WRITE

COLLABORATORS			
	TITLE : WRITE		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 22, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

WRITE

1.1 WRITE.guide

Dokumentation für WRITE 3.1265 vom 8 January 1995
****

1.2 WRITE.guide/Wichtig

Einige wichtige Bemerkungen
****

- WRITE läuft nur unter OS 2.04 oder höher!!!

- Lesen Sie bitte ausführlich die Anleitung bevor Sie WRITE das erste Mal benutzen. Vor allen Dingen die Kapitel Copyright,Registrierung und Installation.

1.3 WRITE.guide/Autor

Autor
*****

Geld,Kritik,Vorschläge,Lob und Tadel in Deutsch oder English an:

Postadresse
Tim Teulings
An der Dorndelle 16
59192 Bergkamen
DEUTSCHLAND

EMailaddresse
MausNet:

Tim Teulings @ UN

InterNet:

rael@edge.ping.de

Bankverbindung
Sparkasse Berkamen - Bönen
BLZ : 410 518 45
KontoNr.: 16186496
Tim Teulings

1.4 WRITE.guide/Danksagungen

Danksagungen
****

Danke...

Chris Gray
Für den grandiosen Draco-Compiler mit dem WRITE bis zur Version 2.50 geschrieben wurde.

A + L
Für ihren Oberon-Compiler, der das Schreiben von WRITE noch einfacher machte.

Lars Hanke

Für Tips, Kritik und Beta-Testing, seine vielen Ideen, Featurewünsche und viele andere Sachen. Ohne ihn würde WRITE heute nicht so leistungsfähig sein.

Hartmut Goebel
Für Teile seines Paketes OberonBOOPSI, die in die hui.library eingebaut wurden, sowie Informationen über die Arbeitsweise der Textklasse des OberonSystems und schließlich einen Haufen von Sources, Ideen, Verbesserungsvorschläge und vieles mehr. Vielen Dank.

Thomas Wagner, der WRITE getest und auch einen Haufen von Fehlern gefunden hat.
Martin Loos, Olaf Peters und Mr. X, die mir sehr bei der Verbreitung von WRITE geholfen haben.

Thomas Lottermoser
Für Tips, Kritik und die Literatur.

Thomas Stuff
Für Tips, Ktitik und Beta-Testing.

Andreas Schulz
Für den sagenumwobenden DRACO-Club Deutschland.

Meine Freudin, für geduldiges Zuhören, Testen und das Interesse, wenn ich von Dingen erzählte, die sie eigentlich gar nicht interessierten :-), sowie für ihre Beta-Testerei. Sie hat es gewagt mit WRITE (und TeX) ihre Staatsarbeit zu schreiben.

COMMODORE
Für diesen schnuckeligen, kleinen Rechner, mit dem man Tag und Nacht viel Spaß haben kann. (Hehe...)

1.5 WRITE.guide/Einleitung

Einleitung
****

Ich besitze den AMIGA nun seit mehreren Jahren. In dieser Zeit bin ich mit vielen Texteditoren in Berührung gekommen. Anfangs das NOTEPAD und ED, später mit verschiedenen Versionen von EMACS sowie PD-Editoren wie DME, AZ, DED und JED, als auch einige kommerzielle Produkte. Mit allen konnte man Texte schreiben, doch auch alle ließen entweder bei der Geschwindigkeit oder beim Bedienungskomfort zu wünschen übrig. So erschütterten einige durch eine unerträgliche Scrollgeschwindigkeit, andere erschwerten einem die Arbeit durch unerträgliche und unmögliche Tastaturbelegungen und - kombinationen. Wieder andere erschlugen mich durch viele, für einen Texteditor unnötige Funktionen, die zu Lasten der Größe des Editors gingen, oder glänzten durch unzureichende Editiermöglichkeiten. So beschloß ich alle Vorteile in einem Programm zu vereinen (was sonst!) und einen Editor zu schreiben, der allen meinen Anforderungen entsprach.

* WRITE als reiner Texteditor ist dazu gedacht Texte oder Programme zu

schreiben.

* Er besitzt alle gängigen Blockoperationen (Cut, Copy, Paste...). Blöcke können dabei nicht nur zeilenweise sondern auch zeichenweise markiert werden.

* WRITE verfügt über umfangreiche Funktionen zum Suchen und Ersetzen von Wörtern etc.... Diese können auch mittels AREXX-Skripts einfach automatisiert und erweitert werden.

* WRITE besitzt eine Undo-Funktion, mit der eine beliebige, voreinstellbare Zahl von Textveränderungen wieder rückgängig gemacht werden kann.

* Seine Speicherverwaltung ist völlig dynamisch.

* Es können somit beliebig viele Textfenster geöffnet werden, der Text kann, beliebig lang sein. Einzige Begrenzung ist der vorhandene Speicherplatz.

* Auch Zeilen können beliebig lang sein.

* WRITE ist voll Style-Guide komform. Die Handhabung des Editors orientiert sich dabei an gängigen Features und Arbeitsweisen. D.h. alle Funktionen können über Menüs aufgerufen werden, der Cursor kann mit der Maus gesetzt, Blöcke mit ihr markiert werden.

* WRITE ist komplett konfigurierbar.Menüs und Tastatur können über eine Textdatei beliebig konfiguriert werden. Auch ist WRITE darüber hinaus weiter programmierbar. WRITE kann somit selbst in Details den Erfordernissen und Wünschen des Benutzers vollständig angepaßt werden.

* Es können unterschiedliche Konfigurationen gleichzeitig benutzt werden. So kann man in einem Fenster mit einer TeX-Konfiguration arbeiten, während gleichzeitig in einem anderen Fenster typische Einstellungen zur Programmierung in C verwendet werden. Gleichzeitig kann man mit der MAILER.CONFIG eine Mail mit Wordwrap etc. schreiben.

* Jede Konfiguration kann auf einem eigenen Screen mit einem eigenen Font in den eigenen Farben arbeiten. WRITE ist in allen seinen Fenstern und Requestern völlig auflösungs- und fontsensitiv. Es wird immer der eingestellte DefaultFont eines Screens verwendet. Dabei kann es sich sogar um einen Proportionalfont handeln.

* Dabei ist es möglich eigene Variablen sowie Makros zu definieren.

* WRITE beherrscht das automatische Generieren von Backups. Dabei kann einfach eine Endung an die Datei angehangen, die Datei unbenannt oder in ein angegebenes Verzeichnis unter einen festen Namen abgespeichert werden.

* WRITE kann Texte falten. Verschachtelte Textfalten sind möglich.

* WRITE läuft unter OS 2.04, OS 2.1, OS 3.0 und 3.1, und unterstützt daher viele Features von OS 2.0, wie z.B. AppWindows, -menus und -icons, sowie auch einige Features von 3.0.

* WRITE unterstützt die locale.library ab OS 2.1 und ist dadurch fast völlig an verschiedene Sprachen anpassbar.

* WRITE besitzt eine komfortable AREXX-Schnittstelle mit über 150 Funktionen und mehr als 80 Variablen. Alle Funktionen der internen MakroSprache sind auch über AREXX verfügbar.

* WRITE beinhaltet einige komfortable AREXX-Scripts für SAS C, DFA und den OBERON-Compiler der Firma A+L AG. Desweiteren liegen Scripts für umfangreichere Textverarbeitungsoperationen, sowie einige Beispielscripts für komplexere Funktionen bei.

* WRITE besticht in vielen Funktionen durch seine Schnelligkeit.

* WRITE ist völlig über eine eingebaute GUI vollständig konfigurierbar. Wahlweise kann aber auch direkt in dem Konfigurationsfile editiert werden.

* WRITE liegt die hui.library, eine objektorientierte fontsensitive GUI-Oberflächen erzeugende Library. Es ist geplant, die hui.library voll zu dokumentieren, so daß es für den User möglich ist, Zusatzapplikationen im selben Look und Feel von WRITE für WRITE zu schreiben, ohne dabei auf andere Oberflächengeneratoren zurückgreifen zu müssen. Ihre Zusatzapplikation bleiben klein und haben eine fontsensitive, schnelle StyleGuide Oberfläche.

* WRITE liegt auch die write.library bei, welche die externe Handhabung von WRITE (z.B. beim Schreiben eines Quickstarters) vereinfachen soll.

* WRITE besitzt sowohl einem Insert- als auch einen Overwritemodus.

* Leistungsfähiger Quickstarter.

* Asyncrone Requester

1.6 WRITE.guide/Copyright

Copyright
****

Demoversion	Copyright der Demoversion
Vollversion	Copyright der kommerziell vertriebenen Version
Allgemein	Allgemeine Hinweise für beide Versionen

1.7 WRITE.guide/Demoversion

Demoversion
==========

* Die Demoversion von WRITE ist Shareware. Sie darf unter folgenden Bedingungen benutzt und weitergereicht werden:

* Eine Diskette, die dieses Demo von WRITE in irgendeiner Form beinhaltet, darf nicht für mehr als 5 DM oder deren Gegenwert in einer ausländischen Währung kopiert, verkauft, weitergereicht werden.

* Fred Fish ist es ausdrücklich erlaubt diese Demoversion in seine Sammlung aufzunehmen.

* Ebenso ist es erlaubt die Demoversion auf Aminet zu packen. Desweiteren darf sie auch auf einer AminetCD erscheinen.

* Das Paket darf nur vollständig weitergereicht werden!

* Dem Benutzer wird es gestattet, WRITE ein bis zwei Monate zu benutzen, dann muß er sich registrieren lassen oder den Gebrauch von WRITE einstellen.

1.8 WRITE.guide/Vollversion

Vollversion
==========

* Diese Version von WRITE fällt ebenfalls unter dem Begriff Shareware. Es gelten jedoch folgende Einschränkungen

* WRITE darf nicht...

    ... in irgendeiner Form weitergereicht werden.

    ... von jemand anders als mir persönlich verkauft oder vertrieben werden.

* Eine vollständige Version ist bei mir unter der oben genannten Adresse für 30DM zu erhalten.

* Alle Limitierungen in der Demoversion sind in der vollständigen Version aufgehoben.

* Jeder, der eine voll funktionsfähige Version von WRITE mehr als einmal benutzt ohne sie offiziell gekauft zu haben, wird aufgefordert, ja sollte sich von seinem Gewissen gezwungen fühlen, diese nachträglich zu erwerben, da in WRITE mittlerweile mehr als vier Jahre Programmierarbeit stecken!

* Besitzer der vollständigen Version werden über Updates informiert und können diese für ein weiteres Entgeld erwerben.

* Ich (als der Autor von WRITE) verpflichte mich gravierende Mängel zu beheben.

* WRITE darf nur auf einen einzigen Rechner installiert sein. Er darf z.B. also nicht in Netzwerken oder auf Mehrplatzsystem benutzt werden. Sollte dies jedoch erforderlich sein, so bitte ich um Rücksprache.

Ich möchte hier außerdem ausdrücklich bemerken, daß ich nicht verpflichtet bin an WRITE weiterzuarbeiten. Ich kann jederzeit Teile des Editors ändern, rauschmeißen, Neues einfügen. Das Format des Konfigurationsfiles kann sich jedezeit ändern. Kompatibilität zu älteren Versionen kann somit auch auf Grund der Leistungsoptimierung nicht immer gewährleistet werden. Ich kann jeder Zeit Preise und Kaufbedingungen ändern.

Das heißt natürlich nicht, daß ich mich nicht den Wünschen der User beuge. Einige Teile von WRITE wie z.B. die Undofunktion und Folds wären ohne die Wünsche meiner Betatester nicht implementiert worden. Dies heißt eher, daß ich, um den Editor zu verbessern, schlechtere Konzepte etc. bereit bin herauszuschmeißen.

Ich mache darauf aufmerksam, daß die Programmiererei nur eines meiner Hobbies ist und ich auch noch nebenbei studiere. Sollte also nicht sofort am nächsten Tag WRITE in der Post liegen, so gilt der klassische Spruch: KEINE PANIK!!!. Einen bis eineinhalb Monate Bearbeitungszeit nehme ich mir in Ausnahmefällen schon heraus. Möglicherweise programmiere ich auch gerade an einer neuen Version mit doppelt so vielen Features...?

Schließlich bin ich nicht in der Lage nach jedem Bugfix allen Besitzer eine neue Version zuzuschicken (ich habe einfach nicht das Geld!). Sollte es jedoch bei einzelnen Personen zu gravierenden, unumgehbaren Fehlern in hochwichtigen Funktionen kommen, so sollte sich immer etwas machen lassen. Spätestens dann, wenn man mir mit der ausführlichen Fehlerbeschreibung einen Rückantwortbrief beilegt.

Registrierte Besitzer erhalten einen Keyfile, mit dem sie in der Lage sind jede neu herausgekommene Version sofort, ohne weitere Umstände, zu benutzen. Über größere Updates werden alle registrierten Benutzer postalisch/EMail benachrichtigt. Ein Update kann man auch direkt von mir gegen eine Gebühr von 5DM bzw. 10DM für alle Länder außerhalb Europas erhalten.

Die Zeit, die ich in WRITE investiere, sowie die Höhe der Registrierungsgebühr hängen entscheidend von der Zahl der Registrierungen ab. Also ... Je mehr sich registrieren lassen ...

1.9 WRITE.guide/Allgemein

Allgemein
=========

* Ich kann keine Haftung für jegliche Schäden, die direkt oder

indirekt durch WRITE entstehen, übernehmen. So bin ich z.B. nicht für versehentlich gelöschte oder durch Absturz verlorengegangende wichtige Dokumente, noch für qualmende Hardware, defekte Software, verwirrte Geister, gescheiterten Ehen etc., die irgendwie berechtigt oder unberechtigt mit WRITE in Verbindung gebracht werden können, verantwortlich zu machen. WRITE ist gut aber nicht perfekt. (noch nicht!)

* Jegliche Änderungen an Teilen dieses Paketes (Docs...) sind verboten! Davon ausgenommen sind:
  1. Übersetzungen. Dem Originalarchiv dürfen von 3. Hand Übersetzungen der Dokumentation, oder auch nur Teile dieser, beigefügt werden. Dabei sollte darauf geachtet werden, daß durch die Übersetzung der Text nicht verändert wird. Bevor jemand anfängt Teile zu übersetzen, sollte er umbedinngt mit mit Kontakt aufnehmen.

  2. Das Ändern der Icons sowie der Tooltypes, als auch die Position der Icons und Drawers.

* Das widerrechtliche Benutzen (Klauen) von Teilen dieses Paketes ist ebenfalls verboten!

* Es ist verboten WRITE zu dissassemblieren, decodieren, decompilieren...

* Der Käufer akzeptiert mit der Benutzung von WRITE obige Konditionen.

* Jeder, der gegen diese Bedingungen verstößt, sollte sich bewußt sein, daß damit gegebenenfalls eine strafbare Handlung begeht, gegen die ich entsprechend vorgehen werde. Speziell das Vertreiben der Vollversion von WRITE als Raubkopie fällt unter diesen Punkt.

1.10 WRITE.guide/Registrierung

Registrierung
****

Wie bekomme ich denn nun die Vollversion von WRITE?

1. Sie schicken mir offiziell eine Bestellung per Post oder EMail mit ungefähr folgendem Wortlaut

   "Hiermit bestelle ich die neuste Vollversion von WRITE. Ich akzeptiere die in der Anleitung unter den Punkten Copyright und Registrierung genannten Kaufbedingungen und Hinweise."

   Fügen Sie dem Brief auf jeden Fall Ihre komplette Adresse, wenn möglich auch Ihre EMailadresse und Ihre Telephonnummer bei. Sollte es zu Rückfragen kommen, ist es sicherlich in Ihrem Sinne, wenn ich Sie so schnell wie möglich erreiche.

   Geben Sie an, wie viele Versionen Sie von WRITE haben wollen.

Geben Sie an, ob Sie WRITE sofort bekommen wollen, oder noch gegebenenfalls auf eine neue Version warten wollen.

2. Lassen Sie mir das Geld zukommen. Sie können mir das Geld in Form von Bargeld oder auch als Scheck zukommen lassen. Auch eine Überweisung auf das unter Autor angegebene Konto, sowie eine postalische Zustellung, sind möglich. Naturalien nehme ich nicht an. Im Einzelfall läßt sich aber auch über eine Begleichung durch ein eigenes, sharewarebehaftetes Profukt reden.

   Für Versendung außerhalb Europas bitte ich Sie, weitere 5DM zu überweisen!

   Stellen Sie auf jeden Fall sicher, daß ich mit dem Geld auch Ihren Namen und Adresse erhalte, damit ich Ihnen auch WRITE zustellen kann.

   Innerhalb Deutschland empfehle ich die Überweisung des Geldes. Egal zu welcher Zusendungform Sie sich entschließen, muß gewährleistet sein, daß ich Abzug aller durch die Zusendungsart entsehenden Kosten 30DM in der Hand halte. Gerade wenn Sie aus dem Ausland kommen, sollten Sie dies unbedinngt beachten!

3. Warten Sie darauf, daß der Postbote Ihnen WRITE ins Haus bringt. Dies sollte im Normalfall 1-2 Wochen dauern, im Einzelfall etwas länger. Sollte ich mich nach mehr als einem Monat noch nicht gemeldet haben, so gehen sie davon aus, daß mich Ihr Brief nicht erreciht hat.

4. Klatschen Sie in die Hände, freuen Sie sich und werden Sie sich bewußt, daß Sie einen weiteren, entscheidenen Schritt in Ihrem Leben getan haben.

5. Ich möchte noch einmal darauf hinweisen, daß wenn Sie über einen Netzanschluß (InterNet,Fido,Zerberus,Maus etc.) mir Ihre Adresse unbedingt mitteilen sollten. Dies ermöglicht es mir Ihnen Updates ggf. zuzuschicken, Sie um Ihre Meinung zu fragen, Ihnen bei Problemen zu helfen etc.

1.11 WRITE.guide/NeueVersion

Neue Version
****

Die neuste Version von WRITE sollte immer an den folgenden Orten erhältlich sein:

* Maus Unna II. Tele.: 02303/66232, öffentlicher Programmteil, unter dem Namen WRITE.lha, WRITEV3.lha oder ähnlich. Am besten nach WRITE* suchen (lassen).

* Aminet. Verzeichnis util/edit

1.12 WRITE.guide/Installation

```
Installation
************

Folgende Sachen sind unbedingt zu beachten:

   * Auf Ihrem System muß OS 2.04 oder höher installiert sein! Einige
     (unwesentliche) Features lassen sich erst ab 3.0 nutzen.

   * Folgende Libraries sollten sich im LIBS:-Verzeichnis befinden.  Sie
     sind Bestandteil des Betriebssystems.

    asl.library
    diskfont.library
    commodities.library
    iffparse.library
     optional für REXX:

    rexxsyslib.library
    rexxsupport.library
   * Es sollte genügend Speicher vorhanden sein!  Obwohl WRITE
     speichersparsam programmiert ist, braucht er doch einige 100KByte an
     Speicher. Doch ist es möglich, durch Verzicht auf ein wenig
     Komfort, den Speicherbedarf zu veringern.

   * WRITE braucht recht viel Stack (ca. 20000 Bytes).  Sollte dieser
     nicht vorhanden sein, setzt WRITE ihn selbst auf 30000 Bytes herauf.

   * Auf der Diskette, die Sie bekommen haben, sollte sich ein
     Installer-Skript für den Installer, den Sie mit Ihren
     Betriebssystemdisketten bekommen haben, befinden.  Alles was Sie
     also im Normalfall zu tun haben, ist mit einem Doppelklick auf das
     entsprechende Icon das automatische Installationsprogramm zu starten
     und die Fragen Ihren Wünschen entsprechend zu beantworten.

     Wollen Sie dennoch WRITE manuell installieren, sollten Sie wie folgt
     vorgehen.

       1. Einige Dateien auf der Diskette sind eventuell mit lha gepackt,
          man erkennt sie an der Endung .lha, und müssen gegebenenfalls
          entpackt werden.  Sollten Sie keine Version von Lha besitzen,
          benutzen Sie das Programm im c-Verzeichnis. Eine Anleitung zu
          diesem Programm finden Sie im Verzeichnis Docs/.

       2. Der(Die) Konfigurationsfile(s) sollte(n) auf folgende Weise
          installiert werden:

            1. Schaffen Sie auf Ihrer Platte ein neues Verzeichnis mit
               beliebigen Namen.

            2. Kopieren Sie alle Konfigurationsdateien dort hinein.
```

3. In der Umgebungsvariable WRITE.CONFIG des Betriebsystems sollte der vollständige Pfad dieses Verzeichnisses stehen. Dies kann dadurch erreicht werden,...

   ... daß man diese mit dem Befehl SetEnv (oder Set für eine COMMODORE-SHELL) manuell nach jedem Bootvorgang setzt.

   ... daß man einen dieser Befehle in seine startup-sequence oder noch besser in die user-startup einfügt.

   ... daß man nach Setzen dieser Variable den File WRITE.CONFIG aus dem ENV:-Verzeichnis ins ENVARC:-Verzeichnis kopiert, so daß die Umgebungsvariable automatisch nach jedem Boot-Vorgang gesetzt wird.

   (Letzteres ist meiner Meinung nach die geschickteste Methode!)

   WRITE sucht nun alle Konfigurationsdateien in diesen Verzeichnis, oder, falls die Umgebungsvariable nicht gesetzt wurde, im aktuellen. Zur Referenz des Konfigurationsfiles brauchen Sie somit immer nur den Dateinamen ohne Pfad.

3. Kopieren Sie die garbagecollector.library, hui.library und die write.library ins LIBS:-Verzeichnis. Desweiteren kopieren sie den Inhalt der Verzeichnisse libs/images und libs/gadgets in die entsprechenen Verzeichnisse im Verzeichnis SYS:classes (Workbench-Verzeichnis auf Ihrer Festplatte). Sollte das Verzeichnis SYS:classes/ nicht existieren, so kopieren sie die Unterverzeichnisse in ihr LIBS:-Verzeichnis.

4. Kopieren Sie die Datei Docs/WRITE.guide in ein Verzeichnis, wo sie von AmigaGuide bzw. MultiView gefunden wird. Der Guide-File ist für die Online-Hilfe wichtig.

   Es besteht auch die Möglichkeit die Umgebungsvariabel WRITE.guide mit dem kompletten Pfad des Guide-Files zu setzen.

5. Kopieren Sie die .catalog-Files in die entsprechenden Verzeichnisse Ihrer Workbench.

6. Kopieren Sie die REXX-Scripts in ein Verzeichnis, welches im Suchpfad von REXX liegt.

7. Kopieren Sie nun WRITE selbst, samt seinem Icon, in das von Ihnen gewünschte Verzeichnis. Setzen Sie die Umgebungsvariable WRITE nach einer der oben beschriebenen Methoden dauerhaft auf den vollständigen Pfad von WRITE (z.B. dh0:WRITE oder C:WRITE). Einige AREXx-Scripts lesen diese Variable aus, um WRITE, wenn noch nicht gestartet, hochzufahren.

8. Zu WRITE gehört auch ein Quickstarter names QuickWrite (Starten). Kopieren Sie auch diesen, wenn

Sie wollen, in ein beliebiges Verzeichnis und geben Sie ihm
einen ebenfalls beliebigen Namen.

* Beenden Sie möglichst alle weiteren Programme, oder machen Sie einen Reset, starten Sie das GarbagePrefs-Programm in der Prefs-Schublade der Installationsdiskette, wählen Sie den Menüpunkt Editieren/Werte vorschlagen, bestätigen Sie mit Speichern und verlassen das Programm.

* Starten Sie WRITE.

Schließlich (auch nach der Installation durch das Installationsscript) sollten unbedingt noch folgende Dinge getan werden:

* Kontrollieren Sie, ob die Einstellungen im File STARTUP.CONFIG in dem Icon von WRITE Ihren Wünschen entsprechen. Es kann sein, daß die dort definierten Hotkeys sich mit bereits vorhandenen Hotkeys überschneiden, oder daß die eingestellten Backup-Modi auf nicht existierende Verzeichnisse zeigen.

1.13 WRITE.guide/Starten

Starten von WRITE (CLI/WB/QuickWrite)
****

Es gibt drei Möglichkeiten WRITE zu starten. Aus einem CLI heraus, von der Workbench und mittels QuickWrite. QuickWrite ist dabei ein kleines Programm, welches WRITE lädt, falls er noch nicht geladen ist, und dann über den AREXX-Port von WRITE die angegebenen Dateien lädt.

