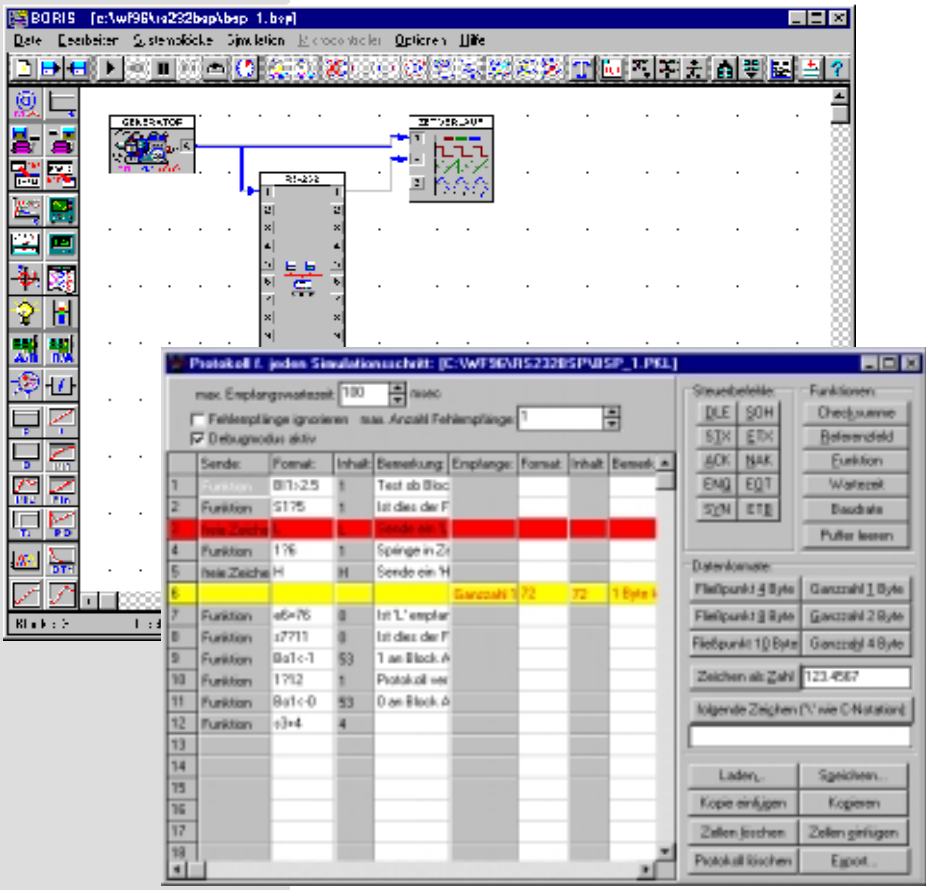
WinFACT 98



WinFACT 98

Universelles RS-232-Modul

Ingenieurbüro Dr. Kahlert

Ludwig-Erhard-Str. 45
D-59065 Hamm

WinFACT 98

WINDOWS FUZZY AND CONTROL TOOLS

BENUTZERHANDBUCH

BORIS-RS-232-MODUL

Ingenieurbüro Dr. Kahlert
Ludwig-Erhard-Str. 45 D-59065 Hamm
Tel. 0 23 81/926 996 Fax 0 23 81/926 997

Inhalt

Leistungsumfang

Die Verwendung eines RS-232- Moduls erlaubt es Ihnen auf umfassende Weise, BORIS mit anderen Geräten in Kommunikation treten zu lassen. Durch eine simple Kombination aus Anweisungen und Übertragungsdaten, die das RS-232- Modul verarbeiten kann, können Sie nahezu beliebige Übertragungsprotokolle nachbilden. Von der einfachen ASCII-Zeichen-Übertragung bis zum komplexen Feldbus-Protokoll können sämtliche Kommunikationen aufgebaut werden. Die Erstellung der Protokolle wird durch ein integriertes Debugging unterstützt.

Lieferumfang und Installation

Vorbemerkungen

Alle zum Betrieb des universellen RS-232-Moduls erforderlichen Dateien befinden sich auf einer 3,5"-Diskette bzw. CD-ROM. Die Installation des Moduls wird vollautomatisch und dialoggeführt vorgenommen. Beachten Sie vor Beginn der Installation bitte folgendes:

Während der Installation des RS-232-Moduls ist das gewünschte Zielverzeichnis anzugeben. Dieses muß unbedingt mit Ihrem aktuellen WinFACT 98-Programmverzeichnis (gemäß Setup-Voreinstellung *\WF98*) identisch sein! Sofern Sie also bei der Installation von WinFACT 98 an früherer Stelle ein anderes Programmverzeichnis angegeben haben, müssen Sie die Vorgabe des RS-232-Setups dementsprechend anpassen.

Lieferumfang

Der Lieferumfang des RS-232-Moduls umfaßt die folgenden Dateien:

Im WinFACT 98-Programmverzeichnis:

Datei	Funktion
RS232.DLL	BORIS-User-DLL für das universelle RS-232-Modul
COMXDLL.DLL	Kommunikationsroutinen (werden von User-DLL-benötigt)
RS232.BMP	Block-Bitmap für RS-232-Modul (Bildschirm)
RS232_P.BMP	Block-Bitmap für RS-232-Modul (Drucker)
RS232_T.BMP	Block-Bitmap für RS-232-Modul (Systemblock-Toolbar)

Im WinFACT 98-Beispielverzeichnis:

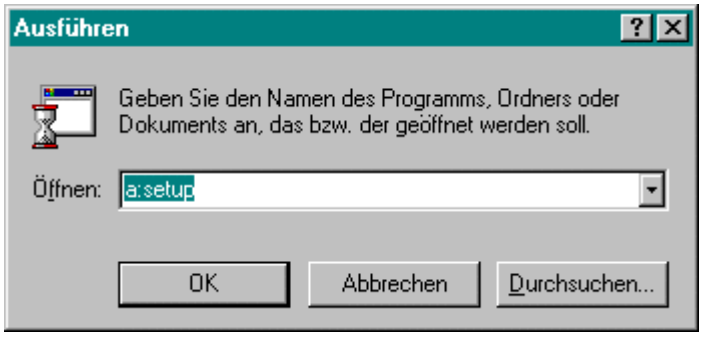
Datei	Funktion
BSP_1.BSY	BORIS-Beispielsystem 1
BSP_1.PKL	Zugehöriges Protokoll
BSP_2.BSY	BORIS-Beispielsystem 2
BSP_2.PKL	Zugehöriges Protokoll
BSP_5.PKL	Protokolldatei zu Beipiel 5

Installation

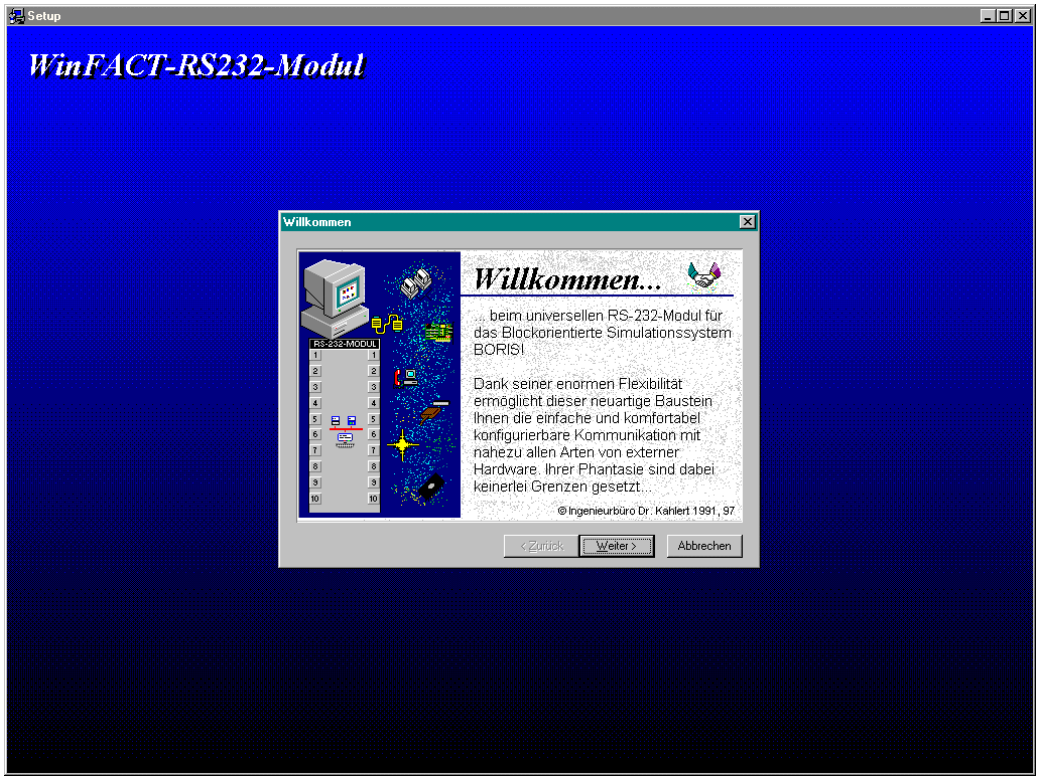
Zur Installation legen Sie die mitgelieferte Diskette in das Diskettenlaufwerk Ihres Rechners ein und wählen Sie über das Windows-Startmenü die Option *Ausführen*. Starten Sie im daraufhin erscheinenden Dialog das Setup-Programm durch Eingabe von *a:setup*. Das Setup-Programm meldet sich daraufhin zunächst mit einem Begrüßungsbildschirm.

Der weitere Installationsverlauf erfolgt vollständig benutzergeführt und umfaßt die folgenden Schritte:

