwrite long 5000 64000 rem 64000 in Zwischenspeicher
tcpwrite text 10000 "Test" rem "Test" in Zwischenspeicher
tcpwrite time 60000 rem Zeitstempel speichern
tcpwrite gps rem Position speichern
```

Direkt nach tcpwrite folgt die Angabe des Datentyps (alle Typen sind oben angegeben). Danach folgt eine ID, um einen geschrieben Wert zu klassifizieren. Dadurch kann ein spezielles Verhalten in DATCOM Fleet erzeugt werden. Die zugeordneten ID-Bereiche und Funktionen sind im Folgenden Erklärt:

Wertbereich	Beschreibung
0 bis 4095	*tcpwrite word id mem**n*** Der Bereich ist für Ganzzahlvariablen bestimmt. Im Bereich von 1 bis 1000 und 1004 bis 4095 können freie Klassifizierungen stattfinden.

[8] Als Geräteschnittstelle kann hier auch ein COM-Port angegeben werden um mit TCP/IP-Fähigen Geräte direkt Kontakt aufzunehmen.

	ID / **Beschreibung** 0 Telemetrische Daten (Variable ina) 1001 Statusmeldung/Tastenmeldung (im Taxibereich) 1002 Einschaltmeldung 1003 Ausschaltmeldung
4096 bis 8191	*tcpwrite long id kmvalue* Der komplette Bereich kann für große Ganzzahlvariablen (Variable kmvalue), mit Ausnahme der Kennung 5000, frei genutzt werden. **ID** / **Beschreibung** 5000 Messwert (kmvalue)
8192 bis 12287	*tcpwrite text id "Text"* Der Bereich ist für Zeichenketten bestimmt. Einige sind mit folgenden Funktionen verknüpft: **ID** / **Beschreibung** 10000 Normale Zeichenkette 10001 Text zu einem zuvor gesendeten Status (ID 1001) 10010 Normale Zeichenkette (Änderbar durch DATCOM Fleet) 10011 Zeigt Zeichenkette in einem Fenster an (Änderbar durch DATCOM Fleet) 10012 Zeigt Zeichenkette in einem Alarmfenster an (Änderbar durch DATCOM Fleet) 10101 Messwert für Kanal 1 (Sichtbar in DATCOM Fleet) 10102 Messwert für Kanal 2 (Sichtbar in DATCOM Fleet) 10103 Messwert für Kanal 3 (Sichtbar in DATCOM Fleet) 10104 Messwert für Kanal 4 (Sichtbar in DATCOM Fleet)
61441	*tcpwrite gps* Diese ID wird nur für die Übertragung der Position benötigt. Der Befehl tcpwrite erwartet hierbei jedoch keine ID.

Nachdem Daten in den Zwischenspeicher geschrieben worden sind, kann über die Anweisung tcpsend dieser Zwischenspeicher an den GPRS-Gateway gesendet werden:

```
tcpsend
```

tcpsend kann in zwei verschiedenen Arten betrieben werden:

Blockier-Modus (Standard)	Bei Aufruf von tcpsend wird das Skript solange angehalten bis eine Übertragung erfolgt ist oder bis eine Zeit abgelaufen ist. Der Vorteil daran ist, das man direkt mit tcpresult prüfen kann ob eine Übertragung funktioniert hat oder nicht:
	`set tcpnonblocking 0`
	`tcpsend`
	`if (tcpresult=0)`
	`  rem *** Fehler bei Übertragung ***`
	`else`
	`  rem *** OK ***`
	`endif`
	Nachteil ist, das Sicherheitsrelevante abfrage nicht mehr durchgeführt werden.

Nicht-Blockier-Modus	Der Aufruf tcpsend veranlasst das H-/S-System die Übertragung parallel zur Abarbeitung des Skriptes durchzuführen. Das bedeutet die Abfrage ob eine Übertragung erfolgreich war, muss für den ständigen Durchlauf entwickelt werden:
	`set tcpnonblocking 1`
	`tcpsend`
	`…`
	`if (tcpready=1)`
	`  if (tcpresult=0)`

```
      rem *** Fehler bei Übertragung ***
    else
      rem *** OK ***
    endif
  endif
```

Wie durch die Beispiele gezeigt wird, kann der „Nicht-Blockierende-Modus“ durch setzen der Variable tcpnonblocking aktiviert werden. Auch ist im obigen Beispiel die Verwendung von tcpresult gezeigt. Diese Variable zeigt an das eine Übertragung erfolgreich war oder nicht.

Um zu vermeiden, dass Daten doppelt gesendet werden, kann folgende Anweisung zum Beispiel nach dem überprüfen ob ein Fehler vorliegt, dazu benutzt werden den Speicher zu löschen:

```
tcpclear
```

Um empfangene Daten auszuwerten muss tcprec zyklisch aufgerufen werden. Solange Daten vorhanden sind, wird die Variable tcpread auf 1 gesetzt. Nach einmaligen Abfragen der Variable tcprec wird automatisch der nächste Datensatz gewählt:

```
tcprec
while (tcpread)
   rem Diese Schleife wird für jeden empfangene Datensatz durchgeführt
wend
```

Empfangene Datensätze enthalten, genau wie beim Senden von Daten, eine ID zur Klassifizierung. Die Variable tcptype gibt Auskunft über die ID des aktuellen Datensatzes. Dabei sind folgende Klassifizierung festgelegt worden:

ID	Beschreibung
0	Steuerbits von DATCOM Fleet
100	GPS einmalig anfordern
101	Aufforderung GPS zyklisch nach **x** Minuten zu senden
102	Aufforderung GPS zyklisch nach **x** Sekunden zu senden
103	Aufforderung GPS zyklisch nach **x** mal 100 Meter zu senden
10000	Zeichenkette **x**
10002	Zeichenkette im Datenbankformat („Z;La;Lo;Atext…“)
10003	Zeichenkette für ein Gebiet („Lat1;Lon1;Lat2;Lon2;Text“)
10004	Zeichenkette für eine Zielortung („Lat;Lon;Text“)
10005	Zeichenkette für eine Meldung wenn die Distanz zu einem Punkt kleiner ist als **x** („Lat;Lon;Distanz“)

Der entsprechende Parameter oder die entsprechende Zeichenkette ist in tcpdata abgelegt:

```
tcprec
while (tcpread)
   if (tcptype=100) rem GPS einmalig anfordern
     tcpwrite gps
   endif
   if (tcptype=10000) rem Freie Zeichenkette
     set string0 tcpdata
   endif
wend
```

Bei der ID 0 handelt es sich um die DATCOM Fleet Steuerbits (Zustände können frei in DATCOM Fleet gesetzt werden) und die Auswertung findet wie folgt statt:

```
if (tcptype=0)
   set mem5 tcpdata
   if (mem5 & 1)
      rem *** Steuerbit 1 ***
```

```
    endif
    if (mem5 & 2)
       rem *** Steuerbit 2 ***
    endif
    if (mem5 & 4)
       rem *** Steuerbit 3 ***
    endif
    if (mem5 & 8)
       rem *** Steuerbit 4 ***
    endif
    …
endif
```

Eine TCP-Verbindung kann manuell durch folgende Anweisung getrennt werden (hierbei wird nicht die GPRS-Verbindung unterbrochen):

```
tcpclose
```

Durch die Variable gprsxmtcnt und gprsrcvcnt kann abgefragt werden wie viele Bytes gesendet und wie viele Bytes empfangen wurden:

```
if (gprsxmtcnt >100)
   rem Mehr als 100 Byte seit start des Systems versendet
endif
if (gprsrcvcnt >100)
   rem Mehr als 100 Byte seit start des Systems empfangen
endif
```

UDP Übertragung

Eine weitere Möglichkeit Daten zu senden besteht darin das UDP-Protokoll zu nutzen. Durch folgende Anweisung könnnen Daten in einen Zwischenspeicher gelegt werden (udpsend datentyp id text):

```
udpwrite word 0 34 rem Schreibt 34 in den Zwischenspeicher
udpwrite long 5000 70000 rem 70000 in den Zwischenspeicher
udpwrite text 9000 "Text" rem Schreibt "Text" in den Zwischenspeicher
udpwrite gps rem Schreibt die Position in den Zwischenspeicher
```

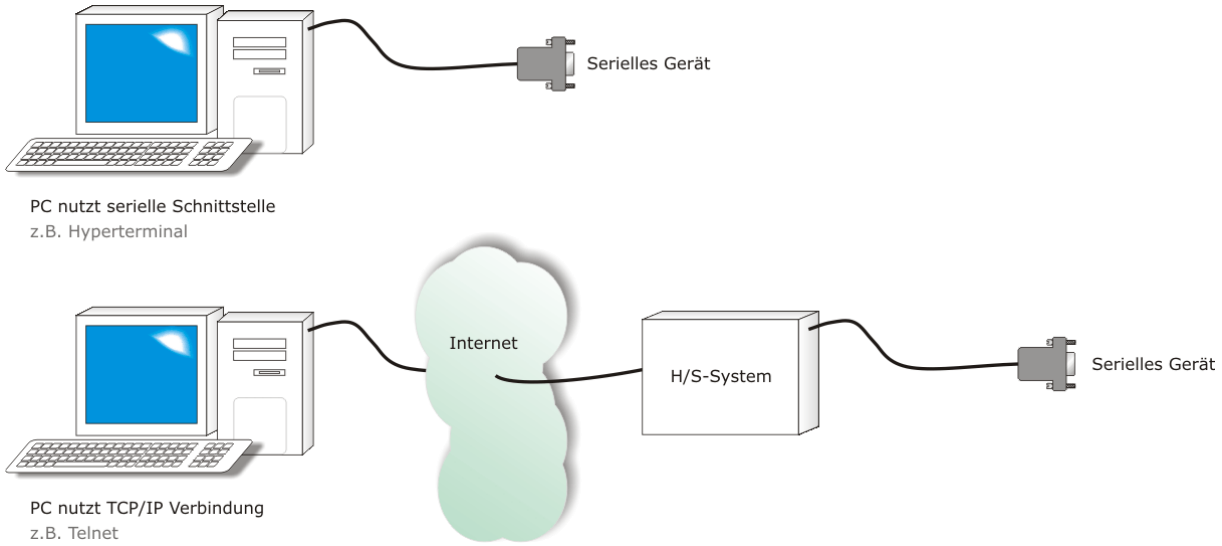
Genau wie bei der Verwendung von tcpwrite muss auch hier zum senden der Befehl udpsend benutzt werden:

```
udpsend "com1" 63402 63401 "111.111.111.112" udpdata
```

Der erste Parameter beschreibt die zu nutzende **Geräteschnittstelle** hier können die beiden seriellen Schnittstellen (Com1 und Com2) genutzt werden. Bei der S-Box muss „com1“ für Data1 und „com3“ für Data2 genutzt werden. Anschließend folgt der **lokale Port** (darf frei gewählt werden) und der **Zielport**. Die nachfolgende Adresse gibt den **Empfänger** der Daten an. Wählt sich ein Gerät bei dem H-/S-System ein so bekommt dieses Gerät automatisch die IP „111.111.111.112“ zugewiesen. Um jetzt ein Datenpaket an das eingewählte Gerät zu senden muss die IP „111.111.111.112“ in der Anweisung udpsend benutzt werden. Anstelle von udpdata, welches die zuvor **gesammelten Daten** (mit udpwrite) beinhaltet, kann auch eine Zeichenkette angebenen, die dann im Klartext über UDP übertragen wird.

Serielle Verbindung via GPRS

Die beiden Systeme können als Brücke zwischen einem PC und einem seriellen Endgerät dienen. Dazu muss im Skript eine Schnittstelle mit einem TCP-Port verbunden werden. Die folgende Skizze zeigt die Nutzung des transparenten Com-Servers:



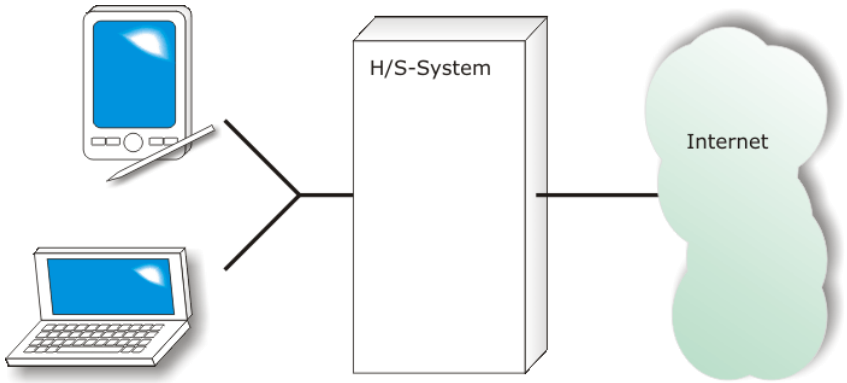
Das umlenken von einem TCP-Port geschieht über folgende Anweisung:

```
set com1 tracomsvr 19200 63456 rem 19200 Baud auf Com 1
```

Das entsprechende Programm für die Kommunikation (z.B. Telnet oder Hyperterminal) muss sich über die IP des H-/S-Systems (abfragbar durch DATCOM Fleet) und der entsprechenden Portnummer (hier z.B. 63456) verbinden.

H-/S-System als Router

Hat sich ein Gerät (zum Beispiel PDA oder Laptop) in das H-/S-System eingewählt und möchte über das System in das Internet oder VPN-Netz, so kann die bestehende GPRS-Verbindung genutzt werden. Die so genannte NAT-Funktion (Network Adress Translation [Netzwerkadressen Umsetzung]) leitet die Daten von einem Endgerät zum Internet weiter und leitet Daten aus dem Internet wiederum an das Endgerät.



Um die Umsetzung in dem System zu starten ist folgende Anweisung zu nutzen:

```
set natdev 1 rem NAT für Com1/Data1 einschalten
```

Dabei bezeichnet der Parameter die zu nutzende Schnittstelle:

Parameter	1	2	3
H-Box	COM 1	COM 2	COM 3
S-Box	Data 1	/	Data 2

Bei der Verwendung von Endgeräte wie zum Beispiel einem PDA, nutzen verschiedene Programme den Internetzugang um Daten mit einem Server auszutauschen. Um zu verhindert das unbenötigte Serveranfragen (einige Programme prüfen ob ein Update für sie vorhanden ist) können in dem H-/S-System Regeln für die Firewall hinterlegt werden:

```
set fwrule inc tcp any pop3 "217.7.100.188"
```

Der Aufbau der Regel gestaltet sich folgendermaßen:

set fwrule	inc	tcp	any	pop3	"217.7.100.188“
	Erstellt die Regel für eingehende Daten. Hier kann auch „out“ für ausgehende Daten oder „any“ für alle Verbindungen angegeben werden.	Gibt das Protokoll an das Freigegeben werden soll. Hier können die Protokolle „tcp“, „udp“, „icmp“. Dürfen alle Protokolle genutzt werden so kann „any“ angegeben werden.	Beschreibt welcher lokale Port Daten senden darf. Im Regelfall ist nicht bekannt welcher lokale Port vom PDA/Laptop genutzt wird und es wird hier „any“ angegeben	Beschreibt den Zielport. Hier wurde Port von „pop3“ gewählt (Port 25). Der Port kann durch die Bezeichner „ftp“, „http“, „smtp“, „pop3“ oder durch eine Zahl beschrieben werden. Gibt es keine Einschränkung dann kann hier „any“ benutzt werden.	Gibt die IP-Adresse oder den Servername, mit dem Kommuniziert werden kann, an. Hier darf auch die Angabe „any“ erfolgen wenn der Zugriff für alle Server erlaubt ist

Um sich von der generellen Funktion des Zugangs zu überzeugen, kann folgende Regel eingesetzt werden:

```
set fwrule any any any any any
```

Damit werden alle Daten in beide Richtungen durchgeleitet.

Kapitel 4: GPS Positionsbestimmung

Das System bietet die Möglichkeit eine Positionsbestimmung durchzuführen, um zum Beispiel die Darstellung auf einer Landkarte zu ermöglichen. Es ist auch möglich zu überwachen, ob ein Gebiet befahren oder verlassen wird. Damit kann sichergestellt werden, dass sich Fahrzeuge nicht unbewacht bewegen können oder, im Falle eines Transportunternehmens, sofort erkannt werden kann welches Fahrzeug sich am dichtestem an einem neuen Auftrag befindet.

Vorraussetzung ist, das in dem H-/S-System ein GPS-Empfänger verbaut ist. Das ist abhängigen von der jeweiligen Variante des H-/S-Systems.

Positionsangaben

Ob eine gültige Position empfangen wurde, kann durch die Variable gpsdata erfragt werden:

```
if (gpsdata=1)
  rem *** Gültige GPS-Daten vorhanden ***
endif
```

Wenn eine gültige Position vorliegt, kann durch die Variable longitude den Längengrad und durch latitude den Breitengrad erfasst werden. Die aktuelle Geschwindigkeit kann durch die Variable velocity genauso abgefragt werden, wie auch die aktuelle Höhe durch die Variable altitude und die Fahrtrichtung durch die Variable direction:

```
if (gpsdata=1) rem Gültige Position von GPS-Empfänger
  set string0 "GPS-Information :"

  rem Längengrad in string0 ablegen
  add string0 longitude

  rem Breitengrad in string0 ablegen
  add string0 "-" latitude

  rem Höhe in string0 ablegen
  add string0 " auf einer Höhe von " altitude

  rem Fahrtrichtung in string0 ablegen
  add string0 " Richtung " direction

  rem Geschwindigkeit in string0 ablegen
  add string0 " mit einer Geschwindigkeit von " velocity
else
  set string0 "Kein  GPS-Empfang"
endif
```

Wenn eine Kommunikation über SMS stattfindet, dann kann eine zyklische Positionsmeldung mit folgender Anweisung gesetzt werden:

```
set gpscycle 300 rem Sendet jede 5 Minuten eine Position
```

Durch die Variable gpsrequest kann durch das H-/S-System bestimmt werden, ob eine zyklische Abfrage bzw. eine einmalige Abfrage der Position erfolgen darf:

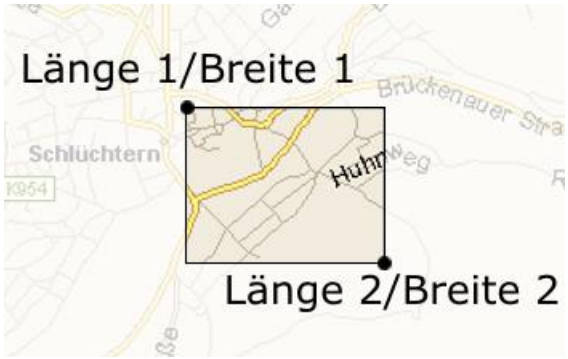
```
set gpsrequest 0 rem Keine Abfrage der Position erlauben
set gpsrequest 1 rem Abfrage der Position ist zulässig
```

Beachten sie, dass zuvor die richtige SMSC (siehe Seite 36) gesetzt wurde. Eine Lösung in dieser Art ist für die Kommunikation über GPRS nur durch ein Skript zu lösen (Intervalle mit GPS, siehe Seite 89)

Um mit dem H-/S-System ein Bereich zu überwachen, können die Variablen mit dem Namen area wie folgt verwendet werden:

```
set area1 50.37402 9.53029 50.32402 9.63229 "Schlüchtern"
```

Es können 200 Bereiche so definiert werden (1 bis 200) und dabei ist das Format „set area**n** Breite1 Länge1 Breite2 Länge2 "Bereichsname"“ einzuhalten. Als Skizze lässt sich der Befehl folgendermaßen zeigen:



Die Variable area kann wie folgt abgefragt werden:

```
if (area1=1)
   rem *** Fahrzeug im Bereich 1 ***
endif
if (area1=0)
   rem *** Fahrzeug nicht im Bereich 1 ***
endif
if (/area1)
   rem *** Fahrzeug hat Bereich 1 befahren ***
endif
if (\area1)
   rem *** Fahrzeug hat Bereich 1 verlassen ***
endif
```

Genau wie bei den anderen Datentypen kann auch der Index nach dem Namen der Variable durch eine ganzzahlige Variabel ersetzt werden. Das kann dann genutzt werden, wenn mehrere Bereiche durchlaufen werden sollen. Zusätzlich kann mit der Variable areaname der Name des entsprechenden Bereiches abgefragt werden (siehe oben):

```
set string0 ""
set mem10 1
while (mem10<21)
  if (area(mem10)=1)
     add string0 "Im Bereich" areaname(mem10)
  endif
  set mem10 (mem10+1)
wend
```

(Sind zwei geographische Punkte bekannt, so kann durch die Anweisung *distance Breitengrad1 Längengrad1 Breitngrad2 Längengrad2* die Entfernung zwischen den beiden Punkten errechnet werden:

```
rem Berechnet die Entfernung von der aktuellen Position bis zur
rem Koordinate 50.1233|8.54534
var2 = distance latitude longitude 50.1233 8.54534
```

Genauigkeit der Position erhöhen

Um die Qualität der Positionsbestimmung zu erhöhen kann die minimal zu empfangende Satellitenzahl erhöht werden und ein maximaler DOP-Wert gesetzt werden:

```
set gpsminsat 5 rem Erst ab 5 Satelliten Position als gültig erklären
set gpsmaxdop 10 rem Maximale Ungenauigkeit setzen
```

Der DOP-Wert (Dilution of Precision) gibt an wie ungenau ein Signal sein darf. Ungenauigkeit kann durch schlechte Verteilung der Satelliten entstehen, da die GPS-Satelliten in einer nicht geostationären Umlaufbahn kreisen. Wenn hier ein zu kleiner Wert eingegeben wird, dann versucht der GPS-Empfänger eine sehr hohe Genauigkeit zu bekommen. Im ungünstigsten Fall wird es nie zum Empfang einer Position kommen. Durch Nutzung der Variable gpsdop kann der DOP-Wert der letzen Position ermittelt werden und die entsprechende Anzahl von Satelliten in der Variable gpssats:

```
if (gpsdop>7) rem Sehr hohe Ungenauigkeit
   set string0 "GPS sehr ungenau"
else
   set string0 "GPS-Empfang ist gut"
endif
if (gpssats<5)
   add string0 "aber wenige Satelliten"
endif
```

GPS ausschalten

Bei einigen Anwendungen sie es erforderlich Energie zu sparen (zum Beispiel bei Übersee-Transport). Wenn auf eine zeitnahe Positionsmeldung verzichtet werden kann und auch keine hohe Genauigkeit erforderlich ist, kann der GPS-Empfänger in ein Energiespar-Modus versetzt werden:

```
set gpspower 50 rem 50% der Leistung
```

Benutzt man anstelle von 50 den Wert 0 so wird der GPS-Empfänger abgeschaltet. Wird der Wert 100 gesetzt so ist der GPS-Empfänger im Normalbetrieb.

NMEA und Uhrzeit vom GPS-Empfänger

Durch Nutzung der Variable nmea kann direkt die NMEA-Zeichkette abgefragt werden:

```
set string0 nmea
```

Genauso bieten die folgenden Variablen Bestandteile der aktuellen empfangenen Uhrzeit:

```
set mem1 gpshour rem Ermittelt die aktuelle Stunde
set mem2 gpsmin rem Ermittelt die aktuelle Minute
set mem3 gpssec rem Ermittelt die aktuelle Sekunde
```

Beachtet sie, das die Angaben in GMT+0 angegeben ist.

Wird die komplette Uhrzeit oder das komplette Datum benötigt, dann können die beiden Folgenden Anweisungen genutzt werden:

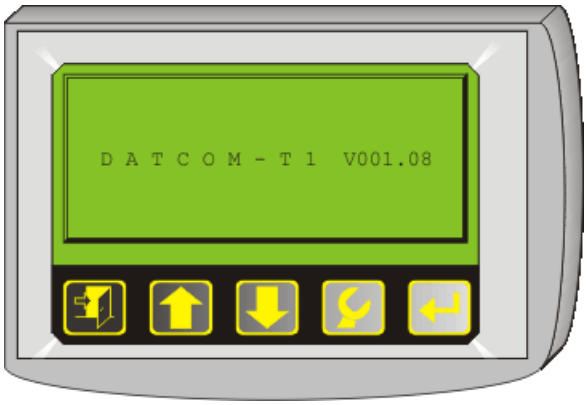
```
set string1 gpstime rem Ermittelt die aktuelle Zeit (hh:mm:ss)
set string2 gpsdate rem Ermittelt das aktuelle Datum (tt.mm.jj)
```

Kapitel 5: Zubehör

Für das H-/S-System steht eine Vielzahl von Zubehörprodukten zur Verfügung. Eine Gruppe bilden die Terminals, die zur Kommunikation mit dem Anwender bestimmt sind. Dabei kann zwischen Navigationsgeräten, universelle Terminals und reine Steuerungsgeräte unterschieden werden.

Eine andere Gruppe besteht aus verschieden Kabelsätze um zum Beispiel ein Nokia 6090 zu nutzen oder um ein Temperaturschreiber anzusprechen.

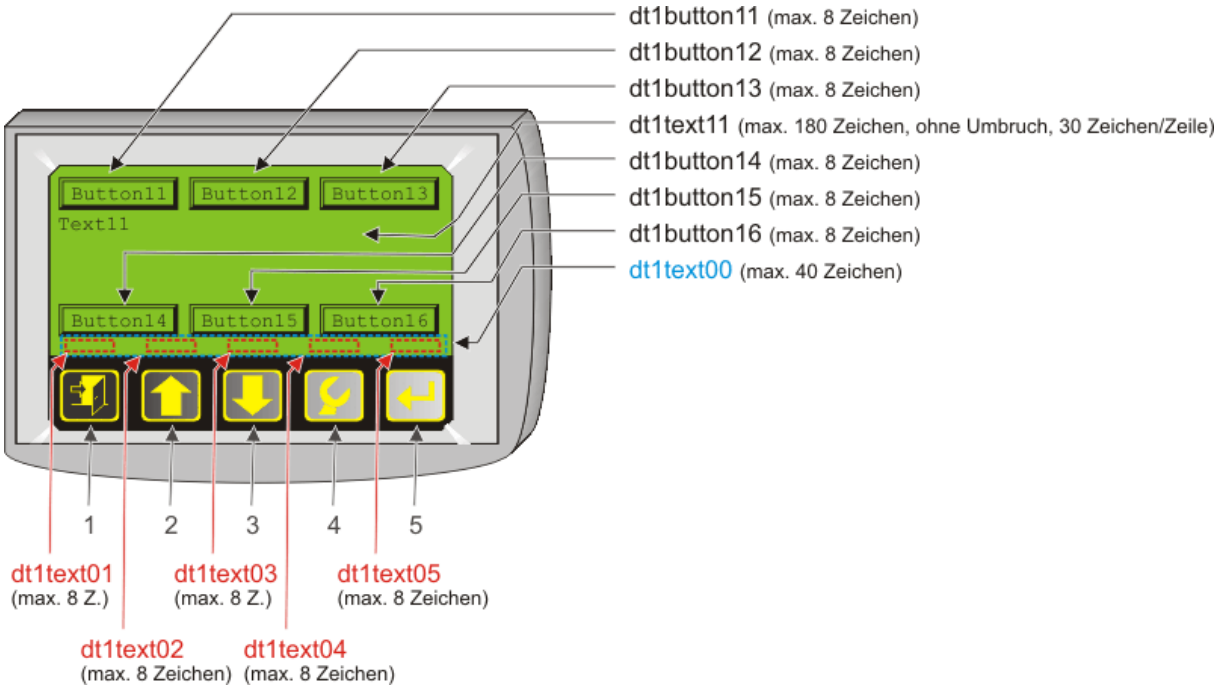
Terminal DATCOM 640T1T



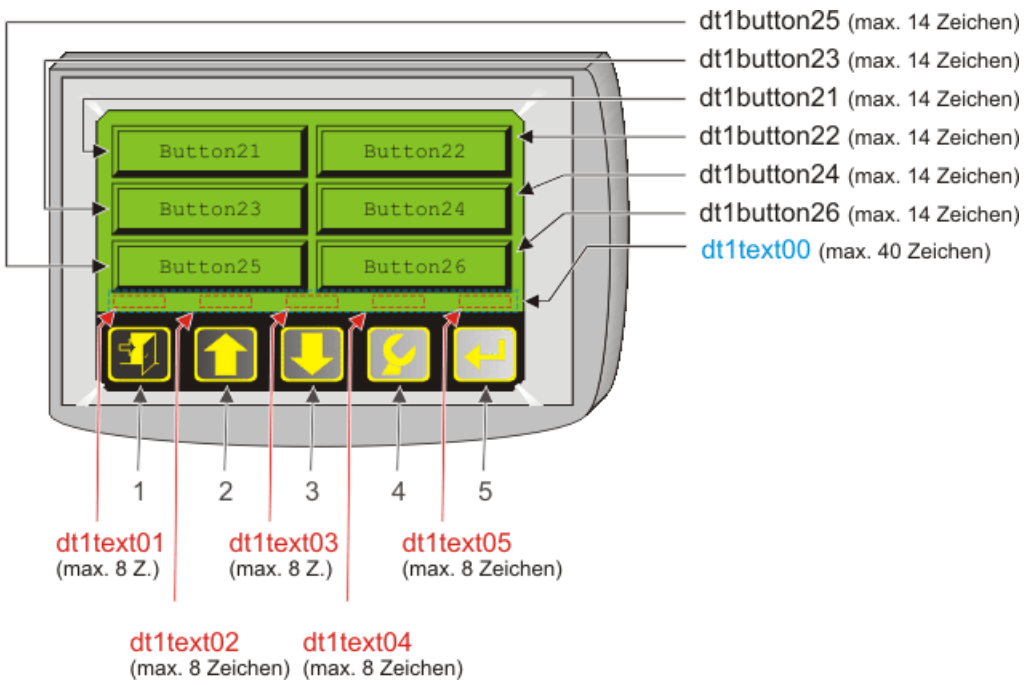
Das DATCOM 640T1T bietet Aufgrund des großen Displays mit einer sichtbaren Fläche von 11x7cm, eine optimale Lösung für Fahrzeuge (Aufträge, Telematik uvm.). Der Aufbau der einzelnen Menüs und wie eine Anpassung geschieht wird im Folgenden erklärt.

Im Folgenden werden die nutzbaren Menüs gezeigt:

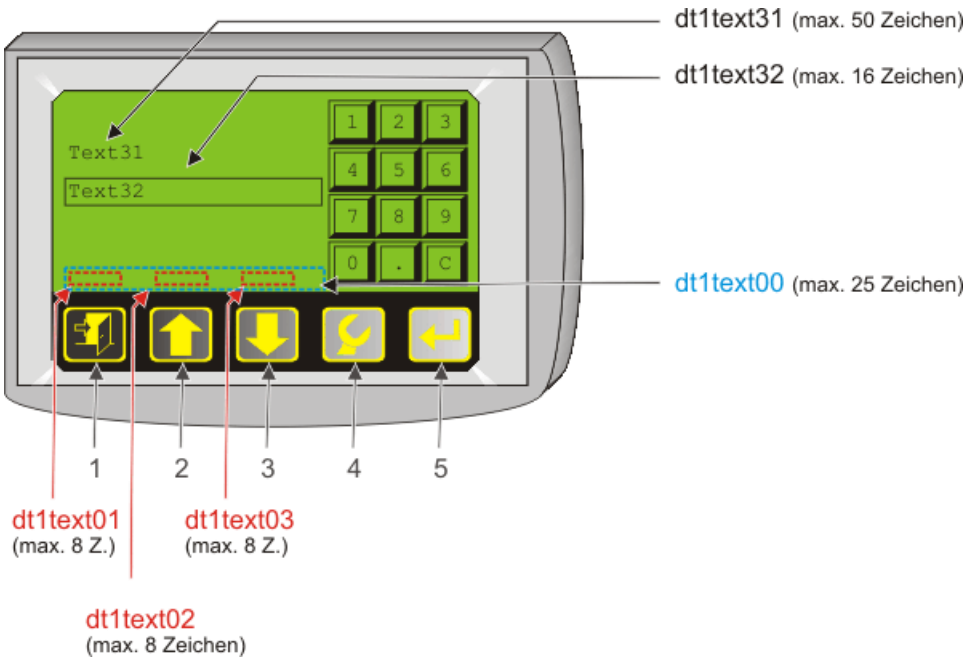
Menü 1



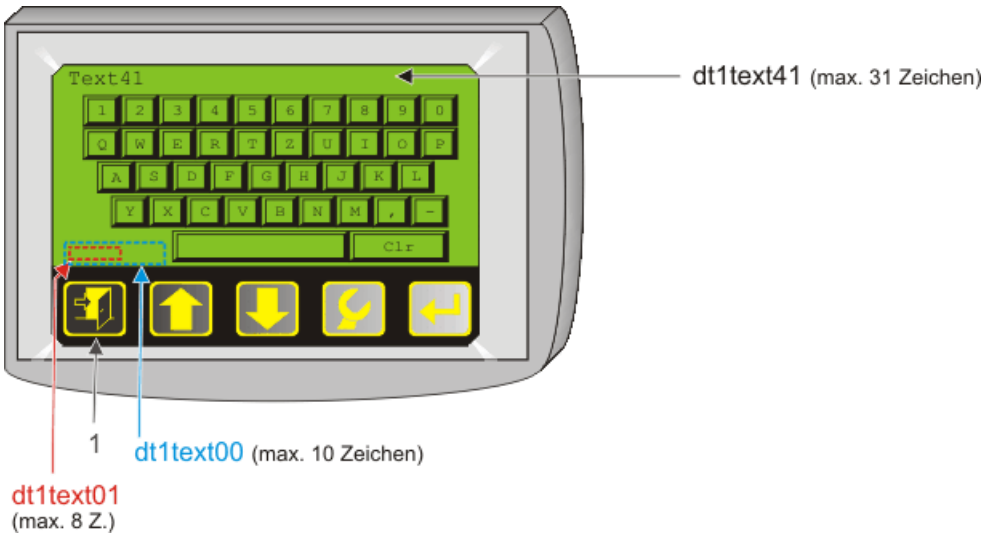
Menü 2


dt1button25 (max. 14 Zeichen)
dt1button23 (max. 14 Zeichen)
dt1button21 (max. 14 Zeichen)
dt1button22 (max. 14 Zeichen)
dt1button24 (max. 14 Zeichen)
dt1button26 (max. 14 Zeichen)
dt1text00 (max. 40 Zeichen)
Button21
Button22
Button23
Button24
Button25
Button26
1
2
3
4
5
dt1text01
(max. 8 Z.)
dt1text03
(max. 8 Z.)
dt1text05
(max. 8 Zeichen)
dt1text02
(max. 8 Zeichen)
dt1text04
(max. 8 Zeichen)

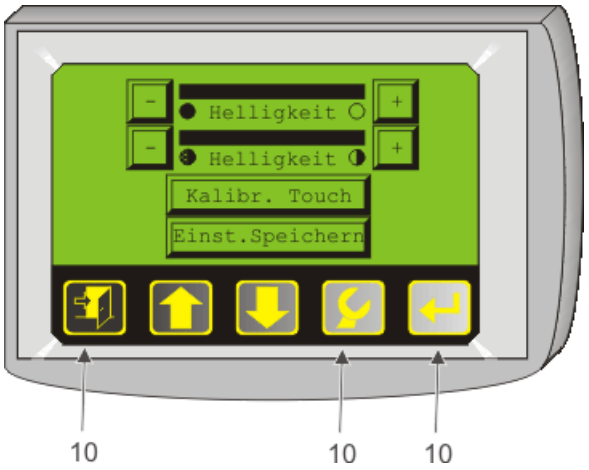
Menü 3


dt1text31 (max. 50 Zeichen)
dt1text32 (max. 16 Zeichen)
dt1text00 (max. 25 Zeichen)
Text31
Text32
1
2
3
4
5
6
7
8
9
0
.
C
1
2
3
4
5
dt1text01
(max. 8 Z.)
dt1text03
(max. 8 Z.)
dt1text02
(max. 8 Zeichen)

Menü 4



Menü 5 (Konfiguration)



Bevor das DATCOM 640T1T genutzt werden kann, muss die Schnittstelle an der das DATCOM 640T1T genutzt werden soll, auf das entsprechende Protokoll gesetzt werden:

```
set com1 dt1
```

Bei dem S-System muss auch der Bezeichner com1 benutzt werden, wenn das DATCOM 640T1T auf Data1 betrieben wird (Com3 für Data2).

Die Menüs können mit freien Texten versehen werden um zum Beispiel Aufträge sichtbar zu machen. Die Schalter können zum Beispiel mit folgender Anweisung gesetzt oder gelöscht werden, dabei gibt die Nummer an, welcher Schalter gesetzt wird:

```
set dt1button21 "Tür auf" rem 21 steht für Schalter 21 im Menü 2
```

Anschließend muss die Anweisung show dt1menue benutzt werden, um das entsprechende Menü anzuzeigen:

```
show dt1menue2 rem 2 steht für das Menü 2
```

Wird ein Schalter dynamisch verändert so kann einfach show dtmenue1 ausgeführt werden, ohne die anderen Schalter erneut zu beschreiben.

Die Variable dt1key kann genutzt werden, um die zuletzt betätigte Taste zu erfassen:

```
if (dt1key=21)
  rem *** Schalter 21 wurde betätigt ***
  set dt1key 0 rem Damit nicht ein zweites mal bearbeitet
endif
```

Die Symbole am unteren Rand geben die Zahl 1 bis 5 zurück (lediglich das Konfigurationsmenü gibt den Tastenwert 10 zurück).
Um den Benutzer auf eine Veränderung aufmerksam zu machen kann ein interner Signalgeber verwendet werden:

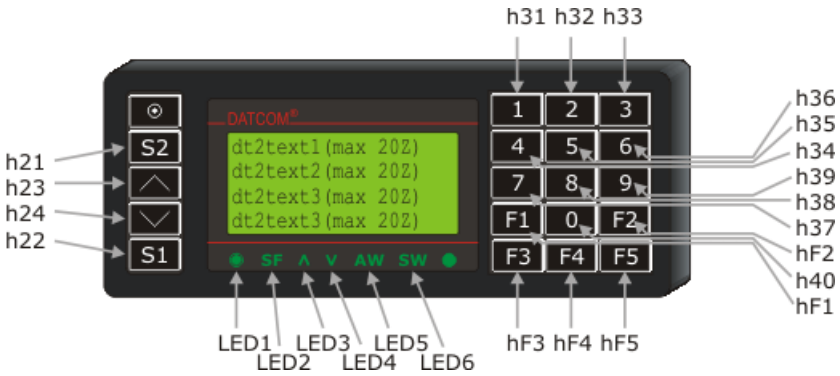
```
set dt1beep 20 rem Angabe im 50ms-Schritten (hier 1 Sekunde)
```

Um eine besser Vorstellung von der Nutzung zu machen werden im Folgenden ein Quellcode und das Aussehen des DATCOM 640T1T beschrieben:

```
set com1 dt1
set dt1button11 "Weiter"
set dt1button14 "Annehmen"
set dt1button15 "Ablehnen"
show dt1menue1
while (1=1)
  if                    (dt1key=14)
    set dt1text11 "Angenommen"
    show dt1menue1
  endif
  if                    (dt1key=15)
    set dt1text11 "Abgelehnt"
    show dt1menue1
  endif
wend
```

Terminal DATCOM 80TF4

Das Terminal DATCOM 80TF4 kann durch seinen einfachen Aufbau zum Beispiel optimal für Unternehmen mit häufig wechselndem Fahrer oder bei Erfassung von einfachen Status benutzt werden.



Bevor das DATCOM 80TF4 genutzt werden kann, muss die Schnittstelle an der das DATCOM 80TF4 genutzt werden soll, auf das entsprechende Protokoll gesetzt werden:

```
set com1 dt2
```

Bei dem S-System muss auch der Bezeichner com1 benutzt werden, wenn das DATCOM 80TF4 auf Data1 betrieben wird (Com3 für Data2).

Die obige Skizze zeigt welche Tasten welchen Tastaturwert zurückgeben. Der Tastenwert kann durch die Variable dt2key wie folgt abgefragt werden:

```
if (dt2key=h24) rem Pfeil abwärts gedrückt
   set dt2key 0 rem Tastenwert zurücksetzen, sonst mehrfach Auswertung
   rem *** Taste bearbeiten ***
endif
```

Um Text auf dem Display darzustellen können die Variable dt2text1 bis dt2text4:

```
set dt2text1 "Text ausgeben auf"
set dt2text1 "DATCOM 80TF4"
```

Um den Benutzer auf eine Veränderung aufmerksam zu machen kann ein interner Signalgeber verwendet werden:

```
set dt2beep 20 rem Angabe im 50ms-Schritten (hier 1 Sekunde)
```

Eingabe über mehrere Zeichen kann mit der Variable dt2input abgefragt werden:

```
if (dt2input="1234") rem Es wurde 1234 auf der Tastatur eingegeben
  set dt2input "" rem Eingabe zurücksetzen
  rem *** Tastenfolge bearbeiten ***
endif
```

Um die LEDs am unteren Rand anzusteuern, muss die entsprechende Variable dt2led1 bis dt2led6 gesetzt werden:

```
set dt2led2 1 rem LED "SF" einschalten
wait 20 rem Kurz wartet
set dt2led2 0 rem LED "SF" wieder ausschalten
```

Über die Variable dt2contrast kann der aktuelle Kontrast bestimmt werden:

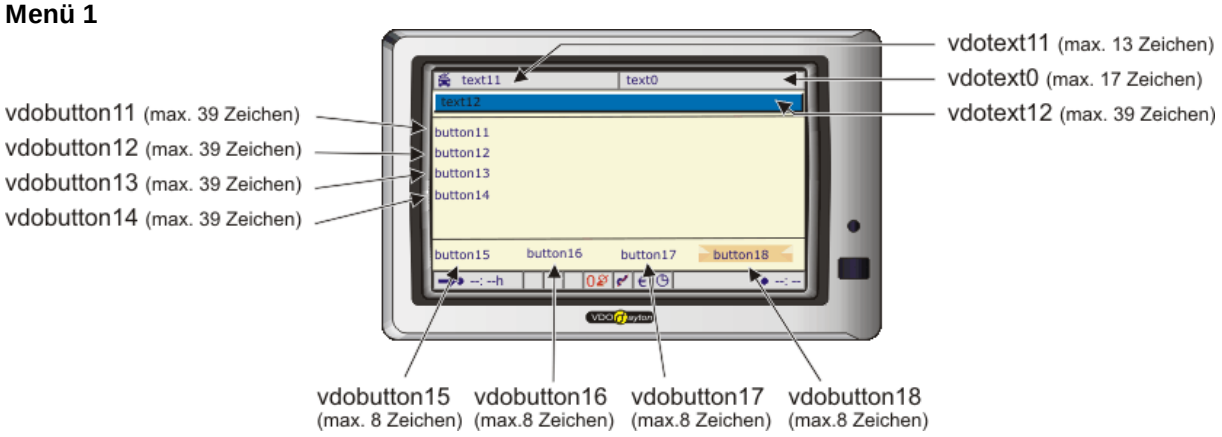
```
set dt2contrast 1 rem Sehr hell
set dt2contrast 6 rem Sehr dunkel
```

Navigationssystem VDO Dayton

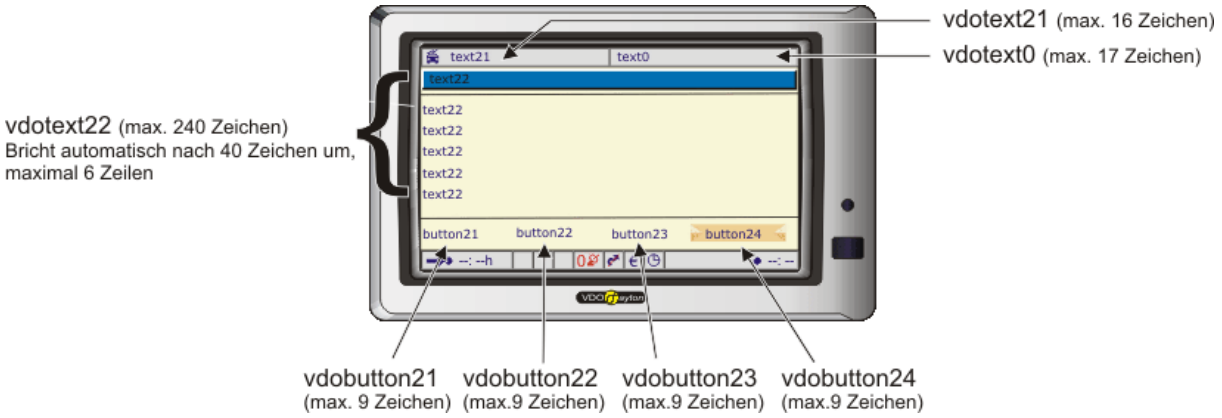
Das VDO Dayton-Navigationssystem bildet zusammen mit dem H-/S-System ein umfangreiches Auftragssystem bei gleichzeitiger Nutzung der Telemetrie.

Startmenü

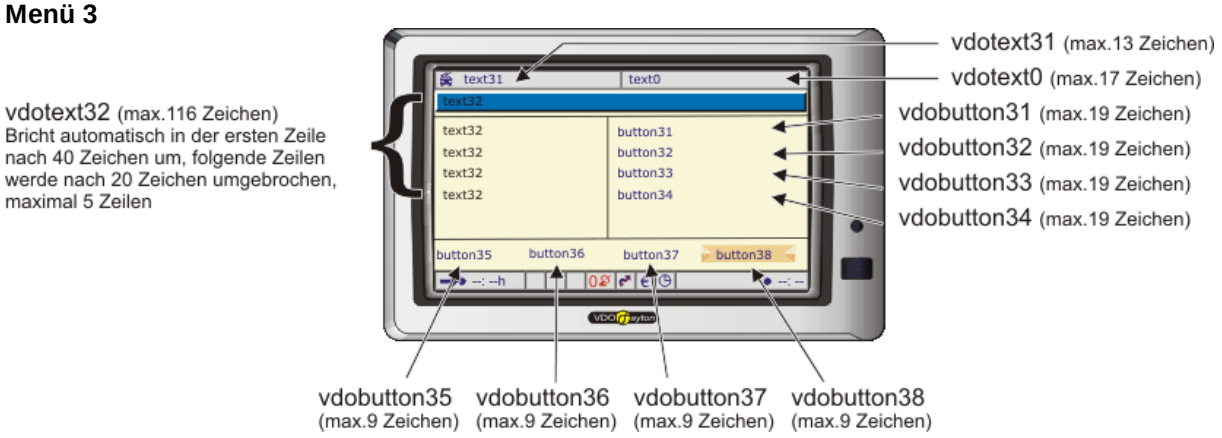
Menü 1



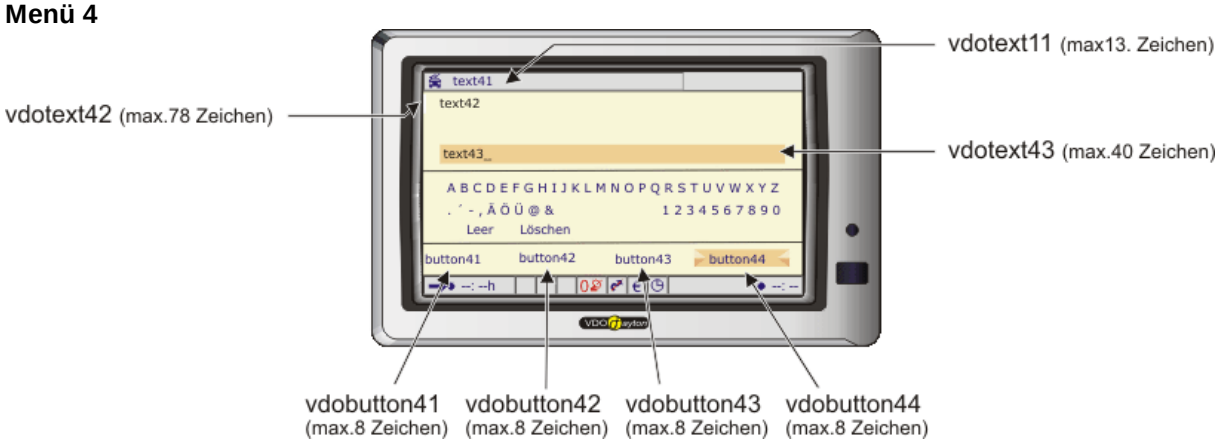
Menü 2



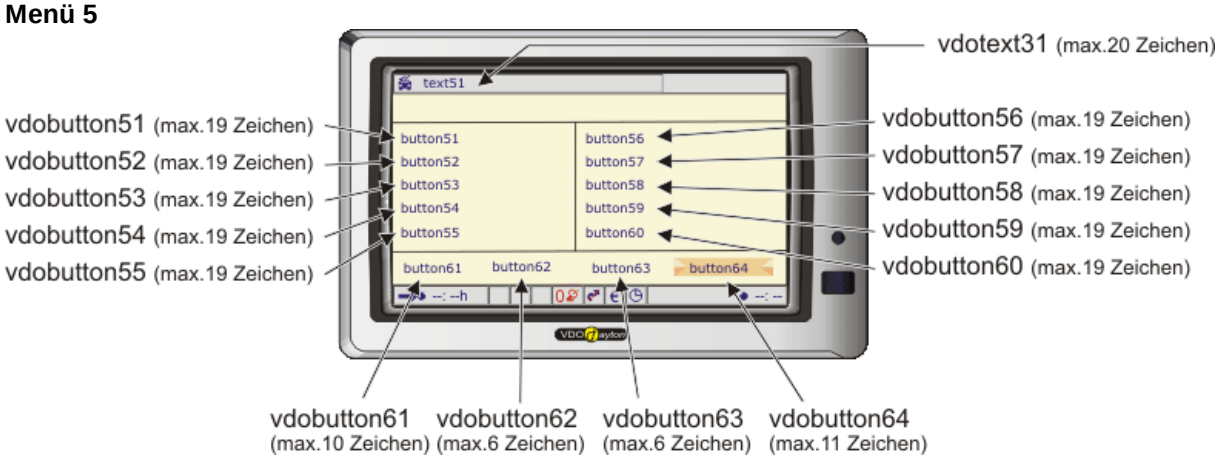
Menü 3



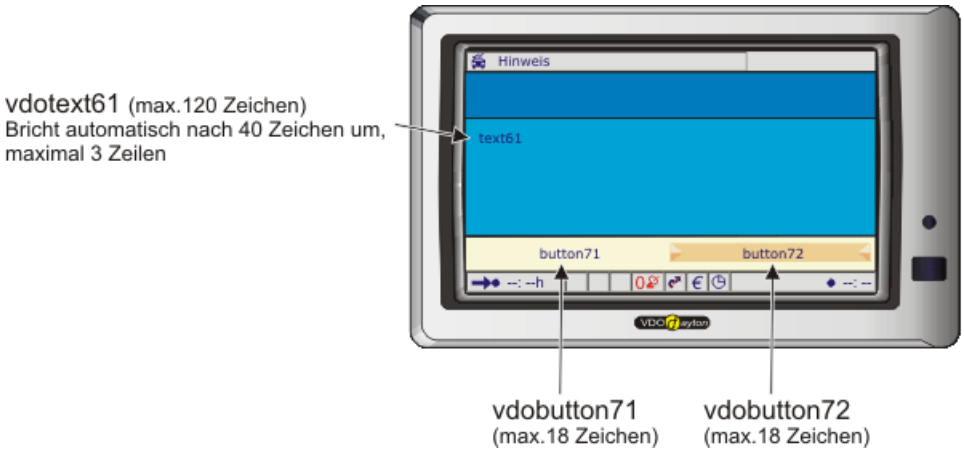
Menü 4



Menü 5



Menü 7



Bevor das VDO Dayton genutzt werden kann, muss die Schnittstelle an der das VDO Dayton genutzt werden soll, auf das entsprechende Protokoll gesetzt werden:

```
set com1 vdo-ms5000
```

Bei dem S-System muss auch der Bezeichner com1 benutzt werden, wenn das VDO Dayton auf Data1 betrieben wird (Com3 für Data2).

Die Menüs können mit freien Texten versehen werden um zum Beispiel Aufträge sichtbar zu machen. Die Schalter können zum Beispiel mit folgender Anweisung gesetzt oder gelöscht werden, dabei gibt die Nummer an, welcher Schalter gesetzt wird:

```
set vdobutton21 "Tür auf" rem 21 steht für Schalter 21 im Menü 2
```

Anschließend muss die Anweisung „show vdomenue“ benutzt werden, um das entsprechende Menü anzuzeigen:

```
show vdomenue2 rem 2 steht für das Menü 2
```

Um ein Menü zu verlassen, kann die Anweisung „close vdomenue“ benutzt werden:

```
close vdomenue2 rem 2 steht für das Menü 2
```

HS Es ist zu beachten das nur zwei Menüs nacheinander geöffnet und wieder geschlossen werden können.

Bei einem Aufruf eines dritten Menüs wird das erste Menü verworfen. Soll eine Menüstruktur die eine Tief von mehr als zwei Menüs aufweist, so ist zu empfehlen vor dem Umblenden auf ein neues Menü das aktuelle zu schließen.

Durch die Anweisung show kann die aktuelle Karte angezeigt werden:

```
show vdomap
```

Die Variable vdokey kann genutzt werden, um die zuletzt betätigte Taste zu erfassen:

```
if (vdokey=21)
  rem *** Schalter 21 wurde betätigt ***
  set vdokey 0 rem Eine zweite Auswertung verhindern
endif
```

Es besteht die Möglichkeit durch eine SMS eine Zielführung zu veranlassen. Das Skript kann durch abfragen der Variable recroute erfassen, ob eine SMS mit Zielführungsinformation empfangen wurde:

```
if (recroute)
  rem Neues Ziel wurde festgelegt
  set vdotext11 "Neues Ziel!"
endif
```

Ist ein neues Ziel mit seinen Koordinaten bekannt, so kann durch die Anweisung route eine Zielführung gestartet werden:

```
route 50.343800 9.53048
…
var1 = 51.345254
var2 = 8.453586
route var1 var2
```

Um von der Entfernung zum Ziel abhängige Funktionen zu realisieren, kann die Variable „vdodist“ genutzt werden (der Wert wird in 100m-Schritte zurückgegeben):

```
if (vdodist<10)
  rem Weniger als 1000m von dem Ziel entfernt
  setport5 rem Ausgang 5 schalten
endif
```

Ist ein VDO Dayton mit einem H-/S-System verbunden, so bestimmt das VDO Dayton die GPS-Position. Soll hingegen der interne GPS-Empfänger des H-/S-Systems genutzt werden, dann kann folgende Anweisung genutzt werden:

```
set gpsms5000 0 rem Interne GPS-Daten nutzen
set gpsms5000 1 rem GPS-Daten von dem VDO Dayton benutzen
```

Durch die DATCOM Fleet Software kann direkt eine Zieladresse übermittelt werden. Das Dayton berechnet nach erhalt der Zieladresse automatisch den neuen Weg. Zusätzlich zu der Adresse kann noch ein Text mitangeben werden, der durch die Variable routeinfo abgefragt werden kann:

```
set string0 routeinfo rem Mitgesendete Daten in String0 legen
```

Navigationssystem Garmin

Das GARMIN-Navigationssystem bildet zusammen mit dem H-/S-System ein Auftragssystem bei gleichzeitiger Nutzung der Telemetrie.

Bevor das GARMIN-Navigationsgerät genutzt werden kann, muss die Schnittstelle an der das GARMIN genutzt werden soll, auf das entsprechende Protokoll gesetzt werden:

```
set com1 garmin
```

Durch die Anweisung garconnected kann erkannt werden ob das Garmin mit der H-/S-Box verbunden ist:

```
if (garconnected) rem Garmin wurde erkannt
   set string0 "GARMIN erkannt"
else
   set string0 "GARMIN nicht erkannt"
endif
```

Im Garmin kann eine Fahrerkennung mit garsetdriverid vorgegeben:

```
garsetdriverid "FahrerXY" rem Fahrerkennung wird auf FahrerXY gesetzt
```

Im Garmin kann man mit garsetdrvstate einen Fahrerstatus eine Bezeichnung vorgegeben, die dann im Garmin angezeigt wird:

```
garsetdrvstate1 "Pause" rem Fahrerstatus1 wird mit Pause ausgegeben
garsetdrvstate2 Ruhe" rem Fahrerstatus2 wird mit Ruhe ausgegeben
garsetdrvstate3 Bereit" rem Fahrerstatus3 wird mit Bereit ausgegeben
garsetdrvstate4 "Fahrt" rem Fahrerstatus4 wird mit Fahrt ausgegeben
```

Kurznachrichten können mit dem Befehl garsetquickmsg an das Garmin übergeben werden, die dann im Menue Kurznachriten zuverfügung stehen:

```
garsetquickmsg 1 "Stau 1h" rem bei Kurztext1 wird Stau 1h gesendet
garsetquickmsg 2 "Stau 2h" rem bei Kurztext1 wird Stau 2h gesendet
garsetquickmsg 3 "Panne" rem bei Kurztext1 wird Panne gesendet
```

Um zu ermitteln ob die Daten übergeben wurden kann man dies mit garresult überprüfen:

```
if (garresult) rem Daten wurde an Garmin übergeben
```

Um Nachrichten ans GARMIN zu übergeben können folgende Befehle genutzt werden.

```
garsendtext "GPS-Empfang ok" rem Text ohne Rückmeldung
garsendoktext "Zurück kommen" rem Text mit OK-Rückmeldung
garsendyesnotext "Kunde angetroffen" rem Text mit Ja/Nein-Rückmeldung
```

Soll eine Zielführung ans Garmin übergeben werden so benutzt man den Befehl garsendroute wobei erst die Koordinate, danach die TAN und danach die Nachricht im übergebenen Text stehen muß:

```
garsendroute "50.123 9.123 12345 Fahre nach X-Dorf" rem TAN ist 12345
```

H S Beachten Sie, dass bei den Koordinaten kein Komma sondern ein Punkt verwendet wird.

Mit `garsetroutestate` kann der Status der Zielführung verändert werden:

```
garsetroutestate "12345" 1 rem Zielführung erledigt
garsetroutestate "12345" 2 rem Zielführung aktiviert
garsetroutestate "12345" 3 rem Zielführung löschen
```

In der Textvariablen `garmessage` werden die Textnachrichten vom GARMIN übergeben:

```
set string0 garmessage rem Nachricht wird in den string0 geschrieben
tcpwrite garmessage rem Nachricht wird in den Sendespeicher geschrieben
```

Durch `garevent` werden Statusänderungen an die H-/S-Box übergeben. Dabei ist der erste Buchstabe die Aktion und danach folgt der betreffende Text:

```
if (garevent= „o“) rem Eine OK-Nachricht wurde bestätigt
  set string0 mid 2 16 garevent rem in string0 wird die TAN der
Nachricht geschrieben
```

Aktion	Beschreibung
y	`Garsendyesnotext` mit Ja beantwortet (y=yes)
n	`garsendyesnotext` mit Nein beantwortet (n=no)
o	`garsendoktext` mit OK bestätigt (o=ok)
d	Auftrag ist erledigt (d=done)
a	Auftrag ist aktiviert (a=active)
u	Auftrag ist empfangen aber ungelesen (u=unread)
i	Auftrag ist gelesen aber nicht aktiv (i=inactive)
r	Auftrag ist gelöscht (r=removed)
s	Fahrerstatus wurde geändert (s=driver state)
I	Fahrername wurde geändert (I=Fahrer ID)

Bedienelement DATCOM TaBo 4

Das DATCOM TaBo4 dient als einfaches Bedienteil für Steuerungsaufgaben. Durch seine zahlreichen LEDs können verschiedene Statis angezeigt werden und durch den akustischen Signalgeber kann auf solche Änderungen hingewiesen werden.

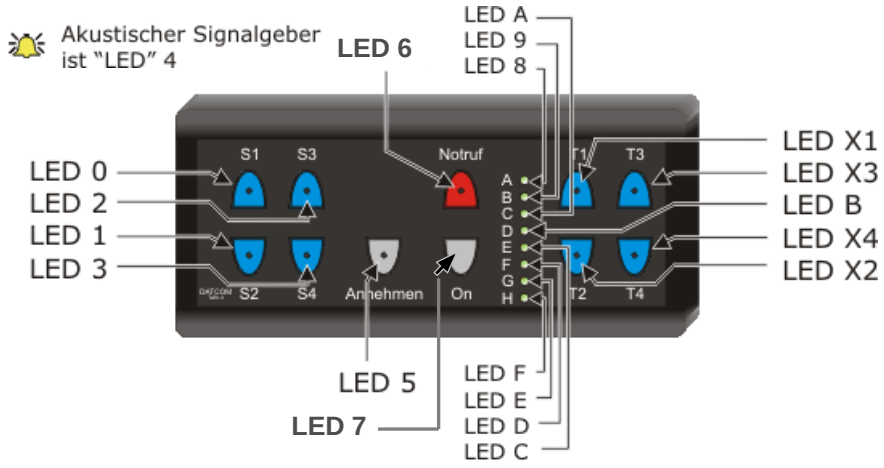
Bevor das DATCOM TaBo4 genutzt werden kann, muss die Schnittstelle an der das DATCOM TaBo4 genutzt werden soll, auf das entsprechende Protokoll gesetzt werden:

```
set com1 tb3 rem Hier tabo3 angeben, da Protokoll gleich mit dem
             rem Vorgänger
```

Die einzelnen LEDs können durch folgende Anweisung angesprochen werden:

```
set tb3cmd h32 rem LED 3 mit 2Hz blinken
```

Die LEDs haben folgende Bezeichnungen:



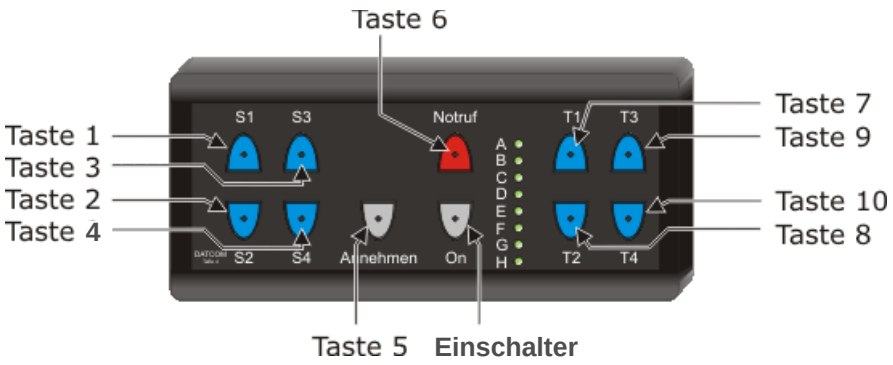
In der folgenden Tabelle werden die Werte für die Anweisung tb3cmd beschrieben:

LED	Ausschalten	Einschalten	2Hz	1Hz	0,5Hz
LED 0	h00	h01	h02	h03	h04
LED 1	h10	h11	h12	h13	h14
LED 2	h20	h21	h22	h23	h24
LED 3	h30	h31	h32	h33	h34
Signalgeber	h40	h41	h42	h43	h44
LED 5	h50	h51	h52	h53	h54
LED 6	h60	h61	h62	h63	h64
LED 7	h70	h71	h72	h73	h74
LED 8	h80	h81	h82	h83	h84
LED 9	h90	h91	h92	h93	h94
LED A	hA0	hA1	hA2	hA3	hA4
LED B	hB0	hB1	hB2	hB3	hB4
LED C	hC0	hC1	hC2	hC3	hC4
LED D	hD0	hD1	hD2	hD3	hD4
LED E	hE0	hE1	hE2	hE3	hE4
LED F	hF0	hF1	hF2	hF3	hF4
LED X1	h08	h09	h0A	h0B	h0C
LED X2	h18	h19	h1A	h1B	h1C
LED X3	h28	h29	h2A	h2B	h2C
LED X4	h38	h39	h3A	h3B	h3C

Das abfragen der Tasten kann über die Variable tb3key oder tb3keyup geschehen:

```
if (tb3key=6) rem Notruftaste betätigt
  set tb3key 0 rem Um mehrfach auswerten zu verhindern
  rem *** Taste bearbeiten ***
endif
if (tb3keyup=6) rem Notruftaste wurde losgelassen
  set tb3keyup 0 rem Um mehrfach auswerten zu verhindern
  rem *** Taste bearbeiten ***
endif
```

Die Tastenwerte sind wie folgt definiert:

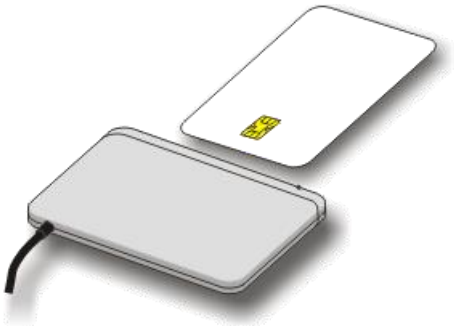


Um zu überprüfen ob das DATCOM TaBo4 angeschlossen ist, kann die Variable tb3connect genutzt werden:

```
if (tb3connect)
  set tb3cmd h22
else
  rem Meldung an die Zentrale: TaBo4 entfernt!
endif
```

Kartenleser

Der Kartenleser kann an jeder beliebigen Com/Data-Schnittstelle betrieben werden. Die Karten können mit individuellen Texten durch DATCOM Fleet versehen werden, so dass zum Beispiel eine Unterscheidung zwischen Service und Fahrerkarte durchgeführt werden kann. Der interne Speicher der Karte kann zum mitschreiben verschiedener Statis genutzt werden.



Bevor der Kartenleser genutzt werden kann, muss die Schnittstelle an der der Kartenleser genutzt werden soll, auf das entsprechende Protokoll gesetzt werden:

```
set com1 chipdrive
```

Die Kennung der Karte kann über die Variable chipdrive erfasst werden, wenn diese Variable leer ist, dann wurde keine Karte gesteckt:

```
if (len chipdrive=0)
  rem *** Keine Karte gesteckt ***
endif
if (chipdrive="Fahrer")
  rem *** Reagieren auf Fahrer-Karte ***
endif
```

Durch die Anweisung write chipdrive können verschieden Angaben auf der eingesteckten Karte gesichert werden:

```
write chipdrive h97 rem Aktuelle Position auf Karte speichern
```

Für den ersten Parameter können folgende Parameter benutzt werden:

Parameter	
hF7	Speichert eine Fahreranmeldung auf der Karte
h97	Speichert die aktuelle Position ab (GPS)
h94	Speichert die Änderung eines Schalteinganges. Hier muss hinter h94 die Angabe des Schalteinganges angegeben werden:

Eingang	Aktiv	Inaktiv
1	h01	h00
2	h11	h10
3	h21	h20
4	h31	h30
5	h41	h40
6	h51	h50
7	h61	h60
8	h71	h70
9	h81	h80
10	h91	h90
11	hA1	hA0
12	hB1	hB0
13	hC1	hC0
14	hD1	hD0
15	hE1	hE0
16	hF1	hF0

Zum Beispiel:

```
write chipdrive h94 h21 rem Schreibt "Eingang 3 aktiv" auf Karte
```

RFid-Leser Legic (Nur S-System)

Der RFid-Leser kann an der Data2-Schnittstelle betrieben werden. Die RFid-Transponder ist vom Hersteller mit einem festen Zahlencode versehen, so dass zum Beispiel eine Unterscheidung zwischen Service und Fahrerkarte durchgeführt werden kann.

Bevor der Kartenleser genutzt werden kann muss die Schnittstelle auf das entsprechende Protokoll gesetzt werden:

```
set comA legicreader
```

Die Kennung der Karte kann über die Variable lcrdata erfasst werden, wenn diese Variable leer ist, dann wurde keine Karte erkannt:

```
if (lcrdata) rem es wurden Daten vom RFid-Leser empfangen
  set extvar0 lcrid rem Daten werden in extvar0 abgelegt
endif
```

Proxi-Pen

Der Proxi-Pen ist ein Gerät das RFID-Tags lesen kann und diese in einem internen Speicher protokolliert. Die Basis-Station kann die protokollierten Daten, nach auflegen des Proxi-Pens, an das H-/S-System weiterreichen. Das H-/S-System kann diese Daten zum Beispiel an die Zentrale schicken.

Um die Proxi-Pen-Basis-Station nutzen zu können, muss die entsprechende serielle Schnittstelle auf das Proxi-Pen-Protokoll eingestellt werden:

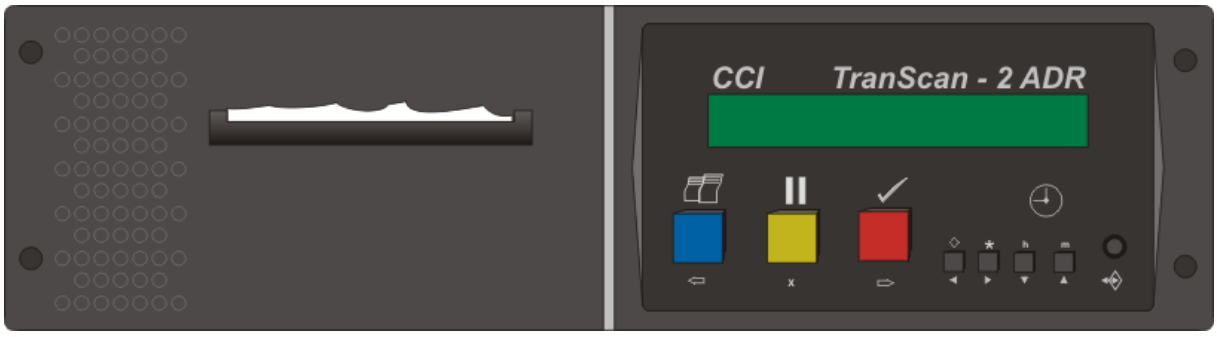
```
set com1 proxipen
```

Das H-/S-System liest beim auflegen des Proxi-Pens auf die Basis-Station automatisch die Daten in eine „.ppr“-Datei ab. Diese Datei kann zum Beispiel über filefind gefunden werden:

```
set string0 filefind "*.ppr"
if (len string0>)
  rem Proxi-Pen-Datei gefunden
endif
```

TranScan

TranScan ist ein Temperaturschreiber mit Drucker. Das Gerät bietet die Möglichkeit die Temperatur von vier Temperaturfühlern zu erfassen und vier digitale Eingänge zu nutzen.



Bevor der TranScan benutzt werden kann, muss das entsprechende Protokoll an der seriellen Schnittstelle aktiviert werden, dabei bezeichnet der Parameter hinter transcan wie viele Fühler angeschlossen sind:

```
set com1 transcan 3
```

Durch die Variable tstemp kann die aktuelle Temperatur des entsprechenden Fühlers gelesen werden (die Nummer des Fühlers wird als Index angegeben):

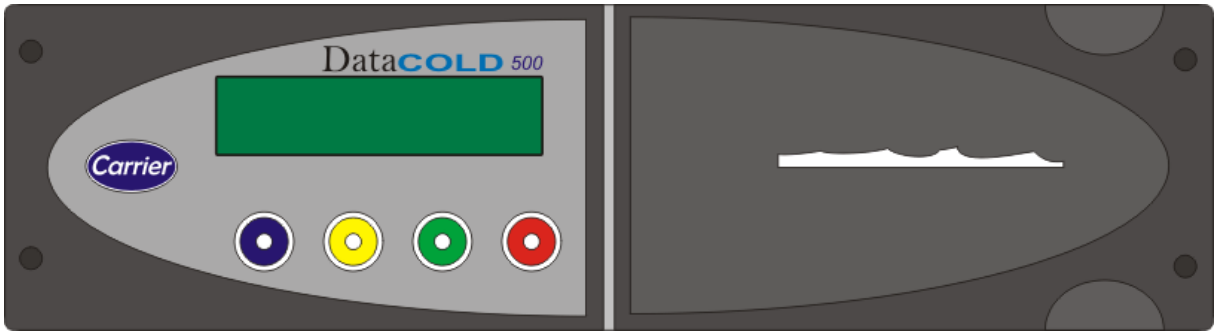
```
if (tstemp1>20)
  rem Meldung: Kühlung ausgefallen!
endif
```

Die Eingänge an der Hinterseite des TranScan können mit der Variable tsport ausgelesen werden. Beachten Sie, dass die Eingänge entsprechend dem TranScan-Handbuch zuvor konfiguriert werden müssen:

```
if (tsport & 1)
  rem Eingang 1 wurde aktiviert
endif
```

DataCold

DataCold ist ein Temperaturschreiber mit Drucker. Das Gerät bietet die Möglichkeit die Temperatur von vier Temperaturfühlern zu erfassen und vier digitale Eingänge zu nutzen.



Bevor der DataCold benutzt werden kann, muss das entsprechende Protokoll an der seriellen Schnittstelle aktiviert werden, dabei bezeichnet der Parameter hinter datacold wie viele Fühler angeschlossen sind:

```
set com1 datacold 3
```

Durch die Variable dctemp kann die aktuelle Temperatur des entsprechenden Fühlers gelesen werden (die Nummer des Fühlers wird als Index angegeben):

```
if (dctemp1>20)
  rem Meldung: Kühlung ausgefallen!
Endif
```

Die Eingänge an der Hinterseite des DataCold können mit der Variable dcport ausgelesen werden. Beachten Sie, dass die Eingänge entsprechend dem DataCold-Handbuch zuvor konfiguriert werden müssen:

```
if (dcport & 1)
  rem Eingang 1 wurde aktiviert
endif
```

Der aktuelle Betriebsmodus kann durch die Variable dcmode erfasst werden (die Bedeutung der einzelnen Modi sind im DataCold-Handbuch nach zu lesen):

```
if (dcmode=0)
  rem Auf Modus 0 reagieren
endif
```

Über die Variable dcdefrost kann erfasst werden, welche Kammer gerade abgetaut wird:

```
if (dcdefrost & 1)
  rem Kammer 1 wird abgetaut
endif
```

Im DataCold können Alarmwerte hinterlegt werden, die durch die Variable dcsetpoint im Skript verwendet werden können:

```
if (dcsetpoint1)
  rem Reagieren auf "Temperaturalarm 1"
endif
```

ELPRO Datenlogger

Der Datenlogger ist ein Temperaturschreiber der Temperaturdaten erfasst und speichert.

Bevor der ELPRO-Datenlogger benutzt werden kann, muss das entsprechende Protokoll an der seriellen Schnittstelle aktiviert werden, dabei bezeichnet der Parameter hinter elpro wie viele Fühler ausgewrtet werden:

```
set com1 elpro 3
if (elpro)
  rem Datenlogger ist angeschlossen
Endif
```

Durch die Variable elptemp kann die aktuelle Temperatur des entsprechenden Fühlers gelesen werden (die Nummer des Fühlers wird als Index angegeben):

```
set string1 elptemp1
set string2 elptemp2
```

Die Eingänge an der Rückseite des Datenloggers können mit der Variable elpport ausgelesen werden. Beachten Sie, dass die Eingänge entsprechend dem ELPRO-Handbuch zuvor konfiguriert werden müssen:

```
set mem1 elpport
```

DTCO 1381 (Digitaler Tachograph)

Durch Nutzung des S-Systems an dem digitalen Tachograph ist es möglich Kilometerstände, Geschwindigkeiten und Status der Fahrer/Beifahrer an die Zentrale zu übermitteln.

Bevor das DTCO genutzt werden kann, muss das entsprechende Protokoll für die zu nutzende Schnittstelle eingestellt werden:

```
set com1 dtco
```

Da während der Abfrage von Variablen auf die gleichen Daten zugegriffen wird, muss über die Anweisung dtcoupdate zuerst die neuen Werte des DCTO in die Variablen kopiert werden:

```
dtcoupdate rem Neue Werte des DTCO übernehmen
```

Ob gültige Daten vorliegen (zum Beispiel wenn das DTCO noch nicht betriebsbereit ist) kann über die Variable dctodata abgefragt werden:

```
dtcoupdate rem Neue Werte des DTCO übernehmen
if (dtcodata)
  rem Es sind gültige Daten vorhanden
endif
```

Im folgenden Skript werden die einzelnen abfragbaren Daten des DTCOs erläutert:

```
dtcoupdate
if (dtcodata)
  set string0 ""

  rem Aktuelle Geschwindigkeit als Gleitkommazahl
  add string0 dtco_speed " km/h "

  rem Drehzahl des Motors als Gleitkommazahl
  add string0 engine_speed " Umdrehung/Minute "

  rem Kilometerstand als Gleitkommazahl
  add string0 total_vehicle_dist " km "

  rem Kilometer der aktuellen Fahrt als Gleitkommazahl
  add string0 trip_dist " km "

  rem K-Faktor als Gleitkommazahl
  rem (Verhältnis Tachoimpulse zu Meter)
  add string0 k_factor " Pulse/Meter "

  rem Fahrzeugkennung als Zeichenkette
  add string0 vehicle_id " "

  rem Fahrzeugregistrationsnummer als Zeichenkette
  add string0 vehicle_reg " "

  rem Kennung von Fahrer 1 als Zeichenkette
  add string0 driver1_id " "

  rem Kennung von Fahrer 2 als Zeichenkette
  add string0 driver2_id " "

endif
```

Durch die Variable dtco_workstates kann der Status des Fahrers, Beifahrers und des Fahrzeugs erfasst werden. Dabei liegen die Informationen in Bitkodierungen vor:

```
if (dtco_workstates & 7 = 1)
  rem Fahrer 1 ist verfügbar
endif
```

Um die einzelnen Zustände zu Erfassen kann folgende Tabelle genutzt werden. Dabei kann die angegebe Bedienung für eine if-Anweisung genutzt werden:

Bedingung	Beschreibung
(dtco_workstates & 7 = 0)	Fahrer 1 pausiert
(dtco_workstates & 7 = 1)	Fahrer 1 ist verfügbar

(dtco_workstates & 7 = 2)	Fahrer 1 arbeitet
(dtco_workstates & 7 = 3)	Fahrer 1 fährt
(dtco_workstates & 7 = 7)	Fahrer 1 ist nicht verfügbar
(dtco_workstates & 56 = 0)	Fahrer 2 pausiert
(dtco_workstates & 56 = 8)	Fahrer 2 ist verfügbar
(dtco_workstates & 56 = 16)	Fahrer 2 arbeitet
(dtco_workstates & 56 = 24)	Fahrer 2 fährt
(dtco_workstates & 56 = 56)	Fahrer 2 ist nicht verfügbar
(dtco_workstates & 192 = 0)	Fahrzeug im stillstand
(dtco_workstates & 192 = 64)	Fahrzeug ist in Bewegung
(dtco_workstates & 192 = 192)	Bewegungszustand kann nicht abgefragt werden

Fahrt relevante Informationen über Fahrer 1 können auf die gleiche Art und Weise abgefragt werden, wie vorherigen Zustände:

Bedingung	**Beschreibung**
(dtco_driver1_states & 15 = 0)	Keine Zeitwarnung
(dtco_driver1_states & 15 = 1)	15 Minuten bis die Fahrtzeit von 4,5 h erreicht ist
(dtco_driver1_states & 15 = 2)	Fahrtzeit von 4,5 h erreicht
(dtco_driver1_states & 15 = 3)	15 Minuten bis optionale Warnung 1 erreicht ist
(dtco_driver1_states & 15 = 4)	Optionale Warnung 1 erreicht
(dtco_driver1_states & 15 = 5)	15 Minuten bis optionale Warnung 2 erreicht ist
(dtco_driver1_states & 15 = 6)	Optionale Warnung 2 erreicht
(dtco_driver1_states & 15 = 15)	Zeitwarnungen können nicht abgefragt werden
(dtco_driver1_states & 48 = 0)	Karte für Fahrer ist aktiv
(dtco_driver1_states & 48 = 16)	Karte für Fahrer ist nicht vorhanden
(dtco_driver1_states & 48 = 48)	Karteninformation kann nicht abgefragt werden
(dtco_driver1_states & 192 = 0)	Keine Geschwindigkeitsübertretung
(dtco_driver1_states & 192 = 64)	Geschwindigkeitsübertretung liegt vor
(dtco_driver1_states & 192 = 192)	Geschwindigkeitsübertretung kann nicht abgefragt werden

Entsprechend dieser Tabelle können relvante Informationen für den Fahrer 2 wie folgt abgefragt werden:

Bedingung	**Beschreibung**
(dtco_driver2_states & 15 = 0)	Keine Zeitwarnung
(dtco_driver2_states & 15 = 1)	15 Minuten bis die Fahrtzeit von 4,5 h erreicht ist
(dtco_driver2_states & 15 = 2)	Fahrtzeit von 4,5 h erreicht
(dtco_driver2_states & 15 = 3)	15 Minuten bis optionale Warnung 1 erreicht ist
(dtco_driver2_states & 15 = 4)	Optionale Warnung 1 erreicht
(dtco_driver2_states & 15 = 5)	15 Minuten bis optionale Warnung 2 erreicht ist
(dtco_driver2_states & 15 = 6)	Optionale Warnung 2 erreicht
(dtco_driver2_states & 15 = 15)	Zeitwarnungen können nicht abgefragt werden
(dtco_driver2_states & 48 = 0)	Karte für Fahrer ist aktiv
(dtco_driver2_states & 48 = 16)	Karte für Fahrer ist nicht vorhanden
(dtco_driver2_states & 48 = 48)	Karteninformation kann nicht abgefragt werden

Über die Variable dtco_status können Informationen über den DTCO selbst abgefragt werden:

Bedingung	**Beschreibung**
(dtco_status & 3 = 0)	Kein Tachographereignis
(dtco_status & 3 = 1)	Tachographereignis
(dtco_status & 3 = 3)	Tachographereignis kann nicht abgefragt werden
(dtco_status & 12 = 0)	Keine Behandlungsinformation vorhanden
(dtco_status & 12 = 4)	Behandlungsinformation vorhanden
(dtco_status & 12 = 12)	Behandlungsinformation kann nicht abgefragt werden
(dtco_status & 48 = 0)	Normaler Tachographdurchsatz

(dtco_status & 48 = 8)	Tachographdurchsatz analyse
(dtco_status & 48 = 48)	Tachographdurchsatz kann nicht abgefragt werden
(dtco_status & 192 = 0)	Fahrzeug bewegt sich vorwärts
(dtco_status & 192 = 64)	Fahrzeug bewegt sich rückwärts
(dtco_status & 192 = 192)	Fahrzeugrichtung kann nicht abgefragt werden

Die Zustände der Eingänge und des Druckerschachtes können durch die Variable dtco_add_info abgefragt werden:

Bedingung	**Beschreibung**
(dtco_add_info & 3 = 0)	Eingang D1 ist auf 0
(dtco_add_info & 3 = 1)	Eingang D1 ist auf 1
(dtco_add_info & 3 = 2)	Eingang D1 weist einen Fehler auf
(dtco_add_info & 3 = 3)	Eingang D1 kann nicht abgefragt werden
(dtco_add_info & 12 = 0)	Eingang D2 ist auf 0
(dtco_add_info & 12 = 4)	Eingang D2 ist auf 1
(dtco_add_info & 12 = 8)	Eingang D2 weist einen Fehler auf
(dtco_add_info & 12 = 12)	Eingang D2 kann nicht abgefragt werden
(dtco_add_info & 48 = 0)	Zündung an
(dtco_add_info & 48 = 16)	Zündung aus
(dtco_add_info & 48 = 32)	Zündung fehlerhaft
(dtco_add_info & 48 = 48)	Zündung kann nicht abgefragt werden
(dtco_add_info & 192 = 0)	Druckerschacht offen
(dtco_add_info & 192 = 64)	Druckerschacht geschlossen
(dtco_add_info & 192 = 128)	Druckerschacht weist einen Fehler auf
(dtco_add_info & 192 = 192)	Druckerschacht kann nicht abgefragt werden
(dtco_add_info & 1792 = 0)	Kein Modus aktiv
(dtco_add_info & 1792 = 256)	Operationeller Modus aktiv
(dtco_add_info & 1792 = 512)	Kontrollmodus aktiv
(dtco_add_info & 1792 = 768)	Kalibrierungsmodus aktiv
(dtco_add_info & 1792 = 1024)	Unternehmensmodus aktiv

Kapitel 6: Erweiterungsmodule für das H-System

Das H-System bietet umfangreiche Erweiterung für Ein-/Ausgänge und spezielle Einsatzgebiete.

Ein-/Ausgabeerweiterung

Durch dieses Modul wird das H-System um zusätzlichen Ein- und Ausgaben erweitert. Die Ausgänge 1 und 2 können wesentlich höhere Belastet werden als die Ausgänge des Hauptmoduls, daher wird hier kein Adapter benötigt um die Belastbarkeit zu erhöhen.

Leiste	Name	Technische Daten	Beschreibung	
38	GND			
37	UB +5V			
36	Ein/Ausgang x16	max. 30mA	Alle Anschlüsse können wahlweise als Ausgang oder als Eingang genutzt werden. Alle Ausgänge werden als „open collector“ betrieben (schalten nach GND).	
35	Ein/Ausgang/Temp x15	max. 30mA		An diese digitalen Eingänge kann ein digitaler Temperatursensor angeschlossen werden.
34	Ein/Ausgang/Temp x14	max. 30mA		
33	Ein/Ausgang/Temp x13	max. 30mA		
32	Ein/Ausgang/Temp x12	max. 30mA		
31	Ein/Ausgang x11	max. 30mA		Die Änderungen an diesen Eingängen kann in DATCOM Fleet angezeigt werden (ine, siehe Seite).
30	Ein/Ausgang x10	max. 30mA		
29	Ein/Ausgang x9	max. 30mA		
28	Ein/Ausgang x8	max. 30mA		
27	Ein/Ausgang x7	max. 30mA		
26	Ein/Ausgang x6	max. 30mA		
25	Ein/Ausgang x5	max. 30mA		
24	Ein/Ausgang x4	max. 30mA		
23	Ein/Ausgang x3	max. 30mA		
22	Ein/Ausgang x2	max. 100mA		
21	Ein/Ausgang x1	max. 100mA		

Die Ausgänge können mit der Anweisung setportx**n** und clrportx**n** gesetzt bzw. zurückgesetzt werden:

```
setportx1 rem Setzt den Ausgang 1
clrportx2 rem Setzt den Ausgang 2 zurück
```

Die Eingänge können durch die Funktion inx wie folgt abgefragt werden

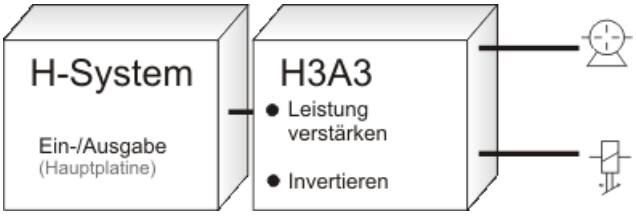
```
if (inx1=1)
   set string1 "Signal aktiv"
endif
if (inx1=0)
   set string1 "Signal inaktiv"
endif
if (/inx1)
   set string1 "Signal von aktiv nach inaktiv gewechselt"
endif
if (\inx1)
   set string1 "Signal von inaktiv nach aktiv gewechselt"
endif
```

Die Temperatursensoren an Eingang X13 bis X16 können mit den Variablen temp15 bis temp16 gelesen werden:

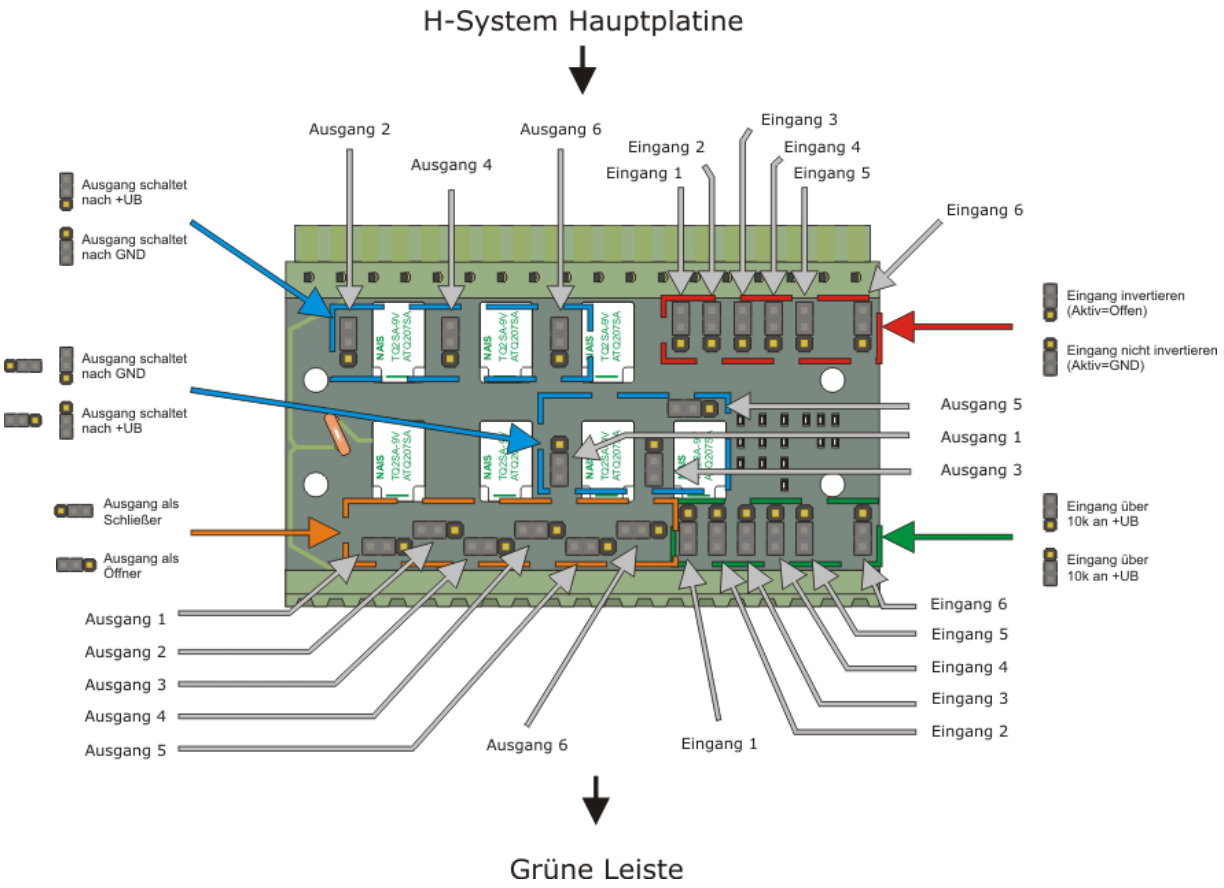
```
if (temp15>30)
   set string1 "Temperatur zu hoch"
endif
```

Das erfassen einer Temperatur benötigt 500ms.

H3A3-Adapter (Adapter für die Anpassung der Ein-/Ausgänge)



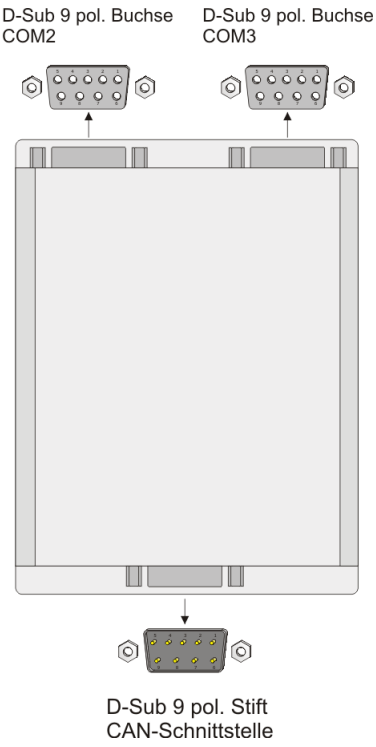
Der H3A3-Adapter dient zur Verstärkung der Ausgänge (mA) damit zum Beispiel Realais oder Motoren angesteuert werden können. Auch kann durch den Adapter die Ausgänge invertiert werden, so das sowohl nach +UB oder nach GND geschaltet wird.
Das Verhalten kann durch folgende Jumper verändert werden:



Eine spezielle Ansteuerung durch das Skript wird nicht benötigt.

Com-/CAN-Modul

Durch das Com-/CAN-Modul kann das H-System um zwei serielle Schnittstellen und um eine CAN-Schnittstelle erweitert werden. Es können alle Geräte an den zusätzlichen Schnittstellen[9] verwendet werden (ausgenommen das DATCOM Tabo4). Alternativ kann die Schnittstelle Com2 auch als RS485 bestückt werden.



RS232 (Com2/3)

2	TX
3	RX
5	GND

Alternative Bestückung
RS485 (Com2)

5	GND
6	DATA A
7	DATA B

2	CAN Lo
3	GND
7	CAN Hi

Um ein Endgerät an den erweiterten Schnittstellen zu betreiben, kann einfach das Protokoll an der jeweiligen Schnittstelle gesetzt werden:

```
set com2 chipdrive rem Kartenleser an COM 2 verwenden
set com3 dt1 rem DATCOM 640T1T an COM 3 verwenden
```

Bevor die CAN-Schnittstelle benutzt werden kann, muss die Schnittstelle auf die entsprechende Datenrate, die an dem CAN-Bus anliegt, gesetzt werden:

```
can init 125 rem Stellt 125kB für die Datenrate ein
```

Das CAN-Modul unterstützt folgende Datenrate: 5kB, 10kB, 20kB, 50kB, 100kB, 125kB, 250kB, 500kB und 1000kB.

Durch canstatus kann der Status CAN-Treiberbaustein ausgelsen werden:

```
set mem0 canstatus rem Setzt mem0 auf den Status des CAN-Bus
```

Durch die Anweisung can select muss dem H-System bekannt gegeben werden, welche Nachricht empfangen und ausgewertet werden sollen:

```
can select 100 rem Nachrichten mit Kennung 100 auswerten
can select 57  rem Nachrichten mit Kennung 57 auswerten
```

[9] Benötigt zusätzlich den Kabeladapter „H-Box +UBsw an Com2, Com 3“ Artikelnummer 4062

Die Anweisung can read wertet eine Nachricht aus und stellt die empfangenen Daten mit der Variable candata zur Verfügung:

```
can read 110 rem Nachricht mit Kennung 110 lesen
if (candata1=14) rem Enthält das erste Byte 14?
   rem *** Reagiern auf 14 ***
endif
```

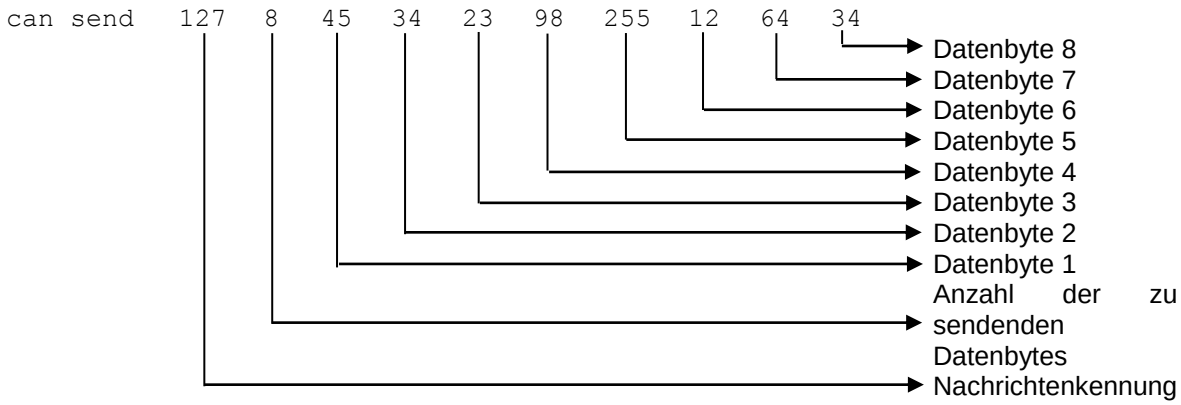
Mit canrec wertet eine Nachricht aus und stellt die empfangenen Daten mit der Variable candata zur Verfügung:

```
can read 110 rem Nachricht mit Kennung 110 lesen
if (canrec) rem Wurden neue Daten empfangen?
endif
```

Nach dem Befehl can read kann mit der Variable canlen die Anzahl der empfangen Bytes abgefragt werden:

```
can read 231 rem Nachricht mit Kennung 231 auswerten
if (canlen<2) rem Weniger als 2 Byte empfangen
   rem *** Reagieren wenn weniger als 2 Byte empfangen ***
endif
```

Mit dem Befehl „can send“ können Nachrichten auf den CAN-Bus gesendet werden:



Folgendes Beispiel sendet eine Nachricht mit der Kennung 127 mit zwei Datenbytes und dem Inhalt 14 und 35:

```
can send 127 2 14 35
```

„canselect“ Option und „canchg“ verfügbar ab Version H2SXRA04

Optional besteht die Möglichkeit für die „can select“ Anweisung als 2. Parameter eine Maske zu übergeben. Diese Maske bestimmt, ob die Änderung eines Datenbytes zum Setzen des canchg Flags führt. Beispiel:

```
can select 127 h0A
```

Der Hexwert h0A ergibt in binärer Darstellung b0000 1010. Bit2 und Bit4 sind gesetzt. Ändert sich nun Datenbyte2 oder Datenbyte4, wird nach dem Lesen der entsprechenden Kennung das canchg Flag gesezt.