Beim Start von WRITE wird dann zuerst der ToolType/CLI-Argument STARTUPCONFIG ausgewertet. Dann die entsprechende STARTUP.CONFIG geladen. Anschließend werden dann die restlichen ToolTypes/CLI-Argumente ausgewertet. Für weitere Informationen über die Toolstypes/CLI-Argumente von WRITE schauen Sie bitte im Kapitel ToolTypes nach.

Eine der dabei wichtigsten Einstellung, ist SLEEPMODE. Wird WRITE im SLEEPMODE gestartet, so werden keine Fenster und keine Konfigurationen geladen. Auch wenn das letzte Textfenster geschlossen ist, wird WRITE nicht beendet. Dieses Verhalten ist sinnvoll, wenn WRITE über QuickWrite benutzt werden soll. Dieser startet WRITE und öffnet Fenster, lädt Texte etc. über dessen AREXX-Port.

Die ToolTypes/CLI-Parameter von QuickWrite sind:

CONFIG/K,LINE/K/N,SCREEN/S,STICKY=-STICKY/S,HIDE/S,FILE/M

* FILE steht hierbei für eine beliebige Zahl von Texten.

* CONFIG bestimmt die zu ladene Konfiguration für die Texte.

* LINE gibt eine Zeile an, zu der nach dem Laden gesprungen werden

soll.

* SCREEN sorgt dafür, daß der Screen, auf dem das Textfenster geöffnet wird, nach dem Laden nach vorne und nach dem Schließen wieder nach hinten gebracht wird.

* STICKY sorgt dafür, daß QuickWrite erst beendet wird, wenn das entsprechende Textfenster geschlossen wurde.

* HIDE schließlich lädt den Text, öffnet aber kein Fenster.

Zu beachten ist, daß die Parameter LINE, SCREEN und STICKY nur dann funktionieren, wenn genau 1 File als Textdatei angegeben wird.

1.14 WRITE.guide/Konzept

Konzept
****

Das Prinzip Allgemeine Einleitung
Der Ed Die interne Repräsentation eines Textes

1.15 WRITE.guide/Das Prinzip

Das Prinzip
===========

Viele Texteditoren sind durch eine feste Tastatur- und Menübelegung recht beschränkt in ihren Möglichkeiten. Stört den Benutzer irgendetwas an der Handhabung, so muß er sich daran gewöhnen oder mit einem anderen Texteditor vorlieb nehmen.

Bei der Programmierung von WRITE wurde versucht, dem Benutzer Möglichkeiten zur Beeinflussung der Handhabung, des äußeren Erscheinungsbildes mitzugeben.

Dies wird dadurch erreicht,daß WRITE mittels einer kleinen, in den Editor eingebauten Interpretersprache, einer Mischung aus der Batchsprache des Betriebssystems und AREXX, programmierbar ist.

Der Editor führt also direkt keine Tastaturkommandos oder Menüpunkte aus, sondern nur noch Befehlsfolgen, die der Tastatur und Menü beliebig zugewiesen werden können. Der Benutzer kann frei bestimmen, was passiert, wenn diese Taste gedrückt oder jener Menüpunkt ausgewählt wird. Ja selbst die Menüs an sich sind frei definierbar; und sollten die Möglichkeiten des internen Interpreters nicht reichen, so können AREXX-Makros verwendet werden. Jeder interne Befehl ist auch über AREXX erreichbar.

1.16 WRITE.guide/Der Ed

```
Der Ed
======

Die interne Struktur zum Verwalten eines Textes wird Ed genannt.  Für
jeden Text, der geladen wird, gibt es einen Ed.  Im Prinzip kann man
einen Ed mit einem Editorfenster identifizieren (Obwohl nicht jeder Ed
ein offenes Fenster besitzen muß).

Jedem Ed kann bei seiner Erschaffung (mittels NewEd) eine eigene
Konfigurationsdatei zugewiesen werden. WRITE lädt diese Datei, wenn sie
nicht schon bereits für einen anderen Ed benutzt wurde, automatisch nach.
D.h., startet man WRITE so, daß er keinen Ed öffnet, lädt er auch
keinen Konfigurationsfile.

Dies klingt alles recht kompliziert. Sie vermuten, daß Sie nun vor der
ersten Benutzung erst einmal stundenlang die Anleitung studieren müssen.
Doch keine Panik!  Es liegt ein dokumentierter Beispielfile bei, der die
Möglichkeiten von WRITE ausführlich demonstriert.  Spielen Sie mit
diesem ein wenig herum, merken Sie sich Dinge, die Sie stören, schauen
Sie sich Ihre Realisierung im Konfigurationsfile an, versuchen Sie
mittels der Dokumentation zu verstehen, was passiert und überlegen Sie
sich , wie sich dies anders machen ließe, um dann schließlich den
Konfigurationsfile zu ändern.  Achten Sie dabei darauf, daß Sie immer
einen lauffähigen Konfigurationsfile parat haben, da WRITE sich mit
einem fehlerhaften Konfigurationsfile nicht starten läßt. Sie währen
somit nicht in der Lage WRITE zu benutzen, um die Fehler zu korrigieren.
Das Beste ist es, die geänderte Stelle zu markieren und sie mit dem
Menüpunkt INTERN/Parse block auf ihre syntaktische Richtigkeit zu
überprüfen.
```

1.17 WRITE.guide/Syntax

```
Die Syntax des internen Interpreters
************************************


  allgemeine Syntax   Allgemeine Syntax
  Typen               Die verschiedenen Typen
  Variablen           Variablen
  Konstanten          Konstanten
```

1.18 WRITE.guide/allgemeine Syntax

```
Allgemeine Syntax
=================

Der interne Interpreter verarbeitet beliebig lange Befehlsfolgen.  Diese
bestehen aus einer Liste von Befehlen, im weiteren auch Funktionen
```

genannt, samt ihrer Argumente. Dabei besitzt jeder Befehl eine feste Zahl von Argumenten von jeweils klar definiertem Typ. Argumente dürfen also z.B. nicht einfach weggelassen werden. Ein Beispiel:

```
Befehl1 Argument1
Befehl2 Argument1 Argument2 Argument3
```

Dem internen Parser (im Gegensatz zu AREXX) ist es dabei egal wie die Befehlsfolge formatiert ist. So kann obriges Beispiel auch folgenderweise geschrieben werden:

```
Befehl1 Argument1 Befehl2
Argument1 Argument2
Argument3
```

Der Interpreter arbeitet nun die Befehlsfolge von oben nach unten ab. Doch ist dies nicht immer sinnvoll. So öffnet der 1. Befehl z.B. ein Fenster, während der 2. in dieses einen Text schreibt. Was würde passieren wenn z.B. das Fenster wegen Speichermangel nicht geöffnet werden kann? Es würde versucht werden, in ein nicht vorhandenes Fenster einen Text zu schreiben. Deshalb geben einige Befehle einen Fehler zurück. Ist ein Fehler aufgetreten, so wird die Abarbeitung der Befehlsfolge sofort abgebrochen.

Eine komplette Auflistung aller Funktionen finden Sie im Kapitel Funktionsbeschreibung.

Beachten sie auch, daß sie mittels des Befehls Macro weitere Funktionen selbst definieren können.

1.19 WRITE.guide/Typen

```
Typen
=====

Die verschiedenen Typen
=======================

Der Interpreter unterscheidet zwischen 4 verschiedenen Typen:

  1. Zahlen. Zahlen sind vorzeichenlose Dezimalzahlen von 0 bis etwas
     über 100000000.  Einige Beispiele:

          1234 , 4567 , 229874 , 0

     Zahlen sind zuweisungskompatibel zu Zeichenketten. Ihr Inhalt wird
     bei einer Zuweisung automatisch in einen String umgewandelt.

  2. Strings. Es gibt zwei unterschiedliche Arten

       1. Wörter.  Wörter bestehen aus einem optionalen Unterstrich
          oder Klammeraffen (@), einem Buchstaben und weiteren
          Buchstaben oder Ziffern. Bei jedem Wort schaut der Interpreter
          nach, ob es sich bei dem Wort nicht um einen Befehl oder eine
```

Variable handelt. Befehle werden ausgeführt, Variablen durch ihren Inhalt ersetzt. Beispiele:

```
_EinWort , W2345r , _w2swd
```

2. Zeichenketten. Zeichenketten können im Gegensatz zu Wörtern aus beliebigen Zeichen bestehen. Sie werden durch sogenannte Quotes umschlossen. Quotes können sein: "",(),‘’.Variablen können mittels $VariableName$ in die Zeichenkette eingefügt werden. Auch können Zeichenketten mittels des +-Zeichens über mehrere Zeilen verteilt werden:

```
"1. Teil des Strings"+
(2. Teil des Strings)
```

Zeichenketten sind bedingt zuweisungskompatibel zu Zahlen. D.h. steht in dem String eine Zahl und nur diese Zahl, dann kann dieser String auch einer Zahl zugewiesen werden, ansonsten beschwert sich der Parser mit einer entsprechenden Meldung. Beachten Sie daß Ihr String zwar auch negative Zahlen beinhalten darf, diese jedoch vom Interpreter wieder in positive Zahlen umgewandelt werden. Auch ist es möglich, daß der String nicht eine Dezimalzahl, sondern eine Hexadezimalzahl der Form "4E75H" beinhalten kann.

3. Tags. Bei Tags handelt es sich um eine Liste, von weiteren, optionalen Parametern, die die Arbeitsweise einer Funktion ändern, varieren. Tags bestehen aus einer einleitenden,geschweiften Klammer { beliebig vielen Zahlen oder Konstanten und einer abschließenden, geschweiften Klammer }. Leere Tags können und sollten weggelassen werden.

4. Funktionsliste. Funktionslisten bestehen aus einer Befehlsfolge (d.h. einer Reihe von Befehlen samt ihren Argumenten) und einem abschließenden Semikolon. Beispiel:

```
Befehl1 Argument1 Befehl2 Argument1 Argument2 Argument3;
```

5. Kommentare. Kommentare fangen mit /* an und hören mit */ auf. Alles, was im Kommentar steht, wird vom Parser ignoriert. Kommentare können überall gesetzt werden. Kommentare können auch verschachtelt werden. Folgendes Konstrukt ist deshalb erlaubt:

```
/* Ein paar Sachen zum Debugen

  /* Aufruf 1 */

  blabla 1 2 3

  /* Aufruf 2 */

  blupblup "Eine Wasserblase "

*/
```

1.20 WRITE.guide/Variablen

```
Variablen
=========
```

Damit der Benutzer nicht nur die internen Funktionen aufrufen, sondern auch auf interne Einstellungen zugreifen und diese verändern kann, gibt es Variablen. Variablen sind vom Typ Zeichenkette oder Zahl. Auch hier gilt: Zahlen sind zuweisungskompatibel zu Zeichenketten, Zeichenketten bedingt zuweisungskompatibel zu Zahlen. Dies ist mit Vorsicht zu behandeln. Da der Parser während des Parsens nicht feststellen kann, ob die Variable nun eine Zeichenkette oder eine Zahl beinhaltet, fällt diese Aufgabe dem Interpreter zu, der dann einfach mit einem Fehler abbrechen wird.

Möchte man nun z.B. als Parameter einer Funktion statt einer Zahl eine Variable angeben, so tätigt man dies einfach durch die Nennung des Variablenamens. Hat zum Beispiel die Variable x den Inhalt 25, so sind die Funktionsaufrufe

```
MacheMitZahl 25
MacheMitZahl x
```

identisch.

Im Zusammenhang mit Funktionen gibt es zwei spezielle Variablen: _RS und _RN. Sie existieren, da einige Funktionen ein Resultat zurückgeben. Ist der Rückgabewert vom Typ Zahl, dann steht er in _RN; ist er vom Typ Zeichenkette, dann steht er in der Variable _RS. Auf den Rückgabewert einer Funktion kann also solange zugegriffen werden, bis er von einer Funktion selben Rückgabetyps überschrieben wird.

Wie schon in Typen erwähnt, können Variablen durch $VariableName$ in einen String eingefügt werden. So ergibt Message "Zeit: $_Time$", zum Beispiel, einen Requester mit der aktuellen Zeit.

Bitte beachten Sie, daß einige Variabeln nur geschrieben/ausgelesen werden können, wenn der entsprechende Ed/die entsprechende Konfiguration aktiviert ist. Dies geschieht mit LockWindow und GetConfig.

Mittels des SetVar-Befehls ist es auch möglich eigene Variablen zu definieren.

Eine ausführliche Auflistung aller Variablen finden Sie im Kapitel Variablebeschreibung.

1.21 WRITE.guide/Konstanten

```
Konstanten
==========
```

Konstanten wurden hauptsächlich für den Typ Tag eingeführt. Durch Kombination der beiden ist es möglich, auf äußerst einfache Weise das

Verhalten von Funktionen zu beeinflussen. Konstanten verhalten sich für
den Benutzer genauso wie Variablen. Der einzige Unterschied ist eben der,
daß der Inhalt einer Variablen nach dem Start von WRITE immer gleich
bleibt und daß die Konstanten nicht geändert werden können.

1.22 WRITE.guide/ToolTypes

ToolTypes
****

Über die Toltypes können einige wichtige Einstellungnen von WRITE gleich
beim Starten von WRITE getätigt werden.

Beachten Sie, daß WRITE, im Gegensatz zum üblichen Vorgehen, seine
ToolTypes immer analysiert. D.h., sowohl beim Start von der Workbench,
als auch beim Start aus einer Shell, werden die ToolTypes gelesen.

Die ToolTypes und ihre Bedeutung:

* APPMENU Setzt die Variable _AppMenu.

* APPICON Setzt die Variable _AppIcon.

* APPWIN Setzt die Variable _AppWindow.

* SLEEPMODE Setzt die Variable _SleepMode.

* STDCONFIG Setzt die Variable _DefaultConfig

* GUIDEMODE Ist diese Variable FALSE, so öffnet WRITE beim Hochfahren
  gleich den Hilfsfile WRITE.guide. Ist die Variable TRUE, so wird
  der Guidefile erst explizit beim Aufruf der Hilfe geöffnet und
  anschließend wieder geschlossen.

  Die erste Methode ist schneller. Der Guide wird einmal geladen und
  steht dann immer schnell, da speicherresident, zur Verfügung. Der
  Nachteil ist, daß der Guidefile permanent im Speicher gehalten
  wird. Nach groben Schätzungen müßten dies mindesten 300 - 400 KB
  sein. Die zweite Methode ist wesentlich langsamer (auf meinen 68020
  dauert es schon ein paar Sekunden bis AmigaGuide da ist), aber halt
  speicherplatzsparend, da der GuideFile bei jeder Anfrage neu geladen
  wird.

* STARTUPCONFIG Hier kann eine alternative Startup.config gewählt
  werden.

Die Parameterzeile für die Shell lautet:

  FILES/M,START=STARTUPCONFIG/K,SLEEP/S,NOSLEEP/S,STDCONFIG/K

* FILES steht für eine beliebige Zahl von Texten

* STARTUPCONFIG und STDCONFIG haben die gleiche Funktion, wie die
  entsprechenden ToolTypes.

* SLEEP und NOSLEEP starten WRITE im entsprechenden Modus. (_SleepMode.

1.23 WRITE.guide/Editorfenster

Editorfenster
****

Das Editorfenster gliedert sich in drei wesentliche Teile: Die Titelzeile mit einigen wichtigen Informationen über den Text, den Scrollbalken auf der rechten Seite, so wie der eigentliche Editierbereich. Der Scrollbalken verhält sich wie üblich und bedarf wohl keiner weiteren Erklärung, ähnlich wie der Editierbereich. Im weiteren soll auf die Bedeutung der verschiedenen Mauszeiger und der kryptischen Zeichen in der Titelzeile des Fensters eingegangen werden.

1. Der normale Mauszeiger. Dies ist der Betriebssystemmauszeiger, der normalerweise benutzt wird.

2. Der Busy-Mauszeiger, eine kleine Uhr. Dieser Mauszeiger erscheint immer dann, wenn WRITE gerade arbeitet (z.B. beim Abspeichern) und auf die Eingaben des Benutzers nicht reagieren kann. Bitte warten Sie dann mit Ihren Eingaben bis der normale Mauszeiger wieder erscheint.

3. Der Sleep-Mauszeiger. Dieser Mauszeiger erscheint immer dann, wenn die graphische Ausgabe von WRITE abgeschaltet ist. Dies ist sinnvoll, wenn ein Script viele Änderungen am Text vornimmt, da das Abschalten zu einem nicht unwesentlichen Performancegewinn führt. Der Benutzer sollte auch hier alle Eingaben unterlassen.

4. Der Markierungs-Mauszeiger, ein Visier. Dieser Mauszeiger erscheint immer dann, wenn man eine Blockmarke gesetzt hat. WRITE erwartet dann, daß man eine zweite Marke setzt, um einen Block zu markieren.

Die Titelzeile gliedert sich ebenfalls in vier Teile.

Im ersten Teil zeigen einige Buchstaben wesentliche Informationen zum Text. Erscheint an der ersten Stelle eine M, so wurde eine Marke gesetzt; erscheint ein B, so wurde eine zweite Marke gesetzt und damit ein Block markiert. Erscheint an der zweiten Stelle ein *, so wurde der Text nach dem letzten Abspeichern oder Laden verändert. Ein R an vierter Stelle bedeutet, daß REXX installiert wurde und daß WRITE REXX-Scripts ausführen kann. Ein C an vierter Stelle bedeutet, daß WRITE beim Suchen zwischen Groß- und Kleinschreibung unterscheidet. Ein S an fünfter Stelle zeigt an, daß sich WRITE im Sleep-Modus befindet. Und ein E an Position 6, zeigt an, daß der Text editiert werden kann.

Als nächstes folgen drei Zahlen. Diese stehen für die Spalte und die Zeile im Text, in der sich der Cursor befindet, und die Länge des Textes in Zeilen.

Anschließend folgt, wenn die Undo-Funktion aktiviert ist (_Undo), in eckigen Klammern die Zahl der Textänderungen, die WRITE sich gemerkt hat

und die man rückgängig machen kann.

Schließlich, als letztes, steht in der Titelzeile der Name des aktuellen Files.

1.24 WRITE.guide/Requester

Requester
****

* Die Requester von WRITE sind alle fontsensitiv. Das heißt, daß sie sich an die Größe des eingestellten Systemfonts anpassen. So ist selbst bei großen Fontgrößen, wie man sie z.B. als Besitzer einer Graphikkarte benutzt, ein übersichtliches, lesbares, komfortables Layout gewährleistet.

* Alle Requester unterstützen Shortcuts. Das bedeutet, daß die meisten Gadgettypen nicht nur mittels der Maus, sondern auch über die Tastatur selektiert werden können. Angezeigt wird der Shortcut durch einen Strich unter dem entsprechenden Buchstaben. Außerdem gibt es für jedes Fenster meist ein Gadget, welches über die Return-Taste selektiert werden kann, und eins, welches man mit der Escape-Taste erreichen kann. Ersteres wird dadurch angezeigt, daß dieses Gadget einen zweiten Rahmen erhält, der es noch weiter hervorgehoben erscheinen läßt, zweiteres dadurch, daß dieses einen Rahmen erhält, der das Gadget in die Oberfläche "eingebettet" aussehen läßt. Bitte machen Sie sich bei der Wahl der Shortcuts, der Belegung von Return- und Esacapegadgets und bei der Reihenfolge der Gadgets Gedanken, versuchen Sie möglichst die Logik der Betriebssystemsrequester und WRITE zu kopieren...

* Man kann für die internen Requesteraufrufe global über die Variable _ReqToMouse, bzw. für die meisten Requester bei externen Aufrufen über das Tag @TOMOUSE WRITE so konfigurieren, daß Requester immer unter dem Mauszeiger geöffnet werden. Dies ist vorteilhaft für Leute mit Tools, die das Fenster unter dem Mauszeiger automatisch aktivieren, oder auch für Leute mit großen Screens.

1.25 WRITE.guide/Listen und Buffer

Listen und Buffer
****

Listen und Buffer werden in WRITE dynamisch verwaltet. Dies bedeutet, daß man beliebig viele Listen und Buffer haben kann, deren Größe nur durch den vorhandenen Speicher begrenzt ist. Alle Listen und Buffer haben einen Namen, über den sie jederzeit referenziert werden können. Eine Liste oder Buffer mit einem bestimmten Namen erzeugt man dadurch, daß man etwas reinschreibt oder löscht.

Ein Eintrag in eine Liste darf maximal 255 Zeichen lang sein. Alles darüber wird abgeschnitten.

WRITE besitzt intern beim Programmstart schon einige Listen:

* List-List Eine Liste, in der alle Listen stehen.

* Buffer-List Die Liste aller Buffer, die momentan existieren.

* FindHistory Liste aller Suchwörter seit dem Programmstart.

* ReplaceHistory Liste aller Ersetzungswörter seit dem Programmstart.

* DeleleteLine-History Liste aller durch DeleteLine gelöschter Zeilen.

Listen sind sehr leistungsfähig. Ein Beispiel dafür ist das REXX-Script ListDemo.wrx.

Ein anderes Beispiel, welches eine DeleteWord / UndeleteWord-Funktion als ein Menüpunt implementiert, soll folgen:

```
ITEM "Delete word" "control delete"
  IF
    COMPARE _CURRENTWORD "";
    NOP;
    PUSH "CED-Word-History" _CURRENTWORD
    DELETEAREA @SOW @SOW @EOW @EOW {@SILENT};
  ;

ITEM "Undelete word" "control alt delete"
  POP "CED-Word-History"
  WRITETEXT _RS
  ;
```

1.26 WRITE.guide/PublicScreens

Public Screens
****

Wenn nicht anders angeben, öffnet WRITE seine Fenster immer auf der Workbench. Es ist aber auch möglich, Fenster auch auf anderen, öffentlichen Screens (Public Screens) zu öffnen. WRITE besitzt dabei keinen eigenen Screenmanager, daß heißt, daß WRITE nicht in der Lage ist, selbst Screens zu öffnen, sondern ist darauf angewiesen, daß andere Programme diese Screens für ihn öffnen.

So kann man in dem entsprechenden Konfigurationsfile ein Programm starten, welches einen öffentlichen Screen öffnet, oder über AREXX ein bereits laufendes Programm dazu veranlassen. Anschließend wird dieser Screen über die Funktion Screen angemeldet. Der Screen wird gelockt, d.h. dem Screenmanager wird nicht erlaubt das Fenster wieder zu schließen. Dieser wird erst wieder freigegeben, wenn man für diese Konfiguration einen anderen Screen anmeldet oder die Konfiguration

beendet. WRITE wird also immer, wenn die Konfiguration aktiv ist, alle seine Fenster auf diesem Screen öffnen. Über die UserFunktion 4 , die mittels SetUserFkt gesetzt werden kann, kann man nun das Screenmanagerprogramm dazu veranlassen, den Screen wieder zu schließen, wenn die Konfiguration freigegeben wird.

1.27 WRITE.guide/Konstantenbeschreibung

Konstantenbeschreibung
****

  TRUE
  FALSE
  LOWER
  HIGHER
  EQUAL

Weitere wichtige Konstanten als Platzhalter für Textpositionen.
Sie können immer dann verwandt werden, wenn Funktionen Positionsangaben im Text (in Form von x,y oder auch nur y-Koordinaten) erwarten. Alle diese Konstanten werden im Editor mit @Konstantenname verwendet.

  CURSOR
  MARKA
  MARKB
  SOL
  EOL
  SOW
  EOW
  SOT
  EOT
  SOP
  EOP
  SOS
  MOS
  EOS
  LEFT
  RIGHT

1.28 WRITE.guide/TRUE

TRUE
====

Schreiben   : Nein

Beschreibung: Ist immer 1. Dient dazu Variablen, die das Verhalten eines Schalters haben (0 = AUS, #0 = AN), verständlich einen Wert zuzuweisen.

