term

Ein Terminalprogramm für Amiga-Computer

Beschreibung der ARexx-Schnittstelle und ARexx-Befehle
16. Juli 1993

von Olaf Barthel

1 Neuigkeiten

Vorausgehende 'term'-Versionen verwendeten eine komplett andere ARexx-Schnittstelle. Um von Commodore verordneten Richtlinien zur Gestaltung von Benutzeroberflächen zu gehorchen, wurde die neue Schnittstelle in Version 3.0 von Grund auf neu geschrieben. Ausführung und Gestaltung orientierten sich am *Amiga User Interface Style Guide* und wurden von Martin Taillefer's *TurboText* ARexx-Schnittstelle beeinflußt.

Kein einziger Befehl der alten ARexx-Schnittstellenimplementierung hat 'überlebt'. Die neue Schnittstelle ist nicht zur alten kompatibel, bestehende ARexx-Programme müssen deshalb überarbeitet oder sogar komplett neu geschrieben werden.

'term' unterscheidet nicht mehr zwischen asychronen und synchronen Befehlen (d.h. Befehlen, die das Hauptprogramm zum Warten zwingen können und Befehlen, deren Behandlung das Hauptprogramm nicht betreffen). Zum Zeitpunkt der Erstellung dieses Dokuments kann davon ausgegangen werden, daß fast alle Befehle synchron abgearbeitet werden, Ausnahmen sind ausdrücklich hervorgehoben.

2 term und ARexx

In diesem Dokument werden die von 'term' unterstützten ARexx(tm)-Befehle[1] beschrieben. Dies ist keine Einführung in die Sprache Rexx selbst, die von Mike F. Cowlishaw auf einem IBM/SP System entwickelt und von William S. Hawes auf dem Amiga implementiert wurde.

ARexx (oder auch Amiga Rexx) ist ein kommerzielles Produkt, das dem AmigaDOS 2.0 Enhancer Package beiliegt. Wer nach einer guten Einführung und Beschreibung der Sprache sucht, dem kann ich das Buch *Die Programmiersprache REXX* von M. F. Cowlishaw, erschienen 1988 im Carl Hanser Verlag München/Wien, '`ISBN 3-446-15195-8`' empfehlen. Die in dieser Dokumentation verwendeten Fachausdrücke und Termini sind dem angegebenen Buch entlehnt.

Der Abschnitt Abschnitt 2.1 [Befehlsausfuehrung], Seite 3 beschreibt kurz und knapp, wie man ARexx-Programm schreibt und zur Ausführung bringt. Weitere Informationen über die Sprachimplementierung sind dem *Handbuch zur Systemsoftware* zu entnehmen.

Üblicherweise richtet 'term' eine Rexx-Wirtsumgebung unter dem Namen `TERM` ein. Startet man mehrere 'term'-Programme, so wird der Name der Umgebung der Nummer des Programmes angepaßt (das erste gestartete Programm wird den Namen `TERM` verwenden, das zweite `TERM.1`, das dritte `TERM.2`, etc.). Der Name kann allerdings durch bestimmte Aufrufparameter verändert werden, siehe hierzu die Dokumentation des Hauptprogrammes.

2.1 Befehlsausführung

Um 'term' zur Ausführung eines Befehls zu bewegen, muß die jeweilige Wirtsumgebung normalerweise direkt angesprochen werden:

```
/* Ansprechen der 'term'-Wirtsumgebung. */

ADDRESS term

/* Aufrufen des 'beepscreen' Befehls. */

BEEPSCREEN
```

Wird ein ARexx-Programm jedoch direkt vom 'term'-Hauptprogramm aus gestartet, so wird es sich jedoch automatisch an die Umgebung wenden, aus der es aufgerufen wurde, d.h. der Befehl `address term` ist überflüssig.

[1] ARexx ist ein eingetragenes Warenzeichen der Wishful Thinking Development Corp

Die meisten Befehle liefern Ergebnis- oder Fehlerwerte zurück. Um an die Ergebniswerte zu gelangen, muß als erste Zeile der Befehl `options results` stehen. Die Ergebniswerte werden dann in der Variable `result` abgelegt:

```
/* Wir gehen davon aus, daß sich das Programm an den Wirt
 * wenden wird, von dem aus gestartet wurde.
 *
 * Befehlsergebnisse zugänglich machen.
 */

OPTIONS RESULTS

/* Fordere eine Eingabe vom Anwender an. */

REQUESTSTRING DEFAULT 'etwas' PROMPT 'Gib etwas ein'

/* Hat der Anwender etwas eingegeben? */

IF rc ~= 0 THEN
   SAY 'keine Eingabe erhalten'
ELSE
   SAY result /* Ausgabe des Ergebnisses. */
```

Fehlerwerte werden immer in der Variable `rc` zurückgeliefert (siehe obiges Beispiel).

Im Fall eines Fehlers (Variable `rc` >= 10) setzt 'term' einen Fehlercode in der Variable `term.lasterror` ab:

```
/* Befehlsergebnisse zugänglich machen. */

OPTIONS RESULTS

/* Einen Fehler produzieren, indem man keine
 * Befehlsparameter angibt.
 */

STOPBITS

/* Den Fehlerwert ausgeben. */

SAY term.lasterror
```

Rexx wandelt Befehlsparameter zur Schlüsselworterkennung in Großbuchstaben um und überprüft diese auf unzulässige Zeichen. Dieser Vorgang verbietet die Verwendung von Doppelpunkten und Leerzeichen in Befehlsparametern, sofern sie nicht in Anführungszeichen eingeschlossen sind. Um es noch komplizierter zu machen, werden in Anführungszeichen eingeschlossene Argumente nicht immer vom Parser erkannt. Dies ist davon abhängig, ob das benutzte Schlüsselwort die gesamte Befehlszeile übernimmt (wie z.B. `TEXT/K/F`, oder ob es nur ein einzelnes Wort erwartet (wie z.B. `TEXT/K`). Im ersten Fall kann der Parameter ohne weiteres

Leerzeichen enthalten, im zweiten Fall muß ein Parameter, der Leerzeichen enthält, in doppelte Anführungszeichen eingeschlossen werden. Aus einem Text wie z.B. `Tee oder Kaffee?` wird damit `'"Tee oder Kaffee?"'`.

```
/* Der folgende Befehl wird die Datei ‘ram:foobar’ nicht
 * verschicken können, da ein Doppelpunkt im Parameter
 * vorkommt:
 */

SENDFILE ram:foobar

/* So wird es richtig gemacht: */

SENDFILE 'ram:foobar'

/* Der folgende Befehl wird die Datei ‘foo bar’ nicht
 * verschicken können, da der Name als zwei Parameter
 * aufgefaßt wird:
 */

SENDFILE foo bar

/* Der folgenden Zeile wird es ebenfalls mißlingen,
 * den Namen korrekt zu übergeben, da der ARexx-Parser
 * den in Anführungszeichen eingeschlossenen Parameter
 * wiederum in zwei Worte zerlegt.
 */

SENDFILE 'foo bar'

/* So wird es richtig gemacht: */

SENDFILE '"foo bar"'

/* Der folgende Befehl wird nicht den Text ‘Hello sailor’
 * verschicken, da die einzelnen Wörter in Großbuchstaben
 * umgewandelt als ‘HELLO SAILOR’ verschickt werden:

SEND Hello sailor

/* So wird es richtig gemacht: */

SEND 'Hello sailor'
```

3 Abbrechen eines Befehls

Programme und Befehle erledigen oft nicht, was der Anwender von ihnen verlangt was es notwendig macht, die Programmausführung anzuhalten. Ein 'einfaches' ARexx-Programm, das keine externen Funktionen oder Wirts-Befehle aufruft, kann immer auf folgende Weise abgebrochen werden:

1. Aufrufen des `HI` Befehls (zu finden in der '`SYS:rexxc`'-Schublade) von der Shell aus. Dieser Befehl versucht, die Ausführung aller gerade laufenden Programme abzubrechen.
2. Läuft ein Programm in einer Umgebung, die ein Ausgabefenster zur Verfügung stellt, ist das Fenster zu aktivieren und die Tasten `Control + C` zu drücken. Ein Unterbrechungssignal wird dadurch verschickt, das das Programm sobald möglich zum Anhalten bringt.

Befehle einer Wirts-Umgebung wie 'term' lassen sich nicht immer ganz so einfach abbrechen, wie oben beschrieben wurde. Sofern 'term' betroffen ist, lassen sich Befehle wie z.B. Abschnitt 4.31 [READ], Seite 30, Abschnitt 4.13 [DELAY], Seite 18 oder Abschnitt 4.55 [WAIT], Seite 46 durch Verschicken eines Unterbrechungssignals auf folgende Weise abbrechen:

1. Sofern das 'term' Programm noch Verbindung zu einem Shell-Fenster hat, das Fenster aktivieren und die Tasten `Control + D` drücken.
2. Wurde das 'term' Programm von einer Shell aus gestartet, ist aber nicht mehr mit ihr verbunden, in der Shell den Befehl `status command term` eintippen, die Zahl, die ausgegeben wird, merken und dann `break <Zahl>`, mit `<Zahl>` entsprechend der ausgegebenen Zahl, eingeben.
3. Drücken der Tastenkombination, die in den Programmeinstellungen unter `Abort ARexx command` eingestellt ist (standardmäßig wäre dies `Rechte Shift-Taste + Linke Shift-Taste + Esscape`). Dies verschickt ein Unterbrechungssignal an das 'term' Programm.

4 Befehle

Die von 'term' unterstützten Befehle werden in einer Tabelle der folgenden Form angegeben:

`Format:`

Der Name des Befehls mit möglichen Aufrufparametern. In dieser Tabelle werden mögliche Parameter in Klammern, getrennt durch Kommas und Striche angegeben; *diese Zeichen dürfen nicht mit eingegeben werden!*:

`< > (Spitze Klammern)`

Spitze Klammern umschließen Parameter, die **unbedingt** notwendig sind und nicht fortgelassen werden können.

`[ ] (Eckige Klammern)`

Eckige Klammern umschließen optionale Parameter, die nicht eingegeben werden müssen.

`{ } (Geschweifte Klammern)`

Geschweifte Klammern umschließen Parameter, die mehrfach eingegeben werden können, wie z.B. Listen von Dateinamen.

`| (Senkrechter Strich)`

Senkrechte Striche trennen alternative, sich gegenseitig ausschließende Parameter.

`, (Komma)`

Kommas trennen mehrere verwendbare Parameter.

`Befehlsmuster:`

Das Muster, nach dem der Befehl aufgerufen wird, ähnlichen den Befehlsmustern, die auch die AmigaDOS-Kommandoebene verwendet. Mögliche Muster sind:

`<Parameter>/A`

Der Parameter muß immer mit angegeben werden.

`<Parameter>/K`

Das zum Parameter gehörende Schlüsselwort muß mit angegeben werden.

`<Parameter>/S`

Der Parameter arbeitet als Schalter. Wird der Parameter angegeben, so wird der Schalter aktiviert, ansonsten bleibt er inaktiv.

`<Parameter>/N`

Als Parameter wird eine Zahl erwartet.

`<Parameter>/M`

Mehrere Parameter werden akzeptiert.