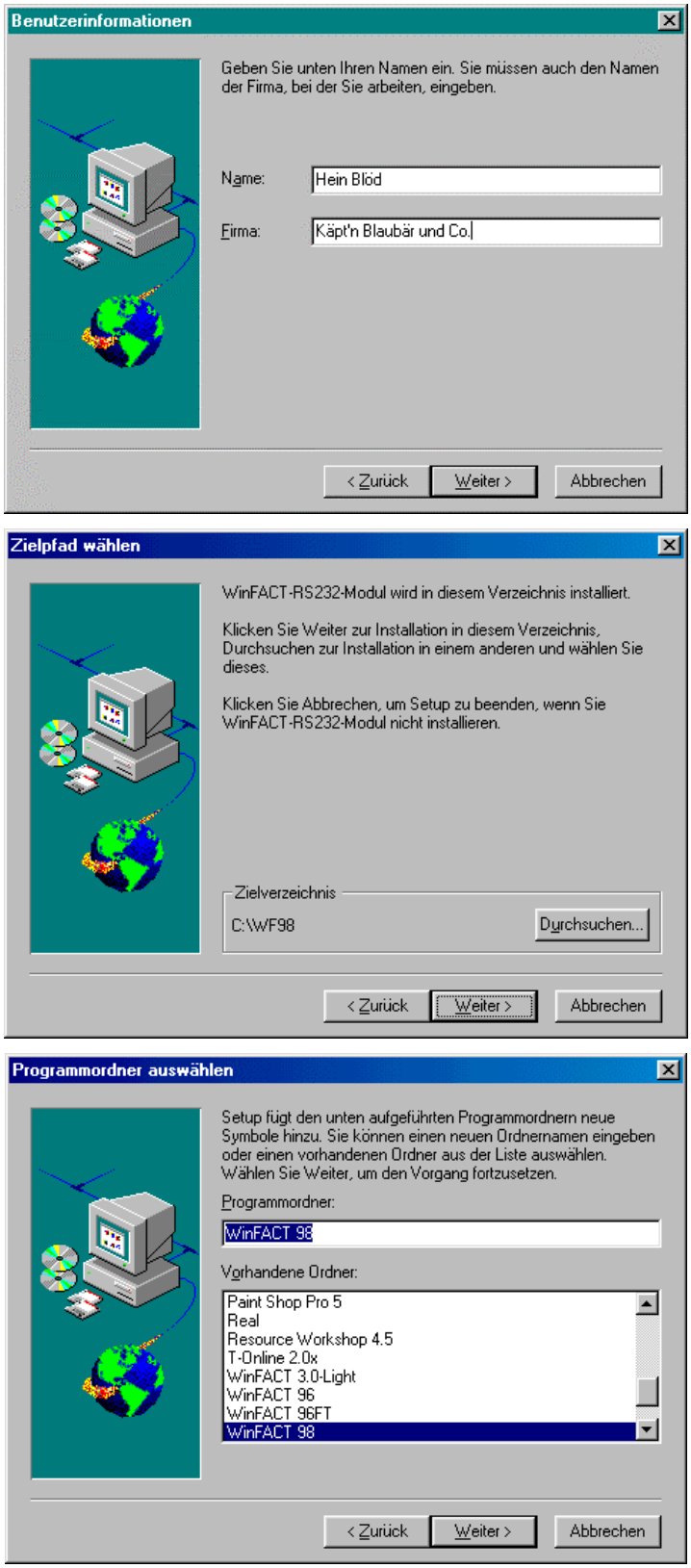
- Abfragen der Benutzerdaten
- Wahl des Zielverzeichnisses (Voreinstellung *\WF98*, siehe dazu Anmerkungen oben!)
- Wahl des Programmordners



Starten des Setup-Programms

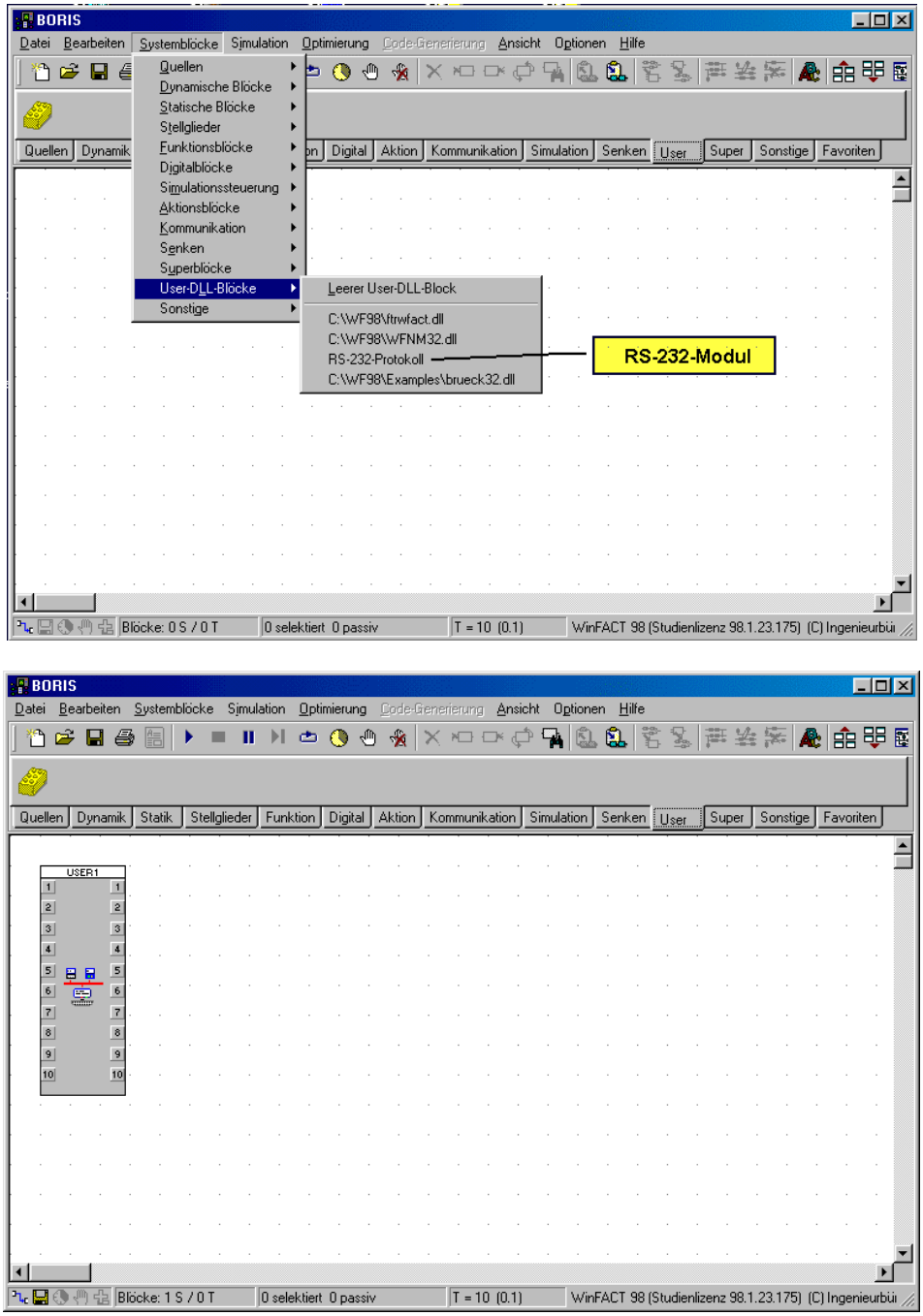


Begrüßungsbildschirm des Setup-Programms



Stadien der Installation: Abfragen der Benutzerdaten (oben), Wahl des Zielverzeichnisses (mitte) und des Programmordners (unten)

Nach erfolgter Installation steht Ihnen das universelle RS-232-Modul beim nächsten Aufruf von BORIS automatisch im *User-DLL*-Untermenü bzw. über die Palette *User* der Systemblock-Toolbar zur Verfügung (siehe nachfolgende Bildschirmgrafik). Alternativ dazu können Sie auch zunächst einen *leeren User-DLL-Block* einfügen und innerhalb dessen Parameterdialogs dann die zugehörige User-DLL *RS232.DLL* angeben.



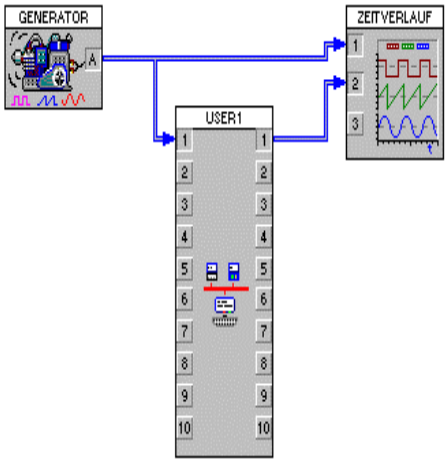
Einfügen eines RS-232-Moduls über das User-DLL-Untermenü (oben) und Bildschirm mit eingefügtem Modul (unten)

Einführung

Für das nachfolgende Beispiel ist es erforderlich, eine freie RS-232-Schnittstelle zu haben. Ferner benötigen Sie ein Gerät, das ein Zeichen-Echo ausübt. Falls dieses nicht vorhanden ist, können Sie sich eines einfachen Tricks bedienen. Schließen Sie an dem Ende der RS-232-Leitung PIN 2 und PIN 3 über einen 10 kΩ Widerstand kurz. Dadurch wird jedes gesendete Zeichen direkt wieder empfangen.

In Abhängigkeit eines digitalen Signals unter BORIS soll das ASCII-Zeichen 'L' - falls Low-Pegel - oder ein 'H' - falls High-Pegel anliegt - über die serielle Schnittstelle ausgegeben werden. Das Echo von der seriellen Schnittstelle soll so verarbeitet werden, daß ein empfangenes 'H' eine 1 am Blockausgang und ein 'L' den Wert 0 produziert.

Bauen Sie folgende Struktur unter BORIS auf (dieses Beispiel befindet sich als BSP_1.BSY im Unterverzeichnis EXAMPLES des WinFACT-Programmverzeichnisses). Das in der Struktur befindliche RS-232 Modul können Sie durch einen Mausklick auf den Menüpunkt *Systemblöcke / User-Blöcke (DLL) / RS-232 Protokoll ...* hinzufügen.



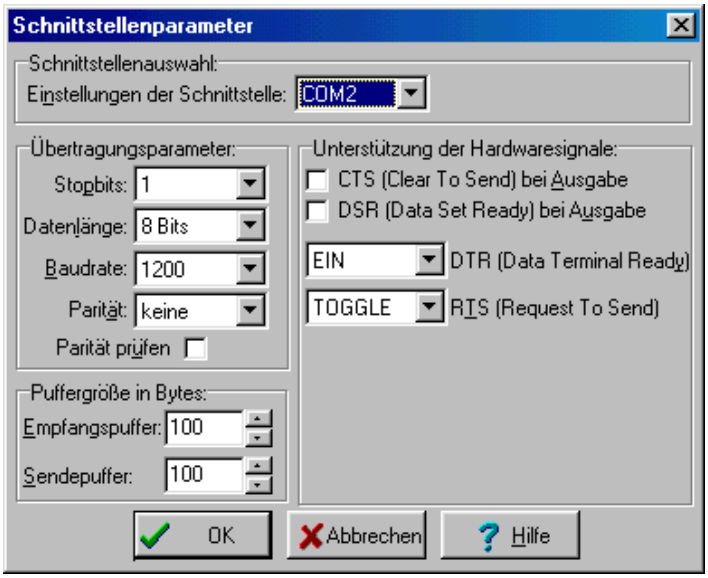
Der Generator sollte so eingestellt sein, daß er abwechselnd zehnmal das Signal für High- und für Low-Pegel liefert. Dafür sind bei einer Abtastzeit von 0.1 folgende Einstellungen am Generator vorzunehmen:

Typ=Puls; Amplitude=5; Offset=0; Verzugzeit=0; Pulsweite=1
Periodendauer=2.