```
can read 127
if (canchg)
   set string1 "Datenbyte2 und Datenbyte4 verändert"
endif
```

Akku-/Lademodul

Durch das Akku-/Lademodul kann ein 8V oder 12V Blei-Gel-Akku an der H-Box betrieben werden. An das Modul kann zum Beispiel eine Solarzelle oder ein Netzteil angeschlossen werden um den Akku auf zu laden.

Anschlussbelegung

Leiste	Name	Technische Daten	Beschreibung
	GND		
	GND		
	GND		
	GND		
	GND		
	+VC/mpu	Ausgang +5V	Nur für Testzwecke, nicht verwenden
	+VNot_Bat		8V Akku bei 12V Bordspannung, 12V bei 24V Bordspannung
	-VNot_Bat		GND des Akkus
	+VC/Solar	Eingang	Solarzelle +: >=21V, min. 350mA
	-VC/Solar	Eingang	GND der Solarzelle
	-V/Netz	Eingang	GND des zusätzlichen Netzteils
	+V/Netz	Eingang	12V-24V für zusätzliches Netzteil
	/Wake(invertiert)	Eingang	Reaktivieren der H-Box (nach Stand-By)
	VBat/Mess	Ausgang	Liefert die Akkuspannung zum messen
	VBat/SW		Keine Funktion, nicht verwenden
	GND		
	+UB	Eingang	12V/24V Bordspannung
	GND		

Für die Benutzung des Moduls werden keine Skriptanweisungen benötigt.

H-Bus-System

Durch den H-Bus kann das H-/S-System um weitere Eingänge und Ausgänge erweitert werden. Dabei können die Ausgänge zur Steuerung von Motoren genutzt werden:

Leiste	Name	Technische Daten	Beschreibung
16	In 7 +	5-60V DC	
15	In 7-		
14	In 5 +		
13	In 5 -		
12	In 3 +		
11	In 3 -		
10	In 1 +		
9	In 1 -		
8	RS485 B		
7	RS485 A		
6	Mot 3		
5	Mot 3		
4	Mot 1		
3	Mot 1		
2	+UB		
1	GND		

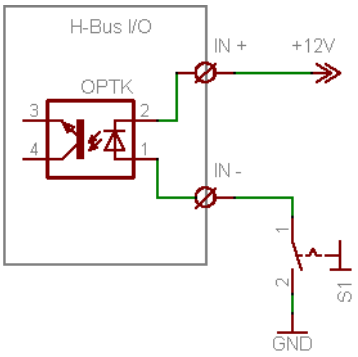
Leiste	Name	Technische Daten	Beschreibung
36	In 8 +	5-60V DC	
35	In 8-		
34	In 6 +		
33	In 6 -		
32	In 4 +		
31	In 4 -		
30	In 2 +		
29	In 2 -		
28	Wake\		Durch anlegen von GND kann das H-/S-System von dem Stand-By_modus in den Aktiv-Modus gesetzt werden
27	Wake\		
26	Mot 4		
25	Mot 4		
24	Mot 2		
23	Mot 2		
22	+UB	+12V	
21	GND		

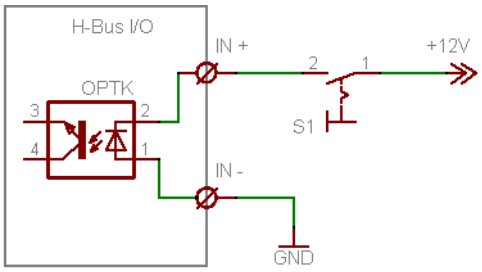
Bevor das H-Bus-System genutzt werden kann, muss die entsprechende Schnittstelle auf das H-Bus-Protokoll gesetzt werden:

```
set com2 bus
```

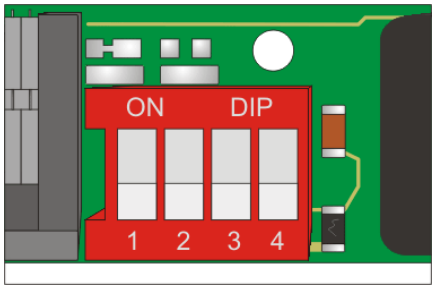
Beachten Sie, dass zur Kommunikation mit den H-System-Komponenten nicht die RS232-Schnittstelle sondern die RS485-Schnittstelle genutzt wird.

Die Eingänge können wie folgt beschaltet werden:





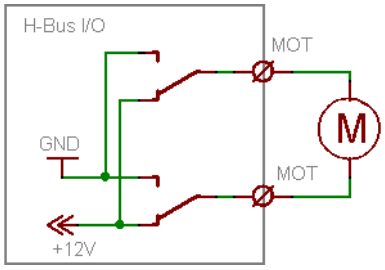
Jede H-Bus-Komponente besitzt eine einstellbare Kennung, um bei einer Verkettung von mehreren Komponenten eindeutig ansprechbar zu sein. Durch den DIP-Schalter auf der Komponenten-Platine wird diese Kennung bestimmt:



Die Eingänge können über die Variable mod gelesen werden. Bei der Benutzung der Module wird die Kennung plus 16 angeben, zum Beispiel für das erste Modul (Kennung 0):

```
if (mod16.1=1)
  rem Eingang 1 von Modul 0 ist aktiv
endif
if (mod17.3=0)
  rem  Eingang 3 von Modul 1 ist inaktiv
endif
```

Die Ausgänge können wie folgt beschaltet werden:



Die einzelnen Ausgänge können wie folgt verändert werden, die somit die Drehrichtung des Motors beeinflussen:

```
rem Schaltet Motor 1 ab
set mod16.1 0
set mod16.2 0
```

```
rem Motor 1 z.B. im Linkslauf
set mod16.1 1
set mod16.2 0

rem Motor 1 z.B. im Rechtslauf
set mod16.1 0
set mod16.2 1
```

Wie aus der Schaltung ersichtlich, können die Ausgänge nicht nur Motoren sondern auch andere Geräte schalten.

Kapitel 7: Entwurfsmuster

Einige Anforderung an den Programmierer erfordern immer gleiche Vorgehensweisen. In diesem Kapitel finden sie die bisher erkannt und genutzten Entwurfsmuster. Nutzen sie diese Entwurfsmuster auch um neue Ideen für ihre Projekte zu finden.

Zeit messen

Die Zeitvariablen timer**n** zählen von einer vorgegeben Zeit herunter. Um einen Zähler zu realisieren der bei 0 beginnt und hochzählt kann folgendes Schema genutzt werden:

```
rem *** Zu beginn des Skripts ***
set timer0 32767 rem 32767 ist der max. Wert für Zeitgeber
…
set mem0 (32767 – timer0) rem Rechner die vergangene Zeit
if (mem0>10) rem mehr als 10 Sekunden vergangen
…
```

Beachten sie, dass sie bevor sie die vergangene Zeit abfragen können, die Berechnung „set mem0 (32767 – timer0)“ durchführen müssen.

Sekunden Zeitgeber

In Anwendungen die zum Beispiel eine Hupe als Warnsignal durch Impulse ansprechen sollen, kann folgendes Skript genutzt werden um ein Signal im Sekundentakt wechseln zu lassen:

```
if (/in1) rem Zum Beispiel Auslöser für einen Alarm
   set timer0 20 rem 20 Sekunden lang das Signal geben
endif
…
if (timer0&1 = 1) rem Wurde das erste Bit gesetzt?
   setport1 rem Signal auf Ausgang 1 geben
else
   clrport1 rem Signal auf Ausgang 1 zurücksetzen
endif
```

Das Prinzip basiert darauf, dass jede Sekunde der timer um eins vermindert wird. Wandelt man die entsprechende Zeit von in das Binäre Zahlensystem so erkennt man, das Bit 1 ständig von 1 auf 0 und zurück auf 1 springt. Die Abfrage „if“ bezieht sich genau auf dieses Bit.

Intervalle mit GPS

Die Flottenmanagment Software DATCOM Fleet bietet die Möglichkeit die aktuelle Position des System einmalig, fahrstreckenabhängig oder zyklische angefordert werden. Die Unterstützung des Systems ist bei Nutzung der GSM-Technologie (SMS) bereits gegeben. Um eine noch größere Flexibilität im Bereich GPRS zu schaffen, müssen diese Unterstützungen im Skript realisiert werden:

```
rem Breiten- und Längengrad merken
var0 = latitude
var1 = longitude
set mem0   900   rem Vorgabe für Zykluszeit (Sekunden)
set timer0 0     rem Zykluszeit
set mem1   10000 rem Vorgabe für Fahrstrecke (Meter)

while (1=1) rem Endlosschleife
```

```
  tcpopen "gprs" 0 63486 "217.7.100.188" rem Verbindung öffnen
  tcprec
  while (tcpread) rem Solange Daten empfangen werden
    if (tcptype=100) rem Einmal GPS angfordert?
      tcpwrite gps
    endif
    if (tcptype=101) rem Zyklisch GPS angefordert (Minuten)?
      set mem0 (60*tcpdata)
      set timer0 0
    endif
    if (tcptype=102) rem Zyklisch GPS angefordert (Sekunden)?
      set mem0 tcpdata
      set timer0 0
    endif
    if (tcptype=103) rem Fahstrecken abhängig melden?
      set mem1 (100*gpsdata)
      set timer0 0
    endif
  wend

  rem Zurückgelegte Strecke errechnen
  var2 = distance latitude longitude var0 var1
  var2 = var2 * 1000

  rem Bedingung siehe weiter unten
  if (loggedin && ((mem0>0) & (timer0=0)) or ((mem1>0) & (var2>mem1))
    set timer0 mem0
    var0 = latitude
    var1 = longitude
    rem Position senden
    tcpwrite gps
  endif
wend
```

Da das Skript nicht einfach zu verstehen ist, folgt hier eine ausführliche Beschreibung:

```
rem Breiten- und Längengrad merken
var0 = latitude
var1 = longitude
set mem0   900   rem Vorgabe für Zykluszeit (Sekunden)
set timer0 0     rem Zykluszeit
set mem1   10000 rem Vorgabe für Fahrstrecke (Meter)
```

Ganz zu beginn wird einmal die aktuelle Position in var0 und in var1 abgelegt. Es handelt sich bei dieser Positionsangabe mit Sicherheit um eine alte Angabe da einige Zeit benötigt wird bis der GPS-Empfänger gültige Daten liefert. Diese Tatsache fällt hier nicht ins Gewicht.
Mit mem0 legen wir die Zeit bis eine neue Position an das GPRS-Gateway gesendet wird fest. Wird hier eine 0 benutzt, erfolgt kein senden (Zeit bezogen). Genau den gleichen Sinn erfüllt mem1, jedoch steht hier die Entfernung die zurückgelegt werden muss, bis eine neue Position gemeldet wird. Wird hier 0 benutzt, so ist diese Funktion deaktiviert. Der timer0 entspricht der Zeit bis eine neue Position gemeldet wird.

```
tcpopen "gprs" 0 63486 "217.7.100.188" rem Verbindung öffnen
tcprec
while (tcpread) rem Solange Daten empfangen werden
  if (tcptype=100) rem Einmal GPS angfordert?
    tcpwrite gps
  endif
  if (tcptype=101) rem Zyklisch GPS angefordert (Minuten)?
    set mem0 (60*tcpdata)
    set timer0 0
  endif
```

```
  if (tcptype=102) rem Zyklisch GPS angefordert (Sekunden)?
    set mem0 tcpdata
    set timer0 0
  endif
  if (tcptype=103) rem Fahstrecken abhängig melden?
    set mem1 (100*gpsdata)
    set timer0 0
  endif
wend
```

Nach dem Aufbau der Verbindung (tcpopen) wird überprüft ob Daten vorhanden sind. Wenn ja werden diese Ausgewertet und umgesetzt. Ausgenommen von der einmaligen GPS Anforderung, wird der timer0 auf 0 gesetzt um eine Position zu melden.

```
rem Zurückgelegte Strecke errechnen
var2 = distance latitude longitude var0 var1
var2 = var2 * 1000

rem Bedingung siehe weiter unten
if (loggedin & ((mem0>0) & (timer0=0)) or ((mem1>0) & (var2>mem1))
  set timer0 mem0
  var0 = latitude
  var1 = longitude
  rem Position senden
  tcpwrite gps
endif
```

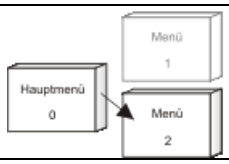
Es wird zuerst die zurückgelegte Strecke berechnet, dann folgt die Bedingung wann eine Position gemeldet wird: Wenn Verbunden mit dem GPRS-Gateway und (Vorgabezeit gesetzt und Zeit abgelaufen) oder (wenn Fahrstrecke gesetzt und zurückgelegte wurde).

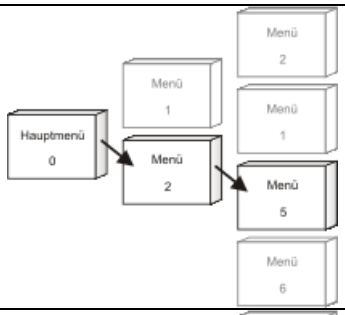
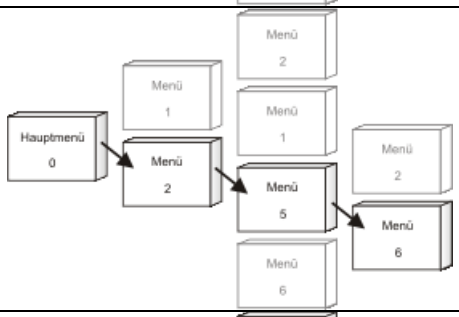
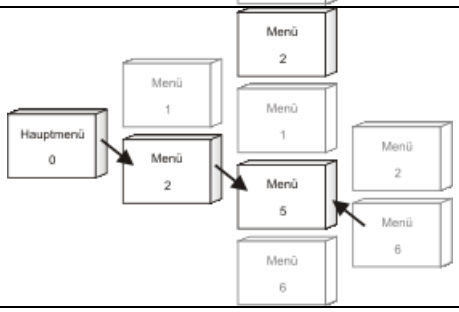
Komplexe Menüführung

Oft steht im Mittelpunkt der Skriptprogrammierung das Erzeugen einer umfangreichen Menüsteuerung zum Beispiel für ein DATCOM 640T1T oder DATCOM 80TF4. Um diese Struktur zu erzeugen benötigt man zwei Mem-Variablen:

- mem0 ist das aktuelle Menü, zum Beispiel bedeutet 0 das Hauptmenü
- mem1 veranlasst ein Neuzeichnen des aktuellen Menüs

Um mem0 leichter zu verstehen, wird in der folgenden Liste das Aufrufen und Verlassen von Menüs simuliert:

Aktives Menü	Mem0 Inhalt	Beschreibung	Skizze
0	0	Durch betätigen eines Schalters wird in den Unterpunkt 2 gesprungen. Dazu wird mem0 mit 10 multipliziert und 2 hinzu addiert.	
0-2	2	Ein weiteres Menü (Unterpunkt 5) wird im aktuellen Menü aufgerufen. Dazu wird mem0 mit 10 multipliziert und 5 hinzu addiert.	

0-2-5	25	Ein weiteres Menü (Unterpunkt 6) wird im aktuellen Menü aufgerufen. Dazu wird mem0 mit 10 multipliziert und 6 hinzu addiert.	
0-2-5-6	256	Das aktuelle Menü wird jetzt wieder verlassen. Dafür wird mem0 durch 10 geteilt und wieder in mem10 zurück geschrieben (da mem eine Ganzzahlvariable ist, wird der Nachkommateil verworfen)	
0-2-5	25		

Wie aus den obigen Beispielen ersichtlich, wird bei dem Aufruf eines Unterprogramms der aktuellen Inhalt von mem0 mit 10 multipliziert und die Nummer des anzuzeigenden Menüs zu mem0 hinzu addiert. Das verlassen eines Menüs geschieht einfach durch dividieren von mem0 durch 10:

```
…
set mem0 0 rem Anfang ist Hauptmenü
set mem1 1 rem Menü neu zeichnen
…
if (mem0=0) rem Ist Hauptmenü aktiv?
  if (mem1=1) rem Ja, neu zeichnen?
    set dt1button21 "Auftragsmenü"
    show dt1menue2
    set mem0 0 rem Menü wurde neu gezeichnet
  endif
  if (dt1key=21) rem Wurde Taste 21 im Hauptmenü gedrückt?
    set dt1key 0
    set mem0 ((mem0*10) + 2) rem Untermenü 2 relativ zum aktuellen Menü
    set mem1 1 rem Neu zeichnen
  endif
endif
…
while (…)
…
if (mem0=2) rem Ist "Auftragsmenü" aktiv?
  if (mem1=1) rem Ja, neu zeichnen?
    set dt1button21 "Telefon"
    set dt1button22 "Zurück"
    show dt1menue2
    set mem0 0 rem Menü wurde neu gezeichnet
  endif
```