`<Text>/F`

Der Text wird als letzter Parameter des Befehls erwartet.

`, (Komma)`

Gibt an, daß keine Parameter erwartet werden.

`Funktion:`

Beschreibt kurz Sinn und Zweck des Befehls.

`Beschreibung:`

Beschreibt den Befehl und seine möglichen Verwendungszwecke genauer.

`Ergebnis:`

Der Typ des Ergebnisses, sofern der Befehl eines zurückliefert.

`Warnung:`

Falls der Befehl mit einem Warncode (Variable `rc` = 5) zurückkehren kann und wann dies der Fall ist.

`Beispiel:`

Ein kurzes Beispielprogramm, das illustriert, wie der Befehl angewendet werden kann. Schlüsselwörter und Befehle sind in Großbuchstaben geschrieben, die Namen von Variablen und Argumenten in Kleinbuchstaben. Lange Befehlszeilen, die nicht in eine Textzeile passen wollten, sind auf zwei direkt aufeinander folgende Zeilen verteilt worden. Das Auslassungszeichen ('...') hat in diesem Zusammenhang die Bedeutung, beide Zeilen zusammenzufügen.

4.1 Der ACTIVATE Befehl

`Format:`

ACTIVATE

`Befehlsmuster:`

,

`Funktion:`

Bringt das Programm in einen Zustand zurück, in dem Eingaben erfolgen können.

`Beschreibung:`

Das Programm kann mit Hilfe des Befehls Abschnitt 4.12 [DEACTIVATE], Seite 17 deaktiviert werden. Um es wieder zu reaktivieren ist der Befehl `ACTIVATE` zu benutzen. Wird dieser Befehl verwendet, während das Programm noch aktiv ist, so wird nur das Hauptfenster aktiviert und in den Vordergrund gebracht.

`Ergebnis:`

-

`Warnung:`

-

`Beispiel:`

```
/* So wird das Programm (re-)aktiviert: */

ACTIVATE
```

4.2 Der ADDITEM Befehl

`Format:`

ADDITEM [To] <Upload|Download|Dial|Wait> [Before|After] [Phone <Eintragsnummer, Name oder Muster>] [Name <Name>]

`Befehlsmuster:`

TO/A,BEFORE/S,AFTER/S,PHONE/K/F,NAME/K/F

`Funktion:`

Fügt einen Eintrag (einen Namen, Telefonnummer, Text, usw.) vor oder hinter den gerade ausgewählten Eintrag ein.

`Beschreibung:`

'term' verwaltet eine Reihe von Listen, dies sind:

`Upload Liste`
: Die Liste der Dateien, die zu verschicken sind.

`Download Liste`
: Die Liste der Dateien, die das Programm empfangen hat.

`Dial Liste`
: Die Liste der Telefonnummern oder Telefonbucheinträge, die anzuwählen sind.

`Wait Liste`
: Die Liste der Texte, auf die der Befehl Abschnitt 4.55 [WAIT], Seite 46 warten soll.

Neue Listeneinträge können mit dem Befehl `ADDITEM` hinzugefügt werden. Die Upload Liste erwartet die Namen von Dateien die der Befehl Abschnitt 4.49 [SENDFILE], Seite 42 verschicken soll. Es macht wenig Sinn, Dateinamen zur Download Liste hinzuzufügen, es dennoch aber möglich. Die Wait Liste erwartet Textzeilen, die der Befehl Abschnitt 4.55 [WAIT], Seite 46 im Eingabestrom suchen soll.

Die Dial Liste erwartet eine Reihe verschiedener Parameter:

`Nummern von Telefonbucheinträgen`
: Diese werden mit Hilfe des `Phone` Parameters übergeben und sollten einen Zahlenwert darstellen, über den der dazugehörende Eintrag aus dem Telefonbuch ermittelt wird.

`Namen von Telefonbucheinträgen`
: Diese werden ebenfalls mit Hilfe des `Phone` Parameters übergeben und können entweder gültige Namen oder Namensmuster darstellen.

`Telefonnummern`
: Diese werden über den `Name` Parameter übergeben.

Einträge können vor oder hinter dem gegenwärtig ausgewählten Listeneintrag (siehe Befehl Abschnitt 4.46 [SELECTITEM], Seite 40) eingefügt werden. Üblicherweise werden sie hinter dem gegenwärtig ausgewählten Listeneintrag eingefügt.

`Ergebnis:`

-

`Warnung:`

-

`Beispiel:`

```
/* Befehlsergebnisse zugänglich machen. */

OPTIONS RESULTS

/* Einen Dateinamen vom Anwender anfordern. */

REQUESTFILE TITLE '"Zu verschickende Datei auswählen"'

/* Den Dateinamen in die Upload Liste einreihen. */

IF rc = 0 THEN ADDITEM TO upload NAME result

/* Telefonbucheintrag in die Dial Liste eintragen. */

ADDITEM TO dial PHONE 2

/* Alle Telefonbucheinträge, deren Namen mit einem
 * 'a' beginnt in die Dial Liste eintragen.
 */

ADDITEM TO dial PHONE a#?

/* Eine einfache Telefonnummer in die Dial
 * Liste eintragen.
 */

ADDITEM TO dial NAME 424242
```

4.3 Der BAUD Befehl

`Format:`

BAUD [Rate] <Baudrate in Bits pro Sekunde>

`Befehlsmuster:`

RATE/A/N

`Funktion:`

Stellt die Übertragungsgeschwindigkeit der seriellen Schnittstelle ein.

`Beschreibung:`

Setzt die Übertragungsgeschwindigkeit der seriellen Schnittstelle auf einen definierten Wert. Der `Rate` Parameter wird mit allen von 'term' unterstützten Geschwindigkeiten verglichen und der am nächsten liegende Wert übernommen.

`Ergebnis:`

-

`Warnung:`

-

`Beispiel:`

```
/* Übertragungsgeschwindigkeit auf 2400 Baud setzen. */

BAUD 2400
```

4.4 Der BEEPSCREEN Befehl

`Format:`

BEEPSCREEN

`Befehlsmuster:`

,

`Funktion:`

Erzeugt ein audiovisuelles Signal.

`Beschreibung:`

Ruft die Signalfunktion auf, gerade so, wie es in den Einstellungen des Hauptprogrammes vorgesehen ist.

`Ergebnis:`

-

`Warnung:`

-

`Beispiel:`

```
/* Signal aufrufen. */

BEEPSCREEN
```

4.5 Der CALLMENU Befehl

`Format:`

CALLMENU [Title] <Titelname oder Muster>

`Befehlsmuster:`

TITLE/A/F

`Funktion:`

Ruft die zu einem Menü gehörende Funktion auf.

`Beschreibung:`

Ruft die mit einem Menüeintrag verbundene Funktion auf, gerade so, als hätte der Anwender es mit der Maus getan. Der `Title` Parameter kann jedem gültigen Menünamen oder einem Namensmuster entsprechen. In letzterem Fall wird nur der erste passende Menüeintrag aufgerufen.

`Ergebnis:`

-

`Warnung:`

Falls kein passender Menütitel gefunden werden konnte.

`Beispiel:`

```
/* Aufruf des 'About...' Menüs. */

CALLMENU abou#?
```

4.6 Der CAPTURE Befehl

`Format:`

CAPTURE [To] <Printer|File> [Name <Dateiname>]

`Befehlsmuster:`

TO/A,NAME/K

`Funktion:`

Aktiviert Mitschnitt auf Drucker oder in Datei.

`Beschreibung:`

Sofern nicht bereits ein Mitschnitt gemacht wird, wird empfangener Text auf dem Drucker oder in einer zu öffnenden Datei ausgegeben. Ist der `File` Parameter angegeben, aber kein Dateiname, so wird nach einem Namen gefragt.

`Ergebnis:`

-

`Warnung:`

Falls der Anwender eine Datei auswählen sollte, dies aber nicht getan hat.

`Beispiel:`

```
/* Öffne eine Mitschnittsdatei. */

CAPTURE TO file NAME 'ram:Mitschnitt'

/* Schließe die Mitschnittsdatei, frage nach einer neuen. */

CLOSE FILE
CAPTURE TO file

/* Aktiviere Mitschnitt zum Drucker. */

CAPTURE TO printer
```

4.7 Der CLEAR Befehl

`Format:`

CLEAR [From] <Upload|Download|Dial|Wait|Buffer> [Force]

`Befehlsmuster:`

FROM/A,FORCE/S

`Funktion:`

Löscht den Inhalt einer Liste oder des Textpuffers.

`Beschreibung:`

Dieser Befehl löscht den Inhalt einer der Listen, die mit Hilfe der Befehle Abschnitt 4.2 [ADDITEM], Seite 11, Abschnitt 4.34 [REMITEM], Seite 32, Abschnitt 4.46 [SELECTITEM], Seite 40, usw. bearbeitet werden können oder den Inhalt des Textpuffers. In letzterem Fall fordert das Programm üblicherweise noch eine Bestätigung an, falls sich noch Text im Puffer befindet. Dies kann mit Hilfe des `Force` Parameters unterdrückt werden.

`Ergebnis:`

-

`Warnung:`

Falls der Anwender das Löschen des Textpuffers nicht bestägigen wollte.

`Beispiel:`

```
/* Löschen der Wait Liste. */

CLEAR FROM wait

/* Löschen des Textpuffers, Frage nach Bestätigung. */

CLEAR FROM buffer

/* Falls keine Bestägigung vorliegt, Löschen ohne Abfrage. */

IF rc ~= 0 THEN CLEAR FROM buffer FORCE
```

4.8 Der CLEARSCREEN Befehl

`Format:`

CLEARSCREEN

`Befehlsmuster:`

,

`Funktion:`

Löscht den Inhalt des Terminalbildschirms.

`Beschreibung:`

Löscht den Inhalt des Terminabildschirms, positioniert die Schreibmarke (Cursor) in der linken oberen Ecke.

`Ergebnis:`

-

`Warnung:`

-

`Beispiel:`

```
/* Löschen des Terminalbildschirms. */

CLEARSCREEN
```

4.9 Der CLOSE Befehl

`Format:`

CLOSE [From] <Printer|File|All>

`Befehlsmuster:`

FROM/A

`Funktion:`

Beendet Mitschnitt auf Drucker oder in Datei.

`Beschreibung:`

Beendet einen über den Befehl Abschnitt 4.6 [CAPTURE], Seite 14 gestarteten Mitschnitt auf Drucker (`Printer`), in eine Datei (`File`) oder beide (`Both`).

`Ergebnis:`

-

`Warnung:`

-

`Beispiel:`

```
/* Beende jeglichen Mitschnitt. */

CLOSE ALL
```

4.10 Der CLOSEDEVICE Befehl

`Format:`

CLOSEDEVICE

`Befehlsmuster:`

,

`Funktion:`

Gibt die serielle Schnittstelle frei.

`Beschreibung:`

Gibt die serielle Schnittstelle frei, sodaß sie auch von anderen Programmen genutzt werden kann. Die Schnittstelle kann über den Befehl Abschnitt 4.23 [OPENDEVICE], Seite 25 wieder aktiviert werden.