1.29 WRITE.guide/FALSE

```
FALSE
=====

Schreiben   : Nein

Beschreibung: Ist immer 0.  Dient dazu Variablen, die das Verhalten eines
Schalters haben (0 = AUS, #0 = AN), verständlich einen Wert zuzuweisen.
```

1.30 WRITE.guide/LOWER

```
LOWER
=====

Schreiben   : Nein

Beschreibung: Ist immer 1.  Möglicher Rückgabewert der Funktion Compare.
```

1.31 WRITE.guide/HIGHER

```
HIGHER
======

Schreiben   : Nein

Beschreibung: Ist immer 2.  Möglicher Rückgabewert der Funktion Compare.
```

1.32 WRITE.guide/EQUAL

```
EQUAL
=====

Schreiben   : Nein

Beschreibung: Ist immer 0.  Möglicher Rückgabewert der Funktion Compare.
```

1.33 WRITE.guide/CURSOR

```
CURSOR
======

Schreiben   : Nein
```

Beschreibung: Steht die x-, bzw. y-Koordinaten des Cursors.

1.34 WRITE.guide/MARKA

MARKA
=====

Schreiben : Nein

Beschreibung: Steht für die Position der Blockanfangsmarke. Nur gültig,
wenn eine entsprechende Marke gesetzt wurde. Siehe auch _Marked

1.35 WRITE.guide/MARKB

MARKB
=====

Schreiben : Nein

Beschreibung: Steht für die Position der Blockendemarke. Nur gültig,
wenn eine entsprechende Marke gesetzt wurde. Siehe auch _Marked

1.36 WRITE.guide/SOL

SOL
===

Schreiben : Nein

Beschreibung: Steht für die x-,y-Koordinaten des Zeilenanfanges der
aktuellen Zeile. Identisch mit(1,_YPos).

1.37 WRITE.guide/EOL

EOL
===

Schreiben : Nein

Beschreibung: Steht für die x-,y-Koordinaten des Zeilenendes der
aktuellen Zeile. Identisch mit(Länge der aktuellen Zeile+1,_YPos).

1.38 WRITE.guide/SOW

```
SOW
===

Schreiben   : Nein

Beschreibung: Steht für die x-,y-Koordinaten des Anfanges des aktuellen
Wortes. Nur gültig, wenn Cursor auf einem Wort steht. Siehe auch
_CurrentWord.
```

1.39 WRITE.guide/EOW

```
EOW
===

Schreiben   : Nein

Beschreibung: Steht für die x-,y-Koordinaten des Endes des aktuellen
Wortes. Nur gültig, wenn Cursor auf einem Wort steht. Siehe auch
_CurrentWord.
```

1.40 WRITE.guide/SOT

```
SOT
===

Schreiben   : Nein

Beschreibung: Steht für die x-,y-Koordinaten des Textanfanges. Identisch
mit (1,1).
```

1.41 WRITE.guide/EOT

```
EOT
===

Schreiben   : Nein

Beschreibung: Steht für die x-,y-Koordinaten des Textendes. Identisch mit
(Länge der letzten Zeile+1,_Length).
```

1.42 WRITE.guide/SOP

```
SOP
===

Schreiben   : Nein

Beschreibung: Steht für die x-,y-Koordinaten des Anfanges des aktuellen
Paragraphen. Paragraphen sind durch Leerzeilen getrennte Textblöcke.
```

1.43 WRITE.guide/EOP

```
EOP
===

Schreiben   : Nein

Beschreibung: Steht für die x-,y-Koordinaten des Endes des aktuellen
Paragraphen. Paragraphen sind durch Leerzeilen getrennte Textblöcke.
```

1.44 WRITE.guide/SOS

```
SOS
===

Schreiben   : Nein

Beschreibung: Steht für die x-,y-Koordinaten der oberen, linken Ecke des
aktuellen Fensters.
```

1.45 WRITE.guide/MOS

```
MOS
===

Schreiben   : Nein

Beschreibung: Steht für die x-,y-Koordinaten der Mitte des aktuellen
Fensters.
```

1.46 WRITE.guide/EOS

```
EOS
===

Schreiben   : Nein

Beschreibung: Steht für die x-,y-Koordinaten der unteren, rechten Ecke
des aktuellen Fensters.
```

1.47 WRITE.guide/LEFT

```
LEFT
====

Schreiben   : Nein

Beschreibung: Steht für die x-,y-Koordinaten des Zeichens links vom
Cursor.  Position ist nicht gültig, wenn der Cursor am Anfang der Zeile
steht.
```

1.48 WRITE.guide/RIGHT

```
RIGHT
=====

Schreiben   : Nein

Beschreibung: Steht für die x-,y-Koordinaten des Zeichens rechts vom
Cursor.
```

1.49 WRITE.guide/Variablebeschreibung

```
Variablebeschreibung
********************


  _AColor
  _BColor
  _CColor
  _DColor
  _ReadTabs
  _WriteTabs
  _CaseSense
  _Optimize
  _XPos
  _YPos
  _Changed
```

```
_FileName
_FindString
_ReplaceString
_Length
_RS
_RN
_WBStarted
_LoadTab
_SaveTab
_Version
_ChipMem
_FastMem
_PublicMem
_Wins
_Time
_Path
_FRPattern
_PatCase
_ConfigPath
_ShowSpace
_ShowEOL
_PatNoCase
_CurrentChar
_CurrentWord
_CurrentLine
_TabLength
_REXX
_WordDef
_WriteIcon
_AppIcon
_AppWindow
_AppMenu
_Cursor
_AutoIndent
_SleepMode
_EditMode
_Marked
_MarkAx
_MarkAy
_MarkBx
_MarkBy
_WinMode
_CurrentID
_VersionString
_File
_FilePath
_REXXPortName
_REG1
_REG2
_DefaultTool
_Language
_CurrentConfig
_DefaultConfig
_AutoFree
_WinWidth
_WinHeight
_WordWrap
```

```
  _RightMargin
  _ScreenWidth
  _ScreenHeight
  _Undo
  _ScrRelWidth
  _ScrRelHeight
  _WordOnly
  _ReqToMouse
  _EColor
  _FColor
  _AutoFold
  _FoldStart
  _FoldEnd
  _DelFoldMark
  _Fold
  _Screenname
  _PathPath
  _User
  _CollectorMem
  _StdNumPad
  _InsertMode
  _OverwriteCursor
  _ReadSpeed
  _UndoMem
  _CurrentUndoMem
```

1.50 WRITE.guide/_AColor

```
_AColor
=======

Schreiben   : Ja

Beschreibung: Setzt die aktuelle Vordergrundfarbe (Farbe der Schrift) auf
die entsprechende Farbnummer (0..MAXCOLOR-1).  Die Farben werden anhand
der Reihenfolge im Palletterequester von 0 an aufwärts durchnummeriert.
```

1.51 WRITE.guide/_BColor

```
_BColor
=======

Schreiben   : Ja

Beschreibung: Setzt den aktuellen Hintergrund auf die entsprechende
Farbnummer (0 ... MAXCOLOR-1).  Die Farben werden anhand der Reihenfolge
im Palletterequester von 0 an aufwärts durchnummeriert.
```

1.52 WRITE.guide/_CColor

```
_CColor
=======

Schreiben   : Ja

Beschreibung: Setzt die aktuelle Vordergrundfarbe (Farbe der Schrift)
für markierten Text auf die entsprechende Farbnummer (0..MAXCOLOR-1).
Die Farben werden anhand der Reihenfolge im Palletterequester von 0 an
aufwärts durchnummeriert.
```

1.53 WRITE.guide/_DColor

```
_DColor
=======

Schreiben   : Ja

Beschreibung: Setzt die aktuelle Hintergrundfarbe für markierten Text
auf die entsprechende Farbnummer (0..MAXCOLOR-1).  Die Farben werden
anhand der Reihenfolge im Palletterequester von 0 an aufwärts
durchnummeriert.
```

1.54 WRITE.guide/_ReadTabs

```
_ReadTabs
=========

Schreiben   : Ja

Beschreibung: Variable , die angibt, ob beim Einlesen eines Files
gefundene Tabulatoren in die entsprechende Anzahl von Spaces umgewandelt
werden sollen.  0=NEIN, ansonsten JA.
```

1.55 WRITE.guide/_WriteTabs

```
_WriteTabs
==========

Schreiben   : Ja

Beschreibung: Variable , die angibt, ob beim Schreiben eines Files Spaces
in Tabulatoren umgewandelt werden sollen.  Die Files werden dadurch
kürzer.  Doch nicht alle Programme konvertieren Tabulatoren wieder
richtig zurück.  0=NEIN, ansonsten JA.
```

1.56 WRITE.guide/_CaseSense

```
_CaseSense
==========

Schreiben   : Ja

Beschreibung: Variable, die angibt, ob beim Suchen ein Unterschied
zwischen Groß- und Kleinschreibung gemacht werden soll.  0=NEIN,
ansonsten JA.
```

1.57 WRITE.guide/_Optimize

```
_Optimize
=========

Schreiben   : Ja

Beschreibung: Wenn gesetzt, werden beim Abspeichern Spaces hinter dem
letzten Buchstaben einer Zeile gelöscht.  0=NEIN, ansonsten JA.
```

1.58 WRITE.guide/_XPos

```
_XPos
=====

Schreiben   : Nein

Beschreibung: Gibt aktuelle Spalte, in der sich der Cursor befindet, an.
```

1.59 WRITE.guide/_YPos

```
_YPos
=====

Schreiben   : Nein

Beschreibung: Gibt aktuelle Spalte, in der sich der Cursor befindet, an.
```

1.60 WRITE.guide/_Changed

_Changed
========

Schreiben   : Ja

Beschreibung: Wenn ungleich 0, ist der Text seit dem letzten Abspeichern
verändert worden.

1.61 WRITE.guide/_FileName

_FileName
=========

Schreiben   : Ja

Beschreibung: Beinhaltet den Namen des Textes im aktuellen Ed.

1.62 WRITE.guide/_FindString

_FindString
===========

Schreiben   : Ja

Beschreibung: Zeichenfolge, nach der Find suchen soll.

1.63 WRITE.guide/_ReplaceString

_ReplaceString
==============

Schreiben   : Ja

Beschreibung: Zeichenfolge, die Replace für ein Vorkommen von
_FindString einsetzten soll.

1.64 WRITE.guide/_Length

_Length
=======

Schreiben   : Nein

Beschreibung: Länge des Textes in Zeilen.

1.65 WRITE.guide/_RS

_RS
===

Schreiben : Nein

Beschreibung: Variable, in die Funktionen eine Variable vom Typ
Zeichenkette zurückgeben.

1.66 WRITE.guide/_RN

_RN
===

Schreiben : Nein

Beschreibung: Numerischer Rückgabewert von Funktionen.

1.67 WRITE.guide/_WBStarted

_WBStarted
==========

Schreiben : Nein

Beschreibung: Ist _WBStarted ungleich null, so ist WRITE von der
Workbench gestartet worden, ansonsten aus einer Shell.

1.68 WRITE.guide/_LoadTab

_LoadTab
========

Schreiben : Ja

Beschreibung: Umrechnungsfaktor für die Konvertierung von Tabulatoren zu
Spaces beim Laden eines Textes. Im Zusammenhang mit _SaveTab können
Texte durch Laden und anschließendes Abspeichern von einer
Tabulatorgröße zu einer anderen konvertiert werden.

1.69 WRITE.guide/_SaveTab

```
_SaveTab
========

Schreiben   : Ja

Beschreibung: Umrechnungsfaktor für die Konvertierung von Tabulatoren zu
Spaces beim Speichern eines Textes.  Im Zusammenhang mit _LoadTab können
Texte durch Laden und anschließendes Abspeichern von einer
Tabulatorgröße zu einer anderen konvertiert werden.
```

1.70 WRITE.guide/_Version

```
_Version
========

Schreiben   : Nein

Beschreibung: Gibt die Kompatibilitätsversion von WRITE in Form einer
Zahl an.
```

1.71 WRITE.guide/_ChipMem

```
_ChipMem
========

Schreiben   : Nein

Beschreibung: Momentan verfügbarer CHIP/Graphik-Speicher.
```

1.72 WRITE.guide/_FastMem

```
_FastMem
========

Schreiben   : Nein

Beschreibung: Momentan verfügbarer FAST-Speicher.
```

1.73 WRITE.guide/_PublicMem

```
_PublicMem
==========

Schreiben   : Nein

Beschreibung: Momentan verfügbarer Gesamt-Speicher.
```

1.74 WRITE.guide/_Wins

```
_Wins
=====

Schreiben   : Nein

Beschreibung: Anzahl des momentan verfügbaren CHIP/Graphik-Speichers.
```

1.75 WRITE.guide/_Time

```
_Time
=====

Schreiben   : Nein

Beschreibung: Die aktuelle Zeit in Form eines String, der Tagesnamen
Datum und Zeit beinhaltet.
```

1.76 WRITE.guide/_Path

```
_Path
=====

Schreiben   : Ja

Beschreibung: Der Pfad des Filerequesters.
```

1.77 WRITE.guide/_FRPattern

```
_FRPattern
==========

Schreiben   : Ja

Beschreibung: Das Auswahlpattern für den Filerequester.
```

1.78 WRITE.guide/_PatCase

```
_PatCase
========

Schreiben   : Ja

Beschreibung: Pattern für die Funktion FindPattern.
```

1.79 WRITE.guide/_ConfigPath

```
_ConfigPath
===========

Schreiben   : Ja

Beschreibung: Das Verzeichnis, in dem WRITE alle Konfigurationsdateien
sucht.
```

1.80 WRITE.guide/_ShowSpace

```
_ShowSpace
==========

Schreiben   : Ja

Beschreibung: Wenn #0, werden Spaces durch das Zeichen · angezeigt.
```

1.81 WRITE.guide/_ShowEOL

```
_ShowEOL
========

Schreiben   : Ja

Beschreibung: Wenn #0, wird das Ende einer Zeile im Text durch das
Zeichen ¶ angezeigt.
```

1.82 WRITE.guide/_PatNoCase

```
_PatNoCase
==========

Schreiben   : Ja

Beschreibung: Pattern für die Funktion FindPattern.
```

1.83 WRITE.guide/_CurrentChar

```
_CurrentChar
============

Schreiben   : Nein

Beschreibung: ASCII-Code des Zeichens unter dem Cursor.  0 wenn kein
Zeichen.
```

1.84 WRITE.guide/_CurrentWord

```
_CurrentWord
============

Schreiben   : Nein

Beschreibung: Das Wort unter dem Cursor.  Definition eines Wortes ist
abhängig vom Inhalt der Variable _WordDef.
```

1.85 WRITE.guide/_CurrentLine

```
_CurrentLine
============

Schreiben   : Ja

Beschreibung: Inhalt der aktuellen Zeile.
```

1.86 WRITE.guide/_TabLength

```
_TabLength
==========

Schreiben   : Ja
```

Beschreibung: Abstand zwischen Tabulatoren während des Editierens. Wird von den Funktionen Tab und BackTab benutzt.

1.87 WRITE.guide/_REXX

_REXX
=====

Schreiben : Nein

Beschreibung: Wenn #0, so ist REXX installiert und es können REXX-Sripte gestartet werden. (WRITE kontrolliert allerdings nur das Vorhandensein von LIBS:rexxsyslib.library.)

1.88 WRITE.guide/_WordDef

_WordDef
========

Schreiben : Ja

Beschreibung: Definiert, was WRITE unter einem Wort zu verstehen hat. Ist _WordDef=0, dann besteht ein Wort nur aus Buchstaben, d.h. a-z, A-Z, _, Umlaute etc. Ist _WordDef=1, werden Wörter durch sogenannte Trennzeichen getrennt. Trennzeichen sind: , ., ,, ;, : , !, ?, ', `, (,) und ". Ist _WordDef=2, so werden Wörter ausschließlich durch Spaces getrennt.

1.89 WRITE.guide/_WriteIcon

_WriteIcon
==========

Schreiben : Ja

Beschreibung: Wenn 1, so wird beim Abspeichern auch ein entsprechendes Icon kreiert, falls kein Icon vorhanden ist. Ist _WriteIcon=2, so wird ein neues Icon erzeugt, wenn noch keines vorhanden ist. Ist bereits eines vorhanden, so wird dieses überschrieben.

1.90 WRITE.guide/_AppIcon

_AppIcon
========

Schreiben : Ja

Beschreibung: Wenn #0, so wird auf der Workbench ein AppIcon installiert,
über welches der WinManager aufgerufen und Texte geladen werden können.

1.91 WRITE.guide/_AppWindow

_AppWindow
==========

Schreiben : Ja

Beschreibung: Wenn #0, so wird jedes Editorfenster auch zu einem
AppWindow. D.h. mit dem Ziehen von Workbenchicons in dieses Fenster,
können diese geladen werden.

1.92 WRITE.guide/_AppMenu

_AppMenu
========

Schreiben : Ja

Beschreibung: Wenn #0, wird im Hilfsmittelmenü der Workbench ein
Menüpunkt installiert, über den man den WinManager aufrufen und
selektierte Texte laden kann.

1.93 WRITE.guide/_Cursor

_Cursor
=======

Schreiben : Ja

Beschreibung: Je nach Wert wird der Cursor auf verschiedene Weise
dargestellt.

1.94 WRITE.guide/_AutoIndent

```
_AutoIndent
===========

Schreiben   : Ja

Beschreibung: AutoIndent heißt, daß nach einem Return der Cursor
automatisch nicht am Anfang der Zeile, sondern unter dem ersten
Buchstaben der Zeile darüber steht.  Dies ist besonders praktisch, wenn
man mit WRITE Programme schreibt, da hier besonders oft mit
Texteinrückungen gearbeitet wird.  _AutoIndent  sagt WRITE nun, wie wie
viele Leerzeilen er rückwärts suchen soll, um nach dem Indent, dem Maß
der Einrückung, zu suchen.  Beispiel: Der Cursor steht an der Position %.
        Hier steht ein Text!!!


                %
     1  2
Ist _AutoIndent größer als 2, so steht der Cursor nach einem Return an
der Position 2, ansonsten an der Position 1.  Ist _AutoIndent=0, so ist
dieses Feature abgeschaltet.
```

1.95 WRITE.guide/_SleepMode

```
_SleepMode
==========

Schreiben   : Ja

Beschreibung: Ist diese Variable ungleich null, so wird WRITE nicht
beendet, wenn der letzte Ed geschlossen worden ist.  Dieser Modus wird
empfohlen und sollte immer eingeschaltet sein, wenn es der Speicherplatz
zuläßt.
```

1.96 WRITE.guide/_EditMode

```
_EditMode
=========

Schreiben   : Ja

Beschreibung: Wenn#0, kann der Text auch verändert werden.  Gilt für
alle Fenster!
```

1.97 WRITE.guide/_Marked

```
_Marked
=======

Schreiben   : Nein

Beschreibung: Ist _Marked=0, so ist weder eine Blockmarke gesetzt, noch
ein Block markiert. Ist _Marked=1, so ist eine Blockmarke gesetzt und ist
_Marked=2, so ist ein Block markiert worden.
```

1.98 WRITE.guide/_MarkAx

```
_MarkAx
=======

Schreiben   : Nein

Beschreibung: Spaltennummer der 1. Blockmarke.  0, wenn keine Marke
gesetzt.
```

1.99 WRITE.guide/_MarkAy

```
_MarkAy
=======

Schreiben   : Nein

Beschreibung: Zeilennummer der 1. Blockmarke.  0, wenn keine Marke
gesetzt.
```

1.100 WRITE.guide/_MarkBx

```
_MarkBx
=======

Schreiben   : Nein

Beschreibung: Spaltennummer der 2. Blockmarke.  0, wenn keine Marke
gesetzt.
```

1.101 WRITE.guide/_MarkBy

```
_MarkBy
=======

Schreiben   : Nein

Beschreibung: Zeilennummer der 2. Blockmarke.  0, wenn keine Marke
gesetzt.
```

1.102 WRITE.guide/_WinMode

```
_WinMode
========

Schreiben   : Ja

Beschreibung: Hier kann ausgelesen werden, ob der aktuelle Ed ein Fenster
geöffnet hat (0) (Window), ikonifiziert worden ist(1) (Iconify), kein
Fenster offen hat  (2) (Hide) oder schließlich ein Fenster offen hat,
aber die graphische Ausgabe unterdrückt wird (3) (Silent).
```

1.103 WRITE.guide/_CurrentID

```
_CurrentID
==========

Schreiben   : Nein

Beschreibung: ID des aktuellen Eds. Ist der ID 0, so ist kein Ed
aktiviert.
```

1.104 WRITE.guide/_VersionString

```
_VersionString
==============

Schreiben   : Nein

Beschreibung: Versionsstring von WRITE.
```

1.105 WRITE.guide/_File

```
_File
=====

Schreiben   : Nein

Beschreibung: Dateiname des aktuellen Textes, ohne kompletten Pfad.
```

1.106 WRITE.guide/_FilePath

```
_FilePath
=========

Schreiben   : Nein

Beschreibung: Directory des aktuellen Textes.
```

1.107 WRITE.guide/_REXXPortName

```
_REXXPortName
=============

Schreiben   : Nein

Beschreibung: Name des globalen REXX-Ports von WRITE.
```

1.108 WRITE.guide/_REG1

```
_REG1
=====

Schreiben   : Ja

Beschreibung: Variable, die für die eigene Verwendung freigestellt ist.
```

1.109 WRITE.guide/_REG2

```
_REG2
=====

Schreiben   : Ja

Beschreibung: Variable, die für die eigene Verwendung freigestellt ist.
```

1.110 WRITE.guide/_DefaultTool

```
_DefaultTool
============

Schreiben   : Ja

Beschreibung: Bestimmt, was als DefaultTool in Icons abgespeichert werden
soll.  Voreingestellt ist WRITE (inclusive kompletten Pfad) selbst.
```

1.111 WRITE.guide/_Language

```
_Language
=========

Schreiben   : Nein

Beschreibung: Hier steht, auf welche Sprache WRITE eingestellt ist.
```

1.112 WRITE.guide/_CurrentConfig

```
_CurrentConfig
==============

Schreiben   : Nein

Beschreibung: Der Name der aktuellen Konfiguration. Ist er "", so ist
keine Konfiguration aktiviert. Eine Konfiguration läßt sich mit
GetConfig aktivieren.
```

1.113 WRITE.guide/_DefaultConfig

```
_DefaultConfig
==============

Schreiben   : Ja

Beschreibung: Der Name der Konfiguration, die immer dann benutzt wird,
wenn explizit eine bestimmte Konfiguration vorgegeben wird.
```

1.114 WRITE.guide/_AutoFree

```
_AutoFree
=========

Schreiben   : Ja

Beschreibung: Ist diese Variable ungleich Null, so wird eine
Konfiguration, die von keinem Ed mehr gebraucht wird, automatisch
freigegeben.
```

1.115 WRITE.guide/_WinWidth

```
_WinWidth
=========

Schreiben   : Nein

Beschreibung: In dieser Variable steht die Breite des aktuellen Fensters
in Zeichen.
```

1.116 WRITE.guide/_WinHeight

```
_WinHeight
==========

Schreiben   : Nein

Beschreibung: In dieser Variable steht die Höhe des aktuellen Fensters
in Zeichen.
```

1.117 WRITE.guide/_WordWrap

```
_WordWrap
=========

Schreiben   : Ja

Beschreibung: _WordWrap steuert das WordWrapping von WRITE.  Ist
_WordWrap=0, so gibt es kein WordWrap. Ist _WrdWrap=1, so ertönt ein
Beepen, wenn der Text einer Zeile über den in _RightMargin angegebenen
Rand geht. Dies ist ähnlich dem Verhalten einer Schreibmaschine.  Ist
_WordWrap=2, so bricht WRITE eine Zeile, wenn sie über den Rand geht,
automatisch um. Der Inhalt der Variable _AutoIndent wird dabei beachtet.
Ist _WordWrap=3 so wird nicht nur gewrappt, wenn Sie Zeichen einfügen,
sondern auch wenn Sie welche löschen.
```

1.118 WRITE.guide/_RightMargin

```
_RightMargin
============

Schreiben   : Ja

Beschreibung: Gibt den rechten Rand für die WordWrap-Funktion
(_WordWrap) an.
```

1.119 WRITE.guide/_ScreenWidth

```
_ScreenWidth
============

Schreiben   : Nein

Beschreibung: Breite des aktuellen Screens.
```

1.120 WRITE.guide/_ScreenHeight

```
_ScreenHeight
=============

Schreiben   : Nein

Beschreibung: Höhe des aktuellen Screens.
```

1.121 WRITE.guide/_Undo

```
_Undo
=====

Schreiben   : Ja

Beschreibung: Gibt die Größe des Undo-Puffers an. WRITE merkt sich alle
Veränderungen am Text. Undo gibt nun an, wie viele Veränderungen er
sich merken soll. Ist _Undo=0, so ist die Undo-Funktion ausgeschaltet.
```

1.122 WRITE.guide/_ScrRelWidth

_ScrRelWidth
============

Schreiben   : Ja

Beschreibung: Wird beim Aufruf des Befehls Window das Tag @SCREENREL
übergeben, so beziehen sich alle Größenangaben nicht auf die reale
Screengröße, sondern auf den hier angegebenen Wert. Dies ist zum
Beispiel nützlich, wenn ein Script WINDOW 0 0 0 0 zum Öffnen eines
bildschirmfüllenden Fensters benutzt. Auf großen Bildschirmen ist dies
aber nicht unbedingt erwünscht. Mit dieser Variablen kann man nun einen
Screen der angegebenen Größe vortäuschen.

