The Last Word

Professionelle Textverarbeitung für den Atari XL/XE

- 40/80-Zeichenanzeige
- Sehr schnelles editieren
- Bis zu 10 Dateien im Speicher
- SpartDOS X-Unterstützung
- DOS 2.5 & MyDOS-Unterstützung
- Fortschrittliche Makros
- Umfassender Dateimanager
- Anpassbares Tastaturlayout
- Flexibler Druckformatierer
- Und vieles, vieles mehr...

Version 3.2

Von Jonathan Halliday

Deutsche Übersetzung von Marc „Sleeπ“ Brings

Für
Atari XL/XE
Computer
mit 64k
oder mehr

The Last Word

The Last Word ist eine der fortschrittlichsten Textverarbeitungen die jemals für den 8-Bit ATARI geschrieben wurden. Das Konzept wurde in den späten 1980gern erdacht, in den späten 1990gern geschrieben und verbrachte dann fast ein Jahrzehnt unveröffentlicht in einer Diskettenbox auf einem Kleiderschrank. Im November 2008 wurden die Disketten wiederentdeckt und erfreuen sich einer positiven Resonanz bei der Vorstellung im Internet. Allerdings war die Welt des ATARIs nach den zehn Jahren die seit der Erstellung von *The Last Word* 2.1 vergangen sind nicht mehr wiederzuerkennen. Anwendungen, Betriebssysteme und Hardware waren sehr viel weiterentwickelt als in der Vergangenheit. So begann ein Entwicklungsprozess der eine Textverarbeitung für den 8-Bit-ATARI hervorbrachte die sowohl 40 als auch 80-Zeichen-Anzeige und eine schnelle Arbeitsgeschwindigkeit ohne irgendwelche zusätzliche Hardware bot.

Seit seiner ersten Veröffentlichung unterzieht sich *The Last Word* einer ständigen Weiterentwicklung und Tests und wird von ATARI-Liebhabern auf der ganzen Welt eingesetzt.

Die Version 3.1 des Programms gewann den „Anwendung des Jahres 2009“-Preis der von den Mitgliedern des Atari Bit Byter User Club (A.B.B.U.C. e.V.) vergeben wurde. Hier einige weitere Beispiele für das Lob das *LW* erhalten hat:

- „Deine Textverarbeitung ist die beste die jemals auf A8-Maschinen zu sehen war.“
- „Phantastische Arbeit!“
- „Eine gewaltige Errungenschaft.“
- „Ein beeindruckendes Stück Software.“
- „Das ist fantastisch. Ich habe es zur „Besten Software des Jahres!“ gewählt.

The Last Word 3.2 wurde entwickelt mit Hilfe von:

- ATASM
- WUDSN IDE for Eclipse
- Atari800WinPlus
- Atari800MacX
- Altirra
- Aspeqt
- SpartaDOS X
- SIO2SD
- VBXE2

Frühere Versionen wurden mit Hilfe des *MA65 Assemblers* für *SpartaDOS* und der *XEDIT* Textverarbeitung (beide erhältlich auf www.atari8.co.uk) und *SpartaDOS X* erstellt.

The Last Word

Professionelle Textverarbeitung für den Atari XL/XE

Mit 40 und 80 Zeichenanzeige.

Version 3.2

Startbildschirm und Zeichensätze von Paul Fisher

Deutsche Übersetzung V 1.0 für Jonathan & A.B.B.U.C. e.V.

Wenn Du Fehler oder Verbesserungsvorschläge für die Übersetzung hast
zögere nicht sie an mich zu schicken. Kontakt siehe im Anhang.

The Last Word 3.2 Bedienungshandbuch

Inhalt

1.Einleitung....4
1.1.ÜBERSICHT ÜBER THE LAST WORD....4
1.2.ÜBER DIESE BEDIENUNGSANLEITUNG....4
1.3.HARDWAREVORAUSSETUNGEN....5
1.4.LW STARTEN....5
1.4.1.LW MIT SPARTADOS X LADEN....6
1.5.GRUNDLEGENDE ARBEITSGÄNGE....6
1.5.1.DER EDITOR-BILDSCHIRM....6
1.5.2.TEXT SPEICHERN UND LADEN....8
1.5.3.TEXTBÄNKE....9
1.5.4.DIE DATEIAUSWAHL....9
1.5.5.GRUNDEINSTELLUNG....10
1.5.6.DAS PROGRAMM VERLASSEN....10
2.EDITOR-BEFEHLE....11
2.1.CURSORBEWEGUNGEN....11
2.2.TEXT EINGABEMODI....11
2.3.TEXT EINFÜGEN UND LÖSCHEN....12
2.4.BEWEGEN UND KOPIEREN VON TEXT MIT TEXTBLÖCKEN....13
2.5.SUCHEN UND ERSETZEN....14
2.5.1.SUCHEN MIT PLATZHALTERN....15
3.ZUSÄTZLICHE FUNKTIONEN DES EDITORS....16
3.1.WÖRTER ZÄHLEN....16
3.2.INDIKATOR FÜR EDITIERTEN TEXT....16
3.3.TABULATOREN....16
3.3.1.TABULATOR MODUS....16
3.4.LESEZEICHEN....17
3.5.TEXT- UND DOKUMENT-MODUS....17
3.6.BENUTZEREINSTELLUNGEN....17
3.7.EDITIEREN VON MEHREREN DATEIEN....18
3.8.DER UMGANG MIT GROSSEN DATEIN....18
4.DISKETTENOPERATIONEN....20
4.1.DISKETTENOPERATIONEN VOM EDITOR AUS....20
4.2.DAS DISKETTENMENÜ....20
4.2.1.ZUSÄTZLICHE KOMMANDOS....23
4.2.2.UNTERVERZEICHNISSE....23
5.DRUCKEN MIT LW....24
5.1.TEXTVORSCHAU....24

5.2.DIE SEITENNUMMERIERNG IM AUGE BEHALTEN....24
5.3.DRUCK-KOMMANDOS....24
5.4.EINGELAGERTE KOMMANDOS....25
5.4.1.EBENE 1 KOMMANDOS....25
5.4.2.EBENE 2 KOMMANDOS....29
5.4.3.EINEN HÄNGENDEN EINZUG ERSTELLEN....30
5.5.WEITERE DRUCKFUNKTIONEN....30
5.5.1.INTERNATIONALER ZEICHENSATZ....31
5.6.DEN DRUCKFORMATIERER EINRICHTEN....31
6.LW FÜR DEINEN DRUCKER EINRICHTEN....32
6.1.DRUCKERTREIBER....32
6.2.EINEN DRUCKERTREIBER ERSTELLEN....32
6.2.1.DRUCKER „SCHALTER“....33
6.2.2.CONTROL-STRINGS....33
6.2.3.INTERNATIONAL CHARACTERS....33
6.2.4.STILE....34
7.MAKROS....36
7.1.MAKROS LADEN....36
7.2.MACROS STARTEN....37
7.2.1.SELBSTSTARTENDE MAKROS....37
7.3.MAKROS SCHREIBEN UND EDITIEREN....38
7.4.SPEZIELLE MAKRO-KOMMANDOS....38
7.4.1.DEN BILDSCHIRM VON EINEM MAKRO AUS ABSCHALTEN....41
7.4.2.SPEZIELLE ZEICHEN....42
7.4.3.KOMMANDOS VON EINEM MAKRO AUS EINGEBEN....42
7.4.4.DER MAKRO-ZEICHENSATZ....42
7.4.5.TASTATURVEREINBARUNGEN FÜR MAKROS....43
7.5.MAKROS ERSTELLEN & BEARBEITEN....43
7.6.BEISPIEL-MAKROS....44
7.7.ZUSAMMENFASSUNG....46
8.LW EINRICHTEN....47
8.1.EINRICHTUNGSMÖGLICHKEITEN IM EDITOR....47
8.2..CFG KONFIGURATIONSDATEIN....48
8.2.1.DAS STANDARDLAUFWERK....50
8.3.DIE LW.SYS-DATEI....50
8.3.1.KONFIGURATIONS-PROFILE MIT LW.SYS....51
8.3.2.KONFIGURATION UNTER EINEM UNTERSTÜTZTEN DOS....51
8.3.3.KONFIGURATION BEI EINEM ANDEREN DOS....52
8.3.4.DER SUCHE-PFAD....53
8.3.5.DER TASTATURPUFFER....53
8.4.VERWENDUNG MEHRERER TEXTPUFFER....54

8.5.BENUTZERDEFINIERTE ZEICHENSÄTZE (FONTS) 54
8.6.DIE TASTATUR ANPASSEN.....54
8.6.1.DIE TASTATURTABELLE.....54
8.6.2.KOMMANDOS DURCH MAKROS NEU ZUWEISEN.....57
8.6.3.KOMMANDOS MIT HILFE DER VECTORTABELLE NEU VERKNÜPFEN58
8.6.4.1200 XL-TASTEN.....60
9.DOS PACKETE UND LW.....61
9.1.SPEICHERANFORDERUNGEN.....61
9.2.ATARI DOS 2.5 61
9.3.ATARI DOS XE.....61
9.4.MYDOS 4.5 61
9.5.DISKETTEN-BASIERENDES SPARTADOS.....62
9.6.SPARTADOS X 62
9.6.1.DIE SPARTADOS X ARBEITSUMGEBUNGS-VARIABLEN.....63
9.6.2.SPARTADOS X SPEICHERKONFIGURATIONEN.....64
10.LW KOMMANDO ZUSAMMENFASSUNG.....65
10.1.EDITOR KOMMANDOS.....65
10.2.SPEZIELLE TASTEN.....68
10.3.MAKRO KOMMANDOS.....69
11.DRUCK-FORMATIERUNGSKOMMANDOS.....70
12.TECHNISCHE ANMERKUNGEN FÜR PROGRAMMIERER.....71
12.1.ASSEMBLER-ADD-INS.....71
12.2.SPEICHERVERWALTUNG.....71
12.3.PROGRAMM DESIGN.....72
12.4.ENTWICKLUNG UND TESTEN.....73
12.5.WARUM LW ENTSTANDEN IST 73
12.6.ENTWICKLUNG.....76
12.7.KONTAKT 78

The Last Word 3.2 Bedienungshandbuch

1 Einleitung

1.1 ÜBERSICHT ÜBER THE LAST WORD

The Last Word (LW) ist eine der leistungsfähigsten Textverarbeitungen die jemals für den 8-bit Atari geschrieben wurden. Das Programm vereint viele der fortschrittlichen Features, wie man sie von den klassischen Atari 8-Bit Textverarbeitungen her kennt in einem Paket, mit zusätzlicher Unterstützung der neusten DOSse.

LW bietet...

- Bildschirmorientiertes editieren mit vollen 80 Zeichen
  (jederzeit auf 40 Zeichen umschaltbar)
- Horizontal scrollendes Editorfenster mit bis zu 240 Zeichen pro Spalte
- Paralleles editieren von bis zu 10 Dateien
  (auf einem speichererweiterten Computer)
- Ausgeklügelte Tastaturmakro-Sprache
- Druckvorschau mit 80-Zeichen
- Vielseitige „Ausschneiden und Einsetzen"-Funktionen (cut and paste)
- Suchen & Ersetzen in beide Richtungen
- Sehr schnelles Editieren, auch am Anfang einer großen Datei
- Ein umfassendes Online-Hilfesystem (englisch)
- Ein umfangreiches Datei-Managmentmenü welches vielen Datei-Managment-Programmen ebenbürtig ist, mit bis zu 80 Dateinamen in einem scrollenden Fenster und Unterstützung von *SpartaDOS X*-Directories mit Datum und Uhrzeit, Löschen von Batch-Dateien, Umbenennen, Kopieren und vielem mehr
- Einen vom Benutzer definierbaren Tabulator
- Individuell anzupassender Druckertreiber (erstellt als normalen Text)
- Eine Konfigurations-Datei (erstellt als normalen Text)
- Ein vollständig neudefinierbares Tastaturlayout
- Microsoft Windows®-kompatible Tastaturkommandos
- Alle internationalen Zeichen sind auf dem Bildschirm sichtbar
- Unterstützt *SpartaDOS X*, *MyDOS*, *DOS 2.5* und viele *DOS 2*-Derivate
- Läuft mit nur 64k
- Umfassende Befehle zur Formatierung des Ausdrucks
- Automatische Erstellung von Überschriftenebenen
- eingerückten Text, Silbentrennung, und vieles, vieles mehr...

1.2 ÜBER DIESE BEDIENUNGSANLEITUNG

Diese Anleitung setzt grundlegende Kenntnisse des Atari Bildschirmeditors und der Tastatur voraus. Befehlseingaben über die Tastatur werden in spitzen Klammern („<" und „>") eingeschlossen welche *nicht* mit eingegeben werden. Wenn zwei oder mehr Tasten zusammen gedrückt werden müssen sind diese mit einem „+"-Zeichen verbunden.

1.3 HARDWAREVORAUSSETUNGEN

LW läuft auf den meisten Standard-XL/XE-Ataris. Die Mindestanforderungen sind:

- 64k Speicher
- Diskettenlaufwerk
- DOS 2.5 (befindet sich bereits auf der Diskette)

Für bessere Leistungen ist folgende Ausstattung zu empfehlen:

- 128k - 1088k of RAM
- SIO2SD, SIO2PC, MyIDE, IDEa oder ein anderes schnelles Speichermedium
- SpartaDOS X (Version 4.42 oder höher empfohlen, benötigt 128k)
- RAM Disk (optional)
- guten LCD oder Röhrenmonitor
- Drucker

Wenn *LW* auf einem 64k-Atari läuft kann lediglich eine Datei editiert werden und der cut-and-paste-Puffer ist sehr klein. Auf Rechnern mit erweitertem Speicher nutzt *LW* bis zu 256k für eigene Zwecke und erlaubt das Laden von Erweiterungen, umfangreicher Makros und bietet einen großen Paste-Puffer.

1.4 LW STARTEN

Schalte den Rechner mit gedrückter <OPTION>-Taste und der *LW*-Diskette in Laufwerk 1 ein. Wenn Du den Rechner ohne <OPTION> bootest und das „READY“ des BASICs erscheint, gebe DOS ein & drücke <RETURN>. Das DOS-Menü erscheint. *LW* wird im *DOS 2.5*-Menü mit „Binary Load“ (Menüpunkt „L“) gestartet. Drücke „L“ und gib „*LW.EXE*“ ein, gefolgt von <RETURN>.

Hinweis: Anders als frühere Versionen ist *LW 3.1* WEDER mit *DOS XE* noch mit einem anderen DOS, welches RAM unter dem OS nutzt, kompatibel!

LW sucht beim Laden auf dem Standart-Laufwerk („D:“) nach folgenden Dateien und lädt sie, wenn es sie findet. Wird eine Datei nicht gefunden nutzt *LW* die Grundeinstellungen.

LW.SYS	System-Konfigurationsdatei: Sie richtet den Speicher, den Tastaturpuffer, die Tastaturbelegung und den Pfad für die Hilfe- und Systemdateien ein.
LW.CFG	Konfigurationsdatei: Sie enthält Editoreinstellungen, Benutzerdefinierte Voreinstellungen und Laufwerksgrundeinstellungen.
LW.FNT	Graphics 0 - Zeichensatz welcher im Editor und im ganzen Programm genutzt wird.
LW.F80	Spezieller 80-Zeichen-Zeichensatz für den 80-Zeichen-Modus.
LW.PDR	Druckertreiber als normalen Text; er kann mit *LW* editiert werden.
LW.MAC	Makrodatei. Sie enthält vom Benutzer erstellte, automatisierte Befehlssequenzen. Wenn ein Makro für die „@“-Taste erstellt wurde wird es unverzüglich gestartet. Siehe Abschnitt 7.
LW.EXT	Erweiterungen in Maschinensprache (nur bei speichererweiterten Geräten verfügbar). Enthält zusätzliche Funktionalitäten wie z.B. Zeichentabelle, Taschenrechner, u.s.w.

1.4.1 LW MIT SPARTADOS X LADEN

Unter *SpartaDOS X* wird *LW* durch die Eingabe von **X LW.EXE** gestartet.

Hinweis: *LW 3.X* ist NICHT kompatibel mit früheren *SpartaDOS*-Versionen vor *SpartaDOS X*. Des weiteren MUSS *SpartaDOS X* für die Benutzung von „BANKED memory" konfiguriert sein um mit *LW* zusammen arbeiten zu können. Das bedeutet das *SpartaDOS X* und *LW* nicht auf Rechnern mit weniger als 128K laufen.

Mit *SpartaDOS X* kann in der Kommandozeile ein Dateiname hinter dem Programmnamen angegeben werden:

X LW LETTER.TXT

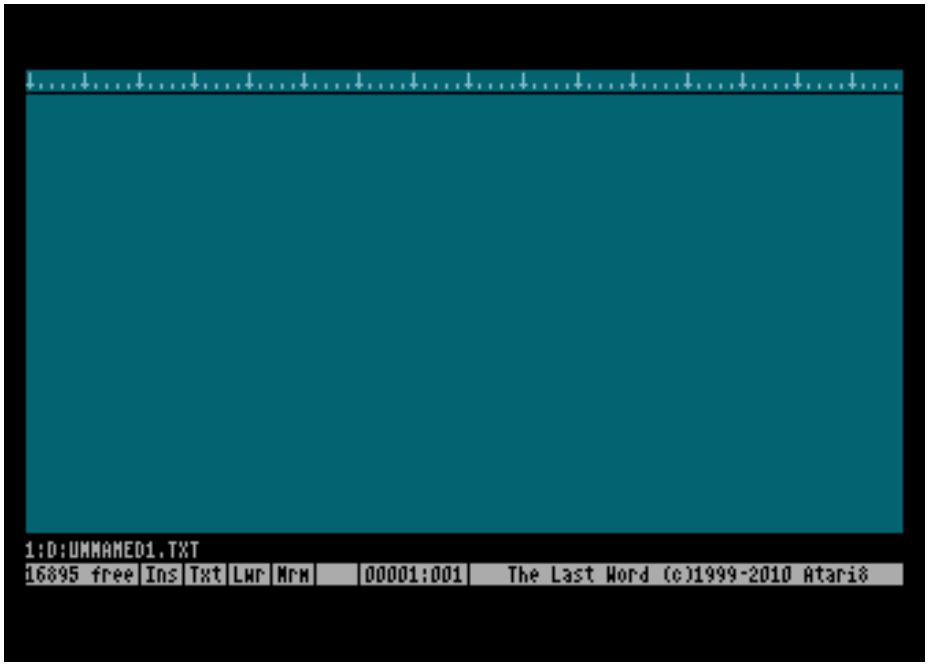
LW wird gestartet und versucht „LETTER.TXT" zu laden. Ist die Datei *nicht* vorhanden wird *LW* eine leere Datei mit diesem Namen anlegen.

1.5 GRUNDLEGENDE ARBEITSGÄNGE

Du kannst *LW* nutzen ohne dieses Handbuch zu lesen. Wenn Du ein Problem hast drücke **<HELP>**, gefolgt von eine Ziffer zwischen 1 und 9 oder 0. Wenn Du dieses Handbuch nicht ließt wird Dir allerdings eine große Menge an wichtigen Informationen fehlen.

1.5.1 DER EDITOR-BILDSCHIRM

Um die Arbeit mit LW zu beginnen, starte das Programm wie oben beschrieben und nimm Dir einen Moment Zeit um Dich mit dem Editorbildschirm vertraut zu machen. Als erstes wirst Du bemerken dass der Bildschirm Platz für 80 Zeichen bietet.



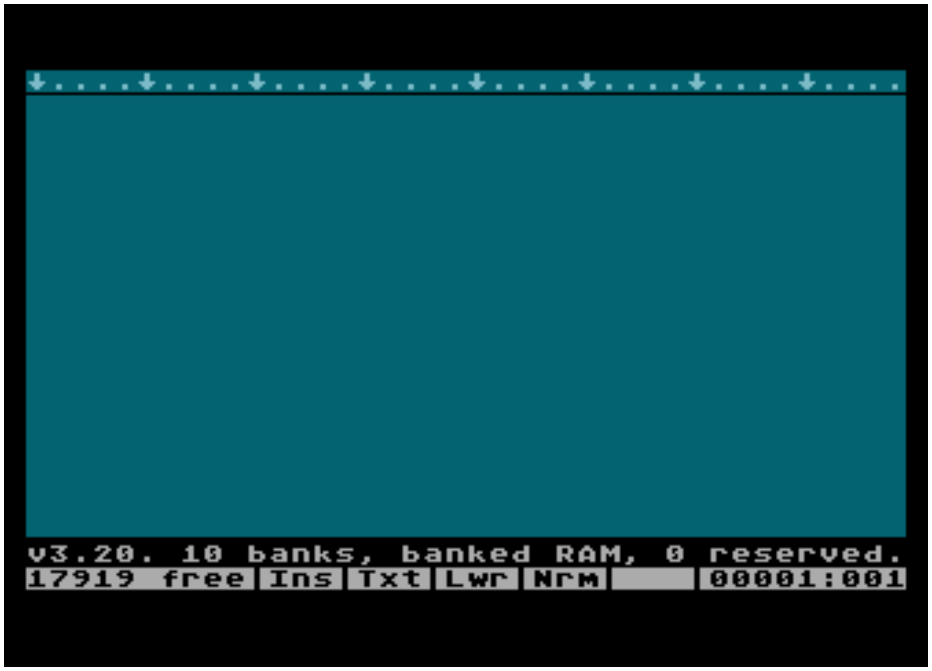
*LW´*s Grundeinstellung ist die 80-Zeichen-Anzeige, die 40-Zeichen-Anzeige kann jedoch jederzeit aktiviert werden (und auch zur Grundeinstellung gemacht werden). Um zur 40-Zeichen-Anzeige umzuschalten drücke

<SHIFT+CTRL+W>

In der unteren linken Ecke des Bildschirmes siehst Du:

80 Columns [Y/N]?

Drücke „N“ und wenn Du aufgefordert wirst die „Weite“ (Width; Anzahl der Zeichen pro Zeile auf dem Blatt) einzugeben drücke erst einmal nur **<RETURN>**. Die 40-Zeichen-Anzeige erscheint:



Du kommst zur 80 Zeichen-Anzeige zurück indem Du die selbe Prozedur ausführst: drücke **<SHIFT+CTRL+W>**, dann aber „Y“ für 80-Zeichen, wenn Du aufgefordert wirst „Width“ einzugeben drücke **<RETURN>**.

In dieser Anleitung zeigen viele Abbildungen *LW* im 40-Zeichen-Modus. Dies dient lediglich der besseren Erkennbarkeit; die Funktionen des Programms im 40-Zeichen-Modus und im 80-Zeichen-Modus sind im wesentlichen identisch.

Sowohl im 40- als auch im 80 Zeichen-Modus siehst Du oben die Tabulatorleiste (welche horizontal scrollt wenn Du ein Blatt definiert hast welches *breiter* ist als der Bildschirm), darunter ein 20 Zeilen großes Fenster zum Editieren, und, am unteren Rand des Bildschirmes, zwei Zeilen mit Statusinformationen. Der blinkende Cursor zeigt die aktuelle Schreibposition an.

Bis Du eine Taste drückst werden in der ersten Zeile der Statusinformationen die Versionsnummer des Programms, die Anzahl der Textbänke, ob erweiterter Speicher (banked) vorhanden ist und die Anzahl der für Erweiterungen reservierten Bänke angezeigt. Anschließend wechselt die Anzeige zur Grundeinstellung in der der Name der geladenen Datei (mit vorangestellter Nummer der Bank wenn mehrere als ein Textpuffer eingerichtet ist) angezeigt wird. Bis Du eine Datei lädst oder ihr einen Namen gibst wird sie „UNNAMED.TXT“ (gefolgt von der Nummer der Bank wenn es mehr als eine Textbank gibt) heißen.

Einen Text in *LW* eingeben ist leicht: schreibe wie Du es gewohnt bist, drücke <RETURN> nur am Ende eines *Abschnittes* und überlasse den Wortumbruch am Ende einer Zeile dem Programm. Die Cursortasten, <DELETE/BK SP> und <INSERT> verhalten sich exakt so wie Du es gewohnt bist.

Du wirst feststellen dass der Editor im 80-Zeichen-Modus *etwas* langsamer arbeitet als im 40-Zeichen-Modus. Das hat seine Ursache darin daß das 80-Zeichen-Display vollständig als Bitmap erstellt wird. Aber *LW* wird trotzdem mit Deinen Eingaben mitkommen und keinen Tastendruck verschlucken. Das liegt daran dass *LW* einen eigenen Tastaturpuffer hat. Beachte: Wenn Du *SpartaDOS X* (*SDX*) nutzt und den *SDX*-Tastaturpuffer (KEY.COM) aktiviert hast während *LW* läuft, dass der *SDX*-Tastaturpuffer den *LW*-eigenen Puffer ersetzt. Um die Benutzung des *LW*-Puffers zu ermöglichen (welcher die automatische Tastenwiederholung ignoriert und so verhindern dass der Cursor einfach „wegläuft“) stelle sicher dass der *SDX*-Tastaturbuffer mit „**KEY OFF**“ deaktiviert wird bevor Du *LW* startest und füge in der *LW.SYS* „BUFFER ON“ ein.

BEACHTE: Die neuste Version von *SpartaDOS X* hat eine überarbeitete *KEY.COM* welche gedrückt gehaltene Tasten ignoriert. Diese Version verträgt sich besser mit *LW* als die vorhergehenden und ist sehr zu empfehlen.

1.5.2 TEXT SPEICHERN UND LADEN

Um einen neuen Text zum ersten Mal auf Diskette zu speichern drücke

<CTRL+S> Text **s**peichern.

Eine Eingabeaufforderung mit dem Default-Dateinamen erscheint. Drücke **<RETURN>** um diesen Namen zu akzeptieren *oder* geben einen anderen Namen ein: der alte Name wird automatisch gelöscht. Nachdem Du **<RETURN>** gedrückt hast wird Dein Text auf Diskette gespeichert. Tritt ein Fehler auf wirst Du darüber informiert. Um das Speichern abzubrechen drücke **<ESC>**.

Wenn Du keinen Extender angibst wird *LW* den in der Konfigurationsdatei angegebenen Extender anhängen. Der in der mitgelieferten Konfigurationsdatei „*LW.CFG“* festgelegte Extender ist „.TXT“. Du kannst ihn allerdings mit Hilfe der **FILEEXT**-Anweisung in der Konfigurationsdatei ändern oder deaktivieren.

Der Namen den Du beim ersten Speichern vergibst wird für alle weiteren Speicheroperationen benutzt. Sobald der Text einmal gespeichert wurde wird ein weiteres **<CTRL+S>** den Text *ohne* Nachfrage mit dem selben Namen speichern. Um den Text unter einem anderen Namen oder in einem anderen Ordner zu speichern nutze...

<SHIFT+CTRL+S> Speichern als

was Dich in die **Save As>** Eingabeaufforderung bringt.

Wenn Du einen bereits vorhandenen Dateinamen angibst wird *LW* Dich warnen und fragen:

[Dateiname] **exists : Overwrite [Y/N]?**
[Dateiname] **existiert : Überschreiben [Ja/Nein]?**

Wenn Du „Y“ eingibst wird die bestehende Datei überschrieben. Gibst Du „N“ ein gelangst Du ohne weitere Aktionen in den Editor zurück.

Um einen früher geschriebenen Text zu laden gib

<CTRL+L> Lade Text

ein. Die Eingabeaufforderung mit dem Default-Laufwerk erscheint. Gib den Namen der Datei die Du laden möchtest (die Laufwerksangabe braucht Du nicht anzugeben) und beende die Eingabe mit **<RETURN>**.

BEACHTE: Die Laufwerksangabe die bei der „Lade“-Eingabeaufforderung erscheint (wie bei vielen anderen Ladeaufforderungen im Programm) geben das Laufwerk wieder welches vom *LW*-Diskmenü verwendet wird und NICHT das aktuell vom DOS verwendete Laufwerk. Die Laufwerksangabe verschwindet wenn Du beginnst zu schreiben, wird aber dem Dateinamen vorangestellt bis Du ein anderes Laufwerk angibst.

1.5.3 TEXTBÄNKE

Wenn Du einen Rechner mit mindestens 128K Ram hast auf dem *SpartaDOS X*, *DOS 2.5* oder *MyDOS* läuft, wird *LW* versuchen den extra-Speicher zu nutzen um bis zu zehn Dateien auf einmal editieren zu können. Du kannst zwischen den Bänken mit...

<SHIFT+CTRL+n> umschalten,

wobei „n“ eine der Zifferntasten ist und „0“ die zehnte Bank bezeichnet. Das Programm ist clever genug eine eingerichtete RAM-Disk *nicht* zu überschreiben. Wenn eine RAM-Disk aktiv ist kann es vorkommen dass, obwohl Du einen erweiterten Speicher hast, *LW* Dir trotzdem nur *eine* Textbank zur Verfügung stellt.

Du kannst Dir anzeigen lassen wie viele Textbänke aktiv sind mit:

<SHIFT+CTRL+?>

Es wird angezeigt:

n banks (*n)*, *[low/banked] RAM, n* reserved.