Ergebnis:

-

Warnung:

-

Beispiel:

```
/* Gib die serielle Schnittstelle frei. */

CLOSEDEVICE
```

4.11 Der CLOSEREQUESTER Befehl

Format:

CLOSEREQUESTER

Befehlsmuster:

,

Funktion:

Schließt das gerade geöffnete Programmfenster.

Beschreibung:

Schließt das gerade geöffnete Programmfenster, wie z.B. das Wählfenster, das Telefonbuch, usw. Fenster wie das Transferfenster werden hiervon nicht beeinflußt.

Ergebnis:

-

Warnung:

-

Beispiel:

```
/* Schließe das gerade geöffnete Fenster,
 * welches auch immer.
 */

CLOSEREQUESTER
```

4.12 Der DEACTIVATE Befehl

Format:

DEACTIVATE

Befehlsmuster:

,

Funktion:

Versetzt das Programm in Schlaf.

`Beschreibung:`

Deaktiviert das Programm und versetzt es in Schlaf. Erfordert daß die Workbench aktiv ist, sodaß ein AppIcon angebracht werden kann. Dieser Befehl wird ignoriert, sofern das Programm bereits deaktiviert ist. Um das Programm zu reaktivieren, benutze man den Befehl Abschnitt 4.1 [ACTIVATE], Seite 10.

`Ergebnis:`

-

`Warnung:`

-

`Beispiel:`

```
/* Deaktivieren des Programms. */

DEACTIVATE
```

4.13 Der DELAY Befehl

`Format:`

DELAY [MIC|MICROSECONDS <Zahl>] [[SEC|SECONDS] <Zahl>] [MIN|MINUTES <Zahl>] [QUIET]

`Befehlsmuster:`

MIC=MICROSECONDS/K/N,SEC=SECONDS/N,MIN=MINUTES/K/N,QUIET/S

`Funktion:`

Verzögert die Programmausführung.

`Beschreibung:`

Hält die Ausführung des Programmes für eine gewisse Zeitperiode an. Von der Schnittstelle eingehende Zeichen werden auf dem Bildschirm ausgegeben, sofern nicht die `QUIET`-Option benutzt wird.

`Ergebnis:`

-

`Warnung:`

Falls der Befehl vor Beendigung unterbrochen wurde.

`Beispiel:`

```
/* Warte fünf Sekunden lang. */

DELAY 5

/* Warte fünf Sekunden und 7 Mikrosekunden lang. */

DELAY MIC 7 SEC 5
```

4.14 Der DIAL Befehl

`Format:`

DIAL [[Num] <Telefonnummer>]

`Befehlsmuster:`

NUM/F

`Funktion:`

Wählt die angegebene Telefonnummer. Falls keine Telefonnummer angegeben wurde, wählt die in der Dial Liste angegebenen Nummern.

`Beschreibung:`

Dieser Befehl erstellt eine Wählliste entweder aus der angegebenen Telefonnummer oder der belegten Dial Liste.

Dieser Befehl wird beendet, sobald der Wählvorgang eingeleitet ist.

`Ergebnis:`

-

`Warnung:`

-

`Beispiel:`

```
/* Wähle eine einzelne Telefonnummer. */

DIAL 424242

/* Warte ein wenig, brich den Wählvorgang ab. */

DELAY 5
CLOSEREQUESTER

/* Lösche die Dial Liste, füge alle Telefonbucheinträge
 * hinzu.
 */

CLEAR FROM dial
ADDITEM TO dial PHONE #?

/* Wähle die Dial Liste. */

DIAL
```

4.15 Der DUPLEX Befehl

`Format:`

DUPLEX [Full|Half|Echo]

`Befehlsmuster:`

FULL/S,HALF=ECHO/S

`Funktion:`

Stellt den Duplex-Modus der seriellen Schnittstelle ein.

`Beschreibung:`

Stellt den Duplex-Modus der seriellen Schnittstelle ein, dies kann entweder Vollduplex (`Full`) oder Halbduplex (`Half`, `Echo`) sein.

`Ergebnis:`

-

`Warnung:`

-

`Beispiel:`

```
/* Halbduplex aktivieren. */

DUPLEX ECHO
```

4.16 Der EXECTOOL Befehl

`Format:`

EXECTOOL [Console] [Async] [Port] [Command] <File name>

`Befehlsmuster:`

CONSOLE/S,ASYNC/S,PORT/S,COMMAND/A/F

`Funktion:`

Führt ein Programm aus.

`Beschreibung:`

Lädt und führt ein AmigaDOS-Programm aus. Der `Console` Parameter bewirkt, daß ein Ausgabefenster für das Programm geöffnet wird, der `Async` Parameter bewirkt, daß das Programm geladen und gestartet wird, der Befehl jedoch sofort beendet wird. Der `Port` Parameter führt dazu, daß der Name des Rexx-Wirts als Aufrufparameter an das gestartete Programm übergeben wird.

`Ergebnis:`

-

`Warnung:`

-

`Beispiel:`

```
/* Starte den ‘Dir’ Befehl. */

EXECTOOL CONSOLE COMMAND 'dir c:'
```

4.17 Der FAULT Befehl

`Format:`

FAULT [Code] <Fehlercode>

`Befehlsmuster:`

CODE/A/N

`Funktion:`

Liefert den zu einem Fehlercode gehörenden Fehlertext.

`Beschreibung:`

'term' legt Fehlercodes in der Variable `term.lasterror` ab. Um den zu jedem Fehler gehörenden Fehlertext zu erhalten, benutze man diesen Befehl. Es werden sowohl die internen Rexx- und AmigaDOS-Fehlercodes als auch die 'term' eigenen Fehlercodes unterstützt.

`Ergebnis:`

Der zum Fehler gehörende beschreibende Text.

`Warnung:`

-

`Beispiel:`

```
/* Befehlsergebnisse zugänglich machen. */

OPTIONS RESULTS

/* Zum Fehler 1001 gehörenden Text abrufen. */

FAULT 1001

/* Ergebnis ausgeben. */

SAY result
```

4.18 Der GETATTR Befehl

`Format:`

GETATTR [Object] <Name> [Field] <Name> [Stem <Name>] [Var <Name>]

`Befehlsmuster:`

OBJECT/A,FIELD,STEM/K,VAR/K

`Funktion:`

Fragt ein Attribut des Programms ab.

`Beschreibung:`

Liefert Informationen über ein Stammobjekt, legt das Ergebnis nach Möglichkeit in der Variable `result` ab. Wird eine Variable über die `Stem` oder `Var` Parameter übergeben, legt die Informationen an dieser Stelle ab. Wird kein `Field` Parameter angegeben, legt das gesamte Stammobjekt in der über den `Stem` Parameter angegebenen Stammvariable ab.

Eine Liste der gültigen Attribute ist unter Abschnitt 5.23 [Attribute], Seite 63 zu finden.

`Ergebnis:`

Legt die Information entweder in der Variable `result` oder in der über `Stem`, bzw. `Var` angegebenen Variable ab.

`Warnung:`

-

`Beispiel:`

```
/* Befehlsergebnisse zugänglich machen. */

OPTIONS RESULTS

/* Ermittle den Namen des Rexx-Wirts. */

GETATTR OBJECT term FIELD arexx

/* Ausgabe des Ergebnisses. */

SAY result

/* Wie oben, jedoch in einer anderen Syntax. */

GETATTR term.arexx
SAY result

/* Ablegen des gesamten Telefonbuchs in einer
 * Stammvariable.
 */

GETATTR phonebook STEM book

/* Ausgabe des Telefonbuchs. */

SAY 'Telefonbuch enthält' book.count 'Einträge'

DO i = 0 TO book.count - 1
   SAY 'Eintrag Nr.' i

   SAY 'Name        :' book.i.name
   SAY 'Nummer      :' book.i.number
   SAY 'Kommentar   :' book.i.commenttext
   SAY 'Benutzername:' book.i.username
END i
```

4.19 Der GETCLIP Befehl

`Format:`

GETCLIP [Unit <Nummer>]

`Befehlsmuster:`
UNIT/K/N

`Funktion:`
Liest den Inhalt des Klemmbretts aus.

`Beschreibung:`
Liest den Inhalt des Klemmbretts aus und liefert ihn in der Variable `result` zurück. Liest entweder eine bestimmte Einheit des Klemmbretts aus, oder verwendet die in den Programmeinstellungen angegebene Einheit. *Vorsicht: es können bis zu 65.536 Zeichen gelesen werden!*

`Ergebnis:`
Inhalt des Klemmbretts, sofern es Text enthält.

`Warnung:`
Falls das Klemmbrett keinen Text enthält.

`Beispiel:`

```
/* Befehlsergebnisse zugänglich machen. */

OPTIONS RESULTS

/* Inhalt des Klemmbretts auslesen. */

GETCLIP

/* Inhalt ausgeben. */

IF rc ~= 0 THEN
   SAY 'Klemmbrett enthält keinerlei Text'
ELSE
   SAY result
```

4.20 Der HANGUP Befehl

`Format:`
HANGUP

`Befehlsmuster:`
,

`Funktion:`
Bricht die Telefonverbindung ab.

`Beschreibung:`
Bricht die Telefonverbingung ab (legt auf), führt dazugehörige Makros aus.

`Ergebnis:`
-

`Warnung:`
-

Beispiel:

```
/* Abbruch der Verbindung. */

HANGUP
```

4.21 Der HELP Befehl

Format:

HELP [[Command] <Name>] [Prompt]

Befehlsmuster:

COMMAND,PROMPT/S

Funktion:

Liefert das Befehlsmuster eines Befehls oder aktiviert die Online-Hilfe.

Beschreibung:

Dieser Befehl liefert entweder das zum über den `Command` Parameter angegebenen Befehl gehörende Befehlsmuster oder aktiviert die AmigaGuide(tm) Online-Hilfe.

Ergebnis:

Angefordertes Befehlsmuster.

Warnung:

-

Beispiel:

```
/* Befehlsergebnisse zugänglich machen. */

OPTIONS RESULTS

/* Zum 'selectitem' Befehl gehörendes Befehlsmuster
 * abrufen.
 */

HELP selectitem

/* Ergebnis ausgeben. */

SAY result

/* Online-Hilfe starten. */

HELP PROMPT
```

4.22 Der OPEN Befehl

Format:

OPEN [Name <Dateiname>] [TO] <Translations|Functionkeys|Cursorkeys| Fastmacros|Hotkeys|Speech|Sound|Buffer|Configuration|Phone>

`Befehlsmuster:`

NAME/K,TO/A

`Funktion:`

Liest Einstellungen, das Telefonbuch oder den Textpuffer.

`Beschreibung:`

Dieser Befehl liest den Inhalt einer Datei und legt die gelesenen Information in den Programmeinstellungen, dem Telefonbuch oder dem Textpuffer ab. Falls Text in den Puffer gelesen wird, so wird er an den bereits bestehenden Text angehängt. Ist kein Dateiname angegeben, so wird der Anwender gebeten, eine Datei auszuwählen.

`Ergebnis:`

-

`Warnung:`

Falls der Anwender eine Datei auswählen sollte, dies aber nicht getan hat.

`Beispiel:`

```
/* Lade die Programmeinstellungen. */

OPEN NAME 'ram:term.prefs' TO configuration

/* Füge Text zum Puffer hinzu. */

OPEN TO buffer
```

4.23 Der OPENDEVICE Befehl

`Format:`

OPENDEVICE [Name <Dateiname>] [Unit <Nummer>]

`Befehlsmuster:`

NAME/K,UNIT/K/N

`Funktion:`

(Re-)Aktiviert die serielle Schnittstelle.