Üben Sie nun einen Doppelklick auf den RS-232-Block aus, um zum User-DLL-Dialogfenster zu gelangen. In diesem wählen Sie den Button *Dialog...*. Der Parameterdialog des Schnittstellenmoduls wird angezeigt:

Parameterdialog des RS-232-Moduls

Wählen Sie hier zunächst *Schnittstellenparameter bearbeiten...* um die folgenden Einstellungen vorzunehmen:



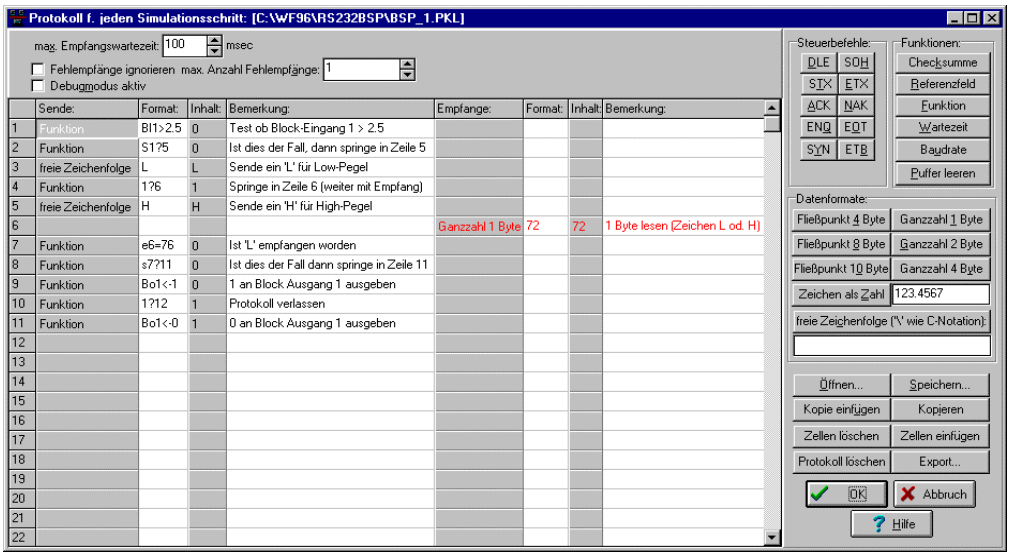
Dialog zur Einstellung der Schnittstellenparameter

Nachdem Sie Ihren Dialog mit diesem abgeglichen haben, verlassen Sie Ihn über *OK*. Die Einstellungen für die von Ihnen gewählte Schnittstelle

Wahl des COM-Ports

werden daraufhin gesetzt. Ist dies nicht möglich, weil die Schnittstelle schon von einer anderen Anwendung verwendet wird (z. B. dem Maustreiber von Windows), so stellen Sie in der Auswahlbox *Einstellungen der Schnittstelle:* einen anderen COM-Port ein (die präparierte Leitung an der Schnittstelle muß natürlich auch umgesteckt werden).

Die Schnittstelle ist nun für das RS-232 Modul vorbereitet. Als nächstes soll das Protokoll erstellt werden. Dazu klicken Sie auf den Button *Simulationsschritt...*. In dem folgenden Dialog wird das Protokoll, das bei jedem Simulationsschritt durchgeführt werden soll, spezifiziert. Klicken Sie innerhalb dieses Dialoges auf *Öffnen...* und laden Sie das erste Beispiel-Protokoll: BSP_1.PKL, das sich im Unterverzeichnis *EXAMPLES* des WinFACT-Programmverzeichnisses befindet.



	Sende:	Format:	Inhalt:	Bemerkung:	Empfange:	Format:	Inhalt:	Bemerkung:
1	Funktion	BI1>2.5	0	Test ob Block-Eingang 1 > 2.5				
2	Funktion	S1?5	0	Ist dies der Fall, dann springe in Zeile 5				
3	freie Zeichenfolge	L	L	Sende ein 'L' für Low-Pegel				
4	Funktion	1?6	1	Springe in Zeile 6 (weiter mit Empfang)				
5	freie Zeichenfolge	H	H	Sende ein 'H' für High-Pegel				
6					Ganzzahl 1 Byte	72	72	1 Byte lesen (Zeichen L od. H)
7	Funktion	e6=76	0	Ist 'L' empfangen worden				
8	Funktion	s7?11	0	Ist dies der Fall dann springe in Zeile 11				
9	Funktion	Bo1<-1	0	1 an Block Ausgang 1 ausgeben				
10	Funktion	1?12	1	Protokoll verlassen				
11	Funktion	Bo1<-0	1	0 an Block Ausgang 1 ausgeben				
12								
13								
14								
15								
16								
17								
18								
19								
20								
21								
22								

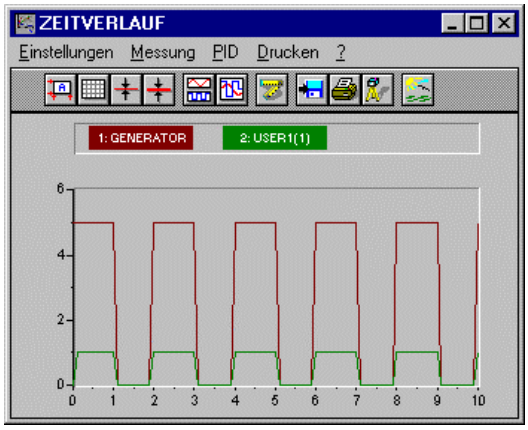
Dialog zur Eingabe eines Übertragungs-Protokolls

Verlassen Sie diesen Dialog ebenfalls mit *Ok*. Nunmehr zeigt Ihnen ein modifiziertes Bitmap auf dem *Simulationsschritt*-Button, daß ein Protokoll für *Simulationsschritt...* vorhanden ist und auch in dieser Form auf der Festplatte vorliegt, also schon gespeichert ist (grüner Haken; wäre dies nicht der Fall, so erschiene ein rotes Ausrufezeichen).

Dialog mit geladenem und in dieser Form auf Festplatte vorhandenem Protokoll für jeden Simulationsschritt

Will man schon im ersten Simulationsschritt, also zum Zeitpunkt t = 0 dieses Protokoll ausführen, so ist dasselbe Protokoll für die *Simulations-Initialisierung...* zu laden.

Starten Sie nun die Simulation, so sollte sich folgender Zeitverlauf ergeben:



Zeitverlauf der Simulation

Spezifikationen eines RS-232-Blockes

Ein RS-232-Modul besteht aus folgenden Komponenten:

1. einer Spezifikation der Schnittstelle,
2. maximal fünf verschiedenen Protokollen,
3. globalen, für jedes Protokoll sichtbaren Blockeingängen, Blockausgängen und frei verfügbaren Parameterfeldern,
4. pro Protokoll 20 internen numerischen Feldern,
5. einer Zahl, die angibt, wann dieser Block ausgeführt wird (Verarbeitungsreihenfolge).

Durch einen Doppelklick auf den Block erscheint der Dialog für den User-DLL-Block. Durch einen anschließenden Mausklick auf den *Dialog*-Button wird der Parameterdialog aufgerufen.

Parameterdialog des RS-232-Moduls

In der folgenden Tabelle werden die Dialogelemente näher erläutert:

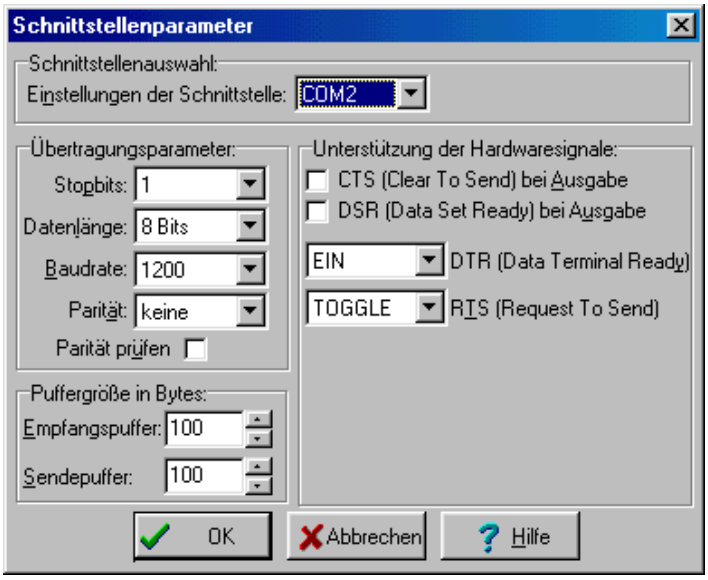
Dialogelement	**Zweck**
diesen Block als X-tes ausführen	Da mehrere RS-232-Blöcke in einem System enthalten sein können, ist es sinnvoll, daß diese in einer definierten Abarbeitungsreihenfolge ausgeführt werden. Diese kann hier vorgegeben werden. Der hier eingegebene Wert darf die Anzahl aller RS-232-Blöcke in BORIS nicht überschreiten.

Schnittstellenparameter bearbeiten...	Öffnet einen weiteren Dialog zur Einstellung der Schnittstellenparameter (Baudrate, Bits pro Byte, Parität etc.) und der Schnittstelle selbst (COM1,COM2,..).
Eingänge, Ausgänge, Variablen...	Öffnet einen weiteren Dialog, in dem Eingaben für globale Variablen gemacht werden können.
Simulation zulässig...	Öffnet den Protokolleingabedialog (s.u.). Das eingegebene Protokoll kann so aufgebaut werden, daß es durch eine globale Variable (BIx od. BOx) beschreibt, ob eine Simulation durchgeführt werden kann. Die Variable muß dabei den Wert Null annehmen, wenn die Simulation nicht zulässig sein soll, sonst einen Wert ungleich Null. Es wird vor dem 0-ten Abtastschritt ausgeführt.
Simulations-Initialisierung...	Öffnet den Protokolleingabedialog (s.u.) zur Eingabe eines Protokolls, das eine Initialisierung durchführt. Es wird zum 0-ten Abtastschritt ausgeführt.
Simulationsschritt...	Öffnet den Protokolleingabedialog (s.u.) zur Eingabe eines Protokolls, das bei jedem n-ten (n>0) Abtastschritt ausgeführt wird.
Ende der Simulation...	Öffnet den Protokolleingabedialog (s.u.) zur Eingabe eines Protokolls, das nach dem letzten Abtastschritt ausgeführt.
Testprotokoll bearbeiten...	Öffnet den Protokolleingabedialog (s.u.) zur Eingabe eines Protokolls. Das hier eingegebene Protokoll ist von der Simulation unabhängig. Es kann durch das in der nächsten Zeile beschriebene Dialogelement ausgeführt werden.
Testprotokoll ausführen	Startet das unter *Testprotokoll bearbeiten...* eingegebene Protokoll.

Es sollen nun die einzelnen Bestandteile des Moduls näher erläutert werden.

Spezifikation der Schnittstelle

Die serielle Schnittstelle kann mit Hilfe des Moduls umfassend parametriert werden. Alle Einstellungen der Schnittstelle, die Sie mit Hilfe eines Moduls treffen, werden für alle Blöcke, die sich auf diese Schnittstelle beziehen, wirksam. Sobald Sie innerhalb des Dialoges die Schnittstelle wechseln, werden die aktuellen Einstellungen der neu ausgewählten Schnittstelle angezeigt. Die Einstellungen werden generell erst beim Verlassen des Dialoges mit *Ok* übernommen.



Dialog zur Einstellung der Schnittstellenparameter

Die Einstellungen betreffen alle RS-232 Module, die dieselbe Schnittstelle verwenden!

Die Parameter werden in der nachfolgenden Tabelle erläutert:

Parameter	Erläuterung
Einstellungen der Schnittstelle	Hier wird die betreffende Schnittstelle ausgewählt, die für diesen Block und für die nachfolgenden Parameter relevant ist
Stopbits	Anzahl der Stopbits pro übertragenem Zeichen
Datenlänge	Anzahl der Bits pro Zeichen
Baudrate	Anzahl der zu übertragenden Bits pro Sekunde.
Parität	Gleichstellung der Anzahl Einsen im zu übertragenden Zeichen. Bei gerader Parität wird das Paritätsbit so gewählt, daß die Anzahl der Einsen einer geraden Zahl entspricht. Ebenso gibt es ungerade Parität, ein immer gesetztes Paritätsbit und ein immer leeres (= Null) Paritätsbit.
Parität prüfen	Paritätsprüfung ein- bzw. ausschalten.
Empfangspuffergröße	Anzahl Zeichen, die ohne zwischenzeitiges Auslesen empfangen werden können, da diese im Puffer zwischengespeichert werden.
Sendepuffergröße	Anzahl Zeichen, die ohne zwischenzeitiges Senden in den Puffer zum Senden zwischengespeichert werden können.