1.123 WRITE.guide/_ScrRelHeight

_ScrRelHeight
=============

Schreiben   : Ja

Beschreibung: Wird beim Aufruf des Befehls Window das Tag @SCREENREL
übergeben, so beziehen sich alle Größenangaben nicht auf die reale
Screengröße, sondern auf den hier angegebenen Wert. Dies ist zum
Beispiel nützlich, wenn ein Script WINDOW 0 0 0 0 zum Öffnen eines
bildschrimfüllenden Fensters benutzt. Auf großen Bildschirmen ist dies
aber nicht unbedingt erwünscht. Mit dieser Variablen kann man nun einen
Screen der angegebenen Größe vortäuschen.

1.124 WRITE.guide/_WordOnly

_WordOnly
=========

Schreiben   : Ja

Beschreibung: Variable, die angibt, ob beim Suchen nur nach ganzen
Wörtern oder auch nach Wortteilen gesucht werden soll. 0=NEIN, ansonsten
JA. Die Definition eines Wortes hängt von dem Inhalt der Variable
_WordDef ab.

1.125 WRITE.guide/_ReqToMouse

_ReqToMouse
===========

Schreiben   : Ja

Beschreibung: Variable, die angibt, ob bei internen Requesteraufrufen,
diese unter dem Mauszeiger geöffnet werden. Siehe auch Requester

1.126 WRITE.guide/_EColor

_EColor
=======

Schreiben : Ja

Beschreibung: Eine Falte wird durch zwei horizontale Linien angezeigt.
Durch diese Variable kann man die Farbe der oberen bestimmen.

1.127 WRITE.guide/_FColor

_FColor
=======

Schreiben : Ja

Beschreibung: Eine Falte wird durch zwei horizontale Linien angezeigt.
Durch diese Variable kann man die Farbe der unteren bestimmen.

1.128 WRITE.guide/_AutoFold

_AutoFold
=========

Schreiben : Ja

Beschreibung: Folds werden beim Abspeichern im Text durch die Markierungen
_FoldStart und _FoldEnd angezeigt. Ist _AutoFold TRUE, so wird nach dem
Einladen eines Textes automatisch nach diesen Markierungen gesucht und
diese wieder zu Falten konvertiert.

1.129 WRITE.guide/_FoldStart

_FoldStart
==========

Schreiben : Ja

Beschreibung: Folds werden beim Abspeichern im Text durch die Markierungen angezeigt. FoldStart zeigt den Anfang einer Falte, _FoldEnd das Ende einer Falte an. Diese Markierungen können durch den Benutzer frei definiert werden, damit gewährleistet werden kann, daß diese Markierungen z.B. einen Compiler nicht stören. C Programmierer würden z.B. /*S*/ und /*E*/ wählen, Oberon Programmierer (*S*) und (*E*), Leute, die mit TeX arbeiten, einfach /%S und %E.

1.130 WRITE.guide/_FoldEnd

_FoldEnd
========

Schreiben : Ja

Beschreibung: Folds werden beim Abspeichern im Text durch die Markierungen angezeigt. _FoldStart zeigt den Anfang einer Falte, _FoldEnd das Ende einer Falte an. Diese Markierungen können durch den Benutzer frei definiert werden, damit gewährleistet werden kann, daß diese Markierungen z.B. einen Compiler nicht stören. C Programmierer würden z.B. /*S*/ und /*E*/ wählen, Oberon Programmierer (*S*) und (*E*), Leute, die mit TeX arbeiten, einfach /%S und %E.

1.131 WRITE.guide/_DelFoldMark

_DelFoldMark
============

Schreiben : Ja

Beschreibung: Folds werden beim Abspeichern im Text durch die Markierungen angezeigt. _FoldStart zeigt den Anfang einer Falte, _FoldEnd das Ende einer Falte an. Werden diese Faltenmarkierungen anschließend durch das _AutoFold-Feature oder durch die Funktion ReFold in Falten umgewandelt, kann es erwünscht sein, daß die Faltenmarkierungen gelöscht werden. Ist die Variable DelFoldMark ungleich TRUE, so geschieht dies. Nach einem anschließenden Entfalten sind die Markierungen wieder verschwunden.

1.132 WRITE.guide/_Fold

_Fold
=====

Schreiben : Nein

Beschreibung: Ist diese Variable ungleich 0, so handelt es sich bei der aktuellen Zeile um eine Falte. Bevor man auf den Inhalt der aktuellen

Zeile zugreifen will, sollte man immer diese Variable kontrollieren.

1.133 WRITE.guide/_Screenname

_Screenname
==========

Schreiben : Nein

Beschreibung: Names des Screens, auf den die aktuelle Konfiguration ihre Fenster öffnet.

1.134 WRITE.guide/_PathPath

_PathPath
=========

Schreiben : Nein

Beschreibung: Der Pfadanteil des Filerequesterpfades.

1.135 WRITE.guide/_User

_User
=====

Schreiben : Nein

Beschreibung: Falls es sich um eine registrierte Version handelt, steht hier der Name des Besitzers.

1.136 WRITE.guide/_CollectorMem

_CollectorMem
=============

Schreiben : Nein

Beschreibung: Größe des Speichers, welcher vom GarbageCollector benutzt wird.

1.137 WRITE.guide/_StdNumPad

```
_StdNumPad
==========

Schreiben   : Ja

Beschreibung: Hiermit können alle Key-Definitionen des Nummeric Pads
abgeschaltet werden. Praktisch z.B., um schnell zwischen Cursorsteuerung
und Zahleneingabe über das Nummeric Pad zu wechseln.
```

1.138 WRITE.guide/_InsertMode

```
_InsertMode
===========

Schreiben   : Ja

Beschreibung: Hiermit kann zwischen dem Einfüge- und Überschreibmodus
umgeschaltet werden. Beachten sie, daß mittels _OverwriteCursor das
Aussehen des Cursors im Overwritemodus bestimmt werden kann.
```

1.139 WRITE.guide/_OverwriteCursor

```
_OverwriteCursor
================

Schreiben   : Ja

Beschreibung: Bestimmt das Aussehen des Cursors im Overwritemodus.
```

1.140 WRITE.guide/_ReadSpeed

```
_ReadSpeed
==========

Schreiben   : Ja

Beschreibung: Bestimmt die Anzeigedauer der Message-Funktion in
Abhängigkeit von der Textlänge. Die Anzeigedauer berechnet sich nach dr
Formel: Delay = (_ReadSpeed*Length(Text)) / 10. Delay hat dabei die
Einheit 1/50 Sekunde.
```

1.141 WRITE.guide/_UndoMem

```
_UndoMem
========

Schreiben   : Ja

Beschreibung: Gibt den maximalen SPeicherverbracuh für den Undobuffer
eines Editorfensters an.
```

1.142 WRITE.guide/_CurrentUndoMem

```
_CurrentUndoMem
===============

Schreiben   : Nein

Beschreibung: Gibt die aktuelle Größe des Undobuffers an.
```

1.143 WRITE.guide/Funktionsbeschreibung

```
Funktionenbeschreibung
**********************

Im folgenden werden die einzelnen Funktionen samt ihrer Parameter
beschrieben.  Der Typ eines Parameters wird dabei durch einen ihm (oder
einer Liste von Parametern getrennt durch ,) angehängtes Kürzel
bestimmt.

   *     /S steht hierbei für einen String. Beachten Sie, daß Zahlen
     zuweisungskompatibel zu Strings sind. Sie können also hier auch
     Parameter sein.

   *     /N steht für eine vorzeichenlose Zahl. Strings sind auch hier
     zuweisungskompatibel, doch jedoch nur dann, wenn Sie nur eine Zahl
     beinhalten. "123" kann zugewiesen werden, "Dash 3" nicht.

   *     /T steht für eine Liste von Tags. Diese muß mit { anfangen und
     mit } aufhören. Dazwischen steht eine beliebige Zahl von TAGS.
     Wollen Sie keine Tags übergeben, können die geschweiften Klammern
     ganz weggelassen werden.

   *     /F steht für eine Funktionsliste. Eine Funktionsliste ist eine
     Reihe von Funktionen sammt ihren Parametern (die wieder
     Funktionslisten sein können), die durch ein ; abgeschlossen werden.


Funktionen für Ed's und Fenster

  NOP              Diese Funktion macht gar nichts
```

```
  Open              Laden eines Textes
  Save              Speichert einen Text
  SetBackUp         Setzt die verschiedenen BackUpmodi
  New               Löscht aktuellen Text
  NewEd             Öffnet einen neuen Ed
  QuitEd            Schließt Ed
  Window            Öffnet Fenster für Ed
  Iconify           Iconfifiziert Ed
  Hide              Versteckt Ed
  Silent            Unterbindet Graphikausgabe in Fenster
  Font              Set neuen Font
  WinArranger       Organisiert offene Fenster
  SetTitle          Setzt FensterTitel
  Screen            Gibt Screen an, auf den Konfig Fenster öffnen soll

Requester

  About             Informationen über Version,Autor...
  Prefs             Änderung einiger konfigurationslokalen Variablen
  GPrefs            Änderung der globalen Variablen
  HelpFkt           Hilfsfunktion
  WinManager        Erleichtert Umgang mit Eds
  ShowVars          Zeigt Variablen
  ShowFunctions     Zeigt Funktionen
  ShowConstants     Zeigt Konstanten
  ShowASCII         Zeigt ASCII-Code aller Zeichen
  ShowIndex         Erstellt einen Index zu einem Suchmuster
  GetString         Requester für Stringeingabe
  GetNumber         Requester für Zahleneingabe
  GetFindReplace    Requester für die Eingabe von Suchwörtern
  GetFile           Der Filerequester
  GetFiles          Filerequester mit MultiSelect
  GetFont           Fontrequester
  Ask               Frage Requester mit definierbaren Gadgets
  Message           Zeigt kurz eine Meldung
  MessageOK         Zeigt Nachricht und erwartet Bestätigung
  Flash             Bildschirmblitz
  Beep              Beep

Funktionen für den internen Parser und das Betriebssystem

  ParseBuffer       Parsen eines Puffers
  DoBuffer          Führt Puffer aus
  DoString          Führt String aus
  PreparseString    Parsed String vor. Variablen etc. werden ersetzt
  SetUserFkt        Set eine UserFunktion
  Compare           Vergleicht zwei Werte
  If                Bedingte Abarbeitung
  Break             Bedingter Abbruch eine Anweisungsfolge
  SetError          Bricht Anweisungsfolge ab
  SetVar            Setzt eine Variable
  GetVar            Liest Variable
  SetEnv            Setzt Umgebungsvariabel
  GetEnv            Liest Umgebungsvariabel
  System            Führt DOS-Befehle etc. aus

Puffer- und Blockopperationen
```

SetMark	Setzt die Marken für Blockoperationen
Mark	Markieren eines Blockes
UnMark	Löscht die aktuelle Blockmarkierung
DeleteBlock	Löscht den markierten Block
CopyBlock	Kopiert den markierten Block
InsertBlock	Fügt einen Puffer an der Cursorposition ein
DeleteArea	Löscht angebaren Bereich
CopyArea	Copiert angebaren Bereich
SaveBuffer	Speichert ein Puffer ab
LoadBuffer	Läd einen Puffer
ClearBuffer	Löscht einen Puffer
BufferToStr	Konvertiert Buffer zu String
StrToBuffer	Konvertiert String zu Buffer
ClipToBuffer	Kopiert Text im clipboard-device in einen Puffer
BufferToClip	Kopiert einen Puffer ins clipboard-device
BlockLeft	Verschiebt Block nach Links
BlockRight	Verschiebt Block nach Rechts
BlockLftAlig	Macht Block linksbündig
BlockRghtAlig	Macht Block rechtsbündig
BlockCenter	Zentriert einen Block

Befehle für die Bewegung im Text

CursorUp	Bewegt den Cursor 1 hoch
CursorDown	Bewegt Cursor 1 runter
CursorRight	Bewegt Cursor 1 rechts
CursorLeft	Bewegt Cursor 1 links
NextWord	Springt zum Anfang des nächsten Wortes
LastWord	Springt zum Anfang des letzten Wortes
PageUp	Springt eine Seite nach oben
PageDown	Springt eine Seite nach unten
ScrollUp	Scrollt den Text unter dem Cursor
ScrollDown	Scrollt den Text unter dem Cursor
Goto	Springt zu einer angegebenen Zeile/Spaltenposition
GotoMouse	Positioniert Cursor unter der Maus
SetTextMark	Setzt eine TextMarke
GoTextMark	Springt zu einer TextMarke

Suchen und Finden

Find	Springt zum gesuchten Text
Replace	Ersetzt Suchstring durch anderen String
FindPattern	Sucht Text nach einen Muster
ReplaceList	Such und ersetzt den Inhalt einer Liste
MatchBracket	Sucht die entsprechende 2. Klammer

Befehle für die Texteditierung

Return	Zeilenumbruch
Delete	Löscht Zeichen unter dem Cursor
DeleteToEOL	Löscht vom Cursor bis Ende der Zeile
DeleteLine	Löscht ganze Zeile
UnDelLine	Setzt mit DeleteLine gelöschte Zeile wieder ein
BackSpace	Einen nach links und dann löschen
Tab	Springt zum nächstem Tab

```
  BackTab          Springt zum letzten Tab
  UpperBlock       Wandelt einen Textbereich in Großbuchstaben um
  LowerBlock       Wandelt einen Textbereich in Kleinbuchstaben um
  WriteChar        Schreibt in Zeichen
  WriteText        Schreibt Zeichenkette
  SwapChar         Vertauscht zwei Zeichen

Menüs und Tastaturbelegung

  Key              Belegt eine Taste mit einer Befehlsfolge
  DoubleKey        Belegt eine Doppellick Tastenkombination
  ClearKeys        Löscht die Tastaturbelegung
  SetHotKey        Definiert einen HotKey mit einer Befehlsfolge
  ClearHotKey      Löscht den Hotkey wieder
  Menu             Neues Menü
  Item             Definiert einen Menüeintrag
  Sub              Definiert einen Untermenüeintrag
  ItemBar          Macht in einem Menü einen Unterteilungsstrich
  SubBar           Macht in einem Untermenü einen Unterteilungsstrich
  ClearMenu        Löscht die Menübelegung
  WaitPointer      Erzeugt eien Wartemauszeiger
  NormalPointer    Normaler Mauszeiger

REXX

  DoREXX           Führt ein REXX-Script aus
  LockWindow       Alle REXX-Komandos gehen an das markierte Fenster
  NextEd           Aktierviert einen bestimmten Ed
  OpenPort         Öffnet einen REXX-Port für ein Fenster
  ClosePort        Schließt diesen
  WaitPort         Wartet bis das Fenster geschlossen wird
  ModifyWin        Verändert Editorfenster
  ModifyScreen     Toggle Screen nach vorne und hinten
  SetREXXClip      Kopiert den Inhalt einer Zeile/Puffer in das REXXClipboard
  GetConfig        Aktiviert eine bestimmte Konfiguration
  SetREXXVar       Setzt eine REXX-Variablen mit den Inhalt einer Zeile etc.
  GetREXXVar       Liest den Inhalt einer REXX-Variablen nach _RS
  Refresh          Aktuallisiert das Fenster
  ChangeConfig     Ändert die Konfiguration eines bestehenden Eds
  SaveGConfig      Speichert globale Konfiguration ab.
  SaveConfig       Speichert lokale Konfiguration in einen File ab.

TASTATUR-MAKROS

  MacroRec         Startet die Aufzeichnung eines Makros
  MacroStop        Stopt die Aufzeichnung eines Makros
  MacroPlay        Spielt ein Makro ab

PANNEL

  SetPannel        Belegt ein Gadgets des Pannels
  Pannel           Zeigen eines ButtonPannels auf das wichtige Fkt gelegt werden ↩
     können

MAKROS
```

```
  Macro           Definiert ein Macro, welches wie eine normale Funktion benutzt ↩
     werden kann.

UNDO

  Undo            Macht Veränderungen wieder Rückgängig

LISTEN

  ClearList       Löscht eine Liste
  AddList         Hängt einen Eintrag an eine Liste
  RemoveList      Löscht einen Listeneintrag
  Push            Schiebt Eintrag auf die erste Position d. Liste
  Pop             1. Eintrag nach _RS und aus Liste löschen
  ShowList        Zeigt Liste/Selektieren aus Liste
  ListToBuffer    Kopiert eine Liste in einen Buffer
  BufferToList    Macht aus jeder Zeile eines Buffers einen Listeneintrag
  DoList          Führt Funktion mit jedem Listenelement aus
  ListSize        Gibt die Größe einer Liste zurück
  GetListEntry    Gibt bestimmten Eintrag einer Liste zurück
  FindListEntry   Such bestimmten Eintrag und gibt dessen position zurück

FILES

  Exists          Schaut, ob ein File existiert
  Delay           Wartet eine eintstellbare Zeit

HELP

  GuideHelp       Ruft AmigaGuide mit bestimmten Stichwort auf

  VersionCheck    Kontrolliert, ob WRITE zur angegebenen Version kompatibel ist

  Inc             Addiert zu einer Zahl eine weitere Zahl
  Dec             Subtrahiert zwei Zahlen voneinander
  RangeCheck      Kontrolliert, ob Zahl im angegebenen Interval liegt
  RangeRound      Vergrößert, verkleiner Zahl, daß sie im angegebenen Intervall  ↩
     liegt

  Begin           Wird beim Starten v. WRITE ausgeführt
  Close           Wird beim Schließen v. WRITE ausgeführt
  Start           Wird beim Laden einer Konfiguration ausgeführt
  Quit            Wird beim Beenden einer Konfiguration ausgeführt

TEXTFALTEN

  Fold            Falten einen bestimmten Textbereich
  UnFold          Entfaltet alle Falten in einem bestimmten Textbereich
  AutoFold        Konvertiert all Faltenmarkierungen zu Falten
  ReFold          Schaut, ob Cursor zwischen zwei Faltenmarkierungen steht und  ↩
     konvertiert diese zu einer Falte
```

1.144 WRITE.guide/NOP

```
NOP
===

Aufruf      : NOP

Benötigt    :

Setzt Fehler:

Ergebnisse  :

Beschreibung: NOP steht für 'no operation'.  Diese Funktion macht gar
nichts.

Siehe auch =>
```

1.145 WRITE.guide/Open

```
Open
====

Aufruf      : Open Dateiname/S Tags/T

Benötigt    : Ed

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Diese Funktion versucht den Text mit dem angegebenen
Dateinamen in den aktuellen Ed zu laden.  Ist dies nicht möglich, so
bricht dies Funktion mit einer Fehlermeldung ab.  Diese Funktion mach
keine Sicherheitsabfrage, falls der alte Text verändert sein sollte! Die
Fehlermeldung durch Requester kann mittels des Tags @SILENT unterdrückt
werden. Open gibt dann einfach nur einen Fehler zurück.

Siehe auch => GetFile, Save, LoadBuffer, SaveBuffer
```

1.146 WRITE.guide/Save

```
Save
====

Aufruf      : Save Dateiname/S Tags/T

Benötigt    : Ed

Setzt Fehler: Ja
```

```
Ergebnisse  : Keine

Beschreibung: Save speichert den Text im aktuellen Ed unter dem
angegebenen Namen ab.  Namensangaben wie PRT: zum Drucken sind möglich.
WRITE restauriert dabei die ursprünglichen Protection-Flags mit Ausnahme
des Archive-Flags (A).  Ist in Mode @RAW gesetzt, werden LF nur bei
leeren Zeilen abgespeichert.  WRITE erzeugt so überlange Zeilen die aus
jeweils einem ganzen Absatz bestehen.  Dies ist praktisch, falls man den
Text anschließend in eine Textverarbeitung einlesen will.

Kann der Text nicht abgespeichert werden, so wird das Abspeichern mit
einem Fehler und dem entsprechenden Fehlerrequester abgebochen. Der
Requester kann durch das Setzten des Tags @SILENT unterdrückt werden.

Siehe auch => GetFile, Open, LoadBuffer, SaveBuffer
```

1.147 WRITE.guide/SetBackUp

```
SetBackUp
=========

Aufruf      : SetBackUp Mode/N Pattern/S To/S

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: SetBackUp stellt die verschiedenen BackUpmodi von WRITE ein.
BackUp bedeutet, daß, wenn ein Text unter einem Dateinamen, der bereits
existiert, abgespeichert werden soll, von der bereits bestehenden Datei
eine Sicherheitskopie gemacht wird.  Momentan gibt es 3 Modi:

  1. Modus 0: Entspricht die Datei dem angegebenen DOS-Pattern, so wird
     sie als angegebe Datei To umbenannt bzw. kopiert.

  2. Modus 1: Entspricht die Datei dem Pattern, so wird die angegebene
     Endung angehangen.

  3. Modus 2: Entspricht die Datei dem Pattern, so wird sie in das
     angegebene Directory kopiert.

WRITE vergleicht die Pattern in absteigender Reihenfolge, d.h. 2...
1...0.

Siehe auch =>
```

1.148 WRITE.guide/New

New
===

Aufruf : New

Benötigt : Ed

Setzt Fehler: Nein

Ergebnisse : Nein

Beschreibung: New löscht den aktuellen Text unwiederruflich. Eine eingebaute Sicherheitsabfrage gibt es dabei nicht. Diese muß programmiert werden. Für ein Beispiel, wie dies zu tun ist, schauen Sie bitte in die beigelegten Standardkonfigurationen.

Siehe auch =>

1.149 WRITE.guide/NewEd

NewEd
=====

Aufruf : NewEd Konfiguration/S

Benötigt : Nichts

Setzt Fehler: Ja

Ergebnisse : Ja

Beschreibung: NewEd öffnet einen neuen Ed mit der angegebenen Konfiguration. Wird als Konfigurations(file) "" angegeben, so die Konfiguration aus der Variable _DefaultConfig benutzt. Kann der Konfigurationfile nicht gefunden werden, oder tritt beim Parsen des selben ein Fehler auf, so bricht NewEd ab und gibt einen Fehler zurück.