`Beschreibung:`

(Re-)Aktiviert die serielle Schnittstelle (die über den Befehl Abschnitt 4.10 [CLOSEDEVICE], Seite 16 freigegeben wurde) oder eine andere Schnittstelle, sofern die `Device` oder `Unit` Parameter angegeben sind.

`Ergebnis:`

-

`Warnung:`

-

`Beispiel:`

```
/* Schnittstelle freigeben. */

CLOSEDEVICE

/* Einen anderen Treiber verwenden. */

OPENDEVICE DEVICE 'duart.device' UNIT 5
```

4.24 Der OPENREQUESTER Befehl

`Format:`

OPENREQUESTER [REQUESTER] <Serial | Modem | Screen | Terminal | Emulation | Clipboard | Capture | Commands | Misc | Path | Transfer | Translations | Functionkeys | Cursorkeys | Fastmacros | Hotkeys | Speech | Sound | Phone>

`Befehlsmuster:`

REQUESTER/A

`Funktion:`

Öffnet ein Programmfenster.

`Beschreibung:`

Öffnet ein Programmfenster, von denen jeweils nur eines geöffnet sein kann. Das geöffnete Fenster kann mit Hilfe des Befehls Abschnitt 4.11 [CLOSEREQUESTER], Seite 17 wieder geschlossen werden.

`Ergebnis:`

-

`Warnung:`

-

`Beispiel:`

```
/* Öffne das Telefonbuchfenster. */

OPENREQUESTER phone
```

4.25 Der PARITY Befehl

`Format:`

PARITY [Even | Odd | None | Mark | Space]

`Befehlsmuster:`

EVEN/S,ODD/S,NONE/S,MARK/S,SPACE/S

`Funktion:`

Stellt den Paritätsmodus der seriellen Schnittstelle ein.

`Beschreibung:`
Stellt den Paritätsmodus der seriellen Schnittstelle ein.

`Ergebnis:`
-

`Warnung:`
-

`Beispiel:`

```
/* Paritätsmodus ändern. */

PARITY NONE
```

4.26 Der PASTECLIP Befehl

`Format:`
PASTECLIP [Unit <Nummer>]

`Befehlsmuster:`
UNIT/K/N

`Funktion:`
Fügt den Inhalt des Klemmbretts in den Eingabestrom ein.

`Beschreibung:`
Fügt den Inhalt des Klemmbretts in den Eingabestrom ein. Liest den Inhalt der gewünschten Klemmbretteinheit oder der in der Programmkonfiguration angegebenen Einheit aus.

`Ergebnis:`
-

`Warnung:`
Falls das Klemmbrett keinen Text enthält.

`Beispiel:`

```
/* Setze den Inhalt von Klemmbrett #2 ein. */

PASTECLIP UNIT 2
```

4.27 Der PRINT Befehl

`Format:`
PRINT [From] <Screentext | Clipboard | Buffer | Dial | Wait | Upload | Download> [TO <Dateiname>] [Serial | Modem | Screen | Terminal | User | Comment | Size | Date | Attr]

`Befehlsmuster:`

FROM/A,TO/K,SERIAL/S,MODEM/S,SCREEN/S,TERMINAL/S,USER/S, COMMENT/S,SIZE/S,DATE/S,ATTR/S

`Funktion:`

Druckt den Inhalt des Bilschirms, des Klemmbretts, des Textpuffers oder einer der Listen aus.

`Beschreibung:`

Gibt den Inhalt des Bilschirms, des Klemmbretts, des Textpuffers oder einer der Listen (siehe Befehl Abschnitt 4.2 [ADDITEM], Seite 11) in einer Datei oder auf dem Drucker aus. Wir der `To` Parameter nicht angegeben, erfolgt die Ausgabe auf dem Drucker. Die Parameter `Serial` bis `Attr` bestimmen, welche Informationen gedruckt werden:

`Screentext, Clipboard, Buffer, Wait list`
: Die Optionen haben keinen Einfluß auf den Ausdruck.

`Dial list`
: Reagiert auf die `Serial`, `Modem`, `Screen`, `Terminal`, `User` und `Comment` Parameter. Der Ausdruck enthält die zu den angegebenen Einstellungen gehörenden Informationen.

`Upload list, Download list`
: Reagiert auf die `Comment`, `Size`, `Date` und `Attr` Parameter. Der Ausdruck enthält die zu den Dateiattributen gehörenden Informationen. *Hinweis: wird auch nur einer dieser Parameter angeben, so werden die Dateinamen ohne die dazugehörenden Pfadnamen ausgegeben.*

`Ergebnis:`

-

`Warnung:`

Falls der Anwender den Ausdruck abbricht.

`Beispiel:`

```
/* Löschen der Dial Liste, unterbringen des
 * gesamten Telefonbuchs.
 */

CLEAR dial
ADDITEM TO dial PHONE #?

/* Ausgabe der gesamten Dial Liste in einer Datei. */

PRINT FROM dial TO 'ram:Telefonbuch' SERIAL MODEM SCREEN...
...TERMINAL USER COMMENT
```

4.28 Der PROTOCOL Befehl

`Format:`

PROTOCOL [None|RTSCTS|RTSCTSDTR]

`Befehlsmuster:`

NONE/S,RTSCTS/S,RTSCTSDTR/S

`Funktion:`

Stellt das Handshaking-Protokoll der seriellen Schnittstelle ein.

`Beschreibung:`

Stellt das Handshaking-Protokoll der seriellen Schnittstelle ein.

`Ergebnis:`

-

`Warnung:`

-

`Beispiel:`

```
/* Ändere das Handshaking-Protokoll. */

PROTOCOL NONE
```

4.29 Der PUTCLIP Befehl

`Format:`

PUTCLIP [Unit <Nummer>] [TEXT] <Text>

`Befehlsmuster:`

UNIT/K/N,TEXT/A/F

`Funktion:`

Bringt Text im Klemmbrett unter.

`Beschreibung:`

Bringt den gegebenen Text im ausgewählten Klemmbrett unter. Wird keine Klemmbrettnummer angegeben, wählt das in der Programmkonfiguration angegebene Klemmbrett.

`Ergebnis:`

-

`Warnung:`

-

`Beispiel:`

```
/* Legt einen kurzen Text im Klemmbrett ab.
 * Vorsicht: kann maximal 65.536 Zeichen
 * lang sein.
 */

PUTCLIP 'Hello sailor!'
```

4.30 Der QUIT Befehl

`Format:`

QUIT [Force]

`Befehlsmuster:`

FORCE/S

`Funktion:`

Beendet das Programm.

`Beschreibung:`

Beendet das Programm, fragt nach einer Bestätigung, sofern der `Force` Parameter nicht angegeben wird.

`Ergebnis:`

-

`Warnung:`

Falls der Anwender keine Bestätigung gegeben hat.

`Beispiel:`

```
/* Beende das Programm, frage nach Bestätigung. */

QUIT

/* Falls keine Bestätigung gegeben, beende sofort. */

IF rc ~= 0 THEN QUIT FORCE
```

4.31 Der READ Befehl

`Format:`

READ [Num <Anzahl Zeichen>] [CR] [Noecho] [Verbatim] [[Prompt] <Text>]

`Befehlsmuster:`

NUM/K/N,CR/S,NOECHO/S,VERBATIM/S,PROMPT/K/F

`Funktion:`

Liest eine Anzahl Zeichen von der seriellen Schnittstelle.

`Beschreibung:`

Ist der `Num` Parameter angegeben, wird versucht eine Reihe Zeichen von der seriellen Schnittstelle zu lesen (Vorsicht: es können nur maximal 65.536 Zeichen gelesen werden). Der Befehl ist beendet, sobald genügend Zeichen gelesen, der Befehl abgebrochen oder die maximal auf Eingehen der Zeichen zu wartende Zeit (einstellbar über den Befehl Abschnitt 4.54 [TIMEOUT], Seite 45) gewartet wurde.

Wird der `CR` Parameter verwendet, können einfache Bearbeitungsfunktionen (`Rückschritt`, `Control-X`) zur Eingabe einer Zeile Text verwendet werden. Die

Eingabe ist beendet, sobald die `Zeilenrücklauf`-Taste gedrückt, der Befehl abgebrochen oder die Wartezeit überschritten wurde.

Der `Noecho` Parameter verhindert, daß getippte Zeichen zur Gegenseite geschickt werden. *Hinweis: soll eine bestimmte Anzahl Zeichen gelesen werden, so werden die gelesenen Zeichen grundsätzlich nicht zur Gegenseite geschickt.*

Wird der `Prompt`-Parameter verwendet, so wird der übergebene Text wie mit dem Befehl Abschnitt 4.47 [SEND], Seite 41 verschickt.

Üblicherweise beachtet dieser Befehl die aktuelle Konvertierungstabelle für eingehende Zeichen. Sollen die Zeichen ohne jegliche Veränderungen gelesen werden, muß der `Verbatim` Parameter angegeben werden.

`Ergebnis:`

Die gelesenen Zeichen.

`Warnung:`

Sofern der Befehl abgebrochen wurde, keine Eingabe erfolgte oder die Wartezeit abgelaufen ist.

`Beispiel:`

```
/* Befehlsergebnisse zugänglich machen. */

OPTIONS RESULTS

/* Wartezeit auf fünf Sekunden setzen. */

TIMEOUT 5

/* Sieben Zeichen lesen. */

READ NUM 7

/* Ergebnis ausgeben. */

IF rc ~= 0 THEN
   SAY 'Es wurden keine Zeichen gelesen'
ELSE
   SAY result

/* Wartezeit abschalten. */

TIMEOUT OFF

/* Eingabe anfordern. */

READ CR PROMPT 'Bitte Text eingeben:'

/* Ergebnis ausgeben. */

IF rc ~= 0 THEN
   SAY 'Es wurden keine Eingaben gemacht'
ELSE
   SAY result
```

4.32 Der RECEIVEFILE Befehl

`Format:`

RECEIVEFILE [Mode <ASCII|Text|Binary>] [Name <Dateiname>]

`Befehlsmuster:`

MODE/K,NAME/K

`Funktion:`

Empfängt Dateien über das XPR-Protokoll.

`Beschreibung:`

Empfängt eine oder mehrere Datei(en) über das eingestellte XPR-Protokoll. Der `Mode` Parameter bestimmt den Übertragungsmodus (normaler ASCII-Text, Textmodus oder Binärmodus). Verschiedene Transferprotokolle benötigen keinen Dateinamen, um Daten zu empfangen, wird jedoch ein Name benötigt und wurde keiner Angegeben, so wird ein Name abgefragt.

Die Namen aller empfangenen Dateien werden in der Download Liste abgelegt, die vor Start des Empfangs gelöscht wird.

`Ergebnis:`

-

`Warnung:`

Sofern ein Dateiname angefordert wurde, der Anwender aber keinen ausgewählt hat.

