

```
*****      *****      *****      *****
*****      *****      *****      *****
**          ** **          ** **          ** **          **
**          ** **          ** **          ** **          **
*****      **          *****      **          **
*****      **          *****      **          **
**          ** **          **          ** **          **
**          ** **          **          ** **          **
**          ** *****      *****      *****
**          ** *****      *****      *****
```

```
**          ** *****
**          ** *****
**          **
**          **
**          **
**          **
**          **
**          **
**          **
*****      *****
***          *****
```

Inhaltsverzeichnis

1	Einführung	1
1.1	Was ist RCBD-UDOS	1
1.2	Was ist RCBD-VI	1
2	Arbeitsweise	2
2.1	Oberfläche	2
2.2	Zeichensatz	3
2.3	Undo	3
3	Programmaufruf	4
3.1	Optionen	4
4	Betriebsarten (Modi)	5
4.1	EDIT-Modus	5
4.2	Bildschirm-Modus oder Screen-Modus	5
4.2.1	Normal-Modus/Kommando-Modus	5
4.2.2	Einfüge-Modus	5
4.2.3	Kommandozeilen-Modus	5
5	Kommandobeschreibung Bildschirm-Modus	6
5.1	Bezeichnung für Sondertasten und Codes	6
5.2	Tastenfolgen	6
5.3	Steuerzeichen (Controlcodes)	8
5.4	Sondertasten	11
5.5	Charactercodes	13
6	Modifikationen an EDIT	21
6.1	Aufrufparameter	21
6.2	Prompt	21
6.3	Befehle im EDIT- und Kommandozeilen-Modus	21
6.3.1	Änderung der Befehlssyntax	21
6.3.2	Neue und geänderte Befehle	22
7	Randbedingungen	23
8	Bekannte Probleme	24

1 Einführung

1.1 Was ist RCBD-UDOS

RCBD-UDOS ist eine Portierung des 8-bit-Betriebssystems *UDOS* 3.1 des EAW Computers P8000¹ auf moderne Z80 und Z180 basierte modulare oder Single-Board-Computer. Das Projekt ist hier² beschrieben und downloadbar.

1.2 Was ist RCBD-VI

RCBD-VI ist ein stark abgespeckter *vi*-Clone für die 8bit-Betriebssysteme *RIO*³ und *UDOS*⁴. Er basiert auf dem Kern des originalen *RIO*-Textzeilen-Editors EDIT 2.1 [2][4] und bietet einen zusätzlichen Bildschirm-Modus mit einem Subset der *vi*-Funktionen. Trotz des stark reduzierten Funktionsumfanges gegenüber dem „echten“ *vi* ergibt sich ein hoher Qualitätsgewinn beim Editieren unter *RIO/UDOS*. Über eine einzige Quelldatei ist *RCBD-VI* an verschiedene Terminals anpassbar, es liegen z. Zt. Anpassungen für ADM31-Terminals sowie ANSI-kompatible Terminals oder Terminal emulationen (VT100, Linux, TeraTerm) vor.

Die Entwicklung des *RCBD-VI* begann etwa 1988. Da mussten größere Quellprogramme unter UDOS erstellt und bearbeitet werden, was mit dem Textzeilen-Editor EDIT auf Dauer äußerst nervig war. Da zu diesem Zeitpunkt schon Erfahrungen mit UNIX und *vi* vorlagen, entstand die Idee, einen *vi*-ähnlichen Editor für UDOS zu entwickeln.

Der Namenszusatz *RCBD* kam erst 2025 dazu, als die Entwicklung des *vi* auf der *RCBD-UDOS*-Plattform wieder aufgegriffen und weiter geführt wurde.

¹<https://www.robotrontechnik.de/index.htm?/html/computer/p8000.htm>

²https://rio.early8bitz.de/_rcbd/index.htm

³Zilog Inc.

⁴Kombinat Robotron, Kombinat EAW

2 Arbeitsweise

RCBD-VI versucht, mit den Ressourcen eines 8-bit-Betriebssystems und 64 KiB Hauptspeicher einen komfortablen Texteditor bereitzustellen, der eine *vi*-ähnliche Oberfläche und Bedienung bietet. Genau genommen wurde kein neuer Editor geschrieben, sondern ein bildschirmorientiertes Frontend vor den existierenden EDIT gesetzt. Alle Textmanipulationen (Einfügen, Ersetzen und Löschen von Zeilen), Bewegung durch die Textdatei (Zeilen vor und zurück), die Lese- und Schreiboperationen zum Dateisystem sowie die Verwaltung von Textblöcken bei Dateien, die auf Grund Ihrer Größe nicht vollständig in den freien Speicher passen, werden vom EDIT-Kern abgearbeitet.

2.1 Oberfläche

Der Bildschirm ist folgendermaßen aufgebaut:

```
Insert      Zeile:       7   Spalte:    8   Anzahl:     20 d
~
~
~
~
~
~
*h vi V4.4 2026-05-19          ; No changes since 2026-05-06
*l off
*p 70
 _rioec_ equ           0
*i rio.equ.s
*i vi.equ.s
*i vi.deb.s
*l on
;*****
;*                               *
;* Main-Modul fuer vi-editor    *
;* RCBd Edition                  *
;* Vers.: 4.4.b Stand: 19.05.2026 *
;*                               *
;*****
fl_up      equ          findfl        ; Doppelverwendung in flag1
```

Die oberste Zeile ist die Statuszeile, die verschiedene Informationen anzeigt. Die Statuszeile kann optional ausgeblendet werden, dann wird in Zeile 1 normaler Text angezeigt.

Die aktuelle Textzeile (Arbeitszeile) ist stets in der Bildmitte. Die Position des Cursors wird in der Statuszeile angezeigt. Die Arbeitszeile bleibt immer in der Bildmitte, d. h., bei vertikaler Bewegung durch die Textdatei (Scrollen) wandert der Text hinter dem Cursor auf dem Bildschirm auf und nieder.

Bildschirmzeilen, die nur eine Tilde '˜' in der ersten Spalte enthalten, sind außerhalb des gerade geladenen Textblocks. Das sind die Bereiche vor

dem Dateianfang oder hinter dem Dateiende. Bei Dateien, die auf Grund ihrer Größe nur partiell (blockweise) in den Hauptspeicher geladen werden können, sind das auch die Bereiche vor dem oder nach dem gerade geladenen Textblock.

2.2 Zeichensatz

RCBD-VI verarbeitet nur Dateien vom Typ A (ASCII). Gemäß *UDOS*-Konventionen dürfen Textdateien nur 7-Bit-ASCII-Zeichen enthalten. Ausnahme ist das 'End of Text' Zeichen (Code 0xFF) hinter der letzten Textzeile. Weitere Zeichen mit gesetztem Bit 7 befinden sich während des Editierens im Textpuffer im Hauptspeicher (RAM). Sie dienen der Steuerung des Bildschirmaufbaus, der Blockverwaltung bei Textdateien, die größer als der verfügbare Editierpuffer sind sowie der Kontrolle der Tabulatorexpansion.