Dieser Befehl zeigt an wie viele Textpuffer zur Verfügung stehen, wie viele freie Bänke im Computer vorhanden sind, ob das Programm seinen internen Puffer im Hauptspeicher („Low RAM“) oder im erweiterten Speicher („banked RAM“) hat, und wenn, wie viele der ausgewählten Bänke für Erweiterungen reserviert sind.

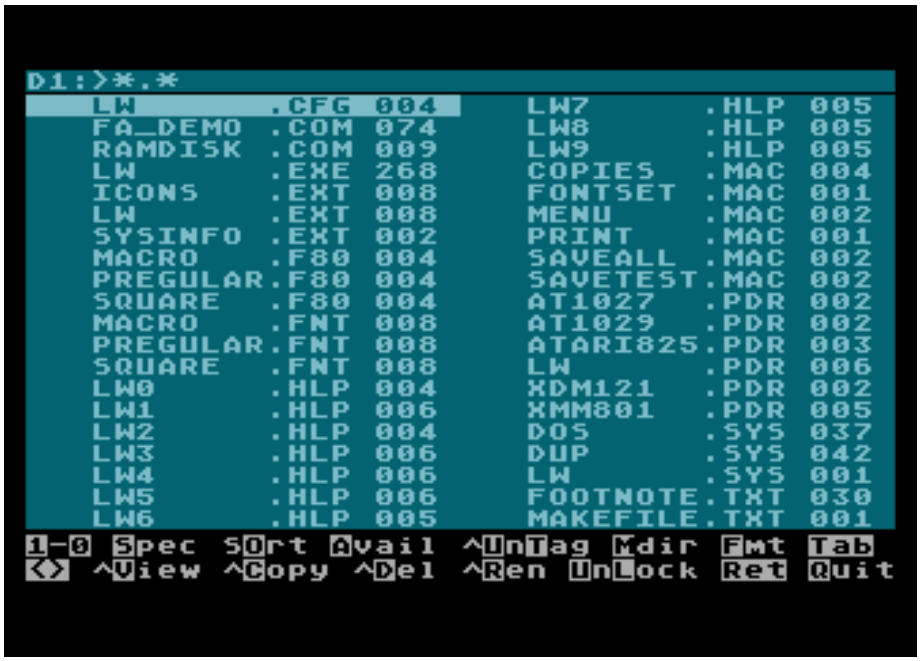
Weitere Informationen findest Du im Kapitel 8.3.2. „Konfiguration unter [...] DOS”

1.5.4 DIE DATEIAUSWAHL

Was wenn Du nicht weißt welche Dateien sich auf der Disk befinden? *LW* hat eine Dateiauswahl welche von jeder Dateinamen-eingabe aus erreichbar ist. Drücke einfach

<TAB>

wenn *LW* auf einen Dateinamen wartet, die Dateiauswahl erscheint und listet alle Dateien in dem aktuellen Ordner auf.



Markiere einfach die gewünschte Datei mit den Cursortasten und drücke **<RETURN>** um sie zu laden. Siehe Kapitel 4.2 „Das Diskettenmenü“ um mehr über die Dateiauswahl zu erfahren.

1.5.5 GRUNDEINSTELLUNG

Du kannst *LW* an Deine Vorlieben anpassen: Ein Druck auf...

<SHIFT+CTRL+O> Konfiguration speichern

erlaubt Dir die aktuelle Konfiguration auf Disk zu speichern. Die Datei wird automatisch den voreingestellten Extender „.CFG“ erhalten. Wenn Du die neuen Voreinstellungen automatisch beim nächsten Start von *LW* nutzen möchtest nenne die Datei „*LW.CFG*“.

Wie schon beschrieben führt *LW* einen automatischen Wortumbruch aus wenn ein Wort nicht mehr in die aktuelle Zeile passt. Du kannst dieses Feature abschalten:

<CTRL+W> Zeilenumbruch (**W**ord wrap) an/aus

In der Grundeinstellung von *LW* erscheinen die <RETURN>-Zeichen als gebogene Pfeile. Du kannst sie unsichtbar machen:

<SHIFT+CTRL+CLR> Return-Zeichen sichtbar/unsichtbar

Das sind nur einige der Einstellungen die in der Konfiguration gespeichert werden. Weitere Informationen über die Konfiguration von LW findest Du in Abschnitt 8 „LW einrichten“.

1.5.6 DAS PROGRAMM VERLASSEN

Um *LW* zu beenden und zum DOS zurückzukehren gebe

<SHIFT+CTRL+Q> Verlassen (**Q**uit) und gehe zum DOS

ein und bestätige mit „Y“. Sollte der Text im Speicher noch nicht gesichert sein bekommst Du jetzt Gelegenheit ihn zu Speichern *bevor* das Programm beendet wird.

2 EDITOR-BEFEHLE

Alle *LW*-Kommandos sind über Tastenkombinationen erreichbar. Hast Du Dich einmal mit den Tastaturkommandos vertraut gemacht steht Dir eine umfangreiche Menge an Befehlen zur Verfügung.

2.1 CURSORBEWEGUNGEN

Die folgenden Befehle erlauben ein schnelles bewegen des Cursors im Text:

<CTRL + ← >	Cursor nach links
<CTRL + → >	Cursor nach rechts
<CTRL + ↑ >	Cursor aufwärts
<CTRL + ↓ >	Cursor abwärts
<TAB>	Nächste Tabulatorposition (im Überschreibe-Modus)
<CTRL + A>	Sprung zum Zeilenanfang
<CTRL + Z>	Sprung zum Zeilenende
<SHIFT + ← >	Sprung zum Anfang des vorhergehenden Wortes
<SHIFT + → >	Sprung zum Anfang des folgenden Wortes
<SHIFT + ↑ >	Sprung zum Anfang des vorhergehendes Abschnitts
<SHIFT + ↓ >	Sprung zum Anfang des folgenden Abschnitts
<SHIFT + CTRL + ↑ >	Blättere einen Bildschirm auf
<SHIFT + CTRL + ↓ >	Blättere einen Bildschirm ab
<CTRL + H>	Anfang des Bildschirm, dann Anfang der Datei
<CTRL + E>	Sprung zum Ende der Datei

2.2 TEXT EINGABEMODI

Diese Befehle beeinflussen verschiedene Einstellungen im Editor:

<SHIFT+CTRL+INS> Wechselt zwischen dem Einfüge (**Ins**ert) und Überschreibe (**Ove**rtype)-Modus. Die aktuelle Einstellung findest Du in der Informationszeile. Im Einfügemodus (**Ins**) wird bei einer Eingabe der Text ab dem Cursor einschliesslich des Zeichens unter dem Cursor nach hinten geschoben. **<DELETE BKS>** löscht das Zeichen *links* vom Cursor und der restliche Text rutscht nach. **<CTRL+DELETE BKS>** löscht das Zeichen *unter* dem Cursor und der Text rutscht nach. Im Überschreibemodus (**Ovr**) überschreibt der neue Text den Alten. Beachte dass die Funktion der **<TAB>**-Taste von dem Modus in welchem sich der Editor befindet abhängt: Im Überschreibemodus springt der Cursor einfach zum nächsten Tab-Stop, im Einfügemodus werden Leerzeichen bis zum nächsten Tab-Stop einfügt.

<CAPS> Wechselt zwischen Groß- und Kleinbuchstaben.

<CTRL+CAPS> Wechselt in den Steuerzeichen-Modus. Erlaubt die Eingabe von Steuerzeichen *ohne* zuerst **<CTRL+ESC>** oder **<SHIFT+ESC>** einzugeben. Wenn Du die Konfiguration, wie später beschrieben, speicherst wird die aktuelle Einstellung gesichert und steht beim nächsten Programmstart zur Verfügung.

<SHIFT+CTRL+CAPS> International-Modus. Ermöglicht die Eingabe aller normalen Kontrollzeichen (CTRL+A bis CTRL-Z) ohne erst **<SHIFT-ESC>** drücken zu müssen. Um CTRL-Funktionen wie Speichern <CTRL-

S> oder Laden <CTRL-L> nutzen zu können muß der International-Modus erst wieder mit **<SHIFT+CTRL+CAPS>** verlassen werden. Dieser Modus vereinfacht es europäischen Nutzern lange Eingaben mit dem internationalen Zeichensatz vorzunehmen.

<SHIFT+CAPS> Wechselt in den Großbuchstabenmodus.

<INVERSE> Wechselt zwischen invertiertem Zeichen an und aus.

<ESCape> Bricht das Markieren eines Blocks ab oder ermöglicht die Eingabe eines Steuerzeichens. Beendet auch Stringeingaben und Operationen.

<CTRL+ESCape> Erlaubt die Eingabe von Steuerzeichen im Text oder Eingabedialogen. Beendet Stringeingaben oder Operationen NICHT.

<SHIFT+ESCape> Alternative zu **<CTRL+ESCAPE>**.

<CTRL+W> Schaltet den Wortumbruch ein und aus. Wird in der Konfigurationsdatei gesichert.

<SHIFT+CTRL+W> Stellt die Breite des Editorfensters ein. Mit diesem Befehl kannst Du den Displaymodus wählen (40 oder 80 Zeichen) und die gewünschte Anzahl der Zeichen pro Zeile (5-240) angeben. Wenn die Zeilenlänge die physikalische Breite des Bildschirmes überschreitet stellt der Bildschirm ein Textfenster dar durch das der Text sowohl horizontal als auch vertikal scrollt. Setzt Du die Zeilenlänge des Editors auf die Zeilenlänge Deines Druckers kannst Du Tabellen exakt so erstellen wie sie auch ausgedruckt werden. Du kannst die Eingabe der Zeichenzahl mit **<RETURN>** überspringen; dann wird die physikalische Breite des Bildschirms übernommen. Der Bildschirmmodus wird in der Konfigurationsdatei gesichert.

2.3 TEXT EINFÜGEN UND LÖSCHEN

Die folgenden Befehle erlauben einfaches Einfügen und Löschen von Text:

<DELETE BKS> Löscht das Zeichen links vom Cursor.

<CTRL+INSERT> Fügt ein Leerzeichen unter dem Cursor ein.

<TAB> Fügt im Einfügemodus (Ins) Leerzeichen bis zum nächsten Tabulatorstop ein. Im Überschreibemodus (Ovr) springt der Cursor zum nächsten Tabulator ohne Leerzeichen einzufügen.

<CTRL+DELETE BKS> Löscht Zeichen unter dem Cursor.

<SHIFT+DELETE BKS> Bietet folgende Optionen:

Delete Word, Line, Sentence, Paragraph?
(Lösche Wort, Zeile, Satz, Absatz?)

Wähle einen der hervorgehobenen Buchstaben oder verlasse die Funktion mit **<ESC>**. **<RETURN>** löscht eine Zeile (default).

Der gelöschte Text wird in den Puffer kopiert. Füge den Text mit **<CTRL+V>** oder **<SHIFT+INSERT>** wieder ein.

<SHIFT+INSERT> Fügt den eben gelöschten Text ein.

<CTRL+V> Fügt den Textpuffer oder den eben gelöschten Text ein

<CTRL+CLEAR> Löscht auf Nachfrage den gesamten Text

<SHIFT+CLEAR> Selbe Funktion wie **<CTRL+CLEAR>**

2.4 BEWEGEN UND KOPIEREN VON TEXT MIT TEXTBLÖCKEN

Mit den folgenden Befehlen können Textblöcke markiert, dann bewegt, kopiert oder gelöscht werden:

<CTRL+M> **Markiert** einen Block. Um einen Block kopieren, bewegen oder löschen zu können muss er zuerst markiert werden. Benutze diesen Befehl um den Startpunkt zu markieren. Ein aktive Markierung wird durch „Mrk“ in der Info-Zeile angezeigt. Wenn Du anschließend den Cursor bewegst wird der Text dazwischen invertiert. Das funktioniert in alle Richtungen. Etliche Blockoperationen arbeiten nur wenn vorher ein Block auf diese Weise markiert wurde. Um die Markierung aufzuheben drücke nocheinmal **<CTRL+M>** oder **<CTRL +ESCape>** .

<CTRL+X> **Schneidet** einen markierten Block aus. Der markierte Text wird in die Zwischenablage kopiert, vorausgesetzt der Text ist nicht zu lang. Beachte dass ein Text, der sich eventuell in der Zwischenablage befindet, überschrieben wird! Anschließend wird der Text **gelöscht** und der Blockmodus verlassen. Du kannst den Text mit **<CTRL+V>** oder **<SHIFT+INSERT>** wieder einfügen.

<CTRL+C> **Kopiert** einen Block wie **<CTRL+X>**, löscht den Text aber **nicht**; er bleibt markiert.

<DELETE> **Löscht** den Block nach Bestätigung unwiederbringlich **ohne** ihn in die Zwischenablage zu kopieren. Der Block kann beliebig lang sein, unabhängig von der Größe der Zwischenablage.

<SHIFT+CTRL+I> **Speichert** den Block in eine Datei. Gebe einen Dateinamen an und der Block wird - unabhängig von seiner Länge - auf Disk gespeichert. Gibst Du keinen Extender an wird „.BLK“ angehängt. Diese Möglichkeit, zusammen mit dem Zusammenfügen-Befehl **<CTRL+I>**, erlaubt es große Textblöcke zwischen verschiedenen Dateien zu bewegen.

<CTRL+I> Eine **Datei einfügen** oder verbinden (**I**nsert). Erlaubt es eine Datei in einen Text im Speicher einzufügen. Wenn Du dem Dateinamen keinen Extender gibst wird der normale Text-Extender angehängt. Wenn die angegebene Datei den freien Platz übersteigt bleibt der Text unverändert.

<CTRL+N> Zeigt die **Position** des **Cursors** im Text (die Zählung beginnt mit Null), Größe der Datei, die und die Anzahl der Wörter an. Zuletzt wird die Textgröße in Bytes angegeben. Wenn ein Block markiert ist beziehen sich die letzen zwei Werte auf den Block.

<CTRL+Y> Ändert den Block in Kleinbuchstaben

<SHIFT+CTRL+Y> Ändert den Block in Grossbuchstaben

<CTRL+]> Invertiert den Text im Block

<CTRL+[> Hebt die Invertierung im Block auf

2.5 SUCHEN UND ERSETZEN

LW bietet umfangreiche Suchfunktionen welche sowohl vorwärts als auch rückwärts im Text arbeiten. Die Groß- und Kleinschreibung *kann* beachten werden. Die Suche- und Ersetze-funktionen können Wort für Wort oder im gesamten Text, mit oder ohne Bestätigung ausgeführt werden.

<SHIFT+CTRL+F> Definiert den zu suchenden String. Gebe den Text (max. 30 Zeichen) ein nach dem gesucht werden soll.

<CTRL+F> Suche (**F**ind) String. Bewegt den Cursor zum nächsten Auftreten des Strings.

<CTRL+U> Aufwärtssuche (**U**pwards). Sucht rückwärts nach dem vorher definierten String.

<SHIFT+CTRL+R> Legt den Ersetze- (**R**eplace) String fest.

<CTRL+R> Ersatze (**R**eplace) String. Wenn ein String mit **<CTRL+F>** oder **<CTRL +U>** gefunden wurde ersetzt dieses Kommando ihn durch den mit **<CTRL+R>** festgelegten Ersetze-String.

<CTRL+G> **G**lobales suchen und ersetzen. Du gibst den Suche & den Ersetze-String ein und *LW* versucht jeden gefundenen String zu ersetzen. Wird das Kommando nicht von einem Makro aus gestartet wird beim ersten gefundenen String folgendes Menü erscheinen:

Change,	**All,**	**To End,**	**Skip?**
Ändern,	**Alle,**	**Bis zum Ende,**	**Überspringen?**

Wähle die gewünschte Funktion durch den hervorgehobenen Buchstaben oder verlasse die Funktion mit **<ESCape>**. „**Change**“ ersetzt den gefundenen String und sucht den Nächsten. „**All**“ ersetzt alle gefundenen Strings im *gesamten* Dokument. “**To End"** ersetz alle Strings von der aktuellen Cursorposition bis zum Ende des Textes. „**Skip**“ lässt den aktuellen Fund unverändert und sucht den nächsten String.

Nach dem globalen Ersetzen wird *LW* den Cursor stets an den Ausgangspunkt im Text setzen von wo aus das globale Ersetzen gestartet wurde. Während des globalen Ersetzen wird die Bildschirmanzeige nicht aktualisiert, ein Zähler zeigt jedoch die Anzahl der Ersetzungen an. Du kannst das Suchen & Ersetzen jederzeit mit der **<BREAK>**-Taste abbrechen.

Ob beim Suchen & Ersetzen Groß- und Kleinschreibung berücksichtigt wird legst Du u.A. mit **<SHIFT+CTRL+U>** fest. Beantwortest Du „Match Case“ mit „N“ wird *LW* bei der Suche *nicht* zwischen Groß- und Kleinschreibung unterscheiden. Du kannst an dieser Stelle auch die

Verwendung von Platzhaltern (Wildcards) unterbinden, wodurch die Suche nach dem inversen „?“ ermöglicht wird.

2.5.1 SUCHEN MIT PLATZHALTERN

Im Suche-String steht das inverse Fragezeichen (?) für ein beliebiges Zeichen, genau wie bei DOS-Dateinamen:

Find>H?SE

...passt sowohl auf „HASE“ als auch auf „HOSE“. Ein Platzhalter im Ersetze-String läßt das betreffende Zeichen unverändert:

Find>(?)

Change to>(?.)

setzt einen Punkt hinter jedes einzeln eingeklammerte Zeichen.

Willst Du sicherzustellen dass nur *ganze* Wörter gefunden werden setze den Suche-String zwischen zwei Leerzeichen. Für den Fall das Wörter von Satzeichen gefolgt werden kann ein Makro erstellt werden welches mehrfaches Suchen und Ersetzen durchführt. Siehe dazu Abschnitt 7 „Makros“.

Beachte: wenn Du nach dem inversen Fragezeichen suchen und es ersetzen lassen willst musst Du vorher mit **<SHIFT+CTRL+U>** „Wildcards“ mit „N“ deaktivieren (siehe oben).

3 ZUSÄTZLICHE FUNKTIONEN DES EDITORS

LW beinhaltet viele nützliche und hilfreiche Funktionen die Dich beim Editieren von Text unterstützen, z.B. Platzhalter, Funktionen zur Seitennummerierung oder Tabulatoren. Der Umfang der verfügbaren Hilfsmittel machen *LW* zu einer ultimativen Textverarbeitung für den Atari.

3.1 WÖRTER ZÄHLEN

LW's schnell arbeitende Wortzählung wird Dir unverzüglich sagen wie viele Wörter Dein Dokument gerade umfasst.

<CTRL+N> Zeigt Dir die Position des Cursors in der Datei und die Anzahl der Bytes der Datei an. Anschließend wird die Anzahl der Wörter und der belegten Bytes angezeigt; wenn Text markiert ist beziehen sich die letzten zwei Angaben auf den markierten Text.

Anders als viele andere Textverarbeitungen zählt *LW nur* den Text und *nicht* die Steuerzeichen für den Drucker. Die Zählfunktion ignoriert alle invertierten Zeichen. Die Definitionen von Kopf- und Fußzeilen sowie Argumente mit Dateinamen werden allerdings mitgezählt da diese mit normalen Zeichen eingegeben werden. Dies musst Du bei der Zählung der Wörter berücksichtigen.

3.2 INDIKATOR FÜR EDITIERTEN TEXT

Wenn ein Text in einer von *LW*´s Speicherbänken geändert wurde ohne ihn zu speichern wird dies in der Info-Zeile durch einen Stern links vom Dateinamen angezeigt. Das erinnert Dich daran die ungesicherte Arbeit zu speichern. Die Markierung verschwindet sobald der Text gespeichert wurde.

3.3 TABULATOREN

Du kannst *LW*'s Tabulatorleiste mit eigenen Tabulatorpunkten versehen welche in der Konfigurationsdatei gespeichert werden können. Das sind die Befehle zum Editieren der Tabulatorleiste:

<SHIFT+TAB> Setzt einen Tabulator an der Cursorposition.

<CTRL+TAB> Löscht einen Tabulator an der Cursorposition.

<SHIFT+CTRL+TAB> Stellt die Grundeinstellung der Tabulatoren wieder her.

<SHIFT+CTRL+E> Löscht ALLE Tabulatoren.

3.3.1 TABULATOR MODUS

Im Einfügemodus (Ins) fügt die **<TAB>**-Taste Leerzeichen bis zum nächsten Tabulator ein.
Im Überschreibmodus (Ovr) springt **<TAB>** einfach zum nächsten Tabulator.

3.4 LESEZEICHEN

LW bietet unsichtbare Lesezeichen die das Bewegen im Text sehr vereinfachen. Wenn Du an einer Stelle im Text arbeitest die Du verlassen möchtest, später aber an sie zurückkehren willst, markiere sie einfach mit einem Lesezeichen.

<CTRL+B> Setzt eines von vier Lesezeichen an die Cursorposition. Wähle Lesezeichen 1..4.

<SHIFT+CTRL+G> **G**ehe zu Lesezeichen. Fragt nach der Nummer des gesuchten Lesezeichens. Vorausgesetzt das Lesezeichen wurde gesetzt und befindet sich nicht in einem gelöschten Teil des Textes, springt der Cursor an die gewünschte Stelle.

Wenn Du im **Dokument-Modus** arbeitest werden Lesezeichen mit der Datei gespeichert.

3.5 TEXT- UND DOKUMENT-MODUS

LW kennt zwei verschiedenen Formate um eine Datei zu speichern: als Text-Datei (**.TXT**) oder als Dokument-Datei (**.DOC**). Während TXT-Dateien *reine* Textdateien sind, enthalten DOC-Dateien die Tabulatorleiste und alle gesetzten Markierungen. Du kannst diese Kopfinformationen in *LW* nicht sehen, da *LW* sie nicht anzeigt. *LW* lädt Dateien automatisch im richtigen Format (TXT oder DOC), ungeachtet des Extenders. Wenn Du *LW* als Editor zum Erstellen eines Quelltext für einen Compiler o.ä. nutzt, solltest Du die Datei *immer* als reine Textdatei (.TXT) speichern!

Speicherst Du eine Datei, wird *LW* die DOC-Informationen nur speichern wenn *LW* im „DOC“-Modus ist. Du kannst in der Statusleiste ablesen in welchem Modus sich *LW* gerade befindet. Wechseln zwischen DOC und TXT-Modus kannst Du, wie im folgenden Abschnitt 3.6 „Benutzereinstellungen“ beschrieben, mit **<SHIFT+CTRL+U>**.

3.6 BENUTZEREINSTELLUNGEN

LW bietet viele Optionen und Einstellungsmöglichkeiten. Du erreichst sie mit dem Kommando **<SHIFT+CTRL+U>** Benutzereinstellungen (**U**ser Options)

Dieses Kommando öffnet eine Liste mit Optionen, die ein- oder ausgeschaltet werden können. Die aktuelle Einstellung steht in eckigen Klammern.

Option	Bedeutung	Default
Match case [N] (Y/N)?	Unterscheidet zwischen Gross- und Kleinschreibung bei den Suchefunktionen	Nein
Warnings [Y] (Y/N)?	Gibt vor dem Beenden des Editierens oder vor dem Überschreiben einer existierende Datei eine Warnung aus.	Ja
Wildcards [Y] (Y/N)?	Benutze „?“ als Platzhalter beim Suchen & Ersetzen	Ja
False spaces [N] (Y/N)?	Zeigt Füllzeichen im Editor an	Ja
Doc mode [N] (Y/N)?	Arbeitet im Dokumenten-Modus	Nein

Möchtest Du ohne Änderung zur nächsten Option gehen drücke **<RETURN>**. Schalte die Funktion mit **<Y>** ein oder mit **<N>** aus. **<ESC>** bringt Dich jeder Zeit zurück zum Editor.

3.7 EDITIEREN VON MEHREREN DATEIEN

LW erlaubt es Dir auf Rechnern mit erweitertem Speicher mehrere Dateien gleichzeitig zu editieren. Wie Du an Deine Speicherkonfiguration anpasst wird später in Kapitel 8: „LW einrichten“ erklärt. Wenn Du *DOS 2.5*, *MyDOS* oder *SpartaDOS X* ohne benutzerdefinierten *LW.SYS*-File (welcher den erweiterten Speicher einrichtet) benutzt, wird *LW* jede RAM-Bank, die nicht für eine RAMdisk verwendet wird, nutzen. Welche und wie viele Bänke *LW* nutzt kannst Du beeinflussen indem Du eine angepasste *LW.SYS*-Konfigurationsdatei erstellst. Wenn Du ein nicht direkt von *LW* unterstütztes DOS verwendest ist das ein grundlegender Schritt: mit einem solchen DOS wird *LW* zwar herausfinden wie groß der Speicher des Computers ist, wird aber keine Bank nutzen solange Du es ihm nicht sagst. Auf diese Weise kannst Du bestimmten Bänken für eine RAMdisk verwenden und *LW* anweisen die übrigen Bänke zu nutzen. Die mitgelieferten *SYS*-Dateien beinhalten verschiedene Beispielkonfigurationen. Um sie zu nutzen, benenne die gewünschte Config-Datei in „*LW.SYS*“ um und lade *LW* vom DOS aus. Benutzt Du ein *unterstütztes* DOS übernimmt *LW* die ganze Arbeit für Dich.

Zugriff auf die erweiterten Text-Bänke erhältst Du mit...

<SHIFT+CTRL+n> wähle Speicherbank

wobei <n> eine Ziffer von 1 bis 9 oder 0 (welche für 10 steht) ist. Beachte, dass Bänke über 5 nur möglich sind wenn LW für Rechner mit **mindestens** 192k eingerichtet ist (siehe Kapitel 8: „LW einrichten“). Bank 1 (der Hauptspeicher) ist IMMER die Hauptbank, also sind maximal 9 zusätzliche Bänke möglich. Jede Bank hat eine Kapazität von 16k, eigene Markierungen und einen eigenen Dateinamen. Ausschneiden und Einfügen von Text funktioniert natürlich auch zwischen den Textbänken. Wenn Du ein grosses, auf mehrere Dateien verteiltes Dokument in verschiedenen Bänken ablegst und in der Hauptbank der verketteten Bänke das Druckkommando anwendest, werden die Seiten nummeriert als wenn Du *eine* große, durchgehende Datei bearbeitest.

Wenn Dein Atari XL/XE kein Zusatz-RAM hat oder Du es nicht benutzen möchtest wird Dir *eine* 16K grosse Text-Bank zur Verfügung gestellt und die Puffer für Kopieren, Makros und das Diskdirectory sind sehr klein (jeweils etwa 1k).

3.8 DER UMGANG MIT GROSSEN DATEIN

Obwohl LW auf einem Rechner mit Z-RAM einen Textbuffer von maximal 18k bietet (und die zusätzlichen Bänke auf 16k festgelegt sind) ist es trotzdem möglich größere Dateien zu bearbeiten indem sie auf mehrere Bänke aufgeteilt werden. Hierfür können Textbänke verschiedene Dateien, verschiedene Segmente einer grossen Datei oder eine Mischung aus beiden enthalten. Selbst wenn Du eine Maschine ohne Z-RAM mit nur einer 16k Textbank nutzt kannst Du trotzdem grössere Dateien bearbeiten.

Wenn Du eine Datei lädst die nicht vollständig in den Speicher passt erscheint die Nachricht „Linked”. Der Puffer wird so viel fassen wie er kann, abzüglich 255 Bytes freien Platzes zum Editieren (Beachte: Aufgrund der Arbeitsweise von LW wird jede Datei, die länger als die Grösse des Buffers minus 255 ist, als “Linked” gekennzeichnet, auch wenn sie *eigentlich* in den Buffer passen würde).

Um das versehentliche Löschen der Originaldatei auf der Diskette zu verhindern wenn das erste Segment geladen wurde ist bei verketteten Segmenten die „Auto Save“-Funktion

<CTRL+S> außer Funktion. Stattdessen gelangst Du mit **<CTRL+S>** *immer* zur „Speichern als“-Eingabeaufforderung, wie mit **<SHIFT+CTRL+S>**. Dadurch wirst Du gewarnt bevor Du eine bestehende Datei überschreibst. Darüber hinaus liegt es in Deiner Verantwortung darauf zu achten dass die Segmente in der richtigen Reihenfolge gespeichert werden.

Jetzt haben wir den ersten Teil einer geteilten Datei geladen und wir können unsere Arbeit auf zwei Weisen fortsetzen:

1. Bearbeite das erste Segment, speichere es unter einem neuen Namen ab und wiederhole dies bis alle Segmente geladen, bearbeitet und an die neue Datei angehängt wurden.

2. Lade alle Segmente der Originaldatei in getrennte Bänke, bearbeite sie gleichzeitig und speichere sie in der richtigen Reihenfolge; entweder in eine neue Datei oder überschreibe die originale Datei.