`Beispiel:`

```
/* Empfange eine Datei im Textmodus. */

RECEIVEFILE MODE text
```

4.33 Der REDIAL Befehl

`Format:`

REDIAL

`Befehlsmuster:`

,

`Funktion:`

Wählt den Inhalt der Wählliste von neuem.

`Beschreibung:`

Wählt die in der Wählliste verbliebenen Einträge von neuem an. Dieser Befehl ist beendet, sobald die Wählfunktion gestartet wurde.

`Ergebnis:`

-

`Warnung:`

Falls Wählliste leer ist.

`Beispiel:`

```
/* Wähle erneut an. */

REDIAL

/* Erfolgreich? */

IF rc ~= 0 THEN SAY 'Wählliste ist leer'
```

4.34 Der REMITEM Befehl

`Format:`

REMITEM [From] <Upload|Download|Dial|Wait> [Name <Name oder Muster>]

`Befehlsmuster:`

FROM/A,NAME/K/F

`Funktion:`

Entfernt einen oder mehrere Einträge aus einer Liste.

`Beschreibung:`

Entfernt einen oder mehrere Einträge aus einer Liste. Ist kein `Name` Parameter angegeben, entfernt den gerade angewählten Listeneintrag (siehe Befehl Abschnitt 4.46 [SELECTITEM], Seite 40). Der `Name` Parameter kann ein gültiger Name oder ein Namensmuster sein.

Hinweis: Entfernt keine benannten Einträge aus der Dial Liste.

`Ergebnis:`

-

`Warnung:`

Falls kein passender Listeneintrag gefunden werden konnte.

`Beispiel:`

```
/* Entferne den gerade angewählten Eintrag aus
 * der Wait Liste.
 */

REMITEM FROM wait

/* Entferne alle Einträge aus der Wait Liste, die
 * mit einem 'z' enden.
 */

REMITEM FROM wait NAME '#?z'
```

4.35 Der REQUESTFILE Befehl

`Format:`

REQUESTFILE [Title <Titeltext>] [Path <Pfadname>] [File <Dateiname>] [Pattern <Muster>] [Multi] [Stem|Name <Variable name>]

`Befehlsmuster:`

TITLE/K,PATH/K,FILE/K,PATTERN/K,MULTI/S,STEM=NAME/K

`Funktion:`

Fordert einen oder mehrere Dateinamen vom Anwender an.

`Beschreibung:`

Fordert einen oder mehrere Dateinamen vom Anwender an. Öffnet ein Dateiauswahlfeld mit angegebenen Titel, Pfad- und Dateiname und Namensmuster. Ist nur ein einzelner Name anzufordern, wird er in der Variable `result` zurückgeliefert. Bei mehreren Namen (über den `Multi` Parameter zu aktivieren) werden diese in der über den `Stem` Parameter anzugebenen Stammvariable abgelegt.

`Ergebnis:`

Der Name der Datei wird in der Variable `result` abgelegt. Wurden mehrere Dateien ausgewählt, werden folgende Informationen in der angegebenen Stammvariable abgelegt:

`<Variablenname>.COUNT`
: Die Anzahl der ausgewählten Dateien.

`<Variablename>.0 bis <Variablenname>.n-1`
: Die Dateinamen.

`Warnung:`

Falls keine Datei ausgewählt wurde.

`Beispiel:`

```
/* Befehlsergebnisse zugänglich machen. */

OPTIONS RESULTS

/* Einen einzelnen Dateinamen ermitteln. */

REQUESTFILE TITLE '"Eine Datei auswählen"'

/* Ergebnis ausgeben. */

IF rc ~= 0 THEN
   SAY 'Es wurde keine Datei ausgewählt'
ELSE
   SAY result

/* Mehrere Dateien auswählen. */

REQUESTFILE TITLE 'Mehrere Dateien auswählen' MULTI STEM names

/* Ergebnis auswählen. */

IF rc ~= 0 THEN
   SAY 'Es wurden keine Dateien ermittelt'
```

```
ELSE DO
   SAY 'Dateien ausgewählt:' names.count

   DO i = 0 TO names.count - 1
      SAY names.i
   END
END
```

4.36 Der REQUESTNOTIFY Befehl

`Format:`

REQUESTNOTIFY [Title <Titeltext>] [Prompt] <Hinweistext>

`Befehlsmuster:`

TITLE/K,PROMPT/A/F

`Funktion:`

Gibt einen Hinweis an den Anwender aus

`Beschreibung:`

Gibt einen Hinweis an den Anwender aus. Der Hinweistext kann Zeichen zur Zeilenschaltung enthalten (`'0A'X`), der Anwender kann das zu öffnende Fenster durch Anklicken des `Continue` schließen.

`Ergebnis:`

-

`Warnung:`

-

`Beispiel:`

```
/* Hinweis ausgeben. */

REQUESTNOTIFY TITLE '"Wichtiger Hinweis"' ...
...PROMPT 'Dieser Hinweis ist wichtig'
```

4.37 Der REQUESTNUMBER Befehl

`Format:`

REQUESTNUMBER [Default <Vorgabe>] [Prompt <Hinweistext>]

`Befehlsmuster:`

DEFAULT/K/N,PROMPT/K/F

`Funktion:`

Fordert einen Zahlenwert vom Anwender an

`Beschreibung:`

Fordert einen Zahlenwert vom Anwender an, zeigt den Hinweistext an und gibt den Standardwert vor.

`Ergebnis:`

Die eingebene Zahl.

`Warnung:`

Falls keine Zahl eingegeben wurde.

`Beispiel:`

```
/* Befehlsergebnisse zugänglich machen. */

OPTIONS RESULTS

/* Eine einzelne Zahl abrufen. */

REQUESTNUMBER DEFAULT 42 PROMPT 'Antwort eingeben'

/* Ergebnis ausgeben. */

IF rc ~= 0 THEN
   SAY 'Es wurde keine Zahl eingegeben'
ELSE
   SAY result
```

4.38 Der REQUESTRESPONSE Befehl

`Format:`

REQUESTRESPONSE [Title <Titeltext>] [Options <Optionen>] [Prompt] <Hinweistext>

`Befehlsmuster:`

TITLE/K,OPTIONS/K,PROMPT/A/F

`Funktion:`

Fordert eine Entscheidung vom Anwender

`Beschreibung:`

Fordert eine Entscheidung vom Anwender, zeigt den gegeben Hinweistext an und gibt die angegebenen Optionen als anklickbare Knöpfe vor. Werden keine Optionen angegeben, so wird `Yes|No` verwendet.

`Ergebnis:`

Werden die Optionen `Ja|Vielleicht|Nein` benutzt, so wird 1 für `Ja`, 2 für `Vielleicht` und eine Warnung für `No` zurückgeliefert.

`Warnung:`

Falls die negative Wahl getroffen wurde.

`Beispiel:`

```
/* Befehlsergebnisse zugänglich machen. */

OPTIONS RESULTS

/* Eine Entscheidung fordern. */
```

```
REQUESTRESPONSE PROMPT 'Sind Sie unentschieden?' ...
...OPTIONS 'Ja|Vielleicht|Nein'

/* Wie sieht das Ergebnis aus? */

IF rc ~= 0 THEN
   SAY 'Nicht unentschieden'
ELSE DO
   IF result = 0 THEN
      SAY 'Unentschieden'
   ELSE
      SAY 'Wahrscheinlich unentschieden'
END
```

4.39 Der REQUESTSTRING Befehl

`Format:`

REQUESTSTRING [Secret] [Default <Vorgabe>] [Prompt <Hinweistext>]

Citem Befehlsmuster:

SECRET/S,DEFAULT/K,PROMPT/K/F

`Funktion:`

Fordert einen Text vom Anwender an

`Beschreibung:`

Fordert einen Text vom Anwender an, verwendet gegeben Hinweistext und Vorgabewert. Wird der `Secret` Parameter verwendet, werden die getippten Zeichen nicht angezeigt.

`Ergebnis:`

Der eingegebene Text.

`Warnung:`

Falls kein Text eingegeben wird.

`Beispiel:`

```
/* Befehlsergebnisse zugänglich machen. */

OPTIONS RESULTS

/* Eine geheime Eingabe anfordern. */

REQUESTSTRING SECRET DEFAULT '"hello sailor!"' ...
...PROMPT 'Geheimen Text eingeben'

/* Ergebnis ausgeben. */

IF rc ~= 0 THEN
   SAY 'Kein Text wurde eingegeben'
ELSE
   SAY result
```

4.40 Der RESETSCREEN Befehl

`Format:`

RESETSCREEN

`Befehlsmuster:`

,

`Funktion:`

Setzt den Terminalbildschirm zurück.

`Beschreibung:`

Setzt den Terminalbildschirm auf Standardwerte zurück, dies schließt das Bildschirmlöschen und das Zurücksetzen von Text und Textattributen und Farben ein.

`Ergebnis:`

-

`Warnung:`

-

`Beispiel:`

```
/* Zurücksetzen des Terminalbildschirms. */

RESETSCREEN
```

4.41 Der RESETSTYLES Befehl

`Format:`

RESETSTYLES

`Befehlsmuster:`

,

`Funktion:`

Setzt den Terminal-Text-Modus zurück

`Beschreibung:`

Setzt den Terminal-Text-Modus zurück, einschließlich Fettdruck, inversem Text, Kursivschrift, usw.

`Ergebnis:`

-

`Warnung:`

-

`Beispiel:`

```
/* Terminal-Textmodus zurücksetzen. */

RESETSTYLES
```

4.42 Der RESETTEXT Befehl

`Format:`

RESETTEXT

`Befehlsmuster:`

,

`Funktion:`

Setzt den Terminal-Text zurück

`Beschreibung:`

Setzt den Terminal-Text zurück, einschließlich Graphik- oder G1 Modus.

`Ergebnis:`

-

`Warnung:`

-

`Beispiel:`

```
/* Terminaltext zurücksetzen. */

RESETTEXT
```

4.43 Der RX Befehl

`Format:`

RX [Console] [Async] [Command] <Programmname>

`Befehlsmuster:`

CONSOLE/S,ASYNC/S,COMMAND/A/F

`Funktion:`

Führt ein ARexx-Programm aus.

`Beschreibung:`

Führt ein ARexx-Programm aus, wird der `Console` Parameter angegeben, wird ein Ausgabefenster geöffnet. Wir der `Async` Parameter angegeben, so wird der Befehl nach Starten des Programms beendet.

`Ergebnis:`

-

`Warnung:`

-

`Beispiel:`

```
/* Starten der ‘term’ Befehlseingabe. */

RX CONSOLE ASYNC ’term:cmdshell.term’
```

4.44 Der SAVE Befehl

`Format:`

SAVE [From] <Translations|Functionkeys| Cursorkeys|Fastmacros|Hotkeys|Speech| Sound|Buffer|Configuration|Phone| Screentext|Screenimage>

`Befehlsmuster:`

FROM/A

`Funktion:`

Speichert Daten in einer Datei

`Beschreibung:`

Speichert Daten in einer anzugebenden Datei. Mehr zu diesem Befehl ist beim Befehl Abschnitt 4.45 [SAVEAS], Seite 40 zu finden.

`Ergebnis:`

-

`Warnung:`

Falls keine Datei ausgewählt wurde.