Prinzipiell können auch Zeichen mit gesetztem Bit 7 (Codes 0x80..0xFF) eingegeben werden, sofern sie die Tastatur des Computers bzw. das verwendete Terminal liefern können. Solche Zeichen konflikteten aber potentiell mit der internen Pufferverwaltung von *RCBD-VI* und können zu unerwünschten Effekten führen. Bei der Ausgabe im Bildschirm-Modus wird generell Bit 7 jedes Zeichens zurück gesetzt.

2.3 Undo

Im Normal-Modus wird die aktuelle Zeile der Textdatei aus dem Textpuffer in einem separaten Zeilenpuffer geladen und dort bearbeitet. So lange die ausschliesslich Kommandos angewendet werden, die nur Änderungen oder Bewegungen innerhalb der aktuellen Zeile bewirken, kann der Originalzustand der Zeile durch das Undo-Kommando wieder hergestellt werden.

Die aktuelle Zeile wird mit allen erfolgten Modifikationen aus dem Zeilenpuffer zurück an die entsprechende Stelle des Textpuffers kopiert, wenn

- die aktuelle Zeile durch ein Scrollkommando (vertikale Bewegung im Text) verlassen wird
- in den Kommandozeilen- oder EDIT-Modus gewechselt wird

3 Programmaufruf

Da *RCBD-VI* auf dem EDIT-Kern beruht, gilt mit Ausnahme des Programmnamens die gleiche Kommandozeilensyntax wie bei EDIT. Optionen sind caseinsensitiv.

`%vi [[device/]filename] [options]`

Ohne Angabe von `'filename'` fragt *RCBD-VI* interaktiv nach dem Dateinamen.

3.1 Optionen

- E Diese Option bewirkt, dass sich *RCBD-VI* nach dem Start nicht im Bildschirm-Modus sondern im EDIT-Modus befindet.
- N (No-Old) Beim Aufruf von *vi* wird keine Backupdatei `'filename.old'` bzw. `'Filename.OLD'` erzeugt.

`O=[device/]oldfile`

Die OLD-Datei wird unter dem angegebenen Namen angelegt. Die Backupdatei hat immer die gleiche Recordlänge wie die Originaldatei.

`RL=xx`

Ist `'filename'` eine neue Datei, wird sie mit der Recordlänge `xx` angelegt. Die Angabe von `xx` erfolgt Hexadezimal. Gültige Werte sind 80, 100, 200 und 400. Ohne RL-Option wird die Standard-RL des Systems verwendet. Beim Öffnen vorhandener Dateien ist die RL-Angabe ohne Wirkung.

`X=[device/]commandfile`

Die in `'commandfile'` enthaltenen Editorkommandos (Kommandozeilen-Modus) werden in der zu editierenden Datei `'filename'` automatisch auf den Text angewendet. *RCBD-VI* verbleibt nach Abarbeitung des Kommandofiles im EDIT-Modus. Die Recordlänge der Kommandodatei darf 100H nicht übersteigen.

4 Betriebsarten (Modi)

4.1 EDIT-Modus

Der EDIT-Modus ist die Betriebsart des ursprünglichen EDIT. Die Kommandos von EDIT werden in dieser Dokumentation nicht beschrieben, hierzu sei auf die Dokumentation zu EDIT verwiesen [2][4]. Der EDIT-Modus entspricht im originalen *vi* dem ex-Modus.

4.2 Bildschirm-Modus oder Screen-Modus

Dies ist die bildschirmorientierte Betriebsart des *vi*. Im Bildschirm-Modus unterscheidet man den Normal-Modus, den Einfüge-Modus und den Kommandozeilen-Modus.

4.2.1 Normal-Modus/Kommando-Modus

Alle eingegebenen Zeichen sind Kommandos, sie führen Operationen am dargestellten Text aus, z.B. scrollen, Zeichen und Zeilen löschen oder Wechsel in einen anderen Modus.

Einigen Kommandos kann ein Zähler *[count]* vorangestellt werden, der die Anzahl der Ausführungen angibt. Default ist *[count]=1*, d.h. das Kommando wird genau einmal ausgeführt. Die Eingabe von '*' für *[count]* führt das Kommando so oft aus, wie es im jeweiligen Kontext möglich ist, aber maximal 32767 mal. Begrenzt wird die Ausführung je nach Kommando am Zeilenende oder -anfang sowie am Datei- oder Blockende bzw. -anfang.

4.2.2 Einfüge-Modus

Im Einfüge-Modus (Insert) werden alle eingegebenen Zeichen an der Cursorposition in den laufenden Text eingefügt. Eine einzelne Textzeile kann dabei maximal 256 Zeichen lang werden, inclusive des abschließenden Zeilenendezeichens. Verlassen wird der Einfüge-Modus durch die Tasten <ESC> oder <DEL>. Ein <CR> im Einfüge-Modus bewirkt das Splitten der Zeile an dieser Stelle, der Cursor steht danach am Anfang der abgetrennten bzw. neuen Zeile.

Als Spezialfall des Einfüge-Modus ist der Ersetzungs-Modus (Replace) zu sehen, in dem die neu eingegebenen Zeichen existierende Zeichen des Textes überschreiben, es sei denn, es gibt an der aktuellen Cursor-Position keine alten Zeichen mehr. Dann wird eingefügt.

4.2.3 Kommandozeilen-Modus

vi verbleibt im Normal-Modus, aber in der untersten Bildschirmzeile erscheint ein Prompt (': '), an dem jeweils ein einzelnes Textzeilen-Kommando

des EDIT eingegeben werden kann. Nach dessen Ausführung wird der Bildschirm im Screen-Modus aktualisiert.

5 Kommandobeschreibung Bildschirm-Modus

Dieses Kapitel beschreibt die Wirkung aller Zeichen im Normal-Modus des Editors. Sofern einzelne Zeichen im Einfüge-Modus Bedeutung besitzen, wird diese zusätzlich erläutert. Die Zeichen sind in ihrer Reihenfolge im ASCII Zeichensatz aufgeführt. Nicht aufgeführte Zeichen haben weder eine Bedeutung im Normal-Modus noch eine Sonderfunktion im Einfüge-Modus.

5.1 Bezeichnung für Sondertasten und Codes

Sondertasten wie <C_Home> oder <C_Pgup> sind nicht bei jedem Terminal vorhanden. Die Entwicklung von *RCBD-VI* erfolgte an einem *UDOS*-System mit serieller Konsole. Es wurde mit einer ANSI-Terminalemulation gearbeitet, wo über die PC-Tastatur alle gelisteten Tasten zur Verfügung stehen.