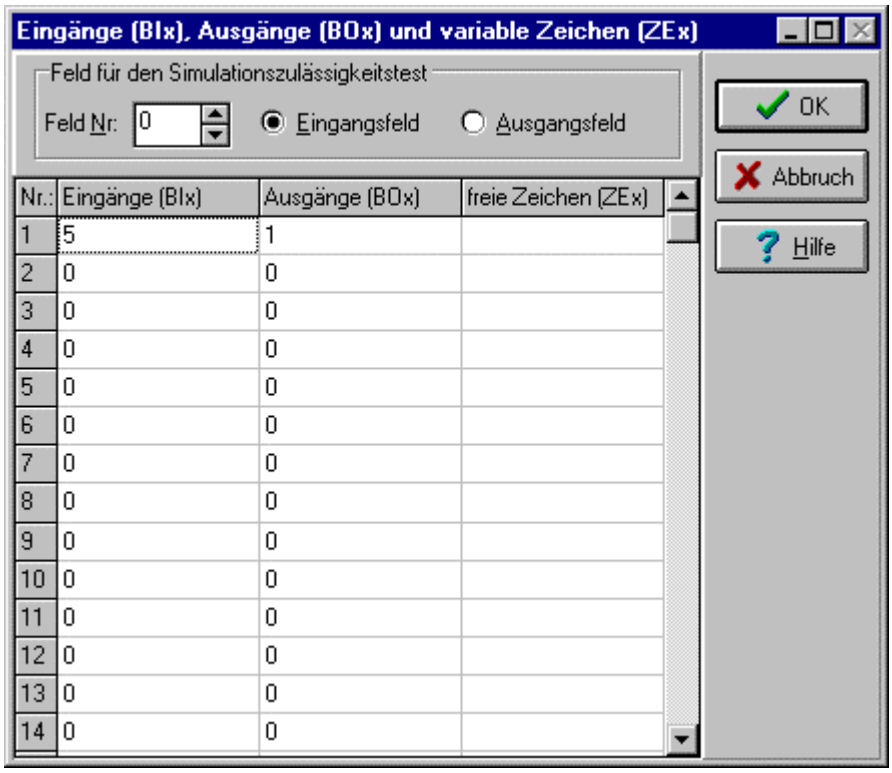
CTS bei Ausgabe	Legt fest, ob das CTS-Signal zur Ausgabesteuerung benutzt werden soll.
DSR bei Ausgabe	Legt fest, ob das DSR-Signal zur Ausgabesteuerung benutzt werden soll.
DTR	Legt die Betriebsart des DTR-Signals fest.
RTS	Legt die Betriebsart des RTS-Signals fest.

Die Übertragung des Zeichens 'A' (=65) sähe bei ungerader Parität mit einem Stopbit folgendermaßen aus:

Start-bit	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Pari-tät	Stop-bit
0	1	0	0	0	0	0	1	0	1	1

Globale Variablen

Globale Variablen, zu denen auch die Blockein- und -ausgänge gehören, können in dem folgenden Dialog bearbeitet werden.



Dialog zur Bearbeitung der globalen Variablen, auf die von jedem Protokoll zugegriffen werden kann

Bedeutung der Variablen

Da ein RS-232-Modul seine zehn Ein- und Ausgänge in diese Tabelle schreibt, sind die ersten zehn Elemente der Spalten BIx und BOx als Ein- bzw. Ausgangswerte dieses Blockes zu interpretieren. Ein Verändern dieser Werte hat also zur Folge, daß das entsprechende Signal manipuliert wird. Die Variablen BIx, BOx und ZEx können innerhalb eines Protokolls verwendet werden. Sie bilden die direkte Schnittstelle zwischen Protokoll und BORIS-Systemstruktur.

Wie schon bei der Erläuterung des Parameterdialogs des RS-232-Moduls angesprochen, kann in dem Protokoll für den Simulationszulässigkeitstest eine globale Variable BIx oder BOx auf einen Wert ungleich 0 für Simulation zulässig bzw. auf den Wert 0 für Simulation nicht zulässig gesetzt werden. Welche Variable aus BI bzw. BO dafür verwendet wird, muß im obigen Dialog kenntlich gemacht werden. Dazu dient der obere Teil des Dialoges. Ist die dort eingetragene *Feld-Nr*. gleich 0 so ist kein Feld spezifiziert. Die Variablen BIx und BOx dürfen nur gültige Zahlenwerte enthalten. Werden dennoch andere Zeichen eingegeben, so wird das betroffene Feld auf 0 gesetzt.

Die Spalte ZEx dient der Speicherung globaler Zeichenketten. Sie kann pro Eintrag maximal 50 Zeichen aufnehmen.

Protokolle und interne Variablen

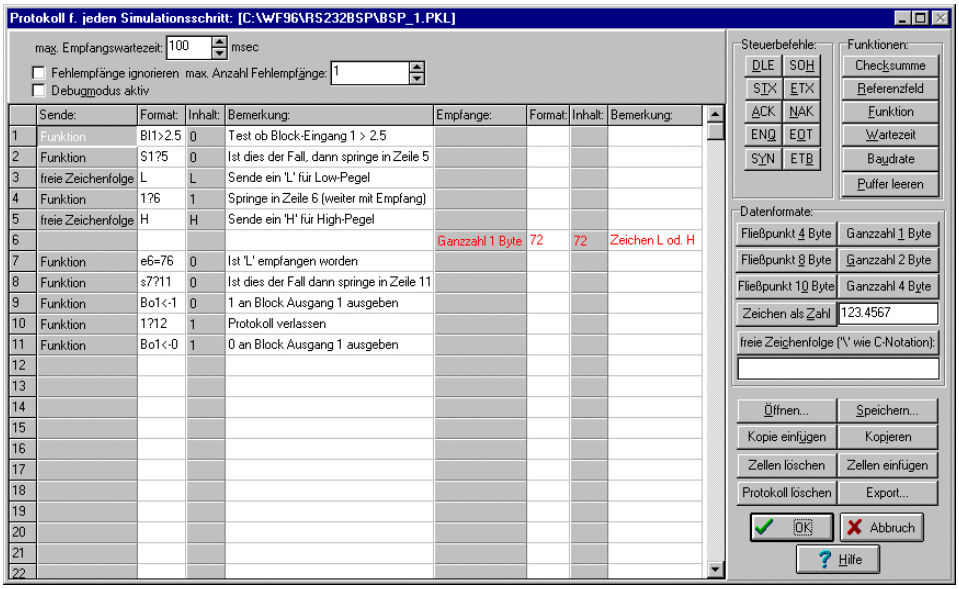
Protokoll-typen

Es gibt fünf verschiedene Protokolle:

(1) 1. Protokoll für den Test, ob eine Simulation zulässig ist
(2) 2. Protokoll für die Initialisierug der Simulation
(3) 3. Protokoll für jeden Simulationsschritt
(4) 4. Protokoll für das Simulationsende
(5) 5. Protokoll für Testzwecke

Alle Protokolle verfügen über exakt die gleichen Möglichkeiten. Lediglich der Ausführungszeitpunkt ist ein anderer.

Die Eingabe eines Protokolls erfolgt mit Hilfe eines entsprechenden Dialoges:



Protokolleingabedialog

Der Dialog gliedert sich in einen linken und einen rechten Teil. Im linken ist das Protokoll zu sehen, im rechten die Argumente, die in das Protokoll eingefügt werden können. Das Protokoll selbst läßt sich wiederum in einen Sende- und einen Empfangsbereich unterteilen. So sind die Spalten 2 bis 5 (die Spalte 1 gibt die Zeilennummer wieder) für die Sendedaten reserviert. Die Spalten 6 bis 9 stellen die Empfangsdaten dar. Die Sende- und Empfangsdaten gliedern sich selbst in einen *Typ* (Spalte *Sende:* bzw. *Emfange:*), ein *Format*, den *Inhalt* und eine *Bemerkung*.

Der Inhalt der Tabelle wird bei Ausführung des Protokolls von links nach rechts zeilenweise abgearbeitet. Dabei werden freigelassene Felder übersprungen. **Es empfiehlt sich, zur Übersichtlichkeit in jeder Zeile entweder nur den Sendeteil oder nur den Empfangsteil der Tabelle auszufüllen.**

Achtung: Alle definierten Protokolle müssen nach einer Änderung explizit gespeichert werden, damit ihre Änderung resident bleibt, da in den eigentlichen BORIS-Systemdateien (BSY-Dateien) nur die Verweise auf die Protokolle, nicht aber die Protokolle selbst abgelegt sind! Das RS-232-Modul gibt im Parameterdialog über entsprechende Grafiken auf den Buttons für die einzelen Protokolle an ob ein Protokoll leer, gespeichert oder nicht gespeichert ist:

(1)  leer

(2)  gespeichert

(3)  nicht gespeichert

Inhalte eines Protokolls

Ein Protokoll kann nicht nur Daten enthalten, die es versenden oder empfangen soll. Es kann auch Befehle zur Datenkonvertierung und Befehle, die den Protokollfluß selbst beeinflussen (z. B. Sprungbefehle), enthalten. Generell soll unterschieden werden zwischen Einträgen, die etwas übertragen und Einträgen, die keine Übertragung bewirken. Zunächst die Einträge, die Daten übertragen.

Einträge, die Daten übertragen

Steuerbefehle:

Die Steuerbefehle übertragen bestimmte, fest vorgegebene Bytes. Diese Steuerzeichen werden oft von komplexeren Protokollen wie z. B. dem Profibus eingesetzt.

Eintrag	überträgt
SOH	#1 start of heading
STX	#2 start of text
ETX	#3 end of text
EOT	#4 end of transmission
ENQ	#5 enquiry
ACK	#6 acknowlegdement
NAK	#21 negative acknowledgement
DLE	#16 data link escape
SYN	#22 synrchonous idle
ETB	#23 end transmission block

Funktionen:

Es gibt lediglich eine Funktion, die Daten überträgt: das Feld *Referenzfeld.*

Eintrag	überträgt
Referenzfeld	Den Inhalt des Feldes, welches durch dieses Feld referenziert wird. **Sx** Inhalt der Zeile x aus dem Sendeteil (x darf nur Zeilennummern enthalten, die kleiner als die dieser Zeile sind) **Ex** Inhalt der Zeile x des Empfangsteils (s. **Sx**). **#SIx** Inhalt der Variable #SIx (1<=x<=10) **#EIx** Inhalt der Variable #EIx (1<=x<=10) **SIx** Inhalt der Zeile *#SIx* aus dem Sendeteil **EIx** Inhalt der Zeile *#EIx* aus dem Empfangsteil **BIx** Inhalt von Block Inputx (1<=x<=50) (siehe Dialog: Eingänge, Ausgänge, Variablen) **BOx** Inhalt von Block Outputx (1<=x<=50) (siehe Dialog: Eingänge, Ausgänge, Variablen) **ZEx** Inhalt einer Zeichenfolge (siehe Dialog: Eingänge, Ausgänge, Variablen) SIx und EIx adressieren indiziert. Sie können als Laufvariablen über Protokollabschnitte definiert werden. Der Inhalt dieser beiden Felder, also #SIx bzw. #EIx, werden vor jeder Protokolldurchführung auf 0 vorinitialisiert.