In beiden Fällen ist die Vorgehensweise beim LADEN des zweiten und aller folgenden Segmente der verketteten Datei gleich: hänge dem Dateinamen ein „**,C**“ (Komma, dann C, ohne Anführungszeichen!) an. Zum Beispiel:

Load>REPORT.DOC,C

...lädt das nächste Segment der Datei unter der Voraussetzung dass der angegebene Dateiname *immer* der der Originaldatei ist. Der Name der „letzten geladenen Datei“ kann mit **<CTRL+L>** der Eingabezeile entnommen werden. Das ist eine hilfreiche Abkürzung wenn man mehrere Segmente von der selben Datei laden muss. Lädst Du das letzte Segment der Datei wird die „Linked“-Meldung nicht angezeigt und ein „,**C**“ wird lediglich eine „End of File“-Fehlermeldung (Dateiende erreicht) ergeben.

Alle verketteten Segmente, mit Ausnahme des ersten Segmentes, sollten mit der Option „**,A**“ gespeichert werden. Ein Beispiel:

Save As>D:REPORT.DOC,A

...wird den Inhalt des Textpuffers an die Datei auf der Diskette anhängen (die Option „/A“ kann auch beim Kopieren von Dateien im Diskettenmenü verwendet werden). Wenn Du verkettete Segmente speicherst wird zur Zeitersparnis automatisch „/A“ an die Dateinamen angehängt, außer beim ersten Segment. Als weiter Hilfe beim Speichern von Segmenten einer segmentierten Datei wird dem Dateinamen in der Informationszeile eine fortlaufende Nummer angehängt. Zum Beispiel:

2:D:THESIS.TXT[2]

...bezeichnet Bank 2, die das zweite Segment der Datei THESIS.TXT enthält. In diesem Fall wird der Anhang [2] sogar beibehalten wenn der Inhalt von Bank 2 mit einem neuen Dateinamen (ohne „**,A**“) gespeichert oder der Textbuffer gelöscht wird.

Mit den verketteten Dateien bietet *LW* eine Lösung bei dem Problem große Dateien zu bearbeiten, der beste Weg ist jedoch die Dateien so zu organisieren dass sie bequem in den Textpuffer passen. Die Möglichkeit des Druckformatierers Dateien einzubinden / zu verketten ist es eine einfache und gute Möglichkeit die Dateigröße unter 16k zu halten und sie beim Ausdruck zu verketten. Der Hauptgrund für die Unterstützung von segmentierten Dateien in *LW* ist die Möglichkeit lange, unhandliche Dateien in kleinere Dateien zu zerlegen.

Auf der *LW*-Systemdiskette befindet sich das Makro „*SPLIT.MAC*“, welches genau diese Arbeit ausführt.

4 DISKETTENOPERATIONEN

LW bietet umfassende Möglichkeiten bei der Bearbeitung von Dateien / Ordnern und unterstützt viele verschiedene DOS. Das DOS-Menü erlaubt das Ansehen, Laden, Löschen, Umbenennen und Kopieren von Dateien mit nur einem Tastendruck. Das Menü öffnet ein Fenster welches bis zu 80 Dateinamen fasst. Die Vorschau zeigt Dateien an wie sie im Editor erscheinen *ohne* sie in den Speicher zu laden.

4.1 DISKETTENOPERATIONEN VOM EDITOR AUS

Zusätzlich zu **<CTRL+L>***aden*, **<CTRL+S>***peichern* und **<SHIFT+CTRL+S>***peichern als* sind folgenden Funktionen vom Editor aus verfügbar:

<CTRL+I> Datei einfügen (**i**nsert) oder verbinden. Ermöglicht das Einfügen eines Textes in den im Speicher befindlichen Text. Wenn Du keinen Extender angibst wird dem Dateiname der normalen Extender angehängt. Wenn die Datei die Du versuchst einzufügen den verfügbaren Platz übersteigt bleibt der Text unverändert.

<SHIFT+CTRL+I> Schreibt einen Block in eine Datei. Gib einen Dateinamen an und der Block - egal wie lang - wird auf Diskette geschrieben. Gibst Du keinen Extender an wird der Datei „.BLK“ angehängt. Diese Möglichkeit, zusammen mit dem **<CTRL+I>**-Kommando, erlaubt es große Textblöcke zwischen Dateien zu bewegen.

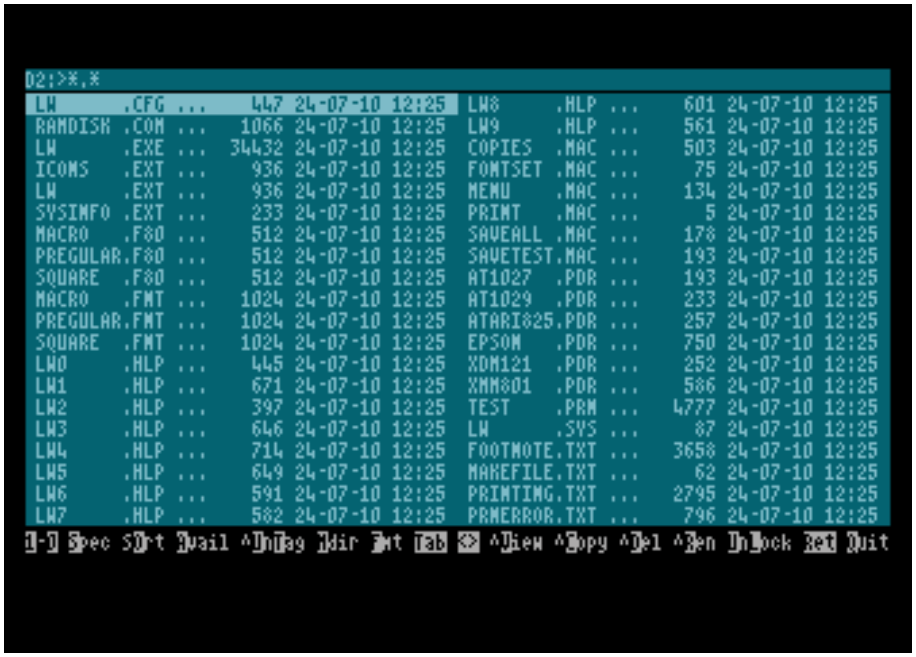
<CTRL+J> Datei ansehen. Erlaubt Dir vom Editor aus einen Dateinamen einzugeben und diese Datei in einem scrollenden Fenster mit Wortumbruch zu betrachten. Du kannst das Listing mit **<CTRL+1>** oder mit einer der drei Funktionstasten anhalten. Die Vorschau kann jederzeit mit der **<Break>**-Taste abgebrochen werden. Wenn Du ein „**,P**“ an den Dateinamen anhängst wird der Text seitenweise statt scrollend angezeigt.

4.2 DAS DISKETTENMENÜ

Die Funktionen des Diskettenmenüs erreichst Du durch die im unteren Menü hervorgehobenen Buchstaben. Der hervorgehobene Balken wird mit den Cursortasten (mit oder ohne <CTRL>) bewegt. Wenn Du den Rand der Seite erreichst wird auf die nächste Seite gesprungen.

<CTRL+D> Ruft das Diskettenmenü auf. Das Programm ließt das aktuelle Inhaltsverzeichnis ein und zeigt im 80-Zeichen-Modus maximal 80 Dateinamen an.

<SHIFT+CTRL+H> Wie oben, ermöglicht aber vor dem Aufruf des Inhaltsverzeichnisses über eine Eingabemaske Angaben zum Laufwerk & zum Pfad.



Folgenden Optionen sind durch Druck auf die hervorgehobenen Buchstaben verfügbar:

Spec Legt eine Suchmaske für das Inhaltsverzeichnis fest. Nutze es um die Suche auszuweiten oder zu verfeinern.

Ret Wurde das Diskettenmenü vom Editor aus mit **<CTRL+D>** aufgerufen wird mit <RETURN> die hervorgehobene Datei geladen. Wurde das Menü in der Eingabeaufforderung für Dateinamen mit **<TAB>** aufgerufen übergibt <RETURN> den hervorgehobenen Dateinamen zur Benutzung mit der aktuellen Lade- oder Speicheroperation. Wenn Du gerade eine Datei ausgibst oder speicherst kommst Du zur ursprünglichen Eingabezeile zurück, wobei der ausgewählte Dateiname den Ursprünglichen ersetzt.

View Zeigt die ausgewählte Datei als Vorschau an. Identisch mit der Vorschau aus dem Editor ohne Option „**P**“.

Beachte: Du kannst mit **<CTRL+V>** eine seitenweise Vorschau (wie mit der Option „P“ vom Editor aus) erreichen.

Del Löscht die markierte Datei. War das Löschen erfolgreich verschwindet die Datei von der Liste. Nutze **<CTRL+D>** um die mit <SPACE> markierten Dateien zu löschen.

Ren Fragt nach einem neuen Dateinamen und nennt die markierte Datei um. Der Eintrag in der Liste wird durch den neuen Namen ersetzt. Platzhalter werden unterstützt. Benutze **<CTRL+R>** um alle mit <SPACE> markierten Dateien umzubenennen.

^Copy Fragt nach dem Dateinamen unter dem der Inhalt der markierten Datei kopiert werden soll. Du kannst eine andere Laufwerksnummer angeben, Unterverzeichnisse verwenden (wenn das DOS sie unterstützt) und auch Platzhalter verwenden. Möchtest Du eine Kopie mit dem selben Namen auf einem anderen Laufwerk erstellen gib die Laufwerks-kennung, gefolgt von „*.*“, ein. Es können Dateien beliebiger Länge kopiert werden, auch wenn sie nicht in den Editor passen.

BEACHTE: Die Kopierfunktion nutzt den freien Platz der aktiven Textbank als Puffer. Je mehr Platz dort ist, desto schneller wird kopiert, also wähle bevor Du etwas kopierst eine möglichst leere Bank. Eine randvolle Bank verfügt über ein Reservebyte, also wird das Kopieren auch mit ihr funktionieren, allerdings quälend langsam!

Dem Zielnamen bei einer Kopie können zwei Schalter angehängt werden: „,A“ hängt die Quelldatei an die Zieldatei an, mit „,N“ wird die Überprüfung auf bereits vorhandene Ziel-Dateinamen unterdrückt, sofern die Warnungen nicht bereits mit Hilfe der Editor-Optionen (<SHIFT+CTRL+U>) deaktiviert wurden. In dem unwahrscheinlichen Fall daß beide Schalter *zusammen* gesetzt werden, hänge beide Schalter kombiniert, mit einem Komma getrennt, an den Dateinamen an (z.B. „,AN“).

Drücke **<CTRL+C>** um alle markierte Dateien zu kopieren. Platzhalter werden unterstützt. Du kannst z.B. alle Dateien auf der Diskette markieren , **<CTRL+C>** drücken um sie nach „D2:*.BAK“ zu kopieren. Alle Dateien auf dem Ziellaufwerk werden den Extender „.BAK“ haben. Um eine Auswahl Dateien nach Laufwerk 8 zu kopieren ohne Warnung daß ein File eventuell bereits existiert, markiere die gewünschten Files, drücke <CTRL+C>, dann gebe...

D8:*.*,N

oder auch nur

D8:,N ...ein.

Die Files werden mit ihrem echten Namen kopiert, etwaige bereits vorhandene Dateien mit dem selben Namen auf Laufwerk 8 werden überschrieben.

Mdir Legt ein neues Verzeichnis im aktuellen Verzeichnis an, vorausgesetzt das DOS unterstützt Unterverzeichnisse.

Esc Verläßt das Menü.

UnLock Aktiviert (**L**ock) bzw. deaktiviert (**U**nlock) den Schutz der markierten Datei(n). Beachte daß „L“ und „U“ ohne <CTRL> gedrückt werden.

^UnTag Markierung auf die hervorgehobene Datei mit **<T>** setzen oder löschen. **<CTRL+T>** markiert *alle* Dateien, **<CTRL+U>** hebt *alle* Markierungen auf.

Format Formatiert die Diskette nach einer Sicherheitsabfrage.

Quit Verlässt das Programm und springt in das DOS.

1-0 Zeigt den Inhalt eines Laufwerkes. 1-9 entspricht der Laufwerksnummer, 0 steht für ein Laufwerk ohne Nummer („D:“). Das ist wichtig wenn Du ein Unterverzeichnis in *MyDOS* öffnen möchtest.

SOrt Öffnet ein Menü welches die Möglichkeit bietet nach Name(n), Extender, Date (Datum) oder Size (Größe) sortieren zu lassen. None (Nichts) schaltet die Sortierung aus. Jede andere Taste verlässt die Eingabe ohne Änderung der im Konfigurationsfile festgelegten Art der Sortierung.

Avail Zeigt unter *DOS 2.5* die Anzahl der verbleibenden freien Sektoren auf Diskette an. Unter *SpartaDOS X* wird die Anzahl der freien Bytes der Diskette angezeigt.

Tab Wenn Du *SpartaDOS X* nutzt wechselt **<Tab>** zwischen kurzem und langen Directory.

Wird *LW* 3.2 auf einem Emulator benutzt wechselt der Anzeigemodus bei einer gepatchten Festplatte (Hn:) zeitweise automatisch zum kurzen Modus (*DOS 2.5*).

4.2.1 ZUSÄTZLICHE KOMMANDOS

Es gibt weitere Kommandos die im Diskettenmenü nicht aufgeführt werden.

<CTRL+H> Cursor an den Anfang (**h**ome). Bewegt den Cursor auf den ersten Dateinamen im Inhaltsverzeichnis.

<CTRL+E> Cursor an das **E**nde. Bewegt den Cursor zum letzten Dateinamen.

<SHIFT+CTRL+ ↑ > Blättert einen Bildschirm aufwärts.

<SHIFT+CTRL+ ↓ > Blättert einen Bildschirm abwärts.

4.2.2 UNTERVERZEICHNISSE

Die folgenden Optionen arbeiten nur mit einem DOS das Unterverzeichnisse unterstützt.

<RETURN> oder **>** Wechselt in das markierte Unterverzeichnis.

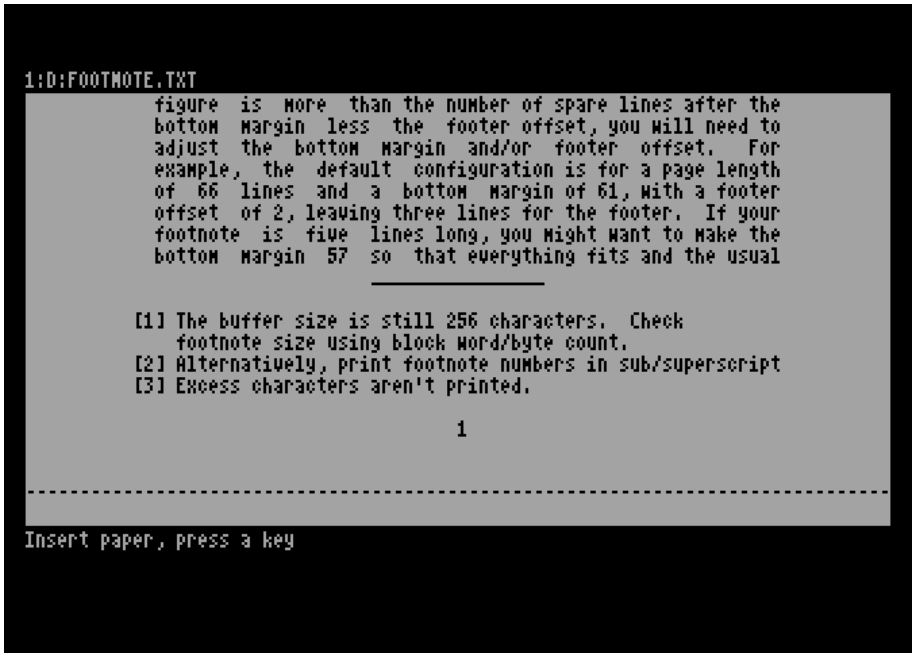
< Wechselt in das übergeordnete Verzeichnis.

5 DRUCKEN MIT LW

*LW´*s Druckfunktionen gehören zu den Umfangreichsten die es für Textverarbeitungen auf dem Atari Computer gibt. Unzählige nützliche Features erleichtern das Drucken von umfangreichen Dokumenten.

5.1 TEXTVORSCHAU

<SHIFT+CTRL+P> Zeigt (**p**review) den zu druckenden Text an. Der Text wird, weitgehend so wie er auf dem Drucker ausgegeben wird, in einem 20 x 80 Zeichen grossen Fenster dargestellt. Seitenumbrüche erscheinen als eine Reihe von Punkten. Wenn die Seitenpause aktiviert ist (siehe Abschnitt 5.4.1 „Ebene 1-Kommandos“) wird nach jeder Seite auf einen Tastendruck gewartet. Die Ausgabe kann mit **<CTRL+1>**, **<SELECT>** oder **<OPTION>** angehalten werden. Mit **<START>** springst Du an den Anfang der nächsten Seite, **<BREAK>** bringt Dich zurück in den Editor.



5.2 DIE SEITENNUMMERIERNG IM AUGE BEHALTEN

<CTRL + ?> „Wo ist der Cursor auf der gedruckten Seite?“ Das ist eine Innovation die man von *TextPro* kennt: es wird die Seite und die Zeile angegeben in der sich der Cursor auf der *gedruckten* Seite gerade befindet.

5.3 DRUCK-KOMMANDOS

<CTRL+P> Drucke (**p**rint) Text. Ruft ein Menü mit drei Optionen auf: „**P**“ schickt den Text direkt zum Drucker. „**S**“ startet, wie auch **<SHIFT+CTRL+P>**, die Vorschau auf dem Bildschirm. „**D**“ erlaubt, nach Eingabe eines Dateinamens, den formatierten Text zu einer Diskette oder einem anderen Gerät zu schicken (die Voreinstellung des Extender ist „.PRN“). Beachte das bei einem Ausdruck die Vorschau immer aktiviert wird. Die Ausgabe kann mit **<BREAK>** abgebrochen werden.

Die oben genannten Kommandos arbeiten auch mit verketteten Dateien (siehe Kapitel 3.7 „Editieren von mehrere Datein“) es sei denn Du kommentierst sie aus. Das bedeutet dass Du immer genau weißt *wo* Du Dich auf einer gedruckten Seite befindest, selbst wenn das Dokument aus mehreren einzelnen Dateien besteht.

5.4 EINGELAGERTE KOMMANDOS

LW bietet eine Fülle von Befehlen zur Formatierung mit welchen Du den Ausdruck genau auf Deine Bedürfnissen zuschneiden kannst. Die Formatierungskommandos folgen einfachen Regeln:

- Formatierungskommandos bestehen aus 1 oder 2 **invers eingegebenen** alphanumerischen Zeichen, oft folgen Numerische oder Textargumente.
- Formatierungskommandos können in Groß- oder Kleinbuchstaben eingegeben werden.
- **Numerische Argumente von Formatierungskommandos werden invers eingegeben.**
- String-Argumente (Fußnoten, Kopfnoten und Dateinamen) werden in normalen Zeichen (nicht invertiert) eingegeben.
- Ebene 1-Formatierungskommandos, sowohl einzeln als auch in Gruppen, müssen am Anfang einer Zeile stehen. Sie *können* mit <RETURN> abgeschlossen werden. Beachte: Wenn der String eines Ebene 1 Formatierungskommandos leer ist und mit mit <RETURN> endet, wird KEINE Leerzeile (Linefeed) ausgedruckt.
- Formatierungskommandos dürfen keine unnötigen Leerzeichen enthalten.

Hier einige Beispiele für Kommandos zum Formatieren des Ausdrucks:

l20<RETURN>

Setzt den linken Rand auf 20.

l20r60Hallo<RETURN>

Setzt den linken Rand auf 20, den rechten Rand auf 60 und druckt dann 20 Leerzeichen vom rechten Rand entfernt „Hallo“ .

fcSeite #<RETURN>

Definiert eine fortlaufende Fußzeile mit dem Text „Seite “ und der aktuellen Seitennummer.

5.4.1 EBENE 1 KOMMANDOS

Die folgenden Kommandos, eingegeben in inversen Groß- oder Kleinbuchstaben, beeinflussen die Größe und das Layout der Seite. Sie sollten grundsätzlich am Anfang einer Zeile stehen. Werden numerische Argumente (n) benötigt, sind diese ebenfalls invers einzugeben, direkt nach dem Kommando. Es können mehrere Kommandos hintereinander in einer Zeile angegeben werden. Kommandos können von einem <RETURN> (welches NICHT ausgedruckt wird) gefolgt werden.

a<n> Erste zu druckende Seite. **a2** wird die Ausgabe bei Seite 2 beginnen. Grundeinstellung ist 1.

b<n> Setzt den unteren (**b**ottom) Rand. Die Grundeinstellung ist 61. Gemessen in Zeilen vom Beginn der Seite an, wird die letzte Zeile angegeben in der Text gedruckt wird. Mit einer Seitenlänge von 66 Zeilen würde ein unterer Rand von 61 fünf freie Zeilen am Ende jeder Seite bedeuten.

Versichere Dich dass Du genug Zeilen (bis zu acht) für Deine Fußzeile übrig lässt. Wenn sie nicht passt wird sie nicht gedruckt!

f<[n]Text> Definiert einen fortlaufenden **F**ußtext der am Ende jeder Seite gedruckt wird. <n> ist ein optionaler Offset, angegeben in Zeilen, gezählt vom mit **b**<n> festgelegten unteren Rand aus, und wird invers unmittelbar *vor* dem Text der Fußzeile angegeben. Hier ein Beispiel:

f3Fußzeile<RETURN>

Gibt die fortlaufende Fußzeile „Fußzeile“ drei Zeilen unter der letzten Zeile des gedruckten Textes. Der Text der Fußzeile wird nicht invertiert eingegeben, außer wenn ein Formatierungskommando der Ebene 2 auftaucht (Kommandos der Ebene 1 können in Kopf- oder Fußzeilen nicht vorkommen), und muss mit <RETURN> beendet werden. Mit **#** wird die Seitennummer gedruckt. Eine Kopf- oder Fußzeile kann bis zu 8 Zeilen hoch sein, jeweils mit <RETURN> beendet. Jeder dieser Zeilen muß ein **f** vorausgehen und sie müssen direkt aufeinander folgen. Wird die Fußzeile irgendwo im Text neu definiert wird die bereits definierte Fußzeile verworfen. Um eine Fußzeile zu löschen gebe einfach nur **f** gefolgt von <RETURN> ein.

g<n> Holt (**g**et) eine Textbank. Sollte in einer eigenen Zeile stehen, gefolgt von <RETURN>. Der Inhalt der Bank wird gelesen und anstelle des Kommandos ausgedruckt.

g<fspec> Lädt (**g**et) eine Datei von Diskette. Sollte in einer eigenen Zeile stehen, gefolgt von <RETURN>. Der Inhalt der Datei wird gelesen und anstelle des Kommandos ausgedruckt. Das geht sehr schnell, auch wenn von Disk gelesen wird, weil zwei Buffer eingesetzt werden um langsame „Lese Single-Byte“-Kommandos zu vermeiden.

Es werden *alle* in der Datei vorkommenden Formatierungsbefehle ausgeführt. Der Vorteil dieser Methode gegenüber dem Verketten in vielen anderer Textverarbeitungen ist, dass sich beim Drucken eines verketteten Textes nach dem Druck wieder der Text im Speicher befindet von dem aus der Druck gestartet wurde. Du kannst in einer Hauptdatei mit Anweisungen zum Laden von Dateien den „Wo ist der Cursor?“-Befehl **<CTRL+?>** und die Vorschau **<SHIFT+CTRL+P>** nutzen und wirst immer die korrekte Seitennummerierung angezeigt bekommen. Beachte dass eine Verschachtelung aufgrund von Speichereinschränkungen bei der Pufferung NICHT möglich ist. So darf eine aufgerufene Datei nicht eine weitere Datei aufrufen, eine aufgerufene Text-Bank hingegen darf eine Datei aufrufen.

h<n/text> Definiert einen fortlaufenden Kopftext (**h**eader) der am Anfang jeder Seite gedruckt wird. Funktioniert genau wie die Fußzeile. Die optionale Ziffer legt den Offset vom oberen Rand der Seite fest; die Voreinstellung ist 2.

jl Der folgende Text wird linksbündig ausgerichtet (**J**ustify text **l**eft).

jr Der folgende Text wird rechtsbündig ausgerichtet (**J**ustify text **r**ight).

jc Der folgende Text wird zentriert (**J**ustify **c**entre).

jf Der folgende Text wird im Blocksatz (links & rechtsbündig) gedruckt (**J**ustify **f**ully).

k Add-In #1

l<n> Setzt den **l**inken Rand. Die Voreinstellung ist 10.

m<n> Rand (**m**argin) um <n> Zeichen ausrücken. Rückt die Textzeile, wie hier, aus. Die folgende Zeilen werden wieder am normalen Rand ausgerichtet. Die Zeile wird eventuell gedehnt um den Platz zu füllen. Dieser Abschnitt z.B. wird mit **>15** vom linken Rand eingerückt und die erste Zeile mit **m15** ausgerückt und so ein hängender Einzug erzeugt. BEACHTE: Der ausgerückte Teil der Zeile von einem Blocksatz-Kommando unberührt. Auch kann eine ausgerückte Zeile kann nicht zentriert oder rechtsbündig sein.

n<n> **N**eue Seite. Das optionale Argument veranlasst die neue Seite nur wenn weniger als <n> Zeilen auf der aktuellen Seite frei bleiben.

p<n> Länge der Seite (**p**age). Die Gesamtlänge der Seite, *inklusive* des oberen und unteren Randes. Die Grundeinstellung ist 66.

q Add-In #2

r<n> Setze **r**echten Rand. Bis zu dieser Stelle wird der Text gedruckt. Die Grundeinstellung ist 70.

t<n> Setze den oberen (**t**op) Rand. Die Grundeinstellung ist 5. Legt die Anzahl leerer Zeilen am Anfang jeder Seite fest. Wenn Du eine Kopfzeile einrichten möchtest richte hier genug freie Zeilen ein.

v<fspec> Verbose include file. Schickt die angegebene Datei, unabhängig von deren Inhalt, an den Drucker. Die Datei kann z.B. eine Grafik sein, auf diese Weise wird es Dir ermöglicht Bilder in Dein Dokument einzubinden. Diese werden jedoch in der Vorschau nicht angezeigt. Wenn Du eine Grafik einbindest stelle sicher dass die Seitenlänge und der unterer Rand korrekt eingerichtet sind.

w<n> Schaltet die Seitenpause (**w**ait) an (n=**1**), oder aus (n=**0**). Die Grundeinstellung wird in der Konfigurationsdatei festgelegt. Beim Ausdruck wird nach jeder Seite eine Pause eingelegt und auf einen Tastendruck gewartet. Funktioniert auch in der Druckvorschau. Drücke <ESC> in der Eingabeaufforderung um das Drucken oder die Vorschau abzubrechen. Beachte dass ein Tastendruck von einem Makro *nicht* akzeptiert wird. Ein laufendes Makro,, welches eine Eingabe verlangt, könnte sonst durcheinander kommen.

y<n> Zeilenabstand. Die Grundeinstellung von **1** bedeutet das zwei Textzeilen unmittelbar aufeinander folgen. Bei **2** wird mit zweifachen Zeilenabstand (ein Leerzeile zwischen zwei Textzeilen) drucken.

z<n> Letzte zu druckende Seite. Stoppt den Ausdruck bei Seite <n>.

><n> Absatz links ausrücken. Der folgende Text bis zum nächsten <RETURN> wird um <n> Zeichen vom linken Rand ausgerückt.

<<n> Absatz rechts ausrücken. Der folgende Text bis zum nächsten <RETURN> wird um <n> Zeichen vom rechten Rand ausgerückt.

[<n> Linker Rand Kopf/Fußzeile. Grundeinstellung ist 10. Arbeitet wie das **l** Kommando, legt aber den linken Rand für die Kopf/Fußzeilen fest, der unabhängig vom normalen linken Rand ist. Der Grund hierfür ist dass der normale linke und rechte Rand im Text variieren kann. Dadurch könnte es Unregelmäßigkeiten bei der Ausrichtung der Kopf- & Fußzeilen geben, die so vermieden werden.

]<n> Rechter Rand Kopf/Fußzeile. Grundeinstellung ist 70. Wie oben, aber für den rechten Rand.

?<n> Legt die Startnummer für die Seitennummerierung fest. Die Grundeinstellung ist **1**. Um die Seitennummerierung eines Dokumentes mit 3 zu beginnen setze <n> auf **3**.