```
    if (dt1key=21) rem Wurde Taste 21 im Auftragsmenü gedrückt?
      set dt1key 0
      set mem0 ((mem0*10) + 4) rem Untermenü 4 relativ zum aktuellen Menü
      set mem1 1 rem Neu zeichnen
    endif
    if (dt1key=22) rem Wurde Taste 22 im Auftragsmenü gedrückt?
      set dt1key 0
      set mem0 (mem0/10) rem Aktuelles Menü verlassen
      set mem1 1 rem Neu zeichnen
    endif
  endif
  …
  wend
  …
```

Ein weiterer Bonuspunkt bei Nutzung von diesem Entwurfmuster ist, dass die Tastenauswertung immer als Block an das Anzeigen gebunden ist, somit eine klare Zuständigkeit entsteht.

Anhang A: Einbau- und Sicherheitshinweis

Allgemein

Beachten sie die jeweiligen Bedienungsanleitungen und Sicherheitshinweise der Gerätehersteller, besonders bei externen Zubehör, dazu zählen GSM-/GPS Antennen, Hörer, Funkgeräte und Mobilfunkgeräte. Der Einbau bzw. das Zusammenfügen der Komponenten darf nur durch geschultes Fachpersonal durchgeführt werden, um Schäden an dem System, an Komponenten und an Personen zu vermeiden. Das System und die Komponenten müssen während dem Ein- bzw. Aufbauen stromlos gemacht werden. Beachten sie die gesetzliche Grundlage für das jeweilige Anwendungsgebiet (zum Beispiel im Einbau in Fahrzeugen die Straßenverkehrsordnung).

Einbau in Innenräume

Beim Einbau in Innenräume (zum Beispiel von Fahrzeugen) muss eine freie Rundumsicht gewährleistet werden. Bringen sie das System oder Komponenten des Systems nicht im Aufblasbereich des Airbags und auch nicht im Kopf-/Knie-Aufschlagbereich an. Beachten sie hierzu die Herstellerangaben. Verbauen sie das System oder die Komponenten nach Möglichkeit vibrationsarm. Das Anbringen des Systems oder der Komponenten darf nicht an tragenden oder sicherheitsrelevanten Karosserieteilen erfolgen. In diesem Zuge achten sie auf verdeckte Kabelbäume, Tank und Kraftstoffleitungen usw. Das System muss mit vier Schrauben an den dafür vorgesehenen Bohrungen befestigt werden.

Elektrischer Anschluss

Benutzen sie zur Absicherung des Systems eine 1 Ampere mittelträge Sicherung. Werden Relais eingesetzt, benutzen sie hier eine Freilaufdiode (Beispiel siehe Seite 19), um eine schädliche Induktionsspannung zu vermeiden. Der Anschluss des Systems oder der Komponenten muss nach dem entsprechenden Anschlussplan erfolgen. Verlegen sie Verbindungsleitungen niemals über scharfe Kanten und knicken sie die Leitungen niemals scharf ab. Beachten sie, dass einige Komponenten nur für 12V ausgelegt sind und somit bei Einsatz eines Bordnetzes von 24V ein Netzteil zu Transformation der Spannung auf 12 V benötigt wird.

GSM/GPRS Antenne

Die Antenne muss für D- und E-Netz geeignet sein, dazu zählt das die Antenne eine maximale Impedanz von 50 Ohm aufweisen darf. Verwenden sie nur zu der Antenne gehörige Kabelsätze um die Empfangsqualität nicht zu beeinträchtigen.

GPS Antenne

Beachten sie das die Antenne folgender Spezifikation entspricht: Frequenzbereich 1575,42 MHz, Verstärkung >28dB, Rauschzahl <1,8 dB, Betriebsspannung 3,5 – 5,5 VDC, Stromaufnahme 18mA, Phantomspeisung.

Anhang B: ASCII-Tabelle

ASCII hex	ASCII dez	Zeichen	ASCII hex	ASCII dez	Zeichen.	ASCII hex	ASCII dez	Zeichen.	ASCII hex	ASCII dez	Zeichen.
00	0	NUL	20	32	SP	40	64	@	60	96	`
01	1	SOH	21	33	!	41	65	A	61	97	a
02	2	STX	22	34	"	42	66	B	62	98	b
03	3	ETX	23	35	#	43	67	C	63	99	c
04	4	EOT	24	36	$	44	68	D	64	100	d
05	5	ENQ	25	37	%	45	69	E	65	101	e
06	6	ACK	26	38	&	46	70	F	66	102	f
07	7	BEL	27	39	'	47	71	G	67	103	g
08	8	BS	28	40	(	48	72	H	68	104	h
09	9	TAB	29	41	)	49	73	I	69	105	i
0A	10	LF	2A	42	*	4A	74	J	6A	106	j
0B	11	VT	2B	43	+	4B	75	K	6B	107	k
0C	12	FF	2C	44	,	4C	76	L	6C	108	l
0D	13	CR	2D	45	-	4D	77	M	6D	109	m
0E	14	SO	2E	46	.	4E	78	N	6E	110	n
0F	15	SI	2F	47	/	4F	79	O	6F	111	o
10	16	DLE	30	48	0	50	80	P	70	112	p
11	17	DC1	31	49	1	51	81	Q	71	113	q
12	18	DC2	32	50	2	52	82	R	72	114	r
13	19	DC3	33	51	3	53	83	S	73	115	s
14	20	DC4	34	52	4	54	84	T	74	116	t
15	21	NAK	35	53	5	55	85	U	75	117	u
16	22	SYN	36	54	6	56	86	V	76	118	v
17	23	ETB	37	55	7	57	87	W	77	119	w
18	24	CAN	38	56	8	58	88	X	78	120	x
19	25	EM	39	57	9	59	89	Y	79	121	y
1A	26	SUB	3A	58	:	5A	90	Z	7A	122	z
1B	27	Esc	3B	59	;	5B	91	[	7B	123	{
1C	28	FS	3C	60	<	5C	92	\	7C	124	|
1D	29	GS	3D	61	=	5D	93	]	7D	125	}
1E	30	RS	3E	62	>	5E	94	^	7E	126	~
1F	31	US	3F	63	?	5F	95	_	7F	127	DEL

Generell können auch Zeichen über 127 genutzt werden, die Interpretation ist jedoch abhängig von der verwendeten Landeseinstellung des Endgeräts oder des PC's.

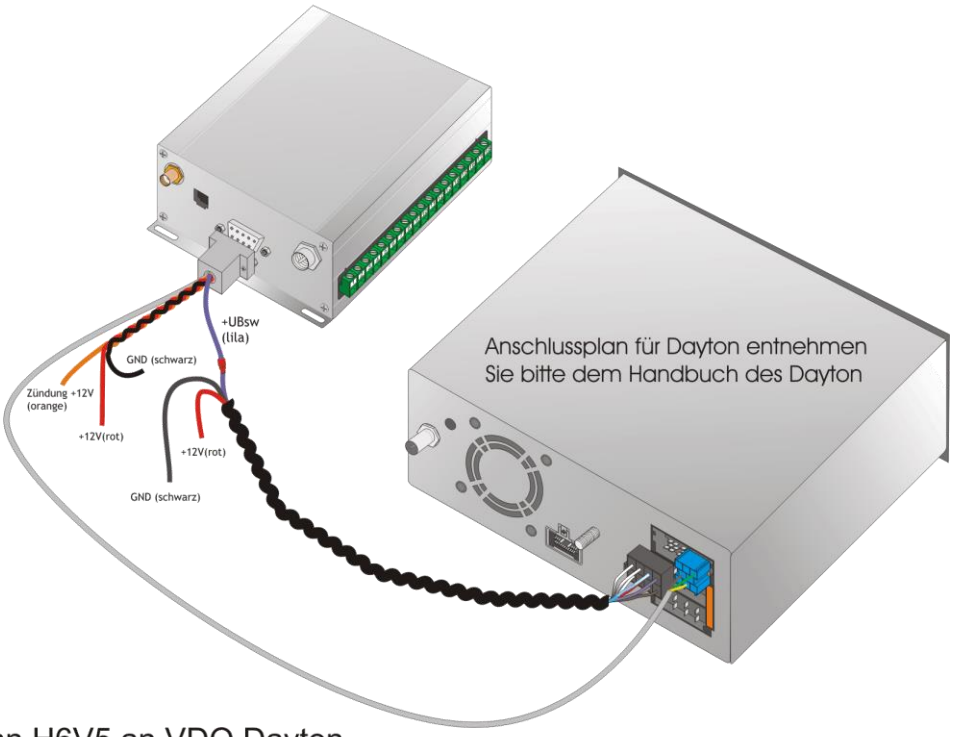
Anhang C: Befehlsübersicht im Terminalbetrieb

Im Terminalbetrieb können die folgenden Anweisungen eingegeben werden:

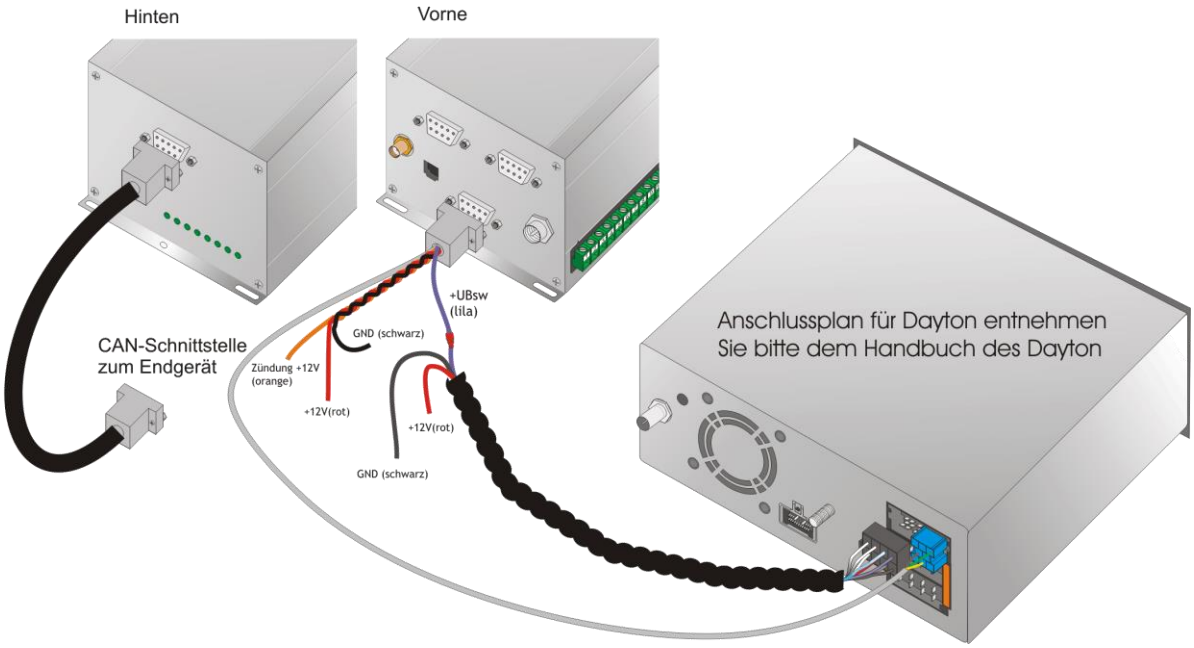
Anweisung	Beschreibung
ain	Gibt die Werte der analog Eingänge aus
cli	Gibt den letzten Fehler aus
date	Erlaubt das manuelle Einstellen des Datums
debug	Verzögert den Skriptablauf und zeigt die gerade ausgeführten Befehle an
del	Löscht eine Datei
din	Gibt die Werte der digitalen Eingänge aus
dir	Zeigt die vorhandenen Dateien auf dem System
download	Veranlasst das System eine Datei zu senden. Diese Datei kann mit dem Terminalprogramm (Y-Modem) empfangen werden
dtb	Zeigt die interne Datenbank an
exp	**Nur H-System**: Gibt die Werte der digitalen Eingänge der Ein-/Ausgabeerweiterung aus
format	Formatiert den internen Speicher (H-System) oder die MMC (S-System)
gps	Gibt Informationen vom GPS-Empfänger aus (Breiten-, Längengrad, Geschwindigkeit, Richtung, Satelliten)
gprs	Gibt Informationen zum Einbuchen in das GPRS-Netz und zum Gatway (IP-Adressen, Port)
gsm	Gibt Informationen vom GSM-Modul aus (Firmwarestand, Qualität, Eingebucht, Anbieter)
help	Zeigt eine Hilfeseite zu den wichtigsten Befehle
quit	Beendet den Terminaldialog
reboot	Startet das System neu
rec	Zeigt die letzte SMS an (Standard SMS)
send	Sendet eine SMS. Dazu muss nach der send-Anweisung eine Telefonummer angegeben werden, gefolgt von einer Zeichenkette zum Beispiel: send +49170123546789 "Alarm!“
set	Durch diese Anweisung können folgende Variablen gesetzt werden: mem, timer, counter, var. Zum Beispiel set mem3 1 set var0 12.5 set counter0 9 set timer1 15
str	Zeigt alle Zeichenketten an
tele	Gibt die Werte der analog Eingänge, digtalen Eingänge und die Frequenzen aus
time	Erlaubt das manuelle Einstellen der Uhrzeit
update	Führt eine Erneuerung der Firmware durch. Dazu muss die Firmware im „.bin“-Format mit dem Befehl upload übertragen werden
upload	Das System bereitet sich auf den Empfang einer Datei vor. Die Datei kann über das Terminal (Y-Modem) gesendet werden

Anhang D: Anschlusspläne

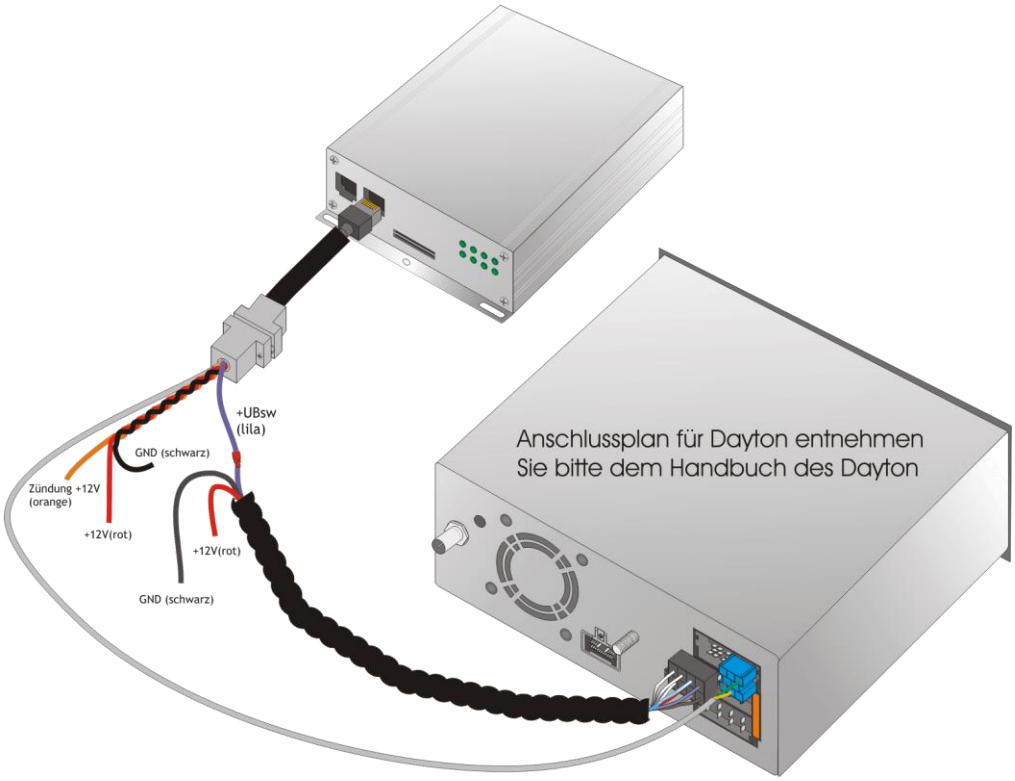
Anschlußplan H3V1 an VDO Dayton



Anschlußplan H6V5 an VDO Dayton



Anschlußplan S2 an VDO Dayton



Änderungen

Version	Datum	Änderung	Autor
1.0	26.11.2008	Erste Ausgabe – DATCOM	Franco Labella

Stichwortverzeichnis

Funktionsreferenz

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y