Datenformate:

Die hier aufgeführten Datenformate ermöglichen die Datenübertragung in dem entsprechenden Format. Es werden Ganzzahlen- und Fließpunktzahlenfomate unterstützt.

Eintrag	überträgt
Fließpunkt 4 Byte	Eine vier Byte lange Fließpunktzahl (IEEE-Format)
Fließpunkt 8 Byte	Eine acht Byte lange Fließpunktzahl (IEEE-Format)
Fließpunkt 10 Byte	Eine zehn Byte lange Fließpunktzahl (IEEE-Format)
Ganzzahl 1 Byte	Ein Byte
Ganzzahl 2 Byte	Zwei Byte
Ganzzahl 4 Byte	Vier Byte
Zeichen als Zahl	Zeichen, die als Zahl interpretierbar sein müssen. Dabei sind Zahleneingaben in Exponentialschreibweise unzulässig.

freie Zeichenfolge eine beliebige Zeichenkette. Der Backslash dient dabei als einleitendes Formatzeichen (Escapesequenzen). So bedeuten

\a	das Zeichen #7 (Beep) des ASCII-Zeichensatzes
\b	das Zeichen #8 (Backspace) des ASCII-Zeichensatzes
\f	das Zeichen #12 (Form feed) des ASCII-Zeichensatzes
\n	das Zeichen #10 (Line feed) des ASCII-Zeichensatzes
\r	das Zeichen #13 (Carriage return) des ASCII-Zeichensatzes
\t	das Zeichen #9 (horizontal Tab) des ASCII-Zeichensatzes
\v	das Zeichen #11 (vertical Tab) des ASCII-Zeichensatzes
\"	das Zeichen "
\?	das Zeichen ?
\\	das Zeichen \
\xHH	das Zeichen des ACII-Zeichensatzes, das an der hexadezimal angegebenen Position HH steht.
\dDDD	das Zeichen des ACII-Zeichensatzes, das an der dezimal angegebenen Position DDD steht.

Stehen diese oben aufgeführten Daten im Sendeteil, so müssen die Fomate gültige Daten enthalten. Ist dies nicht der Fall, erfolgt eine Fehlermeldung. Befinden sich die entsprechenden Einträge im Empfangsteil, so sind die dort eingetragenen Formate als vordefinierte Werte anzusehen, die durch einen korrekten Empfang überschrieben werden.

Beispiel:

Ändern Sie das Protokoll des Beispiels BSP_1.PKL wie folgt ab (auf die Darstellung von Inhalt und Bemerkung wurde aus Platzgründen verzichtet, ebenso findet die Erläuterung der Einträge *Funktion* an späterer Stelle statt):

	Sende:	**Format:**	**Empfange:**	**Format:**
1	Funktion	BI1>2.5		
2	Funktion	S1?5		15
3	*Zeichen als Zahl*	*0*		
4	Funktion	1?6		

5	*Zeichen als Zahl*	*1*		
6			*Zeichen als Zahl*	*9*
7	Funktion	e6=9		
8	Funktion	s7?10		
9	Funktion	BO1<-e6		

Dieses Protokoll erfüllt den gleichen Zweck wie sein Vorgänger. Jedoch hat es den Vorteil, daß es kürzer ist und zusätzlich noch prüft, ob überhaupt ein Zeichen empfangen wurde. In Zeile 6 wird der Inhalt des Empfangsfeldes auf den Wert 9 gesetzt. Wird ein Zeichen empfangen, so wird dieser Inhalt überschrieben. Dieses wird in der nächsten Zeile geprüft und anschließend in Zeile 8 entsprechend verzweigt.

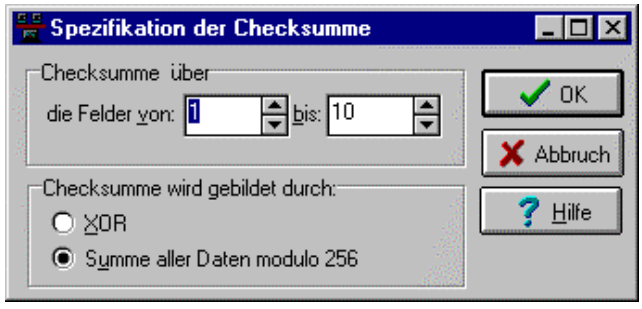
Einträge, die nicht der Übertragung dienen

Es gibt insgesamt vier Einträge, die nicht zur Übertragung beisteuern: *Checksumme, Wait, Funktion, Baudrate* und *Puffer leeren.*

Checksumme

Da sich Checksummen recht kompliziert gestalten, leistet ein Eingabedialog hier Hilfe. Checksummen werden generell über die vor diesem Feld stehende Daten gebildet. Die nachstehende Tabelle erläutert die Funktion Checksumme näher.

Eintrag	**überträgt**
Checksumme	die berechnete Checksumme. Die Checksumme wird nur über Einträge berechnet, die zur Übertragung beisteuern, sie kann demzufolge nicht in der ersten Zeile stehen. Ein Klick auf diesen Button öffnet einen Dialog zur Eingabe der Checksummenart.



Dialog zur Eingabe der Checksumme

Wahlweise kann die Checksumme als *XOR* resp. als *Summe aller Daten modulo 256* aufgebaut werden. Beim Verlassen des Dialoges erfolgt der entsprechende Eintrag in das Protokollgitter:

SUM,-10,-1

Die Zeilenangaben werden hier negativ wiedergegeben, da diese als Offset zur Checksummenzeile anzusehen sind. In der obigen Zeile hieße das für die Generierung der Checksumme:

Bildet den Rest aus der Division der Summe aller Zeilen von der 10-ten vorangehenden bis zur vorangehenden Zeile und 256.

Wait

Ein *Wait* kann nur innerhalb eines Sendeteils eingefügt werden. Die Zahl in der Spalte *Format:* gibt die Wartezeit an genau dieser Stelle des Protokolls in Millisekunden an.

Funktion

Es können nur zwei Argumente innerhalb eines Ausdruck eingegeben werden (z.B: 2+3 ,nicht aber 2+3+1).

Syntax: *Argument1 Operator Argument2*

Ein Argument kann dabei eine Referenz sein (siehe Referenzfeld) oder aber eine konstante Zahl. Ist das Argument eine Referenz, so muß sich hinter dieser ein gültiger Zahlenwert verbergen. Wird dennoch eine freie Zeichenfolge, die keine Zahl darstellt, referenziert, so erhält dieses Argument den Wert 0.

Folgende Operatoren sind vorhanden:

Operator	Operation
+	Addition
-	Subtraktion
*	Multiplikation
/	Division
&	Bitweises AND (bei Verwendung von Fließpunktzahlen wird hier das logische AND angewandt)
|	Bitweises OR (s.o.)
^	Bitweises XOR (s.o.)

&&	logisches AND
||	logisches OR
>	größer als
<	kleiner als
>=	größer oder gleich
<=	kleiner oder gleich
!=	ungleich
=	gleich
?	wenn der linke Ausdruck ungleich 0 ist, erfolgt ein Sprung zur angegebenen Zeile. Die Zeile muß nicht absolut sondern kann auch durch eine Referenz angegeben werden; es entsteht dadurch ein variables Sprungziel. 1 ? #SI1 Ist ein gültiger variabler Sprung in die Zeile, die durch den Wert in #SI1 refernziert wird.
->	Zuweisung an den rechten Operanden
<-	Zuweisung an den linken Operanden
:	Typkonvertierung: Syntax *Typ:Argument* Zulässige Typen sind folgende: **I1** konvertiert zu 1 Byte Ganzzahl **I2** konvertiert zu 2 Byte Ganzzahl **I4** konvertiert zu 4 Byte Ganzzahl **H2** konvertiert in 2 Hexadezimalzeichen **H4** konvertiert in 4 Hexadezimalzeichen **H8** konvertiert in 8 Hexadezimalzeichen **F4** konvertiert zu 4 Byte Fließpunktzahl **F8** konvertiert zu 8 Byte Fließpunktzahl **F10** konvertiert zu 10 Byte Fließpunktzahl **Nx.y** konvertiert zu Zeichen als Zahl. x gibt dabei die Vor kommastellen, y die Nachkommastellen an **Z** konvertiert zu einer Zeichenkette

Ein Funktionsergebnis (ausgenommen die explizite Typkonvertierung) ist eine 4 Byte Ganzzahl, wenn sie als Ganzzahl dargestellt werden kann, sonst eine 10 Byte Fließpunktzahl. Funktionsergebnisse müssen, wenn Sie zur Übertragung beisteuern sollen, durch ein Referenzfeld referenziert werden.

Baudrate

Dieses Feld verändert direkt die eingestellte Baudrate der verwendeten Schnittstelle. Es kann nur im Sendeteil eingefügt werden. Verwenden

Sie die Umschaltung der Baudrate nur, wenn dies unbedingt nötig ist. Da sich die Baudratenänderung auf die Schnittstelle bezieht und mehrere RS-232-Module mit Verwendung gleicher Schnittstelle unter BORIS existieren können, ändert sich die Übertragungsgeschwindigkeit auch für die anderen RS-232-Module.

Puffer leeren

Dieser Eintrag leert den gesamten Puffer. Je nachdem, ob das Feld im Sende- oder Empfangsteil steht, wird der Sende- bzw. Empfangspuffer geleert. Dies hat zur Folge, daß noch nicht gesendete Zeichen resp. noch nicht gelesene Zeichen verloren gehen.

Editieren eines vorhandenen Protokolls

Anhand des obigen Beispiels sollen nun die Editierfähigkeiten des Protokolldialoges durchgesprochen werden. Hier noch einmal das gesamte Protokoll für jeden Simulationsschritt ohne Bemerkungen aus dem Beispiel:

	Sende:	**Format:**	**Inhalt:**	**Empfange:**	**Format:**	**Inhalt:**
1	Funktion	BI1>2.5	0			
2	Funktion	S1?5	0			
3	freie Zeichenfolge	L	L			
4	Funktion	1?6	1			
5	freie Zeichenfolge	H	H			
6			0	Ganzzahl 1 Byte	72	72
7	Funktion	e6=76	0			
8	Funktion	s7?11	0			
9	Funktion	Bo1<-1	48			
10	Funktion	1?12	1			
11	Funktion	Bo1<-0	49			

Zur Eingabe des Protokolls klicken Sie mit der linken Maustaste in das erste Feld des Sendeteils und bewegen nun den Mauszeiger bei gedrückter linker Maustaste in die zweite Zeile. Fügen Sie nun in beide Sendefelder den Eintrag *Funktion* ein. Dazu betätigen Sie den entsprechenden Button im rechten Teil des Dialoges unter der Rubrik *Funktionen*. Anschließend können Sie die Formate der ersten beiden Zeilen des Sendeteils eingeben. Dies erledigen Sie durch einen Mausklick in das entsprechende Feld der Tabelle. Das Feld ist nun selektiert

und Sie können direkt mit der Eingabe ‘BI1>2.5’ fortsetzen. Verwenden Sie die Cursortasten, um in die nächste Zeile zu gehen und dort ‘S1?5’ einzugeben. Auf diese Art können Sie das gesamte Protokoll erstellen. Sie können dabei nahezu gänzlich auf die Maus verzichten, was die Eingabe wesentlich beschleunigt, da nicht zwischen Maus und Tastatur gewechselt werden muß. Bewegen Sie nun den Cursor in die dritte Zeile und betätigen dann die Tastenkombination Alt + c und geben Sie den Buchstaben ‘L’ ein. Verfahren Sie genauso mit den nächsten drei Zeilen. Die letzten fünf Zeilen des Sendeteils sollen über die Tastatur selektiert werden. Unter der Annahme, daß der Cursor in der siebten Zeile in der Spalte *Format* des Sendeteils steht, können Sie durch dreimaliges Drücken der Cursortaste ‘Cursor nach unten bewegen’ bei gleichzeitig gedrückter Shifttaste (Großschreibtaste) die letzen Felder selektieren. Durch Alt + f werden die Felder zu Funktionsfeldern, in denen Sie die oben abgebildeten Formate eingeben können.