@<n> Seitenauswahl. <n> steht für die Anzahl an Seiten die beim Ausdruck übersprungen werden sollen. Die Grundeinstellung ist 0. Dieses Kommando mit dem Argument **1** druckt nur die ungeraden Seiten eines Dokumentes aus. Um die geraden Seiten zu drucken, setze Seitenauswahl auf **1** und nutze **a2** um den Ausdruck mit Seite 2 zu starten. **1** druckt jede zweite Seite aus, **2** druckt jede dritte Seite aus, usw. Hiermit kannst Du z.B. in mehreren Druckgängen zweispaltige Seiten erstellen: Seite 1 enthält die *linke* Spalte von Druckseite 1, Seite 2 die *Rechte*, Seite 3 die *linke* Spalte von Druckseite 2, Seite 4 die *Rechte*. Dann druckst Du erst die geraden Seiten, legst das gerade bedruckte Papier wieder ein und druckst die ungeraden Seiten.
Wenn die Kopf/Fußzeilen für die Bindung versetzt sein sollen: Du kannst die ungeraden Seiten mit einer rechtsbündigen Fußzeile drucken, dann die Fußzeilen linksbündig setzen und anschließend die geraden Seiten drucken.

!<n> Gibt die Ebene der Überschrift an. <n> kann einen Wert von **1**-**9** annehmen. Erzeugt eine automatisch durchnummerierte Überschrift. Dem Kommando folgt ein Leerzeichen und ein Text für die Überschrift.

Angenommen Du möchtest Deinen Text wie folgt strukturieren (mit weiterem Text zwischen den Überschriften):

!1 TRANSPORT
!2 BUS
!2 BAHN
!1 AUSSTATTUNG
!2 BÜCHEREI
!2 FREIZEIT
!3 SCHWIMMEN
!3 ANDERE SPORTARTEN

So wird Dein Ausdruck aussehen:

1 TRANSPORT
1.1 BUS
1.2 BAHN
2 AUSSTATTUNG
2.1 BÜCHEREI
2.2 FREIZEIT
2.2.1 SCHWIMMEN
2.2.2 ANDERE SPORTARTEN

Der Druckformatierer wird, wenn Du den Text ausdruckst, die Nummerierung der Überschriften übernehmen. Wenn Du Dein Dokument neu organisierst brauchst Du Dich nicht um die korrekte Nummerierung der Überschriften zu kümmern.

& Reset Überschriften-Ebene. Setzt alle Überschriften-Ebene auf den Startwert „1“ zurück. Erlaubt es in einem Dokument mehrere Überschriften-Sequenzen zu verwenden.

5.4.2 EBENE 2 KOMMANDOS

Die folgenden Kommandos können überall im Text, auch in der Kopf- oder Fußzeile, eingesetzt werden und beeinflussen eine einzelne Zeile Text oder Zeichen. Einige benötigen Parameter, die meisten aber nicht. Ein geschickter Weg die Kommandos einzugeben ohne zweimal die **<INVERS>**-Taste betätigen zu müssen ist sie zusammen mit **<SELECT>** zu drücken.

Drucke Seitennummer. Gibt, eingefügt in die Kopf- oder Fußzeile, die aktuelle Seitennummer aus.

c Zeile zentrieren (**c**entre). Die *folgende Zeile* wird zentriert. Dieses Kommando kann verwendet werden um Kopf-, Fuß- oder eine einzelne Zeile zu zentrieren. Die zentrierte Zeile sollte mit einem <RETURN> beendet werden. Dieses Kommando muss nicht an erster Stelle der Zeile stehen - Du kannst Text in einer Zeile linksbündig, zentriert und rechtsbündig formatieren. BEACHTE: Dieses Kommando ist nicht das gleiche wie „centre justify“ **j**c welches sich auf den *gesamten* folgenden Text auswirkt. Wenn Du in einem Absatz eine einzelne Zeile zentrierst oder rechtsbündig (siehe **e**) formatierst wird eine eventuelle Ebene 1-Ausrichtung unterdrückt.

d Schaltet **Fettdruck** (**d**ouble strike) an oder aus. Schließe einen beliebigen Text der fettgedruckt werden soll zwischen zwei <d>-Zeichen ein, z.B. **d**das ist Fettdruck**d**. Diese Funktion wird durch den Druckertreiber-Editor bereitgestellt und hängt davon ab ob ein Drucker Fettdruck unterstützt.

e Rechtsbündig. Richtet den restlichen Text *der Zeile* rechtsbündig aus. Siehe auch „Zeile zentrieren“ **c**.

i Schaltet *Schrägschrift (**i**talic)* an oder aus. Siehe Fettdruck **d**.

s<n> Druck**s**til mit n=0-5 als inverse Ziffer. Sendet eine von 5 nicht auszudruckenden Steuersequenzen an den Drucker. Diese Sequenzen werden im Druckertreiber-Editor festgelegt und können aus bis zu jeweils 7 Bytes beliebigen Code bestehen. Eine praktische Lösung um Zeichensätze oder Druckstile auszuwählen die nicht von den Druckkommandos unterstützt werden (siehe Abschnitt 6.2.4 „Stile“). <**s1**> schaltet Stil 1 an, ein weiters <**s1**> schaltet Stil 1 wieder aus.

u Schaltet **U**nterstreichen ein oder aus. Siehe Fettdruck **d**.

o<n> ASCII-Zeichen-Ausgabe (**o**utput). Gibt den ASCII-Code <n> aus. Das Zeichen wird NICHT als druckbares Zeichen gezählt, beeinflusst also keine Formatierungen oder den Zeilenumbruch. Wird benötigt um Steuercodes, die nicht vom Druckertreiber abgedeckt werden, zum Drucker zu schicken.

x<n> Sende druckbaren Code. Arbeitet wie **o**utput ASCII, aber das Zeichen wird als druckbar gezählt und erscheint in der Vorschau als Fragezeichen. Nützlich um internationale Zeichen zu drucken die nicht vom Druckertreiber unterstützt werden.

-	Intelligenter Bindestrich (Gedankenstrich). Eingefügt in besonders lange Wörter. Passen diese Wörter beim Drucken nicht mehr in eine Zeile, wird das Wort an der markierten Stelle getrennt und ein Bindestrich an das Ende der Zeile gedruckt. Passt das Wort in die Zeile wird kein Bindestrich gedruckt.
_	Festes Bindezeichen (inverser Unterstrich). Normale Bindestriche erlauben es das Wort am Ende einer Zeile zu trennen. Benutze den festen Bindestrich um dies zu verhindern.
. **oder** _	Festes Leerzeichen (kann auch ein inverses Leerzeichen sein). Setze es zwischen Wörtern um zu verhindern dass sie am Ende einer Zeile evt. getrennt und *immer* in einer Zeile zusammen gedruckt werden. Eine schnelle Möglichkeit ein festes Leerzeichen einzugeben ist **<SHIFT +CTRL+SPACE>**.
(	Ignoriere bis zur schließenden Klammer). Alles was zwischen den Klammern steht wird beim Drucken ignoriert.
;	Kommentarzeile: alles bis zum nächsten <RETURN> wird beim Drucken ignoriert.
↑	Wechselt zwischen hochgestellt an oder aus
↓	Wechselt zwischen tiefgestellt an oder aus
←	Add-in #3
→	Add-in #4
*	Add-in #5
%	Add-in #6

5.4.3 EINEN HÄNGENDEN EINZUG ERSTELLEN

Mit den Absatzausrückung- und Randausrückung-Kommandos von *LW* ist es einfach einen hängenden Einzug zu erstellen. Möchtest Du zum Beispiel den nächsten Absatz 15 Zeichen einrücken, aber die erste Zeile bündig mit dem linken Rand ausrichten, füge einfach folgende Zeile ein:

>15m15<RETURN>

5.5 WEITERE DRUCKFUNKTIONEN

Leerzeichen am Ende einer nicht mit <RETURN> abgeschlossenen Zeile werden am Anfang der nächsten Zeile unterdrückt. Das Bedeutet, dass bei Sätzen mit zwei oder mehr Leerzeichen nach dem Punkt *kein* Leerzeichen am Anfang der nächsten Zeile gedruckt wird wenn der Zeilenumbruch direkt nach dem Punkt kommt.

Fehlende Argumente und illegale Kommandos halten den Druck an und werden mit einer Fehlermeldung quittiert.

5.5.1 INTERNATIONALER ZEICHENSATZ

LW unterstützt in der Vorschau den internationalen Zeichensatz des ATARI als direktes Gegenstück zum Ausdruck. Zeichen mit dem ASCII-Codes 0 - 26 sowie die Codes 96 und 123, die zum Drucker geschickt werden, können so umdefiniert werden dass sie dem internationalen Zeichensatz des ATARI entsprechen. Du kannst jedes beliebige Zeichen zuweisen, aber wenn es nicht den Zeichen des internationalen Standart-Zeichensatz entspricht wird es in der Vorschau nicht korrekt angezeigt. Diese Funktion wird im Druckertreiber-Editor eingerichtet (siehe Abschnitt 6.2.3 „Internationale Zeichen“).

5.6 DEN DRUCKFORMATIERER EINRICHTEN

Der Grundeinstellung des Druckformatierers (print formatter) für die Seitenränder kann in der CFG-Konfigurationsdatei eingerichtet werden:

Linker / Rechter Rand
Linker / Rechter Rand der Kopf- & Fußzeile
Oberer / Unterer Rand
Kopf- / Fußzeile Offset

Nähere Informationen findest Du in Abschnitt 8.2 „Die .CFG-Konfigurationsdatei“.

6 LW FÜR DEINEN DRUCKER EINRICHTEN

Du kannst *LW*´s Druckkommandos an nahezu jeden Drucker anpassen. Für Schrägschrift, Fettdruck, Unterstreichen, hoch- & tiefgestellt können Schalter eingerichtet werden. Bis zu fünf weitere Gestaltungskommandos können für jeden erdenklichen Zweck definiert werden.

6.1 DRUCKERTREIBER

LW kann mit Druckertreiberdateien (mit dem Extender „.PDR“) für verschiedene Drucker konfiguriert werden. Beim Programmstart wird *LW* versuchen die Datei „LW.PDR“ zu laden, die Einstellungen in dieser Datei sind nach Programmstart verfügbar. Wird die Datei „LW.PDR“ nicht gefunden nutzt *LW* die Grundeinstellung für den Drucker, wobei dann keine speziellen Formatierungen unterstützt werden und der Text „sauber“ an den Drucker geschickt wird. Du kannst während der Arbeit mit *LW* jederzeit einen Druckertreiber laden:

<SHIFT+CTRL+D> lade Druckertreiber (**d**river). Lädt nach der Eingabe des Dateinamen (wenn Du keinen Extender angibst wird „.PDR“ automatisch angehängt) den gewünschten Treiber.

Der Druckertreiber übersetzt die Formatierungskommandos für Schrägschrift, Unterstreichen, Fettdruck ect. ebenso wie internationale Zeichen, in den druckerspezifischen Code.

6.2 EINEN DRUCKERTREIBER ERSTELLEN

Der bei Vorgängerversionen von *LW* verwendete Druckertreibereditor „*LWPD.COM*“ ist bei *LW 3.x* nicht länger nötig. Druckertreiber werden jetzt als einfacher Text gespeichert. Auf der *LW*-Systemdiskette findest Du mehrere Druckertreiber, aber wenn Du die Escape-Sequenzen für Deinen Drucker kennst ist es einfach einen eigenen Treiber zu erstellen.

Druckertreiberkommandos müssen einzeln pro Zeile, abgeschlossen mit <RETURN>, angegeben werden. Folgende Anweisungen stehen zur Verfügung:

Anweisung	Argument	Bemerkung
UNDERLINE ON¦OFF	n,n,n...	Unterstreichen an/aus
ITALIC ON¦OFF	n,n,n...	Schrägschrift an/aus
BOLD ON¦OFF	n,n,n...	Fettdruck an/aus
SUPERSCRIPT ON¦OFF	n,n,n...	Hochgestellt an/aus
SUBSCRIPT ON¦OFF	n,n,n...	Tiefgestellt an/aus
INTERNATIONAL ON¦OFF	n,n,n...	Internationaler Zeichensatz an/aus
CRLF	n,n,n...	Wagenrücklauf/Zeilenvorschub-Codes
INIT	n,n,n...	Druckerinitialisierungs-Codes
CODE	char,n	Internationale Zeichen zuordnen
STYLE	<1 - 5>,n,n,n...	Stil 1 - 5 zuweisen

Lasst uns einen Druckertreiber für einen EPSON-kompatiblen STAR LC-10 erstellen (das ist eigentlich nicht nötig da ein EPSON-Treiber mitgeliefert wird, aber es ist eine gute Übung).

6.2.1 DRUCKER „SCHALTER“

Starte *LW* mit einer leeren Textdatei. In der Anleitung Deines Druckers findest Du die Codes mit denen Du die Schrägschrift einschaltest. Für EPSON-kompatible ist die Sequenz ist 27, 52. Gebe also im Editor

ITALICS ON 27,52 <RETURN> ein.

Schrägschrift aus würde 27,53 sein, also müsste die nächste Zeile

ITALICS OFF 27,53 <RETURN> lauten.

Alle 6 Paare mit An/Aus-Schaltern (Unterstreichen, Schrägschrift, Fettschrift, hochgestellt, tiefgestellt und internationaler Zeichensatz) funktionieren auf die selbe Weise. In einem Dokument wird das erste „**“** in einer Datei Schrägschrift einschalten, das Zweite aus, das Dritte ein, u.s.w.. Die einzige Ausnahme sind die „International An¦Aus”-Code-Sequenzen. Diese werden *automatisch* vor und nach jedem internationalen Zeichen im Dokument gesendet. Das bedeutet dass Du jeden im Drucker eingebauten Zeichensatz nutzen kannst *ohne* dass der Rest des Textes beeinflusst wird.

6.2.2 CONTROL-STRINGS

Die **CRLF**-Anweisung erlaubt Dir eine Escape-Sequenz für den Wagenrücklauf/Zeilenvorschub einzugeben die am Ende jeder Zeile an den Drucker geschickt wird. Für einen EPSON-kompatible Drucker lautet sie

CRLF 13,10

Dadurch wird am Ende jeder Zeile eine ASCII-Wagenrücklauf und Zeilenvorschub-Sequenz an den Drucker gesendet. Das ICD-Druckerkabel (entspricht z.B. dem CompyShop-Druckerinterface) schickt den Zeilenvorschub automatisch für Dich, also wird meistens ein

CRLF 155 ausreichen.

Das ist das normale Ende der Zeile-Zeichen (EoL) des Atari´s und die Grundeinstellung von *LW*, so dass Du die **CRLF**-Sequenz nur in den Druckertreiber einfügen musst wenn Du etwas anderes als 155 benötigst.

Zu Beginn jedes Dokumentes das Du an den Drucker schickst wird der **INIT**-Code gesendet. Möchtest Du einen Reset des Druckers vor dem Ausdruck ausführen gebe

INIT ein.

Hänge ein Leerzeichen und eine durch Kommata getrennte Liste des Escape-Codes aus dem Handbuch für Deinen Drucker an. Wenn der Initialisierungsstring *nicht* an den Drucker geschickt werden soll entferne die „INIT“-Zeile aus der Druckertreiberdatei.

6.2.3 INTERNATIONAL CHARACTERS

LW beinhaltet 29 spezielle „internationale“ Zeichen. Das sind die Zeichen **<CTRL+A>** bis **<CTRL+Z>**, **<CTRL + Komma>**, **<CTRL + Punkt>** und **<CTRL+ Semikolon>**. Wenn Du einen Zeichensatz nutzt der diese Zeichen mit akzentuierten oder internationalen Zeichen neu definiert ist es wichtig die selben Zeichen an den Drucker schicken zu können. Leider

stimmt der internationale EPSON-Zeichensatz und der ATASCII-Code des ATARIs in den wenigsten Fällen überein, aber durch die **CODE**-Anweisung kannst Du jedem der 29 internationalen ATASCII-Zeichen einen entsprechenden ASCII-Code zuweisen.

CODE 1,129

Schickt bei jedem **<CTRL+A>** im Dokument den ASCII-Code 129 an den Drucker. Du musst den Code für alle 29 Zeichen auf diese Weise festlegen.

Die Anleitung des STAR LC-10 bietet verschiedene, per Software oder mit DIP-Schaltern wählbare Zeichensätze. Wir wählen den IBM Zeichensatz #2, weil er die meisten Zeichen aus dem internationalen Zeichensatzes des Atari in den Codes 128-255 enthält. Normalerweise werden diese Zeichen als kursive Version der normalen Zeichen gedruckt, also müssen wir das IBM set #2 mit den DIP-Schaltern auswählen. Im LC-10 gibt es eine Reihe mit 12 DIP-Schaltern. Um die gewünschten Zeichen zu bekommen setze Schalter 1-6 (Printer Mode) auf ON (Standard), und 1-7 auf OFF (Graphics). Setzte die anderen Schalter nach Deinen Bedürfnissen. Im Zeichensatz des Drucker fehlt allerdings noch ein Zeichen ("u" mit Akut -> „ú“) welches wir per Software aus dem Drucker kitzeln müssen. Der länderspezifische Zeichensatz den wir für das „u“ Akut benötigen kann nicht mit den DIP-Schaltern gewählt werden, also muss der Druckertreiber den passenden Code vor dem internationalen Zeichen senden. Das geschieht mit der **INTERNATIONAL**-Anweisung:

INTERNATIONAL ON 27,82,12

INTERNATIONAL OFF 27,82,0

Jedes mal wenn ein internationales Zeichen an den Drucker geht wird vorher ASCII 27,82,12 zum Drucker gesendet um den IBM Zeichensatz #2 auszuwählen. Wenn das internationale Zeichen gedruckt wurde wird mit ASCII 27,82,0 wieder der originale Zeichensatz ausgewählt.

6.2.4 STILE

Du kannst bis zu fünf Druckstile einrichten die nicht von den Schräg/Fettdruck- und Unterstreichen-Kommandos abgedeckt werden:

STYLE1 ON 27,52

STYLE1 OFF 27,53

Das obige Beispiel definiert Style 1 als Schrägschrift An/Aus-Schalter, auch wenn das nicht sinnvoll ist da es bereits ein Kommando für Schrägschrift gibt.

Beachte daß der Format sich mit Version 3.2 geändert hat und daß Druckertreiber von früheren 3.x-Versionen von *LW NICHT mit Version 3.2 zusammenarbeiten wenn sie STYLE-Definitionen enthalten!* Sie müssen umgeschrieben werden um zum neuen System kompatibel zu sein. Die Änderung liegt darin begründet daß es dem Printformatierer möglich sein muß vor dem Druck einer Kopf- oder Fußzeile einen Druckstil gezielt abzuschalten. In früheren Versionen unterschied der Printformatierer nicht zwischen Kopf-/Fußzeile und normalen Text, so daß die Stile der Kopf-/Fußzeile den Haupttext beeinflussen konnten und umgekehrt. Dieses alte Problem ist ein guter Grund dafür auf die neuste Version von *LW* aufzusteigen.

Speichere Deinen Druckertreiber mit **<CTRL+S>** und nenne ihn *LW.PDR* wenn Du ihn beim Start von *LW* automatisch laden lassen möchtest.

Die Vorgehensweise für die meisten EPSON-kompatiblen Drucker sollte der oben gezeigten sehr ähnlich sein, wobei ich beim Erstellen dieser Anleitung leider keinen Zugang zu solchen Geräten hatte.

Ich benutze einen Canon BJ-200ex Tintenstrahldrucker im Epson-Emulationsmodus mit einem ICD-Druckerinterface, und der „*EPSON.PDR*“-Druckertreiber arbeitet perfekt mit dem Canon zusammen wenn die DIP-Schalter einmal korrekt eingestellt sind.

Ich habe auf der *LW*-Diskette sowohl den EPSON-Druckertreiber als auch Treiber für alle Atari-Drucker beigefügt. Obwohl ich keinen ATARI-Drucker besitze war es mir möglich die Codes herauszufinden, indem ich *AtariWriter Plus* glauben ließ dass einer angeschlossen sei, dann den Ausdruck auf Diskette gemacht habe und das Ergebnis untersuchte. Beachte dass nicht alle ATARI-Drucker mit Schrägschrift oder Fettdruck ausgestattet sind. Ich hoffe dass diese Treiber mit den Geräten funktionieren.

Es dürfte etwas Arbeit nötig sein um die internationalen Zeichen aus Deinem Drucker zu kitzeln. Falls Dein Drucker sie nicht unterstützt gibt es viele Utilities mit denen man Zeichensätze in einen Drucker laden kann. Lade einfach einen Zeichensatz herunter der den internationalen Zeichensatz des ATARIs nachbildet, passe den Druckertreiber an und es kann losgehen. Die Möglichkeit internationale Zeichen ohne großen Aufwand drucken zu können war ursprünglich einer der Hauptgründe warum *LW* entstanden ist. Ich wollte eine Textverarbeitung die sie sichtbar auf den Schirm bringt und keine speziellen Kommandos mitten im Text benötigt.

7 MAKROS

Mit seiner Makro-Fähigkeit ist *LW* eine der leistungsfähigsten Atari 8-Bit-Textverarbeitungen. Makros erlauben es Dir oft wiederkehrende Aufgaben zu automatisieren, das Tastaturlayout neu zu definieren, Textpassagen mit einem einzigen Tastendruck aufzurufen, interaktive Menüs anzulegen und gänzlich neue Kommandos durch die Kombination von existierenden Funktionen des Programms zu entwerfen. Damit nicht genug: Makros können jetzt mit einem normalen Tastendruck verknüpft werden so dass sie wie eingebaute Features des Programms arbeiten. Makros können auch den Neuaufbau des Bildschirmes und Eingabeaufforderungen unterdrücken, so dass Du nur das Endergebnis siehst.

LW's Makrokommandos sind die Zusammenfassung der Kommandos der Public Domain Textverarbeitung *TextPro*, und wer mit der Arbeitsweise der Makros in *TextPro* vertraut ist wird nur wenig Probleme mit deren Arbeitsweise in *LW* haben. Makros sind einfache Textdateien (KEINE „.DOC"-Dokumentdatei) die eine Makrodefinition nach der anderen enthalten, ohne Leerzeichen oder EOL-Zeichen dazwischen. Ein Makro wird wie folgt definiert:

<Makro ID>=<Makro Definition>

„Makro ID" ist der Tastendruck mit dem das Makro aufgerufen werden soll, das Gleichsetzungszeichen ist ein inverses Gleichzeichen, und „Makro Definition" ist einfach die Folge von Zeichen und Kommandos die das Makro enthalten soll. Makros werden ***nicht*** mit <RETURN> beendet. Das Ende einer Makrodefinition ist lediglich eine andere Makrodefinition oder das Ende der Makrodatei.

Eine Makrodatei kann bei Verwendung von BANKED memory bis zu 4K lang sein, ansonsten ist sie auf 1K begrenzt. Makros können verkettet werden, und es können gezielt Makros in der zu ladenden Makrodatei gestartet werden.

7.1 MAKROS LADEN

Makros werden mit **<SHIFT+CTRL+M>** geladen. Enthält die Datei ein Makro welches mit dem „&"-Zeichen verknüpft ist wird dieses Makro *unmittelbar* nachdem die Datei geladen wurde ausgeführt. Du kannst dies unterbinden indem Du ein „,N" an den Makro-Dateinamen in der Eingabezeile anhängst.

Wird ein Makro aus einem laufenden Makro heraus geladen ist es möglich ein anderes als das „&"-Makro starten zu lassen.

Beim ersten Start sucht *LW* nach der Makro-Datei „LW.MAC" und versucht ein eventuell vorhandenes „@"-Makro zu starten. Dieses „Start"-Makro wird nur gestartet wenn das Programm „*LW.MAC*" zuerst lädt. Benutzer von *SpartaDOS X* können das Start-Makro von der Eingabezeile aus umgehen oder ein anderes Makro in „LW.MAC" starten. Verwendest Du ein anderes DOS kannst Du die Ausführung von Autoexec-Makros (&) und start-Makros (@) unterbinden indem Du während des Ladeprozess **<OPTION>** drückst.
Siehe auch Kapitel 7.2.1 Selbststartende Makros.

7.2 MACROS STARTEN

Makros werden auf einen der folgenden drei Wege gestartet:

- Drücke **<CTRL+Q>** gefolgt von der dem Makro zugeordneten Taste.
- Halte **<OPTION>** und drücke die dem Makro zugeordneten Taste.
- Drücke die Tastenkombination die dem Makro zugeordnet ist.

Kommandoeingaben (die Tastatureingaben welche normalerweise Editorkommandos wie Laden/Speichern aufrufen wenn ihnen kein **<SHIFT+ESC>** vorausgeht) werden jetzt *zuerst* mit den Makrodefinitionen abgeglichen *bevor* sie mit den internen Kommandos abgeglichen werden. Das bedeutet, dass Du jetzt ein Makro schreiben kannst welches ein eingebautes Kommando vollständig überlagert. Du kannst z.B. ein Makro mit der Tastenkombination **<CTRL+L>** verknüpfen; dieses Makro überlagert dann das interne „Lade"-Kommando. Beachte, dass das Lade-Kommando verloren ist bis Du einen anderen Weg schaffst es aufzurufen. Der einzige Weg das Makro, welches ein internes Kommando überlagert, zu umgehen und das interne Kommando wieder über die Tastatur zu erreichen ist das Festhalten der **<START>**-Taste während Du das Kommando eingibst.

Die grundlegenden Regeln bei der Ausführung von Makros sind folgende:

- Drücke **<CTRL+Q>,** dann die **<Taste>** um ein Makro zu starten welches einem nicht inversem alphanumerischem Zeichen zugeordnet ist (das sind alle Zeichen die keinem Kommando zugeordnet sind).
- Drücke **<OPTION+Taste>** um Makros zu starten die mit einem beliebigen Zeichen (invers oder normal) verknüpft sind und kein Kommando sind
- Drücke **<Taste>** alleine um ein Makro zu starten welches mit einem von *LW´*s Kommandotasten verknüpft ist (und dieses überlagert).

<START> hat zwei Funktionen:

<Start> Wenn ein <#>-Makro angelegt wurde wird dieses gestartet. Das ist ein weitere Anleihe an *TextPro*. In *LW3.2* wurde die Funktionalität der **<START>**-Taste erweitert.
Hältst Du **<Start>** während eine andere Tastenkombination gedrückt wird *umgehst* Du die Makros. Auf diese Weise erreichst Du ein *LW*-Kommando welches von einem Makro überlagert wird.

Mit „*EXAMPLE.MAC*" findest Du auf der LW-Diskette eine Vorlage die Du nach Deinen eigenen Bedürfnissen anpassen kannst.

7.2.1 SELBSTSTARTENDE MAKROS

LW kennt zwei Arten von selbststartenden Makros: Das **Startup**-Makro und das **Autoexec**-Makro. Das **Startup**-Makro ist mit dem „**@**" verknüpft und wird nur beim ersten Start von *LW* geladen. Deshalb muss das „@" in „*LW.MAC*" definiert werden.

Die zweite Art von selbststartenden Makros ist das **Autoexec**-Makro, welches mit „**&**" verknüpft wird. Dieses Makro wird gestartet wenn eine Makrodatei während des Editierens mit **<SHIFT+CTRL+M>** geladen wird. Beachte dass das Autoexec-Makro in „*LW.MAC*" beim Laden von *LW* nicht gestartet wird; anstelle dessen wird das Start-Makro geladen; wird „*LW.MAC*" später während des Editieren *erneut* geladen, wird anstelle des Start-Makro „@" das Autoexec-Makro „&" geladen.

Du kannst das Start-Makro (@) auf verschiedene Weisen umgehen:

- Gebe in der *SpartaDOS X*-Eingabezeile die Option "/M" ein
- Halte **<OPTION>** beim Ladevorgang gedrückt

Du kannst das Autoexec-Makro (&) auf folgende Weisen umgehen:

- Halte **<OPTION>** beim Laden des Makros mit **<SHIFT+CTRL+M>** gedrückt (drücke & halte **<OPTION>** nach dem Druck auf <RETURN> bis die Makrodatei geladen ist).
- Wenn Du das Makro mit **<SHIFT+CTRL+M>** lädst gib unmittelbar nach dem Dateinamen des Makros die Option ",N" an.

7.3 MAKROS SCHREIBEN UND EDITIEREN

Da Makros die Angewohnheit haben eine Menge Steuerzeichen zu enthalten ist es vorteilhaft einen speziellen Makro-Zeichensatz zu laden bevor man mit dem Editieren beginnt. Er ersetzt spezielle alphanumerische Zeichen der Steuerzeichen und macht so Makros besser lesbar. Lade den Makrozeichensatz mit...

<SHIFT+CTRL+N>, dann gebe „MACRO" ein und drücke <RETURN>

Das funktioniert sowohl im 80- als auch im 40-Zeichen-Modus da es für beide Auflösung einen passenden Makro-Zeichensatz gibt: „MACRO.F80" für den 80-Zeichen-Modus und „MACRO.FNT" für den 40-Zeichen-Modus. Du wirst beim Editieren eines Makros wahrscheinlich den 40-Zeichen-Modus vorziehen weil er einfacher zu lesen ist und Genauigkeit beim Schreiben eines Makros sehr wichtig.

Wechsle zum 40-Zeichen-Modus mit...

<SHFT+CTRL+W>

Falls noch nicht geschehen, lade den 40-Zeichen-Makro-Font.

Alle hier mit Bildschirmfoto gezeigten Beispiele sind im 40-Zeichen-Modus und dem „MACRO.FNT"-Zeichensatz erstellt worden.