Siehe auch => Der Ed

1.150 WRITE.guide/QuitEd

QuitEd
======

Aufruf : QuitEd

Benötigt : Ed

Setzt Fehler: Nein

```
Ergebnisse  : Nein

Beschreibung: Die Funktion schließt den aktuellen Ed, den dazugehörigen
Text sowie, wenn geöffnet, das Fenster.

Eine Sicherheitsabfrage gibt es nicht. Zu Realisierung einer
Sicherheitsabfrage schauen Sie bitte in einen beliebigen beigelegten
Konfigurationsfile nach.

Siehe auch => NewEd
```

1.151 WRITE.guide/Window

```
Window
======

Aufruf      : Window dx1/N dy1/N dx2/N dy2/N Tags/T

Benötigt    : Ed

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Window öffnet für den aktuellen Ed ein Fenster mit den
angegebenen Maßen.  Dabei steht (x1,y1) für die linke obere Ecke
relativ zur linken oberen Ecke des Bildschirms und (x2/y2) für die
rechte untere Ecke relativ zur rechten unteren Ecke.
     Window  0  0  0  0
öffnet ein Fenster über den ganzen Screen.
     WINDOW 50 50 50 50
öffnet ein Fenster, dessen Seiten jeweils 50 Pixel vom Bildschirmrand
entfernt sind.  Hatte der Ed schon einmal ein geöffnetes Fenster, so
werden dessen Maße benutzt.  Konnte das Fenster nicht geöffnet werden,
so wird ein Fehler zurückgegeben.

Ist in Tags das @SCREENREL gesetzt, so beziehen sich die Größenangaben
nicht auf die tatsächliche Screengröße sondern auf einen Screen der
bei _ScrRelWidth und _ScrRelHeight angegebenen Größe. Dies ist
nützlich für übergroße Screen wie man sie bei Grafikkarten besitzt.

Siehe auch => Iconify, Hide, Silent
```

1.152 WRITE.guide/Iconify

```
Iconify
=======

Aufruf      : Iconify
```

Benötigt : Ed

Setzt Fehler: Ja

Ergebnisse : Nein

Beschreibung: Schließt ein eventuell vorhandenes Fenster und öffnet ein iconifiziertes Fenster für den aktuellen Ed. Dieses ist ein kleines Fenster mit einen Schließsymbol in der rechten, oberen Ecke des Screens. Betätigt man dieses Schließsymbol, so wird wieder ein Fenster der alten Größe geöffnet. Mehere Icons werden untereinander plaziert. Kann das Iconfenster nicht geöffnet werden, so wird ein Fehler zurückgegeben.

Siehe auch => Window, Hide, Silent

1.153 WRITE.guide/Hide

Hide
====

Aufruf : Hide

Benötigt : Ed

Setzt Fehler: Nein

Ergebnisse : Nein

Beschreibung: Schließt das Icon oder Fenster des aktuellen Eds. Der Text ist nicht verloren, er besitzt nur kein Fenster mehr. Es gibt mehrere Möglichkeiten das Fenster wieder zu öffnen:

1. Über den WinManager

2. In dem REXX-Script , das in den Hidemodus geschaltet hat, mittels Window oder Iconify (Da die Befehle an das aktuelle Fenster gehen!)

3. Oder ähnlich über den PrivatePort (dessen Namen man über OpenPort bekommen hat, oder den man mit dem ID, der von NewEd zurückgegeben wird, öffnet.

Siehe auch => Window, Iconify, Silent

1.154 WRITE.guide/Silent

Silent
======

Aufruf : Silent

```
Benötigt    : Ed

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Das Fenster des aktuellen Ed (wenn er eins geöffnet hat)
wird in den Silentmode geschaltet.  D.h., daß alle graphischen Ausgaben
unterdrückt werden.  Einige Funktionen wie z.B. Replace erreichen so ein
vielfaches ihrer Geschwindigkeit.

Siehe auch => Window, Iconify, Hide
```

1.155 WRITE.guide/Font

```
Font
====

Aufruf      : Font Name/S yGröße/N DiskFont/N

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Der Text im Editierfenster wird in dem angegebenen Font
dargestellt.  Name und yGröße stehen dabei für den Namen und die Höhe
des Textes.  Ist DiskFont#0, so wird der Font von Diskette geladen,
ansonsten wird er im ROM gesucht.  Kann der Font nicht gefunden
werden,wird ein Fehler zurückgegeben

Siehe auch => GetFont
```

1.156 WRITE.guide/WinArranger

```
WinArranger
===========

Aufruf      : WinArranger Gewichtung/N Tags/T

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: WinArranger versucht alle geöffneten Fenster
übersichtlich, flächendeckend und ohne Überlappungen auf dem Screen zu
plazieren.  Momentan werden zwei Modi unterstützt:
```

```
   *     @VERT - Alle offenen Fenster werden untereinander plaziert und
     nach vorne gebracht.

   *     @HORIZ - Alle Fenster werden nebeneinander plaziert und nach
     vorne gebracht.

WinArranger beachtet beim Plazieren der Fenster auch den Inhalt der
Variablen _ScrRelWidth und _ScrRelHeight.

Über Gewichtung kann man einstellen, wie viel das aktuelle Fenster
größer als alle anderen sein soll. Bei dem Wert 2 z.B. ist das atuelle
Fenster genau zweimal so groß wie die anderen Fenster. Eine interessante
Einstellung ist z.B.  WinArranger _Wins.

Siehe auch =>
```

1.157 WRITE.guide/SetTitle

```
SetTitle
========

Aufruf      : SetTitle Titel/S NoHold/N

Benötigt    : Ed

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: SetTitle setzt den Fenstertitel aud Titel.  Die Flags und
Positionsangaben bleiben dabei sichtbar.  Ist NoHold=0, so wird der Titel
nach Ausführung der aktuellen Befehlsfolge gelöscht, ansonsten nicht.

Siehe auch =>
```

1.158 WRITE.guide/Screen

```
Screen
======

Aufruf      : Screen Data/S Titel/S Flags/T

Benötigt    : Konfiguration

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Screen gibt an, auf welchem Screen eine Konfiguration ihre
```

Fenster öffnen soll. Momentan werden nur fremde PublicScreens
unterstützt. Dabei ist Titel der Name des PublicScreens. In Flags muß
@PUBLIC gesetzt sein.

Der neue Screen gilt erst für alle neu geöffneten Fenster der aktuellen
Konfiguration. Screen sollte deshalb schon bereits im Konfigurationsfile
aufgerufen werden.

Kann Screen auf einen Screen nicht zugreifen, so wird der
"DefaultPublicScreen" benutzt. Dies ist meißtens die Workbench.
Außerdem gibt Screen dann einen Fehler zurück.

Es sollte hier noch einmal außdrücklich erwähnt werden, daß WRITE sich
völlig an die Gegebenheiten eines beliebigen Screens anpaßt. Dies sind
vor allem die Auflösung, so wie der eingestellte Zeichensatz des Screens.
Alle Fenster und Requester von WRITE sind fontsensitiv und passen sich
sogar Proportionalfonts problemlos an.

Siehe auch => PublicScreens

1.159 WRITE.guide/About

About
=====

Aufruf   : About

Benötigt   : Nichts

Setzt Fehler: Nein

Ergebnisse : Nein

Beschreibung: About öffnet ein kleines Fenster, in dem die
Versionsnummer, mein Name und meine Adresse, sowie ein kleiner Hinweis in
Sachen Raubkopien und dem legalen Besitz von Originalen stehen. Beachten
Sie, daß das dieser Requester asyncron ist, das heißt, daß Sie in
einem beliebigen Editorfenster bzw. Requester weiterarbeiten können,
ohne den About-Requester zu schließen.

Siehe auch => Copyright, Vollversion, Requester

1.160 WRITE.guide/Prefs

Prefs
=====

Aufruf   : Prefs

Benötigt   : Ed

```
Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Hier kann man einige konfigurationsspezifische
Einstellungen ändern und abspeichern.

Dabei werden die Einstellung nach einzelnen Themenbereichen aufgeteilt.
Alle Einstellungen in den Unterrequestern werden bei drücken der OK-Taste
übernommen. Drückt man im Preferencerequester Speichern..., so wird
automatisch ein neuer Konfigurationsfile mit dem angegebenen Namen
erzeugt. Durch Drücken von Ende werden die Einstellung nur übernommen,
aber nicht abgespeichert.

Durch das Setzen des Tags @TOMOUSE wird der Requester zentriert unter
dem Mauszeiger geöffnet. Siehe dazu auch das Kapitel Requester

Siehe auch => GPrefs, Requester
```

1.161 WRITE.guide/GPrefs

```
GPrefs
======

Aufruf      : GPrefs

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Hier kann man einige globale Einstellungen ändern.

Hier gilt das gleiche wie auch für wie für den Prefsrequester:  Drückt
man im Preferencerequester Speichern, so wird automatisch eine neue
STARTUP.CONFIG erzeugen. Durch Drücken von Ende werden die Einstellung
nur übernommen, aber nicht abgespeichert.

Durch das Setzen des Tags @TOMOUSE wird der Requester zentriert unter
dem Mauszeiger geöffnet. Siehe dazu auch das Kapitel Requester

Siehe auch => Prefs, Requester
```

1.162 WRITE.guide/HelpFkt

```
Help
====
```

Aufruf : Help

Benötigt : Nichts

Setzt Fehler: Nein

Ergebnisse : Nein

Beschreibung: Help öffnet ein kleines Fenster mit verschiedenen Gadgets
über die einige weitere Hilfrequester geöffnet werden können.

Durch das Setzen des Tags @TOMOUSE wird der Requester zentriert unter
dem Mauszeiger geöffnet. Siehe dazu auch das Kapitel Requester.
Beachten Sie, daß das dieser Requester asyncron ist, das heißt, daß
Sie in einem beliebigen Editorfenster bzw. Requester weiterarbeiten
können, ohne den Hilfs-Requester zu schließen.

Siehe auch => ShowVars, ShowFunctions, ShowASCII, Requester

1.163 WRITE.guide/WinManager

WinManager
==========

Aufruf : WinManager

Benötigt : Nichts

Setzt Fehler: Nein

Ergebnisse : Nein

Beschreibung: WinManager öffnet ein Fenster mit einem Listrequester, in
dem alle geöffneten Eds gezeigt werden. Da bei bedeutet die erste Zahl
der ID, ein anschließendes Sternchen zeigt an, daß der Text verändert
wurde, und ein abschließendes h, i, s gibt an, ob sich der Ed im Hiden-,
Iconify- oder im Silent-Mode befindet. Der Eintrag schließt mit dem
Dateinamen des aktuellen Textes ab. Mit den darunter angebrachten
Gadgets kann der Requester wieder verlassen, ein neues Fenster geöffnet,
ein selektierter Ed gezeigt, gelöscht , oder, falls kein Ed vorhanden,
WRITE verlassen werden. Ein Eintrag kann durch anklicken mit der Maus,
oder (ab OS 3.0) über die CursorUp bzw. CursorDown Tasten selektiert
werden.

Das Verlassen von WRITE geht nur, wenn alle Fenster geschlossen sind.

die Taste Neues Fenster führt bei Selektion die Userfunktion 2
(SetUserFkt,GPrefs) aus. Damit läßt sich
konfigurieren, was für ein Fenster (Konfiguration, Größe) geöffnet
werden soll.

Durch das Setzen des Tags @TOMOUSE wird der Requester zentriert unter
dem Mauszeiger geöffnet. Siehe dazu auch das Kapitel Requester.

Beachten Sie, daßdas dieser Requester asyncron ist, das heißt, daß Sie
in einem beliebigen Editorfenster bzw. Requester weiterarbeiten können,
ohne den WinManager-Requester zu schließen.

Siehe auch => Requester

1.164 WRITE.guide/ShowVars

ShowVars
========

Aufruf : ShowVars

Benötigt : Nichts

Setzt Fehler: Nein

Ergebnisse : Nein

Beschreibung: Öffnet ein Fenster mit einem Listgadget, in dem alle
internen Variablen mit ihren Namen sowie dem aktuellen Wert stehen.

Steht hinter dem Variablennamen ein P, so ist die Variable nur auslesbar,
wenn eine Konfiguration aktiviert ist. Steht hinter dem Namen ein E,
sogar nur, wenn ein Ed aktiviert ist.

Durch das Setzen des Tags @TOMOUSE wird der Requester zentriert unter
dem Mauszeiger geöffnet. Siehe dazu auch das Kapitel Requester

Siehe auch => HelpFkt, ShowFunctions, ShowASCII, ShowConstants,
Requester, GetConfig

1.165 WRITE.guide/ShowFunctions

ShowFunctions
=============

Aufruf : ShowFunctions

Benötigt : Nichts

Setzt Fehler: Nein

Ergebnisse : Nein

Beschreibung: Eine Liste mit allen internen Funktionen samt ihrer
Parameter.

Durch das Setzen des Tags @TOMOUSE wird der Requester zentriert unter
dem Mauszeiger geöffnet. Siehe dazu auch das Kapitel Requester

Hinter dem Variablennamen stehen einige Buchstaben:

* C: Die Funktion kann nur mit aktiver Konfiguration ausgeführt werden.
* E: Die Funktion kann nur mit aktiven Ed ausgeführt werden.
* W: Die Funktion kann nur mit aktivem Fenster ausgeführt werden.
* *: Die Funktion ändert den Text.
* S: Die Funktion gibt einen String zurück.
* N: Die Funktion bit eine Zahl zurück.
* !: Die Funktion gibt gegebenenfalls einen Fehler zurück.

Siehe auch => HelpFkt, ShowVars, ShowASCII, ShowConstants, Requester

1.166 WRITE.guide/ShowConstants

ShowConstants
=============

Aufruf : ShowConstants

Benötigt : Nichts

Setzt Fehler: Nein

Ergebnisse : Nein

Beschreibung: Eine Liste mit allen internen Konstanten.

Durch das Setzen des Tags @TOMOUSE wird der Requester zentriert unter dem Mauszeiger geöffnet. Siehe dazu auch das Kapitel Requester

Siehe auch => HelpFkt, ShowVars, ShowASCII, ShowFunctions, Requester

1.167 WRITE.guide/ShowASCII

ShowASCII
=========

Aufruf : ShowASCII

Benötigt : Nichts

Setzt Fehler: Nein

```
Ergebnisse  : Nein

Beschreibung: Öffnet ein Fenster mit einer Reihe von Gadgets, in dem alle
256 ASCII-Zeichen abgebildet sind. Nicht darstellbare Zeichen, werden
durch ein ? repräsentiert.

Durch das Anklicken eines Gadgets wird der entsprechende Buchstabe in das
zuletzt aktivierten Editorfenster an der Cursorposition eingefügt. Auch
Tasteneingaben bei aktiven Requester werden an das zuletzt aktive
Editorfenster weitergeleitet.

Durch das Setzen des Tags @TOMOUSE wird der Requester zentriert unter
dem Mauszeiger geöffnet. Siehe dazu auch das Kapitel Requester

Beachten Sie, daß das dieser Requester asyncron ist, das heißt, daß
Sie in einem beliebigen Editorfenster bzw. Requester weiterarbeiten
können, ohne den ASCII-Requester zu schließen.

Siehe auch => WriteChar, HelpFkt, ShowFunctions, ShowVars, ShowConstants,
Requester
```

1.168 WRITE.guide/ShowIndex

```
ShowIndex
=========

Aufruf      : ShowIndex Pattern/S Flags/T

Benötigt    : Ed

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Sucht im gesamten Text nach dem angegebenen DOS-Pattern.
Wenn @CASE in Flags gesetzt ist, so wird dabei zwischen Groß- und
Kleinschreibung unterschieden. Wenn @NOCASE gesetzt ist , nicht.
Ansonsten wird die Preferenceseinstellung benutzt.  Anschließend werden
alle Zeilen, in den das Pattern gefunden wurde, in einem Requester mit
ihrer Zeilennummer angezeigt.  Durch Selektion einer Zeile kann dann zu
dieser gesprungen werden.

Für Oberon/Modula2-Programmierer ist zum Beispiel nützlich:
     ShowIndex "#?PROCEDURE#?" 1

Durch das Setzen des Tags @TOMOUSE wird der Requester zentriert unter
dem Mauszeiger geöffnet. Siehe dazu auch das Kapitel Requester

Siehe auch =>, Requester
```

1.169 WRITE.guide/GetString

```
GetString
=========

Aufruf      : GetString Title/S Vorbelegung/S

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Ja

Beschreibung: GetString öffnet einen Requester mit dem angegebenen Titel
und einem Stringgadget mit dem angegebenen Inhalt.  Wird im Stringgadget
RETURN gedrückt oder das Bestätigungsgadget betätigt, so wird in der
Variablen _RS der eingegebene String zurückgegeben.  Andernfalls wird
ein Fehler zurückgegeben.

Durch das Setzen des Tags @TOMOUSE wird der Requester zentriert unter
dem Mauszeiger geöffnet. Siehe dazu auch das Kapitel Requester

Siehe auch => GetNumber, Requester
```

1.170 WRITE.guide/GetNumber

```
GetNumber
=========

Aufruf      : GetNumber Titel/S Vorbelegung/N

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Ja

Beschreibung: GetNumber öffnet einen Requester mit dem angegebenen Titel
und einem Integergadget mit dem angegebenen Inhalt.  Wird im
Integergadget RETURN gedrückt oder das Bestätigungsgadget betätigt, so
wird in der Variablen _RN die eingegebene Zahl zurückgegeben.
Andernfalls wird ein Fehler zurückgegeben.

Durch das Setzen des Tags @TOMOUSE wird der Requester zentriert unter
dem Mauszeiger geöffnet. Siehe dazu auch das Kapitel Requester

Siehe auch => GetString, Requester
```

1.171 WRITE.guide/GetFindReplace

```
GetFindReplace
==============

Aufruf      : GetFindReplace FindWord/S ReplaceWord/S Flags/T

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: GetFindReplace öffnet einen Requester in dem kompfortabel
die Variablen _FindString, _CaseSense und, falls @REPLACE in Flags
gesetzt ist, auch die Variable _ReplaceString gesetzt werden können.
Drückt man in einen der beiden StringGadgets die HELP-Taste, so öffnet
sich ein ListRequester, über den man eins der vorher eingebenen Wörter
selektieren kann.

Durch das Setzen des Tags @TOMOUSE wird der Requester zentriert unter
dem Mauszeiger geöffnet. Siehe dazu auch das Kapitel Requester

Siehe auch => Find, Replace, FindPattern, Requester
```

1.172 WRITE.guide/GetFile

```
GetFile
=======

Aufruf      : GetFile Pfad,Pattern/S Typ/T

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Ja

Beschreibung: GetFile öffnet einen Filerequester mit dem Pfad Pfad oder,
falls dieser leer ist ("") mit dem Pfad, der in der Variable _Path steht.
Es werden dabei nur Files, die dem Suchmuster Pattern entsprechen, oder,
falls dieser leer ist, dem Muster in der Variablen _FRPattern.

Ist in Typ @SAVE gesetzt, so wird ein Save-Filerequester benutzt (Es
wird helle Schrift auf dunklem Hintergrund benutzt).

Wird eine Datei selektiert, so wird der dazugehörige Dateiname ,falls
Pfad leer ist, in der Variablen _Path zurückgegeben, ansonsten in _RS.

Andernfalls wird ein Fehler zurückgegeben.

Siehe auch => GetFiles
```

1.173 WRITE.guide/GetFiles

```
GetFiles
========

Aufruf      : GetFiles Liste,Pfad,Pattern/S Typ/T

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Ja

Beschreibung: Gleiches Verhalten wie GetFile. Nur das hier nicht nur ein
File sondern mehrere seletiert werden können, und diese nicht in _Path
oder _RS sondern in der angegebenen Liste eingetragen werden. Diese muß
vorher existieren.

Siehe auch => GetFile
```

1.174 WRITE.guide/GetFont

```
GetFont
=======

Aufruf      : GetFont

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: GetFont öffnet einen Fontrequester.  Wird ein Font
selektiert, so wird versucht, diesen als neuen Font für die
Textdarstellung zu benutzen. Wird kein Font selektiert, oder kann dieser
nicht geladen dargestellt etc. werden, wird die Abarbeitung abgebrochen.

Siehe auch =>
```

1.175 WRITE.guide/Ask

```
Ask
===

Aufruf      : Ask Title/S Gadgets/S Flags/T

Benötigt    : Nichts

Setzt Fehler: Nein
```

```
Ergebnisse  : Ja

Beschreibung: Ask öffnet einen Requester mit Title als Text und den in
Gadgets beschriebenen Gadgets als mögliche Antwort.  Ist in Flags das
Tag @VERT gesetzt, so wird rechts neben dem Text die Gadgeliste vertikal
erzeugt, sonst horzontal unter dem Text.

Die Beschreibung der Gadgets folgt folgender Konvention: Die Gadgettitel
werden hintereinander aufgelistet und durch das Zeichen | von einander
getrennt.  Mittels eines Unterstriches im Titel kann der nächste
Buchstabe als Tastaturkürzel markiert werden.  Ein ^ bedeutet, daß
dieses Gadget auch über die Esacape-Taste selectiert werden kann.
Dieses Gadget sollte demnach für einen Abruch oder für das sichere
Verlassen des Requester stehen.  Ein * bedeutet, daß dieses Gadget auch
mit RETURN beantwortet werden kann.  Dieses Gadget sollte also für die
vorgeschlagene Standardbeanwortung stehen.

Die Gadgets werden von Links nach Rechts von 0 aufwärts durchnummeriert.
Ask gibt in _RN die Nummer des selektierten Gadgets zurück.  Ein
Beispiel:
     Ask "Dies ist ein Test!!!" "Echt _super man!*|_Nicht so toll^"

Durch das Setzen des Tags @TOMOUSE wird der Requester zentriert unter
dem Mauszeiger geöffnet. Siehe dazu auch das Kapitel Requester

Siehe auch => Requester, Message, MessageOK
```

1.176 WRITE.guide/Message

```
Message
=======

Aufruf      : Message Text/S

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Message öffnet einen Requester mit dem angegebenen Text.
Nach kurzer Zeit verschwindet der Requester wieder.

Durch das Setzen des Tags @TOMOUSE wird der Requester zentriert unter
dem Mauszeiger geöffnet. Siehe dazu auch das Kapitel Requester

Siehe auch => MessageOK, Requester
```

1.177 WRITE.guide/MessageOK

```
MessageOK
=========

Aufruf      : MessageOK Text/S

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Funktioniert wie Message, nur das hier die Meldung erst
durch ein Gadget bestätigt werden muß bevor sie verschwindet.

Durch das Setzen des Tags @TOMOUSE wird der Requester zentriert unter
dem Mauszeiger geöffnet. Siehe dazu auch das Kapitel Requester

Siehe auch => Message, Requester
```

1.178 WRITE.guide/Flash

```
Flash
=====

Aufruf      : Flash

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Läßt den Bildschirm einmal aufblinken.

Siehe auch => Beep
```

1.179 WRITE.guide/Beep

```
Beep
====

Aufruf      : Beep Deep/N

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein
```

Beschreibung: Beep erzeugt einen kurzen Ton. Ist Deep=0, dann ist der
Ton relativ hoch, sonst tief.

Siehe auch => Flash

1.180 WRITE.guide/ParseBuffer

ParseBuffer
===========

Aufruf   : ParseBuffer Nummer/N

Benötigt   : Nichts

Setzt Fehler: Ja

Ergebnisse : Nein

Beschreibung: ParseBuffer behandelt den Text in dem Puffer mit der
angegebenen Nummer als WRITE-Script und prüft es auf seine syntaktische
Richtigkeit ohne es dabei jedoch auszuführen. Eventuell auftretende
Fehler werden wie üblich im Textfenster gemeldet. Sind Fehler
aufgetreten, so wird dementsprechend von der Funktion ein Fehler
zurückgegeben.

Diese Funktion ist sinnvoll, wenn man Teile des Konfigurationsfiles
ändert, und sie anschließend auf ihre Richtigkeit überprüfen will.

Siehe auch => DoBuffer, DoString, PreparseString

1.181 WRITE.guide/DoBuffer

DoBuffer
========

Aufruf   : DoBuffer Nummer/N

Benötigt   : Nichts

Setzt Fehler: Ja

Ergebnisse : Nein

Beschreibung: Führt den angegebenen Puffer als WRITE-Script aus. Treten
dabei Fehler auf, so gibt die Funktion ebenfalls einen Fehler aus.

Siehe auch => ParseBuffer, DoString, PreparseString

1.182 WRITE.guide/DoString

```
DoString
========

Aufruf      : DoString String/S

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Führt den angegebenen String als WRITE-Script aus.
Natürlich gibt auch diese Funktion dabei auftretene Fehler als Fehler
zurück.

Siehe auch => ParseBuffer, DoBuffer, PreparseString
```

1.183 WRITE.guide/PreparseString

```
PreparseString
==============

Aufruf      : PreparseString String/S

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: PreparseBuffer parsed den String, kontrolliert auf
Richtigkeit und ersetzt dabei Variablen durch ihre Inhalte.  Der String
wird in _RS zurückgegeben.

Siehe auch => DoBuffer, DoString, ParseBuffer
```

1.184 WRITE.guide/SetUserFkt

```
SetUserFkt
==========

Aufruf      : SetUserFkt Nummer/N Funktionsliste/F

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein
```

Beschreibung: Userfunktionen sind Stellen im internen Ablauf, wo der
Benutzer konfigurierend eingreifen kann. Momentan gibt es folgende
Userfunktionen:

0: Diese Funktion wird für jeden Commandlineparameter einmal aufgerufen.
Der Parameter steht dabei in der Variablen _RS. Diese Funktion sollte
ein Fenster öffnen und die angegebene Textdatei laden.