Kopieren von Protokollen

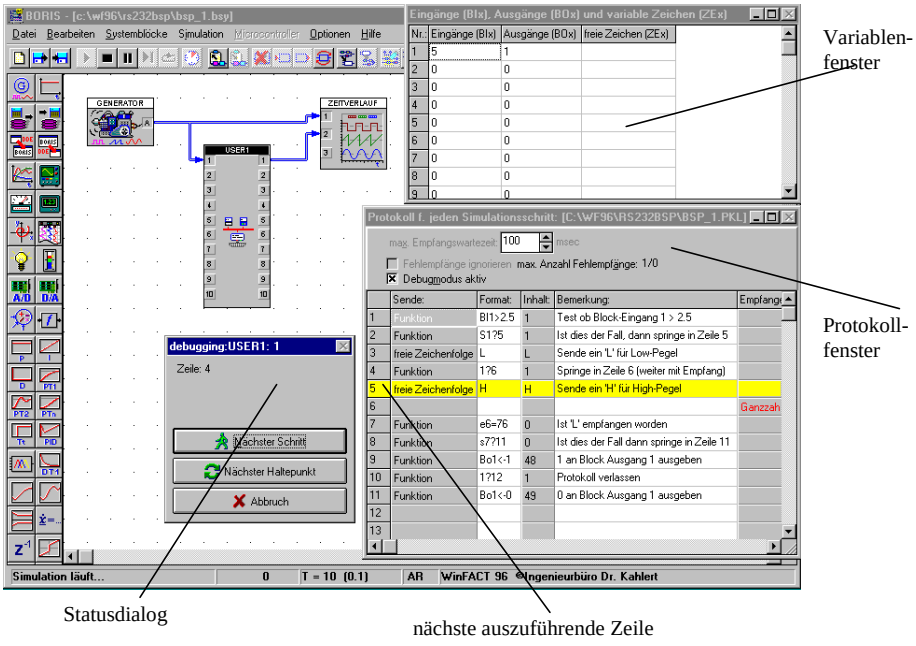
Nachdem Sie das Protokoll eingegeben haben, könnten Sie sich überlegen, daß es vielleicht doch zweckmäßig wäre, dieses auch bei der Simulations-Initialisierung durchführen zu lassen. Dieses läßt sich durch Kopieren und Einfügen dieser Kopie einfach bewerkstelligen. Wählen Sie das gesamte gerade eingegebene Protokoll aus und betätigen anschließend den Button *Kopieren*. Wechseln Sie nun im Protokolleingabedialog zur *Simulations-Initialisierung*, selektieren Sie hier nun die gesamte erste Zeile und betätigen Sie den Button *Kopie* einfügen. Durch diese Vorgehensweise können Sie an einer beliebigen Stelle Teile eines Protokolls einfügen. Wichtig ist, daß Sie zum Kopieren immer den ganzen Sende- oder den ganzen Empfangsteil oder beide komplett selektieren, bevor Sie Kopieren oder Einfügen. Beim Einfügen wird generell der selektierte Teil überschrieben. Soll also nichts überschrieben werden, so ist zunächst eine Zeile einzufügen bevor die Kopie eingefügt wird. Nachfolgend eine kurze Übersicht, welche Funktion die unteren acht Buttons bewerkstelligen:

Button	**Funktion**
Öffnen...	Fordert durch einen weiteren Dialog den Benutzer auf, eine PKL-Datei anzugeben, die dann in diesen Dialog geladen wird.
Speichern...	Fordert durch einen weiteren Dialog den Benutzer auf, eine PKL-Datei anzugeben, in der das in diesen Dialog befindliche Protokoll gespeichert wird.
Kopie einfügen	Fügt den in der Zwischenablage befindlichen Protokollteil an die Stelle des Protokolls ein. Es mussen alle vier Zeilen des Sende- oder des Empfangsteils selektiert werden, wenn die Kopie nur den Sende oder den Empfangsteil umfaßt. Wurden komplette Zeilen beim Kopiervorgang ausgewählt, so muß auch beim Einfügen mindestens eine komplette Zeile des Protokolls angewählt werden. (siehe auch Kopieren)
Kopieren	Kopiert werden können nur vollständige Informationen. Es ist also möglich, zeilenweise die ersten vier Spalten (Sendeteil), die letzten vier Spalten (Empfangsteil) oder alle acht Spalten (Sende und Empfangsteil) des Protokolls zu selektieren und zu kopieren (siehe auch Kopie einfügen). Hingegen können Sie nicht nur den Formatteil eines Protokollteils kopieren.
Zellen löschen	Bevor die Zellen gelöscht werden, wird der selektierte Teil so erweitert, daß er vollständige Informationen umfaßt. Anschließend erfolgt eine Abfrage, ob der nun ausgewählte Teil des Protokolls wirklich gelöscht werden soll.
Zellen einfügen	Fügt die Anzahl der selektierten Zeilen vor der ersten selektierten Zeile ein.
Protokoll löschen	Fragt den Benutzer ob das gesamte Protokoll wirklich gelöscht werden soll.
Export	Fordert durch einen weiteren Dialog den Benutzer auf, eine RTF-Datei (Rich-Text-Format) anzugeben, in der das in diesen Dialog befindliche Protokoll gespeichert wird. Die Speicherung erfolgt formatiert. Die Breite der einzelnen Spalten des Übertragungsprotokoll wird entsprechend der im Dialog dargestellten Breite angepaßt. Sie können diese Datei mit einer unter Windows arbeitenden Textverarbeitung einlesen.

Debugging

Halte-punkte

Das Debugging kann durch Auswahl des *Debug*-Schalters innerhalb eines Protokolleingabe-Dialoges aktiviert werden. Soll nur ab einer bestimmten Zeile eine Fehlersuche gestartet werden, so ist es zweckmäßiger, einen Haltpunkt zu setzen. **Ein Haltepunkt kann durch einen rechten Mausklick auf die entsprechende Zeile und die erste Spalte (Spalte enthält Zeilennummern) des Protokolls gesetzt werden**. Ein weiterer Klick löscht den Haltepunkt wieder. Stellt das RS-232-Modul fest, daß innerhalb eines Protokolls Haltepunkte sind oder der Debugmodus für dieses Protokoll gewählt wurde, so werden bei der Ausführung desselben alle globalen Variablen, das Protokoll selbst und ein Fenster, in dem Sie das Debugging steuern, angezeigt. Die aktuelle Zeile wird innerhalb des Protokolls gelb hinterlegt. Alle Änderungen bezüglich des Protokolls oder der darin befindlichen Variablen werden unmittelbar angezeigt. Das folgende Bild zeigt das Beispiel während der Simulation unter BORIS im Debuggingmodus:



Beispiel während der Simulation unter BORIS im Debuggingmodus

Während der Fehlersuche erscheinen drei Fenster:

Variablenfenster

In diesem Fenster werden die aktuellen Zustände der Variablen angezeigt. Sie sehen also auch die Ein- und Ausgänge des Blockes. Diese werden in den ersten zehn Zeilen in den Spalten BIx und BOx abgebildet. Ein Manipulieren dieser Werte ist während des Debugging nicht möglich.

Protokollfenster

Das Protokollfenster enthält das gerade abzuarbeitende Übertragungsprotokoll. Gelb hervorgehoben ist die als nächstes auszuführende Zeile. Im oberen Teil sind die Anzahl der Fehlempfänge zur zulässigen Anzahl Fehlempfänge eingetragen. Zusätzlich wird angezeigt, ob bei Erreichen der max. zulässigen Anzahl Fehlempfänge das Protokoll verlassen werden soll. Ebenso angezeigt wird die max. Empfangswartezeit und ob der Debugmodus aktiv ist. Der Debugmodus muß nicht unbedingt ausgewählt sein, da Sie ja auch einen Haltepunkt innerhalb des Protokolls gesetzt haben könnten. Lediglich die Einstellung des Debugmodus sowie das Setzen und Löschen von Haltepunkten sind in diesem Dialog zulässig.

Statusfenster

Im Statusfenster, das während der Protokollausführung immer das oberste aller Fenster ist, wird der Zustand der Ausführung wiedergegeben. Es kann sich um einen oder mehrere der nachfolgend beschriebenen Zustände handeln.

CTSHOLD	Bestimmt, ob die Übertragung darauf wartet, daß das CTS-Signal (clear-to-send) gesendet wird.
DSRHOLD	Bestimmt, ob die Übertragung darauf wartet, daß das DSR-Signal (data-set-ready) gesendet wird.
RLSDHOLD	Bestimmt, ob die Übertragung darauf wartet, daß das RLSD-Signal (receive-line-signal-detect) gesendet wird. (RLSD entspricht carrier detect)
TXIM	Bestimmt, ob ein Zeichen darauf wartet, übertragen zu werden.
CE_BREAK	Die Hardware stellte eine Abbruchbedingung fest.
CE_CTSTO	Clear-to-send-Unterbrechung (time out). CTS war für die durch

	im Schnittstellendialog festgelegte Dauer gelöscht, während gleichzeitig versucht wurde, ein Zeichen zu übertragen.
CE_DSRTO	Data-set-ready-Unterbrechung. DSR war für die im Schnittstellendialog festgelegte Dauer gelöscht, während gleichzeitig versucht wurde, ein Zeichen zu übertragen.
CE_FRAME	Die Hardware stellt einen Framing-Fehler fest.
CE_OVERRUN	Ein Zeichen wird von der Hardware nicht rechtzeitig gelesen. Das Zeichen ist verloren.
CE_RLSDTO	Receive-line-signal-detect-Unterbrechung. RLSD war für die im Schnittstellendialog festgelegte Dauer herabgesetzt, während gleichzeitig versucht wurde, ein Zeichen zu übertragen.
CE_RXOVER	Überlauf der Empfangswarteschlange.
CE_RXPARITY	Die Hardware stellte einen Parity-Fehler fest.
CE_TXFULL	Die Sendewarteschlange war voll. Gleichzeitig wurde versucht, ein Zeichen zur Warteschlange hinzuzufügen.

Zeile: n Die Schnittstelle weist bislang keinen Fehler auf,das Protokoll wartet auf die Abarbeitung der Zeile n.

Unter dem angezeigten Status sind drei Buttons, mit denen Sie die Ausführung des im Protokollfenster angezeigten Protokolls steuern. Wahlweise können Sie einen Einzelschritt, das Protokoll bis zum nächsten Haltepunkt ausführen oder aber das Protokoll beenden.

Hinweis für das Beenden des Protokolls für jeden Simulationsschritt: Das Beenden des Protokolls für jeden Simulationsschritt beendet nur für diesen Abtastschritt durchgeführte Protokoll. Das Protokoll wird beim nächsten Simulationsschritt wieder durchgeführt. Dies kann unterbunden werden, wenn die Simulation beendet wird, bevor das Übertra-

gungsprotokoll beendet wird.