7.4 SPEZIELLE MAKRO-KOMMANDOS

Wie bei *TextPro* werden die Makro-Kommandos in INVERSEN **<CTRL+KEY>**-Zeichen eingegeben und sind *nur* innerhalb eines Makros verfügbar. Gib diese mit **<CTRL+ESC>**, entweder gefolgt von **<INVERSE>**, **<CTRL+KEY>**, dann wieder **<INVERSE>**, oder **<SELECT+CTRL+KEY>** ein, ein weiteres *TextPro*-Feature welches übernommen wurde um die Eingabe der inverse Zeichen weniger mühevoll zu machen.

<SELECT+CTRL+A> Frage nach einer Eingabe. Das Kommando erwartet vom Benutzer eine String-Eingabe über die Informationszeile. Dem Kommando folgt ein mit <RETURN> zu beendender Text, der in der Informationszeile angezeigt wird. Ein folgender Text wird als Default in das Eingabefeld gesetzt. Um letztendlich die Eingabe vom Benutzer zu erlangen mußt Du das **<CTRL+L>ine Input**-Kommando verwenden. Das Makro pausiert und erlaubt eine Texteingabe die mit <RETURN> beendet wird. Die Eingabe landet im Einfüge-Puffer und überschreibt dessen Inhalt, auch wenn keine Eingabe vorgenommen wurde. Steht dem **<CTRL+A>**-Kommando ein **<CTRL+B>ranch**

Macro voran so wird der Sprung bei einem leeren Eingabestring ausgeführt.

Der Einfüge-String kann, wie jede Einfügeoperation, mit der Paste-Funktion in ein Dokument eingefügt werden. Er kann auch mit dem **<CTRL+V>**-Kommando in eine Eingabezeile bei Dateiname/suche/ ersetze eingefügt werden (siehe unten).

Die Frage nach einer Eingabe mit **<SELECT+CTRL+A>** bietet praktisch unbegrenzter Spielraum bei der Entwicklung von Makros und erlaubt die Erstellung von professionell wirkenden interaktiven Applikationen. Bedenke dass bei der Benutzung dieses Kommandos der Inhalt des Einfüge-Puffers gelöscht wird.

<SELECT+CTRL+B> Springe (**b**ranch) zu Makro. Nutze dieses Makro um Sprünge zu realisieren. Es folgt die Makrokennung und eines der folgenden Makro-Kommandos. Der Sprung wird abhängig von den folgenden Bedingungen ausgeführt:

Zeichen	Kommando	Sprung...
<SHIFT+CTRL+M>	Makro laden	zum Zielmakro in der zu ladenden Makrodatei
<CTRL+F>	Finde Sting	wenn der String NICHT gefunden wird
<SHIFT+CTRL+G>	Gehe zu Lesezeichen	wenn das Lesezeichen NICHT gefunden wird
<Select+CTRL+C>	Makro bestätigen	wenn der User „N“ eingibt
<Select+CTRL+A>	Makro “Frage”	wenn der User einen leeren String eingibt
<Select+CTRL+Z><n>	Wähle Bank	wenn die Textbank nicht gefuden wird
<CTRL+L>	Text laden	Bei „verketteten Laden“

Siehe <CTRL+Z>-Kommando für weitere bedingte Sprünge.

<SELECT+CTRL+C> (**C**onfirm) mit Y/N bestätigen. Gefolgt von einem mit <RETURN> beendeten Text. *LW* gibt den Text, gefolgt von einem Fragezeichen und „(Y/N):“ aus. Der Benutzer antwortet mit der entsprechenden Taste. „Y“ lässt das Makro weiterlaufen. „N“ beendet alle Makros (selbst wenn das laufende Makro verschachtelt ist), oder, wenn einem **<CTRL+B>**-Kommando ein Sprungmakro anhängt, wird dieses Makro gestartet. **BEACHTE**: **<SELECT+CTRL+A>sk**, das vergleichbare „Y/N“-Kommando von *TextPro* vor der Einführung von „Makro-Bedingungen“ in späteren *TextPro*-Versionen, versucht, wenn „N“ gedrückt wurde, immer das „&“-Makro zu starten. *LW* unterstützt dieses Merkmal *nicht*: nutze **<CTRL+B>** um ein Makro vorzuwählen welches gestartet wird wenn „N“ gedrückt wird.

<SELECT+CTRL+J> Makro Menü. Gefolgt von einer Zeile Text, beendet mit <RETURN>. Der Text sollte die Form eines kleines Menüs haben. Die Nachricht wird ausgegeben, dann wird das Makro gestartet welches mit dem nächsten Tastendruck verknüpft ist.

Das Makromenü wurde seit Version 2.1 von *LW* etwas erweitert. Laut Definition ist das Makro-Menü-Kommando das letzte Kommando in einem Makro, jetzt ist es möglich zwei zusätzliche Parameter nach

dem Menütext anzugeben. Gib in der nächsten Zeile alle gültigen Tasten für die Menüoptionen an, beendet durch <RETURN>. Achte darauf das sie in der selben Reihenfolge stehen wie die zugehörigen Menüpunkte. In der nun folgenden Zeile gibst Du die ID-Zeichen der Makros an die mit den Tasten verknüpft werden sollen. Achte auch hier auf die richtige Reihenfolge. Auch diese Zeile wird mit <RETURN> beendet.

Hier ein Beispiel:

#=<INV CTRL+J>**Laden, Speichern, Drucken**<RETURN>
LSD<RETURN>
LSD<RETURN>

Dieses Makro wird, wenn **<START>** gedrückt wird, folgendes anzeigen:

Laden, Speichern, Drucken

...und auf einen Tastendruck warten. Ein Druck auf <L> wird das mit dem inversen „L“ verknüpfte Makro starten, <S> das mit dem inversen „S“ verknüpfte Makro und <D> wird versuchen das inverse „D“-Makro zu starten. Auf diese Weise kannst Du bereits anderweitig vergebene Tasten freihalten, die Makros aber trotzdem durch einfache Tastendrücke innerhalb eines Menüs aufrufen.

BEACHTE: Es ist möglich die zwei Extrazeilen mit Informationen wegzulassen. Das Menü startet dann einfach das Makro welches mit der gedrückten Taste verknüpft ist.

<SELECT+CTRL+K> Hole Taste. Wartet auf einen Tastendruck und führt dann das Makro weiter aus.

<SELECT+CTRL+L> Hole Zeile. Sowohl im Editor als auch in einem Eingabedialog wird dieses Zeichen das Makro anhalten und eine mit <RETURN> zu beendete Eingabe ermöglichen. Das Makro kann mit **<BREAK>** gestoppt werden. **Beachte**, dass viele Features des Editors, einschließlich der Befehlszeile, während der „hole Zeile“-Funktion nicht verfügbar sind. Wenn Du ein anders Makro startest während Du im „hole Zeile“-Modus bist wird das aktuelle Makro verlassen und der „hole Zeile“-Modus wird beendet. **BEACHTE**: Der Eingabemodus von *TextPro* arbeitet *immer* im ÜBERSCHREIBE-Modus - *LW* Eingabezeile arbeitet IN DEM MODUS IN DEM SICH DER EDITOR GERADE BEFINDET. Auch das Doppelpunkt-Trennzeichen von *TextPro* wird von *LW* NICHT unterstützt.

Das hole-Zeile-Kommando filtert nicht länger, wie in Version 1.0, alle Cursorbewegungen ausser links/rechts heraus. Nur einige wenige Kommandos - hauptsächlich die die Eingaben erfordern - sind jetzt im Makro-Eingabemodus deaktiviert. Diese Änderung wurde implementiert um den Wirkungsbereich interaktiver Makros zu vergrößern. Ein Makro kann jetzt zum Beispiel pausieren während Du einen Textblock markierst und, wenn Du <RETURN> drückst, den Textblock bearbeiten.

<SELECT+CTRL+V> Gebe Nachricht aus. Gefolgt von Text, beendet mit <RETURN>. Der Text wird ausgeben und beim nächsten Tastendruck wieder gelöscht.

<SELECT+CTRL+X> Führe Makro aus. Gefolgt von der Makro-ID. *LW* wird versuchen das Makro wie eine Unterroutine oder Prozedur auszuführen. Das Bedeutet dass, wenn das Makro beendet ist, das aufrufende oder übergeordnete Makro mit dem nächsten Kommando fortfährt. Makroaufrufe können bis zu einer Tiefe von 128 verschachtelt werden.

<SELECT+CTRL+Z> Setzt Schalter und Flags. Gefolgt von einem der folgenden Zeichen:

U	Aktiviert die Grossschrift	(**u**pper case)
L	Aktiviert die Kleinschrift	(**l**ower case)
I	Aktiviert den Einfügemodus	(**i**nsert)
O	Aktiviert den Überschreibmodus	(**o**verwrite)
R	Aktiviert den inversen Modus	(**r**everse)
N	Aktiviert den normalen Modus	(**n**ormal)

1-9 oder 0 (0=10) Wählt die entsprechende Textbank wenn mehrere Bänke eingerichtet sind. Bank 1 ist *immer* die Haupt-Bank (im normalen Speicher), und 2-10 entsprechen den Bänken im erweiterten Speicher.

B Wählt die Textbank in der sich das Programm beim Aufruf des Makros befand.

H Schaltet Bildschirmupdates ab.

U Schaltet Bildschirmupdates wieder an.

X Löscht das „Text wurde editiert“-Flag(„*“).

Diese Parameter werden in normalen Video (nicht invers) eingegeben und benötigen jeweils ein eigenes **<SELECT+CTRL+Z>**.

Genau wie das **<SELECT+CTRL+Z>**-Kommando ein Flag setzt oder löscht kann es auch bestimmte Bedingungen testen:

M Überprüft ob ein Block markiert wurde.

S Überprüft ob bereits ein Text gewählt wurde.

C Überprüft ob der Text im Speicher seit dem letzten Speichern editiert wurde.

A Überprüft ob die Datei bereits gespeichert wurde.

Du kannst diesen Tests ein Sprung-Kommando voranstellen. Der Sprung wird ausgeführt wenn die Bedingung FALSCH ist.

7.4.1 DEN BILDSCHIRM VON EINEM MAKRO AUS ABSCHALTEN

Du kannst das...

<SELECT+CTRL+Z> „setze Schalter“-Kommando

nutzen um den Bildschirm von einem Makro aus an- oder abzuschalten. Mit „H“ (**h**ide) werden alle Änderungen auf dem Bildschirm unsichtbar, mit „V“ (**v**isible) werden sie sichtbar. Wenn der Bildschirm deaktiviert ist wird nichts mehr auf ihm ausgegeben - die Änderungen werden erst sichtbar wenn der Bildschirm wieder aktiviert wird. Die Unterdrückung der Anzeigen die normalerweise über die Eingabezeile flitzen ermöglicht es Makros so *aussehen* zu lassen als wenn sie eingebaute Kommandos wären. Bestimmte Kommandos - solche wie print, view, spool, disk menu und die vier Makro-Kommandos welche Informationen zu einer Eingabeaufforderung anzeigen - aktivieren die Anzeige wieder. Die Anzeige wird nach Beendigung des Makros automatisch wieder aktiviert falls dies nicht schon vorher geschehen ist.

7.4.2 SPEZIELLE ZEICHEN

<CTRL+P> setzt in einer Eingabeaufforderung für Dateinamen das Laufwerk, den Pfad und den Namen der aktuellen Datei ein.

<CTRL+N> setzt in einer Eingabeaufforderung für Dateinamen den Namen der aktuellen Datei *ohne* Laufwerkskennung ein.

<CTRL+V> fügt (ausser wenn ein <CTRL+ESC> vorangeht) in *jedem* Eingabedialog soviel vom Inhalt des Einfüge-Puffers in die Eingabezeile ein wie passt. Mit Hilfe dieser Methode kannst Du einen vorher mit dem **<SELECT+CTRL +A>** „**A**sk for input"-Kommando eingelesenen Text in jedes *LW*-Kommando, welches eine Eingabe verlangt, übertragen. Auf die gleiche Weise kannst Du einen aus einem Dokument ausgeschnittenen Text in ein *LW*-Kommando einfügen. **Beachte** dass der Text in Großbuchstaben eingesetzt wird wenn der Eingabedialog mit einer Dateioperation verbunden ist.

BEACHTE: Das Einfügen des Puffers mit **<CTRL+B>** ist in dieser Version von *LW* nicht mehr verfügabr; benutze stattdessen **<CTRL+V>**.

<CTRL+L> fügt den Namen der zuletzt in eine Bank geladenen Datei ein (siehe Kapitel 3.8 „Umgang mit grossen Dateien").

Um diese neuen Kommandos so flexibel wie möglich zu gestalten sind die Device/Pfad/ Name und Pfad/Name-Variablen jetzt von *jedem* Eingabedialog aus erreichbar. Stelle ein **<CTRL+ESC>** voran um sie buchstabenweise einzugeben.

7.4.3 KOMMANDOS VON EINEM MAKRO AUS EINGEBEN

Während im Editor ein Makro mit **<CTRL+Q>** gestartet wird geschieht dies innerhalb eines Makros mit **<CTRL+X>**. Das bedeutet dass *in* einem Makro <ESC> **wieder seine normale Aufgabe als vorzustellendes Zeichen bei Steuerzeichen hat**. Möchtest Du innerhalb eines Makros ein Kommando als Teil Deines Textes anstatt als Kommando eingeben, stelle innerhalb eines Makros einfach ein <ESC>-Zeichen voran. Viele *LW*-Kommandos sind mit einer <SHIFT+CTRL>-Tastenkombination verknüpft. Für diese Kombinationen gibt es kein ASCII-Gegenstück, wie können diese Kommandos also in einem Makro bezeichnet werden? Ganz einfach - in einem Makro nutzt Du **<INVERSE CTRL>** statt <SHIFT+CTRL>. Um also das <SHIFT+CTRL+F>inde String-Kommando innerhalb eines Makros einzugeben gebe stattdessen <INVERSE CTRL+F> oder <SELECT+CTRL+F> ein. Deshalb nutzen die speziellen Makro-Kommandos nur Zeichen die sich auf ungültige <CTRL+SHIFT> Tastatureingaben beziehen.

Wenn ein Makro läuft sind die einzigen Eingaben die akzeptiert werden das „Hole Taste"-Kommando **<INV+CTRL+K>**, Bestätigungen **<INV+CTRL+C>**, Texteingaben im „Hohle Zeile"-Modus **<INV+CTR+L>** und Tasten die während aktiver Seitenpause gedrückt werden. Die „Press a key"-Aufforderung nach einer Dateiansicht/Drucke-Operation zum Löschen und Rückkehr zum Editor verlangt nach einem Tastendruck vom aktiven Makro.

7.4.4 DER MAKRO-ZEICHENSATZ

Der Zeichensatz „MACRO.FNT" bzw. „MACRO.F80" auf der Diskette kann wie folgt geladen werden:

<SHIFT+CTRL+N> „**N**euer Zeichensatz", gebe „MACRO" <RETURN> ein. Das funktioniert sowohl im 80- als auch im 40-Zeichen-Modus. Diese Zeichensätze zeigen alle

Steuerzeichen als spezielle, festgedruckte Zeichen anstelle der internationalen Zeichen an um das Editieren der Makros einfacher zu gestalten.

7.4.5 TASTATURVEREINBARUNGEN FÜR MAKROS

Zu Beginn mag es schwierig sein zu verstehen wie die Tasten in *LW* arbeiten, also lasst uns, bevor wir einige Beispiele durchgehen, kurz zusammenfassen:

- **<CTRL+ESC>** oder **<SHIFT+ESC>** erlaubt es Steuerzeichen im Editor oder einer Eingabeaufforderung einzugeben so wie es <ESC> normalerweise im BASIC tut. Um das Escape-Zeichen *an sich* im Text zu erhalten (welches in *LW* als gebogener, abwärts gerichteter Pfeil erscheint) drücke <CTRL+ESC> oder <SHIFT+ESC> zweimal.

  BEACHTE: In Version 3.2 kannst Du im Editor das Escape-Zeichen einfügen indem Du zweimal hintereinander <ESC> drückst.

- Mit **<SHIFT+CTRL+ESC>** fügst Du ebenfalls das Escape-Zeichen direkt in den Text ein.

- **<CTRL/SHIFT+ESC>** während ein Text markiert ist wird die Markierung aufheben, wie auch <BREAK>, <ESC> oder irgendein Text den Du eingibst.

- Das <ESC>-Zeichen in einem Makro kopiert ein auf der Tastatur eingegebenes <CTRL/SHIFT+ESC>. Um das <ESC>-Symbol von einem Makro aus in den Editor zu bekommen füge zwei aufeinanderfolgende <ESC>-Zeichen in das Makro ein.

- **<CTRL+Q>** im Editor gedrückt startet Makros.

- In einem Makro werden Makrokommandos und <SHIFT+CTRL>-Kommandos in inversen <CTRL+KEY>-Kommandos eingegeben.

Der beste Weg Makros zu verstehen ist es ein paar Beispiele durchzugehen. Es wird Dir auch helfen *LW´s* Makrosprache zu verstehen wenn Du die auf der Diskette mitgelieferten Makros untersuchst.

7.5 MAKROS ERSTELLEN & BEARBEITEN

Makros tendieren dazu eine Menge Steuerzeichen zu enthalten, deswegen ist es am besten vor Beginn der Arbeit den speziellen Makro-Zeichensatz zu laden. Er ersetzt die Steuerzeichen durch spezielle alphabetische Zeichen und macht das Makro besser lesbar. Lade den Makro-Zeichensatz wie folgt:

<SHIFT+CTRL+N> dann gebe „MACRO“ ein und drücke <RETURN>

Das funktioniert sowohl im 80- als auch im 40-Zeichen-Modus weil es für jeden Modus einen extra Zeichensatz gibt: „MACRO.F80“ für den 80-Zeichen-Modus und „MACRO.FNT“ für den 40-Zeichen-Modus. Du wirst beim Bearbeiten von Makros wahrscheinlich den 40-Zeichen-Modus vorziehen, weil der 40-Zeichen-Zeichensatz besser zu lesen ist, und Genauigkeit ist beim Erstellen von Makros sehr wichtig.

Wechsle zum 40-Zeichen-Modus mit:

<SHIFT+CTRL+W>

Wenn noch nicht geschehen musst Du den 40-Zeichen-Makro-Zeichensatz noch laden.

Alle Bildschirmfotos der gezeigten Beispiele sind im 40-Zeichen-Modus mit dem Zeichensatz „MACRO.FNT“ erstellt worden.

7.6 BEISPIEL-MAKROS

Der beste Weg die Entstehung eines Makros zu veranschaulichen sind ein paar hilfreiche Beispiele.

ZWEI-ZEICHENSATZ-LADER

Lädst Du einen Zeichensatz (ZS) in *LW* wird normaler Weise nur der ZS geladen der zu der gerade gewählten Auflösung passt. Befindet sich *LW* z.B. gerade im 80-Zeichen-Modus und Du lädst einen ZS wird nur der 80-Zeichen-ZS (.F80) geladen, während der 40-Zeichen-ZS unverändert bleibt. Du kannst den anderen ZS durch die Angabe des entsprechenden Extenders laden. Wenn sich *LW* z.B. im 40-Zeichen-Modus befindet kannst Du den „MACRO.F80“-ZS laden, während der aktuelle 40-Zeichen-ZS unverändert bleibt.

Was aber wenn Du den 40- und den 80-Zeichen-ZS *gleichzeitig* laden möchtest? Wir können ein Makro schreiben welches als neues Kommando *beide* ZS lädt.

Wir wollen das Makro mit <OPTION+A> starten, lege also eine leere Datei an und gebe „a“ ein. Danach **<SHIFT+ESC>** und **<SELECT+=>** um das Zuweisungszeichen zu erhalten. Jetzt können wir die Kommandos, die das Makro ausmachen, eingeben.

Als erstes müssen wir den Namen des ZS, den wir laden wollen, erhalten. Dazu nutzen wir das „Frage nach“-Kommando. Also geben wir **<SHIFT+ESC>**, **<SELECT+CTRL+A>** ein um das Zeichen für das „Frage nach“-Makro einzufügen. Es folgt der Text der auf dem Schirm zu sehen sein soll. Gebe...

„**Zeichensatz“** ein.

Drücke **<RETURN>** um den Text zu beenden. Als nächstes müssen wir das Makro anhalten und dem Benutzer die Möglichkeit geben den Namen des Zeichensatzes einzugeben und **<RETURN>** zu drücken. Dies geschieht mit dem Eingabe-Kommando (hole Zeile), also gebe ein:

<SHIFT+ESC>, **<SELECT+CTRL+L>**

Gebe anschließend **<RETURN>** ein - dies ist das <RETURN> welches die Eingabe des Benutzers beendet. Das <RETURN> mit dem der Benutzer *seine* Eingabe beendet ist *nicht* Bestandteil der Eingabe an sich; es teilt dem Makro lediglich mit dass der Benutzer *seine* Eingabe beendet hat. Deswegen müssen wir das <RETURN>-Zeichen ausdrücklich angeben um eine Texteingabe zu beenden.

Der String, der mit dem „Frage nach einer Eingabe“-Kommando vom Makro erfasst wurde, wird im Einfügepuffer abgelegt. Er wird dort gespeichert, weil der Text so leicht in ein Dokument eingefügt werden kann. Der Inhalt des Einfügepuffers kann auch mit Hilfe des **<CTRL+V>**-Kommandos in die Eingabezeile eines anderen Kommandos geschrieben werden. Und das tun wir jetzt: der vom Benutzer vorgegebene Dateiname wird in die „Lade Zeichensatz“-Eingabezeile geschrieben.

Die nächsten Zeichen die wir also in das Makro einfügen sind **<SHIFT+ESC>**, **<SELECT +CTRL+N>**. Dann gebe **<SHIFT+ESC>**, **<CTRL+V>** ein, gefolgt von „**.FNT**“, und schließlich **<RETURN>** ein. Das hängt einfach „.FNT“ an den vom Benutzer vorgegebenen Dateinamen an; wir wollen den 40-Zeichen-ZS laden. Um den entsprechenden 80-Zeichen-ZS zu laden

geben wir die selbe Zeile noch einmal ein, abgesehen davon dass wir diesmal „.F80“ anhängen.

Das vollständigen Makro

Speichere das Makro unter dem Namen „FONTSET.MAC” ab und lade es anschließend mit **<SHIFT+CTRL+M>**. Wenn Du jetzt **<OPTION+A>** drückst und „MACRO“ in der Eingabeaufforderung eingibst wird *LW* sowohl den 40- als auch den 80-Zeichen-ZS laden. Wenn einer der beiden ZS nicht auffindbar ist wird das Makro einfach beendet.

Wir können das Makro verfeinern indem wir Änderungen des Bildschirmes unterdrücken. Dazu fügen wir das **<SELECT+CTRL+Z>** Kommando in das Makro ein, gefolgt von einem „**H**“, unmittelbar vor dem ersten **<SELECT+CTRL+F>** „Lade ZS“-Kommando, wodurch wir an dieser Stelle die Bildschirmupdates abschalten. Wir sollten an das Ende des Makros **<SELECT+CTRL+Z>** gefolgt von einem „**V**“ setzen um die Updates wieder einzuschalten, obwohl es in diesem Fall nicht *wirklich* nötig ist weil die Updates automatisch aktiviert werden sobald ein Makro beendet oder vorzeitig durch einen Fehler, **<ESC>** oder **<BREAK>** beendet wird.

Das überarbeitete Makro sieht wie folgt aus:

Dieses Makro ist bei der Ausführung nicht mehr von einem eingebauten Kommando unterscheidbar.

ZEICHEN TAUSCHEN

In *LW* gibt es kein Kommando um verdrehte Zeichen zu tauschen, aber wir können dieses Kommando mit Hilfe eines Makros erstellen.

Beachte: dieses Makro befindet sich, zusammen mit Makros die Wörter & Absätze tauschen, auf der *LW.MAC*-Datei auf der *LW*-Diskette.

Wir werden also zuerst einmal ein „tausche benachbarte Zeichen“-Makro schreiben.

Wir legen dieses Makro auf **<ESCAPE> <CTRL+T>**auschen. Um die Eingabe des Makros zu vereinfachen gebe zuerst **<CTRL+CAPS>** ein um in den Steuerzeichen-Modus zu gelangen. Dadurch werden *LW's* Kommandos abgeschaltet und Du kannst Steuerzeichen direkt eingeben ohne vorher ständig **<CTRL+ESC>** drücken zu müssen. Wenn Du Korrekturen machen musst schalte den Steuerzeichen-Modus mit **<CTRL+CAPS>** wieder aus.

Drücke in einem leeren Editorfenster **<CTRL+T>**.

Gebe dann **<SELECT+=>** ein.

Drücke **<CTRL+M>**. Das bedeutet das wir dabei sind einen Block zu markieren.

Gebe jetzt **<CTRL+→>** ein. Wenn das Makro läuft markiert dies das Zeichen unter dem Cursor als Block. Gebe anschließend **<CTRL+C>** ein. Das ist das „Schneide markierten Text aus"-Kommando und übergibt das Zeichen an den Einfügepuffer.

Gebe jetzt noch einmal **<CTRL+→>** ein um den Cursor hinter das nächste Zeichen zu setzen. Gebe jetzt **<CTRL+P>** ein. Damit wird das ausgeschnittene Zeichen rechts von dem folgenden Zeichen eingesetzt. Gebe zum Schluss **<CTRL+←>** ein um den Cursor wieder an seine ursprüngliche Position zu setzen.

Wir beenden das Makro mit einem netten Effekt:

Gebe **<SELECT+CTRL+V>** (das „drucke Nachricht"-Kommando) ein, gefolgt von dem Text „Zeichen getauscht" und beende mit **<RETURN>**.

Jetzt musst Du mit **<CTRL+CAPS>** den Steuerzeichenmodus verlassen. Speichere und lade das Makro wie oben beschrieben. Wenn Du nun **<ESC>** gefolgt von **<CTRL+T>** eingibst wird das Zeichen unter dem Cursor mit dem Zeichen recht getauscht und eine Nachricht über die Aktion wird ausgegeben.

7.7 ZUSAMMENFASSUNG

Wie Du siehst sind die Möglichkeiten der Makrosprache nur durch Deine Vorstellungskraft begrenzt. Wenn Du an etwas im Kopf hast was *LW* nicht kann hast Du die Möglichkeit es durch ein Makro zu realisieren. *LW* ist schnell genug um Deine Makros genau so schnell auszuführen als ob sie ein eingebautes Kommando wären. Du kannst z.B. Deine Adresse oder andere, oft benötigte Textpassagen mit einem Makro verknüpfen, oder einen Text von Disk an die aktuelle Cursorposition einfügen lassen. Schau Dir die auf der *LW*-Diskette mitgelieferten Makrodateien an um eine Vorstellung von der Vielfalt der Anwendungs-möglichkeiten von Makros zu bekommen.

8 LW EINRICHTEN

Du kannst *LW* so einrichten das es immer mit Deinen bevorzugten Einstellungen geladen wird. Selbst *während* der Arbeit im Editor kannst Du andere Konfigurationen laden. Du kannst alles festlegen, von der Farbe des Bildschirmes bis hin zu den zusätzlichen Speicherbänken für den Text.

LW unterstützt zwei Arten von Konfigurationsdateien: *LW.SYS* wird beim ersten Start geladen und enthält Informationen über Speicher, Tastaturpuffer und neudefinierte Tasten. *LW.SYS* wird *einmal* beim Programmstart geladen und die darin enthaltenen Einstellungen können *nicht* mehr verändert werden wenn das Programm einmal geladen ist.

8.1 EINRICHTUNGSMÖGLICHKEITEN IM EDITOR

Die folgenden Kommandos ändern oder richten verschiedene Features von *LW* ein. Diese Einstellungen werden alle in der Konfigurationsdatei *LW.CFG* gespeichert. Alle, mit Ausnahme der Tabulatorleiste, können mit dem Konfigurationsprogramm eingestellt werden. Nicht alle Optionen des Konfigurationsprogramms können beim Schreiben im Editor verändert werden: es sind nur die Optionen verfügbar bei denen es wahrscheinlich ist dass sie während des Programmablaufs geändert werden müssen.

<CTRL+W>	Schaltet den Wortumbruch ein/aus
<SHIFT+CTRL+W>	Setzt die Auflösung (40/80 Zeichen) und Spaltenzahl (5-240)
<CTRL+TAB>	Löscht den Tabulatorstop an der aktuellen Spalte
<SHIFT+TAB>	Setzt einen Tabulatorstop an der aktuellen Spalte
<SHIFT+CTRL+E>	Löscht alle Tabulatorstops
<SHIFT+CTRL+TAB>	Stellt die Grundeinstellung der Tabulatorstops her
<SHIFT+CTRL+INS>	Wechselt zwischen Einfüge/Überschreibe-Modus
<CAPS>	Wechselt zwischen Groß- und Kleinschrift
<SHIFT+CTRL+U>	Benutzerfunktionen ändern

Einige Einstellungen des Diskettenmenüs werden ebenfalls in der Konfigurationsdatei gespeichert:

<S>pec	Legt die Maske für Dateinamen fest
<1-9, 0>	Legt die aktuelle Laufwerksnummer fest
<Tab>	Darstellungsweise des Inhaltsverzeichnisses

Mit den folgende Kommandos kannst Du verschiedene Konfigurationen während des editieren zu laden oder speichern:

<CTRL+O>	Konfiguration laden
<SHIFT+CTRL+O>	Konfiguration speichern

Wenn Du keinen Extender angibst wird automatisch „.CFG“, sowohl beim Laden als auch beim Speichern, an den Dateinamen angehängt.