<NUL>	Nullcharacter
<BS>	Taste Backspace
<HT>	Taste Horizontaltabulator
<LF>	Linefeed
<FF>	Formfeed
<CR>	Carriage Return, Return oder Enter-Taste
<ESC>	Taste Escape
<DEL>	Rubout, Taste Delete oder Del
<C_Up>	Cursortaste Hoch
<C_Down>	Cursortaste Runter
<C_Right>	Cursortaste Rechts
<C_Left>	Cursortaste Links
<C_Einfg>	Cursortaste Insert oder Einf(ügen)
<C_Entf>	Cursortaste Delete oder Entf(ernen)
<C_Home>	Cursortaste Home oder Pos1
<C_End>	Cursortaste End(e)
<C_Pgup>	Cursortaste Page Up (PgUp) oder Bild hoch
<C_Pgdn>	Cursortaste Page Down (PgDn) oder Bild runter
Ctrl_X	Zeichentaste 'X' zusammen mit der <CTRL> oder <STRG> Taste

5.2 Tastenfolgen

In der Kommandobeschreibung werden Kommandos, die aus zwei Tasten bestehen, z. B. als **dh** dargestellt. Hier sind **d** und **h** nacheinander einzugeben, da **d** allein noch kein vollständiges Kommando darstellt.

Einige Kommandos sind Kurzformen für die Folge zweier eigenständiger Kommandos. In diesem Fall wird die abkürzbare Kommandofolge in der Form `$ a` notiert.

5.3 Steuerzeichen (Controlcodes)

Ctrl_A

Dieses Kommando schaltet die zusätzliche Markierung des Zeilenendes durch das Zeichen '\$' sowie die Anzeige von Sonderzeichen im Text abwechselnd ein und aus. Das entspricht im originalen *vi* den Befehlen `:set list` und `:set nolist`. Sonderzeichen im Text werden in der normalen Ansicht nur als Punkt '.' dargestellt. Bei aktivierter erweiterter Anzeige werden:

- Steuerzeichen im Text (Codes 0x00 bis 0x1F) mit dem Großbuchstaben des Controlcodes angezeigt. Beispiel: 0x04 = **Ctrl_D** wird als 'D' angezeigt.
- <DEL> Zeichen im Text (Code 0x7F) werden als '^' angezeigt.

[count] **Ctrl_B**

bewegt den Cursor eine oder [count] Bildschirmseite(n) rückwärts (back, nach oben). Eine Anzahl Seiten kann vorgegeben werden. Von der Ursprungsseite bleiben zwei Zeilen stehen, wenn es möglich ist. Das Verhalten am Datei- oder Blockanfang ist beim **k** Kommando nachzulesen.

Ctrl_C

Abbruchtaste alternativ zu <ESC>.

[count] **Ctrl_D**

Die Cursorposition wird um eine halbe Bildschirmseite nach unten verlagert (down). Wird eine Anzahl vorgegeben, so wird der Cursor um die entsprechende Anzahl Zeilen nach unten bewegt. Die eingegebene Zeilenzahl bleibt für weitere **Ctrl_D** und **Ctrl_U** Kommandos gespeichert. Das Verhalten am Datei- oder Blockende ist beim **j** Kommando nachzulesen.

[count] **Ctrl_F**

bewegt den Cursor eine oder [count] Bildschirmseite(n) vorwärts (forward, nach unten). Eine Anzahl Seiten kann vorgegeben werden. Von der Ursprungsseite bleiben zwei Zeilen stehen, wenn es möglich ist. Das Verhalten am Datei- oder Blockende ist beim **j** Kommando nachzulesen.

[count] **Ctrl_H**, [count] <BS>

(Backspace) Im Normal-Modus bewirken diese Kommandos eine Cursorbewegung nach links, analog zu **h**.

Im Einfüge-Modus dienen **Ctrl_H** und <BS> zur Eingabekorrektur. Es wird jeweils das Zeichen links vom Cursor gelöscht. Eine [count]-Angabe kann im Einfüge-Modus nicht vorangestellt werden.

Ctrl_I, <HT>

(Horizontal-Tabulator) Kein Kommandozeichen im Normal-Modus.

Im Einfüge-Modus wird ein Tabulatorzeichen (Code 0x09) im Text eingetragen und auf dem Bildschirm werden Leerzeichen bis zum nächsten Tab-Stopp eingefügt. Tab-Stopp's sitzen standardmäßig (nach Programmstart) in jeder achten Spalte und können alternativ mit dem 'TS' Kommando im Textzeilen-Modus auf 2 oder 4 umgestellt werden. Die Tabulatorweite im Bildschirm-Modus ist unabhängig vom Tabulatorsatz des \$CON-Treibers.

[count] Ctrl_J, [count] <LF>

(Linefeed) Es erfolgt eine Cursorbewegung um eine oder um [count] Zeilen nach unten, analog zu **j**.

Befindet sich *RCBD-VI* im Einfüge-Modus, verbleibt das Programm in diesem. Eine [count]-Angabe kann im Einfüge-Modus nicht vorangestellt werden.

Ctrl_L

(Formfeed <FF>) Es wird ein Neuzeichnen des Bildschirnes (Redraw) ausgeführt. Das kann notwendig werden, wenn durch einen Fehler Bildschirmanzeige und Text im Speicher nicht mehr übereinstimmen.

Hinweis: Bei ADM-31 kompatiblen Terminals ist **Ctrl_L** identisch mit dem Code, den die **C_Right** Taste liefert. In diesem Fall muss für die Redraw-Funktion das Kommando **Ctrl_R** benutzt werden.

[count] Ctrl_M, [count] <CR>

(Carriage Return, Enter) Im Normal-Modus bewegt **Ctrl_M** den Cursor an den Anfang der nächsten Zeilen, oder an den Anfang der n-ten Zeile, wenn vorher eine Anzahl angegeben wurde (siehe **+**). Das Verhalten am Datei- oder Blockende ist äquivalent zu **j**.

Im Einfüge-Modus wird die aktuelle Zeile an der Stelle, an der sich der Cursor gerade befindet, beendet und unter dieser eine neue Zeile eröffnet, in der die Eingabe fortgesetzt wird. Der Rest der vorherigen Zeile wird mit in die neue Zeile übernommen (Zeilensplittung).

[count] Ctrl_N

Dieses Kommando ist identisch mit dem **j** Kommando.

[count] Ctrl_P

Dieses Kommando ist identisch mit dem **k** Kommando.

Ctrl_Q

Kein Kommandozeichen im Normal-Modus. Im Einfüge-Modus wirkt

dieses Zeichen als Führungszeichen vor allen nicht direkt eingebbaren Zeichen analog zu `Ctrl_V`. Es ist zu beachten, dass `Ctrl_Q` unter *UDOS* von einigen Konsolentreibern abgefangen wird und der Editor es eventuell nicht zu sehen bekommt.

`Ctrl_R`

Im Normal-Modus ist `Ctrl_R` eine Alternative zu `Ctrl_L`, da `Ctrl_L` bei manchen Terminaltypen als Code für eine Cursortaste belegt ist.