1: Diese Funktion wird aufgerufen, wenn das Schließ-Gadget eines Fenster
angeklickt wird. Die Funktion sollte eventuell eine Sicherheitsabfrage
machen und anschließend das Fenster schließen.

2: Diese Funktion wird bei Selektion des Gadgets Neues Fenster im
WinManager aufgerufen. Sie sollte ein Editorfenster öffnen.

3: Diese Funtion wird nicht mehr unterstützt.

4: Diese Funktion wird aufgerufen, wenn eine Konfiguration aus dem
Speicher befreit wird. Gibt diese Funktion einen Fehler zurück. So wird
die Freigabe abgebrochen. Hat man z.B. im entsprechenen
Konfigurationsfile z.B. einem externen Screenmanager gesagt, daß er
einen Screen für unsere Konfiguration öffnen soll, kann man ihn in
dieser Funktion sagen, daß der Screen nicht mehr gebraucht wird und somit
geschlossen werden kann. Sehen Sie hirzu auch bei der Funktion Screen und
im Kapitel PublicScreens nach. Für diesen Zweck existieren mittlerweile
die Befehle Begin/Closefür Konfigurationsfiles und Start/Quit für die
STARTUP.CONFIG. Bitte diese Befehle nutzen!

Für Beispiele hierzu schauen Sie bitte in die Konfigurationsdateien.

Siehe auch =>

1.185 WRITE.guide/Compare

Compare
=======

Aufruf   : Compare String1/S String2 /S

Benötigt   : Nichts

Setzt Fehler: Nein

Ergebnisse : Ja

Beschreibung: Vergleicht die beiden Strings und gibt je nach Ergebnis die
Werte LOWER,HIGHER oder EQUAL in _RN zurück. Da Zahlen bei der
Übergabe automatisch in Zeichenketten umgewandelt werden, kann man mit
dieser Funktion auch Zahlen untereinander als auch Zahlen und
Zeichenketten miteinander vergleichen.

Siehe auch => If, Break

1.186 WRITE.guide/If

```
If
==

Aufruf      : If Command/F Then/F Else/F Tags/T

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: If führt die angegebene Funktionsliste Command aus.  Gibt
diese Funktion keinen Fehler zurück und ist _RN=0, so wird die
Funktionsliste Then ausgeführt, ansonsten die Funktionsliste Else. Ist
das Tag @CLEARRN gesetzt, so wird _RN vor der Ausführung von Command
gelöscht. Dies ist z.B. wichtig, wenn _RN ungleich 0 ist, Command _RN
nicht ändert und If dementsprechend nur auf die Rückgabe eines Fehlers
reagieren soll.

Siehe auch => Break
```

1.187 WRITE.guide/Break

```
Break
=====

Aufruf      : Break Compare/N String1/S String2/S

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Break vergleicht String1 und String2 Und stellt fest, das
sie gleich (EQUAL), String1 kleiner als String2 (LOWER) oder größer
(EQUAL) ist.  Ist nun Compare gleich dem Resultat,
so gibt Break einen Fehler zurück und die aktuelle Befehlsfolge wird
abgebrochen.  Beispiel:
     Break EQUAL "Hallo" "Hallo"
     Break LOWER 255 13
Das erste Beispiel bricht ab, das Zweite nicht.

Siehe auch => If
```

1.188 WRITE.guide/SetError

```
SetError
========

Aufruf      : SetError

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Bricht die anktuelle Anweisungsfolge ab.

Siehe auch => Break, If, Compare
```

1.189 WRITE.guide/SetVar

```
SetVar
======

Aufruf      : SetVar Variable/S Wert/S

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Setzt den Inhalt der einer internen Variable auf den Wert
Wert.  Variable ist dabei der gequotete Name der Variable.  Existiert die
Variable nichts, so wird ein Fehler zurückgegeben.  Beispiel:
     SetEnv "_FileName" "s:startup-sequence"
Setzt den Dateinamen des aktuellen Textes auf s:startup-sequence.

Siehe auch => GetVar
```

1.190 WRITE.guide/GetVar

```
GetVar
======

Aufruf      : GetVar Variable/S

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Ja
```

```
Beschreibung: Liest den Wert der Variablen Variable aus und schreibt ihn
als String in die Variable _RS.  Existiert die angegebene Variable nicht,
so wird ein Fehler zurückgegeben.

Siehe auch => SetVar
```

1.191 WRITE.guide/SetEnv

```
SetEnv
======

Aufruf      : SetEnv Variable/S Wert/S

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Setzt den Wert einer lokalen DOS-Enviromentvariabeln auf
den Wert Wert.

Siehe auch =>
```

1.192 WRITE.guide/GetEnv

```
GetEnv
======

Aufruf      : GetEnv Variable/S

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Ja

Beschreibung: Ließt den Wert der angegebenen Dos-Enviromentvariabeln aus
und schreibt ihn in die Variable _RS

Siehe auch =>
```

1.193 WRITE.guide/System

System
======

Aufruf : System Befehl/S Flags/T

Benötigt : Nichts

Setzt Fehler: Ja

Ergebnisse : Nein

Beschreibung: System startet einen Befehl (mit Argumenten). Ist @ASYNC ind Flags gesetzt, so wird der Befehl im Hintergrund gestartet. D.h., daß mit WRITE während das Programm läuft, weiter gearbeitet werden kann. Ansonsten wartet WRITE darauf, daß das Programm beendet wird. Aus leicht einsichtigen Gründen, kann nur ein Fehler zurückgegeben werden, wenn @ASYNC nicht gesetzt ist.

Siehe auch =>

1.194 WRITE.guide/SetMark

SetMark
=======

Aufruf : SetMark

Benötigt : Ed

Setzt Fehler: Nein

Ergebnisse : Nein

Beschreibung: Setzt den Anfang bzw. das Ende eines Blockes,welcher dann mit Cut, Copy ... bearbeitet werden kann. Setzt mann zum ersten Mal eine Marke, so verändert sich der Mauszeiger in ein Visier (???) und ein M erscheint bei den Flags in der Titelzeile. Desweiteren ist ein zeichengroßes Kästen unter dem Cursor zu sehen. Man ist im Mark-Modus. Dieser bleibt solange erhalten , wie man nicht auf irgendeine Art einen Zeilenumbruch tätigt. Nun kann mit den gleichen Verfahren das Ende des Blockes gewählt werden. Ist das Ende des Blockes vor dem Anfang, so werden Anfangs- und Endmarken automatisch vertauscht. Ist dies getan, so wird der Block als Ganzes farblich hervorgehoben. Der Block ist markiert und bleibt dies auch, bis ein Zeilenumbruch durchgeführt wurde. Nun kann der Block mit den Blockfunktionen bearbeitet werden.

Mit einen Doppelklick mit der linken Maustaste ist es z.B. möglich Marken zu setzen.

Siehe auch => UnMark,Mark

1.195 WRITE.guide/Mark

```
Mark
====

Aufruf      : Mark VonX,VonY,BisX,BisY/N Tags/T

Benötigt    : Ed

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Markiert den angegebenen Block.

Durch das Setzten des Tags @SILENT können eventuelle Fehlerrequester
unterdrückt werden.

Siehe auch => Mark, UnMark, Konstantenbeschreibung
```

1.196 WRITE.guide/UnMark

```
UnMark
======

Aufruf      : UnMark

Benötigt    : Ed

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: UnMark löscht den markierten Block.  Diese Funktion wird
auch aufgerufen, wenn der rechte Mausknopf gedrückt wurde, ohne ein
Menü auszuwählen.

Siehe auch => SetMark,Mark
```

1.197 WRITE.guide/DeleteBlock

```
DeleteBlock
===========

Aufruf      : DeleteBlock

Benötigt    : Ed

Setzt Fehler: Ja
```

```
Ergebnisse  : Nein

Beschreibung: Löscht den markierten Block.  Ist kein Block markiert, so
wird mit einem Fehler abgebrochen.

Durch das Setzten des Tags @SILENT können eventuelle Fehlerrequester
unterdrückt werden.

Siehe auch => CopyBlock, InsertBlock, CopyArea, DeleteArea
```

1.198 WRITE.guide/CopyBlock

```
CopyBlock
=========

Aufruf      : CopyBlock Puffername/S

Benötigt    : Ed

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Kopiert den markierten Block in den angegebenen
Zwischenspeicher.  Ist kein Block markiert, so wird mit einem Fehler
abgebrochen.

Durch das Setzten des Tags @SILENT können eventuelle Fehlerrequester
unterdrückt werden.

Siehe auch => DeleteBlock, InsertBlock, CopyArea, DeleteArea
```

1.199 WRITE.guide/InsertBlock

```
InsertBlock
===========

Aufruf      : InsertBlock Puffername/S

Benötigt    : Ed

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Fügt den Inhalt des angegebenen Blocks an der aktuellen
Cursorposition in den Text ein.  Ist der angegebene Block leer, so wird
mit einem Fehler abgebrochen.

Durch das Setzten des Tags @SILENT können eventuelle Fehlerrequester
```

unterdrückt werden.

Siehe auch => CopyBlock, DeleteBlock, CopyArea, DeleteArea

1.200 WRITE.guide/DeleteArea

```
DeleteArea
==========

Aufruf      : DeleteArea VonX,VonY,BisX,BisY/N

Benötigt    : Ed

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Genau die gleiche Funtion wie DeleteBlock, nur das hier ein
beliebiger Block angegeben werden kann.

Durch das Setzten des Tags @SILENT können eventuelle Fehlerrequester
unterdrückt werden.

Siehe auch => CopyBlock, InsertBlock, CopyArea, Konstantenbeschreibung
```

1.201 WRITE.guide/CopyArea

```
CopyArea
========

Aufruf      : CopyArea Buffername/S VonX,VonY,BisX,BisY/N

Benötigt    : Ed

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Genau die gleiche Funtion wie CopyBlock, nur das hier ein
beliebiger Block angegeben werden kann.

Durch das Setzten des Tags @SILENT können eventuelle Fehlerrequester
unterdrückt werden.

Siehe auch => DeleteBlock, InsertBlock, DeleteArea, Konstantenbeschreibung
```

1.202 WRITE.guide/SaveBuffer

```
SaveBuffer
==========

Aufruf      : SaveBuffer Dateiname/S Puffername/S

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Speichert den Inhalt des angegebenen Puffers als Datei mit
dem Namen Dateiname ab.  Treten beim Abspeichern Fehler auf, so wird mit
einem Fehler abgebrochen.  Achtung!  Die Einstellungsmöglichkeiten für
Textfiles, wie z.B. die Abspeicherung von Tabs, gelten hier nicht.
Blöcke werden immer im Standard-ADSCII-Format abgespeichert.

Durch das Setzten des Tags @SILENT können eventuelle Fehlerrequester
unterdrückt werden.

Siehe auch => LoadBuffer
```

1.203 WRITE.guide/LoadBuffer

```
LoadBuffer
==========

Aufruf      : LoadBuffer Dateiname/S Puffername/S

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Läd die Datei Dateiname in den angegebenen Block.  Treten
beim Laden Fehler auf, so wird mit einem Fehler abgebrochen.  Achtung!
Die Einstellungsmöglichkeiten für Textfiles, wie z.B. die Abspeicherung
von Tabs, gelten hier nicht.  Blöcke werden immer im
Standard-ADSCII-Format abgespeichert.

Durch das Setzten des Tags @SILENT können eventuelle Fehlerrequester
unterdrückt werden.

Siehe auch => SaveBuffer
```

1.204 WRITE.guide/ClearBuffer

```
ClearBuffer
```

```
===========

Aufruf      : ClearBuffer Puffername/S

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Löscht den angegebenen Puffer. Ist der Buffer noch nicht
vorhanden wird er erzeugt.

Siehe auch =>
```

1.205 WRITE.guide/StrToBuffer

```
StrToBuffer
===========

Aufruf      : StrToBuffer String/S Puffername/S

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Der angegebene Buffer wird mit dem angegebenen String
belegt.

Siehe auch =>
```

1.206 WRITE.guide/BufferToStr

```
BufferToStr
===========

Aufruf      : BufferToStr Puffername/S

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Ja

Beschreibung: Die maximal ersten 256 Zeichen des angegebenen Buffers
werden in _RS zurückgegeben.

Durch das Setzten des Tags @SILENT können eventuelle Fehlerrequester
```

unterdrückt werden.

Siehe auch =>

1.207 WRITE.guide/ClipToBuffer

```
ClipToBuffer
============

Aufruf      : ClipToBuffer Puffername/S Clipboardnummer/N

Benötigt    : iffparse.library

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Kopiert den Inhalt des angebenen Clipboarddevices in eine
internen Puffer.  Für weitere Informationen über die Funktion und die
Arbeitsweise des Clipboarddevices schauen Sie bitte in die Handbücher,
die Sie mit Ihrem Amiga erhalten haben.

Siehe auch => BufferToClip
```

1.208 WRITE.guide/BufferToClip

```
BufferToClip
============

Aufruf      : BufferToClip Puffername/S Clipboardnummer/N

Benötigt    : iffparse.library

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Kopiert den angegeben internen Puffer in ein
Clipboarddevice.  Für weitere Informationen über die Funktion und die
Arbeitsweise des Clipboarddevices schauen Sie bitte in die Handbücher,
die Sie mit Ihrem Amiga erhalten haben.

Durch das Setzten des Tags @SILENT können eventuelle Fehlerrequester
unterdrückt werden.

Siehe auch => BufferToClip
```

1.209 WRITE.guide/BlockLeft

```
BlockLeft
=========

Aufruf      : BlockLeft x/N

Benötigt    : Ed

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Verschiebt den markierten Block als Ganzes um x Zeichen
nach links.

Siehe auch => BlockRight
```

1.210 WRITE.guide/BlockRight

```
BlockRight
==========

Aufruf      : BlockRight x/N

Benötigt    : Ed

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Verschiebt den markierten Block als Ganzes um x Zeichen
nach rechts.

Siehe auch => BlockRight
```

1.211 WRITE.guide/BlockLftAlig

```
BlockLftAlig
============

Aufruf      : BlockLftAlig Spalte/N

Benötigt    : Ed

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Versucht alle Zeilen im markierten Block linksbündig auf
```

```
die angegebene Spalte auszurichten.

Siehe auch => BlockRghtAlig
```

1.212 WRITE.guide/BlockRghtAlig

```
BlockRghtAlig
=============

Aufruf      : BlockRghtAlig Spalte/N

Benötigt    : Ed

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Versucht alle Zeilen im markierten Block rechtsbündig auf
die angegebene Spalte auszurichten.

Siehe auch =>
```

1.213 WRITE.guide/BlockCenter

```
BlockCenter
===========

Aufruf      : Diese Funktion ist noch nicht implementiert

Benötigt    :

Setzt Fehler:

Ergebnisse  :

Beschreibung:

Siehe auch =>
```

1.214 WRITE.guide/CursorUp

```
CursorUp
========

Aufruf      : CursorUp Mode/N

Benötigt    : Ed
```

```
Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Ist Mode=0, so wird der Cursor einfach eine Zeile nach oben
bewegt.  Ist Mode=1, so wird der Cursor ebenfalls eine Zeile nach oben
bewegt und, falls der Cursor dann hinter dem letzten Zeichen in dieser
Zeile steht, einen Buchstaben hinter dem Ende der Zeile positioniert.
Steht der Cursor bereits in der ersten Zeile, so wird ein Fehler
zurückgegeben.

Siehe auch =>
```

1.215 WRITE.guide/CursorDown

```
CursorDown
==========

Aufruf      : CursorDown Mode/N

Benötigt    : Ed

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Funktioniert wie CursorUp, nur daß der Cursor eine Zeile
nach unten bewegt wird.

Siehe auch =>
```

1.216 WRITE.guide/CursorRight

```
CursorRight
===========

Aufruf      : CursorRight Mode/N

Benötigt    : Ed

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Ist Mode=0, so wird der Cursor eine Stelle nach rechts
bewegt.  Ist Mode=1, so wird der Cursor ebenfalls eine Stelle nach rechts
bewegt.  Steht er dann jedoch hinter dem letzten Zeichen der Zeile,
springt er an der Anfang der nächsten.  Existiert diese nicht, so wird
ein Fehler zurückgegeben.
```

Siehe auch =>

1.217 WRITE.guide/CursorLeft

```
CursorLeft
==========

Aufruf      : CursorLeft

Benötigt    : Ed

Setzt Fehler: Ja

Ergebnisse  : Ja

Beschreibung:  Ist Mode=0, so wird der Cursor eine Stelle nach links
bewegt.  Steht der Cursor bereits in der ersten Spalte, so wird ein Fehler
zurückgegeben.  Ist Mode=1, so wird der Cursor ebenfalls eine Stelle
nach links bewegt.  Steht er jedoch in der ersten Spalte der Zeile,
springt er ans Ende der letzten.  Existiert diese nicht, so wird ein
Fehler zurückgegeben.

Siehe auch =>
```

1.218 WRITE.guide/NextWord

```
NextWord
========

Aufruf      : NextWord

Benötigt    : Ed

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Springt zum nächsten Wort.  Die aktuelle Wortdefinition
wird durch den Inhalt der Variable _WordDef angegeben.  Schauen Sie für
weitere Information bitte dort nach.  Kann keine nächstes Word gefunden
werden, so gibt NextWord einen Fehler zurück.

Siehe auch => LastWord
```

1.219 WRITE.guide/LastWord

```
LastWord
========

Aufruf      : LastWord

Benötigt    : Ed

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Springt zum nächsten Wort.  Die aktuelle Wortdefinition
wird durch den Inhalt der Variable _WordDef angegeben.  Schauen Sie für
weitere Information bitte dort nach.  Kann keine vorheriges Wort gefunden
werden, so gibt NextWord einen Fehler zurück.

Siehe auch => NextWord
```

1.220 WRITE.guide/PageUp

```
PageUp
======

Aufruf      : PageUp Percent/N

Benötigt    : Ed

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Blättert Percent Prozent einer dargestellten Seite nach
oben.  Beispiel:
     PageUp 75
Blättert 75% der dargestellten Seite nach oben.  Bei einem Fenster von
z.B. 30 Zeilen sind dies 21 Zeilen.

Siehe auch => PageDown
```

1.221 WRITE.guide/PageDown

```
PageDown
========

Aufruf      : PageDown Percent/N

Benötigt    : Ed

Setzt Fehler: Nein
```

```
Ergebnisse  : Nein

Beschreibung: Blättert Percent Prozent einer dargestellten Seite nach
unten.  Beispiel:
     PageDown 75
Blättert 75% der dargestellten Seite nach unten.  Bei einem Fenster von
z.B. 30 Zeilen sind dies 21 Zeilen.

Siehe auch => PageUp
```

1.222 WRITE.guide/ScrollUp

```
ScrollUp
========

Aufruf      : ScrollUp

Benötigt    : Ed

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Bewegt den Text unter dem Cursor nach unten.  nach unten.

Siehe auch => ScrollDown
```

1.223 WRITE.guide/ScrollDown

```
ScrollDown
==========

Aufruf      : ScrollDown

Benötigt    : Ed

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Bewegt den Text unter dem Cursor nach oben.  nach unten.

Siehe auch => ScrollUp
```

1.224 WRITE.guide/Goto

```
Goto
====

Aufruf      : Goto x/N y/N

Benötigt    : Ed

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Goto springt, wenn vorhanden zur y. Zeile und x. Spalte.
Dabei werden auch Positionsangaben mittels der Konstanten @CURSOR bis
@EOT unterstützt.

Siehe auch => Konstantenbeschreibung
```

1.225 WRITE.guide/GotoMouse

```
GotoMouse
=========

Aufruf      : GotoMouse Tags/T

Benötigt    : Ed

Setzt Fehler: ja

Ergebnisse  : Nein

Beschreibung: Positioniert den Cursor unter den Mauszeiger. Ist das Tag
@SAMEPOS gesetzt, wird der Cursor nur gesetzt, wenn sich die Maus bereits
über dem Cursor befindet. Dies ist in Zusammenhang mit Doppelklicks
(DoubleKey und SetMark nützlich.

Siehe auch =>
```

1.226 WRITE.guide/SetTextMark

```
SetTextMark
===========

Aufruf      : SetTextMark Nummer/N

Benötigt    : Ed

Setzt Fehler: Nein

Ergebnisse  : Nein
```

Beschreibung: SetTextMark setzt in der aktuellen Zeile eine (nämliche Nummer Nummer) temporäre Marke. Beachten Sie bitte, daß sich WRITE nur die Zeilennummer merkt, so daß nach Veränderungen im Text die Marke auf die falsche Stelle zeigt.

Siehe auch => GoTextMark

1.227 WRITE.guide/GoTextMark

GoTextMark
==========

Aufruf : GoTextMark Nummer/N

Benötigt : Ed

Setzt Fehler: Nein

Ergebnisse : Nein

Beschreibung: Springt zur angegebenen Textmarke.

Siehe auch => SetTextMark

1.228 WRITE.guide/Find

Find
====

Aufruf : Find VonX/N VonY/N BisX/N BisY/N Mode/T

Benötigt : Ed

Setzt Fehler: Ja

Ergebnisse : Ja

Beschreibung: Find sucht ab der aktuellen Cursorposition nach den Inhalt der Variable _FindString. Ist der Inhalt der Variable _CaseSense ungleich 0, so unterscheidet Find dabei zwischen Groß- und Kleinschreibung, ansonsten nicht. Kann das gesuchte Wort nicht gefunden werden, so wird ein Fehler zurückgegeben.

Ist der Inhalt der Variable _WordDef ungleich 0, so sucht Find nur nach ganzen Wörtern, ansonsten auch nach Wortteilen. Die Definition eines Wortes hängt von dem Inhalt der Variable _WordDef ab.

Kann das gesuchte Wort nicht gefunden werden, so wird ein Fehler zurückgegeben.

Ist in Mode @SILENT gesetzt, so wird das Nichtfinden des Wortes nicht mittels eines Requesters angezeigt.

Mittels der Tags @CASE, @NOCASE, @WORD und @NOWORD kann der Wert der Variablen _CaseSense und _WordDef für diesen Funktionaufruf direkt geändert werden, ohne daß globale Einstellungen verändert werden.

Über die Variablen VonX/Y und BisX/Y kann der Textbereich angegeben werden, auf den sich die Suche beziehen soll. @CURSOR @CURSOR @EOT @EOT steht z.B. für die Suche von der Cursorposition bis zum Ende des Textes. @MARKA @MARKA @MARKB @MARKB für die Suche in dem markierten Block.

Beachten Sie, daß, wenn VonX/Y gleich der Cursorposition ist, Find aus einsichtigen Gründen erst ab den nachfolgenden Buchstaben sucht. Dieses Verhalten kann durch das Setzten des Tags @FIRST abgestellt werden.

Wird das Tag @COUNT gesetzt, so springt Find nicht zum ersten Vorkommnis sondert die die zahl der gefundenen Wörter im angegebenen Bereich in _RN zurück.

Siehe auch => Replace, FindPattern, Konstantenbeschreibung

1.229 WRITE.guide/Replace

Replace
=======

Aufruf : Replace VonX/N VonY/N BisX/N BisY/N Mode/T

Benötigt : Ed

Setzt Fehler: Ja

Ergebnisse : Ja

Beschreibung: Replace sucht nach dem unter Find angegebenen Schema nach dem angegebenen Wort und ersetzt es durch den Inhalt der Variable _ReplaceString.

Ist @NOREQ nicht in Modegesetzt, so wird vorher ein Requester geöffnet, in dem das Ersetzen explizit bestätigt werden muß.

Ist auch hier in Mode @SILENT gesetzt, so wird das Nichtfinden des Wortes nicht mittels eines Requesters angezeigt.

Ist @ALL gesetzt, so ersetzt die Funktion nicht nur das nächste Vorkommen, sonndern alle Vorkommen bis des Ende des Textes erreicht ist oder die Suche abgebrochen wurde.