`Beispiel:`

```
/* Abspeichern des Bildschirms als IFF-ILBM Bild. */

SAVE FROM screenimage
```

4.45 Der SAVEAS Befehl

`Format:`

SAVEAS [Name <Dateiname>] [From] <Translations|Functionkeys|Cursorkeys| Fastmacros|Hotkeys|Speech|Sound|Buffer| Configuration|Phone|Screentext| Screenimage>

`Befehlsmuster:`

NAME/K,FROM/A

`Funktion:`

Speichert Daten in einer Datei.

`Beschreibung:`

Speichert Daten in einer Datei, fragt nach einem Dateinamen, falls keiner angegeben wurde. Speichert entweder Programmeinstellungen, den Inhalt des Telefonbuchs (`Phonebook` Parameter), den Inhalt des Terminalbildschirms als ASCII Text (`Screentext` Parameter) oder den Inhalt des Terminalbildschirms als IFF-ILBM-Bild (`Screenimage` Parameter).

`Ergebnis:`

-

`Warnung:`

Falls kein Dateiname angegeben wurde.

`Beispiel:`

```
/* Speichern der Programmeinstellungen. */

SAVEAS NAME 'ram:term.prefs' FROM configuration
```

4.46 Der SELECTITEM Befehl

`Format:`

SELECTITEM [Name <Name>] [From] <Upload|Download|Dial|Wait> [Next|Prev|Previous|Top

`Befehlsmuster:`

NAME/K,FROM/A,NEXT/S,PREV=PREVIOUS/S,TOP/S,BOTTOM/S

`Funktion:`

Wählt einen Eintrag aus einer Liste aus

`Beschreibung:`

Wählt einen Eintrag aus einer Liste aus, liefert den Namen des Eintrags in der `result` Variable zurück. Der `Top` Parameter wählt den ersten Listeneintrag, `Bottom` den letzten, `Next` den nächstfolgenden und `Previous` den vorhergehenden Eintrag.

Anstelle eines Positionsparameters kann auch ein Name, bzw. ein Namensmuster angegeben werden. Der erste passende Listeneintrag wird ausgewählt.

Hinweis: Kann nicht mit der Dial Liste verwendet werden.

`Ergebnis:`

Liefert den Namen des ausgewählten Eintrags in der Variable `result`.

`Warnung:`

Falls das Ende der Liste erreicht ist.

`Beispiel:`

```
/* Befehlsergebnisse zugänglich machen. */

OPTIONS RESULTS

/* Inhalt der Download Liste ausgeben. */

SELECTITEM FROM download TOP

DO WHILE rc = 0
   SAY result
   SELECTITEM FROM download NEXT
END
```

4.47 Der SEND Befehl

`Format:`

SEND [Noecho] [Local] [Byte <ASCII-Code>] [Text] <Text>

`Befehlsmuster:`

NOECHO/S,LOCAL/S,BYTE/K/N,TEXT/A/F

`Funktion:`

Verschickt den gegebenen Text über die serielle Schnittstelle, führt enthaltene Befehlssequenzen aus.

`Beschreibung:`

Verschickt den gegebenen Text über die serielle Schnittstelle, führt enthaltene Befehlssequenzen aus (siehe Dokumentation des Hauptprogrammes). Um ein einzelnes Zeichen zu verschicken, ist der `Byte` Parameter zu verwenden. Wird der `Noecho` Parameter verwendet, wird die Terminalausgabe unterdrückt. Der `Local` Parameter bewirkt, dass der Text nur im Terminal ausgegeben, aber nicht über die serielle Schnittstelle verschickt wird.

`Ergebnis:`

-

`Warnung:`

-

`Beispiel:`

```
/* Etwas Text ausgeben. */

SEND 'Etwas Text.\r\n'

/* Ein einzelnes Zeichen (eine Null) ausgeben. */

SEND BYTE 0

/* Einen Befehl ausführen (Break-Signal verschicken). */

SEND '\x'
```

4.48 Der SENDBREAK Befehl

`Format:`

SENDBREAK

`Befehlsmuster:`

,

`Funktion:`

Verschickt ein Unterbrechungssignal über die serielle Schnittstelle

`Beschreibung:`

Verschickt ein Unterbrechungssignal über die serielle Schnittstelle

`Ergebnis:`

-

`Warnung:`

-

`Beispiel:`

```
/* Unterbrechungssignal verschicken. */

SENDBREAK
```

4.49 Der SENDFILE Befehl

`Format:`

SENDFILE [Mode <ASCII|Text|Binary>] [Names] {Dateinamen}

`Befehlsmuster:`

MODE/K,NAMES/M

`Funktion:`

Verschickt Dateien über das XPR-Protokoll

`Beschreibung:`

Verschickt eine oder mehrere Datei(en) über das eingestellte XPR-Protokoll. Der `Mode` Parameter bestimmt den Übertragungsmodus (normaler ASCII-Text, Textmodus oder Binärmodus). Einige Übertragungsprotokolle haben eigene Mittel und Zwecke, die zu verschickenden Dateien zu bestimmen. Sind keine Dateinamen angegeben und ist die Upload Liste leer, wird das Programm nachfragen.

Erfolgreich aus der Upload Liste verschickte Dateien werden aus der Liste entfernt, nur die Dateinamen verbleiben in der Liste, die nicht übertragen werden konnten.

Zu verschickende Dateien, deren Name keinen kompletten Pfadnamen enthält, werden immer in der Schublade gesucht, die in den Pfadeinstellungen für Binär-/Text- und ASCII-Upload angegeben ist.

`Ergebnis:`

-

`Warnung:`

Falls keine Dateinamen ausgewählt wurden.

`Beispiel:`

```
/* Verschicke Dateien. */

SENDFILE

/* Verschicke eine einzelne Datei. */

SENDFILE 'c:list'

/* Lösche die Upload Liste, füge eine
 * einzige Datei hinzu.
 */
```

```
CLEAR upload
ADDITEM TO upload NAME 'c:dir'

/* Verschicke die Datei. */

SENDFILE
```

4.50 Der SETATTR Befehl

`Format:`

SETATTR [Object] <Name> [Field] <Name> [Stem <Name>] [Var <Name>]

`Befehlsmuster:`

OBJECT/A,FIELD,STEM/K,VAR

`Funktion:`

Stellt ein Attribut des Programms ein.

`Beschreibung:`

Stellt ein Attribut des Programms ein, liest die benötigten Informationen aus der angegebenen Stamm- oder einfachen Variable.

Eine Liste der gültigen Attribute ist unter Abschnitt 5.23 [Attribute], Seite 63 zu finden.

`Ergebnis:`

-

`Warnung:`

-

`Beispiel:`

```
/* Einstellen der Baudrate. */

SETATTR serialprefs baudrate 2400
```

4.51 Der SPEAK Befehl

`Format:`

SPEAK [Text] <Text>

`Befehlsmuster:`

TEXT/A/F

`Funktion:`

Spricht den gegebenen Text über den Amiga-Sprachsynthesizer.

`Beschreibung:`

Spricht den gegebenen Text über den Amiga-Sprachsynthesizer, die Sprachfunktion muß hierfür aktiviert sein.

Ergebnis:

-

Warnung:

-

Beispiel:

```
/* Beispieltext aussprechen. */

SPEAK 'This is Amiga speaking.'
```

4.52 Der STOPBITS Befehl

Format:

STOPBITS [0|1]

Befehlsmuster:

0/S,1/S

Funktion:

Ótellt die Anzahl Stop-Bits der seriellen Schnittstelle ein.

Beschreibung:

Stellt die Anzahl Stop-Bits der seriellen Schnittstelle ein.

Ergebnis:

-

Warnung:

-

Beispiel:

```
/* Anzahl Stop-Bits einstellen. */

STOPBITS 1
```

4.53 Der TEXTBUFFER Befehl

Format:

TEXTBUFFER [Lock|Unlock]

Befehlsmuster:

LOCK/S,UNLOCK/S

Funktion:

Ver- oder entriegelt den Textpuffer.

Beschreibung:

Ver- oder entriegelt den Textpuffer.

Ergebnis:

-

Warnung:

-

Beispiel:

```
/* Verriegele den Textpuffer. */

TEXTBUFFER LOCK
```

4.54 Der TIMEOUT Befehl

Format:

TIMEOUT [[Sec|Seconds] <Number>] [Off]

Befehlsmuster:

SEC=SECONDS/N,OFF/S

Funktion:

Stellt die Befehls-Wartezeit ein

Beschreibung:

Stellt die Befehls-Wartezeit ein, die die Befehle Abschnitt 4.55 [WAIT], Seite 46 und Abschnitt 4.31 [READ], Seite 30 verwenden.

Ergebnis:

-

Warnung:

-

Beispiel:

```
/* Wartezeit einstellen. */

TIMEOUT SEC 5
```

4.55 Der WAIT Befehl

Format:

WAIT [Noecho] [[Text] <Text>]

Befehlsmuster:

NOECHO/S,TEXT/F

Funktion:

Wartet auf das Eingehen eines bestimmten Textes.

`Beschreibung:`

Wartet auf das Eingehen eines bestimmten Textes. Wird kein Wartetext mit dem `TEXT` Parameter angegeben, wird der Text aus der Wait Liste entnommen. Der `Noecho` Parameter unterdrückt Terminal-Ausgabe. *Hinweis: Groß- und Kleinschreibung haben keinen Einfluß auf den Vergleich der Texte.*

`Ergebnis:`

Liefert den gefundenen Text zurück.

`Warnung:`

Falls die Wartezeit überschritten oder der Befehl abgebrochen wurde.

`Beispiel:`

```
/* Befehlsergebnisse zugänglich machen. */

OPTIONS RESULTS

/* Wartezeit einstellen. */

TIMEOUT SEC 30

/* Warte auf etwas Text. */

WAIT 'etwas Text'

/* Lösche die Wait Liste, trage zwei
 * Worte ein.
 */

CLEAR wait
ADDITEM TO wait NAME 'dings'
ADDITEM TO wait NAME 'da'

/* Warte auf das Eingehen der Texte. */

WAIT

/* Ausgabe des Ergebnisses. */

IF rc ~= 0 THEN
   SAY 'Kein Text wurde erkannt'
ELSE
   SAY result
```

4.56 Der WINDOW Befehl

`Format:`

WINDOW [Names] {<Buffer|Review|Packet|Fastmacros| Status|Main>} [Open|Close] [Activate] [Min|Max] [Front|Back] [Top|Bottom|Up|Down]

`Befehlsmuster:`

NAMES/A/M,OPEN/S,CLOSE/S,ACTIVATE/S,MIN/S,MAX/S,FRONT/S,BACK/S, TOP/S,BOTTOM/S,UP/S,DOWN/S

`Funktion:`

Verändert die Eigenschaften eines Fensters.

`Beschreibung:`

Verändert die Eigenschaften eines Fensters. Nicht alle Fenster unterstützen alle Befehle. Unterstützte Fenster sind:

`Buffer`

Der Textpuffer-Bildschirm, unterstützt die Befehle `Open`, `Close`, `Activate` und `Front`.

`Review`

Das Ausgabepuffer-Fenster, unterstützt die Befehle `Open`, `Close`, `Activate`, `Min`, `Max`, `Front`, `Back`, `Top`, `Bottom`, `Up`, und `Down`.