Im Einfüge-Modus schaltet `Ctrl_R` den Repeat-Modus ein, das folgende Eingabezeichen wird automatisch solange wiederholt, bis der Repeat-Modus mit `<ESC>` beendet wird. Während des Repeat-Modus kann die automatische Zeicheneingabe mit Space angehalten und wieder fortgesetzt werden. Das Drücken einer anderen Taste führt zu einer Tempoänderung der Eingabe, dabei bestimmt der Lowteil des Zeichencodes das Tempo. `a` (Code `0x61`) erzeugt eine hohe Eingabefrequenz, `p` (Code `0x70`) die niedrigste.

`[count] Ctrl_U`

Im Normal-Modus wirkt `Ctrl_U` invers zum `Ctrl_D` Kommando, d. h., der Cursor bewegt sich eine halbe Bildschirmlänge nach oben (up). Wurde in einem vorherigen `Ctrl_U` oder `Ctrl_D` Kommando eine andere Zeilenzahl vorgegeben, so wird um diese Anzahl Zeilen gerollt. Das Verhalten am Datei- oder Blockanfang ist beim `k` Kommando nachzulesen.

Im Einfüge-Modus hat `Ctrl_U` die Undo-Funktion. Die aktuelle Zeile wird neu aus dem Textspeicher in den Arbeitsspeicher geholt. Damit wird der Stand vor Beginn der Änderungen hergestellt.

`Ctrl_V`

Kein Kommandozeichen im Normal-Modus. Im Einfüge-Modus dient `Ctrl_V` als Führungszeichen zur Eingabe der Zeichen, die im Einfüge-Modus eine Sonderbedeutung besitzen und demzufolge nicht direkt eingetippt werden können. `Ctrl_Q` hat die gleiche Bedeutung.

`Ctrl_W`

Kein Kommandozeichen im Normal-Modus. Im Einfüge-Modus löscht `Ctrl_W` das links vom Cursor stehende Wort analog zu `db` im Normal-Modus.

`Ctrl_Y`

schaltet die Statuszeile am oberen Bildrand abwechselnd ein und aus.

`Ctrl_[, <ESC>`

bricht alle unvollständig eingegebenen Kommandos ab und versetzt

den Editor in den Normal-Modus. Ebenso werden repetierende Kommandos sofort abgebrochen. Im Einfüge-Modus beendet `<ESC>` die Eingabe und versetzt den Editor ebenfalls in den Normal-Modus.

`Ctrl_\`

Wechsel in den EDIT-Modus. Die Steuerung wird an den zeilenorientierten Editor EDIT übergeben. Alle Zeilen-Kommandos werden durch das Prompt `ed:` angefordert. Zurück in den Bildschirm-Modus gelangt man durch die Kommandos `'I<CR>'` oder `'VI<CR>'`. `Q` wirkt genauso wie `Ctrl_\`.

5.4 Sondertasten

`<DEL>`

(Rubout) Im Normal-Modus wird genau ein Zeichen unter dem Cursor gelöscht, analog zu `1 dl`. Wurde vorher ein Wert für `[count]` eingegeben, erfolgt keine Aktion, der Text wird nicht verändert.

Im Einfüge-Modus wird das Zeichen unter dem Cursor gelöscht.

`[count] <C_Up>`

(Cursortaste aufwärts) Es erfolgt eine Cursorbewegung um eine oder um `[count]` Zeilen nach oben, analog zu `k`.

Befindet sich *RCBD-VI* im Einfüge-Modus, verbleibt das Programm in diesem. Eine `[count]`-Angabe kann im Einfüge-Modus nicht vorangestellt werden.

`[count] <C_Down>`

(Cursortaste abwärts) Es erfolgt eine Cursorbewegung um eine oder um `[count]` Zeilen nach unten, analog zu `j`.

Befindet sich *RCBD-VI* im Einfüge-Modus, verbleibt das Programm in diesem. Eine `[count]`-Angabe kann im Einfüge-Modus nicht vorangestellt werden.

`[count] <C_Right>`

(Cursortaste rechts) Der Cursor wird in der aktuellen Zeile um eine oder um `[count]` Zeichen nach rechts bewegt, analog zu `l`.

Befindet sich *RCBD-VI* im Einfüge-Modus, verbleibt das Programm in diesem. Eine `[count]`-Angabe kann im Einfüge-Modus nicht vorangestellt werden.

`[count] <C_Left>`

(Cursortaste links) Der Cursor wird in der aktuellen Zeile um eine oder um `[count]` Zeichen nach links bewegt, analog zu `h`.

Befindet sich *RCBD-VI* im Einfüge-Modus, verbleibt das Programm in diesem. Eine *[count]*-Angabe kann im Einfüge-Modus nicht vorangestellt werden.

<C_Einfg>

(Cursortaste Insert oder Einf(ügen)) Im Normal-Modus wechselt diese Taste in den Einfüge-Modus.

Im Einfüge-Modus schaltet dieses Kommando zwischen Einfüge-Modus (Insert) und Ersetzungs-Modus (Replace) hin und her.

<C_Entf>

(Cursortaste Delete oder Entf(ernen)) Im Normal-Modus wird genau ein Zeichen unter dem Cursor gelöscht, analog zu **1 dl**. Wurde vorher ein Wert für *[count]* eingegeben, erfolgt keine Aktion, der Text wird nicht verändert.

Im Einfüge-Modus wird das Zeichen unter dem Cursor gelöscht.

<C_Home>

(Cursortaste Home oder Pos1) setzt den Cursor auf den Anfang der aktuellen Zeile, analog zu **^** (Zirkumflex) oder **0**.

Befindet sich *RCBD-VI* im Einfüge-Modus, verbleibt das Programm in diesem.

<C_End>

(Cursortaste End(e)) setzt den Cursor an das Ende der aktuellen Zeile, analog zu **\$**.

Befindet sich *RCBD-VI* im Einfüge-Modus, verbleibt das Programm in diesem.

[count] **<C_Pgup>**

(Cursortaste Bild aufwärts) bewegt den Cursor eine oder *[count]* Bildschirmseite(n) nach oben, analog zu **Ctrl_B**.

Befindet sich *RCBD-VI* im Einfüge-Modus, verbleibt das Programm in diesem. Eine *[count]*-Angabe kann im Einfüge-Modus nicht vorangestellt werden.

[count] **<C_Pgdn>**

(Cursortaste Bild abwärts) bewegt den Cursor eine oder *[count]* Bildschirmseite(n) nach unten, analog zu **Ctrl_F**.

Befindet sich *RCBD-VI* im Einfüge-Modus, verbleibt das Programm in diesem. Eine *[count]*-Angabe kann im Einfüge-Modus nicht vorangestellt werden.