Ein weiteres Kommando ist

<SHIFT+CTRL+N>	Installiere/Lade einen alternativen Zeichensatz

Die Informationen zum Zeichensatz (ZS) werden *nicht* im Konfigurationsfile gesichert. Um den gewünschten ZS beim Start zu laden nenne ihn in „LW.FNT“ um und speichere ihn auf der *LW*-Diskette.

8.2 .CFG KONFIGURATIONSDATEIN

CFG-Dateien enthalten Benutzereinstellungen wie Bildschirmfarbe, Bildschirmauflösung und Weite (die Anzahl der Spalten pro Zeile), Laufwerksnummer, Angaben zur Datei u.s.w.. Viele dieser Einstellungen können vom Editor aus geändert werden und jederzeit als CFG-Datei mit **<SHIFT+CTRL+O>** gespeichert werden.

Wenn sich während des Programmstarts eine Datei mit dem Namen "*LW.CFG*" auf dem in der Grundeinstellung vermerkten Laufwerk befindet wird diese Konfigurationsdatei beim Programmstart geladen. Auf diese Weise kannst Du Deine am häufigsten genutzten Einstellungen in *LW.CFG* speichern und während einer Editorsitzung mit **<CTRL+O>** eine andere Konfigurationen laden.

Eine CFG-Datei besteht aus einfachen Text und mit <RETURN> abgeschlossenen Zeilen. Jede Zeile besteht aus einem Schlüsselwort, gefolgt von einem Leerzeichen, dann ein oder mehrere numerischen oder Text-Argumente, getrennt durch ein Kommata oder Leerzeichen.

Hier ist eine vollständige Liste mit allen Schlüsselwörtern:

Zahlenwert	Argument	Bemerkung	Vorgabe
TMARGIN	0-255	oberer Rand (**t**op **margin**)	5
BMARGIN	0-255	unterer Rand (**b**ottom **margin**)	61
LMARGIN	0-255	linker Rand (**l**eft **margin**)	10
RMARGIN	0-255	rechter Rand (**r**ight **margin**)	70
PAGELEN	0-255	Seitenlänge (**page len**gth)	66
HFLEFTMARG	0-255	linker Rand Kopf/Fußzeile (**h**eader/**f**ooter **left marg**in)	10
HFRIGHTMARG	0-255	rechter Rand Kopf/Fußzeile (**h**eader/**f**ooter **right marg**in)	70
SPACING	0-255	Zeilenabstand (line **spacing**)	1
HEADOFF	0-255	Offset Kopfzeile (**head**er **off**set)	2
FOOTOFF	0-255	Offsett Fußzeile (**foot**er **off**set)	2
EOLCHAR	0 oder 219	Zeichen für Zeilenende (**e**nd-**o**f-**l**ine **char**acter (Interner Code)	219
PADCHAR	0 oder 125	Zeichen für Füllzeichen (false space character) (Interner Code)	0
TEXTCOL	0-255	Textfarbe (**text col**or)	10
SCREENCOL	0-255	Farbe des Textfensters (**screen col**our)	148
PROMPTCOL	0-255	Farbe Eingabezeile (**prompt** line **col**our)	10
BORDERCOL	0-255	Rahmenfarbe (**border col**our)	0
BARCOL	0-255	Farbe des Fortschrittsbalken (Progress **bar col**our)	184

Zahlenwert	Argument	Bemerkung	Vorgabe
KEYDELAY	0-255	Tastenverzögerung (initial **key delay**)	30
KEYREPEAT	0-255	Tastenwiederholrate (**key repeat** rate)	3
TABWIDTH	0-255	Schrittweite Tabulator (**tab**ulator **width**)	5
FILESORT	0-4	Legt fest wie Dateien sortiert werden: 0 = keine Sortierung 1 = nach Namen 2 = nach Extender 3 = nach Datum/Uhrzeit 4 = nach Grösse	0
PAGEWIDTH	5-240	Spaltenbreite	

Flags (An/Aus)	Argument	Bemerkung	Vorgabe
80COLUMNS	ON ¦ OFF	80 Zeichen-Modus: an 40 Zeichen-Modus: aus	An
PAGEWAIT	ON ¦ OFF	Set page wait mode during printing	Aus
WORDWRAP	ON ¦ OFF	Spaltenumbruch an/aus	An
INSERT	ON ¦ OFF	Einfügemodus an/aus	An
CASESENS	ON ¦ OFF	Beachte Gross- und Klein-schreibung beim Suchen & Ersetzen.	Aus
CAPSLOCK	ON ¦ OFF	Grosschrift an/aus	Aus
KEYCLICK	ON ¦ OFF	Tastaturklick an/aus	An
SIONOISE	ON ¦ OFF	SIO-(Lade)geräusch an/aus	An
WILDCARDS	ON ¦ OFF	Platzhalter beim Suchen / Ersetzen	An
ATTRACT	ON ¦ OFF	Bildschirmschoner an/aus	An
DOCMODE	ON ¦ OFF	Dokument-Modus: an Text-Modus: aus	Aus
SDXDIR	ON ¦ OFF	Langes Inhaltsverzeichniss (nur bei SpartaDOS X)	Aus
Text Settings	**Argument**	**Bemerkung**	**Vorgabe**
FILEEXT	<EXT>	Extender für den Text-Modus; bitte ohne Punkt eingeben!	TXT
DRIVE	Dn:	Laufwerkskennung	D:
FILESPEC	<filemask.ext>	Maske für das Disketten-Menü	*.*

Eine gute Möglichkeit die Funktion der CFG-Dateien zu verstehen ist es die *LW.CFG*-Datei in den Editor zu laden. Ebenso gut kannst Du die aktuelle Konfiguration mit **<SHIFT+CTRL +O>** mit dem Namen „*TEST.CFG*“ auf Diskette speichern um zu sehen wie sich die Änderungen, die Du vorgenommen hast, sich auf die Konfigurationsdatei auswirken. Es hält Dich nichts davon ab eine CFG-Datei in den Editor zu laden und sie von Hand zu ändern, obgleich Du darauf achten solltest keine Syntaxfehler zu machen. Einige Einstellungen - wie die Druckränder - können nur von Hand in der CFG-Datei geändert werden da es keine Kommandos gibt um sie zu ändern.

Beachte: Bei den Vorgängerversionen von *LW* 3.0 waren die Konfigurationsdateien Binärfiles und wurden mit einem mitgelieferten Konfigurationseditor bearbeitet. Jetzt handelt es sich bei der Konfigurationsdatei um eine einfache Textdatei, und der Konfigurationseditor ist nicht länger nötig. Beachte dass die Konfigurationsdateien (.CFG) früherer Versionen nicht kompatibel mit Version 3.0 sind. Die Konfigurationsdateien von 3.0 sind aber „aufwärtzkompatibel“ mit 3.1 und 3.2.

8.2.1 DAS STANDARDLAUFWERK

Das „**DRIVE**“-Kommando in einer Konfigurationsdatei legt das Standard-Laufwerk in *LW* fest. Das ist die Laufwerkskennung die vor jedem Dateinamen gesetzt wird den Du ohne „Dn:“ angibst. Es ist ebenfalls die Laufwerksnummer wenn das Diskettenmenü aufgerufen wird.

DRIVE D1:

Das obige Beispiel legt Laufwerk 1 als Standardlaufwerk fest.

8.3 DIE LW.SYS-DATEI

Die *LW.SYS*-Datei wird einmalig beim Start von LW gelesen. Die Informationen in dieser Datei legen z.B. die Speicherkonfiguration und andere, während des Programmablaufes unveränderliche Einstellungen fest. *LW.SYS* wird als einfache Textdatei mit dem Editor erstellt. In der Regel sind die Kommandos zur Speicherkonfiguration in *LW.SYS* unwichtig, da die Einstellungen normalerweise automatisch vorgenommen werden. In manchen Fälle empfiehlt es sich aber diese Einstellungen zu ändern. In *LW.SYS* kann der *LW*-interne Tastaturpuffer abgeschaltet, das Tastaturlayout geändert oder der Pfad, in dem ein System, das nicht unter SpartaDOS X arbeitet, sucht,festgelegt werden.

LW.SYS kann die folgenden Instruktionen enthalten:

Anweisung	Argument(e)	Kommentar	Vorgabe
BANKED	ON¦OFF	(De)aktiviert BANKED Memory	AN (bei unterstützten DOSen)
BANKS	n,n,n,n…	Gibt die zu benutzenden Bänke an (von der LW-internen Bank-Liste)	Hängt vom verfügbaren Speicher ab
RESERVE	n	Reserviert # Bänke für Erweiterungen	0
EXTPAGES	n	Reserviert # Seiten für Erweiterungs-programme	0
BUFFER	ON¦OFF	Schaltet den internen Tastaturpuffer an oder aus	An (außer wenn der SDX-Puffer installiert ist)
PATH	Dn:<path>	Legt den Suchpfad fest	keine
KEY	code,atascii	Verknüpft Tastaturcode-Code mit atascii-Code	keine
COMMAND	Command #, ATASCII-code	Ordnet Tastaturcode einem internen Kommando zu (Erweiterte)	keine

LOADCFG	[Dn:][path]filename[.ext]	Config-Datei die beim ersten Start von LW geladen wird.	LW.CFG
LOADPDR	[Dn:][path]filename[.ext]	Druckertreiber der beim ersten Start von LW geladen wird.	LW.PDR
LOADEXT	[Dn:][path]filename[.ext]	Erweiterung die beim ersten Start von LW geladen wird.	LW.EXT
LOADMAC	[Dn:][path]filename[.ext]	Makro-Datei die beim ersten Start von LW geladen wird.	LW.MAC
LOADFNT	[Dn:][path]filename[.ext]	40Z-Font der beim ersten Start von LW geladen wird.	LW.FNT
LOADF80	[Dn:][path]filename[.ext]	80Z-Font der beim ersten Start von LW geladen wird.	LW.F80
DOCEXT	<ext>	Extander für Dateien die im Dokumentenmodus gespeichert werden.	LWD

8.3.1 KONFIGURATIONS-PROFILE MIT LW.SYS

Ab *LW* 3.2 gibt es die Möglichkeit beim Start andere Dateien als die *LW.** anzugeben welche beim Start geladen werden, das bedeutet daß verschiedene *LW.SYS*-Dateien gewählt werden können um beim ersten Start verschiedene Konfigurationen zu laden. Unter *SpartaDOS X* wird diese Funktion noch mächtiger, da dort der Ort und der Name der *LW.SYS* mit dem /S-switch angegeben werden kann. Im Abschnitt 9.6f über *SpartDOS X* findest Du weitere Informationen.

8.3.2 KONFIGURATION UNTER EINEM UNTERSTÜTZTEN DOS

Läuft *LW* unter einem unterstützten DOS (*DOS 2.5*, *MyDOS* oder *SpartaDOS X*), erkennt *LW* einen erweiterten Speicher, unter Berücksichtigung der Bänke die von einer RAM-Disk verwendet werden, automatisch (vorausgesetzt dass die *DOS 2.5* bzw. *MyDOS*-Standart-RAM-Disk-Treiber verwendet werden). Obwohl die manuellen Verwaltung der Speicherbänke unter diesen Umständen unnötig ist, ist es dennoch möglich die Standarteinstellungen zu überschreiben.

Mit einem unterstützten DOS untersucht *LW* die Hardware um herauszufinden, wie viele zusätzliche Speicherbänke das System hat und zieht dann die vom DOS benutzten Bänke ab (die einzige Ausnahme ist *SpartaDOS X*, welches hilfreicher Weise eine eigene Liste mit unbenutzten Bänken zur Verfügung stellt). Das Ergebnis ist eine Liste der *freien* Bänke des Systems. Du kannst eine Auswahl der Bänke aus der Liste wie folgt festlegen:

BANKS 1,2,3,4

Unter *DOS 2.5*, *MyDOS* oder *SDX* weist diese Zeile *LW* an die ersten vier Bänke der Liste der *freien* Bänke, die das Programm bei der ersten Initialisierung erzeugt hat, zu benutzen (die Bänke sind, beginnend bei eins, durchnummeriert entsprechend der Werte der Bank an PORTB). Wenn nicht genügend Bänke zur Verfügung stehen um den von „BANKS“ gestellten Bedarf zu decken, werden so viele Bänke wie möglich zugewiesen. In dem obigen Beispiel - unter der Voraussetzung dass auf dem Zielgerät wenigstens vier freie Bänke zur Verfügung stehen - würde *LW* eine Bank dem Makro/Einfüge/Directory-Puffer zuweisen (dies wird beim Einsatz von erweitertem Speicher immer gemacht bevor irgendeine andere Bank dem Textpuffer zugewiesen wird), drei Bänke für zusätzlichen Textpuffer nutzen und der Hauptpuffer wird auf 19k vergrößert.

Beachte: Wenn BANKED memory genutzt wird ist *LW*'s Makro-Puffer 4K groß, der Einfüge-Puffer umfasst 6.5K, und der Puffer für Directories bietet Platz für maximal 255 Dateien.

Die **RESERVE** Anweisung ist dafür gedacht um Speicherbänke des erweiterten Speichers für die Verwendung von Programmerweiterungen in Maschinensprache zu reservieren. Es wird erwogen in der Zukunft Applikationen wie eine Rechtschreibprüfung oder einen Inhaltsverzeichnis-Generator als Erweiterung für *LW* zu schreiben. Die Erweiterungen werden ab $3300 geladen und es muss zuerst Speicher mit „**EXTPAGES**“ zugewiesen werden, was lediglich die angegebene Anzahl von 256-Byte-Blöcken ab $3300 (bis max. 12) reserviert um ausführbaren Code der Erweiterung aufzunehmen (dieser Speicher wird von den 19k des Haupttextpuffers genommen: Erweiterungen sind nur bei der Verwendung von BANKED memory möglich).

EXTPAGES 4
RESERVE 1

Diese Anweisungen in *LW.SYS* reservieren vier Seiten RAM des Hauptspeichers ab $3300 für ein Erweiterungsprogramm und eine einzelne Bank des erweiterten Speichers für die Benutzung durch die Erweiterung wenn sie sie benötigt. Wenn der Speicher nicht reicht oder *LW* BANKED memory nicht nutzt, haben die Anweisungen keinen Effekt.

Ein anderes Beispiel:

BANKED ON
BANKS 1,2,3,4,5,6
RESERVE 1
EXTPAGES 8

Auf einem Rechner mit 320K der unter *DOS 2.5* mit einer 64K-RAMdisk läuft, konfiguriert die obige *LW.SYS*-Datei *LW* mit vier zusätzlichen Textpuffern, einer Bank für die Makro/Einfüge/Directory-Puffer und einer einzelne Bank (die letzte in der Liste) für die Verwendung von Erweiterungen. Da *LW* BANKED memory nutzt ist der Textpuffer 19K groß. Die **EXTPAGES**-Anweisung reserviert 8 Seiten (2K) ab $3300 für die Verwendung von ausführbaren Code, dadurch reduziert sich der Textpuffer auf 17k.

Durch den folgenden Befehl in *LW.SYS* kann BANKED memory abgeschaltet werden:

BANKED OFF

Das veranlasst *LW* **BANK**, **EXTPAGES** oder **RESERVED**-Anweisungen in *LW.SYS* zu ignorieren. Der Haupt- (und einzige) Text-Puffer wird 16K umfassen und der Einfüge-, Makro- und Directory-Puffer umfasst lediglich 1K. Das ist die selbe Konfiguration die *LW* auf einem Standard-XL/XE-Rechner mit 64k haben wird.

8.3.3 KONFIGURATION BEI EINEM ANDEREN DOS

Wird *LW* auf einem System geladen welches unter einem nicht unterstütztes DOS (also weder *SpartaDOS X*, *DOS 2.5* oder *MyDOS* mit den Standard-Ramdisk-Treibern) läuft, macht es keine Annahmen darüber ob das OS erweiterten Speicher für eine RAMdisk oder andere Zwecke benutzt. Deswegen ist unter diesen Umständen die Grundeinstellung KEIN BAKED memory zu nutzen. Um unter einem nicht unterstützen DOS BANKED memory nutzen zu können MUSST du eine eigene *LW.SYS*-Datei erstellen welche die folgende Zeile enthält:

BANKED ON

Da *LW* unter einem nicht unterstützen DOS nicht zwischen benutzten und freien Bänken unterscheiden kann enthält die Liste der Bänke, die beim ersten Start angelegt wird, ALLE im System verfügbaren Bänke. Wenn Du *LW* wegen einer bestehenden RAM-Disk oder

eines resistent, im erweitertem Speicher befindlichen DOS so konfigurieren möchtest dass nur bestimmte Bänke des erweiterten Speichers benutzt werden musst Du zuerst wissen welche Bänke vom OS genutzt werden und nach der „**BANKS**“-Anweisung die *freien* Bänke angeben.

Beispiel:

BANKED ON
BANKS 1,2,3,4

Die obige *LW.SYS*-Datei mit einem nicht unterstützten DOS auf einem 128K-Rechner veranlasst *LW* drei zusätzlicher Textbänke, 19K Puffer für den Haupttext und eine einzelne Bank für den Makro/Einfüge/Directory-Puffer einzurichten.

BANKED ON
BANKS 5,6,7,8,9,10,11,12
RESERVE 1

Auf einem Rechner mit 320K und einem nicht unterstützten DOS, welches die unteren vier Bänke des erweiterten Speichers als RAM-Disk nutzt, wird *LW* in diesem Beispiel eine Bank für den Makro/Einfüge/Directory-Puffer, sechs Bänke für zusätzlichen Text-Puffer und eine einzelne Bank die von Erweiterungen genutzt werden kann, einrichten. Es meidet dabei die unteren vier Bänke des Zusatz-Ram (die mit den Bank-Werten $23, $27, $2C und $2F, welche für die RAMdisk des OS benötigt werden).

Natürlich ist bei der Verwendung einer nicht unterstützten RAMdisk eine penible Planung nötig. Bei einem nicht unterstützten DOS ohne RAMdisk oder anderen Sachen, die sich ständig im erweiterten Speicher befinden, ist es am sichersten wenn *LW.SYS* nur aus einem „**BANKED ON**“-Befehl besteht. Das veranlasst *LW* so viele Bänke einzurichten wie es benötigt (maximal 16), das bringt bis zu 10 Textbänke. Bietet das System keine 16 Bänke, nutzt *LW* so viele wie es findet.

8.3.4 DER SUCHE-PFAD

Der Suchepfad erlaubt es Dir auf *LW*'s Konfigurationen, Makros, Druckertreiber und Zeichensätze in den angegebenen Laufwerken/Pfaden zuzugreifen *ohne* jedes mal den Pfad anzugeben wenn Du eine Datei laden möchtest.

PATH D8:;D1:>LW

Diese Zeile in einer Konfigurationsdatei veranlasst *LW*, wenn kein anderer Pfad angegeben ist, beim Laden von Zeichensätzen, Makros, Druckertreibern und Konfigurationsdateien *zuerst* im aktuellen Verzeichnis von „D8:“ zu suchen, anschließend im Ordner „LW“ auf „D1:“. Beachte, dass im aktuellen Verzeichnis immer *zuletzt* gesucht wird falls die Datei in den angegebenen Pfaden nicht gefunden wird. Beachte auch, dass das Standardlaufwerk von *LW* nicht unbedingt das selbe wie das Standardlaufwerk des DOS (üblicherweise „D:“) ist. *LW*'s Standardlaufwerk ist das selbe wie im Diskettenmenü angegeben, und wird *allen* Dateinamen vorangestellt wenn nicht ein anderes Laufwerk angegeben wird. Wenn Du aus irgendeinem Grund sicherstellen möchtest dass auch im Standard-Laufwerk des DOS gesucht wird füge „D:“ (ohne Anführungszeichen) ein.

8.3.5 DER TASTATURPUFFER

Die fünfte Sorte Kommandos in *LW.SYS* ist das „**BUFFER**“-Kommando (gefolgt von ON oder OFF). Das schaltet den Tastaturpuffer an oder aus. Der Tastaturpuffer ist grundsätzlich eingeschaltet, außer er entdeckt einen anderen aktiven Puffer (wie z.B. den *SpartaDOS X* Tastaturpuffer).

8.4 VERWENDUNG MEHRERER TEXTPUFFER

Mehrere Textpuffer erlauben es bis zu 160k Text auf einmal im Speicher zu halten. Anders als bei *AtariWriter Plus* bleiben die Dateien getrennt, können aber beim Drucken mittels des „include-bank"-Befehls verknüpft werden. Beim Speichern kannst Du mit „/A" in der Befehlszeile Text anhängen. Mit dem „include bank #"-Kommando im Haupttext (Bank #1) kannst Du die Seitennummerierung mit **<CTRL+?>** im Auge behalten, oder das ganze Dokument mit **<CTRL+V>** in der Vorschau ansehen ohne auf einzelne Dateien zugreifen zu müssen.

Vom Hauptprogramm aus ruft man die Bänke mit...

<SHIFT+CTRL+n> auf,

wobei <n> eine der Zifferntasten ist. <1> ruft immer die (nicht erweiterte) Hauptbank auf, während die anderen 9 Ziffern nach Deinen Wünschen verwendet werden können. In einem Makro folgen die selben Ziffern dem **<SELECT+CTRL+Z>**-Kommando.

8.5 BENUTZERDEFINIERTE ZEICHENSÄTZE (FONTS)

Auf der *LW*-Systemdisk werden mehrere alternative Zeichensätze mitgeliefert. Viele Fonts liegen sowohl in einer 40- als auch in einer 80-Zeichen-Version vor. 80-Zeichen-Fonts sind 512 Byte lang und haben ".F80" als Extender während 40-Zeichen-Fonts im Standard-Atari-Format vorliegen und den Extender „.FNT" tragen. Diese können jederzeit mit **<SHIFT +CTRL+N>** geladen werden. Abhängig davon ob sich der Editor im 40- oder 80-Zeichen-Modus befindet, wird ein weglassen des Extenders automatisch den zur Auflösung passenden Font laden. Du kannst das Laden der 40- oder 80-Zeichen-Version auch erzwingen indem Du den Extender in der Eingabeaufforderung angibst. Um Makros zu editieren ist der *MACRO.FNT* / *MACRO.F80* - Font am besten geeignet, da die Steuerzeichen fetter dargestellt werden und besonders in der 80-Zeichen-Version leicht von den alphanumerischen Zeichen zu unterscheiden sind. Die anderen Zeichensätze bieten neben viele verschiedene Arten und Gewichten alle den vollen internationalen Zeichensatz.

8.6 DIE TASTATUR ANPASSEN

LW erlaubt es die Tastatur auf zwei Weisen anzupassen: durch die Verwendung von Makros und durch den Einsatz eines benutzerdefinierten Tastaturlayouts (in der *LW.SYS*-Datei). Die Neudefinition durch ein Makro ist der beste Weg um Tastatureingaben zuzuweisen während die Tastaturdefinition per Datei es Dir ermöglicht die Tastatur vollständig neu zu belegen (um z.B. ein DVORAK-Layout zu realisieren).

8.6.1 DIE TASTATURTABELLE

Betrachten wir zuerst das **KEY**-Kommando, welches in der *LW.SYS*-Datei eingesetzt wird.

Die persönlichen Anpassung der Tastatur besteht aus Zeilen der Form...

KEY n,n

Das KEY-Kommando hat zwei numerische Argumente. Das erste Argument ist der Index der Hardware-Scan-Code-Tabelle. Die Hardware-Scan-Code-Tabelle umfasst 256 Byte und ist in vier Gruppen zu je 64 Byte unterteilt. Die ersten 64 Byte repräsentieren die normalen Tasten (ohne <SHIFT> oder <CONTROL>, die nächsten 64 Byte die <SHIFT>-Zeichen, gefolgt von 64 Byte <CONTROL>-Zeichen. Die letzten 64 Byte repräsentieren die Tasten die mit

<SHIFT> und >CONTROL> zusammen gedrückt werden (in dieser Anleitung <SHIFT +CTRL>+Taste). *LW´*s Tastaturtabelle entspricht exakt der Tabelle die im ATARI-OS enthalten ist, abgesehen von dem <SHIFT-CTRL>-Block (die Tabelle im ATARI-OS ist nur 192 Byte lang).

Die Tabelle auf der folgenden Seite zeigt das vollständige Standart-Tastenzuordnung von *LW*. Die Tasten stehen in der linken Spalte, die grauen Spalten stehen für das normale, <SHIFT>, <CONTROL> bzw. <SHIFT+CONTROL>-Zeichen.Das zweite Argument ist das Zeichens das mit der durch das erste Argument definierten Taste verknüpft werden soll.

Key		Normal		Shift		Ctrl		Shift+Ctrl
l	0	108	64	76	128	12	192	200
j	1	106	65	74	129	10	193	200
;	2	59	66	58	130	123	194	200
F1	3	28	67	8	131	19	195	200
F2	4	29	68	5	132	12	196	200
k	5	107	69	75	133	11	197	200
+	6	43	70	92	134	30	198	200
*	7	42	71	94	135	31	199	200
o	8	111	72	79	136	15	200	143
Invalid	9	200	73	200	137	200	201	200
p	10	112	74	80	138	16	202	144
u	11	117	75	85	139	21	203	149
Return	12	155	76	155	140	155	204	155
i	13	105	77	73	141	9	205	137
-	14	45	78	95	142	28	206	223
=	15	61	79	124	143	29	207	252
v	16	118	80	86	144	22	208	200
Help	17	200	81	200	145	200	209	200
c	18	99	82	67	146	3	210	200
F3	19	30	83	1	147	4	211	200
F4	20	31	84	26	148	22	212	200
b	21	98	85	66	149	2	213	200
x	22	120	86	88	150	24	214	200
z	23	122	87	90	151	26	215	200
4	24	52	88	36	152	180	216	164
Invalid	25	200	89	200	153	200	217	200
3	26	51	90	35	154	5	218	163
6	27	54	91	38	155	182	219	166
Esc	28	27	92	27	156	27	220	200
5	29	53	93	37	157	181	221	165
2	30	50	94	34	158	253	222	162
1	31	49	95	33	159	200	223	161
,	32	44	96	91	160	0	224	128
Space	33	32	97	32	161	32	225	160
.	34	46	98	93	162	96	226	224
n	35	110	99	78	163	14	227	142
Invalid	36	200	100	200	164	200	228	200
m	37	109	101	77	165	13	229	141
/	38	47	102	63	166	224	230	221
Inverse	39	129	103	129	167	129	231	129
r	40	114	104	82	168	18	232	146
Invalid	41	200	105	200	169	200	233	200
e	42	101	106	69	170	5	234	133
y	43	121	107	89	171	25	235	153
Tab	44	127	108	159	172	158	236	220
t	45	116	109	84	173	20	237	148
w	46	119	110	87	174	23	238	151
q	47	113	111	81	175	17	239	145
9	48	57	112	40	176	185	240	168
Invalid	49	200	113	200	177	200	241	200
0	50	48	114	41	178	176	242	169
7	51	55	115	39	179	183	243	167
Backspace	52	126	116	156	180	254	244	222
8	53	56	117	64	181	184	245	192
<	54	60	118	125	182	125	246	219
>	55	62	119	157	183	255	247	222
f	56	102	120	70	184	6	248	134
h	57	104	121	72	185	8	249	136
d	58	100	122	68	186	4	250	132
Invalid	59	200	123	200	187	200	251	200
Caps	60	130	124	131	188	132	252	132
g	61	103	125	71	189	7	253	135
s	62	115	126	83	190	19	254	147
a	63	97	127	65	191	1	255	129

Indem Du herausfindest wo in der Tabelle die Tastenkombination liegt die Du neu definieren möchtest, kannst Du die Tastatur völlig neu belegen. Du kannst z.B. die folgende Zeile in *LW.SYS* einfügen:

KEY 10,97

Die Taste <P> wird so belegt dass Du ein kleines „a“ erhältst wenn Du sie drückst. Beachte dass dies bedeutet dass <P> *überall im Programm* ein „a“ erzeugt. Die nützlichste Anwendung hierfür sind Aufgaben wie die Erstellung eines DVORAK-Tastaturlayout.

Etwas was Du bei der Neudefinition einer Taste beachten musst ist dass Du für *jede* umdefinierte Taste auch das Gegenstück definieren musst, sonst hast Du zwei Tasten die die selbe Funktion auslösen und eine Funktion die nicht mehr erreichbar ist.