An späterer Stelle wird anhand verschiedener Beispiele das Debugging in der Praxis erläutert. Es werden dabei zunächst einfache, später aber auch recht komplexe Protokolle behandelt, so daß Sie ein Gefühl für dem Umgang mit dem RS-232-Modul bekommen.

Simulation mehrerer RS-232 Blöcke

Wie schon erwähnt, können Sie jedem RS-232-Block eine Nummer zuweisen, die den Ausführungszeitpunkt dieses RS-232-Blockes relativ zu den anderen RS-232-Blöcken angibt.

Die Abarbeitung erfolgt folgendermaßen: Zunächst werden alle RS-232-Module in einer Ausführungsliste gesammelt. Ist diese Liste vollständig so wird Sie abgearbeitet. Dies bedeutet, daß alle RS-232-Module direkt nacheinander, ohne daß zwischendurch ein anderer Block der Systemstruktur durchgeführt wird, ausgeführt werden. Durch dieses Vorgehen wird es ermöglicht, die Ausführungsreihenfolge der RS-232-Blöcke selber zu bestimmen. Es besteht somit auch die Möglichkeit, Teile einer Übertragung durch verschiedene Blöcke zu repräsentieren, wodurch die Übersichtlichkeit erhöht werden kann. Ein folgender Anwendungsfall verdeutlicht den Vorteil. Nehmen Sie an, Sie wollten eine beliebige Anzahl von Signalen über ein Modem als Text versenden. Die Problematik besteht darin, daß es Probleme bei der Initialisierungsroutine für das Modem geben könnte, wenn nicht bekannt ist, welcher Block zuerst abgearbeitet wird. Alle übrigen Blöcke können in einer frei definierbaren Reihenfolge ausgeführt werden.

Es ist jedoch nicht möglich, einen einzelnen RS-232-Block zu deaktivieren und dann mit den übrigen eine Simulation durchzuführen. Die Abarbeitungsliste wird bei Deaktivierung eines RS-232-Blockes nicht vollständig gefüllt, da der deaktivierte Block nicht eingetragen werden kann. Dies hat zur Folge, daß keiner der RS-232-Blöcke bearbeitet wird!

RS-232-Modul in Beispielen

In diesem Kapitel soll alles, was bislang angesprochen wurde, an Beispielen erläutert werden. Zunächst werden die Beispiele aus wenigen

Zeilen bestehen. In den späteren Beispielen geht es jedoch um die Bewältigung komplizierterer und umfangreicherer Protokolle.

Beispiel 1: Unbedingter Sprung:

Geben Sie DLE aus und erzeugen Sie einen unbedingten Sprung zu Zeile 10. In dieser soll ACK ausgegeben werden. Zur Kontrolle sollen die ausgegebenen Zeichen im 1 Byte Format zurückgelesen werden und in Zeile 3 die unmittelbar auf der Zeile mit dem Sprung folgt ein NAK ausgegeben werden.

Aufbau des Protokolls:

	Sende	Format:	Inhalt:	Bemerkung;	Empfange	Format	Inhalt	Bemerkung:
1	DLE	16	16					
2	Fkt	1?10	1					
3	NAK	21	21					
4								
5								
6								
7								
8								
9								
10	ACK	6	6					
11					G1	0	0	
12					G1	0	0	

Unbedingter Sprung (Zeile 2)

Wie zu sehen ist, dürfen innerhalb eines Protokolls ganze Zeilen leer bleiben, diese werden bei der Ausführung einfach ignoriert.

Beispiel 2: Bedingter Sprung

In Abhängigkeit eines in Zeile 1 empfangenen Bytes soll ein Sprung zur Zeile 10 wenn der Wert des empfangenen Bytes kleiner als 142 und größer als 32 (entspricht dem sichtbaren ASCII-Zeichenbereich) ist, sonst das Protokoll verlassen werden. In Zeile 10 soll der Text ‘ASCII-Zeichen’ gesendet werden.

Aufbau des Protokolls:

	Sende:	Format:	Inhalt:	Bemerkung:	Empfange:	Format:	Inhalt:	Bemerkung:
1					G1	0	0	Ein Byte empfangen
2	Fkt	e1<142	1	sichtbares ASCII-Zeichen...				
3	Fkt	e1>32	0	... empfangen				
4	Fkt	s2&&s3	0					
5	Fkt	s4?10	0	J1; Sprung falls ASCII-Zeichen				
6	Fkt	1?1000	0	Protokoll verlassen				
7								
8								
9								
10	abc	ASCII-Zeichen	ASCII-Zeichen	J1;				

Bedingter Sprung

Die Bedingung wird in den Zeilen 2 bis 4 des Sendeteils formuliert. In Zeile 5 erfolgt der bedingte Sprung, also eine abhängige Verzweigung des Protokollflusses. Durch einen unbedingten Sprung in Zeile 6 wird das Protokoll direkt verlassen. Hier würde in diesem Fall ein Sprung in die Zeile 11 genügen, da damit schon hinter das Protokollende gesprungen würde was ein direktes Verlassen desselben zur Wirkung hat.

Beispiel 3: Unbedingte Schleife

Es sollen 10 auf einander folgende durch 7 teilbare Zahlen gesendet werden. In BI1 soll die erste Zahl die nicht unbedingt durch 7 teilbar ist stehen. Die Zahlen sind als 8 Byte Hexadezimalzahlen gesendet werden.

Aufbau des Protokolls:

	Sende:	Format:	Inhalt:	Bemerkung:	Empfange:	Format:	Inhalt:
1	Fkt	Bi1 / 7	?	Eingang durch 7 teilen			

2	Fkt	I4 : s1	?	und in Ganzzahl wandeln			
3	Fkt	s2 * 7	?	mit 7 multilizieren.			
4	Fkt	s3 + 7	?	J1: Nächste durch 7 teilbare Zahl			
5	Fkt	s4 -> s3	?	in s3 speichern.			
6	Fkt	H8 : s4	?	formatieren zu HEX			
7	Ref.	s6	?	Hexadezimalzahl senden			
8	Fkt	#si1 + 1	?	Zähler erhöhen			
9	Fkt	#si1 <- s8	?	Erhöhten Zähler speichern			
10	Fkt	#si1 <= 10	?	10 Druchläufe erreicht ?			
11	Fkt	s10 ? 4	?	J1: Sprung			

Unbedingte Schleife

Das Protokoll beginnt damit den Blockeingang 1 durch 7 zu dividieren, das Ergebnis vom Fließkommateil zu befreien und anschließend mit 7 zu multiplizieren. Die erste durch sieben teilbare Zahl steht nun in s3. Von dieser sollen die zehn nächsten als 8 Byte Hexadezimalzahlen übertragen werden. Also wird zunächst zur Zahl in S3, 7 addiert und wieder in S3 gespeichtert (Zeile 4 und 5). Nach der Konvertierung des Ergebnisses wird dies durch das Referenzfeld in Zeile 7 übertragen. Die Zeilen 8 und 9 erhöhen die Variable #SI1, die hier als Zähler verwendet wird um 1. Der Inhalt der Variablen #SIx und #EIx werden zu Beginn der Protokollausführung auf 0 gesetzt. Dadurch muß in diesem Fall der Zähler nicht vorinitialisiert werden. Durch den Vergleich und den bedingten Sprung in Zeile 11 wird die unbedingte Schleife abgeschlossen. Die Terminierung dieser Schleife hängt ausschließlich von der Zählvariable ab; sie kann daher als unbedingte Schleife angesehen werden.

Beispiel 4: Bedingte Schleife

Hardware an COMx: Echo gebendes Gerät

Der Blockeingang 1 soll als ASCII-Zeichen gesendet werden. Für die Genauigkeit genügen 2 Vorkommastellen und 5 Nachkommastellen. Die Bytes des empfangenen Echos sollen immer wieder solange aufsummiert werden, bis die Summe 1000 überschreitet. Die Anzahl der Summierungszyklen soll an BO1 ausgegeben werden.

Aufbau des Protokolls:

Sen-	**Format:**	**Inhalt:**	**Bemerkung:**	**Empfange:**	**Format:**	**Inhalt:**	**Bemerkung:**

	de:							
1	Fkt	N2.5 : Bi1	03.02225	formatieren				
2	Ref.	S1	03.02225	formatiert senden				
3					G1	0	0	byteweise empfangen
4					G1	0	0	
5					G1	0	0	
6					G1	0	0	
7					G1	0	0	
8					G1	0	0	
9					G1	0	0	
10					G1	0	0	
11	Fkt	#EI1 <- 3	0	J1; Zeiger (re-)initialisieren				
12	Fkt	S11+1	1					
13	Fkt	s12 -> s11	0	Zähler erhöhen				
14	Fkt	#EI1>10	0	J2				
15	Fkt	s14 ? 11	0	J1; Zeiger limitieren				
16	Fkt	#SI1 + EI1	0	Summe				
17	Fkt	#SI1 <- s16	0	Summe speichern				
18	Fkt	#EI1 + 1	0					
19	Fkt	#EI1 <- S18	0	Zeiger erhöhen				
20	Fkt	#SI1 <= 1000	0	Summe testen				
21	Fkt	s20 ? 14	0	J2				
22	Fkt	S11->BO1	0					

Bedingte Schleife

Der vordere Teil des Protokolls dient lediglich der Datenausgabe und demEmpfang der gesendeten Daten in dem entsprechen Datenformat. In Zeile 11 wird der Zeiger auf ein Empfangsfeld definiert (EI1 zeigt danach auf E3). Da ein Funktionsfeld, das eine Zuweisung enthält, noch 4 Bytes aufnehmen kann, können wir dieses Feld als Zähler für die Anzahl der Summationen wählen. Dieser Zähler könnte auch in #SI2 abgelegt werden, jedoch kann beim Debugging der Zählerstand nicht direkt eingesehen werden. Nach der Definition der Variable #EI3 in Zeile 11 folgt eine Erhöhung des Zählers um 1. Anschließend wird getestet ob der Zeiger EI1 einen gültigen Wert (<=10, da in Zeile 10 das letzte Empfangsfeld definiert ist) hat. Ist dem nicht so, wird

ihm der Anfangswert 3 zugewiesen. Diese Überprüfung entspricht dem Anfang der Schleife. Die Summenbildung erfolgt in den Zeilen 16 und 17. Zeilen 18 und 19 erhöhen den Zeiger EI1, so daß beim nächsten Durchlauf das nächste Empfangsfeld für die Summenbildung herangezogen wird. Die hier dargestellte bedingte Schleife hat ihr Ende mit den nächsten beiden Zeilen, in denen geprüft wird, ob die Summe schon 1000 überschreitet. Terminiert die Schleife so, wird lediglich noch der Zähler an BO1 zugewiesen.

Da dieses Protokoll schon einige interessantere Dinge zu bieten hat als die vorangehenden, soll hieran das Debuggingkonzept gezeigt werden. Laden Sie also unter BORIS das Beispiel BSP_2.BSY und starten Sie die Simulation. Es erscheinen die drei Debuggingfenster. Gehen Sie nun per Einzelschritt durch dasProtokoll und beobachten Sie die Wirkungen der einzelnen Kommandos. Nachdem Sie den Ablauf des Protokolls nachvollzogen und sichergestellt haben daß die Schleife terminiert, können Sie den Debugmodus verlassen, indem Sie ihn im Protokollfenster ausschalten (Das Protokoll darf dabei keinen Haltepunkt aufweisen).