5.5 Charactercodes

- (Space) wirkt im Normal-Modus genauso wie **1**, der Cursor wird eine Zeichenposition nach rechts gesetzt.
- \$** setzt den Cursor auf das Ende der aktuellen Zeile.
- '<x>** (Apostroph) setzt den Cursor auf den Anfang der mit **m<x>** im Register **<x>** gespeicherten Zeilennummer im Text. Die Register **<x>** werden durch die Kleinbuchstaben von 'a' bis (derzeit) 'e' adressiert. Wurde Register **<x>** im Vorfeld nicht mit einer Cursorposition belegt, erfolgt keine Bewegung im Text. Hat sich der Text durch Löschen oder Einfügen von Zeilen inzwischen verschoben, wird der Cursor in der Zeile positioniert, die jetzt die im Register **<x>** gespeicherte Zeilennummer besitzt.
- ''** (2* Apostroph) setzt den Cursor auf den Anfang der Zeile, die zuletzt mit einem **G**, **gg**, **gj**, **gk**, **'** (Apostroph) oder **`** (Backtick) Kommando verlassen wurde. Hat sich der Text durch Löschen oder Einfügen von Zeilen inzwischen verschoben, wird der Cursor in der Zeile positioniert, die jetzt die im Last-Line-Register gespeicherte Zeilennummer besitzt.
- *** setzt das numerische Argument (Anzahl der Ausführungen) auf den maximalen Wert von 32767. In der Statuszeile wird dies mit **Anzahl=**** angezeigt.
- +** verhält sich genauso wie **Ctrl_M** (Carriage Return) im Normal-Modus.
- verhält sich invers zu **+** und **Ctrl_M**, das heißt, der Cursor wird an den Anfang der vorherigen Zeile oder der n-ten Zeile vor der aktuellen bewegt. Wird der Datei- oder ein Blockanfang erreicht, gilt das unter **Ctrl_K** Gesagte.
- 0** bewegt den Cursor an den Anfang der aktuellen Zeile, und entspricht damit dem **^** (Zirkumflex) Kommando. **0** wird bei der Eingabe von numerischen Argumenten (**[count]**) benutzt, wenn vorher eine der Ziffern **1...9** eingegeben wurde.
- 1...9** Eingabe numerischer Argumente (**[count]**) vor Kommandozeichen.
- :** In der untersten Bildschirmzeile erscheint das Promptzeichen **':'** und fordert zur Eingabe eines zeilenorientierten Kommandos des Editors EDIT auf. Während der Eingabe sind die üblichen Korrekturen in der

Eingabezeile mit <BS> und <DEL> bzw. den in den Systemvariablen CHRDEL und LINDEL vereinbarten Korrekturzeichen möglich. Die Eingabe wird durch <RETURN> (<CR>) quittiert. Nach Ausführung des Kommandos wird das Bild neu gezeichnet, mit der aktuellen Zeile in der Mitte. Die Eingabe kann durch <ESC> vorzeitig abgebrochen werden, die bisherige Eingabe wird dabei ignoriert und das Kommando nicht ausgeführt.

A Der Cursor wird an das Ende der Zeile gesetzt und der Einfüge-Modus eingeschaltet. **A** ist äquivalent zur Kommandofolge **\$ a**.

[count] **B**

Der Cursor wird in der aktuellen Zeile um ein oder mehrere Worte rückwärts bewegt und auf dem Wortanfang positioniert. Als Wort wird eine Folge von Nicht-Leerzeichen definiert.

C ist eine Kurzform für **c\$**.

D ist eine Kurzform für **d\$**.

[count] **E**

Der Cursor wird in der aktuellen Zeile um ein oder mehrere Worte vorwärts bewegt und auf dem Wortende positioniert. Ein Wort ist wie in **B** und **W** definiert.

[count] **G**

Ohne Angabe einer Anzahl wird der Cursor auf die letzte Zeile des Textes (Dateiende) gesetzt. Wird eine Anzahl eingegeben, wird diese als Zeilennummer interpretiert, in die das **G** Kommando den Cursor setzen soll. Ist die angegebene Zeilennummer größer als die Zeilenzahl der Datei, wird der Cursor auf die letzte Zeile des Textes gesetzt.

I Der Cursor wird an den Anfang der aktuellen Zeile gesetzt und der Einfüge-Modus eingeschaltet. **I** entspricht der Kommandofolge **^ i** (Zirkumflex + **i**).

J (Join) Die der aktuellen Zeile folgende Zeile wird an die aktuelle Zeile angehängt. Zwischen dem letzten Zeichen der aktuellen Zeile und dem ersten Zeichen der angehängten Zeile wird ein Leerzeichen eingetragen. Die Zeilenverkettung wird nicht ausgeführt, wenn der Cursor in der letzten Zeile der Datei oder des aktuellen Blockes steht. Es erfolgt kein Test auf Einhaltung der maximalen Zeilenlänge von 256 Zeichen, es können Zeichen der angehängten Zeile verloren gehen.

O eröffnet über der Zeile, in der sich der Cursor gerade befindet, eine neue leere Zeile. Der Cursor wird am Anfang der neuen Zeile platziert und der Einfüge-Modus eingeschaltet.

- Q** Der Editor verlässt den Bildschirm-Modus und geht in den EDIT-Modus über wie bei **Ctrl_**.****
- R** Versetzt den Editor in den Ersetzungs-Modus, dh., jedes eingegebene Zeichen überschreibt genau ein Zeichen des vorigen Textes an der aktuellen Cursorposition.
- S** ist eine Kurzform für **cc**.
- U** (Undo) Holt die aktuelle Zeile neu aus dem Textspeicher in den Arbeitsspeicher. Damit wird der Stand vor Beginn der Änderungen hergestellt. Das Undo-Kommando funktioniert nur, wenn der Arbeitsspeicher noch nicht wieder in den Textpuffer zurückgeschrieben wurde, wie es bei den Scroll-Kommandos sowie bei den **:**, **Q** und **Ctrl_**Kommandos passiert.****
- [count] W**
Der Cursor wird in der aktuellen Zeile um ein oder mehrere Worte vorwärts bewegt und auf den Wortanfang gesetzt. Ein Wort ist analog zu **B** und **E** als Folge von Nicht-Leerzeichen definiert.
- [count] X**
X ist eine Kurzform für **dh**.
- ZZ**
Der augenblickliche Stand der Datei wird abgespeichert und der Editor verlassen. Die Steuerung kehrt zum System Kommandointerpreter zurück.
- ^** (Zirkumflex) setzt den Cursor auf den Anfang der aktuellen Zeile.
- `<x>**
(Backtick) setzt den Cursor auf die mit **m<x>** im Register **<x>** gespeicherte Cursorposition (Zeilennummer und Spalte) im Text. Die Register **<x>** werden durch die Kleinbuchstaben von **'a'** bis (derzeit) **'e'** adressiert. Wurde Register **<x>** im Vorfeld nicht mit einer Cursorposition belegt, erfolgt keine Bewegung im Text. Hat sich der Text durch Löschen oder Einfügen von Zeilen inzwischen verschoben, wird der Cursor in der Zeile positioniert, die jetzt die im Register **<x>** gespeicherte Zeilennummer besitzt.
- ``**
(2* Backtick) setzt den Cursor auf die Cursorposition der Zeile, die zuletzt mit einem **G**, **gg**, **gj**, **gk**, **'** (Apostroph) oder **`** (Backtick) Kommando verlassen wurde. Hat sich der Text durch Löschen oder Einfügen von Zeilen inzwischen verschoben, wird der Cursor in

der Zeile positioniert, die jetzt die im Last-Line-Register gespeicherte Zeilennummer besitzt.