Ist bei gesetztem @ALL @NOREQ nicht in Modegesetzt, so wird bei jedem gefundenen Wort ein Requester geöffnet, in dem man Replace abbrechen kann, das aktuelle Wort uberspringen oder ersetzten kann.

```
Für die Parameter VonX/Y, BisX/Y sowie weitere Tags sehen Sie bitte bei
der Beschreibung der Funktion Find nach.

Die Zahl der Ersetzungen wird in _RN zurückgegeben.

Siehe auch => Find, FindPattern, Konstantenbeschreibung
```

1.230 WRITE.guide/ReplaceList

```
ReplaceList
===========

Aufruf      : ReplaceList xStart/N yStart/N xEnd/N yEnd/N Liste/N

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Ja

Beschreibung: Noch nicht implementiert.

Siehe auch => Konstantenbeschreibung
```

1.231 WRITE.guide/FindPattern

```
FindPattern
===========

Aufruf      : FindPattern Von,Bis/N Flags/T

Benötigt    : Ed

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Sucht von der Zeile von bis zur Zeile bis nach dem
DOS-Pattern, welches in der Variable _PatCase bzw.  _PatNoCase angegeben
ist. Dies hängt davon ab, ob @CASE oder @NOCASE in Flags angegeben
wird. Wird kein von beiden angegeben, so wird die aktuelle
Preferenceseinstellung benutzt. Durch das setzten von @SILENT kann die
Meldung des Nichtfindens des Patterns abgeschaltet werden.

Steht der Cursor in der Startzeile, so wird es ab der Position hinter dem
Cursor gesucht.

Siehe auch => Find, Replace, Konstantenbeschreibung
```

1.232 WRITE.guide/MatchBracket

```
MatchBracket
============

Aufruf      : MatchBracket

Benötigt    : Ed

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: MatchBracket sucht zu dem Zeichen unter dem Cursor das
konjugierte.  Unterstützt werden folgende Paare: ( ), { }, [ ], < >, «
».  MatchBracket unterstützt dabei auch Verschachtelungen.  Im
folgenden Beispiel springt MatchBracket, wenn der Cursor auf dem ersten
Zeichen steht, zum letzten:
     {
       {
         /* Dies ist ein Test */
         arg[0]:=0;
       }
     }
Desweiteren wird für die folgenden Zeichen, zum nächsten Vorkommen
gesprungen: ', `, ".

Eine interessante Tastaturbelegung ist:


     KEY ")"
       WRITETEXT ")"
       CURSORLEFT 0
       MATCHBRACKET
       DELAY 10
       MATCHBRACKET
       CURSORRIGHT 0
       ;

Was dafür sorgt, daß der Cursor, immer wenn man ) drückt, der Cursor
kurz zu entsprechenden öffnenden Klammer springt.

Siehe auch =>
```

1.233 WRITE.guide/Return

```
Return
======

Aufruf      : Return

Benötigt    : Ed
```

```
Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Bricht die aktuelle Zeile an der Cursorposition um.
Beispiel:
       sadlkjsadf asf gra htrhtrzrtgre
                              ^
                              Cursor

      wird zu...

       sadlkjsadf asf gra h
     trhtrzrtgre

Siehe auch => _AutoIndent
```

1.234 WRITE.guide/Delete

```
Delete
======

Aufruf      : Delete

Benötigt    : Ed

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Löscht das Zeichen unter dem Cursor.  Steht der Cursor
hinter dem letzten Zeichen einer Zeile, so wird die nächste Zeile in die
Aktuelle geholt.

Siehe auch =>
```

1.235 WRITE.guide/DeleteToEOL

```
DeleteToEOL
===========

Aufruf      : DeleteToEOL

Benötigt    : Ed

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: DeleteToEOL löscht die aktuelle Zeile von der aktuellen
```

```
Cursorposition bis zum Ende der Zeile.

Siehe auch => Delete, DeleteLine, UnDelLine
```

1.236 WRITE.guide/DeleteLine

```
DeleteLine
==========

Aufruf      : DeleteLine

Benötigt    : Ed

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: DeleteLine löscht die aktuelle Zeile.  Die zuletzt
gelöschte Zeile kann kann mit UnDelLine wieder eingefügt werden.
Desweitern wird sie an den Anfang der Liste DeleteLine-History angefügt.

Siehe auch => Delete, DeleteToEOL, UnDelLine
```

1.237 WRITE.guide/UnDelLine

```
UnDelLine
=========

Aufruf      : UnDelLine

Benötigt    : Ed

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: UnDelLine fügt die zuletzt mit DeleteLine gelöschte Zeile
wieder über der aktuellen ein. Ist das Tag @NODUP gesetzt, wird diese
Zeile gleichzeitig aus der Liste DeleteLine-History gelöscht.

Siehe auch => Delete, DeleteToEOL, DeleteLine
```

1.238 WRITE.guide/BackSpace

```
BackSpace
=========
```

1.239 WRITE.guide/Tab

```
Tab
===

Aufruf      : Tab Mode/N

Benötigt    : Ed

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Tab springt zur nächsten Tabulatormarke.  Diese sind
gerade _TabLength Zeichen von einander entfehrnt.  Ist Mode=0, so springt
nur der Cursor zum Tabulator.  Ist Mode=1 so der Text unter dem Cursor
bis zum Tabulator bewegt.

Siehe auch => BackTab
```

1.240 WRITE.guide/BackTab

```
BackTab
=======

Aufruf      : BackTab

Benötigt    : Ed

Setzt Fehler: Nichts

Ergebnisse  : Nichts

Beschreibung: BackTab springt zur vorherigen Tabulatormarke.  Diese sind
gerade _TabLength Zeichen von einander entfehrnt.

Siehe auch =>
```

1.241 WRITE.guide/UpperBlock

```
UpperBlock
==========

Aufruf      : UpperBlock VonX,VonY,BisX,BisY/N

Benötigt    : Ed

Setzt Fehler: Nein

Ergebnisse  : Nein
```

Beschreibung: Konvertiert alle Buchstaben im angebenen Textbereich in Großbuchstaben. Dabei werden auch internationale Zeichen je nach der aktuellen Ländereinstellung des Betriebssystems berücksichtigt.

Die Positionen werden dabei mittels der Konstanten @CURSOR bis @EOT angegeben.

Siehe auch => LowerBlock, Konstantenbeschreibung

1.242 WRITE.guide/LowerBlock

LowerBlock
==========

Aufruf : LowerBlock VonX,VonY,BisX,BisY/N

Benötigt :

Setzt Fehler:

Ergebnisse :

Beschreibung: Konvertiert alle Buchstaben im angegebenen Textbereich zu Kleinbuchstaben. Dabei werden auch internationale Zeichen je nach der aktuellen Ländereinstellung des Betriebssystems berücksichtigt.

Die Positionen werden dabei mittels der Konstanten @CURSOR bis @EOT angegeben.

Siehe auch => UpperBlock, Konstantenbeschreibung

1.243 WRITE.guide/WriteChar

WriteChar
=========

Aufruf : WriteChar DecimalCode/N

Benötigt : Ed

Setzt Fehler: Nein

Ergebnisse : Nein

Beschreibung: WriteChar schreibt das Zeichen mit dem ASCII-Code DecimalCode an die aktuelle CursorPosition. Das Zeichen geht dabei an die gleiche Routine die auch für die über die Tastatur eingegebenen Zeichen zuständig ist. D. h. alle nicht dastellbaren Zeichen, werden auch nicht gedruckt. Diese Funktion ist auch völlig unabhängig von der

aktuellen Tastaturbelegung. Ist A mit einer Funktion belegt, so wird
WriteChar 65 dennoch ein A schreiben.

Siehe auch => WriteText

1.244 WRITE.guide/WriteText

WriteText
=========

Aufruf   : WriteText Zeichenkette/S

Benötigt   : Ed

Setzt Fehler: Nein

Ergebnisse : Nein

Beschreibung: WriteText funktioniert genauso wie WriteChar. Nur wird
hier ein ganzer String ausgegeben. Auch hier weren nicht dastellbare
Zeichen herausgefiltert. Die Möglichkeit der Angabe von Steuersequencen
in der Zeichenkette besteht also nicht.

Siehe auch => WriteChar

1.245 WRITE.guide/SwapChar

SwapChar
========

Aufruf   : SwapChar einsX, einsY, zweiX, zweiY/N

Benötigt   : Ed

Setzt Fehler: Ja

Ergebnisse : Nein

Beschreibung: Vertauscht das Zeichen an der Position eins mit dem an der
Position zwei.

Siehe auch =>

1.246 WRITE.guide/Key

```
Key
===

Aufruf      : Key Definition/S Befehlsfolge/F

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Key belegt eine beliebige Taste mit einer Funktionsfolge.
Bei Definition handelt es sich um eine Tastatur- oder Mausbeschreibung,
wie sie auch für die Commodities des Betriebssystem verwendet wird.
Schauen Sie bitte für eine genauere Beschreibung in Ihren Handbüchern
unter dem Stichwort Commodities nach.  Erlaubt ist momentan nur der Typ
rawkey.

Siehe auch => DoubleKey
```

1.247 WRITE.guide/DoubleKey

```
DoubleKey
=========

Aufruf      : DoubleKey Definition1,Definition2/S Befehlsfolge/F

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Diese Funktion arbeitet im Prinzip genauso, nur das hier
die Befehlsfolge nur dann ausgeführt wird, wenn erst die Befehlsfoge 1
und dann im Zeitintervall eines Doppellickes die Definition 2 (die
unterschiedlich zu der Definition 1 sein kann) gedrückt wird.

Siehe auch => Key
```

1.248 WRITE.guide/ClearKeys

```
ClearKeys
=========

Aufruf      : ClearKeys

Benötigt    : Nichts
```

```
Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Löscht die komplette mit Key eingegebene Tastaturbelegung.
Diese Funktion sollte mit Vorsicht benutzt werden, da nach ihrem Aufruf
ein sinnvolles Arbeiten mit WRITE kaum noch möglich ist.

Siehe auch => Key
```

1.249 WRITE.guide/SetHotKey

```
SetHotKey
=========

Aufruf      : SetHotKey Nummer/N Tastensequenz/S Funktion/F

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: WRITE besitzt intern eine Reihe von Hotkeys, die mit einer
beliebigen Tastenkombinationsbeschreibung über die commodities.library
mit einer beliebigen WRITE-internen Funktion belegt werden können.  Da
die commodities.library benutzt wird, muß die Tastensequenz den
Richtlinien, die auch für alle anderen Commodities gelten, folgen.  Für
das genaue Format der Tastensequenz schauen Sie demnach in Ihrer
Dokumentation, die Sie beim Kauf Ihres Computers bekommen haben, nach.
Funktion ist eine beliebige Funktionsfolge.  Ein bereits bestehender
HotKey wird durch eine Redefinition überschrieben.

Siehe auch =>
```

1.250 WRITE.guide/ClearHotKey

```
ClearHotKey
===========

Aufruf      : Nummer/N

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Löscht einen definierten HotKey.
```

Siehe auch => DoBuffer, DoString, ParseBuffer

1.251 WRITE.guide/Menu

Menu
====

Aufruf : Menu Titel/S

Benötigt : Nichts

Setzt Fehler: Nein

Ergebnisse : Nein

Beschreibung: Menu hängt an die Menüleiste ein neues Menü mit dem
Titel Titel an.

Siehe auch => Item, Sub, ItemBar, SubBar, ClearMenu

1.252 WRITE.guide/Item

Item
====

Aufruf : Item Titel/S ShortCut/S Funktionliste/F

Benötigt : Nichts

Setzt Fehler: Nein

Ergebnisse : Nein

Beschreibung: Item hängt an den zuletzt initialisierten Menü einen
Menüpunkt mit dem Titel Titel und dem Shortcut ShortCut an und belegt
diesen mit der angegeben Funktionsliste. Diese wird immer dann
automatisch ausgeführt, wenn der Menüpunkt selektiert wird. Anstelle
eines ShortCut-Buchstabens kann auch eine Kommodities-Tastenbeschreiben
angegeben werden. Diese erscheint dann ab OS 3.0 neben dem Menüpunkt.
Gleichzeitig wird die entsprechenden Taste mit der Funktion des
Menüpunktest belegt.

Wird (ab OS3.0) die Helptaste während der Mauszeiger über einem Menü
ist gedrückt, so wird ein Requester geöffnet, der die Funktionsbelegung
dieses Menüpunktes zeigt. c

Siehe auch => Menu, Sub, ItemBar, SubBar, ClearMenu

1.253 WRITE.guide/Sub

```
Sub
===

Aufruf      : Sub Titel/S ShortCut/S Funktionliste/F

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Sub hängt an den zuletzt initialisierten Menüpunkt ein
Untermenü mit dem Titel Titel und dem Shortcut ShortCut an und belegt
diesen mit der angegeben Funktionsliste.  Diese wird immer dann
automatisch ausgeführt, wenn das Untermenü selektiert wird.

Weitere Informationen, siehe Item

Für Menüpunkte gibt es auch eine Onlinehilfe. Siehe Item.

Siehe auch => Menu, Item, ItemBar, SubBar, ClearMenu
```

1.254 WRITE.guide/ItemBar

```
ItemBar
=======

Aufruf      : Itembar

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Hängt an den letzten Menüpunkt einen nicht selektierbaren
Querstricht an.  Diese Funktion sollte zur optischen Aufteilung eines
Menüs benutzt werden.

Siehe auch => Menu, Item, Sub, SubBar, ClearMenu
```

1.255 WRITE.guide/SubBar

```
SubBar
======

Aufruf      : SubBar
```

```
Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Hängt an das letzte Untermenü einen nicht selektierbaren
Querstricht an.  Diese Funktion sollte zur optischen Aufteilung eines
Menüs benutzt werden.

Siehe auch => Menu, Item, Sub, ItemBar, ClearMenu
```

1.256 WRITE.guide/ClearMenu

```
ClearMenu
=========

Aufruf      : ClearMenu

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Löscht die komplette eingegebene Menübelegung.  Diese
Funktion sollte mit Vorsicht benutzt werden, da nach ihrem Aufruf ein
sinnvolles Arbeiten mit WRITE kaum noch möglich ist.

Siehe auch => Menu, Item, Sub, ItemBar, SubBar
```

1.257 WRITE.guide/WaitPointer

```
WaitPointer
===========

Aufruf      : WaitPointer

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Setzt für alle Fenster den Warte-Mauszeiger.  Sinnvoll,
wenn z.B. ein Makro oder REXX-Script etwas länger dauert und man dem
Benutzer zeigen möchte, daß WRITE noch arbeitet und nicht etwa
abgestürzt ist.

Siehe auch => NormalPointer
```

1.258 WRITE.guide/NormalPointer

```
NormalPointer
=============

Aufruf      : NormalPointer

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Setzt Setzt den Mauszeiger wieder in den Normalzustand.

Siehe auch => WaitPointer
```

1.259 WRITE.guide/DoREXX

```
DoREXX
======

Aufruf      : DoREXX DateiName/S Console/S Flags/T

Benötigt    : REXX-Libraries im libs:-Verzeichnis. RX-Befehl.

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Führt die angegebe Datei als REXX-Script aus.  Je nach
dem, ob @ASYNC in Flags gesetzt ist oder nicht, wartet WRITE auf
Beendigung des Scripts oder nicht.  Soll das Script auf auch Kommandos an
WRITE schicken, so sollte @ASYNC gesetzt (das Script wartet sonst
darauf, daß WRITE die Befehle ausführt, während WRITE darauf wartet
das das Script ausgeführt wird), außerdem sollte man vorher eventuell
LockWindow aufrufen, da die Befehle immer an das aktive Fenster gehen und
wenn während der Ausführung ein anderes Fenster aktiviert wird, gibt es
ein heilloses Chaos.

Mittels des Strings Console kann eine Ein-Ausgabedatei für das gestartete
REXX-Skript angegeben werden. Beispiel:

     DOREXX "listdemo.wrx" "CON:0/0/640/256/WRITEConsole/AUTO/QUIT" {@ASYNC}

Siehe auch => LockWindow, OpenPort, ClosePort, WaitPort
```

1.260 WRITE.guide/LockWindow

```
LockWindow
==========

Aufruf      : LockWindow ID/N

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: LockWindow lockt den Ed mit dem zugehörigen ID.  Das
heißt, daß alle Befehle, die über die REXX-Schnittstelle kommen, auf den
gelockten Ed wirken, selbst wenn zwischendurch andere Fenster aktiviert
werden etc.

Ein gelocktes Fenster sollte mit LockWindow 0 vor Verlassen des
REXX-Script wieder entlockt werden, da sonst Scripte, die ein Fenster
nicht explizit locken, alle Nachrichten ebenfalls an diesen Ed schicken.
Ein erneuter Aufruf von LockWindow überschreibt einen alten Lock.  Kann
der angegebene ID nicht gefunden werden, so wird ein Fehler
zurückgegeben.

Sollen mehrere Script gleichzeitig gestartet werden und diese auf
verschiedene Fenster wirken, so benutzen Sie bitte die Funktionen
OpenPort, ClosePort, WaitPort.

Siehe auch => OpenPort, ClosePort, WaitPort
```

1.261 WRITE.guide/NextEd

```
NextEd
======

Aufruf      : NextEd ID/N

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Aktiviert den nächsten Ed, dessen ID nach dem angegebenen
kommt.  Kann kein weiterer ID gefunden werden, so gibt NextEd einen Fehler
zurück.

Siehe auch =>
```

1.262 WRITE.guide/OpenPort

```
OpenPort
========

Aufruf      : OpenPort ID/N

Benötigt    : Ed

Setzt Fehler: Ja

Ergebnisse  : Ja

Beschreibung: OpenPort öffnet einen persönlichen REXX-Port für den Ed
mit dem angegebenen ID.  Der Name des Ports wird in _RS zurückgegeben.
Kann der Port nicht geöffnet werden (ein Task kann nur eine begrenzte
Zahl von Ports verwalten) so wird ein Fehler zurückgegeben.

Bitte beachten Sie das man für einen Ed nicht eines zweiten PrivtaePort
öffnen kann. Ein zweiter Aufruf von OpenPort gibt nur einenFehler
zurück.

Siehe auch => ClosePort, WaitPort
```

1.263 WRITE.guide/ClosePort

```
ClosePort
=========

Aufruf      : ClosePort ID/N

Benötigt    : Ed

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: ClosePort schließt den mit OpenPort geöffneten REXX-Port
wieder.  Kann der Ed mit dem angegebenen ID nicht gefunden werden, oder
existiert kein Port, so wird ein Fehler zurückgegeben.

Siehe auch => OpenPort, WaitPort
```

1.264 WRITE.guide/WaitPort

```
WaitPort
========

Aufruf      : WaitPort ID/N
```

```
Benötigt    : Ed

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Wird dieser Befehl von einem REXX-Script aus aufgerufen, so
kehrt er erst zurück wenn der REXX-Port des Ed geschlossen wird.  Dies
passiert im algemeinen, wenn der Ed verlassen (d.h. meisten, das Fenster
geschlossen) wird. Dabei darf der Befehl nicht an den globalen, sondern
an den lokalen, mit OpenPort geöffneten, Port geschickt werden.

Siehe auch => OpenPort, ClosePort
```

1.265 WRITE.guide/ModifyWin

```
ModifyWin
=========

Aufruf      : ModifyWin Mode/N

Benötigt    : Fenster

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Mit ModifyWin läßt sich ein offenes Fenster auf
verschiedene Weise manipulieren.  Mode=0: Fenster wird gezipt.  Mode=1:
Fenster wird aktiviert.  Mode=2: Fenster wird nach vorne gebracht.
Mode=3: Fenster wird nach hinten gebracht.

Siehe auch => ModifyScreen
```

1.266 WRITE.guide/ModifyScreen

```
ModifyScreen
============

Aufruf      : ModifyScreen Mode/N

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Mit ModifyScreen läßt sich der Screen, auf welchen alle
Fenster und Requester geöffnet werden, nach vorne und nach hinten
bringen.  Mode=0: Screen nach vorne.  Mode=1: Screen nach hinten.
```

Siehe auch => ModifyWin

1.267 WRITE.guide/SetREXXClip

```
SetREXXClip
===========

Aufruf      : SetREXXClip ClipName/S Typ/N Nummer/N

Benötigt    : Nichts/Ed

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: SetREXXClip kopiert den Inhalt einer Zeile/eines Puffers in
das REXXClipboard unter dem angegebenen Namen, wo sie dann mit
getclip(Name) wider ausgelesen werden kann. Ist Typ=0, so wird die Zeile
mit der angegebenen Nummer gelesen. Ist Typ= 1, so wird der angegebene
Puffer ins REXXClipboard geschrieben. Dabei markieren Linefeets den
Zeilenumbruch. Ist beim Auslesen der Zeile kein Ed aktiviert, oder
existiert die angegebene Zeile/der angegebene Puffer nicht, so wird ein
Fehler zurückgegeben.

Siehe auch =>
```

1.268 WRITE.guide/GetConfig

```
GetConfig
=========

Aufruf      : GetConfig Name/S

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: GetConfig aktiviert die Konfiguration mit dem angegebenen
Namen, damit auf dessen Einstellungen/Variabeln zurückgegriffen werden
kann. GetConfig lädt nicht nocht nicht geladene Konfigurationen nach,
auch kann nicht "" für die Standardkonfiguration angegeben werden.

Siehe auch =>
```

1.269 WRITE.guide/SetREXXVar

```
SetREXXVar
==========

Aufruf      : SetREXXVar Name/S Which/N Welcher/S

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Schreibt den Inhalt einiger WRITE-internen Werte direkt in
eine AREXX-Variable. Dadurch wird die 256 Bytes Limtierung des Parsers
von WRITE bei Strings umgangen.

Which=0 schreibt den Inhalt der Zeile Welcher in die angegebene Variable.
Which=1 schreibt den Inhalt des Buffers Welcher in die Variable.

Siehe auch => GetREXXVar
```

1.270 WRITE.guide/GetREXXVar

```
GetREXXVar
==========

Aufruf      : GetREXXVar Name/S Which/N Welcher/S

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Im Gegensatz zu SetREXXVar wird hier der Inhalt einer
REXX-Variablen in eine WRITE-interne Varaile geschrieben. Die
Übergabeparameter sind die gleichen wie bei SetREXXVar

Siehe auch =>
```

1.271 WRITE.guide/Refresh

```
Refresh
=======

Aufruf      : Refresh

Benötigt    : Fenster
```

```
Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Aktuallisiert das EditorFenster, d.h. der sichtbare Text
wird neu geschrieben.

Siehe auch =>
```

1.272 WRITE.guide/ChangeConfig

```
ChangeConfig
============

Aufruf      : ChangeConfig ConfigName/S

Benötigt    : Ed

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Ändert die Konfiguration eines Eds auf die Konfiguration
ConfigName. Die genannte Konfiguration wird ggf. nachgeladen.  Diese
Funktion ist nützlich, wenn man zum Beispiel mit einer minimalen
Konfiguration arbeitet (Mailer etc.) und die Leistungfähigkeit einer
komplexeren Konfiguration braucht.

Siehe auch =>
```

1.273 WRITE.guide/SaveGConfig

```
SaveGConfig
===========

Aufruf      : SaveGConfig

Benötigt    : Nein

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Aus den aktuellen globalen Einstellungen wird eine neue
startup.config und startup.config.pp.

Siehe auch => SaveConfig
```

1.274 WRITE.guide/SaveConfig

```
SaveConfig
==========

Aufruf      : SaveConfig Name/S

Benötigt    : Nein

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Aus den aktuellen lokalen Einstellungen, die zu einer
Konfiguration gehören, wird eine neue Konfiguration mit dem angegebenen
Namen erzeugt.

Siehe auch => SaveGConfig
```

1.275 WRITE.guide/MacroRec

```
MacroRec
========

Aufruf      : MacroRec

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: MacroRec startet die Aufzeichnung eines Macros.  D. h. Alle
Komandos nach dem Befehl MacroRec bis ausschließlich dem Befehl
MacroStop werden aufgezeichnet.  Dabei sind ein paar Dinge zu
beachten:
  1. Mausbewegungen, sowie Aktionen die mit Mausdrücken verbunden sind,
     als auch Bewegungen am Scrollbalken werden nicht aufgezeichnet.

  2. Fensterwechsel etc. werden nicht aufgezeichnet.

  3. Der MacroSaver speichert nicht den (im Konfigurationsfile
     angegebenen) Quelltext, sondern den erzeugten Zwischencode.  So
     haben z.B. Variablen den Wert zum Zeitpunkt des Abspeicherns nicht
     den zum Zeitpunkt der Macroausführung.

  4. Macros können nicht rekursiv verschachtelt werden.  Erneutes
     Aufrufen von MacroRec löscht das alte und beginnt ein neues Makro.

Siehe auch => MacroStop, MacroPlay
```

1.276 WRITE.guide/MacroStop

```
MacroStop
=========

Aufruf      : MacroStop

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Stopt die Aufzeichnung eines Makros.

Siehe auch => MacroRec, MacroPlay
```

1.277 WRITE.guide/MacroPlay

```
MacroPlay
=========

Aufruf      : MacroPlay Anzahl/N

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Führt das aufgezeichnete Makro Anzahl mal aus.

Siehe auch => MacroRec, MacroStop
```

1.278 WRITE.guide/SetPannel

```
SetPannel
=========

Aufruf      : SetPannel Nummer/N Name/S Funktionsliste/F

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Mittels SetPannel ist man in der Lage den Text und die
Funktion für ein Gadget des Pannels zu definieren.
```

Siehe auch => Pannel

1.279 WRITE.guide/Pannel

Pannel
======

Aufruf : Pannel

Benötigt : Nichts

Setzt Fehler: Nein

Ergebnisse : Nein

Beschreibung: Nach dem Aufruf dieser Funktion öffnet sich ein Requester, in dem alle mit SetPannel definierten Makros namentlich in Form von einem Gadgetpannel aufgeführt werden. Durch drücken eines Gadgets startet man das entsprechende Makro. Das MacroPannel kann auch mit einem Doppelklick der rechten Maustaste geöffnet werden.