`Packet`

Die Zeileneingabe, unterstützt die Befehle `Open`, `Close`, `Activate`, `Min`, `Max`, `Front` und `Back`.

`Fastmacros`

Das Fast! Makro Fenster, unterstützt die Befehle `Open`, `Close`, `Activate`, `Min`, `Max`, `Front` und `Back`.

`Status`

Das Statusfenster, unterstützt die Befehle `Open`, `Close`, `Activate`, `Front` und `Back`.

`Main`

Das Hauptfenster, unterstützt die Befehle `Open`, `Close`, `Activate`, `Front` und `Back`.

`Ergebnis:`

-

`Warnung:`

-

`Beispiel:`

```
/* Öffne alle Fenster. */

WINDOW buffer review packet fastmacros status main OPEN
```

5 Attribute

Viele der Programminternen Variablen können durch die Befehle Abschnitt 4.18 [GETATTR], Seite 21 und Abschnitt 4.50 [SETATTR], Seite 43 gelesen und verändert werden. Die verfügbaren Informationen (Attribute) können, wie folgt beschrieben ist, abgerufen werden:

`Zahlenwert`

`<Stammobjekt>.<Feld>`
Zahlenwert

Die Information ist ein Zahlenwert.

`Text`

`<Stammobjekt>.<Feld>`
Text

The Information ist ein Text.

`Wahrheitswert`

`<Stammobjekt>.<Feld>`
Wahrheitswert

The Information ist ein Wahrheitswert und kann entweder `ON` (wahr, aktiv, eingeschaltet) oder `OFF` (falsch, inaktiv, ausgeschaltet) sein.

`Name`

`<Stammobjekt>.<Feld>`
<Wert 1> ... <Wert n>

Die Information ist einer der angegebenen Werte.

5.1 Das TERM Objekt (Nicht veränderbar)

`TERM.VERSION`
Text

Die Version des 'term' Programms.

`TERM.SCREEN`
Text

Der Name des öffentlichen Bildschirms, auf dem sich das 'term' Hauptfenster befindet.

`TERM.SESSION.ONLINE`
Wahrheitswert

Ob gerade eine Verbindung besteht oder nicht.

`TERM.SESSION.SESSIONSTART`
: Text

  Zeit und Datum zu dem das 'term' Programm gestartet wurde.

`TERM.SESSION.BYTESSENT`
: Zahlenwert

`TERM.SESSION.BYTESRECEIVED`
: Zahlenwert

`TERM.SESSION.CONNECTMESSAGE`
: Text

  Der vom Modem beim Verbindungsaufbau ausgegebene Text.

`TERM.SESSION.BBSNAME`
: Text

`TERM.SESSION.BBSNUMBER`
: Text

`TERM.SESSION.BBSCOMMENT`
: Text

`TERM.SESSION.USERNAME`
: Text

`TERM.SESSION.ONLINEMINUTES`
: Zahlenwert

  Die Minuten, die das Programm bereits mit einer Mailbox verbunden ist.

`TERM.SESSION.ONLINECOST`
: Zahlenwert

  Die Kosten der aktuellen Verbindung.

`TERM.AREXX`
: Text

  Der Name des Rexx-Wirts, mit dem das ARexx-Programm kommuniziert.

`TERM.LASTERROR`
: Zahlenwert

  Der Fehlercode, den der letzte Befehl hinterlassen hat.

`TERM.TERMINAL.ROWS`
: Zahlenwert

  Die Anzahl der Textzeilen des Terminalbildschirm.

`TERM.TERMINAL.COLUMNS`
: Zahlenwert

  Die Anzahl der Textspalten des Terminalbildschirm.

`TERM.BUFFER.SIZE`
: Zahlenwert

  Der Umfang des Textpuffers.

5.2 Das PHONEBOOK Objekt (Nicht veränderbar)

Verfügbare Felder sind:

`PHONEBOOK.COUNT`
: Zahlenwert

  Die Anzahl der verfügbaren Telefonbucheinträge. Die einzelnen Einträge sind unter `PHONEBOOK.0...` bis `PHONEBOOK.n-1...` zugänglich.

`PHONEBOOK.n.NAME`
: Text

`PHONEBOOK.n.NUMBER`
: Text

`PHONEBOOK.n.COMMENTTEXT`
: Text

`PHONEBOOK.n.USERNAME`
: Text

`PHONEBOOK.n.PASSWORDTEXT`
: Text

5.3 Das SERIALPREFS Objekt

Verfügbare Felder sind:

`SERIALPREFS.BAUDRATE`
: Zahlenwert

`SERIALPREFS.BREAKLENGTH`
: Zahlenwert

  Die Länge des Unterbrechungssignals ins Mikrosekunden.

`SERIALPREFS.BUFFERSIZE`
: Zahlenwert

`SERIALPREFS.DEVICENAME`
: Text

`SERIALPREFS.UNIT`
: Zahlenwert

`SERIALPREFS.BITSPERCHAR`
: Zahlenwert

  Anzahl Bits pro Zeichen, entweder sieben oder acht.

`SERIALPREFS.PARITYMODE`
: `NONE EVEN ODD MARK SPACE`.

`SERIALPREFS.STOPBITS`
: Zahlenwert

  Anzahl Stopbits, entweder 0 oder 1.

`SERIALPREFS.HANDSHAKINGMODE`
: `NONE RTSCTS RTSCTSDSR`

`SERIALPREFS.DUPLEXMODE`
: `HALF FULL`

`SERIALPREFS.XONXOFF`
: Wahrheitswert

`SERIALPREFS.HIGHSPEED`
: Wahrheitswert

`SERIALPREFS.SHARED`
: Wahrheitswert

`SERIALPREFS.STRIPBIT8`
: Wahrheitswert

`SERIALPREFS.CARRIERCHECK`
: Wahrheitswert

`SERIALPREFS.PASSXONXOFFTHROUGH`
: Wahrheitswert

`SERIALPREFS.QUANTUM`
: Zahlenwert

5.4 Das MODEMPREFS Objekt

Verfügbare Felder sind:

`MODEMPREFS.MODEMINITTEXT`
: Text

`MODEMPREFS.MODEMEXITTEXT`
: Text

`MODEMPREFS.MODEMHANGUPTEXT`
: Text

`MODEMPREFS.DIALPREFIXTEXT`
: Text

`MODEMPREFS.DIALSUFFIXTEXT`
: Text

`MODEMPREFS.NOCARRIERTEXT`
: Text

`MODEMPREFS.NODIALTONETEXT`
: Text

`MODEMPREFS.CONNECTTEXT`
: Text

`MODEMPREFS.VOICETEXT`
: Text

`MODEMPREFS.RINGTEXT`
: Text

`MODEMPREFS.BUSYTEXT`
: Text

`MODEMPREFS.OKTEXT`
: Text

`MODEMPREFS.ERRORTEXT`
: Text

`MODEMPREFS.REDIALDELAY`
: Zahlenwert

  Die Nachwahlpause in Sekunden.

`MODEMPREFS.DIALRETRIES`
: Zahlenwert

`MODEMPREFS.DIALTIMEOUT`
: Zahlenwert

  Die Anwahlpause in Sekunden.

`MODEMPREFS.CONNECTAUTOBAUD`
: Wahrheitswert

`MODEMPREFS.HANGUPDROPSDTR`
: Wahrheitswert

`MODEMPREFS.REDIALAFTERHANGUP`
: Wahrheitswert

`MODEMPREFS.NOCARRIERISBUSY`
: Wahrheitswert

`MODEMPREFS.CONNECTLIMIT`
: Zahlenwert

  Verbindungsdauer in Minuten.

`MODEMPREFS.CONNECTLIMITMACRO`
: Text

`MODEMPREFS.TIMETOCONNECT`
: Zahlenwert

5.5 Das SCREENPREFS Objekt

Verfügbare Felder sind:

`SCREENPREFS.COLOURMODE`
: `TWO FOUR EIGHT SIXTEEN`

`SCREENPREFS.FONTNAME`
: Text

`SCREENPREFS.FONTSIZE`
: Zahlenwert

`SCREENPREFS.MAKESCREENPUBLIC`
: Wahrheitswert

`SCREENPREFS.SHANGHAIWINDOWS`
: Wahrheitswert

`SCREENPREFS.BLINKING`
: Wahrheitswert

`SCREENPREFS.FASTERLAYOUT`
: Wahrheitswert

`SCREENPREFS.TITLEBAR`
: Wahrheitswert

`SCREENPREFS.STATUSLINEMODE`
: `DISABLED STANDARD COMPRESSED`

`SCREENPREFS.USEPUBSCREEN`
: Wahrheitswert

`SCREENPREFS.PUBSCREENNAME`
: Text

`SCREENPREFS.ONLINEDISPLAY`
: `TIME COST BOTH`

5.6 Das TERMINALPREFS Objekt

Verfügbare Felder sind:

`TERMINALPREFS.BELLMODE`
: `NONE VISIBLE AUDIBLE BOTH SYSTEM`

`TERMINALPREFS.ALERTMODE`
: `NONE BELL SCREEN BOTH`

`TERMINALPREFS.EMULATIONMODE`
: `INTERNAL ATOMIC TTY EXTERNAL HEX`

`TERMINALPREFS.FONTMODE`
: `STANDARD IBM IBMRAW`

`TERMINALPREFS.SENDCRMODE`
: `IGNORE CR CRLF`

`TERMINALPREFS.SENDLFMODE`
: `IGNORE LF LFCR`

`TERMINALPREFS.RECEIVECRMODE`
: `IGNORE CR CRLF`

`TERMINALPREFS.RECEIVELFMODE`
: `IGNORE LF LFCR`

`TERMINALPREFS.NUMCOLUMNS`
: Zahlenwert

`TERMINALPREFS.NUMLINES`
: Zahlenwert

`TERMINALPREFS.KEYMAPNAME`
: Text

`TERMINALPREFS.EMULATIONNAME`
: Text

`TERMINALPREFS.FONTNAME`
: Text

`TERMINALPREFS.FONTSIZE`
: Zahlenwert

5.7 Das EMULATIONPREFS Objekt

Verfügbare Felder sind:

`EMULATIONPREFS.CURSORMODE`
: `STANDARD APPLICATION`

`EMULATIONPREFS.NUMERICMODE`
: `STANDARD APPLICATION`

`EMULATIONPREFS.CURSORWRAP`
: Wahrheitswert

`EMULATIONPREFS.LINEWRAP`
: Wahrheitswert

`EMULATIONPREFS.INSERTMODE`
: Wahrheitswert

`EMULATIONPREFS.NEWLINEMODE`
: Wahrheitswert

`EMULATIONPREFS.FONTSCALEMODE`
: `NORMAL HALF`

`EMULATIONPREFS.SCROLLMODE`
: `JUMP SMOOTH`

`EMULATIONPREFS.DESTRUCTIVEBACKSPACE`
: Wahrheitswert

`EMULATIONPREFS.SWAPBSDELETE`
: Wahrheitswert

`EMULATIONPREFS.PRINTERENABLED`
: Wahrheitswert

`EMULATIONPREFS.ANSWERBACKTEXT`
: Text

`EMULATIONPREFS.CLSRESETSCURSOR`
: Wahrheitswert

5.8 Das CLIPBOARDPREFS Objekt

Verfügbare Felder sind:

`CLIPBOARDPREFS.UNIT`
: Zahlenwert

`CLIPBOARDPREFS.LINEDELAY`
: Zahlenwert

Einfüge-Verzögerung in 1/100 Sekunden.