- a** Der Editor geht in den Einfüge-Modus über und fügt die eingegebenen Zeichen hinter der aktuellen Cursorposition ein. Die Eingabe kann in einer neuen Zeile durch Eingeben von <CR> (RETURN, **Ctrl_M**) fortgesetzt werden.

[count] **b**

Der Cursor wird in der aktuellen Zeile um ein oder mehrere Worte rückwärts bewegt und auf dem Wortanfang positioniert. Als Wort wird eine Folge von alphanumerischen Zeichen oder eine Folge von Sonderzeichen definiert.

- c** Einleitungszeichen für **c**-Subkommandos. Ein weiteres Zeichen zur Spezifizierung des Subkommandos wird erwartet. <ESC> oder nicht als Subkommando definierte Zeichen brechen den Vorgang ab.

c\$, **c<C_Right>**

Ersetzt den Rest der aktuellen Zeile, beginnend bei der Cursorposition. Der Rest der aktuellen Zeile vom Cursor bis zum Zeilenende wird gelöscht und der Einfüge-Modus eingeschaltet.

c0, **c^**, **c<C_Home>**

Ersetzt den Anfang der aktuellen Zeile, beginnend am Zeilenanfang bis vor die aktuelle Cursorposition. Alle Zeichen der aktuellen Zeile links vom Cursor bis zum Zeilenanfang werden gelöscht und der Einfüge-Modus eingeschaltet.

[count] **cB**, [count] **cb**

Ersetzt die Zeichen vor dem Cursor bis zum [count]'en Wortanfang. Die entsprechenden Zeichen links vom Cursor werden gelöscht und der Einfüge-Modus eingeschaltet. **B** und **b** sind wie bei wortweise rückwärts definiert.

- cc** Ersetzt die aktuelle Zeile durch die nachfolgende Eingabe. Die Zeile wird auf dem Bildschirm gelöscht und der Einfüge-Modus eingeschaltet. **S** ist äquivalent zu **cc**.

[count] **cE**, [count] **ce**

Ersetzt die Zeichen unter dem Cursor bis zum [count]'en Wortende. Die entsprechenden Zeichen unter dem Cursor werden gelöscht und der Einfüge-Modus eingeschaltet. **E** und **e** sind wie bei wortweise vorwärts definiert.

[count] **ch**, [count] **c<C_Right>**

Löscht [count] Zeichen links vom Cursor, aber maximal bis zum Zeilenanfang und geht dann in den Einfüge-Modus.

`[count] cl`, `[count] c<C_Left>`, `[count] c<BS>`

Löscht `[count]` Zeichen unter und rechts vom Cursor, aber maximal bis zum Zeilenende und geht dann in den Einfüge-Modus.

`[count] cW`, `[count] cw`

Ersetzt die Zeichen unter dem Cursor bis zum `[count]`'en Wortanfang. Die entsprechenden Zeichen unter dem Cursor werden gelöscht und der Einfüge-Modus eingeschaltet. `W` und `w` sind wie bei wortweise vorwärts definiert.

`d` Einleitungszeichen für `d`-Subkommandos. Ein weiteres Zeichen zur Spezifizierung des Subkommandos wird erwartet. `<ESC>` oder nicht als Subkommando definierte Zeichen brechen den Vorgang ab.

`d$`, `d<C_End>`

In der aktuellen Zeile wird der Text von der Cursorposition bis zum Zeilenende gelöscht. `D` ist eine Kurzform von `d$`.

`d0`, `d^`, `d<C_Home>`

Löscht die Zeichen links vom Cursor bis zum Zeilenanfang.

`[count] dB`, `[count] db`

Löscht die Zeichen vor dem Cursor bis zum `[count]`'en Wortanfang, aber maximal bis zum Zeilenanfang. `B` und `b` sind wie bei wortweise rückwärts definiert.

`[count] dd`

Es wird die Zeile, in der sich der Cursor befindet oder dort beginnend eine vorher eingegebene Anzahl von Zeilen gelöscht, aber nicht über das Ende der Datei oder das aktuelle Blockende hinaus. Nach dem Löschvorgang steht der Cursor auf der ersten nicht gelöschten Zeile oder auf der ersten Zeile des nächsten Blockes, falls ein Blockende erreicht wurde.

`[count] dE`, `[count] de`

Löscht die Zeichen unter dem Cursor bis zum `[count]`'en Wortende, aber maximal bis zum Zeilenende. `E` und `e` sind wie bei wortweise vorwärts definiert.

`[count] dh`, `[count] d<BS>`, `[count] d<C_Left>`

Löscht `[count]` Zeichen links vom Cursor, aber maximal bis zum Zeilenanfang.

`[count] dl`, `[count] d<C_Right>`

Löscht `[count]` Zeichen unter dem Cursor, aber maximal bis zum Zeilenende.

`[count] dW`, `[count] dw`

Löscht die Zeichen unter dem Cursor bis zum `[count]`'en Wortanfang, aber maximal bis zum Zeilenende. `W` und `w` sind wie bei wortweise vorwärts definiert.

`[count] e`

Der Cursor wird in der aktuellen Zeile um ein oder mehrere Worte vorwärts bewegt und auf dem Wortende positioniert. Ein Wort ist wie in `b` und `w` definiert.

`g` Einleitungszeichen für `g`-Subkommandos. Ein weiteres Zeichen zur Spezifizierung des Subkommandos wird erwartet. `<ESC>` oder nicht als Subkommando definierte Zeichen brechen den Vorgang ab.

`[count] gg`

Ohne Angabe einer Anzahl wird der Cursor auf die erste Zeile des Textes (Dateianfang) gesetzt. Wird eine Anzahl eingegeben, wird diese als Zeilennummer interpretiert, in die das `gg`-Kommando den Cursor setzen soll.

`gj`, `g<LF>`, `g<C_Down>`

Der Cursor wird auf die letzte Zeile des aktuell geladenen Blockes gesetzt, oder auf die letzte Zeile des Textes (Dateiende), wenn der gesamte Text in einem Block in den Speicher geladen werden konnte.

`gk`, `g<C_Up>`

Der Cursor wird auf die erste Zeile des aktuell geladenen Blockes gesetzt, oder auf die erste Zeile des Textes (Dateianfang), wenn der gesamte Text in einem Block in den Speicher geladen werden konnte.

`[count] h`

Der Cursor wird um ein oder um eine vorgegebene Anzahl Zeichen nach links bewegt, aber maximal bis zum Zeilenanfang.

`i` fügt Text vor dem Cursor ein. Sonst verhält sich `i` wie `a`.