Beachte dass die Funktionstasten des 1200XLvon *LW* vollständig unterstützt werden. Du kannst Kommandos Deiner Wahl auf alle 12 möglichen (F1..F4 einzeln und mit CTRL, SHIFT, SHIFT+CTRL) Tastenkombinationen legen.

8.6.2 KOMMANDOS DURCH MAKROS NEU ZUWEISEN

Die zweite Möglichkeit die Tastaturbelegung umzudefinieren bietet die *M*akro-fähigkeit von *LW* und hat den Vorteil dass jede Taste weiter das zu erwartende Zeichen erzeugt. Durch die Verwendung von Makros ist es möglich eine Tastatureingabe in eine andere zu wandeln *bevor* sie vom Programm verarbeitet wird.

Betrachten wir noch einmal die Windows-Tastaturkommandos für Ausschneiden und Einfügen und sehen uns das folgende Makro an (der *MACRO.FNT*-Font wurde bereits geladen bevor wir den Text eingeben):

In dieser Datei finden sich fünf Makros. Das Erste zeigt <CTRL+X> und gibt einfach <CTRL+C> aus wann immer <CTRL+X> im Editor gedrückt wird. Das Zweite gibt <CTRL+P> aus wenn <CTRL+V> gedrückt wurde und so weiter. Die letzten zwei Makros weisen den zwei Funktionen eine neue Taste zu die Anfangs umdefiniert wurden.

8.6.3 KOMMANDOS MIT HILFE DER VECTORTABELLE NEU VERKNÜPFEN

LW 3.2 bietet die Möglichkeit die Tastatur durch Ändern der internen Kommandotabelle umzudefinieren. Das wird durch das **COMMAND**-Kommando in der *LW.SYS* bewerkstelligt. COMMAND wird in der Form...

COMMAND <Kommando-Nummer>,<Tasten-Code>

...angegeben. Die „Kommando-Nummer“, die im Bereich von 0 bis 100 liegt, verweist auf die folgende Tabelle. „Tasten-Code“ entspricht dem ASCII-Code den Du mit dem Kommando verknüpfen möchtest.

Kommando-Nummer	Funktion	Default Tasten-Code
0	Cursor auf	28
1	Cursor ab	29
2	Cursor links	30
3	Cursor rechts	31
4	Seite auf	223
5	Seite ab	252
6	Makro wechseln	154
7	Schneide markierten Text aus	24
8	Tabulator-Stop setzen	159
9	Tabulator-Stop löschen	158
10	Tausche gefundenen String	18
11	Füge Text ein	22
12	Lösche Zeile	156
13	Markiere Text	13
14	Füge Text ein	157
15	Kopiere markierten Text	3
16	Verlasse Anwendung	145
17	Wandle Zeichen/Block in Kleinbuchstaben	153
18	Wechsle un/sichtbare Returns	253
19	Wandle Zeichen/Block in Grossbuchstaben	25
20	Extension hook 1	20
21	Extension hook 2	148
22	Suche String	6
23	Suche String rückwärtz	21
24	Cursor auf Anfang	8
25	End of text (EOF)	5
26	Anfang der Zeile	1
27	Ende der Zeile	26
28	Satz rechts	96

Kommando-Nummer	Funktion	Default Tasten-Code
29	Satz links	0
30	Wort rechts	94
31	Wort links	92
32	Füge Tabulator/nächster Tabulatorstop ein	127
33	Absatz links	95
34	Absatz rechts	124
35	Backspace	126
36	Füge Leerzeichen ein	255
37	Lösche Zeichen	254
38	Wechsle un/sichtbare Returns	219
39	Wandle Block in inverse Schrift	174
40	Wandle inverse Schrift in normale Schrift	172
41	Wechsle zwischen Einfüge/Überschreibe-Modus	222
42	Wähle Bank 1	161
43	Wähle Bank 2	162
44	Wähle Bank 3	163
45	Wähle Bank 4	164
46	Wähle Bank 5	165
47	Wähle Bank 6	166
48	Wähle Bank 7	167
59	Wähle Bank 8	192
50	Wähle Bank 9	168
51	Wähle Bank 10	169
52	Wechsle aut. Zeilenumbruch an/aus	23
53	Speichere markierten Text	137
54	Lade Druckertreiber	132
55	Gebe Finde-String an	134
56	Gehe zu Lesezeichen	135
57	Lade Makro	141
58	Lade Zeichensatz	142
59	Gebe Ersetze-String an	146
60	Gebe Bildschirmbreite (Anzahl d. Spalten) an	151
61	Gehe zu Laufwer/Pfad	136
62	Betrachte Datei	10
63	Lade Datei	12
64	Speiche Datei	19
65	Lösche gesammten Text	125
66	Globales Suchen &Ersetzen	7
67	Diskettenmenü	4

Kommando-Nummer	Funktion	Default Tasten-Code
68	Lade Konfigurations-Datei	15
69	Verbinde Text	9
70	Gebe Escape-Zeichen ein	27
71	Drucke Datei	16
72	Vorschau Datei	144
73	Setze Tabulatoren zurück	220
74	Zeige Version und Info an	221
75	Speichere Datei als...	147
76	Speichere Konfiguration	143
77	Lösche alle Tabulatoren	133
78	Zähle Wörter	14
79	Starte Makro	17
80	Setze Lesezeichen	2
81	Lade Erweiterung	123
82	Ändere Farben	11
83	Setze Optionen	149
84	Zeige Position im gedrucktem Dokument	224
85	Gehe zu Fenster Nummer	180
86	Vorhergehendes Fenster	181
87	Nächstes Fenster	182
88	Extension hook 3	183
89	<nicht definiert>	184
90	<nicht definiert>	185
91	Hilfe	176
92	Makro “ask”-Kommando	129
93	Alternatives ask-Kommando-Makro	251
94	Eingabe-Kommando-Makro	140
95	Zeige-Nachricht-Makro	150
96	Bestätige-Makro	131
97	Menü-Makro	138
98	Hole-Taste-Makro	139
99	Führe Subroutine-aus-Makro	152
100	Definiere bedingter-Sprung-Makro	130

8.6.4 1200 XL-TASTEN

Benutzer eines 1200XL können den Funktionstasten *LW*-Kommandos zuweisen. Die vier Tasten des 1200XL’s erscheinen in der Hardware-Scan-Code-Tabelle und können, genau wie alle anderen Tasten, mit dem **KEY**-Kommando definiert werden.

9 DOS PACKETE UND LW

Unter vielen auf den ATARI XL/XE-Computern verbreiteten DOS-Versionen arbeitet und konfiguriert *LW* sich selbst. Folgende DOSse werden unterstützt:

Atari *DOS 2.5*
MyDOS 4.5
SpartaDOS X (mit BANKED memory)

Auch wenn Dein DOS nicht in dieser Liste auftaucht *kann LW* gut damit harmonieren, vorausgesetzt es nutzt die selben CIO-Protokolle wie *Atari-DOS* und nutzt nicht das RAM unter dem OS.

BEACHTE: bis einschließlich Version 2.1 arbeitet *LW* auch mit diskettenbasierenden Versionen von *SpartaDOS*, *Atari DOS XE*, und vielen anderen DOSsen welche das RAM unter dem OS nutzen. Da seit *LW* 3.0 die 14K RAM zwischen $C000 und $FFFF nutzt arbeitet es **nicht** mehr mit einem DOS zusammen welches den gleichen Speicherbereich nutzt. Wenn Du *LW* mit *SpartaDOS 3.2* oder *DOS XE* nutzen möchtest lade eine Kopie von *LW 2.1* auf www.atari8.co.uk herunter.

9.1 SPEICHERANFORDERUNGEN

Unabhängig davon welches DOS Du verwendest musst Du sicherstellen dass MEMLO ($2E7,$2E8) unter $2000 liegt. Aus diesem Grund arbeitet *LW* am besten mit *SpartaDOS X* zusammen welches – wenn BANKED memory genutzt wird (wie es für *LW* nötig ist) – einen erstaunlich niedrigen MEMLO bietet.

LW harmoniert gut mit *DOS 2.5* und *MyDOS* mit der normalen Sektor-Puffer-Konfiguration. Resistente Handler und TSR-Programme unter diesem DOS arbeiten wahrscheinlich nicht mit *LW* zusammen.

Neben dem Speicherbereich von $2000 bis $9FFF und dem Speicherbereich für Module von $A000 bis $BFFF nutzt *LW* zusätzlich die 14KB RAM unter dem Betriebssystem von $C000 bis $FFF. LW schaltet das interne BASIC unter *DOS 2.5* bzw. *MyDOS* ab. Anwender von *SpartaDOS X* müssen *LW* mit dem „X“-Kommando starten da dessen Library deaktiviert werden muss.

9.2 ATARI DOS 2.5

Dieses System benötigt keine spezielle Behandlung durch *LW*, spezielle DOS-Features wie Unterverzeichnissen und Unterstützung der Eingabezeile sind nicht verfügbar. *LW* schaltetet BASIC automatisch ab. In der Grundeinstellung nutzt *LW* den ganzen verfügbaren Speicher (der nicht durch eine RAMdisk belegt ist) für zusätzliche Textpuffer.

9.3 ATARI DOS XE

LW 3.x arbeitet **nicht** mit *DOS XE* zusammen.

9.4 MYDOS 4.5

Dieses DOS harmoniert gut mit *LW*. Unterverzeichnisse werden durch einen vorangestellten Doppelpunkt im Diskettenmenü gekennzeichnet, Bewegungen im Verzeichnisbaum sind möglich, genau wie das Anlegen und Löschen von Unterverzeichnissen. Um den Inhalt des aktiven Verzeichnisses anzeigen zu lassen vergesse nicht im Diskettenmenü zuerst die Laufwerksnummer zu löschen indem Du <0> drückst.

LW erkennt RAMdisks unter diesem DOS korrekt und meidet den belegten Speicherbereich. Bei sorgsamer Konfiguration ist es auf einem Rechner mit genügend Speicher möglich eine grosse RAMdisk und zehn Textbänke nebeneinander zu nutzen.

9.5 DISKETTEN-BASIERENDES SPARTADOS

LW 3.x arbeitet **nicht** mit diskettenbasierenden Versionen von *SpartaDOS* zusammen. Mit *LW 3.x* musst Du *SpartaDOS X* nutzen.

9.6 SPARTADOS X

SpartaDOS X erweist sich als die bei weitem beste Arbeitsumgebung für *The Last Word*. Dieses DOS biete eine Fülle an Besonderheiten welche flexibles Arbeiten und eine effiziente Dateiverwaltung ermöglichen.Um eine größtmögliche Konfigurierbarkeit zu erreichen unterstützt *LW SDX* Eingabezeilenkommandos und Arbeitsumgebungsvariabeln.

Wenn Du *LW* mit *SpartaDOS X* nutzt, kannst Du in der Eingabezeile folgende Optionen nutzen:

X LW [file[,file…]] [/M[macro_ID]] [/Ppath] [/Ddrive] [/Ssys_file] [/Q] [/X] [/T]

M	: Deaktiviert das selbststartendes Makro oder startet das Makro „macro_ID““
P	: Setzt Suche-Pfad
D	: Setze Default-Laufwerk
S	: Setze SYS-Dateiangaben
Q	: Deaktiviert den Startbildschirm (wenn keine Dateiname in der Kommandozeile angegeben wird)
T	: Zeige Startbildschirm (wenn Dateiname in der Kommandozeile angegeben wird)
X	: „*LW* sauber“ laden (ohne eine Konfig, Fonts, Druckertreiber oder Makros)

Beachte das sich zwischen **/M** und **/P** und deren Argumenten kein Leerzeichen befindet. Wenn Du in der Eingabezeile einen Suchpfad für *LW* angibst überschreibt dies die **LWPATH**-Variable. Das Argument für **/P** ist exakt das gleiche wie das Argument für das **SET LWPATH**-Kommando. Im folgenden Abschnitt findest Du eine ausführliche Erklärung.

X LW TEST.DOC /M% /PD1:>LW>

Diese Eingabe veranlasst *LW* die Datei „*TEST.DOC*“ zu laden. Wenn die Datei nicht existiert wird eine leere Datei mit dem Namen „*TEST.DOC*“ angelegt und im Editor geöffnet. *LW* wird auch versuchen das Makro „%“ in *LW.MAC* zu starten. Letztlich setzt *LW* den Suchepfad (**LWPATH**) auf “D1:>LW>”. (siehe Kapitel 8.3.4: “Der Suche-Pfad” für eine ausführliche Beschreibung).

X LW.EXE /SD3:>TEXT>TEXTEDIT.SYS /Q

Dieses Kommando wird LW ohne Startbildschirm starten und versuchen anstelle von *LW.SYS* die Datei *TEXTEDIT.SYS* auf Laufwerk 3 im Ordner TEXT zu laden.

X LW.EXE README /T

Dieses Kommando wird versuchen beim Start von *LW* die Datei *README.TXT* zu laden **und** dabei den Startbildschirm anzeigen welcher normalerweise, bei der Angabe eines Dateinamens beim Laden von LW, unterdrückt wird.

Der *LW*-Tastaturpuffer wird automatisch deaktiviert wenn das Programm feststellt daß der *SDX*-Tastaturpuffer **aktiviert** ist. Wenn Du den *LW*-Tastaturpuffer bevorzugst stelle sicher daß der *SDX*-Tastaturpuffer mit „KEY OFF“ in der Kommandozeile deaktiviert wird bevor Du das Programm lädst. Du kannst eine kleine Batch-Datei erstellen:

KEY OFF
X LW
KEY ON

Damit wird sichergestellt daß der *SDX*-Puffer aus ist , dann *LW* gestartet und anschließend der Puffer wieder aktiviert. Der aktuelle *SDX*-Tastaurpuffer ist dem *LW*-Puffer allerdings so ähnlich daß Du genauso gut den *SDX*-Puffer nutzen kannst.

9.6.1 DIE SPARTADOS X ARBEITSUMGEBUNGS-VARIABLEN

Um ein Maximum an Komfort zu erreichen können Einstellungen von *LW,* welche über die *SDX*-Kommandozeile vorgenommen werden können, auch als *SDX*-Arbeitsumgebungs-variable eingerichtet werden. Um die **LWPATH**-Variable zu einzurichten musst Du eine Zeile ähnlich der folgenden in Deine *CONFIG.SYS* einfügen:

SET LWPATH=D8:;D1:\LW

Wenn kein anderer Pfad angegeben ist wird dies *LW* veranlassen beim Laden von Fonts, Makros, etc, zuerst das aktuelle Verzeichnis von Laufwerk 8 zu durchsuchen und anschließend das Verzeichnis „LW“ auf Laufwerk „D1:“. Beachte das, wenn die Datei unter den **LWPATH**-Pfaden nicht gefunden wird, auf dem Standardlaufwerk *immer* zuletzt gesucht wird. Beachte auch, dass *LW*'s Standardlaufwerk nicht zwangsweise identisch mit dem Standardlaufwerk des DOS (normalerweise „D:“) ist. *LW*'s Standardlaufwerk ist im Diskettenmenü zu sehen und wird, wenn nicht explizit ein anderes Laufwerk angegeben wird, allen Dateinamen vorangestellt. Wenn Du aus irgendeinem Grund sicher sein willst dass auch im Standardlaufwerk des DOS gesucht wird, füge „D:“ (ohne Anführungsstriche) als Eintrag in **LWPATH** ein.

Ein weiterer Punkt dem Beachtung geschenkt werden sollte ist dass beim Laden von *LW* ein in der *LW.CFG*-Datei mit dem **LWPATH**-Kommando angegebener Suche-Pfad unverzüglich den Suche-Pfad, der Teil der *SDX*-Konfiguration-Datei (*CFG*) ist, überschreiben wird. Es ist ein sorgsames Abgleichen der Konfigurationsdateien erforderlich. Wenn Du sicherstellen willst dass nur die Pfade in **LWPATH** genutzt werden entferne alle **PATH**-Zeilen in Deinen *CFG*-Dateien.

Folgende Arbeitsumgebungsvariablen können ebenfalls genutzt werden um *LW* zu paramentieren:

LWSYS
LWDRIVE
LWSPLASH

Ein paar Beispiele:

SET LWSYS=D3:TEXT.SYS

Diese Zeile in der *CONFIG.SYS* läßt *LW* beim ersten Start anstelle *LW.SYS* die Datei *TEXT.SYS* von Laufwerk drei laden.

SET LWDRIVE=D2:

LW´s Default-Laufwerk wird auf D2: gesetzt.

SET LWSPLASH=1

Beim Laden von LW wird auf jeden Fall der Startbildschirm angezeigt, unabhängig davon ob ein Dateiname in der Kommandozeile angegeben wurde. Wird LWSPLASH=0 gesetzt wird der Startbildschirm auf jeden Fall unterdrückt.

9.6.2 SPARTADOS X SPEICHERKONFIGURATIONEN

LW 3.x setzt voraus dass *SDX* im „BANKED memory“ Modus läuft. Das liegt daran dass das Programm unter dem OS geladen wird (dadurch wird „**USE OSRAM**“ unbrauchbar), und wenn *SDX* den konventionellen (LOW) Speicher nutzt wird **MEMLO** zu hoch liegen um *LW* laden zu können. Also musst Du, außer Dein Rechner ist mit mehr als 128k ausgestattet, Deinen Rechner mit folgender Zeile in der *CONFIG.SYS* booten:

USE BANKED

Wenn Dein Atari *mehr* als 128K hat wird *SDX* automatisch BANKED RAM nutzen, außer Du gibst ihm andere Anweisungen.

Wie auch unter *DOS 2.5* und *MyDOS* wird *LW* automatisch die Anzahl der Speicherbänke erkennen die nicht vom DOS genutzt werden und wird sich bis zu zehn von ihnen für eigene Zwecke sichern. Wenn Du eine RAMdisk (mit *RAMDISK.SYS*) eingerichtet hast wird *LW* diese ebenfalls berücksichtigen und nur die vom DOS als unbenutzt gekennzeichneten Speicherbereiche nutzen.

10 LW KOMMANDO ZUSAMMENFASSUNG

Es folgt eine vollständige Liste mit allen Kommandos des Editors und des Druckformatierers (print formatter). In der ersten Spalte steht das Editor-Kommando, in der zweiten Spalte die Beschreibung und in der dritten Spalte, wenn es von dem Editor-Kommando abweicht, das entsprechende Makro-Kommando. Wenn kein Makro-Kommando angegeben ist können die Kommandos auch innerhalb eines Makros angewendet werden. Beachte daß Makro-kommandos welche <SHIFT+CTRL>-Eingaben vervielfältigen als INVERSE <CTRL>-Eingaben gemacht werden.

10.1 EDITOR KOMMANDOS

EDITOR-KOMMANDO	FUNKTION	ENTSPRECHENDES MAKRO
CTRL A	Zeilenanfang	
CTRL B	Setze Lesezeichen	
CTRL C	Kopiere markierten Text	
CTRL D	Diskettenmenü	
SHIFT+CTRL D	Lade Druckertreiber	inverses CTRL+D
CTRL E	Dateiende	
SHIFT+CTRL E	Lösche alle Tabulatoren	inverses CTRL+E
CTRL F	Suche String	
SHIFT+CTRL F	Gebe zu suchenden String an	inverses CTRL+F
CTRL G	Globales Suchen & Ersetzen	
SHIFT+CTRL G	Gehe zu Lesezeichen	inverses CTRL+G
CTRL H	Textanfang	
SHIFT+CTRL H	Maske für Diskettenmenü	inverses CTRL+H
CTRL I	Verbinde Datei	
SHIFT+CTRL I	Schreibe Block auf Disk	inverses CTRL+I
CTRL J	Zeige Datei	
CTRL K	Ändert Bildschirmfarben	
CTRL L	Lade Datei	
CTRL M	Markiere Textblock	
SHIFT+CTRL M	Lade Makro	inverses CTRL+M
CTRL N	Zeige Anzahl Wörter	
SHIFT+CTRL N	Lade Zeichensatz	inverses CTRL+N
CTRL O	Lade Konfiguration	
SHIFT+CTRL O	Speichere Konfiguration	inverses CTRL+O
CTRL P	Drucke Datei	
SHIFT+CTRL P	Datei Vorschau	inverses CTRL+P
CTRL Q	Starte Makro	
SHIFT+CTRL Q	Verlasse Anwendung	inverses CTRL+Q
CTRL R	Ersetze gefundenen String	
SHIFT+CTRL R	Gebe Ersetze-String an	inverses CTRL+R

KOMMANDO	FUNKTION	ENTSPRECHENDES MAKRO
CTRL S	Speicher Datei	
SHIFT+CTRL S	Speichere Datei unter...	inverses CTRL+S
CTRL T	Extension hook 1	
SHIFT+CTRL T	Extension hook 2	inverses CTRL+T
CTRL U	Suche String aufwärts	
SHIFT+CTRL U	Setze Optionen	inverses CTRL+U
CTRL V	Füge Text ein	
CTRL W	Zeilenumbruch an/aus	
SHIFT+CTRL W	Bildchirmauflösung	inverses CTRL+W
CTRL X	Füge Text ein	
CTRL Y	Wandelt Zeichen/Block in Kleinbuchstaben	
SHIFT+CTRL Y	Wandelt Zeichen/Block in Großbuchstaben	inverses CTRL+Y
CTRL Z	Zeilenende	
CTRL ↑	Zeile auf	
CTRL ↓	Zeile ab	
CTRL ←	Spalte links	
CTRL →	Spalte rechts	
SHIFT ↑	Absatz zurück	
SHIFT ↓	Absatz vor	
SHIFT ←	Wort links	
SHIFT →	Wort rechts	
SHIFT+CTRL ↑	Bildschirm auf	
SHIFT+CTRL ↓	Bildschirm ab	
CTRL [	Satz links	
CTRL]	Satz rechts	
SHIFT+CTRL [	Bei markierten Text invertierung aufheben	
SHIFT+CTRL]	Markierten Text invertieren	
TAB	zum nächsten Tabulator	
CTRL TAB	Lösche Tabulator	
SHIFT TAB	Setze Tabulator	
SHIFT CTRL TAB	Standart-Tabulatoren setzen	
ESCAPE	Abbruch / Gebe Steuerzeichen ein	
CTRL ESCAPE	Gebe Steuerzeichen ein	[ESC]
SHIFT ESCAPE	Gebe Steuerzeichen ein	[ESC]
CTRL ?	Gebe Druckposition an	
SHIFT+CTRL ?	Zeige Informationen zum Programm an	[siehe „SET“-Kommando]
CTRL ;	Zeige die Cursor Position	

KOMMANDO	FUNKTION	ENTSPRECHENDES MAKRO
CTRL <	Lösche gesamten Text	
SHIFT <	Lösche gesamten Text	
SHIFT+CTRL <	RETURN anzeigen/verbergen	
CTRL >	Füge Leerzeichen ein	
SHIFT >	Füge gelöschten Text ein	
SHIFT+CTRL >	Einfügen/Überschreiben	[siehe „SET“-Kommando]
CTRL DELETE	Lösche Zeichen unter Cursor	
SHIFT DELETE	Lösche Wort / Zeile / Satz / Absatz	
RETURN	Absatzende	
CAPS	Groß/Kleinschrift	[siehe „SET“-Kommando]
CTRL CAPS	verriegelt die <CTRL>-Taste	
SHIFT CAPS	Großschrift	
SHIFT+CTRL CAPS	Sonderzeichen verriegeln	
INVERSE	Inverse/Normal	[siehe „SET“-Kommando]
SHIFT CTRL INV	konvertiert Normal/Invers	[Kein]
SHIFT CTRL SPC	Festes Leerzeichen	
CTRL 1	Listen anhalten	[Kein]
CTRL 2	Sichtbare RETURNs an/aus	Inverse 2
CTRL 3	End of File / End of Text	Inverse 3
CTRL 4	Gehe zu Textbank	Inverse 4
CTRL 5	Vorhergehende Textbank	Inverse 5
CTRL 6	Nächste Textbank	Inverse 6
CTRL 7	Extension hook 3	Inverse 7
CTRL 0	Hilfe	Inverse 0
SHIFT+CTRL 1	Wähle Haupttextbank	[siehe „SET“-Kommando]
SHIFT+CTRL 2-0	Wähle erweiterte Textbank	[siehe „SET“-Kommando]
START	Starte das #-Macro (Wenn mit einem Zeichen gedrückt startet das verknüpfte Kommando)	
SELECT+CHAR	Gebe inverses Zeichen ein	
OPTION	Starte Makro	
HELP	Starte Hilfe-System	

10.2 SPEZIELLE TASTEN

Die folgenden Zeichen haben, in der Eingabezeile eingegeben, eine spezielle Bedeutung, außer wenn ein **<CTRL+ESC>** vorangeht (**<ESC><ESC>** von einem Makro aus):

TASTE	FUNKTION
CTRL+P	Fügt bei der Eingabe eines Dateinamens das Laufwerk, den Pfad und den Namen der aktuellen Datei im Editor ein.
CTRL+N	Fügt bei der Eingabe eines Dateinamens nur den Pfad und den Namen der aktuellen Datei ein.
CTRL+L	Fügt bei der Eingabe eines Dateinamens den vollständigen Pfad der zuletzt in eine beliebige Bank geladenen Datei ein.
CTRL+V	Fügt den Inhalt des Einfügepuffers in die Eingabezeile.
ESC oder BREAK	Verlässt eine Eingabe
CTRL oder SHIFT+ESC	Erlaubt die Eingabe von Steuerzeichen in der Eingabezeile.

10.3 MAKRO KOMMANDOS

Die folgenden Kommandos sind nur innerhalb eines Makros verfügbar und werden invers eingegeben.

KOMMANDO	FUNKTION	
INV CTRL A	Frage nach einer Eingabe. Gefolgt von einem mit <RETURN> zu beendenden Text und einer optionalen default-Eingabe. Es folgt ein <INV CTRL+L> „Hole Zeile“-Kommando“.	
INV CTRL B	Springe (Branch) zu Makro <Makro Taste>. Legt das Makro fest welches nach negativer Bestätigung, Laden eines Makros, negativen suche String/ nächstes Gehe-zu-Lesezeichen gestartet wird. Das angegebene Makro wird auch gestartet wenn eine (verlinkte) Lade-Operation nicht beendet werden kann.	
INV CTRL C	Bestätige (Y/N). Arbeitet das Makro weiter ab wenn „Y“ gedrückt wird, sonst wird abgebrochen oder ein vorgewähltes Makro gestartet.	
INV CTRL J	Macro Menu <menu text><RETURN>. Gibt eine Nachricht aus und startet dass mit dem nächsten Tastendruck verknüpfte Makro	
INV CTRL K	Warte auf Tastendruck durch den Benutzer	
INV CTRL L	Hole Zeile-Modus. Akzeptiert Tastatureingaben bis <RETURN> gedrückt wird. Funktioniert auch in Eingabedialogen.	
INV CTRL V	Gibt Nachricht aus <Nachricht-Text><RETURN>.	
INV CTRL X	Führt ein Makro aus <Makro Taste>.	
INV CTRL Z	Setze Option/Testbedingung <option>. Gefolgt von einem der folgenden Zeichen (nicht invers) um Optionen zu wählen oder Tests auszuführen.	
	U	Großbuchstaben (Uppercase)
	L	**Kleinbuchstaben (Lowercase)**
	R	Invers
	N	**Normal**
	I	Einfügemodus (Insert Mode)
	O	**Überschreibemodus (Over-Type Mode)**
	H	Unterdrücke Bildschirmupdates (Hide)
	V	**Zeige Bildschirmupdates an (view)**
	K	Starte Textmarkieren (Block) an aktueller Cursorposition
	1	Wähle Haupttext (Textbank 1)
	2-0	Wähle erweiterten Text (Textbank 2 - 10)
	X	**Lösche Text Flags**
	B	Kehre in die Textbank zurück in der sich der Cursor vor dem Aufruf des Makros befand
	M	**Überprüfe ob ein Block markiert wurde (selbst wenn nichts ausgewählt ist: stelle ein Sprungkommando voran)**
	S	Überprüfe ob ein Text ausgewählt wurde (mit einem vorangestellten Sprungkommando)
	C	**Überprüfe ob seit dem letzten Speichern Änderungen am Text vorgenommen wurden (mit einem vorangestellten Sprungkommando)**
	A	Überprüfe ob die Datei bereits gespeichert wurde (mit einem vorangestellten Sprungkommando)

11 DRUCK-FORMATIERUNGSKOMMANDOS

Die folgenden Kommandos wirken sich auf das *gedruckte* Dokument aus und werden in inversen Groß- oder Kleinbuchstaben eingegeben. Alle numerischen Argumente müssen ebenfalls invers eingegebene werden (nutze **<SELECT+KEY>** um ein einzelnes Zeichen invers einzugeben). Dateinamen sowie Kopf/Fusszeilen sollten als normaler Text, abgeschlossen mit **<RETURN>**, eingegeben werden.