`CLIPBOARDPREFS.CHARDELAY`
: Zahlenwert

Einfüge-Verzögerung in 1/100 Sekunden.

`CLIPBOARDPREFS.LINEPROMPTTEXT`
: Text

`CLIPBOARDPREFS.SENDTIMEOUT`
: Zahlenwert

Wartezeit in 1/100 Sekunden.

`CLIPBOARDPREFS.TEXTPACING`
: `DIRECT ECHO ANYECHO PROMPT DELAY KEYBOARD`

`CLIPBOARDPREFS.INSERTPREFIXTEXT`
: Text

`CLIPBOARDPREFS.INSERTSUFFIXTEXT`
: Text

5.9 Das CAPTUREPREFS Objekt

Verfügbare Felder sind:

`CAPTUREPREFS.LOGACTIONS`
: Wahrheitswert

`CAPTUREPREFS.LOGFILENAME`
: Text

`CAPTUREPREFS.LOGCALLS`
: Wahrheitswert

`CAPTUREPREFS.CALLLOGFILENAME`
: Text

`CAPTUREPREFS.MAXBUFFERSIZE`
: Zahlenwert

`CAPTUREPREFS.BUFFER`
: Wahrheitswert

`CAPTUREPREFS.BUFFERSAVEPATH`
: Text

`CAPTUREPREFS.CONNECTAUTOCAPTURE`
: Wahrheitswert

`CAPTUREPREFS.AUTOCAPTUREDATE`
: `NAME`, `INCLUDE`

`CAPTUREPREFS.CAPTUREFILTER`
: Wahrheitswert

`CAPTUREPREFS.CAPTUREPATH`
: Text

`CAPTUREPREFS.OPENBUFFERWINDOW`
: `TOP`, `END`

`CAPTUREPREFS.REMEMBERBUFFERWINDOW`
: Wahrheitswert

`CAPTUREPREFS.OPENBUFFERSCREEN`
: `TOP`, `END`

`CAPTUREPREFS.REMEMBERBUFFERSCREEN`
: Wahrheitswert

`CAPTUREPREFS.BUFFERSCREENPOSITION`
: `LEFT`, `MID`, `RIGHT`

`CAPTUREPREFS.BUFFERWIDTH`
: Zahlenwert

5.10 Das COMMANDPREFS Objekt

Verfügbare Felder sind:

`COMMANDPREFS.STARTUPMACROTEXT`
: Text

`COMMANDPREFS.LOGOFFMACROTEXT`
: Text

`COMMANDPREFS.UPLOADMACROTEXT`
: Text

`COMMANDPREFS.DOWNLOADMACROTEXT`
: Text

5.11 Das MISCPREFS Objekt

Verfügbare Felder sind:

`MISCPREFS.PRIORITY`
: Zahlenwert

`MISCPREFS.BACKUPCONFIG`
: Wahrheitswert

`MISCPREFS.OPENFASTMACROPANEL`
: Wahrheitswert

`MISCPREFS.RELEASEDEVICE`
: Wahrheitswert

`MISCPREFS.OVERRIDEPATH`
: Wahrheitswert

`MISCPREFS.AUTOUPLOAD`
: Wahrheitswert

`MISCPREFS.SETARCHIVEDBIT`
: Wahrheitswert

`MISCPREFS.COMMENTMODE`
: `IGNORE FILETYPE SOURCE`

`MISCPREFS.TRANSFERICONS`
: Wahrheitswert

`MISCPREFS.CREATEICONS`
: Wahrheitswert

`MISCPREFS.SIMPLEIO`
: Wahrheitswert

`MISCPREFS.TRANSFERPERFMETER`
: Wahrheitswert

5.12 Das PATHPREFS Objekt

Verfügbare Felder sind:

`PATHPREFS.ASCIIUPLOADPATH`
: Text

`PATHPREFS.ASCIIDOWNLOADPATH`
: Text

`PATHPREFS.TEXTUPLOADPATH`
: Text

`PATHPREFS.TEXTDOWNLOADPATH`
: Text

`PATHPREFS.BINARYUPLOADPATH`
: Text

`PATHPREFS.BINARYDOWNLOADPATH`
: Text

`PATHPREFS.CONFIGPATH`
: Text

`PATHPREFS.EDITORNAME`
: Text

`PATHPREFS.HELPFILENAME`
: Text

5.13 Das TRANSFERPREFS Objekt

Verfügbare Felder sind:

`TRANSFERPREFS.DEFAULTLIBRARY`
: Text

`TRANSFERPREFS.ASCIIUPLOADLIBRARY`
: Text

`TRANSFERPREFS.INTERNALASCIIUPLOAD`
: Wahrheitswert

`TRANSFERPREFS.ASCIIDOWNLOADLIBRARY`
: Text

`TRANSFERPREFS.INTERNALASCIIDOWNLOAD`
: Wahrheitswert

`TRANSFERPREFS.QUIETTRANSFER`
: Wahrheitswert

`TRANSFERPREFS.TEXTUPLOADLIBRARY`
: Text

`TRANSFERPREFS.TEXTDOWNLOADLIBRARY`
: Text

`TRANSFERPREFS.BINARYUPLOADLIBRARY`
: Text

`TRANSFERPREFS.BINARYDOWNLOADLIBRARY`
: Text

5.14 Das PROTOCOLPREFS Objekt

Dieses Objekt enthält keinerlei Felder, es enthält nur eine einzige Textzeile: die Optionen des XPR-Protokolls.

5.15 Das TRANSLATIONPREFS Objekt

Indizes bewegen sich zwischen 0 und 255 und beziehen sich auf die ASCII-Codes von Zeichen. Verfügbare Felder sind.

`TRANSLATIONPREFS.n.SEND`
: Text

`TRANSLATIONPREFS.n.RECEIVE`
: Text

5.16 Das FUNCTIONKEYPREFS Objekt

Indizes bewegen sich zwischen 1 und 10 und beziehen sich auf die Funktionstasten F1 bis F10. Verfügbare Felder sind:

`FUNCTIONKEYPREFS.n`
: Text

`FUNCTIONKEYPREFS.SHIFT.n`
: Text

`FUNCTIONKEYPREFS.ALT.n`
: Text

`FUNCTIONKEYPREFS.CONTROL.n`
: Text

5.17 Das CURSORKEYPREFS Objekt

Verfügbare Felder sind:

`CURSORKEYPREFS.UPTEXT`
: Text

`CURSORKEYPREFS.RIGHTTEXT`
: Text

`CURSORKEYPREFS.DOWNTEXT`
: Text

`CURSORKEYPREFS.LEFTTEXT`
: Text

`CURSORKEYPREFS.SHIFT.UPTEXT`
: Text

`CURSORKEYPREFS.SHIFT.RIGHTTEXT`
: Text

`CURSORKEYPREFS.SHIFT.DOWNTEXT`
: Text

`CURSORKEYPREFS.SHIFT.LEFTTEXT`
: Text

`CURSORKEYPREFS.ALT.UPTEXT`
: Text

`CURSORKEYPREFS.ALT.RIGHTTEXT`
: Text

`CURSORKEYPREFS.ALT.DOWNTEXT`
: Text

`CURSORKEYPREFS.ALT.LEFTTEXT`
: Text

`CURSORKEYPREFS.CONTROL.UPTEXT`
: Text

`CURSORKEYPREFS.CONTROL.RIGHTTEXT`
: Text

`CURSORKEYPREFS.CONTROL.DOWNTEXT`
: Text

`CURSORKEYPREFS.CONTROL.LEFTTEXT`
: Text

5.18 Das FASTMACROPREFS Objekt

`FASTMACROPREFS.COUNT`
: Zahlenwert

  Die Anzahl der verfügbaren Fast! Makros, die Einträge sind unter `FASTMACROPREFS.0...` bis `FASTMACROPREFS.n-1...` zugänglich.

`FASTMACROPREFS.n.NAME`
: Text

`FASTMACROPREFS.n.CODE`
: Text

5.19 Das HOTKEYPREFS Objekt

Verfügbare Felder sind:

`HOTKEYPREFS.TERMSCREENTOFRONTTEXT`
: Text

`HOTKEYPREFS.BUFFERSCREENTOFRONTTEXT`
: Text

`HOTKEYPREFS.SKIPDIALENTRYTEXT`
: Text

`HOTKEYPREFS.ABORTAREXX`
: Text

`HOTKEYPREFS.COMMODITYPRIORITY`
: Zahlenwert

`HOTKEYPREFS.HOTKEYSENABLED`
: Wahrheitswert

5.20 Das SPEECHPREFS Objekt

Verfügbare Felder sind:

`SPEECHPREFS.RATE`
: Zahlenwert

`SPEECHPREFS.PITCH`
: Zahlenwert

`SPEECHPREFS.FREQUENCY`
: Zahlenwert

`SPEECHPREFS.SEXMODE`
: `MALE FEMALE`

`SPEECHPREFS.VOLUME`
: Zahlenwert

`SPEECHPREFS.SPEECH`
: Wahrheitswert

5.21 Das SOUNDPREFS Objekt

Verfügbare Felder sind:

`SOUNDPREFS.BELLNAME`
: Text

`SOUNDPREFS.CONNECTNAME`
: Text

`SOUNDPREFS.DISCONNECTNAME`
: Text

`SOUNDPREFS.GOODTRANSFERNAME`
: Text

`SOUNDPREFS.BADTRANSFERNAME`
: Text

`SOUNDPREFS.RINGNAME`
: Text

`SOUNDPREFS.VOICENAME`
: Text

`SOUNDPREFS.PRELOAD`
: Wahrheitswert

5.22 Das CONSOLEPREFS Objekt

Dieses Objekt enthält keinerlei Felder, es enthält nur eine einzige Textzeile: die Spezifikation des Ausgabefensters.

5.23 Das FILEPREFS Objekt

Verfügbare Felder sind:

`FILEPREFS.TRANSFERPROTOCOLNAME`
: Text

`FILEPREFS.TRANSLATIONFILENAME`
: Text

`FILEPREFS.MACROFILENAME`
: Text

`FILEPREFS.CURSORFILENAME`
: Text

`FILEPREFS.FASTMACROFILENAME`
: Text

6 Gesucht!

Zum Zeitpunkt der Erstellung dieses Dokuments sind nur wenige ARexx-Beispielprogramme im 'term'-Paket enthalten (siehe '`Rexx`'-Schublade). Es wäre jedoch wünschenswert, wenn mehrere Programme beiliegen würden, damit mehr Anwender Nutzen aus der ARexx-Schnittstelle von 'term' ziehen können.

Wer ihre/seine ARexx-Programme mit der 'term'-Anwendergemeinde teilen möchte, der schicke die Programme samt Dokumentation an:

Olaf Barthel
Brabeckstraße 35
D-30559 Hannover
Bundesrepublik Deutschland
Z-Netz: O.BARTHEL@A-LINK-H.ZER
Internet: olsen@sourcery.han.de

Index

P

Q

R

S

T

U

W

Inhaltsverzeichnis