`[count] j`

Der Cursor wird in der gleichen Spalte (wenn möglich) eine Zeile nach unten bewegt. Wird eine Anzahl vorgegeben, so wird der Cursor um diese Anzahl Zeilen bewegt. Wird das Dateiende erreicht, erfolgt in der untersten Bildschirmzeile die Meldung „eof“ und das Kommando wird abgebrochen. Wird bei großen Dateien ein Blockende erreicht, erscheint die Meldung „End of block proceed?“ und es wird die Eingabe eines Zeichens erwartet. Eine Quittung mit 'Y' oder 'y' bewirkt den Wechsel zum nächsten Block, alle anderen Eingaben brechen das `j`-Kommando ohne Blockwechsel ab. Nach einem Blockwechsel bleibt

der Cursor in der ersten Zeile des neuen Blockes stehen, auch wenn die geforderte Anzahl noch nicht abgearbeitet ist.

[count] **k**

Der Cursor wird in der gleichen Spalte (wenn möglich) eine Zeile nach oben bewegt. Wird eine Anzahl vorgegeben, so wird der Cursor um diese Anzahl Zeilen bewegt. Wird der Dateianfang erreicht, erfolgt in der obersten Bildschirmzeile die Meldung „top“ und das Kommando wird abgebrochen. Wird bei großen Dateien ein Blockanfang erreicht, erscheint die Meldung „Top of block proceed?“ und es wird die Eingabe eines Zeichens erwartet. Eine Quittung mit 'Y' oder 'y' bewirkt den Wechsel zum davorliegenden Block, alle anderen Eingaben brechen das **k**-Kommando ohne Blockwechsel ab. Nach einem Blockwechsel bleibt der Cursor in der letzten Zeile des neuen Blockes stehen, auch wenn die geforderte Anzahl noch nicht abgearbeitet ist.

[count] **l**

Der Cursor wird um ein oder um eine vorgegebene Anzahl Zeichen nach rechts bewegt, aber maximal bis zum Zeilenende.

m<x>

speichert die aktuelle Cursorposition (Zeilennummer und Spalte) im Register <x>. Die Register <x> werden durch die Kleinbuchstaben von 'a' bis (derzeit) 'e' adressiert. Mit dem **'<x>** Kommando kann später zu der gespeicherten Position im Text gesprungen werden. Eine wiederholte Anwendung des Kommandos überschreibt die vorherige im Register <x> gespeicherte Cursorposition.

o eröffnet unter der Zeile, in der sich der Cursor gerade befindet, eine neue Zeile. Der Cursor wird am Anfang der neuen Zeile platziert und der Einfüge-Modus eingeschaltet.

r Ersetzt das Zeichen, auf dem sich der Cursor gerade befindet, durch das nächste eingegebene Zeichen. Wird als Zeichen <CR> eingegeben, wird die aktuelle Zeile in zwei Zeilen aufgespalten, der Cursor steht dann am Anfang der zweiten Hälfte.

[count] **s**

s ist eine Kurzform für **cl**.

u hat die gleiche Funktion wie **U**.

[count] **w**

Der Cursor wird in der aktuellen Zeile um ein oder mehrere Worte vorwärts bewegt und auf den Wortanfang gesetzt. Ein Wort ist analog zu **b** und **e** als Folge von alphanumerischen Zeichen oder als Folge von Sonderzeichen definiert.

[count] x
x ist eine Kurzform für dl.

6 Modifikationen an EDIT

Der *RCBD-VI* zugrunde liegende Textzeilen-Editor EDIT wurde bei der Entwicklung von *RCBD-VI* modifiziert und erweitert.

6.1 Aufrufparameter

Die für *RCBD-VI* neu hinzugekommenen Kommandozeilenoptionen '*X=commandfile*' und '*E*' wurden bereits im Abschnitt 2 beschrieben.

6.2 Prompt

Der Prompt zur Eingabeaufforderung im EDIT-Modus wurde vom originalen Prompt '>' zu '*ed:*' geändert. Der Prompt zur Eingabe im Kommandozeilen-Modus ist ':'

6.3 Befehle im EDIT- und Kommandozeilen-Modus

6.3.1 Änderung der Befehlssyntax

Die Befehlssyntax für originale und zusätzliche Editorkommandos wurde bei *RCBD-VI* leicht modifiziert. Die Eingabe eines Textzeilenkommandos erfolgt nur noch durch die in der EDIT-Dokumentation ([2][4]) als Abkürzung beschriebenen Zeichen. Ein Schreiben von weiteren Zeichen des Kommandonamens wird als Fehler behandelt. D. h., Ist in der EDIT-Dokumentation das Kommando als '*LI**no*' beschrieben, darf in *RCBD-VI* nur noch '*LI*' notiert werden. Die Eingabe ist caseinsensitiv, '*LI*' und '*li*' sind gleichwertig. Dafür darf zwischen Kommando und dem ersten Argument das Leerzeichen weggelassen werden. Als Beispiel das '*Next*'-Kommando.

Original EDIT:

```
>n 10
>ne 10
>nex 10
>next 10
>n /suchmuster/
>ne /suchmuster/
>nex /suchmuster/
>next /suchmuster/
```

RCBD-VI:

```
:n 10
:n10
:n /suchmuster/
:n/suchmuster/
```

Bei Kommandos, die mehr als ein Argument haben, ist ein Leerzeichen zwischen den Argumenten notwendig. Außerdem ist bei den Kommandos 'Input' und 'Replace' genau ein Leerzeichen zwischen dem Kommandozeichen ('I' bzw. 'R') und dem einzufügenden bzw. zu ersetzenden Text notwendig.

6.3.2 Neue und geänderte Befehle

TS <tabsize>

Setzt die Tabulatorweite für den Bildschirm-Modus auf <tabsize>. Für <tabsize> sind die Werte 2, 4 und 8 erlaubt. Nach dem Aufruf von *RCBD-VI* ist die Tabulatorweite auf 8 eingestellt. Um die gleiche Tabulatorweite im EDIT-Modus zu benutzen, muss dies auf *UDOS*-Systemebene im \$CON-Treiber mit `set tabsize to <tabsize>` eingestellt werden. Das trifft auch zu, wenn Textdateien mit 2 oder 4 als Tabulatorweite mit dem *UDOS* PRINT Kommando korrekt ausgegeben werden sollen (Drucken).

<zeilennummer>

(Goto linenumber) Wird als Editorbefehl nur eine Dezimalzahl eingegeben, wird der Cursor auf die angegebene Zeile gesetzt oder auf das Ende der Datei, falls <zeilennummer> größer als die Zeilenanzahl der Datei ist.

I [textzeile]

(Input) Wird 'I' nur gefolgt von <CR> eingegeben, erfolgt ein Wechsel vom EDIT-Modus in den Bildschirm-Modus. Folgt eine `textzeile` als Argument, wird diese unterhalb der aktuellen Zeile in die Datei eingefügt. Dabei wird das erste Leerzeichen zwischen 'I' und 'textzeile' als Trennzeichen gewertet und erscheint nicht im Text.

VI (VIsual) Es erfolgt ein Wechsel vom EDIT-Modus in den Bildschirm-Modus.

BB (Bottom of Block) Der Cursor wird auf die letzte Zeile des aktuell geladenen Blockes gesetzt, oder auf die letzte Zeile des Textes (Dateiende), wenn der gesamte Text in einem Block in den Speicher geladen werden konnte.