KOMMANDO (invers)	FUNKTION	GRUNDEINSTELLUNG
A<n>	Erste zu druckende Seite	1
B<n>	Unterer Rand	61
C<line>	Zeile zentrieren	
D	Fettdruck an/aus	
E<line>	Randlinie rechts	
F<line>	Fußnote festlegen	
G<file>	Datei einbinden	
G<n>	Textbank einbinden	
H<line>	Kopfzeile festlegen	
I	Schrägschrift an/aus	
J<n>	Blocksatz	0
L<n>	Linker Rand	10
M<n>	Rand ausrücken	
N<n>	Neue Seite (wenn Rest weniger als n Zeilen)	0
O<n>	Gebe Steuerzeichen aus	
P<n>	Seitenlänge	66
R<n>	Rechter Rand	70
S<n>	Druckstil	
T<n>	Oberer Rand	5
U	Unterstreichen an/aus	
V<file>	1:1-Ausdruck auf dem Drucker	
W<n>	Pause bei neuer Seite	0
X<n>	Druckercode ausgeben	
Y<n>	Zeilenabstand	1
Z<n>	Letzte zu druckende Seite	
><n>	Absatz links einrücken	
<<n>	Absatz rechts einrücken	
[<n>	Linker Rand Kopf/Fußzeile	10
]<n>	Rechter Rand Kopf/Fußzeile	70
?<n>	Start-Seitennummer	1
#	Füge Seitennummer ein	
@<n>	Überspringe beim Drucken <n> Seiten	
!<n>	Ebene der Überschrift	
&	Reset Überschriften-Ebene	
(	Ignoriere bis zur „)“	
_	Fester Bindestrich	
;	Zeile mit Kommentar	
-	intelligenter Bindestrich	
.	Festes Leerzeichen	
Pfeil ↑	Hochgestellt an/aus	
Pfeil ↓	Tiefgestellt an/aus	

12 TECHNISCHE ANMERKUNGEN FÜR PROGRAMMIERER

Dieser Abschnitt gibt einen Abriss von LW´s Speicheraufteilung und behandelt anschließend einige grundlegende Ideen des Programms. Es ist *keine* Anleitung zum Schreiben von Erweiterungen für LW (hierfür verweise ich auf die „extension developer's documentation"). Wenn Du Probleme beim Betrieb von LW hast solltest Du Dir den Abschnitt über die Speicherverwaltung ansehen. Ich spreche auch verschiedene Tricks an die LW anwendet und warum es so geworden ist wie es ist.

12.1 ASSEMBLER-ADD-INS

Von Version 2.1 an bietet *LW* interessierten Maschienenspracheprogrammierern die Möglichkeit ihre eigenen MC-Routinen zu schreiben um die Funktionalität der Software zu erweitern. Diese reine Maschinensprache-Datei muss sich an strickte Richtlinien halten, und ist (ab Version 3) nur mit erweiterten Speicher möglich. Add-ins (oder Erweiterungen) werden ab $3300 geladen und können bis zu 3K umfassen. Sie können sich bei Tastatureingaben im Editor einhaken, in die Reset-Routine, den Druckformatierer, etc.. Es ist alles vorgesehen; von einer Zeichentabelle bis hin zur Rechtschreibprüfung.

Ein Toolkit zum Schreiben von Add-ins ist in der Retail-Version von *The Last Word* verfügbar, es beinhaltet eine vollständige Liste mit Einsprungadressen für das Programm, Beispiel- Add-ins, Beispielprogramme, und Richtlinien zum Schreiben eigener Programmmodule.

12.2 SPEICHERVERWALTUNG

Die ausführbare Datei *LW.COM* ist über 34K lang, etwa 6K davon dienen der Initialisierung und werden, wenn das Programm einmal läuft, nicht mehr benötigt. 14K des Programmcodes liegen unter dem OS-ROM zwischen $C000-$CFFF und $D800-$FFFF, der Speicher zwischen $2000 und $3200 ist mit Programmcode, Daten & Puffern belegt. Wird BANKED Memory verwendet belegt der Haupttextpuffer den Speicherbereich von $3900-$7FFF (abhängig von der Größe des add-in-Puffer welcher, wenn er aktiviert ist, sich ab $3300 befindet),. Der Bereich von $4000 bis $7FFF ist ein Fenster auf den Textpuffer welcher sich bei BANKED Memory in einer Bank befindet. Ist BANKED Memory deaktiviert belegt der Haupttextpuffer den Bereich von $4000 bis $7FFF, $3300 - $3FFF werden von internen Puffern benutzt.

LW nutzt mehrere andere Bereiche zum Ablegen von Daten, einschließlich der gesamten oberen Hälfte der Page Zero ($80-$FF) und $5A-$5D. Der *gesamte* Speicher von $400 bis $6FF wird von LW für Puffer genutzt. Dieser Bereich schließt den Kassettenhandler des OS ein den Du (hoffentlich) nicht benötigst. Beachte dass es (ab Version 3.0) nicht länger möglich ist LW aus dem DOS mit "Run at Address" neu zu starten. Das kommt daher dass sich das Programm selbstständig aus dem Bereich unter dem OS entfernt wenn Du LW in das DOS verlässt.

Du solltest sicherstellen dass keine residenten Handler den Speicherbereich von $400 bis $6FF belegen, sonst wird es sicherlich Konflikte mit LW geben. Gleichermaßen dürfen TSRs oder residente Handler nicht über $2000 reichen. Das Setup von SpartaDOS X kann sehr niedrige Werte für MEMLO bereitstellen was es Dir erlaubt residente Handler zu verwendent. Bei DOS 2.5-Systemen gibt es höchstwahrscheinlich keinen zusätzlichen Platz für zusätzliche Sektorpuffer ect.

Mit den größten Speicherbedarf in *LW* hat die 80-Zeichen-Anzeige, welche um die 9K RAM benötigt. Aus diesem Grund ist es ein Meisterwerk die vollständige Funktionalität von LW in den restlichen Speicher zu kriegen. Trotzdem bleibt LW seinem ursprünglichen Vorsatz treu: es sollte auf einem nicht erweiterten 800XL lauffähig sein.

Eine Trick der es ermöglicht *LW* mit so wenig Code zu realisieren ist alle Programmvariablen in Page Zero zu legen. Die gesamte obere Hälfte von Page Zero wird von *LW* verwendet. Das wird durch die Tatsache ermöglicht dass *LW* die Fließkomma-Routinen des OS nicht nutzt, welche $D4 - $FF für sich beanspruchen. *LW* verwendet für Bildschirmausgaben noch nicht einmal die CIO; es nutzt seine eigene ausgeklügelte Routine für die formatierte Bildschirmsnzeige. Das Bildschirmeditor-Device wird beendet sobald das Programm gestartet wird und wird erst wieder gestartet wenn man *LW* zum DOS verlässt. Der Einsatz von Page 0 anstelle von absoluten Adressen macht *LW* zwischen 20 und 30 Prozent schlanker. Das heißt auch dass das Programm bedeutend schneller läuft als wenn es sich stärker auf absolute Adressen gestützt hätte.

Es gibt noch andere Techniken die den Code schlanker gemacht haben. *LW* nutzt viele Speicherstellen als Flag, welche nur an oder aus sind. Diese Flags werden mit der „BIT“-Instruktion des 6502 abgefragt, was bedeutet dass nur Bit 7 benötigt wird um ein Bit zu setzen oder zu löschen. Anstatt eine „0“ im Flag zu speichern um das Flag zu löschen oder eine „128“ um es zu setzen nutze ich...

LSR <Adresse> was mit nur 2 Byte Code eine „0“ in bit 7 setzt.

Um es zu setzen nutze ich...

SEC
ROR <Adresse>

Das sind lediglich 3 Byte (vorausgesetzt <Adresse> liegt in Page 0). Die anderen Bits in dem Byte sind unwichtig, mitunter ist mehr Genauigkeit nötig wenn sowohl Bit 6 als auch Bit 7 aus dem Byte genutzt werden. Bit 6 wird mit der „BIT“-Instruktion ebenfalls getestet und wird in das Overflow-Flag übertragen. Verzweigungen können mit „BVC“ und „BVS“ realisiert werden. Diese Technik spart nicht wirklich viel Platz wenn eine lange Kette Flags gelöscht wird, es ist so einfach wie eine 0 zu laden und sie in den Flags zu speichern. Wenn aber das Setzen oder Löschen von weit verstreut liegenden Flags nötig ist lohnt sich die Einsparungen bald.

12.3 PROGRAMM DESIGN

LW hat eine spezielle Art den Text im Speicher zu verwalten um seine Geschwindigkeit zu erreichen. Während viele Textverarbeitungen den Text ständig im Speicher halten benutzt *LW* ein Zeigersystem um sicherzustellen das der freie Speicher im Puffer direkt vor dem Cursor ist. Deshalb gibt es, unabhängig von der Länge des Textes, keine Verzögerung bei der Eingabe wenn Du schreibst. Während Du den Cursor bewegst wird der Text durch den Speicher bewegt. Wenn die Routine für den Bildaufbau auf den Cursor trifft überspringt sie den freien Speicher im Puffer und zeigt den restlichen Text an welcher sich ganz vorne im Puffer befindet.

Obwohl LW *i*n kompakter Maschinensprache erstellt wurde ist es trotzdem ein modulares Programm. Kaum etwas vom Code kommt doppelt vor und - um Platz zu sparen - werden Unterroutinen anstelle von Makros verwendet. Die Reduzierung der Codegröße die durch die Verwendung von Unterroutinen erreicht wird kann beachtlich sein. Ungeachtet der Tatsache das *LW* 2.1 im Jahr 2000 als „fertiges“ Programm gedacht war gab es acht Jahre später immer noch bedeutende Platzeinsparungen und Verbesserungen der Effizienz durch die kontinuierlichen Überarbeitung des Quellcodes.

Diese 80-Zeichen-Version von *LW* kommt gute 20 Jahre nach meinen ersten Experimenten mit dem 8-bit Atari. Ich begann das Programmieren mit BASIC, dann wechselte ich zu C und schließlich zu Assembler. Nach Versuchen mit CC65, Action!, PL65 und viele andere Sprachen denke ich dass Maschinensprache die beste Wahl ist wenn Schlankheit ein

entscheidender Faktor ist und das war bei *LW* immer der Fall. Nachdem ich viele Jahre versucht habe den Atari Makro-Assembler mit *SpartaDOS X* zusammen arbeiten zu lassen habe ich letztlich meinen eigenen Makro-Assembler geschrieben. Bis vor kurzem benutzte ich das Ergebnis - *MA65* - um *LW* zu kompilieren. Erst als die großartige WUDSN IDE Cross-Entwicklungsplattform für Eclipse veröffentlicht wurde portierte ich LW's 20,000 Zeilen Sourcecode auf den PC, wo ich die Entwicklung der Software mit dem vorzüglichen WUDSN Plug-In für die Eclipse-Platform zusammen mit dem ATASM Cross-Assembler fortsetzte. Jetzt braucht es nur eine Sekunde das ganze Programm zu kompilieren. Nichtsdestotrotz bleibt der MA65 meine erste Wahl wenn es um einen diskettenbasierenden Assembler auf dem Atari geht.

12.4 ENTWICKLUNG UND TESTEN

Nachdem ich neun Jahre darüber nachgedacht habe eine Textverarbeitung zu schreiben begann ich Anfang 1999 mit der Arbeit an *LW*. Es brauchte nur drei Monate bis zu einer voll funktionsfähigen Version, und weitere zwei Monate bis sie stabil arbeitete. Ich testete *LW* in dem ich Makros als Hilfsprogramme schrieb und diese umfangreiche Dokumentation erstellte, was mir Fehler im Programm und im Design zur späteren Korrektur aufzeigte. In dieser Zeit habe ich wahrscheinlich hunderte Neucompilierungen durchgeführt. Das Programm gipfelte im Jahr 2000 in der Version 2.1.

Nachdem Version 2.1 im Jahr 2000 veröffentlicht wurde (die Öffentlichkeit es aber nicht zu sehen bekam!), gab es eine Pause von acht Jahren die mein Atari in einer Kiste auf dem Kleiderschrank verbrachte. Dann, im November 2008, ließ mich eine nostalgische Unterhaltung mit einem Freund, der ebenfalls Interesse an der Programmierung hat, den Atari wieder entdecken. Begeistert darüber das er noch funktionierte, habe ich mich bald daran gemacht mit dem fantastischen Zubehör, welches inzwischen erhältlich, war meine Dateien zum PC zu übertragen und zur Jahreswende wurde *LW* 2.1 letztendlich der Öffentlichkeit zugänglich gemacht. Es zeichnete sich so ein Interesse bei den Atarianern ab dass es auf der Hand lag das Programm zu aktualisieren um Nutzen aus der ganzen neuen Hardware zu ziehen die inzwischen entwickelt wurde. 320K waren praktisch das Minimum mit dem ein Atari XE ausgestattet ist, während 1MB die Regel sind. Dank SIO2IDE, SIO2PC, SIO2SD und SIO2USB sind dem Speicherplatz auf Disk praktisch keine Grenzen mehr gesetzt, und dank Emulatoren kann sich die Entwicklungszeit für ein Programm drastisch reduzieren. Das Austattungsmerkmal das sich die meisten Leute für eine neue Textverarbeitung auf dem Atari 8-Bit wünschten war eine 80-Zeichen-Anzeige, welche ohne zusätzliche Hardware auskommt. Es brauchte nur wenige Wochen die 80-Zeichen-Anzeige zu programmieren. Dann fing ich an andere Dinge zu entdecken die ich verbessern konnte...

12.5 WARUM LW ENTSTANDEN IST

Der Atari 8-Bit ist eine beliebte Spielmaschine, es gibt aber auch eine umfangreiche Sammlung „seriöser" Anwendungsprogramme, einschließlich vieler Textverarbeitungen. Die beliebtesten kommerziellen Angebote in den 80gern waren *AtariWriter* (später *AtariWriter Plus*), *Paperclip*, *The First XLEnt Word Processor* und *Superscript*, obgleich es noch dutzende weniger „schwergewichtiger" Textverarbeitungen gab. Es wurden auch viele Public-Domain und Open Source Texteditoren und Textverarbeitungen für den Atari 8-Bit geschrieben wie *Speedscript* und *TextPro*. Letztere hat viele Überarbeitungen erhalten und ist eine der wenigen Textverarbeitungen die eine enge Verbindung mit SpartaDOS, dem fortschrittlichen DOS von ICD, bieten. *AtariWriter 80* und *TurboWord* nutzen 80 Zeichenanzeigen (die meisten Atari 8-Bit Textverarbeitungen waren auf die 40 Zeichen des Atari´s begrenzt), wobei diese Programme spezielle Hardware in Form der XEP-80 benötigen.

Ich bekam meinen ersten Atari 8-Bit - einen 65XE - in den späten Achtzigern, als ich noch zur Schule ging, und, als ich endlich ein Diskettenlaufwerk bekam, nutze ich *TextPro* zur

Textverarbeitung. Zur gleichen Zeit erlernte ich das Programmieren: zuerst nutze ich *Atari BASIC*, dann *Turbo BASIC XL*, anschließend *C* und letztendlich *Assembler*. Ich schrieb eine Datenbank, ein Zeichenprogramm, verschiedene *SpartaDOS*-Hilfsprogramme und einen Makro-Assembler, ich strebte aber immer an eine Textverarbeitung für den Atari 8-Bit zu schreiben welche die innovativen und flexiblen Ansätze von Public-Domain Programmen wie *TextPro* mit der Stabilität, Professionalität und handelstauglichen Qualität von *PaperClip* und *AtariWriter* vereint.

1989 begann ich grobe Ideen auf Papier zu skizzieren, aber es brauchte fast zehn Jahre bis ich so viel Erfahrung in Assembler hatte dass ich letztendlich mit dem Schreiben des Programms beginnen konnte. Sechs Monate später und nach hunderten Stunden Arbeit war Version 1.0 von *The Last Word* fertig gestellt. Zu dieser Zeit war das Programm lediglich 19K lang, belegte keinen Speicher unter dem OS (wodurch es kompatibel zu *SpartaDOS 3.2* und *DOS XE* war), und beinhaltete einige Innovationen wie eine scrollende Druckvorschau in einem Zehn-Zeilen-Fenster mit 80-Zeichen, eine Makro-Sprache, ein eingebautes Mini-DOS-Menü, die Möglichkeit auf Rechnern mit erweitertem Speicher mehrere Dokumente auf einmal zu laden und einen nicht-linearen Textpuffer, welcher die Trägheit, die viele andere Textverarbeitungen bei der Bearbeitung und Löschung von Text am Anfang einer grossen Datei zeigen, umgeht.

Das fertige Programm wurde Mitte 1999 an den „New Atari User“ (vormals Page 6) geschickt, aber zu dieser Zeit wurde das Magazin bestenfalls sporadisch veröffentlicht und ich erhielt nie eine Antwort von **Les Ellingham**, dem Herausgeber des Magazins. Ungefähr zur selben Zeit wurde die Herausgabe des „New Atari Users“ völlig eingestellt, und, ohne Zugang zum Internet, wurde mein Kontakt zur Atari-Gemeinde so vollständig abgebrochen. Dennoch nutzte ich - ein wenig unglaublich - meinen Atari als Haupt- (und einzigen) Computer zur Textverarbeitung. Da ich immer noch Spaß am Programmieren hatte verbesserte ich das Programm weiter bis zur Version 2.1, welche im Jahr 2001 fertig gestellt wurde. Ich glaubte dass das Programm alles das tat was es sollte und ich konnte mir nicht vorstellen dass es noch etwas hinzuzufügen gab.

Im selben Jahr kam ein IBM-kompatibler 286 in meinen Besitz, und *The Last Word* wurde etwas von *Windows 3.1* und *Word für Windows 2.0* überschattet. Selbst wenn ich immer noch der Zeit etwas hinterherhinkte, eröffnete mir der PC eine völlig neue Welt und ließ das Erstellen von Schreibprogrammen für den Atari wie eine ziemlich sinnlose Aufgabe erscheinen (besonders da ich keine Zielgruppe für meine Software sah). Innerhalb einiger Jahre, mit dem Zugang zum Internet und leistungsfähigeren Windows-Computern, begann ich mich auf den Bau und die Wartung von PCs zu konzentrieren. Der Atari wurde einer Box auf dem Schrank übergeben und ich habe viele Jahre nicht mehr an ihn gedacht.

Es war Ende 2008 nach der nostalgischen Unterhaltung mit einem Freund der, wie ich selbst, liebevoll an die Programmierung des 6502 zurückdachte, als ich mich entschloss den Atari wieder zu entdecken und herausfand dass es immer noch einige Foren im Internet gab die sich diesem alten Computer widmen. Ich war überrascht die blühende Online-Gemeinschaft des Atari zu entdecken und über die Vielzahl und Art der technologischen Entwicklungen die in der Zwischenzeit erdacht wurden.

Glücklicher Weise funktionierten sowohl mein 65XE (welcher vor ungefähr fünfzehn Jahren auf 130XE-Standard erweitert wurde) als auch das XF551 Diskettenlaufwerk noch einwandfrei; so waren sogar überraschend gut in Schuss, und kurz darauf legte ich mir ein SIO2SD zu, übertrug die meisten meiner 5¼” Disketten auf den PC und startete *The Last Word 2.1* zum ersten mal auf einem emulierten Atari. Kurze Zeit später richtete ich die Internetseite www.atari8.co.uk ein und veröffentlichte die Textverarbeitung zusammen mit dem Makroassembler und einigen anderen Hilfsprogrammen. Nachdem die Foren von www.atariage.com mein zweites zu Hause wurden erhielt ich rasch (meist positives) Feedback zu der Textverarbeitung, wodurch ich auf einen schwerwiegenden Fehler aufmerksam wurde: das Programm läuft nicht auf NTSC-Gerätent. Glücklicherweise war der

Fehler leicht behoben und LW baute seine Position unter den Anhängern von „seriösen“ Atari-Anwendungen aus.

Obgleich ich Version 2.1 von *LW* immer als „finale“ Version angesehen habe , kamen bald Benutzer mit neuen Ideen die sie gerne in dem Programm sehen würden. Ganz oben auf ddr Liste stand mit Abstand die Unterstützung einer 80-Zeichen-Anzeige (einschließlich Geräten wie XEP-80 und VBXE). Ich zögerte es auszuprobieren, nicht zuletzt weil ich keinen Zugang zu der nötigen Hardware habe und die Unterstützung von XEP-80 und VBXE im Emulator zur Zeit eingeschränkt ist. Die einzige Alternative war der Versuch die 80-Zeichen per Software zu realisieren, wobei ich zögerte weil ich nicht glaubte dass es schnell genug sein würde. Nach mehreren Diskussionen mit **Claus Bucholz** und einigen von seinem ACE-80-Displaytreiber entliehenen Ideen, habe ich etwas ausgeknobelt was mit ähnlich eindrucksvoller Geschwindigkeit arbeitete. Der Schlüssel zur Leistungsfähigkeit der 80-Zeichenanzeige ist die Art wie nur die Bereiche der Anzeige neugezeichnet werden in der Änderungen durch Editieren erfolgten. Beim Scrollen kommt auch eine dynamische Display-List zum Einsatz.

Die ursprüngliche Absicht waren zwei separate Versionen des Programms: eine welche mit 80 Zeichen arbeitet und eine andere welche mit 40 Zeichen arbeitet. Bald wurde es nicht nur schwierig die Entwicklung beider Versionen zu synchronisieren, ein einfacher Weg zwischen den zwei Auflösungen umzuschalten wurde offensichtlich. Bald danach hatte LW eine duale Anzeige und ist meines Wissens nach die einzige Atari 8-Bit-Textverarbeitung mit der Fähigkeit ohne zusätzliche Hardware jederzeit von 40 nach 80 Zeichenanzeige und zurück umzuschalten *ohne* dabei die aktuelle Position des Cursors im Dokument zu ändern.

Eine „Nebenwirkung“ der 80-Zeichen-Anzeige war der Anstieg des Speicherbedarfs was es erforderlich machte 14k des Programmcodes unter dem OS des Atari´s unterzubringen. Das bedeutete dass das Programm nicht länger zu *DOS XE* und Diskettenversionen von *SpartaDOS* kompatibel ist. Allerdings erlaubte es diese Änderung auch dem Programm zusätzliche Features hinzuzufügen, und im Laufe des ersten Halbjahres 2009 war der größte Teil umgearbeitet und die Unterstützung langer *SpartaDOS*-Dateinamen im Diskettenmenü mit Zeit und Datum sowie das Aufteilen langer Dateien auf Textbänke war hinzugefügt

Eine der wichtigsten Erweiterungen für das Programm ist die von **Paul Fisher** entworfene Zeichensatzsammlung. Diese große Sammlung speziell für LW entworfener 80- und 40-Zeichen-Zeichensatzpaare verleihen dem Programm, zusammen mit dem neuen Titelbild und Logo, einen professionellen Touch und heben die Qualität des Programms über alle Erwartungen hinaus. Paul’s Beitrag bei der Planung von *The Last Word 3.1* darf man nicht unterschätzen und ich hoffe dass ich bei zukünftigen Projekten auf sein beachtliches Talent und seine Kompetenz zurückgreifen kann.

Juni 2009 überführte ich, nachdem ich das Programm anfänglich mit meinem eigenen *XEDIT* Texteditor und *MA65* Makro-Assembler entwickelt hatte – zu Beginn auf echter Hardware und später mit einem Emulator auf dem PC - die Daten zum *ATASM Cross-Compiler* und der gerade veröffentlichten *WUDSN* - integrierte Entwicklungsumgebung. So vernarrt ich auch in die Benutzung meiner eigenen Atari-basierenden Entwicklungstools war, gab es doch keinen Zweifel daran dass die Entwicklung unter *WUDSN* meine Produktivität mehr als verdoppeln würde, und, Mitte Oktober 2009, war *The Last Word 3.0* (inzwischen beinahe 32K groß) bereit zur Veröffentlichung.

The Last Word hätte nie das Licht der Welt erblickt wenn ich nicht so viel Zuspruch und Hilfe von den Mitgliedern der Atari-Gemeinschaft in Europa und in den Vereinigten Staaten erhalten hätte. Spezieller Dank geht an **Paul Fisher** und **Konrad Kokoszkiewicz**, ohne die die Fertigstellung des Programms, so wie es heute ist, nicht möglich gewesen wäre. Mein Dank geht ebenfalls an alle **Betatester** die Zeit gefunden haben mich über Fehler zu unterrichten und im AtariAge-Forum Verbesserungsvorschläge machten; Dank an

Sebastian Bartkowicz und **Cabell Clarke** für ihre Hilfe bei *SpartaDOS*, und an **Claus Bucholz** für den Quellcode und Anregung durch Ace-80, **Marcin Prusisz** für SIO2SD und SIO2IDE, und an **Mark Grebe** für seine stetige Verbesserung des *Atari800MacX*-Emulators.

Wie geht´s mit *The Last Word* weiter? Mit der Einführung von Videoerweiterungen wie VBXE, größeren Speicher als ein Standard-8-Bit Atari, nahezu unbegrenzten Massenspeichern und leicht verfügbaren Flash-Module scheint die Zeit reif für eine echte WYSIWYG-Textverarbeitung auf dem Atari 8-Bit zu sein. Viele beachtliche Versuche mit grafischen Betriebssystemen (oder Anwendungen die ein GUI verwenden) wurden seit Mitte der 80ger Jahre auf dem Atari gemacht. Eines der ersten war **Alan Reeve´s** *Diamond GOS*, welches sehr nah an *GEM* auf dem Atari ST heran kam. Unglücklicher Weise gab es keine Unterstützung durch Anwendungen, was wohl bei vielen GUI Betriebsystemen der Grund des Scheiterns war. Letztendlich haben wir schon viele anspruchsvolle GUI gesehen, aber keine war *wirklich* zusammenhängend und gut zu gebrauchen wie das in den 80gern geschriebene *GEOS* für den Commodore 64. Meine Absicht ist *The Last Word* zum Mittelpunkt von Anwendungen für ein neues Atari-GUI zu machen welches es ermöglicht Vorteile aus den neuen Hardwareentwicklungen, die wir in den letzten Jahren gesehen haben, zu ziehen.

Ich hoffe in der Zwischenzeit verschiedene Zusätze für LW3.1 herauszugeben, wie z.B. eine Rechtschreibprüfung, eine Zeichentabelle, einen Taschenrechner und ein Programmierertoolkit. Es wird auch eine spezielle VBXE-Kompatible Ausgabe von LW 3.1 geben; sie bietet klarere 80 Zeichen-Fonts (spezielle 80-Zeichen-Zeichensätze werden nicht mehr nötig sein) und schnelleres arbeiten im 80-Zeichen Modus. Es gibt eines Tages auch die Möglichkeit einer Modul-Version des Programms, oder zumindest eine Modulversion der GUI unter der es eventuell laufen wird.

12.6 ENTWICKLUNG

Es gibt immer noch viele Funktionen die ich in zukünftige Versionen von LW implementieren möchte, zum Beispiel:

- Displaytreiber mit denen eine einzige Version von LW zur Benutzung des VBXE oder der XEP80 konfiguriert werden kann.
- Einen größeren nahtloser Textpuffer mit der Möglichkeit Dateien bis zu 32k zu laden.
- Drop-Down-Menüs und Dialogboxen.

Wenn ich *eine* Sache bei der Programmierung von *The Last Word* gelernt habe, dann diese dass kein Programm jemals *wirklich* beendet wird. Selbst wenn ein Programm frei von Fehlern ist gibt es immer noch Möglichkeiten es zu verbessern und den Wunsch eine weitere Funktion in den verbleibenden freien Platz zu implementieren. Die Erfahrung hat auch gezeigt dass ein immenses Unterfangen ist eine professionelle Anwendung zu schreiben die zuverlässig mit verschiedensten DOSsen und Speicherkonfigurationen zusammenarbeitet. Ich erschaudere wenn ich daran denke wie viele Arbeitsstunden ich in dieses Programm gesteckt habe. Ein Jahrzehnt nachdem das Programm ursprünglich geschrieben wurde tauchen immer noch Bugs im ursprünglichen Code auf. Ein großer Teil des letzen Jahres der Entwicklung wurde darauf verwendet die grundlegenden Funktionen des Programms narrensicher zu bekommen.

Glücklicherweise macht es eine Menge Spaß solch ein Programm zu erstellen. Das befriedigende Gefühl, als der Algorithmus, der die SpartaDOS X-Angaben zu Bytes pro Sektor / freie Sektoren in die freie Restkapazität der Diskette umwandelt, funktionierte war die vielen Stunden, die ganzen Breakpoints und das Single-Step-Tracing wert.

Der 8-Bit-Atari ist seit über 20 Jahren ein Teil meines Lebens - mal mehr, mal weniger, und ich hoffe daß ich in 20 Jahren immer noch mit meinem 130XE herumfrickle. Ich denke dass

der PC, auf dem ich diese Zeilen schreibe, dann nur noch eine weit zurückligende Erinnerung ist.

12.7 KONTAKT

Über Anfragen, Anregungen, Hinweise zu Fehlern im Programm freut sich Jonathant:

Jonathan Halliday / jon@atari8.co.uk

Über Anfragen, Anregungen, Hinweise zu Fehler in der deutsche Übersetzung freut sich Marc:

Marc Brings / last-word@arcor.de