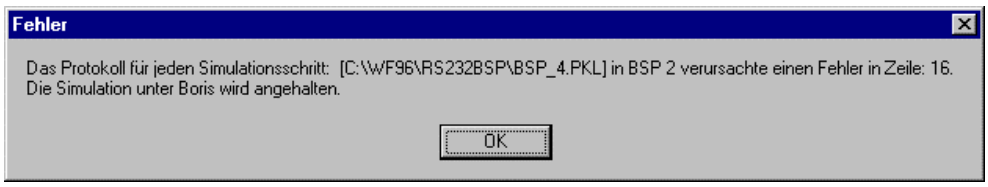
Vorsicht Endlosschleife !

Sollte eine Schleife des Protokolls nicht terminieren, so darf auf keinen Fall der Debugmodus ausgeschaltet werden, da das RS-232-Modul keine Endlosschleifen erkennen kann. Soll das Protokoll aufgrund der Feststellung einer Endlosschleife und auch die Simulation beendet werden, so ist zunächst der *Simulationsende*-Button bzw. der entsprechende Menüpunkt unter BORIS zu betätigen. Hierbei wird jedoch die Simulation noch nicht beendet, sondern nur zum Beenden vorgemerkt. Erst wenn jetzt das Protokoll durch den Abbruch-Button des Statusdialoges beendet wird und keine weiteren Protokolle folgen, wird die Simulation definiert (die Protokolle für das Simulationsende werden auf jeden Fall abgearbeitet) beendet.

Fehlerbehandlung

Fast alle Fehler werden bei der Protokolleingabe abgefangen. Lediglich Zeigeroperatoren (SIx und EIx) können zu diesem Zeitpunkt nicht überprüft werden. Es soll nun das obige Protokoll so verändert werden, daß der Zeiger (EI1) irgendwann ins „Nirwana“ zeigt. Dazu ändern Sie die Zeile 14 von #EI1 >10 in #EI1>11. Diese Änderung bewirkt, daß bis zum elften Empfangsfeld aufsummiert wird. Da in Zeile 11 kein Empfangseintrag vorhanden ist, kann also in diesem Fall nichts Gültiges referenziert werden. Schalten Sie nun noch den De-

bugmodus aus und starten Sie die Simulation. Es erscheint folgende Fehlermeldung:

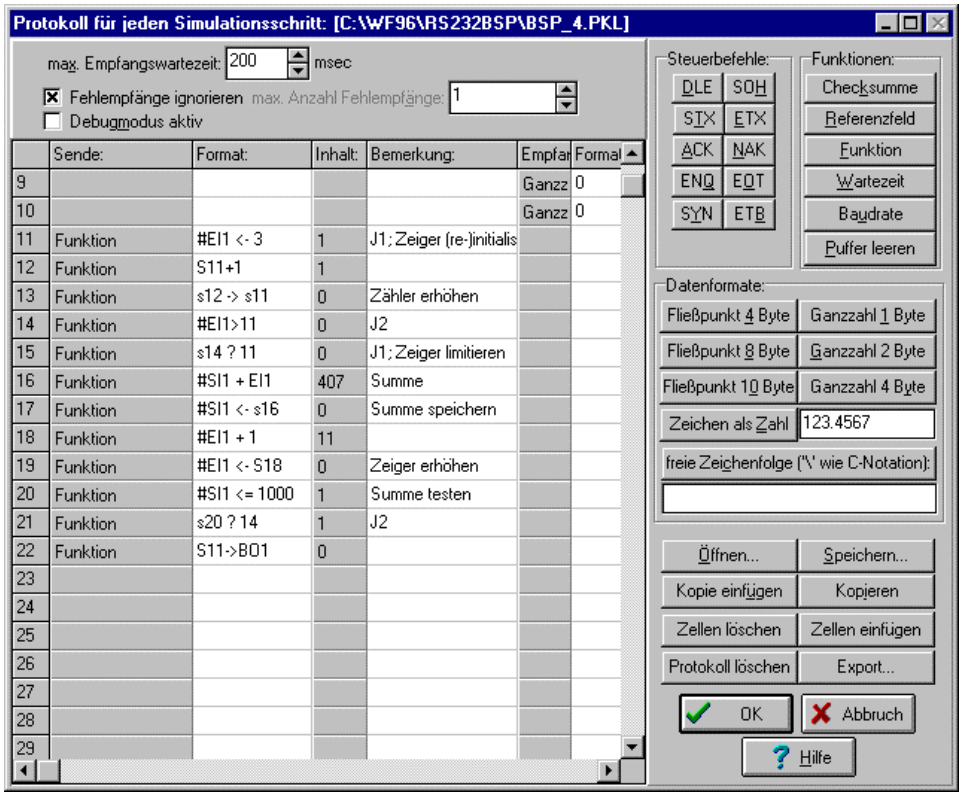


Fehlermeldung bei fehlerhafter Zeigeroperation

Durch die Angaben der Fehlermeldung

- Protokolltyp
- Protokollname (PKL-Dateiname)
- Blockname unter BORIS
- fehlerverursachende Zeile

kann sofort ausfindig gemacht werden, welche Zeigervariable einen fehlerhaften (nicht dereferenzierbaren) Wert enthielt. Wird nun der Protokolleingabedialog des Blockes aufgerufen, so kann diese Zeile gesichtet werden.



	Sende:	Format:	Inhalt:	Bemerkung:	Empfar	Formal
9					Ganzz	0
10					Ganzz	0
11	Funktion	#EI1 <- 3	1	J1;Zeiger (re-)initialis		
12	Funktion	S11+1	1			
13	Funktion	s12 -> s11	0	Zähler erhöhen		
14	Funktion	#EI1>11	0	J2		
15	Funktion	s14 ? 11	0	J1;Zeiger limitieren		
16	Funktion	#SI1 + EI1	407	Summe		
17	Funktion	#SI1 <- s16	0	Summe speichern		
18	Funktion	#EI1 + 1	11			
19	Funktion	#EI1 <- S18	0	Zeiger erhöhen		
20	Funktion	#SI1 <= 1000	1	Summe testen		
21	Funktion	s20 ? 14	1	J2		
22	Funktion	S11->BO1	0			
23						
24						
25						
26						
27						
28						
29						

Zeile 16 verursachte den Fehler. Die dort einzige Zeigervariable ist EI1 (#SI1 wird als Speicher, nicht aber als Zeiger verwendet). Sie ist also die Ursache des Fehlers. In Zeile 18 steht auch noch der Wert, dem die Variable durch Zeile 19 zugewiesen wurde. Letztendlich resultiert der Fehler daraus, daß wir das Limit von 10 auf 11 erhöht haben, den die Zeigervariable nun aufweist. Ändern wir also Zeile 14 in Ihr ursprüngliches Aussehen so ist alles wieder in Ordnung.

Beispiel 5: DLE-Verdopplung und variabler Sprung

Die in der Industrie verwendeten Protokolle besitzen häufig eine Konstellation, daß zunächst eine Anfangsinformation, dann die Daten und schließlich eine Endinformation gesendet werden. Im Datenteil tritt häufig eine sogenannte DLE-Verdopplung auf. Das bedeutet, daß jedes Vorkommen des Wertes des DLE-Bytes (16 dezimal) zweimal gesendet wird. Es soll hier nun ein Protokoll mit DLE-Verdoppelung und variabler Länge empfangen werden. Gesendet werden:

DLE	1 Byte Datensatzlänge	DatenInformationsteil	DLE

Die Datensatzlänge ist die Länge des Datensatzes ohne DLE-Verdopplung. Nach dem Empfang des ganzen Datensatzes soll die

Checksumme als Modulo-Division durch 256 gebildet werden. Diese soll am Blockausgang 1 ausgegeben werden.

Aufbau des Protokolls:

	Sende	Format	Inhalt	Bemerkung	Empfange	Format	Inhalt	Bemerkung
1	DLE	16	16	START				
2	G1	13	13	Datensatzlänge				
3	Fkt	1 ? 11	1	Daten senden				
4	DLE	16	16	ENDE				
5	Fkt	1 ? 50	1	springe zum Empfang				
6								
7								
8								
9								
10								
11	G1	1	1	Zu				
12	G1	2	2	sendender				
13	G1	4	4	Datensatz				
14	G1	8	8					
15	G1	16	16					
16	G1	16	16					
17	G1	32	32					
18	G1	64	64					
19	G1	32	32					
20	G1	16	16					
21	G1	16	16					
22	G1	8	8					
23	G1	4	4					
24	G1	2	2					
25	G1	1	1					
26	Fkt	1 ? 4	1	Rücksprung				

27								
28								
29								
30								
31	G1	0	0	Speicher bereit stellen				
32	G1	0	0	um die 13 zu sendenen				
33	G1	0	0	Bytes ablegen zu können.				
34	G1	0	0					
35	G1	0	0					
36	G1	0	0					
37	G1	0	0					
38	G1	0	0					
39	G1	0	0					
40	G1	0	0					
41	G1	0	0					
42	G1	0	0					
43	G1	0	0					
44	G1	0	0					
45	G1	0	0					
46	Checksumme	SUM;-15;-1	190	Checksumme bilden				
47	Fkt	1 ? #SI10		variabler Sprung (Rücksprung)				
48								
49								
50					DLE	16	16	START
51					G1	0	0	Datensatzlänge
52	Fkt	#Si1<-30	0					
53	Fkt	#Si1 + 1	1	J2				
54	Fkt	s53 ->	0					

		#Si1						
55					G1	0	0	Daten empfangen
56	Fkt	E55 != 16	1	DLE empfangen				
57	Fkt	S56 ? 59	1	J1;				
58					G1	0	0	zusätzliches DLE empfangen
59	Fkt	SI1 <- E55	0	J1; Datum im Speicher ablegen				
60	Fkt	E51 - 1	0					
61	Fkt	E51<- S60	0	Anzahl zu empfangener Bytes				
62	Fkt	E51>0	0					
63	Fkt	s62 ? 53	0	J2				
64					DLE	16	16	ENDE
65	Fkt	#SI10 <- 67	0					
66	Fkt	1 ? 46						
67	Fkt	S46 -> BO1		Checksumme an BO1				

Beispielprotokoll für die Berücksichtigung einer DLE-Verdopplung im Datenteil

Wie zu sehen ist, wurde das Protokoll in verschiedene Abschnitte zergliedert. Zunächst wird in den ersten Zeilen das komplette Sendeprotokoll durchlaufen. Abschließend wird in Zeile 5 zum Empfang gesprungen. Dort wird zunächst die zwei Byte lange Startinformation, die die Datensatzlänge enthält, gelesen. Durch eine unbedingte Schleife, die selbst eine Verzweigung enthält (wenn DLE empfangen wird, soll einfach noch ein Byte empfangen werden) wird der komplette Datensatz von der Schnittstelle eingelesen. Die gültigen Bytes werden in einem im Sendeteil zur Verfügung gestellten Speicher im Byteformat abgelegt.

Nach Terminierung der Empfangsschleife wird die Endemarke empfangen. Nun wird die Checksumme über den Datensatz gebildet. Durch einen variablen Rücksprung (Zeile 47) wird in die Zeile 67 verzweigt in der die Ausgabe der Checksumme an BO1 erfolgt.