Durch das Setzen des Tags @TOMOUSE wird der Requester zentriert unter dem Mauszeiger geöffnet. Siehe dazu auch das Kapitel Requester

Siehe auch => SetPannel, Requester

1.280 WRITE.guide/Macro

Macro
=====

Aufruf : Macro Name/S Funktionen/F

Benötigt : Nichts

Setzt Fehler: Nein

Ergebnisse : Nein

Beschreibung: Mittels dieses Befehles ist es möglich, neue, parameterlose Funktionen zu deklarieren, die genauso wie die internen Funktionen gehandhabt werden können. Name ist dabei der Name des Makros. Bereits deklarierte Makros können auch durch Neudefinition überschrieben werden.

Siehe auch =>

1.281 WRITE.guide/Undo

```
Undo
====

Aufruf      : Undo Count/S

Benötigt    : Ed

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Mittels dieses Befehles, kann man die angegebene Anzahl von
Textveränderungen rückgängig machen. Beachten Sie, daß die maximale
Anzahl der gespeicherten Veränderungen vom Wert der Variablen _Undo
abhängt.

Siehe auch =>
```

1.282 WRITE.guide/ClearList

```
ClearList
=========

Aufruf      : ClearList List/N

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Lösche die angegebene Liste.

Siehe auch =>
```

1.283 WRITE.guide/AddList

```
Addlist
=======

Aufruf      : AddList String/S List/N Tags/T

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Hängt an die angegebene Liste den String String an. Ist in
```

```
Tags @NODUP gesetzt, wird der String nun an die Liste angehängt, wenn
er nicht schon bereits vorhanden ist.

Siehe auch =>
```

1.284 WRITE.guide/RemoveList

```
RemoveList
==========

Aufruf      : RemoveList Name/S Element/N

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Löscht das n-te Element aus der angegebenen Liste. Die
Elemente weden von 1 bis zur Größe der Liste heraufgezählt.

Siehe auch =>
```

1.285 WRITE.guide/Push

```
Push
====

Aufruf      : Push Name/S Eintrag/S

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Fügt den Eintrag an Position 1 der Liste ein.

Siehe auch =>
```

1.286 WRITE.guide/Pop

```
Pop
===

Aufruf      : Pop Name/S
```

```
Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Ja

Beschreibung: Gibt den Inhalt des ersten Elementes der Liste in _RS
zurück und löscht den Eintrag anschließend.

Siehe auch =>
```

1.287 WRITE.guide/ShowList

```
ShowList
========

Aufruf      : ShowList Lists/N Zahl/N Tags/T

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Ja

Beschreibung: Zeigt die angegebene Liste mittels eines Listerequester.
Ist @Select in Tags gesetzt, so kann einer der Strings selektiert
werden. Ist ein String selektiert worden, so gibt ShowList keinen Fehler
zurück. Der String ist aus der Variable _RS auslesbar.

Siehe auch =>
```

1.288 WRITE.guide/ListToBuffer

```
ListToBuffer
============

Aufruf      : ListToBuffer List/N Buffer/N

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Konvertiert die angegebene list zu einem Block.

Ist die Liste leer, so wird ein Fehler zurückgegeben.

Siehe auch => BufferToList
```

1.289 WRITE.guide/BufferToList

```
BufferToList
============

Aufruf      : BufferToList Buffer/N List /N

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Konvertiert den angegebenen Buffer in eine Liste.

Ist der angegebene Buffer leer, so wird ein Fehler zurückgegeben.

Folds werden nicht konvertiert.

Siehe auch => ListToBuffer
```

1.290 WRITE.guide/DoList

```
DoList
======

Aufruf      : DoList

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Noch nicht implementiert.

Siehe auch =>
```

1.291 WRITE.guide/ListSize

```
ListSize
========

Aufruf      : ListSize List/N

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Ja
```

```
Beschreibung: Gibt die Zahl der Einträge in der angegebenen Liste
zurück.

Siehe auch =>
```

1.292 WRITE.guide/GetListEntry

```
GetListEntry
============

Aufruf      : GetListEntry List/N Entry/N

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Ja

Beschreibung: Die den angegebenen Eintrag in einer Liste in _RS zurück.

Siehe auch =>
```

1.293 WRITE.guide/FindListEntry

```
FindListEntry
=============

Aufruf      : FindListEntry List/N String/S

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Ja

Beschreibung: Such nach dem Eintrag String und gibt, wenn gefunden, seine
Position in der Liste zurück.

kann der Eintrag nicht gefunden werden, wird ein Fehler zurückgegegen.

Siehe auch =>
```

1.294 WRITE.guide/Exists

```
Exists
======

Aufruf      : Exists FileName/F

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Schaut nach, ob der angegebene File existiert. Wenn nicht,
wird ein Fehler zurückgegeben.  Siehe auch =>
```

1.295 WRITE.guide/Delay

```
Delay
=====

Aufruf      : Delay Time/N

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Wartet die angegebene Zahl von 1/50 Sekunden. Auf Grund des
Parser/Executeroverheads von WRITE, wartet diese Funtion natürlich nie
genau die angegebene Zeit.

Siehe auch =>
```

1.296 WRITE.guide/GuideHelp

```
GuideHelp
=========

Aufruf      : GuideHelp Keyword/S

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Dieser Befehl versucht die Anleitung im Guide-Format über
AmigaGuide bzw. MultiView zu laden und versucht eine Referenz zu dem
angegebenen Wort zu finden.
```

```
Siehe auch =>
```

1.297 WRITE.guide/VersionCheck

```
VersionCheck
============

Aufruf      : VersionCheck Version/N Name/S

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Kontrolliert, ob die übergebene Versionsnummer kompatibel
zur der aktuellen Versionsnummer von WRITE ist. Ist dem nicht so, wird ein
Requester geöffnet, der darauf hinweist, daß das angegebene
Programme/Konfiguration etc. möglicherweise nicht kompatibel ist. Der
Benutzer kann dann entscheiden, ob er weiter machen, oder abbrechen
möchte. Bircht er ab, gibt VersionCheck einen Fehler zurück. Durch
Setzten des Tags @SILENT kann der Requester unterbunden werden.
VersionChec gibt dann bei unterschiedlichen Versionsnummern gleich einen
Fehler zurück.

Die aktuelle Kompatibilitätsversionsnummer steht auch in der Variablen
_Version.

Siehe auch =>
```

1.298 WRITE.guide/Inc

```
Inc
===

Aufruf      : Inc Zahl,Wert/N

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Ja

Beschreibung: Erhöht Zahl um Wert. Das Ergebnis wird in _RN
zurückgegeben.

Siehe auch => Dec, RangeCheck, RangeRound
```

1.299 WRITE.guide/Dec

```
Dec
===

Aufruf      : Dec Zahl,Wert/N

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Ja

Beschreibung: Erniedrigt Zahl um Wert. Ist das Ergebnis kleiner als null,
so wird ein Fehler zurückgegeben, ansonsten enthlt _RN das Ergebnis.

Siehe auch => Inc, RangeCheck, RangeRound
```

1.300 WRITE.guide/RangeCheck

```
RangeCheck
==========

Aufruf      : RangeCheck Zahl,Min,Max/N

Benötigt    : Nichts

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Liegt Zahl außerhalb des durch Min und Max angegebenen
Intervalls, so wird ein Fehler zurückgegeben.

Siehe auch => Inc, Dec, RangeRound
```

1.301 WRITE.guide/RangeRound

```
RangeRound
==========

Aufruf      : RangeRound Zahl,Min,Max/N

Benötigt    : Nichts

Setzt Fehler: Nein

Ergebnisse  : Ja

Beschreibung: Ist Zahl < Min, so wird Zahl gleich Min gesetzt. Ist Zahl >
```

```
Max, so wird Zahl gleich Max gesetzt. Ansonsten bleibt Zahl unverändert.
Das Ergebnis wird in _RN zurückgegeben.

Beispiel: Setzten des Cursor 12 Zeilen niedriger


     INC _YPos 12
     RANGECHECK _RN 1 _Length
     GOTO _xPos _RN

Siehe auch =>
```

1.302 WRITE.guide/Begin

```
Begin
=====

Aufruf      : Begin Funktionsliste/F

Benötigt    : Konfiguration

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Die angegebene Funtionsliste wird ausgeführt, wenn eine
Konfiguration geladen wird.  Sollte am Anfang der Konfiguration stehen.
Hier können manuell Befehle eingefügt werden, die durch die
Benutzeroberfläche nicht direkt programmiert werden können. (Was nicht
heißt, daß man den Inhalt der Funktionsliste nicht über die
Oberfläche geändert werden kann). Alle manuell eingefügten Befehle
müssen hier eingefügt werden, da sie an anderer Stelle eingefügt, von
WRITE vergessen, und beim nächsten Abspeichern der Konfiguration
vergessen werden.

Siehe auch => Screen, PublicScreens
```

1.303 WRITE.guide/Close

```
Close
=====

Aufruf      : Close Funktionsliste/F

Benötigt    : Konfiguration

Setzt Fehler: Nein

Ergebnisse  : Nein
```

Beschreibung: Hier können Funktionen angegeben werden, die ausgeführt
werden, wenn die Konfiguration oder der Editor beendet werden. Hier kann
z.B. ein Screenmanager dazu bewegt werden, einen Screen zu schließen.

Siehe auch => Screen, PublicScreens

1.304 WRITE.guide/Start

Start
=====

Aufruf : Start Funktionsliste/F

Benötigt : Nichts

Setzt Fehler: Nein

Ergebnisse : Nein

Beschreibung: Die angegebene Funtionsliste wird ausgeführt, wenn der
Editor gestartet wird. Sollte am Anfang der STARTUP.CONFIG stehen. Hier
können manuell Befehle eingefügt werden, die durch die
Benutzeroberfläche nicht direkt programmiert werden können. (Was nicht
heißt, daß man den Inhalt der Funktionsliste nicht über die
Oberfläche geändert werden kann). Alle manuell eingefügten Befehle
müssen hier eingefügt werden, da sie an anderer Stelle eingefügt, von
WRITE vergessen, und beim nächsten Abspeichern der Konfiguration
vergessen werden.

Siehe auch =>

1.305 WRITE.guide/Quit

Quit
====

Aufruf : Quit Funktionsliste/F

Benötigt : Nichts

Setzt Fehler: Nichts

Ergebnisse : Nichts

Beschreibung: Hier können Funktionen angegeben werden, die ausgeführt
werden, wenn der Editor beendet wird. Hier kann z.B. ein Screenmanager
dazu bewegt werden, einen Screen zu schließen.

Siehe auch =>

1.306 WRITE.guide/Fold

```
Fold
====

Aufruf      : Fold Von/N Bis/N

Benötigt    : Ed

Setzt Fehler: Ja

Ergebnisse  : Nein

Beschreibung: Fold faltet den Text von Zeile Von bis Zeile Bis.
Verschachtelte Falten sind möglich.  Siehe auch => Konstantenbeschreibung
```

1.307 WRITE.guide/UnFold

```
UnFold
======

Aufruf      : Unfold Von/N Bis/N Ebenen/N

Benötigt    : Ed

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Entfaltet in dem angegebenen Bereich alle Falten bis zur
angegebenenTiefe.

Siehe auch => Konstantenbeschreibung
```

1.308 WRITE.guide/AutoFold

```
AutoFold
========

Aufruf      : AutoFold

Benötigt    : Ed

Setzt Fehler: Nein

Ergebnisse  : Nein

Beschreibung: Such im ganzen Text nach Foldmarkierungen und faltet die
entsprechenen Textteile.
```

Siehe auch =>

1.309 WRITE.guide/ReFold

ReFold
======

Aufruf : ReFold

Benötigt : Ed

Setzt Fehler: Nein

Ergebnisse : Nein

Beschreibung: Schaut, ob der Cursor zwischen zwei Faltmarken steht und faltet den entsprechenden Block.

Können keine Markierungen gefunden werden, wird ein Fehler zurückgegeben.

Siehe auch =>

Hier endet die Anleitung...

Viel Spaß...

TIM

1.310 WRITE.guide/RexxScripts

RexxScripts
****

Für die Beschreibung der hier nicht verzeichneten Skript schauen Sie bitte im Header der entsprechenden Skripts nach.

* autodoc.wrx Versucht zu dem Wort unter dem Cursor über AmigaGuide bzw. MultiView Informationen zu bekommen.

* ClearUmlauts.wrx Wandelt alle deutschen Umlaute in der Form ä => ae etc. um.

* CloseDown.wrx Schließt alle Fenster nach Sicherheitsabfrage und beendet WRITE.

* define.wrx Ein Script für den SAS C-Compiler. Renumiert #defines (z.B. für Lcale-Files) und sucht zu dem Wort unter dem Cursor modulübergreifend das entsprechenden #define.

* ListeDemo.wrx Demonstriert den Umgang mit den WRITE-internen Funktionen zur Listenverwaltung.

* locale.wrx Diverse Routinen für das Arbeiten mit Locale-Files für den SAS C-Compiler.

* OberonError.wrx Script für den Oberon2-Compiler der Firma A+L. Zeigt die Fehler im Quelltext an.

* PrivatePortDemo.wrx Demonstriert den Umgang mit PrivatePorts. Siehe auch OpenPort, ClosePort und WaitPort.

* Renumber.wrx Renummiert eine eine Reihe untereinanderstehender Zahlen so, daß sie sowohl rechtsbündig als auch aufeinanderfilgend sind.

* ResortIcon.wrx Resortiert iconifizierte Fenster, so daß Lücken zwischen den Fenstern verschwinden.

* sc.wrx Komfortable Fehler-Anzeige für den SAS c_Compiler.

* scman.wrx Zeigt den entsprechende AmigaGuideeintrag zu der SAS-Libraryfunktion unter dem Cursor.

* SetRexxClipDemo.wrx

  Demonstriert die Funktion SetREXXClip. Mit dieser Funktion ist es möglich. Zeilen und Buffer beliebiger Länge nach REXX zu exportieren.

* ShowConfig.wrx Zeigt alle geladenen Konfigurationen.

* templates.wrx Erzeugt automatische Funktionsbeschreibungen / Autodocsfür / Manpages für den SAS C-Compiler.

* texadr.wrx Script, um Addressen von DFA nach WRITE im TeX-Format zu importieren.

1.311 WRITE.guide/Index

Index
*****

Credits	Danksagungen
Danksagungen	Danksagungen
Das Prinzip	Das Prinzip
Demoversion	Demoversion
Der Ed	Der Ed
Einleitung	Einleitung
EMail	Autor
Fehlerreport	Autor
Fred Fish	Demoversion
Funktionen	Funktionsbeschreibung
Funktionslisten	Typen
Geld	Autor
GUIDEMODE	ToolTypes
Haftung	Allgemein
Index	Index
Installation	Installation
Installer	Installation
Kaufkonditionen	Vollversion
KaufPreis	Vollversion
Kommentare	Autor
Konfigurationsfile	Installation
Konstanten	Konstanten
Konstantenbeschreibung	Konstantenbeschreibung
Konzept	Konzept
Kritik	Autor
Libraries	Installation
Listen	Listen und Buffer
MausNet	Autor
Mauszeiger	Editorfenster
Mousepointer	Editorfenster
Neue Version	NeueVersion
OS 2.0	Installation
Preis	Vollversion
Public Screens	PublicScreens
QuickWrite	Starten
Registrierung	Registrierung
Requester	Requester
RexxScripts	RexxScripts
Screenmanager	PublicScreens
Scrollbalken	Editorfenster
Scrollgadgets	Editorfenster
Shortcuts	Requester
SLEEPMODE	ToolTypes
SOUND	ToolTypes
Speicher	Installation
Stack	Installation
Strings	Typen
Syntax	Syntax
System	Installation
Tags	Typen
Tastaturshortcuts	Requester
Titelzeile	Editorfenster
ToolTypes	ToolTypes
Update	Vollversion
Variablen	Variablen
Variablen	Variablebeschreibung
Vollversion	Vollversion

1.312 WRITE.guide/Funktionsindex

Funktionsindex
****

```
DoREXX
DoString
DoubleKey
Exists
Find
FindListEntry
FindPattern
Flash
Fold
Font
GetConfig
GetEnv
GetFile
GetFiles
GetFindReplace
GetFont
GetListEntry
GetNumber
GetREXXVar
GetString
GetVar
GoTextMark
Goto
GotoMouse
GPrefs
GuideHelp
Help
Hide
Iconify
If
Inc
InsertBlock
Item
ItemBar
Key
LastWord
ListSize
ListToBuffer
LoadBuffer
LockWindow
LowerBlock
Macro
MacroPlay
MacroRec
MacroStop
Mark
MatchBracket
Menu
Message
MessageOK
ModifyScreen
ModifyWin
New
NewEd
NextEd
NextWord
NOP
```

```
DoREXX
DoString
DoubleKey
Exists
Find
FindListEntry
FindPattern
Flash
Fold
Font
GetConfig
GetEnv
GetFile
GetFiles
GetFindReplace
GetFont
GetListEntry
GetNumber
GetREXXVar
GetString
GetVar
GoTextMark
Goto
GotoMouse
GPrefs
GuideHelp
HelpFkt
Hide
Iconify
If
Inc
InsertBlock
Item
ItemBar
Key
LastWord
ListSize
ListToBuffer
LoadBuffer
LockWindow
LowerBlock
Macro
MacroPlay
MacroRec
MacroStop
Mark
MatchBracket
Menu
Message
MessageOK
ModifyScreen
ModifyWin
New
NewEd
NextEd
NextWord
NOP
```

```
NormalPointer
Open
OpenPort
PageDown
PageUp
Pannel
ParseBuffer
Pop
Prefs
PreparseString
Push
Quit
QuitEd
RangeCheck
RangeRound
ReFold
Refresh
RemoveList
Replace
ReplaceList
Return
Save
SaveBuffer
SaveConfig
SaveGConfig
Screen
ScrollDown
ScrollUp
SetBackUp
SetEnv
SetError
SetHotKey
SetMark
SetPannel
SetREXXClip
SetREXXVar
SetTextMark
SetTitle
SetUserFkt
SetVar
ShowASCII
ShowConstants
ShowFunctions
ShowIndex
ShowList
ShowVars
Silent
Start
StrToBuffer
Sub
SubBar
SwapChar
System
Tab
UnDelLine
Undo
UnFold
```


```
NormalPointer
Open
OpenPort
PageDown
PageUp
Pannel
ParseBuffer
Pop
Prefs
PreparseString
Push
Quit
QuitEd
RangeCheck
RangeRound
ReFold
Refresh
RemoveList
Replace
ReplaceList
Return
Save
SaveBuffer
SaveConfig
SaveGConfig
Screen
ScrollDown
ScrollUp
SetBackUp
SetEnv
SetError
SetHotKey
SetMark
SetPannel
SetREXXClip
SetREXXVar
SetTextMark
SetTitle
SetUserFkt
SetVar
ShowASCII
ShowConstants
ShowFunctions
ShowIndex
ShowList
ShowVars
Silent
Start
StrToBuffer
Sub
SubBar
SwapChar
System
Tab
UnDelLine
Undo
UnFold
```

UnMark
UpperBlock
VersionCheck
WaitPointer
WaitPort
WinArranger
Window
WinManager
WriteChar
WriteText

1.313 WRITE.guide/Variableindex

Variableindex
****

CURSOR
EOL
EOP
EOS
EOT
EOW
EQUAL
FALSE
HIGHER
LEFT
LOWER
MARKA
MARKB
MOS
RIGHT
SOL
SOP
SOS
SOT
SOW
TRUE
_AColor
_AppIcon
_AppMenu
_AppWindow
_AutoFold
_AutoFree
_AutoIndent
_BColor
_CaseSense
_CColor
_Changed
_ChipMem
_CollectorMem
_ConfigPath
_CurrentChar

```
_CurrentConfig
_CurrentID
_CurrentLine
_CurrentUndoMem
_CurrentWord
_Cursor
_DColor
_DefaultConfig
_DefaultTool
_DelFoldMark
_EColor
_EditMode
_FastMem
_FColor
_File
_FileName
_FilePath
_FindString
_Fold
_FoldEnd
_FoldStart
_FRPattern
_InsertMode
_Language
_Length
_LoadTab
_MarkAx
_MarkAy
_MarkBx
_MarkBy
_Marked
_Optimize
_OverwriteCursor
_PatCase
_Path
_PathPath
_PatNoCase
_PublicMem
_ReadSpeed
_ReadTabs
_REG1
_REG2
_ReplaceString
_ReqToMouse
_REXX
_REXXPortName
_RightMargin
_RN
_RS
_SaveTab
_ScreenHeight
_Screenname
_ScreenWidth
_ScrRelHeight
_ScrRelWidth
_ShowEOL
_ShowSpace
```


```
_CurrentConfig
_CurrentID
_CurrentLine
_CurrentUndoMem
_CurrentWord
_Cursor
_DColor
_DefaultConfig
_DefaultTool
_DelFoldMark
_EColor
_EditMode
_FastMem
_FColor
_File
_FileName
_FilePath
_FindString
_Fold
_FoldEnd
_FoldStart
_FRPattern
_InsertMode
_Language
_Length
_LoadTab
_MarkAx
_MarkAy
_MarkBx
_MarkBy
_Marked
_Optimize
_OverwriteCursor
_PatCase
_Path
_PathPath
_PatNoCase
_PublicMem
_ReadSpeed
_ReadTabs
_REG1
_REG2
_ReplaceString
_ReqToMouse
_REXX
_REXXPortName
_RightMargin
_RN
_RS
_SaveTab
_ScreenHeight
_Screenname
_ScreenWidth
_ScrRelHeight
_ScrRelWidth
_ShowEOL
_ShowSpace
```

```
_SleepMode
_StdNumPad
_TabLength
_Time
_Undo
_UndoMem
_User
_Version
_VersionString
_WBStarted
_WinHeight
_WinMode
_Wins
_WinWidth
_WordDef
_WordOnly
_WordWrap
_WriteIcon
_WriteTabs
_XPos
_YPos
```


```
_SleepMode
_StdNumPad
_TabLength
_Time
_Undo
_UndoMem
_User
_Version
_VersionString
_WBStarted
_WinHeight
_WinMode
_Wins
_WinWidth
_WordDef
_WordOnly
_WordWrap
_WriteIcon
_WriteTabs
_XPos
_YPos
```