TB (Top of Block) Der Cursor wird auf die erste Zeile des aktuell geladenen Blockes gesetzt, oder auf die erste Zeile des Textes (Dateianfang), wenn der gesamte Text in einem Block in den Speicher geladen werden konnte.

S /old_string/new_string/[#_of_lines [#_of_times_per_line]]

(Substitute) identisch mit 'C' (Change), wurde zwecks Annäherung an das gewohnte Substitute-Kommando des originalen *vi* eingeführt.

CQ /old_string/new_string/[#_of_lines [#_of_times_per_line]]

(Change with Query) Das Change-Kommando mit zusätzlicher Rückfrage vor jeder Substitution. Die Abfrage 'Change (y/n) ?' muss mit 'y' oder 'Y' beantwortet werden, damit die Änderung durchgeführt wird. Alle anderen Antworten überspringen diese Instanz des Suchmusters und suchen das nächste Auftreten des Musters.

SQ /old_string/new_string/[#_of_lines [#_of_times_per_line]]

(Substitute with Query) Identisch zu 'CQ'.

QON

(Query ON) Schaltet die Abfrage 'String not in Block! Proceed?:' bei Blockwechseln ein (Standard nach Start von *RCBD-VI*).

QOF Y|N

(Query OFF) Schaltet die Abfrage 'String not in Block! Proceed?:' bei Blockwechseln ab und die automatisch angenommene Antwort auf den angegebenen Parameter. Dabei werden alle Parameter außer 'Y' oder 'y' als 'N' (Nein) behandelt.

WQ (Write and Quit) Alternative zum originalen QUIT Kommando. Für Nutzer die *vi* aus dem Bildschirmmodus mit ':wq' verlassen.

X [[device/]commandfile]

(Xecute) Ohne Argument verhält sich 'X' wie im originalen EDIT und führt die Kommandofolge aus dem internen Makro aus.

Mit dem Argument 'commandfile' wird die auszuführende Kommandofolge der angegebenen Kommandodatei entnommen. Das ist identisch zum Aufruf von *RCBD-VI* mit der Option 'X=[device/]commandfile', mit dem Bonus, dass die Kommandodatei während einer Editor-Sitzung im EDIT-Modus beliebig oft aufgerufen werden kann.

7 Randbedingungen

- Die maximale Zeilenlänge inklusive des abschließenden Zeilenendezeichens (Carriage Return) beträgt 256 Zeichen. Längere Zeilen bei Dateien anderer Herkunft bleiben so lange erhalten, wie sie nicht im *RCBD-VI* modifiziert werden. Beim Rückschreiben längerer Zeilen werden diese auf maximal 256 Zeichen gekürzt.
- Die maximale angezeigte Länge einer Zeile auf dem Bildschirm beträgt 256 Zeichen. Bei Zeilen, die durch Tabulatorexpansion mehr als 256 Zeichen in der Anzeige ergeben, kann der Cursor nicht hinter das 256. angezeigte Zeichen bewegt werden.

- Die maximale Zeilenlänge für Kommandoeingaben im EDIT-Modus und im Kommandozeilen-Modus ist ebenfalls auf 256 Zeichen begrenzt. Das gleiche gilt für die Länge von Makros oder Zeilen in Kommando-dateien.
- Bei Aufruf einzelner EDIT-Kommandos aus dem Bildschirm-Modus (Kommandozeilen-Modus) werden die Kommandos 'Input', 'Join', 'Visual' und 'Execute' ignoriert. Das Kommando 'Again' kann unter Berücksichtigung des zuletzt eingegebenen EDIT-Kommandos im Kommandozeilen-Modus benutzt werden, was besonders bei wiederholenden Suchvorgängen mit 'Locate' oder 'Find' nützlich ist.

8 Bekannte Probleme

- In der Erprobung wurde beobachtet, dass bei Dateien, die auf Grund ihrer Länge in zwei Blöcken editiert werden mussten, beim Wechsel vom ersten zum zweiten Block in einigen Fällen ein Fehler auftrat. Dieser äußerte sich darin, dass aus dem zweiten Block eine Zeile oder der Beginn einer Zeile fehlte und am Ende der Datei angehängt wurde. Der Fehler trat bei mehreren Kommandos, die den Blockwechsel bewirken, auf (z.B. 'Find', 'Get', 'Change') und war bisher nicht reproduzierbar bzw. lokalisierbar. Informationen über eigene Feststellungen bitte an den Autor melden.
- Die *RCBD-VI* hat als Addon zum EDIT-Kern den Nachteil, dass die aktuelle Datei während der Arbeit geöffnet bleibt. Systemabstürze und Spannungsausfälle haben dadurch einen negativen Effekt auf die Konsistenz des Disk-Belegungs-Vektors. Bei Maschinen mit NDOS⁵ als Masterdevice ist eine Rekonstruktion der abgestürzten Datei auf Grund des im geöffneten Zustand gelöschten Zeigerfeldes nicht möglich. Bei ZDOS⁶-Maschinen kann die Datei und der Diskettenstatus gerettet werden, wenn nur ein Block existierte. Bei mehreren Blöcken existiert eine komplizierte Zeigerstruktur in der geöffneten Datei, so dass eine Rekonstruktion echten Gurus vorbehalten bleibt. Das Dienstprogramm `FILE.DEBUG`⁷ [4] hat sich als sehr nützlich bei der Rettung erwiesen.

Wer ein RAM-Disk hat, kann auf dieser editieren, denn dort sind solche Abstürze ungefährlich. Natürlich darf man dann nicht vergessen, das fertige Produkt auf eine richtige Disk zurück zu kopieren, wenn alles gut gegangen ist. (Vieleicht auch mal zwischendurch?)

⁵New Disk Operating System; Masterdevice bei EAW UDOS und *RCBD-UDOS*

⁶Zilog Disk Operating System; Masterdevice bei Zilog RIO und Robotron UDOS

⁷Ein RIO/UDOS-Dienstprogramm zur Dateisystemreparatur auf Disketten

Ein regelmäßiges Backup wichtiger Dateien auf eine (oder mehrere) andere Disketten ist grundsätzlich nie verkehrt.

Literatur

- [1] Zilog Inc.: *Z80-RIO Operating System User's Manual*
- [2] Zilog Inc.: *Z80-RIO Text Editor User's Manual*
- [3] Kombinat EAW: *UDOS Software Systemhandbuch*, Kombinat VEB ELEKTRO-APPARATE-WERKE Berlin-Teltow, Ausgabe 05/87
- [4] Kombinat EAW: *UDOS Software Dienstprogramme*, Kombinat VEB ELEKTRO-APPARATE-WERKE Berlin-Teltow, Ausgabe 05/87
- [5] robotron Anwenderdokumentation: *Systemhandbuch UDOS-1526*, VEB Robotron-Buchungsmaschinenwerk Karl-Marx-Stadt, 1984

Finaly

Dieses Dokument wurde mit L^AT_EX 2_ε formatiert.